UNIVERSITY OF HERTFORDSHIRE

School of Engineering and Technology

Science and Technology Research Institute

Smart Distributed Processing Technologies For Hedge Fund Management

Sinnathurai Thayalakumar

A thesis submitted in partial fulfilment of the requirements for the award of

Doctorate in Engineering

March 2017

Abstract

Distributed processing cluster design using commodity hardware and software has proven to be a technological breakthrough in the field of parallel and distributed computing. The research presented herein is the original investigation on distributed processing using hybrid processing clusters to improve the calculation efficiency of the compute-intensive applications. This has opened a new frontier in affordable supercomputing that can be utilised by businesses and industries at various levels. Distributed processing that uses commodity computer clusters has become extremely popular over recent years, particularly among university research groups and research organisations. The research work discussed herein addresses a bespoke-oriented design and implementation of highly specific and different types of distributed processing clusters with applied load balancing techniques that are well suited for particular business requirements. The research was performed in four phases, which are cohesively interconnected, to find a suitable solution using a new type of distributed processing approaches.

The first phase is an implementation of a bespoke-type distributed processing cluster using an existing network of workstations as a calculation cluster based on a loosely coupled distributed process system design that has improved calculation efficiency of certain legacy applications. This approach has demonstrated how to design an innovative, cost-effective, and efficient way to utilise a workstation cluster for distributed processing.

The second phase is to improve the calculation efficiency of the distributed processing system; a new type of load balancing system is designed to incorporate multiple processing devices. The load balancing system incorporates hardware, software and application related parameters to assigned calculation tasks to each processing devices accordingly. Three types of load balancing methods are tested, static, dynamic and hybrid, which each of them has their own advantages, and all three of them have further improved the calculation efficiency of the distributed processing system.

The third phase is to facilitate the company to improve the batch processing application calculation time, and two separate dedicated calculation clusters are built using small form factor (SFF) computers and PCs as separate peer-to-peer (P2P) network based calculation clusters. Multiple batch processing applications were tested on theses clusters, and the results have shown consistent calculation time improvement across all the applications tested. In addition, dedicated clusters are built using SFF computers with reduced power consumption, small cluster size, and comparatively low cost to suit particular business needs.

The fourth phase incorporates all the processing devices available in the company as a hybrid calculation cluster utilises various type of servers, workstations, and SFF computers to form a high-throughput distributed processing system that consolidates multiple calculations clusters. These clusters can be utilised as multiple mutually exclusive multiple clusters or combined as a single cluster depending on the applications used. The test results show considerable calculation time improvements by using consolidated calculation cluster in conjunction with rule-based load balancing techniques.

The main design concept of the system is based on the original design that uses first principle methods and utilises existing LAN and separate P2P network infrastructures, hardware, and software. Tests and investigations conducted show promising results where the company's legacy applications can be modified and implemented with different types of distributed processing clusters to achieve calculation and processing efficiency for various applications within the company. The test results have confirmed the expected calculation time improvements in controlled environments and show that it is feasible to design and develop a bespoke-type dedicated distributed processing cluster using existing hardware, software, and low-cost SFF computers. Furthermore, a combination of bespoke distributed processing system with appropriate load balancing algorithms has shown considerable calculation time improvements for various legacy and bespoke applications. Hence, the bespoke design is better suited to provide a solution for the calculation of time improvements for critical problems currently faced by the sponsoring company.

Acknowledgements

I would like to thank my supervisors, Dr Lily Meng and Mr Johann Siau, for their support, constructive comments, and guidance throughout this research. I would also like to thank internal examiner, Dr Georgious Passandis, for his comments on the submitted research and reports to ensure that this document complies with the academic standards set by the University.

I express my appreciation to my industrial supervisor, Mr David Rogers, for his support, comments on test results and outcomes of the research and their relevance to the company to confirm that the research conducted has a direct and positive impact on the company.

I would especially like to thank Professor Kevin Connolly for his guidance on designing complex financial models for various types of derivative products and his contributions to implementing highly sophisticated derivative pricing models for the sponsoring company, Northwest Investment Management (HK) Ltd.

A special thank is given to Northwest Investment Management (HK) Ltd for facilitating this research, and I would like to acknowledge and thank the company directors, Mr George Philips and Mr David Rogers, for approving the proposed research for the Professional Engineering Doctorate programme. I would also like to thank the Northwest team for their support.

I take this opportunity to express my gratitude to Dr Pandelis Kourtessis for approving the registration of the proposed research for the Professional Engineering Doctorate programme.

Finally, I would like to thank all of the staff at the doctoral college for their support in providing the required information and documents promptly to allow me to pursue my professional doctorate programme without any delays.

Table of Contents

List of Figures

List of Tables

Abbreviations

ASCOT	Asset-Swapped Convertible Option Transaction
CB	Convertible Bond
CESDIS	Centre of Excellence in Space Data and Information Sciences
COM	Component Object Model
COP	Cluster of PCs
DGC	Desktop Grid Computing
DIMM	Dual In-Line Memory Module
DLL	Dynamic Link Library
DPM	Dynamic Power Management
DR	Disaster Recovery
DSDM	Dynamic System Development Method
FDF	Financial Data Feed
FIFO	Fist In First Out
FLOPS	Floating Point Operations Per Second
GPU	Graphic Processing Unit
HFT	High-Frequency Trading
HPC	High-Performance Computing
IP	Internet Protocol
IR	Interest Rate
IV	Implied Volatility
LAN	Local Area Network
MIC	Many Integrated Core
MIMD	Multiple Instruction and Multiple Data
MISD	Multiple Instruction and Single Data
MPI	Message Passing Interface
NAS	Network Attached Storage
NASA	National Aeronautics and Space Administration
NIC	Network Interface Card
NoC	Network On Chip
NOW	Network of Workstations
Northwest	Northwest Investment Management (HK) Ltd
NUC	Next Unit of Computing

ODBC	Open Database Connectivity
P2P	Peer To Peer
PCI	Peripheral Component Interconnect
PVM	Parallel Virtual Machine
RAD	Rapid Application Development
RDBMS	Relational Database Management System
RISC	Reduce Instruction Set Computer
SAN	Storage Area Network
SATA	Serial Advance Technology Attachment
SC	Stock Connect
SFF	Small Form Factor
SIMD	Single Instruction and Multiple Data
SISD	Single Instruction and Single Data
SoC	System on Chip
SPM	Static Power Management
SQL	Structured Query Language
SSD	Solid State Drive
TCP	Transmission Control Protocol
UDF	User-Defined Function
VBA	Visual Basic for Application
WAN	Wide Area Network
WMI	Windows Management Instrumentation

Glossary

A-H Trade	Specialised trading strategy using shares
Beowulf	Commodity component based on loosely coupled processing cluster
Blade Server	Server chassis housing multiple thin form factor servers
CUDA	Parallel computing platform created by NVIDIA
Delta Hedge	Trading strategy for hedge portfolio
Dispersion Trade	Specialised trading strategy using options and shares
Dolphin	High-speed network interconnect implementation
HSI Index	Hong Kong stock market index
HT-Condor	Specialised task management system for compute jobs
InfiniBand	Computer communications standard used in high-performance computing
Mathematica	Technical computing and programming system
MATLAB	High-level technical computing language
MPICH1/2	MPI implementations
Myrinet	High-performance packet-communication and switching technology
OPEN-CL	Open Computing Language used for distributed processing
Option	Derivative financial instrument
PnL	Profit and Loss
Position	Number of shares that company holds
PxExec	Windows-based remote control utility program
QSNet	High-speed interconnect designed by Quadrics
Risk Analysis	Portfolio risk calculations using various methods
Security	Financial instrument such as Share, Option and Warrant
SETI	Search for Extra-terrestrial Intelligence
Simulink	Graphical programming environment by MathWorks
Stock	Financial instrument Share or Equity
Top500	Consortium that ranks the top 500 supercomputers in the world
Tracking Error	Deviation between portfolio and benchmark index

1 Introduction

1.1 Introduction

Distributed processing using commodity-type computers as processing nodes has been investigated for the last 20 years [1–4] and has been used in various industries for specific research and development [5, 6]. Most of these types of distributed processing systems use the Linux platform as the operating system, and a few systems use Windows-based applications [7, 8]. A specific type of distributed processing cluster design is based on commodity-type components and a loosely coupled configuration called the Beowulf-class cluster [9]. This type of cluster configuration method is used with certain required modifications and with new and specific methods to implement the distributed processing systems for the sponsoring company. Several advantages of using the Beowulf-class type distributed processing method are investigated in detail in this EngD research.

The research conducted in this EngD study is an investigation on distributed processing that utilises smart technologies for hedge fund management, and this thesis explains how this new method provides overall benefits to the sponsoring company and the research contributions made to the distributed processing technologies. The focus of the research is mainly to successfully implement distributed and parallel processing systems for compute-intensive calculations for the sponsoring company. The primary aim is to develop a collection of distributed processing systems that utilises Windows network topologies, networked workstations, servers, and dedicated distributed processing clusters formed by small form factor (SFF) computers to perform time- and data-critical calculations for various applications that are currently used in the company and for the future development of new trading strategies. The distributed processing systems have various components that must be incorporated efficiently to ensure the system performance is maintained at high levels [10]. These include system reliability, expandability, usability, application support, and so forth. Hence, each component of the distributed system has to be designed and implemented depending on the system requirements to ensure that the system operations are

adequate to support the specific applications used in the system. Several factors must be considered when designing a bespoke-type distributed processing system using first principle methods, and these are investigated in this EngD research.

In recent years, various systems and applications used by the company have become highly compute-intensive. One of the complex and compute-intensive systems that is currently used is the proprietary derivative Convertible Bond (CB) pricing model. Due to the complexity involved in the financial model calculations, particularly the derivative models, the calculation time becomes a critical issue [13, 14]. In addition, multiple scenarios need to be developed for the due diligence requirements by investors and compliance authorities [15, 16]. The financial models that are currently used to calculate various parameters for these products have been developed internally using first principle methods employing binominal and trinomial tree models, and these are highly compute-intensive [14, 17]. The major problems encountered in recent years with the systems used in the company are bespoke and legacy types of systems that require certain hardware and software configurations to work together. Hence, just replacing current hardware and software to improve the calculation speed is not a viable as a long-term solution.

Therefore, an alternative approach has been investigated in this research to improve the calculation efficiency by using existing hardware and software as calculation clusters and by utilising distributed processing methods [18, 19]. The approach is to consolidate all the existing hardware as multiple calculation clusters that are managed by a centralised distributed processing management controller. The research is conducted in four phases, which each phase investigates a particular research challenge, and all four phases are cohesively interconnected to find a best possible distributed process solution for improving the computing efficiency of bespoke and legacy applications. How these phases are implemented is discussed in detail in the forthcoming sections.

1.2 Motivation

As hedge fund's managers are facing increasingly rigorous transparency and reporting demands, the need for establishing a robust and high quality IT system has never been higher. In addition to the various regulations that are specifically targeted at hedge funds, the added complication of an increasing variety of asset classes needs to be managed by the IT systems. Over the past few years, the environment has been challenging for the hedge fund management industry with decreasing yields and more rigid regulation being introduced, which places added pressure on the business [20]. The key challenges faced by smaller hedge fund companies are follows:

- Complying with ever-tightening financial regulations that demand high standards of IT infrastructure and systems.
- Producing various reports for investors and regulators on a regular and on-demand basis.
- Generating a range of risk scenario reports for continuously changing investor requirements.

During the financial crash in 2009 [21], most small fund management companies disappeared. This was due to various factors, and one of the factors is that they were unable to function efficiently during disastrous market conditions where the required IT services were not provided by the IT support companies. Meanwhile, from inception, Northwest adapted its IT and systems as part of the business by employing dedicated IT staff and has actively promoted the use of internally developed systems for day-to-day business activities. As the result, the company can continue trading under difficult market conditions by rapidly adapting its IT systems to cater to the business, and this facilitated the company to relocate to different geographical locations without any interruption to its business. In addition, to keep IT expenditure low and still maintain a competitive edge, a novel approach of implementing IT systems is needed in the hedge-fund industry. Thus, the Northwest example has proven that the bespoke development methods used is highly beneficial to the company, and this is the primary reason for pursuing this present research to develop a bespoke distributed processing system for the company.

1.3 Research Aim

The overall aim of the presented research is to implement a distributed processing and parallel processing system for compute-intensive calculations utilising a combination of dedicated and non-dedicated workstations, servers and SFF computers. The task is to design, develop and implement cost-effective and high-performance distributed processing cluster systems that utilise Windows network topologies, networked workstations and servers to perform time-critical and data-critical calculations for the sponsoring company. The research objectives are as follows:

- Design and build a calculation cluster for distributed processing using user workstations as calculation nodes.
- Design and implement distributed process management SQL database to capture various data from calculation nodes, hardware and software related data, cluster performance data and historical data.
- Design and implement distributed process management system and calculation node's management system.
- Design and implement the adaptive load balancing systems that allocate tasks and data to each calculation node based on defined load balancing algorithms.
- Design and build dedicated SFF cluster and PC cluster using P2P network for testing various legacy applications and bespoke load balancing algorithms.
- Design and build multiple clusters using conventional servers, virtual servers and blade servers.
- Design and implement multiple clusters as consolidated hybrid cluster for performing heavy-duty batch processing tasks.
- Design and implement multiple logical clusters using single or multiple physical clusters for segregated and secure processing.
- Perform various tests to evaluate the calculation time improvements of distributed process based calculation against serial calculation method for different types of applications.
- Conduct a feasibility study of real-time trading strategies using distributed process based calculation methods.

Most of the current applications are MS-Excel-based, and these have been developed using the Rapid Application Development (RAD) method and the Dynamic System Redevelopment Method (DSDM) [22, 23]. The methodologies used for application development have proven to be considerably successful, and the same methods are used for distributed processing system development. The subject area of investigation is high-performance computing (HPC), and HPC is generally referred to as aggregating computing power in a highly organised way such that it delivers far better performance than a typical computer [24]. For HPC, several systems for distributed and parallel processing are available from various vendors and from third-party providers, and these systems utilise a combination of high-end servers, blade servers, virtualisation techniques, high-speed networks and Storage Area Network (SAN) technologies [25, 26]. The vendor-provided HPC systems have many advantages; HPC can be implemented in-house and can be configured to work with varying business requirements at lower cost than public grid computing in the long term. Even though these systems are easy to install and maintain in-house, there are initial costs of setting up the system, and the software and systems provided by the vendors and third-party providers are highly specific to their own systems. Grid computing is also part of HPC; it is a powerful concept that has proven to be successful in various industries [27]. However, to use grid computing, the applications must be compliant with the particular grid-computing provider, and there are costs related to processing time. Hence, the utility-based grid computing option is not feasible for the company's business model [28]. Meanwhile, it is possible to have a smaller-scale private calculation grid that operates with the company's local area network using peer-to-peer (P2P) network configurations [29].

Therefore, the main aim of the research is to design and implement a highly specific bespoke distributed processing system that can support the existing legacy and bespoke applications to improve the calculation efficiency. Moreover, the system must be complaint with existing design methodologies and technologies used. In addition, the system must improve the calculation efficiency of various batch-processing tasks currently used by the company. Furthermore, the system must be able to provide adequate processing power for the quantitative analysts and researchers in the company to test the new financial models within acceptable periods.

1.4 Research Methodology

This research consists of four phases of investigations, and all four of them are interconnected to find the best possible bespoke-type distributed processing solution for the company using existing hardware and software and utilising technologies that are currently used in the company. In addition, further investigations have been performed to identify the possible alternatives and improvements that can be made to the existing systems and applications to improve the calculation efficiencies. The four investigation phases are as follows:

1. Build a distributed processing cluster using existing workstations that are used by company staff and designing cluster management controller and calculation node's controller software design.
2. Implement adaptive load balancing and task scheduling techniques for the distributed processing clusters.
3. Design and build a dedicated calculation cluster using SFF computers and PCs.
4. Build a combination of various types of distributed processing as hybrid clusters using available processing units in the company and incorporating conventional server, virtual server and CPU clusters.

The first phase is implementing a distributed processing cluster using an existing network of workstations that are used by company staff as a calculation cluster. The design is based on a loosely coupled distributed process system design with a centrally managed control mechanism [10, 30]. The second phase is implementing adaptive task-scheduling [31] and load balancing [32] techniques to further improve the calculation cluster performance. In this phase, efficient task allocation and the load balancing mechanisms have been introduced, which have reduced the overall calculation times for nonlinear hardware and software-based calculation clusters. The third phase is implementing dedicated calculation clusters as private calculation grids in the company. The designs are based on using SFF computers and spare unused PCs that are currently available. These clusters are logically separated from the company's main LAN using P2P network setups for 24x7 operations for highly compute-intensive simulations. The fourth phase is implementing hybrid types of clusters that incorporate

all the available processing devices in the company to act as multiple clusters. Figure 1.1 shows the distributed processing high-level diagram. The distributed processing system has four phases, as described earlier, and must be able to support wide variety of bespoke and legacy systems. How these applications are used in the distributed processing system is discussed in detail in the forthcoming chapters.

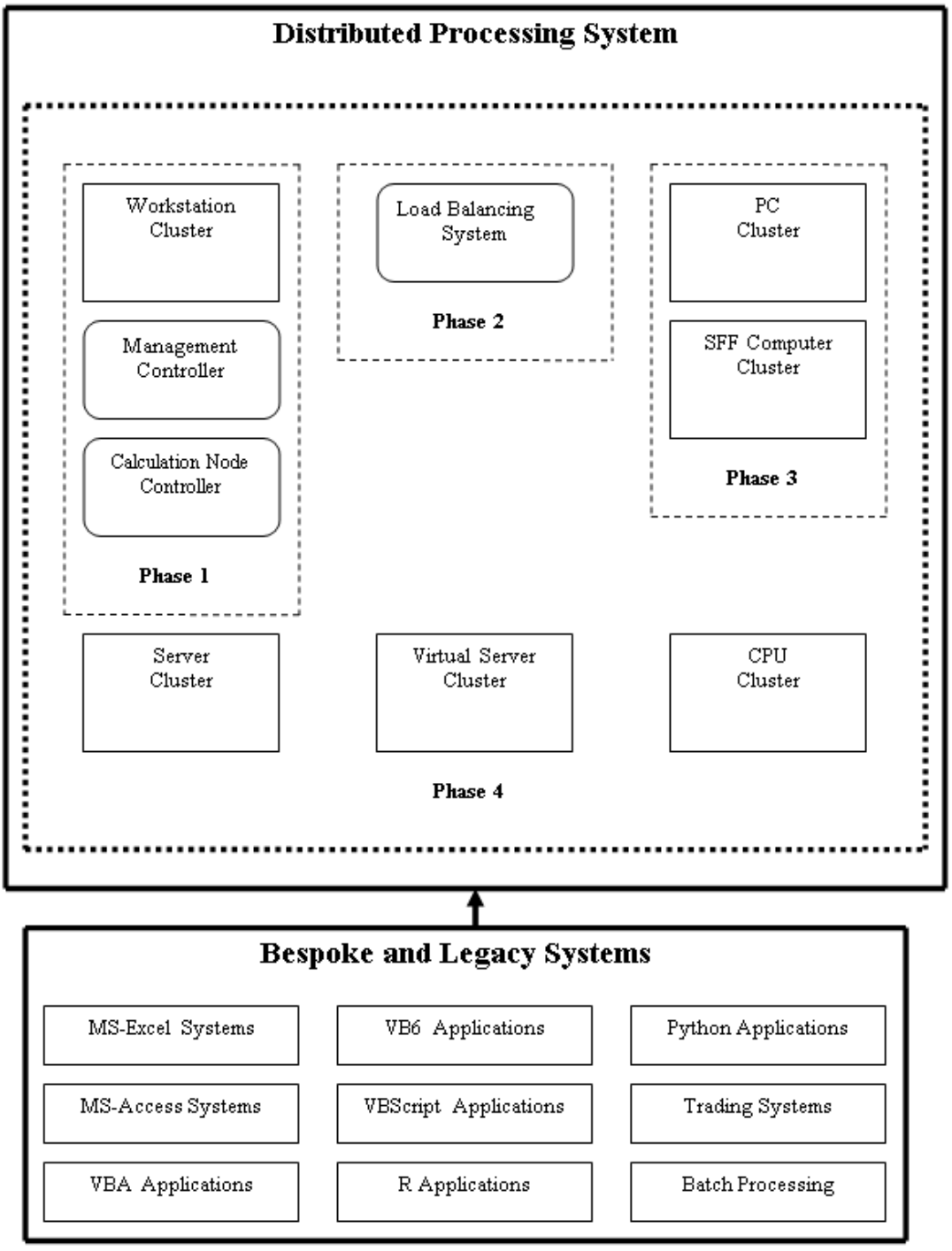


Figure 1.1: Distributed Processing System

1.5 Research Contributions

The investigations and tests performed have provided the company with new and significant insights about using bespoke-type calculation clusters for compute-intensive applications and about task scheduling and load balancing methodologies. Furthermore, detailed examination and development of new techniques are presented to apply within the current setups in the company that are most suited to provide effective solutions to the calculation-intensive application problems. The key research achievements and contributions of this research work are as follows:

- Implement in-house grid computing and private cloud type of systems for small companies that have limited resources with a cost-effective approach.
- Create an innovative approach to building a SFF Compute-Grid using inexpensive off-the-shelf PCs with flexible configurations.
- Design intelligent grid management software based on original concepts of adapting and self-tuning according to the computational requirements.
- Further knowledge in distributed processing technologies. In addition, investigations and tests conducted in a controlled environment facilitate the company improving its practical implementation of existing technologies.
- Achieved a detailed understanding of applicable techniques and methods for research and development and acquired a substantial body of knowledge in distributed processing technologies that is relevant in the forefront of academic discipline and professional practice.
- Conceptualised design, practice-based tests, and implementation have generated new knowledge in system implementations, application development applications and understanding of various distributed processing techniques that facilitates the research designs unforeseen problems.
- Facilitate the company for developing new methods and techniques using innovative approaches for implementing distributed processing using Windows networks. In addition, simplify cluster architecture to perform compute-intensive tasks to be split as process and data parallel tasks for legacy applications.

1.6 Thesis Organisation

Chapter 2 discusses the related research in distributed processing and correlates or draws distinctions in various areas of distributed processing applications investigated in this EngD research.

Chapter 3 outlines the background on distributed processing, theory, and governing laws related to distributed processing.

Chapter 4 describes the first phase performed to investigate the feasibility of distributed processing implemented in the current set-up in the company using a network of workstations, as well as the configuration required and the preliminary results. The design is based on a loosely coupled distributed process system design with a centrally managed control mechanism. A group of user workstations are selected to act as the calculation nodes, and these workstations are grouped to act as a calculation cluster. To manage the calculation cluster, two types of cluster management controllers are designed: distributed process management controller software, which resides in the management server; and calculation node controller software, which resides in each calculation node.

Chapter 5 describes the second phase, which applies various methods of load balancing and task allocation techniques that are used within the workstation cluster. In addition, the chapter discusses how these techniques are performed under different task allocation and load conditions. Implementation of efficient task allocation and the load balancing mechanisms have been introduced, which has reduced the overall calculation times for nonlinear hardware and software-based calculation clusters. The task allocation and load balancing algorithms are derived using hardware- and software-related parameters, and application-specific parameters. Hence, the distributed processing system uses dynamic and static rules depending on various parameters collected during the operations of the calculation cluster and using fixed parameters to allocate appropriate tasks and load to each calculation node to yield a better overall calculation time efficiency for a given batch process.

Chapter 6 describes the third phase, which is the designing and developing dedicated calculation clusters using SFF computers and PCs for the company as dedicated calculation gird, and the clusters tests, which is a detailed investigation performed using derivative technical analysis mathematical models. These clusters are logically separated from the company's main LAN and setup as a P2P network for 24x7 operations for highly compute-intensive simulations. These clusters are test clusters for investigating various parameters that affect the distributed processing implementations such as cost, power consumption, cluster size, and so forth. Two types of clusters are built, commodity-type SFF computer-based cluster with linear hardware and software configurations and PC cluster using spare PCs and workstations using nonlinear hardware and software configurations. Various tests are performed using compute-intensive applications to compare the performance matrix of these clusters.

Chapter 7 describes the fourth phase, which is a hybrid method for building distributed processing clusters using different types of hardware such as servers, workstations and SFF computers. These clusters are used to test the existing applications' performance under distributed processing conditions and the possibility of using multiple clusters to perform data- and time-critical calculations for real-time trading applications. In addition to tests for calculation improvements, various tests performed to reduce the data processing time using financial data feed terminals and linked SQL servers as part of the distributed processing system. Furthermore, tests are performed to investigate how multi-core CPU workstations and servers can be used as distributed processing devices by using a single CPU core as a calculation node. In addition, how the physical clusters can be configured as multiple logical clusters using CPU core-based cluster configurations is investigated.

Chapter 8 discusses the experimental results and analyses the data recorded during extensive tests and simulations performed on the PC and NUC clusters and during the application of load balancing techniques. In addition, the chapter discusses how this research benefits the sponsoring company and discusses the advantages and disadvantages of the methods used and further recommendations and improvements.

Chapter 9 concludes by presenting the findings, results and research outcomes.

1.7 Chapter Summary

The investigations and test results have shown that it is feasible to build a practical Beowulf-class distributed system using existing hardware and software within the company infrastructure. The design approach is based on loosely coupled design with dedicated or non-dedicated distributed processing system depending on the requirements that are best suited for the company. Further investigation using adaptive load balancing techniques and implementing a dedicated calculation cluster using SFF computers with low power consumption that utilises a logically separated P2P network has shown considerable improvement in calculation times for various internally developed applications used in the company. Hence, the bespoke high-throughput distributed processing cluster computer system that utilises the company's workstations, SFF computers, and blade servers in the office and disaster recovery site as intelligent processing devices are the best possible solution for the company's requirements.

Implementing workstation and server virtualisation techniques for logically separating a single workstation or server as multiple calculation nodes and using parallel processing in a single workstation using multi-core processing and multi-threading using logically separated CPU-cores show considerable calculation time improvement to certain types of applications. Furthermore, improvement in load balancing and task allocations based on application, hardware, CPU and memory usage-specific parameters using logical CPU cores and hyper-threading techniques is proved to be beneficial for application-specific load balancing, in addition to hardware and software-based load balancing. The test results show that SFF computer cluster is most suited for the long-term development of a dedicated calculation grid for the company due to its compact size, high reliability, lower power consumption and considerably low cost per calculation node. Meanwhile, the PC cluster and server cluster that were designed using spare PCs and servers are suitable for testing load balancing algorithms and testing prototype mathematical models for real-time trading algorithms. The implementation of internally developed bespoke-type distributed processing systems has a positive impact on hedge fund management technologies.

2 Related Research and Literature Review

2.1 Introduction

This chapter analyses and discusses the research related to distributed processing and describes how particular research areas are related to the research conducted. The focus of the subject area is cluster-based loosely coupled distributed processing [9, 19, 30], which is closely related to the research conducted. In addition, this chapter investigates the available distributed processing techniques and their usage in wider aspects of research and development. Further analysis is made regarding how these technologies can benefit the company's distributed processing systems. Additionally, this chapter discuss the background of various distributed processing technologies and how they have progressed, current improvements and developments that are happening and the future directions of these technologies and techniques [34]. Another area of interest is the miniaturisation technologies that drive the processing power higher and keep the cost low; in addition, it facilitates the efficient design of computing devices with low power consumption, less heat dissipation, and smaller size compared with conventional designs [35, 36]. In effect, these computing devices can be used as compute-nodes for small form factor distributed processing clusters [37, 38]. Various types of implementations are possible, such as network of workstations (NOW) [39] or cluster of PCs (COP) [40]. All these implementations are based on the loosely coupled cluster design paradigm called the Beowulf compute cluster [9]. Furthermore, in distributed computations, load balancing techniques and task allocations are employed to improve the process efficiency [41]. New methods are used for the design of the bespoke-type distributed processing systems for the company that are highly specific to the company's compute-intensive applications and that, in addition, comply with the requirements.

2.2 Distributed Processing Background

The research in distributed processing using cluster-based computing has progressed due to various practical limitations of using conventional and utility-type supercomputers [42, 43]. Some problems require a high level of computing power, and no complex algorithms can substitute. Some of examples of these types of problems are large particle-number physical simulations such as internal reactor simulations and geophysical simulations, meteorological, weather prediction, high-frequency trading simulations and fluid dynamics problems [44–46]. The computing power required to execute these types of programs are far beyond the capability of any workstation or server; hence, supercomputers seems to be the only solution. However, the use of supercomputers has many drawbacks, because supercomputers are extremely expensive, their maintenance cost is high, and they are provided by few companies with proprietary designs [47, 48]. The distributed process computing approach is focussed on utilising the idle resources in the network to aid processes that are running within that network and using commodity type low-cost computers to provide supercomputer-level processing power using computer clusters [49, 50].

2.3 Distributed Processing Cluster Development

The first attempt for distributed processing based cluster computing was developed in the 1960s by IBM as an alternative to linking large mainframes to provide a more cost-effective form of commercial parallelism [19]. In 1977, Attach Resources Computer Network (ARCNet) was developed by Datapoint using commodity components, and it was not commercially successful due to its complex design [51]. Clusters that are built using commodity off-the-shelf type hardware components as well as free or commonly used software have become a major part of high-performance computing and new concept of supercomputing [52, 53]. Figure 2.1 shows a typical cluster architecture using workstations as cluster nodes.

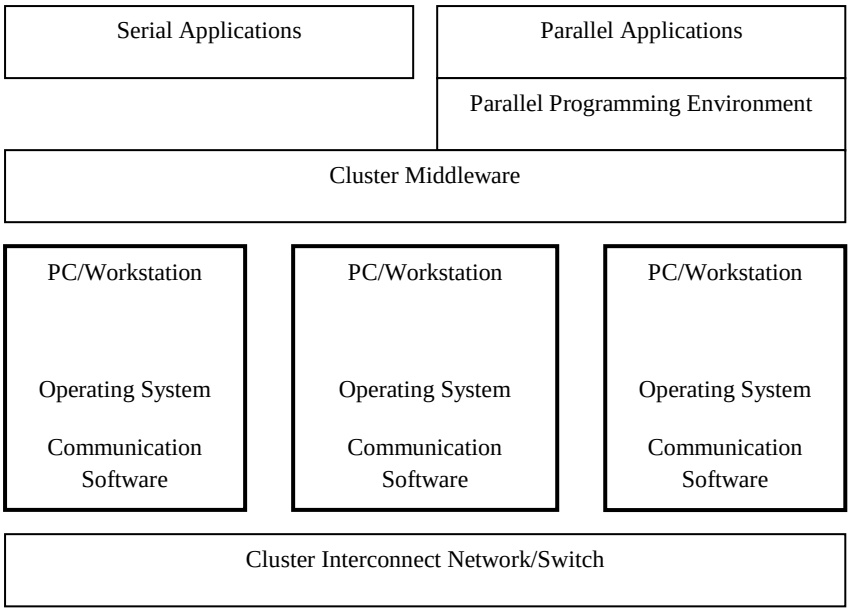


Figure 2.1: Typical cluster architecture

Distributed and parallel processing technologies have been in development for over 30 years [54] and started with smaller systems with few processing units for localised, specific implementations, and now they are components of the supercomputing architecture [55]. However, the processor power is continuously increasing by using innovative design architectures such as 3D design, composite materials and similar improvements rather than just increasing the clock speed [56, 57]. In recent years, distributed processing research and development has focussed on building powerful supercomputers using cluster architectures [58]. For high-speed interconnect, InfiniBand [59] and Myrinet [60] are used in certain forms to configure the networks as a high-speed bus that helps to make the system look like a single many-core computer. Another type of high-speed interconnect call the n-dimensional torus topology for high-speed, low-latency networks that are used in certain types of configurations [61]. The cluster-based supercomputing design provides petaflop computing power for comparatively lower cost [62] and is expected to be crossing the exaflop barrier [63] within a few years by using various types of improvements of the existing design, such as using improved inter-process communications and the incorporation of multiple and many integrated core processors.

2.4 Beowulf-Class Distributed Processing Cluster

The concept of Beowulf clusters originated at the Centre of Excellence in Space Data and Information Sciences (CESDIS), located at the NASA Goddard Space Flight Centre [9, 64]. The main goal of building this type of cluster was to create a cost-effective parallel computing system using mass-market commodity, off-the-shelf components to satisfy specific computational requirements in the Earth and space sciences community. The rapid and continuous advancement of microprocessors, high-speed network interconnects and related computer component technologies have facilitated many successful deployments of this type of cluster [65–67]. The Beowulf-class distributed processing clusters have successfully been implemented in various fields, such as medicine, theoretical physics, data storage, image processing and climate modelling [68–70]. The Beowulf-class clusters are fully dedicated to distributed processing, and they are optimised for this purpose; this gives a better price/performance ratio as a single unit of distributed processing computer [71]. These type of clusters are mainly based on Linux operating system, but a few Windows operating systems based cluster designs are available [72].

2.5 CPU-GPU Processing

Another processing method that is used in distributed processing is called GPU-accelerated computing [73], that is, use of a graphics processing unit (GPU) together with a CPU. A CPU consists of a few cores optimised for sequential serial processing, while a GPU consists of thousands of smaller, more efficient cores designed for handling multiple tasks simultaneously [74] and writing programs such as CUDA or Open-CL [75]. The recent adaptation of this type of processing method can be seen in supercomputers [76]. The GPU was originally designed to improve the graphic rendering for video games, and now this technology is integrated into the microprocessor chips such as Intel's Xeon Phi, which is a Many Integrated Core (MIC) architecture co-processor [77], to improve the overall performance of the integrated microprocessor design.

2.6 Load Balancing

Load balancing is a technique used to improve cluster performance by fully utilising computing devices with idle or under-utilised resources [78]. A number of distributed load balancing schemes for distributed processing clusters have been developed considering a variety of resources, including CPU, memory, disk I/O or a combination of CPU and memory [79]. Various methods have been investigated to improve the load balancing resource management, fault tolerance and scheduling techniques for different types of cluster configurations [80]. Fundamentally, two main types of load balancing methodologies exist: static load balancing [81] and dynamic load balancing [82]. In addition, these two methods can be combined to form a type of hybrid load balancing method [83] to achieve better performance. Certain systems such as HT-Condor [84] can integrate both dedicated resources such as rack-mounted clusters and non-dedicated desktop machines into one computing environment. One of the techniques used for task scheduling is called pre-emptive scheduling [85], and this technique is widely used in operating systems. Load balancing is discussed in detail in section 3.7 in Chapter 3.

2.7 Amdahl’s Law and Gustafson's Law

The effort to quantify the potential for performance increases by means of parallelisation draws from a series of study that traces its roots to the work of Gene Amdahl and later, John Gustafson. Amdahl’s law [11] is a formula used to find the maximum improvement possible by improving a particular part of a system. In parallel and distributed processing, Amdahl's law is mainly used to predict the theoretical maximum speed-up for program processing using multiple processors. Meanwhile, Gustafson's law [12] is a refinement of Amdahl's law, assuming fixed problem size; Gustafson's law states that massively parallel processing allows computations previously unfeasible, since they enable computations on very large data sets in a fixed amount of time. Hence, a parallel system does more than speeding up the execution of a code, and in addition, it enables dealing with larger problems. These two laws are discussed in detail in section 3.4 and 3.5 in Chapter 3.

2.8 On-Chip Distributed and Parallel Processing

The recent trends in the semiconductor industry moving towards microprocessors with large complex cores that emphasise high frequencies are being replaced by microprocessors composed of multiple simpler cores [86]. Due to various limitations on manufacturing cost-effective microprocessors, such as heat management, power consumption, and miniaturisation limits due to quantum tunnelling effects, it becomes no longer viable to produce microprocessors by employing conventional manufacturing techniques using silicon wafers [87]. One of the approaches that has been used to improve speed of a microprocessor is to use multiple cores rather than the high-speed single core, and this method has been successful in improving processing power for the last 15 years [88]. The GPU manufacturers have introduced the multi-core processing method to improve the processing speed by using many simple processing cores to work as a single processing unit to speed up the graphical processing for 3D games [89]. Recently, the GPU design architecture was incorporated with multi-core parallel processing chips such as the MIC architecture and the tiled Reduced Instruction Set Computing (RISC) processor architecture [90].

In 2007, Intel released an 80-core Terascale processor, and it was the first generally programmable microprocessor to break the teraflops barrier [91]. In 2009, Intel also has developed a 48-core IA32 processor called the Single-Chip Cloud Computer (SCC) because of the way it resembles cloud data centres [92]. The SCC consists of 24 tiles, two IA32 cores and a router per tile creating a mesh network with 256 GB/s of bandwidth. Various types of designs investigated for multiprocessor system-on-chip (SoC) platforms for designing multi-core CPUs with on-chip distributed processing capabilities [93]. To enable these platforms, specific requirements for scalable communication interconnects such as networks-on-chip (NoC) possess many features that are particularly useful for multi-core chips that are connected using high-speed interconnects within the chip architecture [94].

2.9 Interconnect Technology

Interconnect technology plays an important role in high-performance cluster computing, because it is designed to provide inter-process communication across various processing devices. The Gigabit Ethernet, which uses the industry-standard TCP/IP protocol, is currently the most popular choice for many standard cluster configurations [95]. Specialised cluster interconnects such as InfiniBand, Myricom Myrinet, Quadrics QsNet and Dolphin SCI provide mechanisms to pass messages in a highly efficient manner compared with Gigabit Ethernet, thereby providing lower CPU usage and lower latency than the Gigabit Ethernet interconnect [96]. Meanwhile, most of the classic supercomputer designs use proprietary interconnect technologies to achieve high performance [97]. The requirement for an industry-standard interconnect architecture that provides high-bandwidth and low-latency data transfers initiated the InfiniBand Trade Association, a consortium of hardware and software vendors, to develop InfiniBand in the late 1990s [98, 99]. Meanwhile, 100 Gbps Ethernet [100] is under development, and the next generation of PCI-e supports 100 Gbps; hence, it will be possible to have high-speed communications within the Ethernet network framework and not require specialist communication protocols.

2.10 Distributed Processing in Investment Banking Industry

Various development attempts have been made using distributed and parallel processing for solving complex financial models [101]. Certain companies are engaged in high-speed trading system developments for high-frequency trading (HFT), and these companies have integrated the latest technologies for distributed and parallel processing [102]. When electronic trading using high-frequency trading started in 1998, trade execution time was several seconds; in contrast, the systems can now execute trades in microseconds [103]. For different types of derivate financial model calculations, certain methods have been investigated using distributed processing such as CPU-GPU hybrid utilisation to solve the binomial tree model and parallel computing in finance [104, 105].

2.11 Chapter Summary

Distributed processing using commodity hardware and software is a technological breakthrough in the concept of parallel, distributed processing, and is the future of high-power computing. The important factors in the design of these types of high-level systems are a scalable, high-bandwidth, low-latency network and a low-overhead network interface. The important factor of this type of technology is the opportunity for large-scale computing within an everyday framework of interactive computing. For the last twenty years, distributed processing research and development has facilitated many businesses and universities to have an alternative way of implementing supercomputing for their own requirements at low costs using Beowulf-class computing clusters at various levels. This type of distributed processing has opened a new direction of research to improve the computing power for using loosely coupled, low-cost commodity components and multi-core and many-core single-chip-based parallel processing. Another factor driving the distributed and parallel design is the limitation of miniaturisation due to the current physical laws; hence, the chip manufacturing companies are moving towards distributed processing within a single physical chip to improve the processing performance. For loosely coupled distributed processing systems, the improvements on commodity components' technological contribution have a greater direct impact than the fabricated technology-based systems. The detail analysis in the distributed processing literature shows that the Beowulf type of distributed processing is highly suitable for a loosely coupled distributed processing system design.

The technical limitation of widely available distributed processing system design approach is that the software runs on these systems has to be designed to work with the distributed system concerned. Meanwhile, the research addresses a specific design methodology that supports legacy software applications that primarily designed for serial processing. Hence, the distributed processing system implemented in this research that utilise unique and new methods based on distributed processing fundamental theories. Chapter 3 discusses the distributed processing concept and theories in more detail.

3 Distributed Processing Concept and Theory

3.1 Introduction

This chapter presents an overview of the underlying theories, laws, and technical background related to distributed and parallel processing applications and their relationship with the research conducted. In addition to the academic and theoretical research in distributed and parallel processing, the discussion also addresses the industrial research and development in various technologies that are closely related to distributed and parallel processing. Distributed processing and parallel processing are often referred to interchangeably and are referenced as the umbrella term of *parallel computing*. However, there are distinguishable differences between distributed processing and parallel processing techniques. In some cases, a hybrid of these two is also called *distributed parallel processing systems*. The parallel processing model uses a shared memory within tightly coupled systems, such as a single board with multiple processes, and the distributed processing model uses distributed memory within loosely coupled systems such as PCs connected through a gigabit network. The main discussion in this chapter focuses on a loosely coupled system design with static and dynamic load balancing and process-termination detection mechanism that is highly suited for this EngD research. Hence, the rest of this chapter discusses distributed processing system designs and compute-cluster designs, message-passing interfaces and task allocation mechanisms.

3.2 Concurrency

Concurrency refers to sharing multiple resources in the same timeframe such as sharing CPU processing power using context switching, memory and I/O devices at same time. When the different processing units use separate memory in different computers, it is referred to as distributed processing [107].

3.2.1 Parallel Processing

Parallel processing is mainly considered as a process that shares the same memory resource for all the processes. Certain hardware-based parallel systems use a master clock across multiple processors, and these types of systems are fully deterministic. Parallel processing systems are generally considered as tightly coupled systems [107].

3.2.2 Distributed Processing

Distributed processing accelerates processing by distributing the workload to multiple computers that have been chosen to provide additional processing power. In distributed processing, each processing unit has its own private memory and information is exchanged by passing messages between the processing units [107].

3.2.3 Hybrid Processing

Hybrid processing often refers to systems that utilise the combination of distributed and parallel processing techniques to solve a particular problem in an efficient manner. Hence, this form of processing takes advantage of the best parts of both parallel and distributed processing system configurations for computations, and how they are implemented depends on the system requirements and applications used. Hybrid processing can consist of multiple physical processors and logical processors working in a coherent way to improve the overall computation performance of the system.

3.3 Characteristics of Distributed Processing Systems

Distributed processing systems have many differences; however, several common terms are used to define how the systems are supposed to work coherently to achieve the maximum benefits from the particular system concerned [107–109]. Table 3.1 shows the distributed processing attributes and their descriptions.

Table 3.1: Distributed processing attributes and descriptions

Attribute	Description
Node	A node is an entity on the network that can perform computing tasks.
Job	A job relates to the overall processing work that needs to be completed to solve a given problem.
Task	A task is a logically discrete entity of an overall job.
Synchronisation	Each processing node works on the given task to complete the overall job, and this process involves synchronisation between nodes and processes as well as maintaining the balance.
Race Condition	The race condition is the behaviour of systems where the output is dependent on the sequence of other uncontrollable events.
Deadlock	Deadlock is a condition that occurs when two processes are each waiting for the other to complete before proceeding, and the result is that both processes hang.
Bandwidth	Bandwidth refers to the data rate, that is, the amount of data that pass through a communication channel over a given period.
Latency	Latency refers to the time between an action being initiated and the action actually having some effect.
Speedup	In distributed computing, speedup describes how many times faster a job is running on multiple processors rather than a single processor.

3.4 Amdahl's Law

The Amdahl's Law [11] suggests that unless the system is 100% efficient, adding more calculation nodes to the cluster makes the system become less efficient. That is, increasing the parallelism of the computing environment by some number N can never increase performance by a factor of N for a given task. Equation (3.1) illustrates the Amdahl's Law.

$$S = \frac{1}{(1-p) + \frac{p}{N}} \tag{3.1}$$

where

S Speedup factor
p Percentage of distributed calculation
N Number of calculation nodes

3.5 Gustafson's Law

Gustafson [12] made certain modifications to Amdahl's model by adjusting some of its underlying assumptions to accurately reflect speedup of parallelising. As computation resources increased, the problem size also increased, and inherently, the serial portion became much smaller as a proportion of the overall problem. Gustafson modifies Amdahl's Law that the overall problem size should increase proportionally to the number of processors (N), while the size of the serial portion of the problem should remain constant as N increases. Equation (3.2) illustrates the Gustafson Law.

$$S = p + (1 - p)N \tag{3.2}$$

where

S	Speedup factor
p	Percentage of distributed calculation
N	Number of calculation nodes

3.6 Processing Cluster Building Concepts

There are three main types of clusters: compute clusters, high-availability clusters, and load-balancing clusters. For applications involving compute-intensive calculations, the compute clusters are used. Hence, the computation clusters are designed mainly for providing processing power for compute-intensive tasks. Table 3.2 describes the common attributes of a processing cluster.

Table 3.2: Processing cluster attributes and descriptions

Attribute	Description
Scalability	Scalability describes how the cluster can be expanded or reduced depending on the requirements by adding or removing each compute-node without affecting the communication process.
Reliability	Reliability of the cluster ensures that the system does not experience degradability or malfunction when unexpected events occur.
Cost/ Performance Ratio	Cost/Performance ratio is intended to ensure that performance is gained by utilising the distributed processing cluster. In addition, the effort is needed to distribute the tasks.
Topology	Topology describes how the distributed processing cluster is constructed using each compute-node and the network infrastructure used to connect all the nodes together.

3.7 Load Balancing

Load balancing is a term that describes the method used to distribute the workload evenly within a system that has multiple processing entities [79, 110, 111]. The term load balancing is an umbrella term that involves many different techniques for maintaining a certain balance condition within the system to perform operations in a uniform manner. The overall load balancing mechanism that is applied to the systems includes the following parts: perform task decomposition, manage task process (scheduling) and monitor and manages the resources and process status.

3.7.1 Static Load Balancing

In static load balancing, the assignment of tasks to each calculation node is completed before the calculation starts [112]. In addition, the estimated calculation times, resources and distributed processing-associated parameters are assumed to be known at the time of task allocation. Hence, a task is executed on the selected calculation node to which it is assigned, that is, in static load balancing methods, calculation nodes are non-pre-emptive. The main aim of static load balancing methods is to minimise the overall calculation time while minimising the communication.

3.7.2 Dynamic Load Balancing

Dynamic load balancing is based on the redistribution of tasks among the processing units during execution time, and this redistribution is performed by transferring tasks from the heavily loaded processing unit to the lightly loaded processing unit with the aim of improving the performance of the application [113]. The dynamic load balancing operations may be centralised in a single controller unit or distributed across all the processing units that participate in the load balancing process, and many combination of different policies can be utilised. In addition, it uses auxiliary calculation nodes to protect against the calculation node failures during execution.

3.8 Task Allocation and Granularity

The distributed process computation is performed by splitting up a larger task into smaller chunks of tasks that can be calculated simultaneously and independently. If the task cannot be separated in a required way to perform the calculations, then the distributed processing has no benefit, and serial processing is better suited for these types of tasks [114]. In distributed processing, granularity is a measure of the ratio of computation to communication time and the computation time. Table 3.3 shows task decomposition and granularity methods and their descriptions [108, 109, 115]. Table 3.4 shows task scheduling methods and their descriptions [116].

Table 3.3: Task decomposition and granularity methods

Attribute	Description
Domain Decomposition	In this type of partitioning, the data associated with a problem is decomposed. Each parallel task then works on a portion of the data. This is also called *data parallelism*, in which each processor performs the same task on different data subsets.
Functional Decomposition	In this type of partitioning, each computation is different, and each task then performs a portion of the overall work. Functional decomposition is better suited to problems that can be split into different tasks.
Fine-Grain Processing	Relatively small amounts of computational works are performed between communication and synchronisation events.
Coarse-Grain Processing	Relatively large amounts of computational work are performed between communication and synchronisation events.
Task Scheduling	Task scheduling mechanism assigns tasks to a calculation node according to different scheduling policies.
Context Switching	The state of the execution context of a process or thread is stored and restored so that the execution can be resumed from the same point at a later time. This allows one CPU to handle many processes or threads without conflicts.

Table 3.4: Task scheduling methods and descriptions

Method	Description
First In First Out (FIFO)	Schedules tasks based on FIFO method regardless of task size, and this is a basic type of scheduler implementation.
Shortest Job First	General scheduling principle can be applied to any system in which the estimated calculation time per task is determined before assigning the tasks to the processing units.
Pro-rata	Weighted average calculation times are used to allocate the number of tasks to each processing unit.
Pre-emptive	Task can stop/start in the middle of execution to give priority to another task by using a context switch mechanism, that is, stopping the running process and starting/resuming another.
Round Robin	Instead of executing the tasks to completion, the tasks are executed in a time-sliced manner.

3.9 Chapter Summary

The development of various types of distributed processing and parallel processing systems have produced several related theories and implementation methods in this field. However, some of the approaches were proposed specifically towards large-scale distributed processing systems and supercomputing-level systems. Hence, most of the specialised distributed processing concepts such as high-speed networks and message processing speeds are not applicable to the research conducted in this phase, which is of a highly specialised bespoke type distributed processing system that uses a small scale and loosely coupled cluster with multiple processing devices for private use for a company. Meanwhile, the fundamental principles of distributed and parallel processing remain the same; therefore, the system is designed based on original design and in conjunction with other known theories, methods and concepts that are particularly useful for bespoke-type highly specialised design. For loosely coupled distributed systems, each processing unit's performance is a critical factor for the overall performance of the system.

4 Distributed Processing Cluster Design Using Network of Workstations

4.1 Introduction

This chapter describes the implementation of distributed processing clusters using a network of workstations for MS-Excel applications used for calculating hedge funds' portfolio risk scenarios. This is the first phase of the investigation that explained in section 1.4 in Chapter 1. The primary aim of the investigation is to examine whether it is possible to design, develop, and implement a cost-effective and high-performance distributed processing cluster system that uses Windows network topologies and existing networked workstations to perform time-critical and data-critical calculations. Most of the applications used in the company, specifically the derivative pricing models, are based on MS-Excel and VBA, VBScript, and SQL Server databases and are developed internally as bespoke systems. Since the distributed processing system has to be capable of working with existing systems, the methodology used for the design and development of the distributed processing system is similar to the systems currently used for their compatibility and ease of use. The distributed processing system has a number of different modules, and all the modules are designed and developed to work coherently with each other. Additionally, they are designed in coordination with existing systems with minimum complexity to reduce the failure rate and maximise the efficiency to ensure system robustness. The testing and simulations have shown considerable improvement in calculation times using distributed processing against using a single server as a serial process for certain batch processing calculations. However, due to the nonlinear nature of risk calculation mathematics, complex issues relating to the number of calculation nodes used and the reduction in calculation time. These complexities can be eliminated or reduced by introducing efficient task scheduling and resource management algorithms. Distributed processing system cluster design, database development, cluster controller design, calculation node's controller designs, and test results are presented in this chapter. The task scheduling mechanisms and load balancing are discussed in detail in Chapter 5.

4.2 Northwest Systems

The applications and systems used within the company have been developed internally as bespoke systems for the last eighteen years. These systems are designed and developed to facilitate the processes in every part of the business area, and the systems have been continuously improved and modified as the business requirements change with the market conditions, compliance requirements, and management. Moreover, the systems will continue to be developed, modified, and be enhanced; hence, the distributed processing system has to be capable of adapting to these changes accordingly. Compute-intensive applications are used within the distributed processing system; therefore, these types of compute-intensive applications are modified in certain ways, hence that the applications can be employed by users in a standard way and are able to be used within distributed processing systems without major modifications to the overall systems. Figure 4.1 shows the hardware configurations, and Figure 4.2 shows software configurations. Table 4.1 lists the general overview of the technologies used for the company's applications and systems development, Table 4.2 lists the software implementations, and Table 4.3 lists a typical server and workstation configuration.

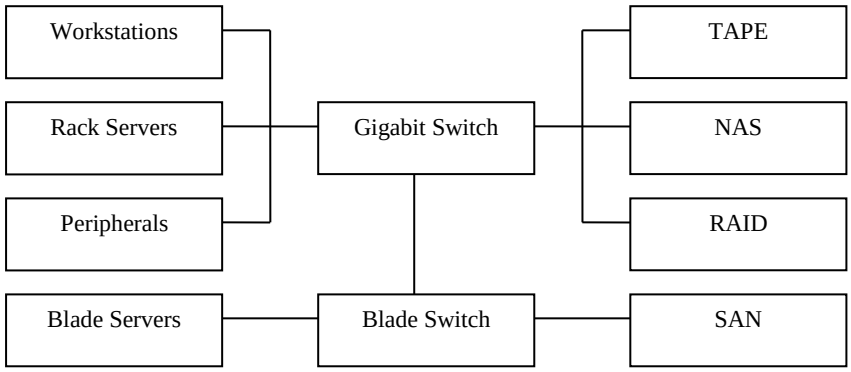


Figure 4.1: High-level diagram of hardware configuration

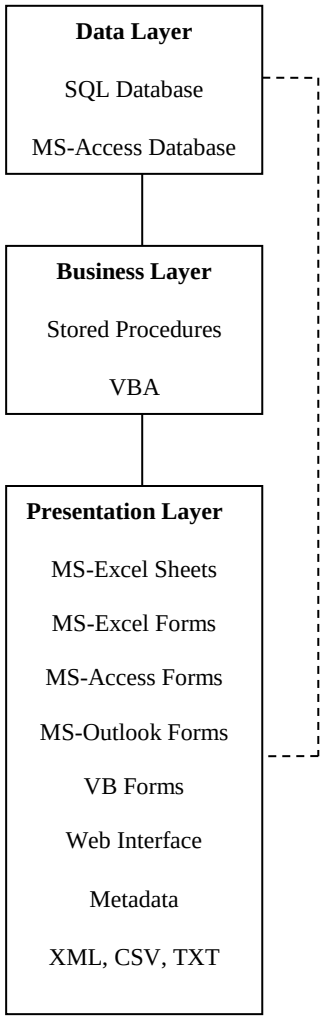


Figure 4.2: High-level diagram of software configuration

Table 4.1: Hardware and software technology used in the company

System	Type	Technology used
Operating system	Software	MS-Windows 7, MS-Windows Server 8 Enterprise System
Database	Software	MS-SQL Server 2005/2012, MS-Access 2010
User interface	Software	MS-Excel 2010, MS-Access 2010
Programming	Software	VB, VBScript, VBA, T-SQL
Workstations	Hardware	HP Z400, Z420, Z440, Z820
Servers	Hardware	BL460C Series G5/G6/G7
SAN	Hardware	Net-App Series

Table 4.2: Software systems used in the company and their implementation

Application	Implementation
MS-Excel-based applications	MS-Excel functions and MS-Excel VBA, charts and reports MS-Excel forms, Bloomberg data feeds. COM add-ins SQL data access components
MS-Access-based applications	Access functions and Access VBA, charts and reports, Access forms, COM add-ins SQL data access components
VB applications	EXE, DLL, COM add-ins, SQL data access System-level monitoring
Management systems	VB6,VBScript,WMI, Batch programming SQL data access, SQL jobs, Windows-based scheduling, Application monitoring

Table 4.3: Typical server and workstation configuration

System	Type	Description
CPU	Workstation	Pentium Xeon processor Single CPU with four cores (2.86 to 3.06 GHz per core)
Memory	Workstation	12 GB DIMM
Hard Disk	Workstation	Primary: 128 GB SSD Secondary: 500 GB SATA
Operating System	Workstation	Windows 7 (64)
Applications	Workstation	Microsoft Office Professional 2010/2013 Bespoke applications, Utility applications
CPU	Server	Intel Pentium Xeon processor Dual CPUs with four cores (2.90 GHz per core)
Memory	Server	64 GB DIMM
Hard Disk	Server	Raid: 3 x 500 GB SAS
Operating System	Server	Windows Server 2008 Enterprise
Applications	Server	Windows Server Utilities

4.2.1 Northwest CB Pricing Model

The company has developed number of mathematical models for calculating various parameters for different type of derivate instruments. These mathematical models are based on fundamental principle of option pricing mathematics, and in addition, certain modifications are made to the fundamental mathematics to improve the parameter accuracy and able to use wider range of derivative instruments such as Convertible Bond (CB) by introducing company specific modifications [13, 14]. The company uses three type of lattice (tree): Binomial-Tree, Trinomial-Tree and Multinomial-Tree. Each type has its own advantages and disadvantages and all of them require high level of computing power to calculate various parameters. However, the Binomial-Tree method takes less computing power than the other two methods. To illustrate how the lattice mathematical model works, Binomial-Tree method is used. Figure 4.3 shows the general Binomial-Tree method used for derivative pricing. Equations (4.1) to (4.4) are used for the derivative's parameter calculations.

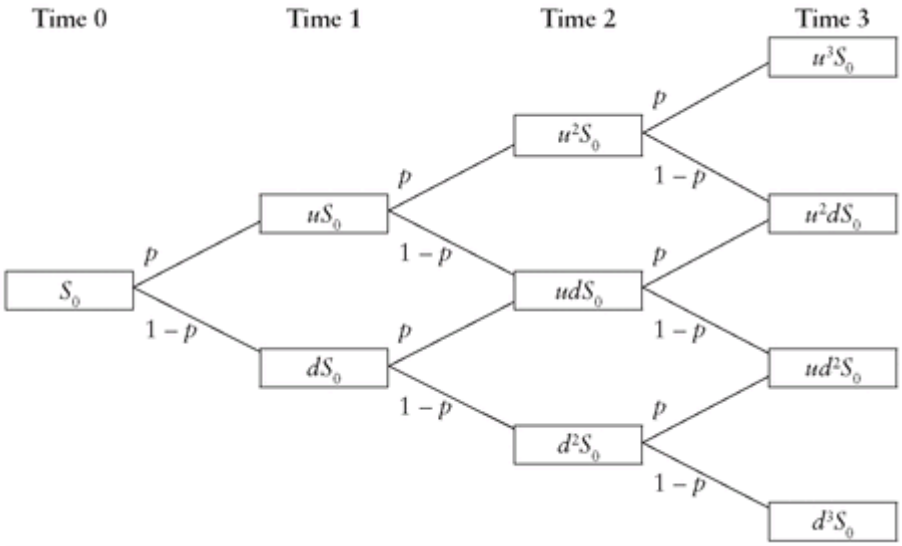


Figure 4.3: General Binomial-Tree Method

$$p = \frac{e^{(RT/N)} - d}{u - d} \tag{4.1}$$

$$u = e^{\mu} \tag{4.2}$$

$$\mu = v^{(T/N)^{0.5}} \tag{4.3}$$

$$d = e^{-\mu} \tag{4.4}$$

where

p Probability number for share price goes up

T Number years to maturity in years

N Number of steps (Time-period)

R Fixed interest rate

v Share volatility

S_0 Initial share price

For pricing different type of CBs, number of modifications and enhancements are made to the basic lattice model. These modifications are highly specific to the company and designed as proprietary bespoke mathematical models. The binomial pricing model traces the evolution of the convertible bond's key underlying variables in discrete-time. This is done by means of a binomial lattice for a number of time steps between the valuation and maturity dates. Each tree-node in the lattice represents a possible price of the underlying at a given point in time. Valuation is performed iteratively, starting at each of the final nodes those that may be reached at the time of maturity, and then working backwards through the tree towards the first node, that is, valuation date. The value computed at each stage is the value of the CB at that point in time. Figure 4.4 shows the modified Binomial-Tree method for calculation of CB parameters. Equations (4.5) to (4.9) are used for the CB's parameter calculations.

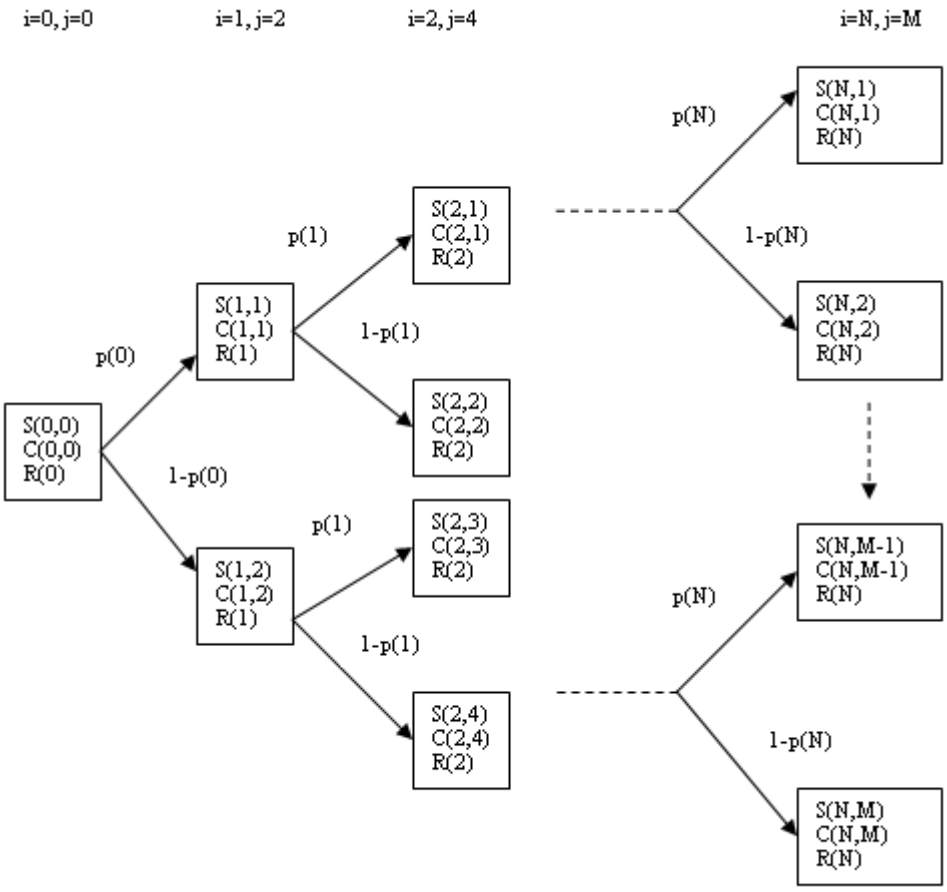


Figure 4.4: Modified Binomial-Tree Method

CB price calculation is performed using the following steps:

1. Forward interest rate calculation for each tree-node
2. Up and down share price probability calculation for each tree-node
3. CB's underlying share price calculation for each tree-node
4. Calculation of CB price at each final node
5. Sequential calculation of the CB price at each preceding node

During the calculation process, there are number of data points are captured at various stages to calculate *Delta*, *Gamma*, *Vega* and so forth. The whole calculation process for each CB is highly compute-intensive, and each portfolio can have many hundreds of CBs. Hence, to perform portfolio level calculation, using a single server will take few hours to complete. Furthermore, if input parameters need to be changed to view what-if scenarios for various portfolios calculation, then the calculation time will be increased to many hours.

$$p(i) = f(i, Y, N, M, v) \tag{4.5}$$

$$R(i) = f(i, Y, N, M) \tag{4.6}$$

$$S(i) = f(i, S(0), p(i), N, M, E, v) \tag{4.7}$$

$$C'(i) = f(i, R(i), S(i), p(i), N, M, \{X\}) \tag{4.8}$$

$$C(i) = f(i, C'(i), C'(i+1), R(i), S(i), N, M, \{X\}) \tag{4.9}$$

where

p(i) Probability number for share price goes up at tree-node i

R(i) Forward interest rate at tree-node *(i, j)*

S(i, j) Selected share price at tree-node *(i, j)* based on defined rules

C(i, j) Selected CB price at tree-node *(i, j)* based on defined rules

S(0,0) Initial share price at tree-node 0 (current price)

N Number of steps

M Maturity date of the CB

Y Time series yield rates for CB currency

E Redemption price of the CB

v Share's 90 day volatility

{X} is a data set that depends on the following input parameters:

Parity	Volatility	Redemption price
Credit spread	Bond currency	Stock loan fee
Dividend rate	Dividend frequency	Next dividend date
Coupon rates	Coupon frequency	Next coupon date
Call dates	Call triggers	Put dates
Put triggers	Put prices	Soft put dates
Soft put triggers	Soft put prices	Re-fix dates
Re-fix triggers	Re-fix up	Recall spread
Denomination	Re-fix down	Fixed recall rate
Conversion price	Conversion ratio	Day count
Bond currency yield rate	Short currency yield rate	FX rate

For binomial tree calculation method, the following program structure gives the number of calculation nodes needed to complete the whole calculation for a CB.

```
For j = 0 to N
        z=z+1
Next j
For i = N to 0 Step -1
            For j = 1 to i
                z=z+1
            Next j
Next i
```

where

N Number of steps and depends on each CB's maturity date.

z Number of calculations nodes needed to complete the whole calculation.

The value of *N* is calculated as the number of days between maturity date and calculation date as shown in Equation (4.10).

$$N = INT(M - T) \tag{4.10}$$

where

M Maturity date

T Calculation date

INT Integer number round-down to floor

Thus, at each tree-node, number of calculations is performed and number of tree-node per CB is depends on CB's maturity date. Furthermore, each portfolio can have many hundreds of CBs with different maturity dates; therefore, shorter maturity date CBs takes less calculation time compared to longer maturity date CBs. In addition, the portfolio is continuously modified to keep as fully hedged portfolio, that is, in regular basis, portfolio constituents will change according to set of company specific rules. This will cause the number of CBs in the portfolio to change and in effect, total calculation time of the portfolio. The can cause calculation time discrepancies and further delays. The value of *z* is exponentially increases with furthering maturity date as shown in Table 4.4.

Table 4.4: CB's Maturity Date vs Number of Tree-Nodes

Maturity Years	Number of Tree-nodes (z)
1	67,161
2	267,546
3	601,156
4	1,067,991
5	1,668,051
6	2,401,336
7	3,267,846
8	4,267,581
9	5,400,541
10	6,666,726

The research conducted herein, addresses various methods of overcoming calculation time inefficiencies by implementing highly specific distributed processing systems with appropriate load balancing techniques using tree-node parameters. These methods are discussed in detail in the forthcoming chapters. For performing various scenario analysis tests, Equations (4.5) to (4.9) reformulated with certain key parameters as variables and the rest of the parameters kept as static for each CB. Equation (4.11) shows the simplified version of CB price calculation formula that used for scenario analysis tests in the forthcoming chapters.

$$C = f(N,T,s,r,v,c,\{X\}) \tag{4.11}$$

where

C CB price

N Number of steps

T Time to maturity in years

s Parity

r Interest rate

v Volatility

c Credit spread

$\{X\}$ Static parameter set

C=Theoretical value (fair value) when v= 90 day volatility

C= Market price when v= Implied volatility

During the scenario analysis calculations, number of parameters is calculated. To illustrate the calculation of one of the parameter called DV01, which is, CB price impact of 0.01% increase in interest rate of long-term government bond. Equations (4.12) to (4.14) show the how the parameter DV01 is calculated.

$$DV01 = \sum_{j=1}^{m} [C' - C] \times O(j) \times K(j) \times F(j)^{-1} \tag{4.12}$$

$$C = f(N, T, s, r, v, c, \{X\}) \tag{4.13}$$

$$C' = f(N, T, s, (r + 0.0001), v, c, \{X\}) \tag{4.14}$$

where

O(j) Open position of CB j

K(j) Price multiplier of CB j

F(j) FX rate of CB j

m Number of CBs in the portfolio

{X} Static parameters

The CB model described in this section is used in various supplications tested in forth coming chapters; however, the input and output data varies according to the application used. The following is a typical example of input and output parameters for a single CB that used for scenario calculations. In a single portfolio can have many hundreds of CBs with different input and output parameters, hence, the portfolio level calculation has to go through many hundred times to complete all the required parameters. This process is highly compute-intensive task, requires many hours of computing time in a single server and in addition, depends on number of CBs, maturity date and so forth. In similar way, other parameters such as implied volatility, credit spread and so forth are shifted accordingly to calculate different scenarios for selected CB portfolio and how these parameter shifts are performed is discussed in detail in forthcoming chapters.

Table 4.5 shows typical input parameter dataset for a CB.

Table 4.5: Sample Input Parameters

[X]	Input Parameter	Value
X(0)	Ensemble ID	22347
X(1)	Fund Code	NNW
X(2)	Book Code	NNW-MV6
X(3)	Ensemble Name	CRRC 2021
X(4)	Security Type	CB
X(5)	Ticker	JV8124077 Corp
X(6)	Security ID	17249
X(7)	Security CCY	USD
X(8)	Maturity Date	05/02/2021
X(9)	Red Price	100
X(10)	Conversion Price	1.21948
X(11)	Conversion Ratio	3.2632
X(12)	PutDate1	05/02/2018
X(13)	PutDate2	05/02/2019
X(14)	PutDate3	05/02/2020
X(15)	PutPrice1	100
X(16)	PutPrice2	100
X(17)	PutPrice3	100
X(18)	CallDate1	05/02/2020
X(19)	CallPrice1	100
X(20)	CallTrigger1	130
X(21)	Re-fixDate1	05/02/2019
X(22)	Re-fixDown1	100
X(23)	Coupon Frequency	1
X(24)	Coupon	0.04
X(25)	Per or Currency	1
X(26)	Denomination	250,000
X(27)	CDS Price	0.014
X(28)	Stock Loan Fee	0.008
X(29)	Dividend Pass Thru	Y
X(30)	Dividend Amount	0.03
X(31)	Recall Spread	120
X(32)	Put Call	P

[X]	Input Parameter	Value
X(33)	Strike	300
X(34)	Total Position	10,500,000
X(35)	Security Price	105.0916667
X(36)	FX Rate	1
X(37)	Price Multiplier	0.01
X(38)	Cash Exposure	-2,898,864
X(39)	MV Short	-3,013,577
X(40)	Short Ticker	1766 HK Equity
X(41)	Short Price	7.47
X(42)	Short FX	7.7656
X(43)	CB Price	105.0916667
X(44)	Stub Price	1.07
X(45)	Stub Maturity	44232
X(46)	Security Name	CRRC 2021
X(47)	NAV	564,990,000
X(48)	Long Market Value	11,034,625
X(49)	Stub Market Value	0
X(50)	Short Market Value	0
X(51)	Long Rate	0.019097
X(52)	Short Rate	0.01109
X(53)	Credit Spread	0.014
X(54)	SLF	0.015
X(55)	Parity	78.8806694
X(56)	90 Day Volatility	21.93493
X(57)	Dividend Yield	0.023791893
X(58)	BDVD_NEXT_EST_EX_DT	05/02/2018
X(59)	EQY_DVD_FREQ	1
X(60)	DIVIDEND_YIELD	2.379189295
X(61)	CDS Price	0
X(62)	CDS Price 2	0
X(63)	CDS Price 3	0
X(64)	CDS Price 4	0
X(65)	AH Ticker	NNWJV8124077 Corp

Table 4.6 shows typical output parameter dataset for a CB.

Table 4.6: Sample Output Parameters

[Y]	Output Parameter	Value
Y(0)	Implied Volatility	0.3415379
Y(1)	Implied Volatility2	0.3415379
Y(2)	BF	94.369855
Y(3)	BF CS	94.352438
Y(4)	BF DV	94.352438
Y(5)	Theo Value	105.09167
Y(6)	Delta	105.46738
Y(7)	Vega	104.54727
Y(8)	CS	105.07709
Y(9)	CS2	105.09167
Y(10)	DV	105.07556
Y(11)	S+V	105.46738
Y(12)	Px Delta	0.375713
Y(13)	Px Vega	-0.5443987
Y(14)	Px CS	-0.0145766
Y(15)	Px CS2	0.00
Y(16)	Px DV	-0.016106

[Y]	Output Parameter	Value
Y(17)	Px S+V	0.37571298
Y(18)	$ Delta	28,988.64
Y(19)	$ Vega	-57,161.86
Y(20)	$ CS	-1,530.54
Y(21)	$ CS2	0.00
Y(22)	$ DV	-1.691.13
Y(23)	$ S+V	39,449.86
Y(24)	Delta bps	0.513082399
Y(25)	Vega bps	-1.011732397
Y(26)	CS bps	-0.027089727
Y(27)	CS2 bps	0.00
Y(28)	DV bps	-0.029932083
Y(29)	S+V bps	0.698240021
Y(30)	$ CDS_CS01	0.00
Y(31)	Theta	104.9430816
Y(32)	Px Theta	-0.148585021
Y(33)	$ Theta	-15,601.42

Hence, the number of input data and output datasets varies for different type of calculation performed. However, for testing purpose, number of input and output dataset is fixed and only the values are changed for each CB. To illustrate the input and output datasets for a selected CB, Equation (4.15) shows the input dataset and Equation (4.16) shows output dataset respectively.

$$[X(i)] = \{x(i,1), x(i,2), ...x(i,n)\} \tag{4.15}$$

$$[Y(i)] = \{y(i,1), y(i,2), ...y(i,m)\} \tag{4.16}$$

where

n Number of input data

m Number of output data

[X(i)] Input dataset for CB *i*

[Y(i)] Output dataset for CB *i*

4.3 Bespoke Software Development

Northwest is a small hedge fund company that is highly specialised in hedge fund management using complex derivative products as part of its portfolio management strategies. Due to the nature of the business, no single off-the-shelf product available in the market currently, that fulfills the overall requirements of the business' needs. Hence, to implement third-party software or off-the-shelf type applications require a number of applications and systems from various vendors to work as coherent business solutions. This will cause serious complexities and require a considerable number of technical staff internally and externally to support the systems, and in effect, it will become a complex and costly system that does not deliver what the company needed. The company has a policy for developing all the software that is used for day-to-day business-critical operations that specifies that this software should be internally developed as bespoke applications that are highly suited for the business' needs. Hence, the business applications and systems are designed and developed internally to facilitate the business in every critical business area, and the applications have been continuously improved and modified as the business requirements change. The company has the following requirements for system development:

- All the systems that are used for critical business-related activities must be designed and built in house.
- Use existing well-tested technologies to develop the systems.
- No new technologies are to be used that cost extra time and money.
- For critical business applications, third-party software should not be used.
- Any new system must work with the existing system without interrupting the current critical systems.
- The system has to be designed and built with minimum complexity. It must be easy to troubleshoot and fix any problem within the shortest possible period without affecting business.
- No black-box-type system developments or utilisations are permitted, and the systems must be transparent and auditable within Northwest's business.

Advantages of using bespoke systems:

- Designed specifically for particular requirements and it can be tailored to fit in exactly with the business requirements.
- Various applications and software can be integrated with customised design.
- For users, it is more intuitive to use and easy to operate in the way that they are used to working.
- Can be modified and changed over time as the requirements and business change.
- Users can directly interact and communicate with the developers for improvements and modifications.
- The use of professionally developed bespoke software applications can provide a significant business advantage.

4.4 Implemented Solutions for Company's Requirements

Due to the nature of the company's business, the distributed system that is proposed has to comply with existing development structures. Hence, the system has to work with the existing systems without interrupting the day-to-day business, and in addition, it has to prove that it can considerably improve the calculation time of various systems that are used within the company. To solve compute-intensive problems, a solution must be implemented that uses existing technologies in alternative and innovative ways to design and develop a system based on original concept that facilitates the business within the current development model rather than using new technologies, third-party systems, or similar. The solution has to be best suited for the company's business requirements, and it can be used fully or partly with similar types of business environments. The company has a Disaster Recovery (DR) site situated in a different location from the main office and connected through high-speed network. In case the company decided to expand the distributed processing capabilities in the future, the DR site can be utilised for constructing a small cluster farm with minimum cost. The following steps are used for the development:

- Develop all the required components for the distributed processing using existing technologies, software, and hardware.
- Transparent development and use only compliant code for system controllers.
- Applications must have manual overrides for monitoring and troubleshooting.
- Result-oriented based development process.
- Implemented solution must demonstrate considerable improvement compared with the current systems.

4.5 Application Development Strategy

Distributed processing systems and applications are developed utilising the Rapid Application Development (RAD) method and the Dynamic Systems Development (DSDM) method. Due to the nature of the business structure and operations are carried out within the company, all of the applications used within the company have to be highly flexible and adaptable to the changing business environment while maintaining robustness. For the research purpose, the applications used in the distributed processing system are modified with appropriate changes as prototype applications with core functionality that is required for testing and investigation under various conditions, methods, and algorithms. Hence, these applications are not engaged by the users in a live environment while under development, and once the testing is completed, then the applications will be further developed as usable robust applications with appropriate user interfaces, business logic, and validation rules for the company's quantitative research and development team. Figure 4.5 shows the application development process using RAD and DSDM.

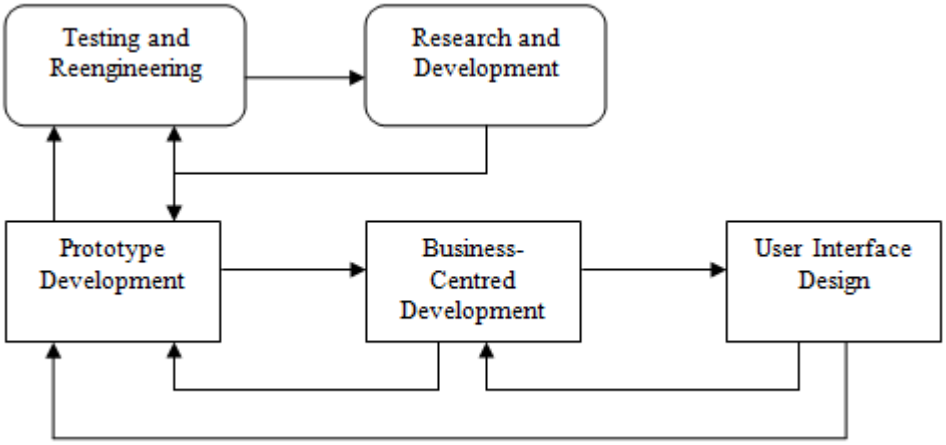


Figure 4.5: Distributed processing application development process

4.6 Application Structure

Any application that is used within the distributed processing cluster environment has to have a certain process flow structure to be executed in the selected calculation node. Most of the applications that are used by the company can be modified to work within the calculation cluster by adding extra optional code to comply with the logical data flow shown in Figure 4.5. Applications can be VBA-based or executable such as VBScripts, batch commands, or any executable applications. The calculate and execute process has a sequence of processes depending on the type of application, every action within the application is logged in the event log table, and the data collected is used for monitoring the cluster performance and fine-tuning the cluster to perform efficiently using adaptive algorithms. Figure 4.6 shows the distributed processing application process structure.

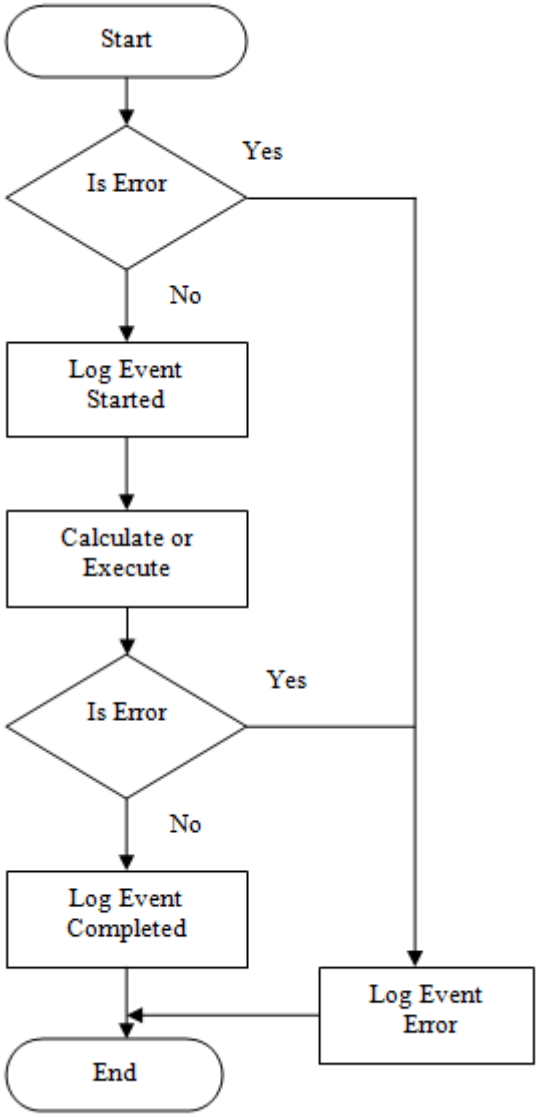


Figure 4.6: Distributed processing application execution process

4.7 MS-Excel Application Implementation

Excel is the main application platform for most of the applications developed by the company; hence, every user's workstation has MS-Excel installed as part the Microsoft Office professional suite. Some MS-Excel applications are simple, and some of them are highly complex with external data sources, SQL data feeds, and COM add-ins. Therefore, these applications need to be modified accordingly to work with the distributed processing system. For batch processing applications, the data structure is relatively simple to modify, and for systems that are employed by users on a daily basis, various methods of modification needed, hence, that the system can be used in user mode and in distributed processing mode. Figure 4.7 shows a typical Northwest MS-Excel application configuration.

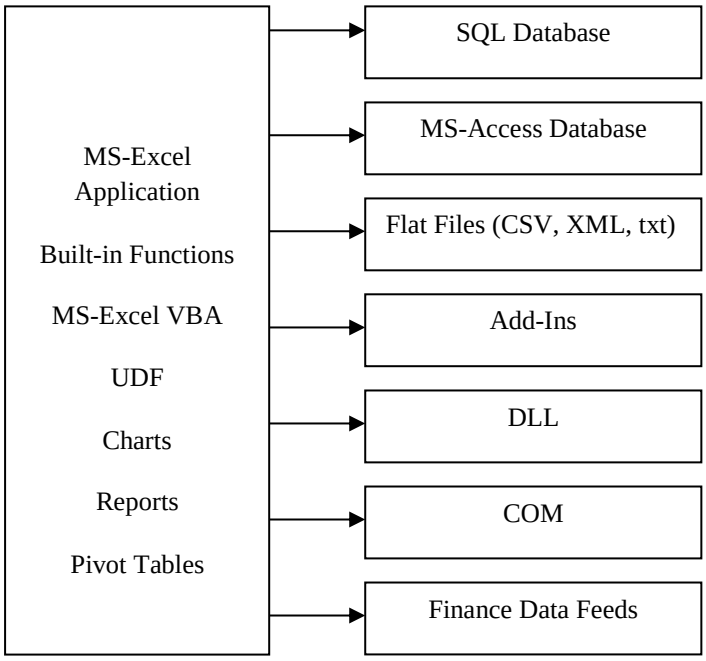


Figure 4.7: Typical MS-Excel application configuration for Northwest

4.8 MS-Excel Configuration for Distributed Processing

Each workstation has the Microsoft Office Professional suite as part of the company's software build. The MS-Excel applications that are used for the distributed processing system are copied to each workstation's local hard disk, and this process is managed by the distributed processing controller. MS-Excel has out-of-process instantiation functionality, that is, many mutually exclusive separate instances of MS-Excel can be instantiated in a single workstation. For distributed processing, each workstation starts a separate instance of MS-Excel and opens the appropriate MS-Excel application in that instance only. Hence, it will not interact with the user-initiated instances. The cluster node instance of MS-Excel has the following characteristics:

- Stay open while the workstation is in operation.
- Cluster-instantiated MS-Excel instance is not visible to the user interface.
- Users cannot access the cluster-instantiated MS-Excel instance.
- Excel instance operates as a background process.
- Excel instance is managed by the distributed processing controller.

Users can carry on working with their own MS-Excel application; meanwhile, the cluster instances of MS-Excel can be closed or opened remotely by the distributed processing management controller and for debugging purposes, the cluster instance can be switched to visible mode, if needed. Multiple instances of MS-Excel can be instantiated in a single workstation, and how many can be opened at the same time depends on the memory availability on the particular workstation. Figure 4.6 show multiple instances of MS-Excel in a single workstation. Each MS-Excel application that works with the distributed processing cluster is developed with the following functionalities:

- Maintain time stamp in the local message repository. This will facilitate the node controller to query each calculation node to monitor the MS-Excel status.
- Receive and send messages to local message repository.
- Maintain event logs that are continuously monitored by the distributed processing controllers and SQL database connections retrieve and update data.

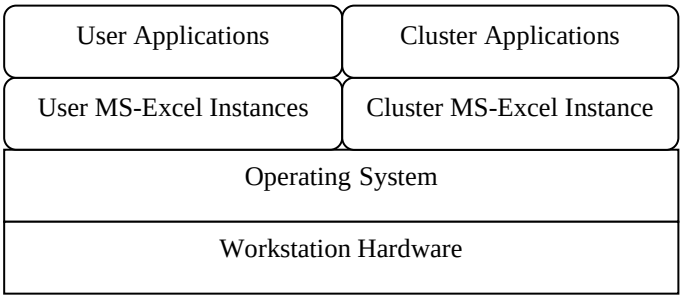


Figure 4.8: Multiple instances of MS-Excel in a single workstation

In each workstation, the calculations are performed by the cluster node's instantiated MS-Excel object and coordinates with the operating system to perform an efficient way of calculation depending on the workstation hardware such as number of CPUs, CPU-core, and memory size. The MS-Excel application executes the *[Object]*.*Calculate* command, where *[Object]* can be an individual cell, worksheet, or workbook. Once the calculate command is issued, from that point onward, MS-Excel takes control of the calculation and coordinates with the operating system to manage the best possible path for calculation algorithms and number of threads to be used for the calculations. To reduce the memory fill-up, minimum data is stored in the local MS-Excel application object, and most of the calculation-related data is kept in the SQL database and is only used when the calculation is performed. Using the MS-Excel multi-instance method, most of the MS-Excel applications that are used in the company can be modified to work with the distributed processing system in similar way. However, only compute-intensive applications need to be configured to work with the distributed processing system. The configurations for the MS-Excel applications required are as follows:

- Each workstation has its own local copy of the MS-Excel application.
- Applications have a common add-in that references the business logics, financial models, and user-defined functions (UDF).
- Applications fetch the data from the SQL server, perform calculations using a combination of built-in MS-Excel functions and UDF, and write back the output data to the SQL server database or local XML files.
- In a distributed batch processing scenario, the input and output data sets are kept as temporary XML files before the SQL batch update.

Hence, any MS-Excel application that has the data flow shown in Figure 4.9 can be used within the distributed processing system, and using local data files for distributed batch processing greatly reduces the network traffic-related time delays. The local file can be of CSV, txt, XML, MS-Excel type, or similar.

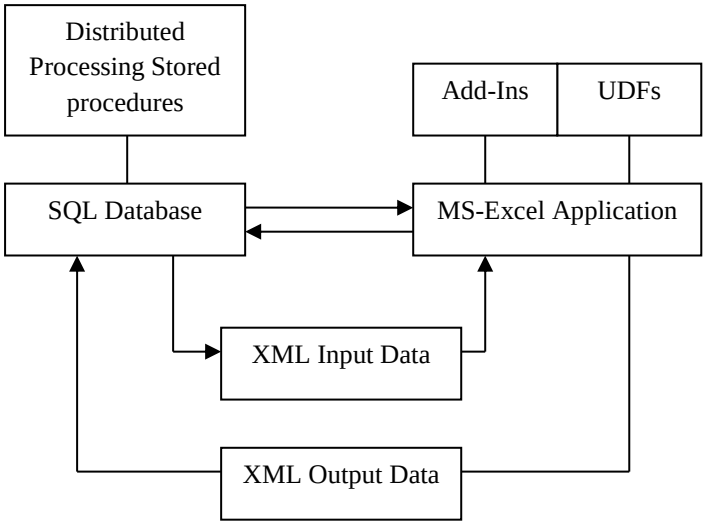


Figure 4.9: MS-Excel application structure for distributed processing

4.9 Risk Scenario Calculation System

For testing the MS-Excel application performance in the distributed processing system, the risk scenario's calculation system is used. This system is used in the company for calculating various portfolio risk scenarios and it has a complex CB pricing model that uses the binomial tree method to calculate various parameters for risk analysis. The reason for choosing the risk calculation system for testing is that the application is highly compute-intensive and requires considerable time to complete in a single workstation. Hence, the risk calculation system is a good candidate for testing the distributed processing performance against serial calculation methods. The input data that is used for the risk calculation system is a snapshot of the current data used in real-time trading. Figure 4.10 shows the risk calculation system data flow.

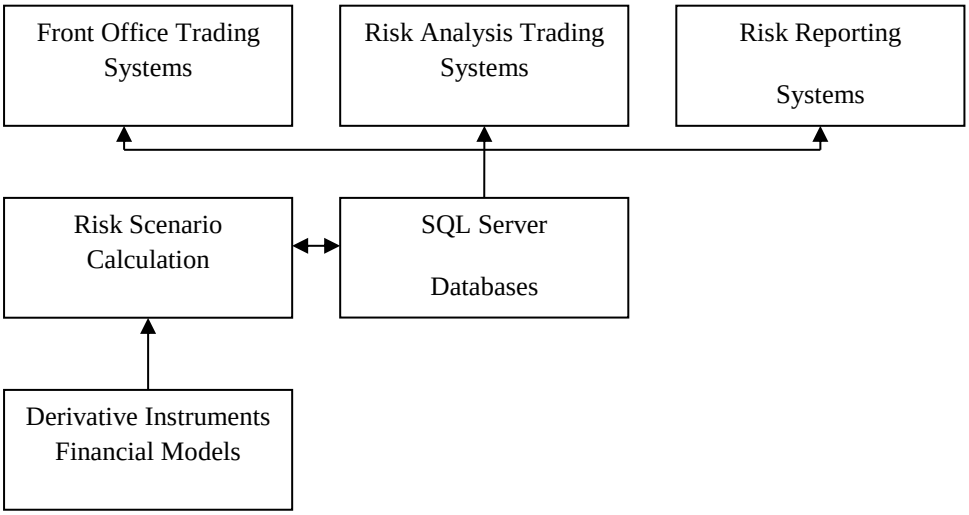


Figure 4.10: Risk calculation system data flow

Number of calculations is performed in the risk scenarios calculation system, and most of the calculations are simple arithmetic operations that require comparatively little time to calculate. However, the time-consuming part of the calculation is the CB pricing model calculations that employ binomial tree methods. Using the current portfolio, a single risk scenario using a 100-step tree model will take approximately 13 minutes to complete on an HP Z420 workstation with Xeon 3.06 GHz CPU and 12 GB memory. For certain risk simulation, over 100 scenarios have to be used, and in certain cases, due to time constraints, linear interpolation methodology is used to produce risk numbers without calculating. This method is acceptable where the parameter changes are small; however, for larger parameter changes or combinations of parameter changes, this method produces inaccurate results. Currently, the system calculates all the scenarios in similar calculations paths, and the calculations required for different scenarios vary depending on the number of factors. To implement this, requires various logics to make the system an adaptive processing system, and this method is discussed in detail in Chapter 5. To calculate a single risk scenario, the process sequence is shown in Table 4.7, and this process has to be performed for every scenario. The number of scenarios to be calculated depends on market conditions and analysis requirements by fund managers or by investors and auditors. Hence, the time required to complete the full calculation depends on a number of factors, such as what type of derivative is in the portfolio, type of scenario, number of calculation steps used in the mathematical models, and so forth.

Table 4.7: Risk calculation process sequence

Process	Description
(1) Acquire Data	This process will calculate the data using the SQL stored procedures and return a result set to the client MS-Excel application. The SQL stored procedure uses linked servers to get data from different SQL servers to calculate the required data. The MS-Excel application then sets the data for calculation in the input data ranges.
(2) Scenario Selection	This process sets the scenario parameters for the calculation. The scenario data is maintained in the SQL database.
(3) Calculation	This process is an MS-Excel calculation using MS-Excel formula and add-on VBA functions that utilise various CB models to calculate the number of output parameters for selected scenarios.
(4) Save	Calculated output data and corresponding input data are saved in the SQL database table with timestamp. The saved data will be used in various systems to produce different types of reports.

To calculate every scenario for each position, a number of ways of grouping the process and data. Three types of distributions exist: Data distribution, process distribution, or a combination of both. The data distribution splits the input data into smaller groups according to the process requirements and allocates each group of data to a particular calculation node workstation using algorithms to specify how to group the data efficiently. The process distribution allocates certain processes to certain calculation node workstations, and algorithms specifying how to allocate processes to the calculation node workstations efficiently. However, for testing calculation time efficiency, the same process is used in each calculation node workstation, and the data is grouped equally for comparing serial and distributed processing calculation time improvements. Hence, for testing the distributed process-based calculation efficiency using a workstation clusters, only data decomposition is used. Implementations of both data and process decompositions methods are discussed in detail in Chapter 7.

4.10 Message Passing Interface for the Calculation Cluster

The messaging passing interface for the Northwest cluster (MPI-NW) is designed to facilitate the communication between calculation nodes and the management controller within the distributed processing cluster. Each calculation node has its own message repository where the messages are stored and processed. The message repository is protected, and only the distributed processing management controller or local workstation has these access rights. Each calculation node has two main types of message repositories, calculation node-related and application-related, and has multiple sub-repositories for storing different types of messages. Currently, six messages used to communicate with calculation nodes, as shown in Table 4.8.

Table 4.8: Distributed processing control message descriptions

Message Description	Message
Initialise Workstation	*WSINI.message*
Open MS-Excel Application	*XLOpenApp.message*
Start MS-Excel Calculation	*XLCalc.message*
Execute VBS Application	*VBSExec.message*
Execute EXE Application	*EXEExec.message*
Read Application Parameter	*AppParam.message*

The message object has three methods for message management: *Send, Check, and Delete*, and each method has sub-classes. Figure 4.11 shows how the message class is constructed for each type of message.

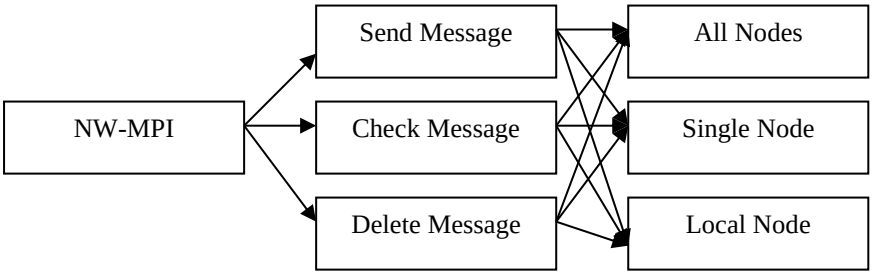


Figure 4.11: Message construction methods

The message object also has three methods for repository management: *Create, Check, and Delete,* and each method has sub-classes. Figure 4.12 shows how the message class is constructed for each type of command.

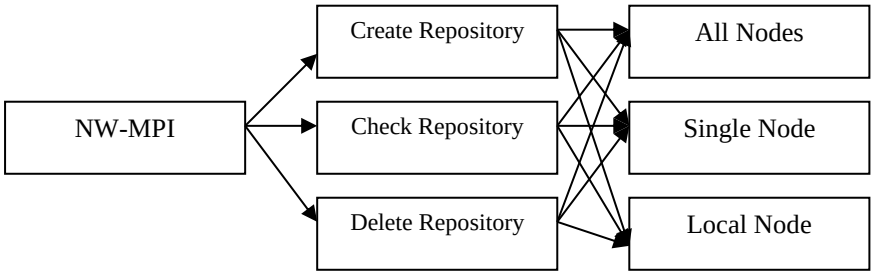


Figure 4.12: Repository construction methods

The distributed processing system implemented using the workstation cluster in the company's infrastructure is mainly intended for reducing calculation time of existing applications. In addition, it facilitates the future developments of various applications that require more processing power. A typical scenario is reducing a calculation time of 60 minutes that is usually required in a single workstation to around 3 to 4 minutes. Similarly, many hours of batch processing applications could be reduced to under an hour of batch processing using a distributed calculation cluster with multiple workstations. Hence, the processing time is considerably high compared with communication time, and the implemented message passing communication method that uses the TCP/IP is adequate for the required purpose. Therefore, in this scenario, communication time delays have negligible or minimal impact on overall performance. Although it is possible to reduce the calculation time even further by using inter-process message passing technologies, to implement the inter-process communication requires complex programming using various languages and involves modifying the existing applications to work with the inter-process communication infrastructure. This approach is not feasible for the current state of the applications used in the company. However, it may be possible in the future developments that these techniques can be implemented. Figure 4.13 shows workstation message repositories, and Figure 4.14 shows the message and data flow process.

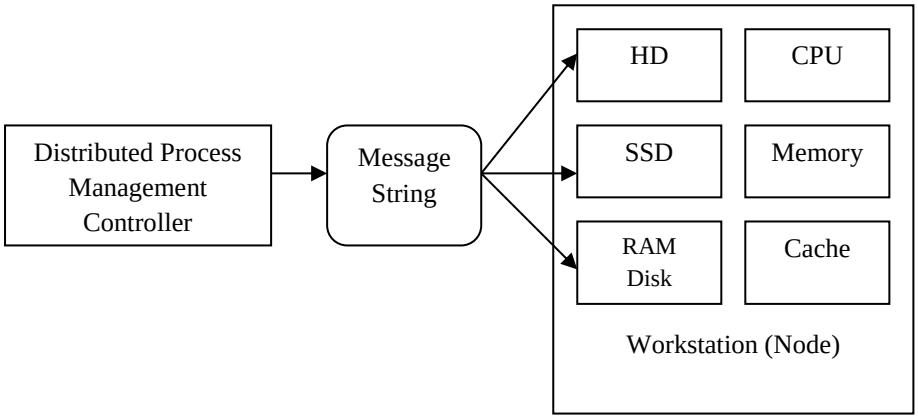


Figure 4.13: Workstation message storage repositories

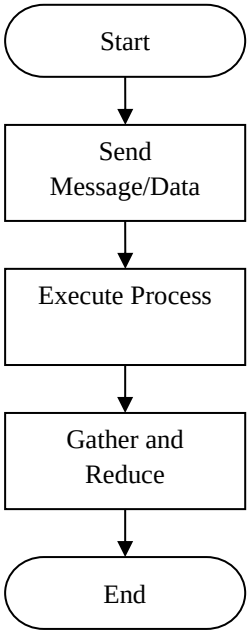


Figure 4.14: Message and data flow process

4.11 SQL Database Design for Distributed Processing System

The database is the core part of the distributed processing data management system and holds the business rules, cluster parameters data, and calculation data. The database is a part of the cluster controller system; hence, the database resides in the dedicated cluster controller server. Most of the database entities are designed to facilitate different types of calculation cluster configurations, and they have the flexibility to develop further if required in the future. The database is a relational database; hence, it has a certain order that connects each database object, and different types of database object can be added when needed. Table 4.9 lists SQL database table types and their roles.

Table 4.9: SQL database table types and their roles

Table Type	Description
Calculation Node Reference Table	All the calculation node workstation's details are maintained in the calculation node's reference table. This table is the main driver table, and the data is used for collecting various parameters in the cluster.
Parameter Table	Used for maintaining the system parameters and metadata for the distributed processing systems operations.
Temporary Tables	A number of temporary data holding tables that used for collecting each calculation node's various status data during the calculation node checking process. The data collected in these tables are used by the cluster controller for monitoring the cluster status and operations.
Computer System Parameter Tables	Used for holding each calculation node's system-related data. The data is provided by the Windows operating system's WMI classes. Collected data is used by the cluster controller for monitoring.
Event Log Table	Used for capturing critical events during the cluster operations; captured events data is used to analyse and monitor varous events during the cluster operations.
Historical Data Tables	Used for capturing historical data; captured data is used for monitoring and analysing cluster performance and applying effective scheduling algorithms.

4.11.1 SQL Database Table Design

Each table is designed to represent each entity, and a logical relationship between the entities corresponds to the relationship between SQL tables. System data tables are populated using WMI classes to capture the appropriate data from each calculation node workstation. Some data in the main table (*tbl_NW_COMPUTER*) is maintained manually for monitoring purposes, and this table has most of the calculation node workstation-related data. Table 4.10 shows SQL table names ad their descriptions.

Table 4.10: SQL table names and their descriptions

Table Name	Description
tbl_APPLICATION	Application-related data
tbl_BATCH_DATA	Batch processing-related data
tbl_BATCH_ID	Batch ID-related data
tbl_COMPUTER_TEMP	Workstation parameters
tbl_CPU	Workstation CPU-related data
tbl_CPU_CORE	Workstation CPU core-related data
tbl_CPU_TEMP	CPU parameters collected by WMI classes
tbl_CPU_USAGE	Workstation CPU usage data
tbl_DISK_DRIVE_TEMP	Workstation HD-related data
tbl_EVENT_LOG	Event/Error-related data for the whole system
tbl_MEMORY	Workstation memory-related data
tbl_MEMORY_TEMP	Memory parameters collected by WMI classes
tbl_MEMORY_USAGE	Workstation memory usage data
tbl_NETWORK_ADAPTER_TEMP	Workstation network card-related data
tbl_NW_COMPUTER	Workstation data
tbl_NW_COMPUTER_SCAN	Workstation status monitoring data
tbl_NW_DP_RISK_DATA_TEST	Risk calculation test data
tbl_PARAM_DATA	Distributed processing system parameters

The SQL database is mainly used to manage the following:

- Calculation node-related parameters
- Cluster controller-related parameters
- Historical performance data
- Calculation-related data
- Batch processing-related data
- Calculation performance-related data

The following parameters are collected on demand using WMI classes:

- Workstation hardware and software data
- Memory capacity data
- CPU, CPU-core, and CPU speed data
- Storage devices data
- Network card data
- Network connection speed data

The following parameters are continuously collected using WMI classes:

- Workstation availability related data
- Memory usage data
- CPU usage data
- Calculation start and finish times
- Processing events, warnings, and errors

Collected data is used by the cluster management controller for allocating tasks to each calculation node to perform a distributed calculation in an efficient way. How these collected data are used for efficient scheduling and load balancing algorithms is discussed in detail in Chapter 5. The SQL RDBMS database has sets of rules to which the data must comply for efficient database operations, and for distributed process management database, the rules are split into two categories: Hardware-based rules and software-based rules. Figure 4.15 shows the hardware-related entity relationship, and Figure 4.16 shows the software-related entity relationship.

The hardware-related entity relationship logical rules are as follows:

Each management controller can have many clusters

Each cluster can have many calculation nodes

Each calculation node can have many CPU units

Each calculation node can have many memory units

Each calculation node can have many network cards

Each calculation node can have many storage devices

Each CPU can have many cores

The software-related entity relationship logical rules are as follows:

Each application can have many programs

Each program can have many calculations

Each calculation can have may processes

Each process can have many events

Each event can have many time-slots

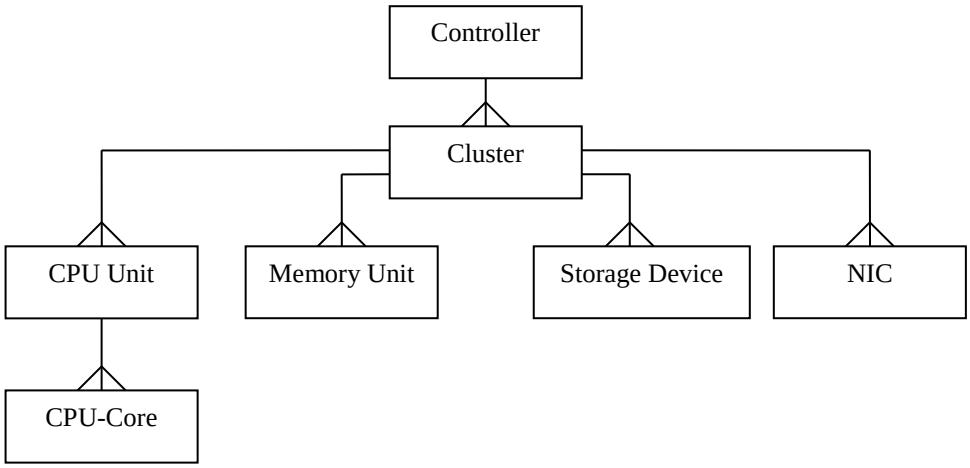


Figure 4.15: Hardware-related entity relationship

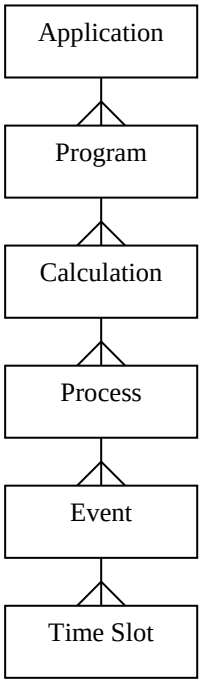


Figure 4.16: Software-related entity relationship

4.12 Distributed Processing System Design

The distributed processing system that has been implemented is based on the Beowulf-class cluster [9, 10]. The cluster consists of a head node, referred to as a management controller node, and a number of processing nodes, referred to as calculation nodes. The management controller node acts as the cluster controller and has the full responsibility of managing the whole cluster and coordinating with the calculation nodes. The calculation nodes are responsible for acting as calculation engines according to the commands given by the management controller. Figure 4.17 shows each calculation node's configuration, and Figure 4.18 shows the high-level schema of the distributed processing cluster. The distributed processing system consists of the following modules:

- Cluster management control server
- Calculation nodes
- Distributed processing management SQL database
- Distributed processing controller software
- Calculation node controller software
- Message-passing interface
- Distributed applications

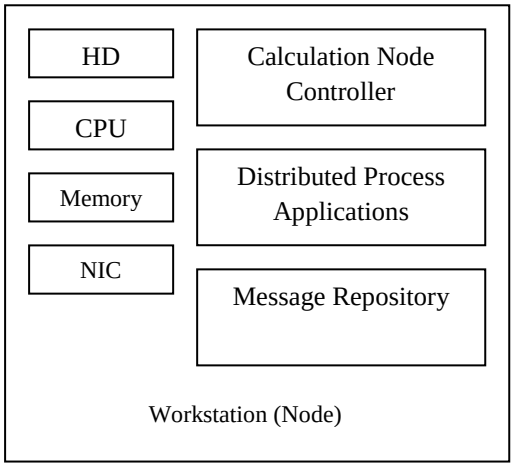


Figure 4.17: Calculation node configuration

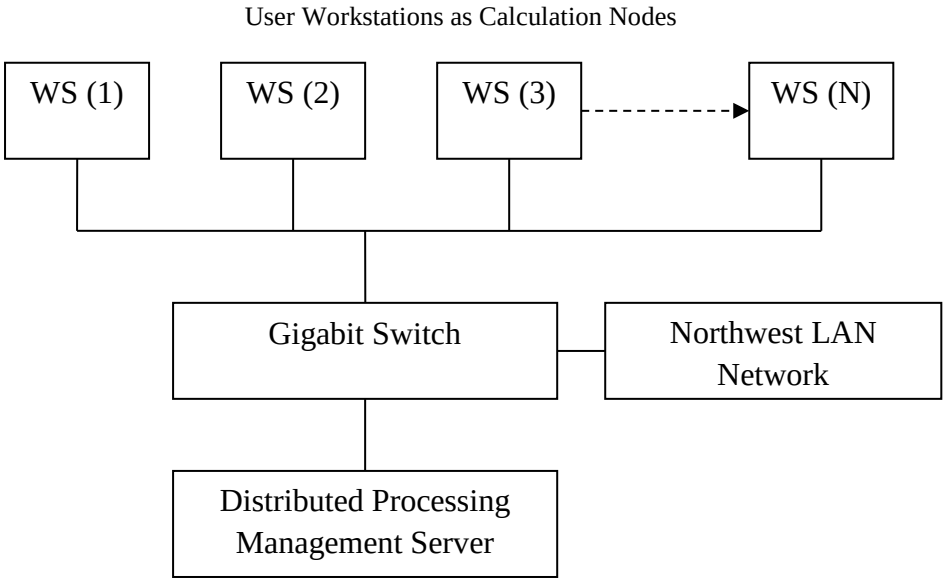


Figure 4.18: Calculation cluster setup high-level diagram

4.12.1 Distributed Processing Management Controller

The cluster controller node is configured with the SQL server database that is used for the cluster management system's data repository. The distributed processing management software has full access to each calculation node and required network's access security settings. The controller node is also used for calculating certain summary calculations at the end of the distributed processing calculations and performs intermediate calculation checks during the distributed processing. The distributed processing controller software is the main part of the distributed processing system that manages the overall systems. The controller communicates with calculation nodes and assigns tasks to each calculation node accordingly. The controller software is designed using Microsoft Access, SQL Server, VBA, VBScript, and WMI components to comply with the existing technologies used in the company. The controller application's user interface is shown in Appendix B. The controller has the following functionalities:

- Check each workstation for availability.
- Collect hardware, software, and operational data from each workstation.
- Communicate with calculation node's controller.
- Allocate tasks to each calculation node.
- Send and receive messages from calculation nodes.
- Manage load balancing within the calculation cluster.
- Manage adaptive processing to reduce calculation time.
- Process summary calculations and SQL query processing.
- Monitor the overall performance and collect various data during the operations.
- Perform diagnostic checks on each workstation.
- Calibrate the cluster to maximise the cluster performance.

Table 4.11 lists workstation selection to act as a calculation node based on workstation status and Table 4.12 lists control server data and message repository paths. Figure 4.19 shows the workstation selection process, and Figure 4.20 illustrates the workstation availability checking process. Figure 4.21 shows the message send process, and Figure 4.22 shows the task allocation process.

Table 4.11: Calculation node workstation's selection criteria

Workstation Status	**Description**
Not responding	Cluster controller queries each workstation before the allocate calculation task to check whether the workstation is available. Workstation may not be available due to fault or being switched off.
Fully excluded	Some workstations are manually excluded from the cluster due to various reasons such as fault or removed for maintenance.
Excluded during working hours	Some of the workstations are excluded from participating within the calculation cluster during working hours. These workstations are used by fund managers and traders, and these workstations can be used during out of trading hours.
Usage index	Workstation usage index calculated using current, historical, and point-in-time CPU and memory usage data.
Calculation index	Workstation calculation index calculated using historical calculation speed, CPU speed, number of cores, and memory size.

Table 4.12: Cluster control server data and message repositories

Repository Path	Description
\\WS\NWDP\WS\MSG\	Workstation Messages Repository
\\WS\NWDP\WS\DATA\	Workstation Data Repository
\\WS\NWDP\APP\MSG\	Application Messages Repository
\\WS\NWDP\APP\DATA\	Application Data Repository
\\$SRV\NWDP\CONTROL\MSG\	Controller Messages Repository
\\$SRV\NWDP\ CONTROL\DATA\	Controller Data Repository
\\$SRV\NWDP\ CONTROL \APP\	Controller Applications Repository

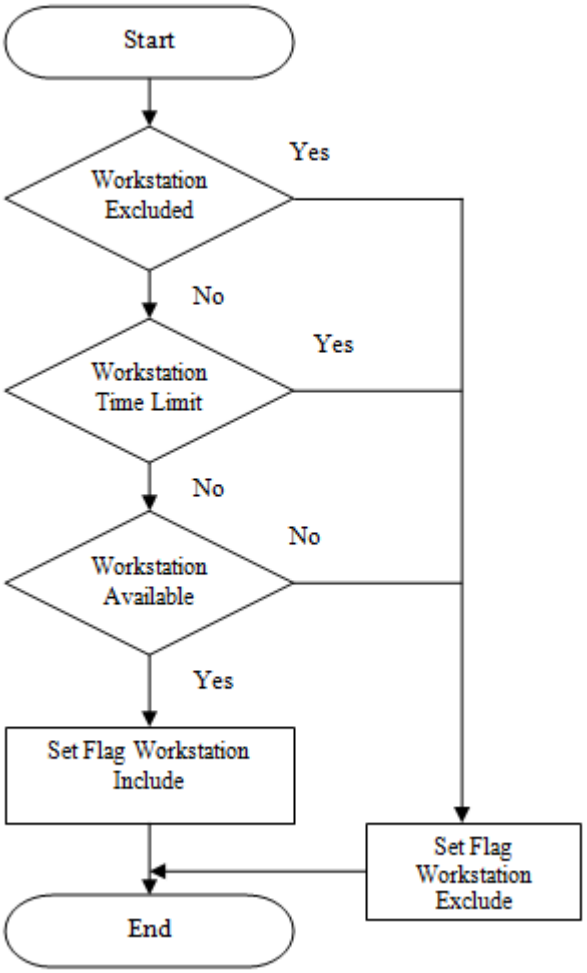


Figure 4.19: Workstation selection process

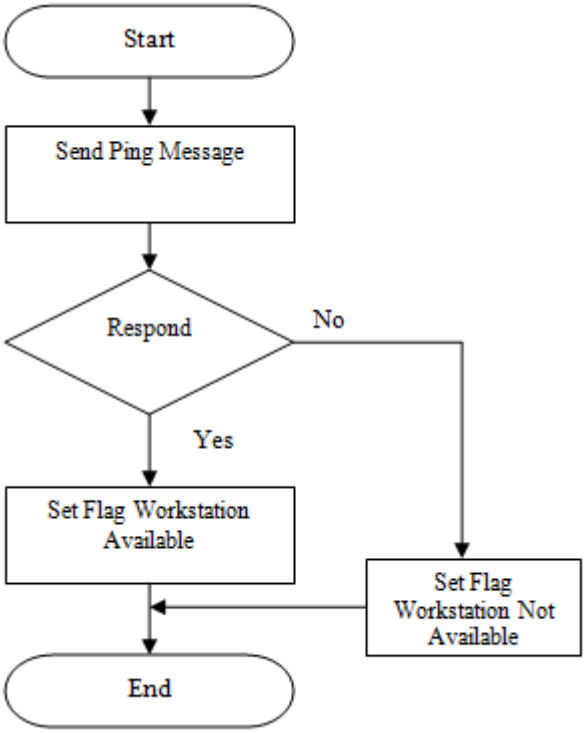


Figure 4.20: Workstation availability checking process

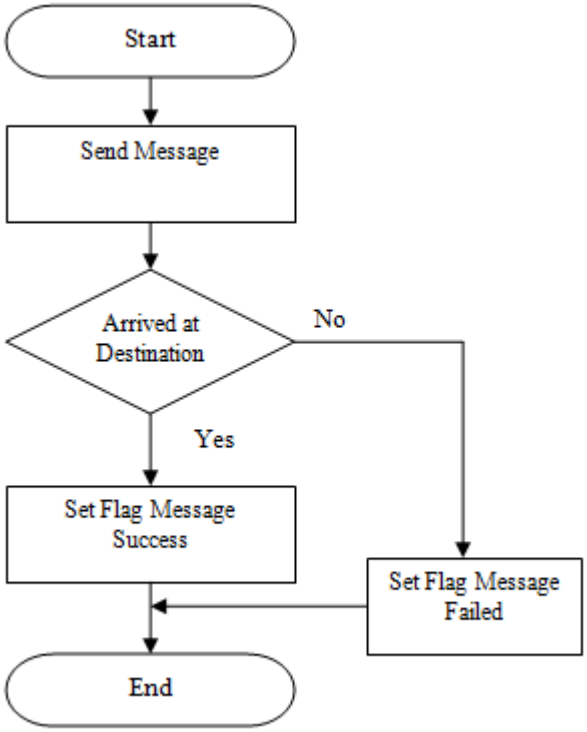


Figure 4.21: Message sending process to each calculation node

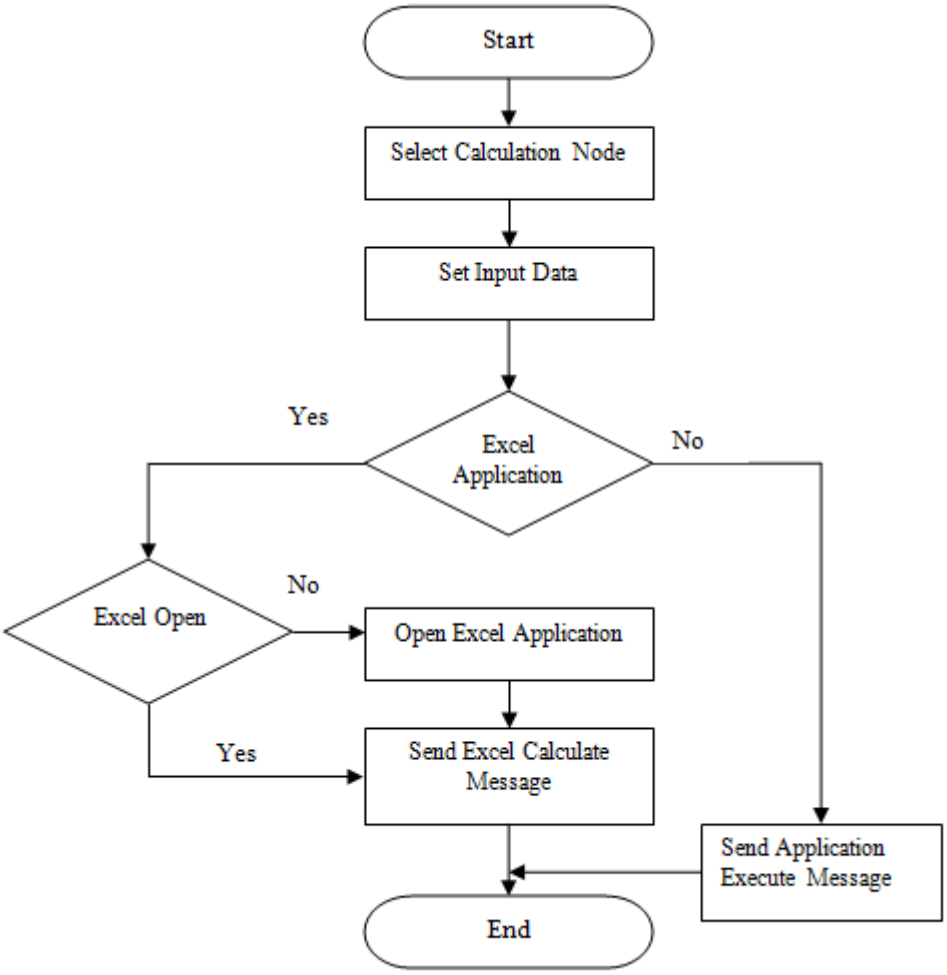


Figure 4.22: Task allocation process to each calculation node

The calculation index and usage index is not used in the distributed processing tests conducted that are described in this chapter; however, these are used in load balancing algorithms that are explained in detail in Chapter 5. Because the distributed process controller is used across all the calculation clusters, the initial design incorporates all the required functionalities that are needed for the forthcoming investigations in the following chapters.

The distributed processing controller has three modes of operations: Manual mode, Scheduler mode and Auto mode. These operation modes are used depending on what type of application is used in the distributed processing system. In manual mode, the controller application's user interface is used for managing the systems, and it has many functions for on-demand calculations and testing. The manual mode has the following functionalities:

- Select application for distributed processing.
- Check workstation status.
- Collect workstation parameters.
- Send messages to workstation.
- Select group of workstations to act as calculation node.
- Execute batch processes.
- Test prototype applications.

In the scheduler mode operation, the system executes predefined tasks at certain time intervals. These tasks are set as batch processing tasks, and each batch is independent from each other; hence, even if one batch failed, the system continues to process the next batch, and so on. During scheduler mode operation, all the available workstations will be utilised and most of the batch processing is executed during out office hours, such as during nights or weekends. Hence, maximum processing power is available for batch processing tasks. The auto-start process is a quick way of initiating the distributed calculation process as a single command rather than a sequential process. The auto-start process checks all the workstations that are available for calculation using defined sets of rules and assigns tasks accordingly. The auto-start is a type of batch command that starts all required processes in order and at once on demand.

4.12.2 Calculation Node Setup

Certain selected workstations in the company's network are configured as calculation nodes for the distributed processing cluster. The workstations are used by the company staff and categorised by their usage during working hours. Hence, the workstations that are heavily used will be excluded from participating within the cluster during working hours, and how the workstations participate within the cluster calculation is managed by the cluster's control manager software. The followings are checked by the cluster controller for each workstation before allocating calculation tasks:

- Workstation is manually excluded.
- Workstation is responding.
- Workstation is excluded during working hours.

Each workstation is configured to act as a calculation node with the following configurations:

- Cluster access security settings.
- Database access rights for distributed processing database.
- Remote access rights for remote management.
- Protected file share directories for local data and messages.
- Communication protocols for message processing.
- Local copy of distributed processing applications installed.
- Calculation node controller installed.

When the workstation boots up, the calculation node controller starts automatically, stays in the memory, and listens for messages from the cluster controller. When a message arrives from the cluster controller, the calculation node controller processes the message and executes instructions accordingly. Each workstation has a protected directory with multiple subdirectories for distributed processing applications and message processing. This directory is protected with required security settings. Table 4.13 lists the calculation node's message repository paths.

Table 4.13: Calculation node data and message repository paths

Directory	Description
C:\NWDP\	Shared distributed processing directory
C:\NWDP\WS\DATA\	Workstation-related data
C:\NWDP\WS\MGS\	Workstation-related messages
C:\NWDP\APP\$AppName$\Data\	Application-related data
C:\NWDP\APP\$AppName$\Msg\	Application-related messages

4.12.3 Workstation Usage

Most of the workstations used in the company can be categorised as shown in Table 4.14, and Table 4.15 lists the hardware and software configuration of each calculation node workstation.

Table 4.14: Workstation usage categorisation

Usage Status	Description
Heavy use	These computers are employed by users who perform multi-tasks, compute-intensive analysis, etc. These computers are used by financial analysts and traders.
Medium use	These computers are mostly used by general staff who rarely use the computer to full potential. These computers are used by administrative staff.
Occasional use	These computers are for special applications and/or terminal sessions, etc. This type of computer is occasionally or rarely used.
Never used	These computers are standby, surplus, or older computers. These computers are not utilised by any users.

Each workstation in the company's network is maintained at a high standard for 24x7 operations, and the following procedures are carried out frequently by the infrastructure team to ensure that the workstations are operating at optimal level:

- Hardware diagnostics
- Operating system updates
- Application and utility software updates
- Security software updates
- Password and access rights control

Table 4.15: Hardware and software configuration of each calculation node workstation

Parameter	Description
Model	HP Z420
CPU	Intel Pentium Xeon processor Single CPU with four cores (2.86 GHz per core)
Memory	12 GB DIMM
Hard Disk	Primary: 128 GB SSD Secondary: 500 GB SATA
Operating System	Windows 7 (64)
Application	Microsoft Office Professional Northwest applications Utility applications

The company's network infrastructure is managed and maintained at a high standard with minimum outage time. Hence, the distributed processing cluster is a part of the existing network and no need to perform extensive hardware and software tests before each batch calculation. Only distributed processing-related tests are performed before the calculation starts. All the workstations are configured for 24x7 operations; however, in certain situations, users might switch off their workstations or the workstations may become faulty or be removed for maintenance. Some workstations are manually excluded from the calculation cluster due to their heavy use, and these workstations are fully excluded. However, they are used in out-of-office hours when these workstations not used and are fully available. For example, traders' and fund managers' workstations are not used as calculation nodes during office hours. Each workstation is continuously monitored for memory and CPU usage, and the collected data is stored in the SQL database to analyse the workstation usage every 10 minutes. The collected data is used for calculating each workstation's usage index and allocating calculation tasks to each calculation node depending on each workstation's usage index. How these indexes are used is described in detail in Chapter 5.

4.12.4 Workstation Security

Workstation security is managed by the company's IT policy and procedures, and unauthorised access to the company network is not allowed. Only authorised staffs have access to the workstation using a network username and password.

- Protected by username and password for each user
- Local administrator rights are restricted
- Group security policy to limit the local access and control
- Physical cover removal alert system
- Restrictions on removal media usage

4.12.5 Windows 7 Operating System

Distributed processing system utilises the operating system for both management and communication processes, and it employs the WMI libraries for various tasks. The Windows 7 operating system manages the followings for each workstation at the operating system level:

- Memory management
- Process management
- Network control and communication
- Multiuser management
- Virtualisation management
- Security management
- User interface

4.12.6 Calculation Node Controller

The calculation node controller resides in the calculation node workstation and is responsible for communicating with the cluster controller software, local applications, and calculation engine. It collects data from the workstation such as hardware configuration, monitoring software status, and calculation performance. The calculation node's controller software is designed using VB6 and VBScript for consistency with the company's applications. However, it can be designed using any suitable programming language. The business logic is a crucial part of the controller design, and the programming language depends on the business environment that is used. The calculation node controller is a standalone EXE application that automatically starts when the workstation boots up. It has a timer-based looping mechanism and listens for messages from the cluster management controller. When a message is received from the controller, the calculation node controller executes appropriate actions accordingly. The application is designed as a standalone EXE program for simpler implementation and management for ease of implementation; however, it can be designed as a Windows service. The calculation node controller performs the following:

- Process messages from cluster controller.
- Collect workstation parameters and submit to the cluster controller.
- Collect CPU and memory usage in predefined interval and save as XML file.
- Collect performance parameters and send to the cluster controller.
- Open MS-Excel-based applications.
- Execute VBS, EXE, and CMD applications.
- Send events, warnings, and errors to cluster controller.

Every workstation in the calculation cluster is continuously monitored for various parameters, and the cluster management controller and the calculation node controller coordinate together to collect the parameters from each workstation to maintain them in a failproof status to maximise the cluster performance. It has the option to process messages from the cluster management controller to monitor and diagnose the

calculation node hardware and software. However, these functionalities are fully utilised in the further investigation that is described in the forthcoming chapters. Every 10 minutes, CPU and memory usage of each workstation is captured by the calculation node controller and saved in a local XML file. Data is recorded for monitoring between 6.00AM and 9.00PM and saved as a local XML file. The saved XML file is used by the cluster management controller for analysing workstation usage. On a daily basis, the saved data is uploaded to SQL database tables for historical analysis. Figure 4.23 shows the calculation node's controller process. Figure 4.24 shows the calculation node's controller message process.

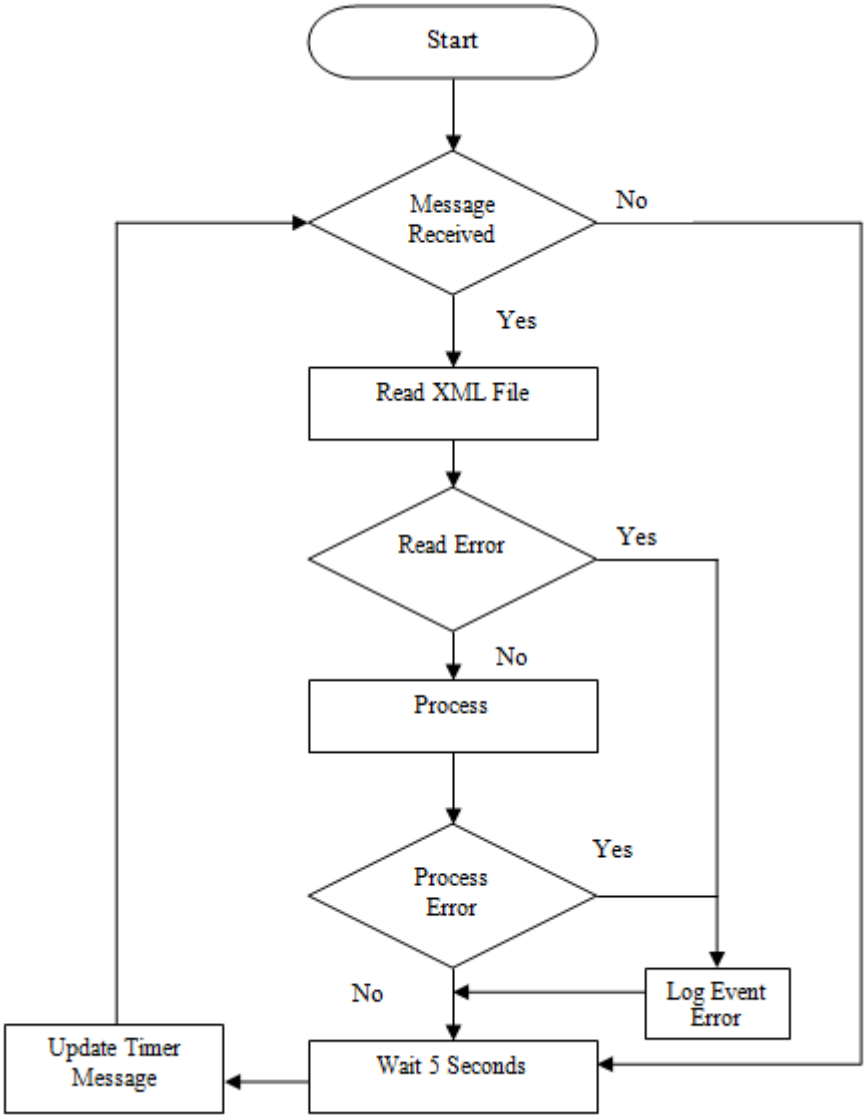


Figure 4.23: Calculation node controller process

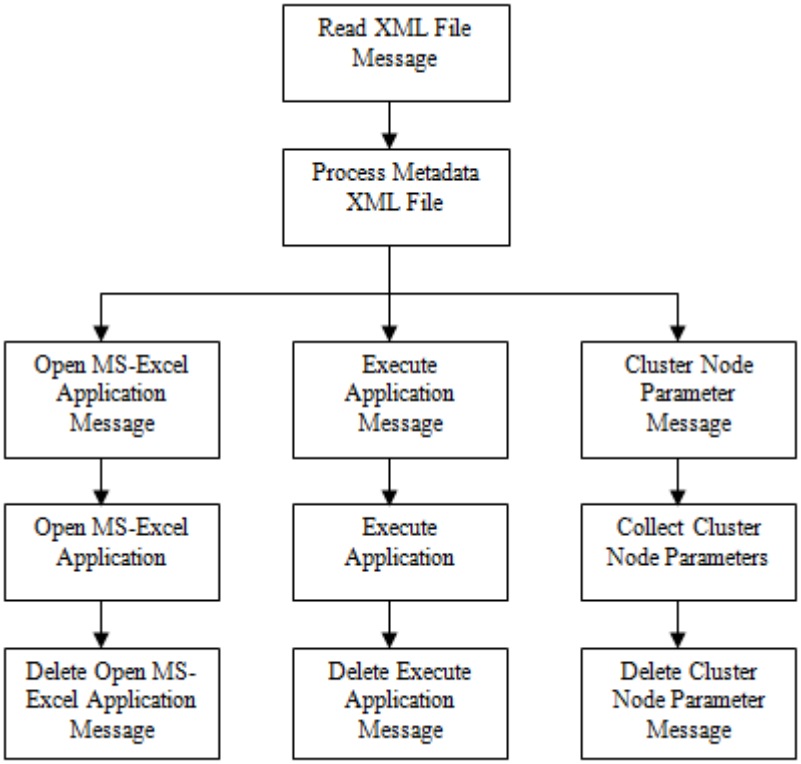


Figure 4.24: Calculation node controller message processing

The calculation node controller has a five-second delay, and it checks for messages in the workstation message repository; when a message, it processes the message accordingly. The cluster management controller sends a metadata XML file and a separate message to instruct the calculation node controller to read the file. Once the read message has been received, the calculation node's controller processes the message, reads the metadata XML file, and executes processes accordingly. Three types of messages processed by the calculation node controller:

- Excel application open
- Executable application execute
- Collect parameter or monitor workstation CPU and memory usage.

For MS-Excel applications, the calculation node controller only opens the MS-Excel application and once the application has been opened, then onwards, the application takes control of it. For executable applications, the calculation node controller executes the application using a shell command. If any errors are encountered while

opening an MS-Excel application or running a shell command to execute, then the calculation node controller logs the error description to an event log table. Otherwise, the applications will take control of it once it has been successfully opened or executed. The distributed processing is performed efficiently by coordination between the cluster controller, the calculation node controller, and the applications that are utilised within the cluster. Figure 4.25 shows the application execution process. Figure 4.26 presents the MS-Excel application calculation process.

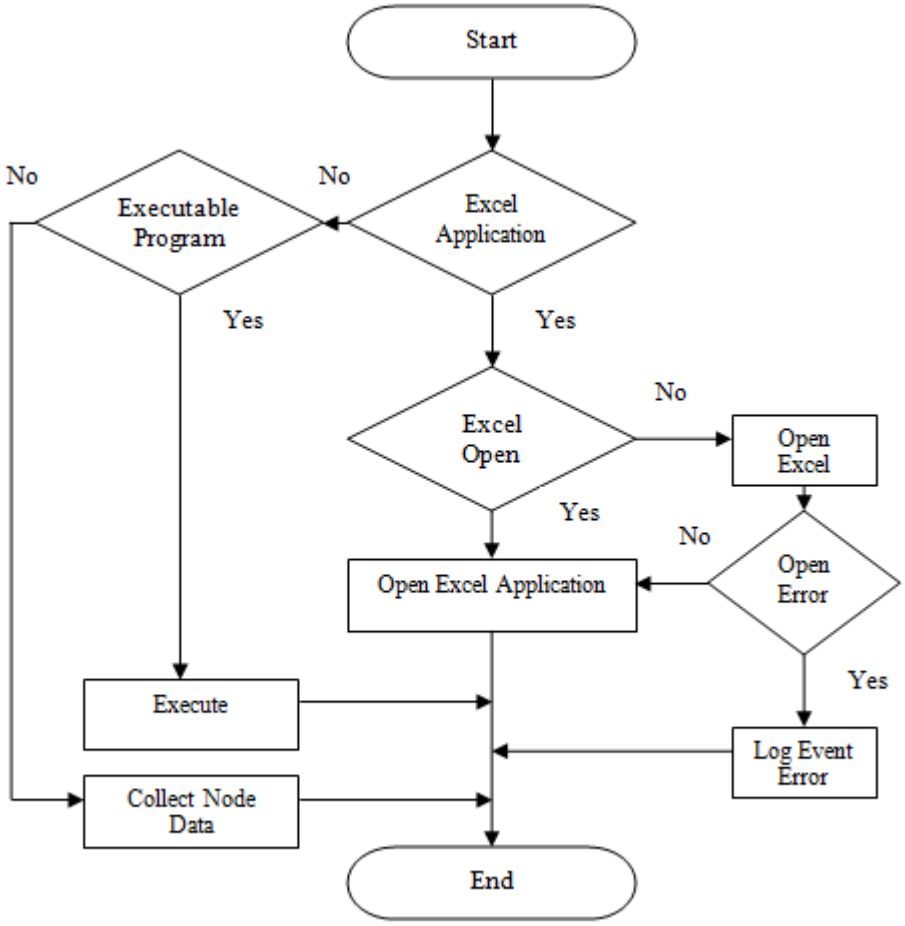


Figure 4.25: Application execution process in each node

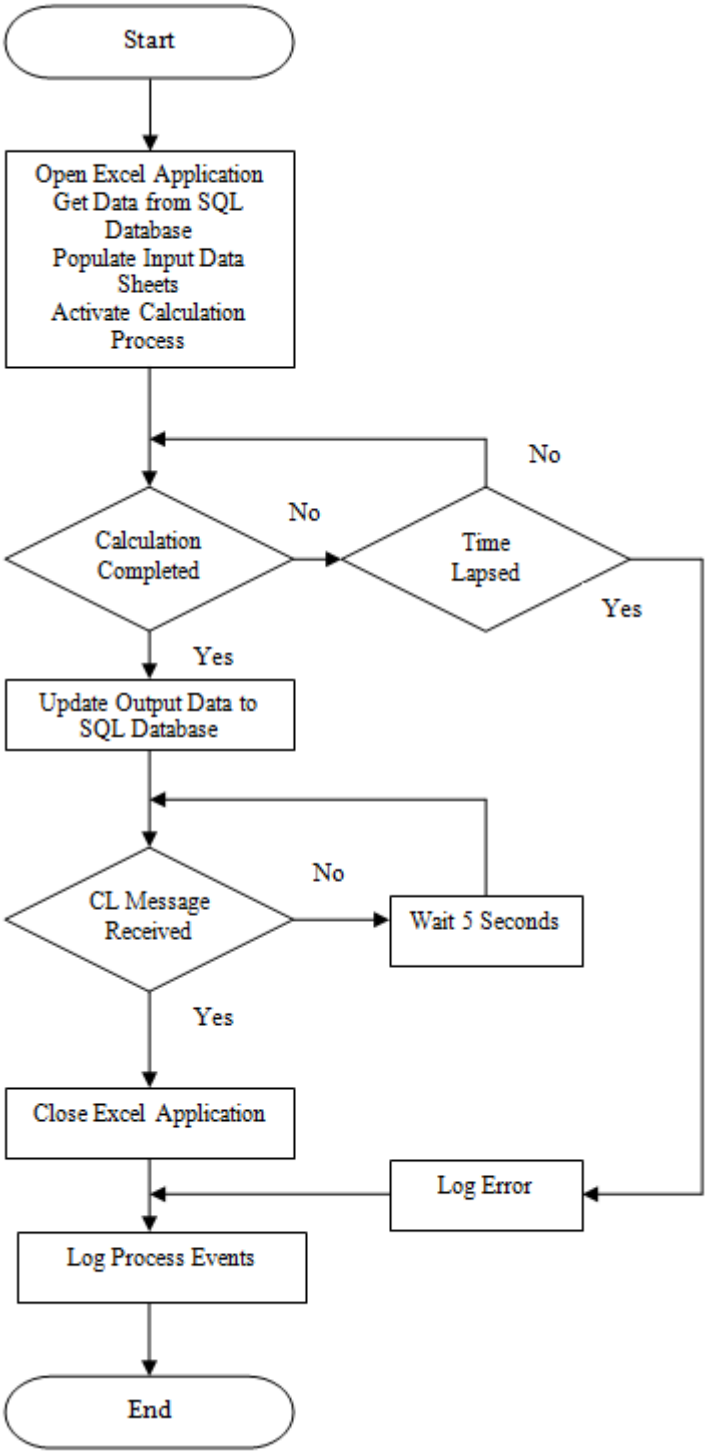


Figure 4.26: MS-Excel application's calculation process

4.12.7 Process Distribution Method

The calculation process is distributed by selecting the appropriate processes that can be executed as parallel calculations using the domain decomposition method. All other processes that cannot be parallelised are calculated by the controller node server as serial calculation. These processes are mostly single calculations that require high levels of dedicated processing power such as summarising calculations and query processing. For illustration, if four process A, B, C, and D need to be calculated and only process B is distributed {B(1), B(2)…B(N)}, where N is number of calculation nodes in the cluster, the others (A,C,D) are not suitable for distributed calculation. Figure 4.27 shows how optimised processing can be achieved by a combination of serial and distributed processing cluster.

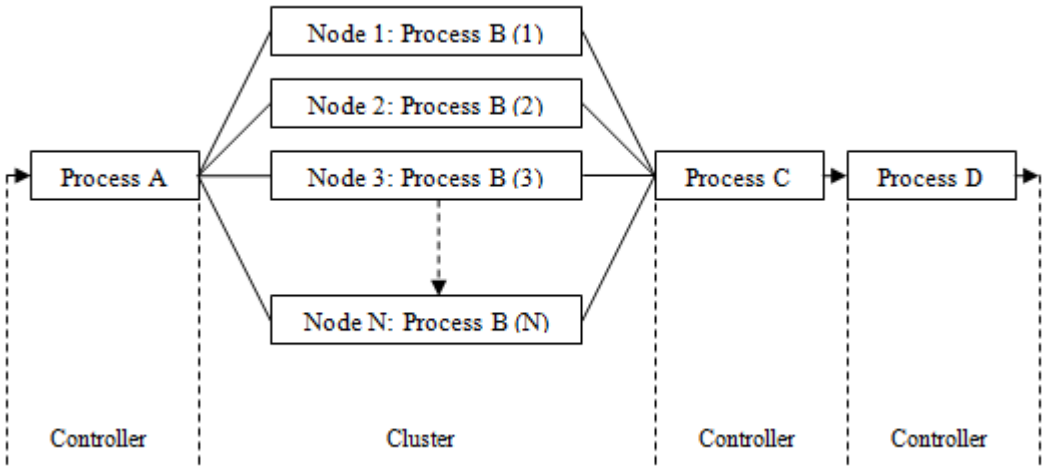


Figure 4.27: Process distribution as distributed and serial calculations

4.12.8 Total Calculation Time

The total calculation time required for a given calculation task to complete consists of several types of time delays. In calculation time only testing, the calculation time is mainly concerned with the wall clock time for the calculation to be completed. However, in the real application simulation scenario, a number of factors need to be considered. For the selected distributed processing test scenario, six separate time delays are involved per calculation, as shown in Figure 4.28.

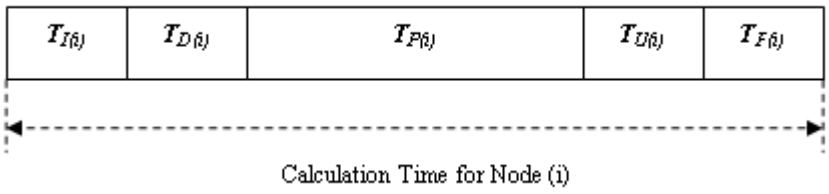


Figure 4.28: Total processing time of each node

where

T_I Initialising the applications time

T_D Fetching input data from database and setting data time

T_P MS-Excel processing time (calculation time)

T_U Updating calculated data back to SQL database time

T_F Finalising the applications time

T_M Messaging time (Total time taken for send and receive messages)

In comparison, messaging time (T_M) is small compared with calculation time (T_P) and the total calculation time depending on various factors such as network latency, SQL server calculation time, data transfer time, and in addition, how the distributed processing is executed. A few scenarios can introduce certain time delays, such as if an MS-Excel instance is not running on the particular calculation nodes, then the distributed processing controller sends messages to each calculation node to start and initialise the MS-Excel application; this will cause initialisation time delay (T_I). In addition, if the MS-Excel application is running on the calculation node and has to be closed at the end of a calculation, then the distributed processing controller sends messages to the calculation node to finalise and close the MS-Excel application; this will cause finalisation time delay (T_F). Furthermore, where each batch process of calculation requires new input and output datasets, then the distributed processing controller sends messages to each calculation node to fetch the data from the SQL server for each calculation; this will cause time delay ($T_D + T_U$). Figure 4.29 shows the cluster's total processing time. Total calculation time for a given task *k* in a calculation node *n* is shown in Equations (4.17) and (4.18).

$$T_K(n) = \sum_{i=1}^{N} (T_p(i) + T_c(i) + T_I(i) + T_F(i)) \tag{4.17}$$

$$T_c(i) = \sum_{j=1}^{M} t_c(j) \tag{4.18}$$

where

$T_K(n)$ Total calculation time taken by node n to complete the task k

$T_p(i)$ Processing time taken by process i

$T_c(i)$ Total communication time taken by process i

$t_c(j)$ Discrete communication time for message j

$T_I(i)$ Initialising time taken by process i

$T_F(i)$ Finalising time taken by process i

N Number of processes required to complete task k

M Number of communications messages required to complete task k

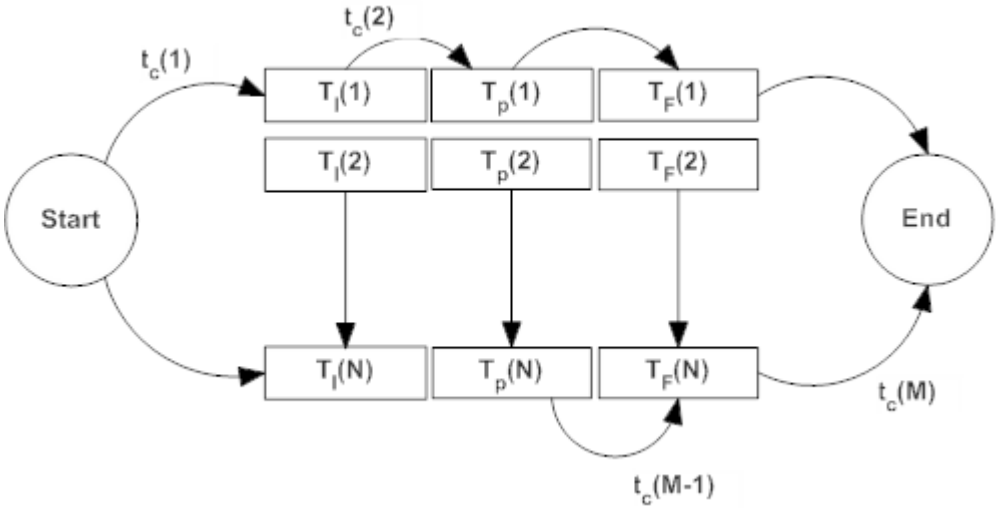


Figure 4.29: Total processing time of cluster

4.13 Distributed Processing System Tests

For testing and comparing of different parameters under controlled conditions, equation 4.11 that described in section 4.2.1 is used. In addition, workstations with similar hardware are selected to act as calculation nodes in the calculation cluster. With respect to test data, the input dataset is extracted from the current portfolio, and the current portfolio consists of complex nonlinear datasets; however, for testing, a suitable dataset is selected with similar data input and output points. The testing environment is established for investigating the following scenarios:

- Serial and distributed processing-based calculation performance comparison.
- Correlation between numbers of workstations used in the cluster and total calculation speed improvements.
- Office hours and out-of-office hours batch processing comparisons.
- Analysis of the cluster performance in real-time simulation scenario.

For testing the distributed processing performance, the company's risk analysis system is used with appropriate modifications that are needed for work within the distributed processing cluster. The risk analysis system that is currently used is based on MS-Excel and it has the following properties:

- Connections to SQL database for input and output data.
- Built-in MS-Excel functions are used for various calculations.
- Company's proprietary financial models written in VBA are used for calculations.
- Uses batch processing using schedulers and on-demand-based manual calculation process for producing risk reports.

The risk calculation system is modified to work within the cluster by implementing additional programming to perform as a calculation service in the calculation node. The modified system has the following functionalities that are added to act as a calculation service within the cluster:

- Processing distributed processing messages.
- Processing metadata input and output parameters.
- Autonomous data processing and calculations based on system timer loop.

For test data, Japanese CBs (convertible bonds) are selected for calculating various risk scenarios. The data is extracted from the current portfolio that has many CBs issued by many countries; however, the Japanese CBs have simpler structures compared with other countries such as China. Hence, the dataset selected is highly suitable for various performance comparisons within the controlled test environment. The test dataset has 75 input parameters, 32 output parameters, and 10 different risk scenarios. All the tests and experiments performed are under certain controlled conditions to ensure that the data, applications, techniques, and methods used are consistent throughout the tests and experiments to guarantee that the results can be compared accordingly. However, in real-time trading scenarios, the parameters will continuously change, and these investigations are discussed in detail in Chapter 7. In addition, further investigations performed on remote process management to make the system more stable and robust by utilising various control mechanisms and load balancing techniques, and these are discussed in Chapter 5.

4.13.1 Workstation Allocation Method

Because the workstations used as calculation nodes are employed by the users to various degrees, all the workstations are continuously screened for availability and system status. Some workstations are manually excluded due to various factors such as hardware or software fault or being under maintenance. Some workstations have rule-based exclusions such as only being available during certain periods to avoid disruption to certain users as discussed earlier. How these workstations participate in the calculation cluster is based on various rules that are managed by the distributed processing controller software. However, the risk scenario test is conducted during out-of-office hours to ensure that the selected workstations are fully available for processing.

4.13.2 Calculation Node Test

For testing the performance of each calculation node, local data is used and by using the local data, the total calculation time measured only includes the MS-Excel calculation time (T_P), that is, the calculation time taken by MS-Excel to complete the task from start to finish. Hence, the calculation performance is measured against the number of CBs and the time taken for the calculation to be completed. As expected, the number of CBs and the calculation time are inversely correlated, as shown in Table 4.16 and Figure 4.30. However, for performing distributed calculation using a number of calculation nodes, the SQL Server database, and message passing, the calculation time reduction becomes nonlinear and complex due to various factors.

Table 4.16: Correlation between number of CBs and total calculation time

Calculation Time (sec)	Number of CBs
2,663	4,000
1,394	2,000
747	1,000
415	500
195	250
78	125

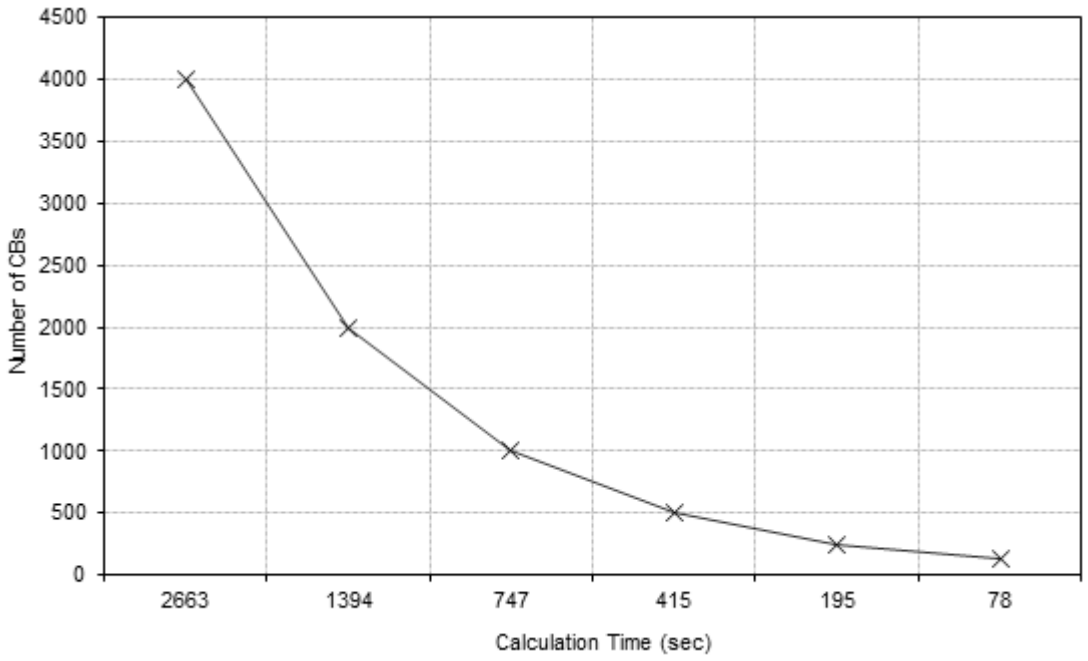


Figure 4.30: Correlation between number of CBs and total calculation time

4.13.3 Distributed Processing Tests Using CB Financial Model

Using equation 4.11, the following parameters are used for testing a single risk scenario, that is, using volatility (*v)* is equal to 1% and using the domain decomposition method to distribute the calculation to each calculation node. Test parameters are as follows:

Number of CBs used:	350
Number of scenarios:	1 (v=1%)
Number of calculation nodes:	18

The calculation tests have demonstrated that by increasing the calculation nodes, the calculation time is reduced substantially. However, the reduction in calculation time is nonlinear due to the following factors:

- The financial CB model calculation itself is nonlinear in nature.
- Hardware related nonlinearities.
- Network related latencies.
- SQL server data processing and time delays due to updates
- CPU and memory usage is varied for each calculation node workstation.
- Data transfer time delays.

In addition, a minimum calculation time is observed depending on time delays T_I, T_D, T_U, and T_F. Hence, the benefit of using multiple calculation nodes to reduce the overall calculation time depends on the minimum calculation time limit. Therefore, the risk scenario's calculation application, the calculation task distribution, has to be optimised to get the maximum benefit from the distributed processing implementation. The efficient task distribution algorithms using load balance methods to reduce the calculation further are investigated in Chapter 5. Table 4.14 and Figure 4.29 show the correlation between the number of calculation nodes and total calculation time.

The time delays, T_I, T_D, T_U, and T_F are constant for small datasets, and if the dataset is getting larger, in the range of over 10,000 calculations per node, then these time delays can be considerably higher. By excluding initialising, finalising time delays, and using the same hardware configurations, the calculation time reduction is inclined towards linear reduction. Hence, the calculation reduction is inversely proportional to the number of calculation nodes for linear types of calculations. Therefore, it is proved that in practice, calculation time can be reduced further by increasing the number of calculation nodes. However, a certain limit that calculation time no longer improves with the increase of calculation nodes. This is due to the minimum calculation time required to complete each process, and any reduction below this limit will have no impact on further improvements.

Table 4.17: Number of calculation nodes against calculation time

Number of Calculation Nodes	Calculation Time (sec)	Speedup Factor
1	289	1.00
2	271	1.07
3	115	2.51
4	96	3.01
5	96	3.01
6	83	3.48
7	88	3.28
8	75	3.85
9	75	3.85
10	72	4.01
11	79	3.66
12	76	3.80
13	68	4.25
14	61	4.74
15	79	3.66
16	65	4.45
17	64	4.52
18	62	4.66

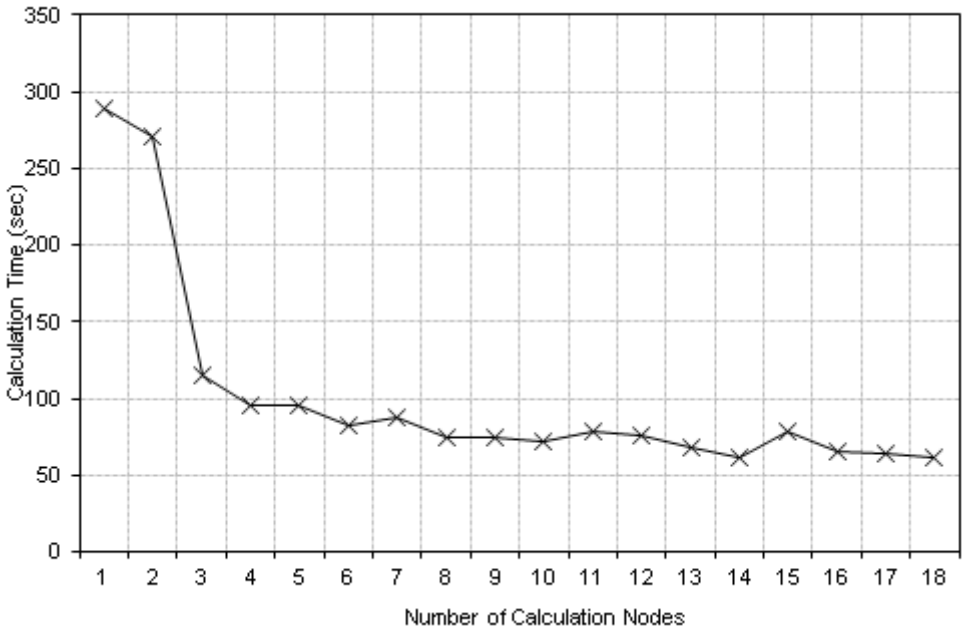


Figure 4.31: Number of calculation nodes against calculation time

4.13.4 Minimum and Maximum Calculation Time

When a group of workstations is selected to execute calculations, the calculation time varies within the group and depends on various factors mentioned earlier. Table 4.18 and Figure 4.32 show the minimum and maximum calculation time discrepancies. By observing the minimum and maximum calculation times for calculation-only execution, that is, excluding initialisation (T_I) and finalisation time (T_F) delays, the system behaviour leans towards linear inverse correlation between calculation time and number of calculation nodes.

Table 4.18: Minimum and maximum calculation time for each node

Number of Calculation Node	Minimum Time (sec)	Maximum Time (sec)	Minimum Speedup Factor	Maximum Speedup Factor
1	290	290	1.00	1.00
2	156	160	1.86	1.81
3	102	121	2.84	2.40
4	88	96	3.30	3.02
5	76	91	3.82	3.19
6	71	79	4.08	3.67
7	62	72	4.68	4.03
8	59	69	4.92	4.20
9	54	69	5.37	4.20
10	51	69	5.69	4.20
11	53	66	5.47	4.39
12	44	66	6.59	4.39
13	42	66	6.90	4.39
14	38	60	7.63	4.83
15	24	58	12.08	5.00
16	23	56	12.61	5.18
17	21	57	13.81	5.09
18	19	51	15.26	5.69

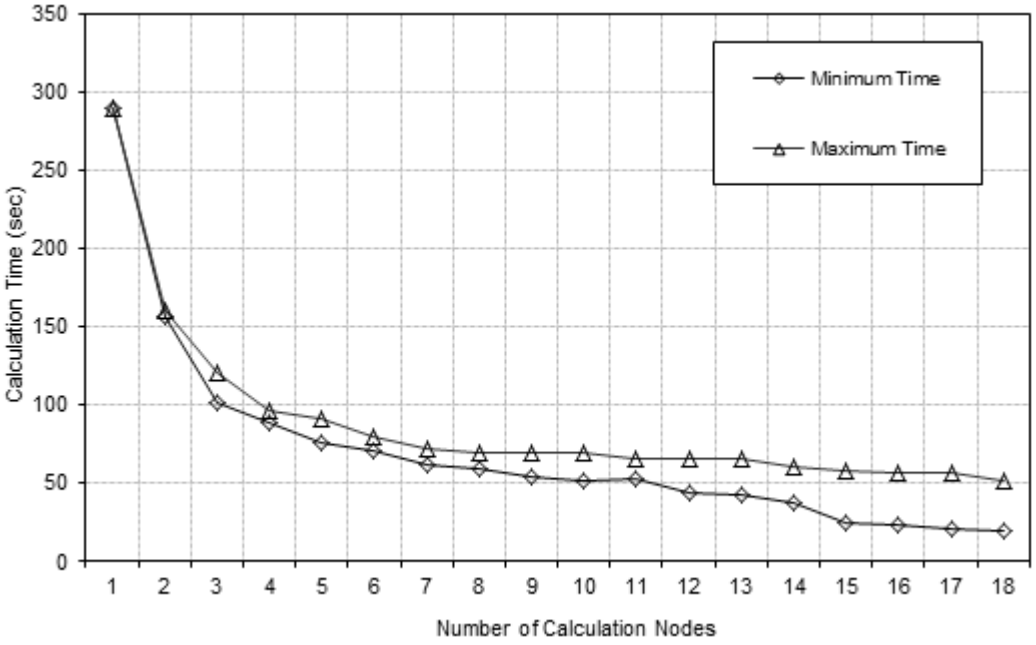


Figure 4.32: Minimum and maximum calculation time for each node

4.13.5 Using Simplified Linear CB Financial Model

For further testing and proving the concept of the process distribution and calculation time relationship, a linear binomial tree-based CB model is used to calculate gamma values for varying volatility and parity. However, in real-time trading scenarios, the gamma calculation is considerably complex and inherently nonlinear. For testing purposes, the nonlinear parameters are removed and the model is simplified, so that the testing can be carried out in an environment with controlled parameters. Table 4.19 and Figure 4.33 show the calculation results indicating linearly correlated reduction in calculation time with number of calculation nodes. Table 4.20 and Figure 4.34 show the minimum and maximum calculation times for linear model calculations. To test the linear CB model, the input parameter for the equation 4.11 is changed those nonlinear parameters in dataset *{X}* is set to zero values. The test parameters used for the linear calculation model are as follows:

Number of CBs:	350
Number of calculation nodes:	16

Table 4.19: Linear model calculation time and calculation node correlation

Number of Calculation Node	Calculation Time (sec)	Speedup Factor
1	133	1.0
2	68	2.0
4	35	3.8
8	18	7.4
16	10	13.3

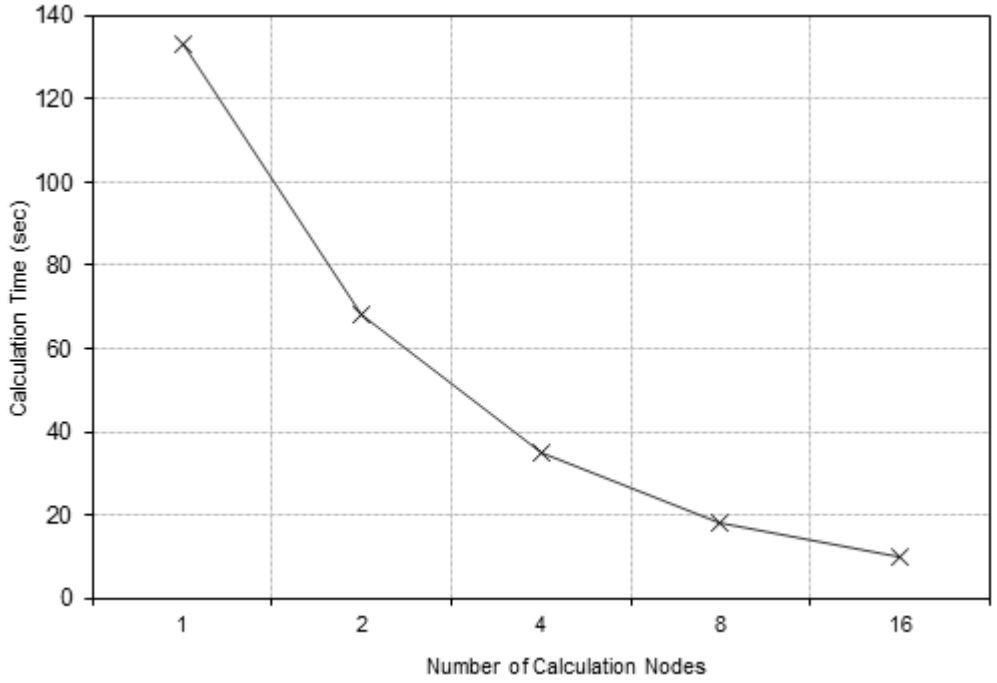


Figure 4.33: Linear model-based calculation time and calculation node correlation

Table 4.20: Linear correlation between calculation time and number of calculation nodes

Number of Calculation Node	Minimum Time (sec)	Maximum Time (sec)	Minimum Speedup Factor	Maximum Speedup Factor
1	290	290	1.0	1.0
2	156	160	1.9	1.8
4	88	96	3.3	3.0
8	59	69	4.9	4.2
16	23	56	12.6	5.2

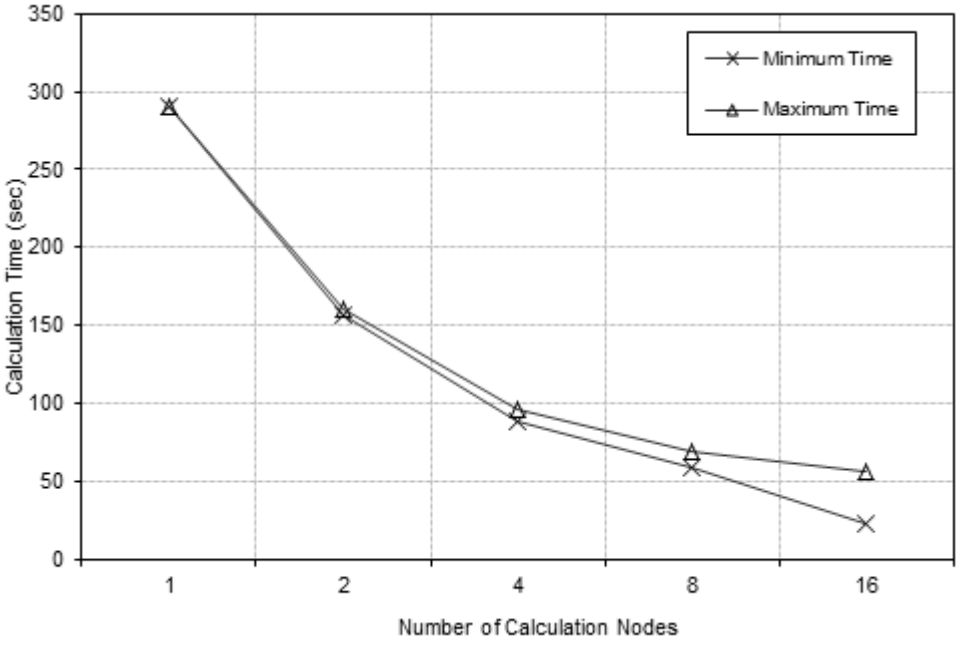


Figure 4.34: Linear correlation between calculation time and number of calculation nodes

4.13.6 Multiple Scenario Calculations on Same Dataset

For testing multiple scenarios on the same dataset, the following is implemented:

- Same data set supplied to each calculation node as local XML file.
- Each node calculates multiple scenarios and saves the data in local XML file.
- Single node calculates all scenarios.

Using equation 4.11, volatility (v) is used as varying parameter for scenario testing:

Number of CBs: 350

Number of scenarios: 10 (v = 1% to 10% step size=1)

Number of calculation nodes: 10

The calculation result shows that the multiple scenario calculation on the same dataset is a more efficient way of reducing calculation time for a given batch and, as expected, a linear inverse correlation between calculation node and total calculation time is observed. Table 4.21 and Figure 4.34 show the correlation between calculation time and number of calculation nodes for the multiple scenario calculation.

Table 4.21: Number of calculation nodes used vs calculation time for multiple scenario calculations

Number of Calculation Nodes	Calculation Time (sec)	Speedup Factor
1	5,726	1.0
2	3,107	1.8
3	2,377	2.4
4	1,983	2.9
5	1,254	4.6
10	613	9.3

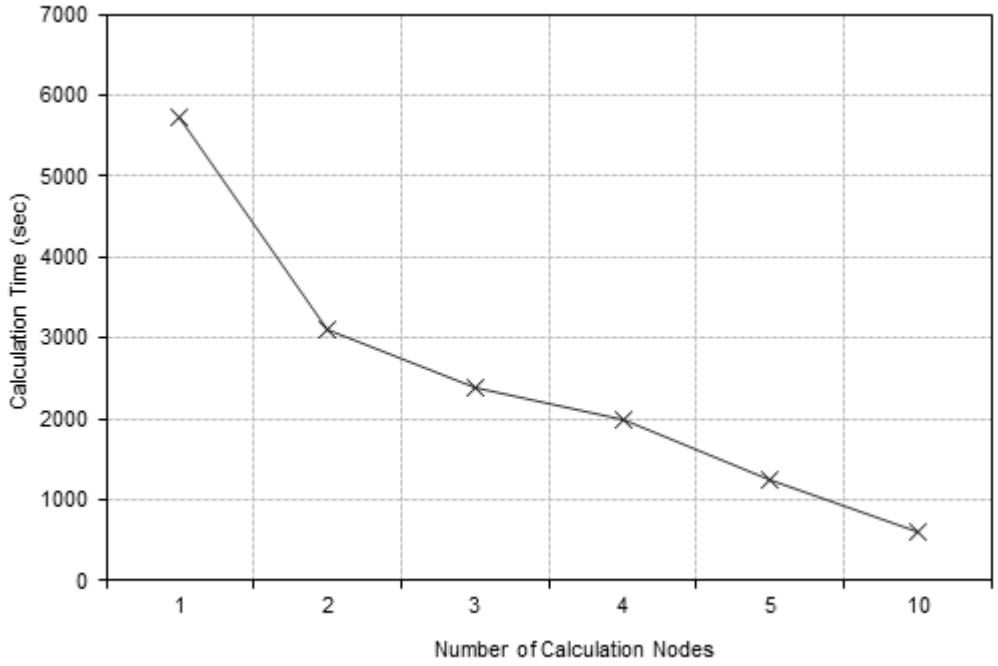


Figure 4.35: Number of calculation nodes used against calculation time for multiple scenario calculations

4.14 Chapter Summary

The investigation conducted has demonstrated the original contribution to the company specific design methods and implementation techniques using workstations as calculation nodes to form a highly specific calculation cluster for the company requirements. The aim is to investigate whether it is possible to implement a suitable solution for improving calculation efficiency for compute-intensive calculations. In addition, the goal is to be able to use existing technologies in the company to find an alternative, cost-effective, and optimally efficient way to utilise the workstation cluster for distributed processing that is easy to use and facilitates functionalities that are simple to manage. The original aim of the investigation is to set up a distributed processing cluster using existing workstations that are used by company staff as calculation nodes and a separate dedicated server as a distributed processing controller. The tests and investigations have shown promising results that the company's compute-intensive applications can be modified accordingly to work with the bespoke-type distributed process-based calculation system. The test results confirm the expected calculation time improvements under a controlled environment. This has proved that it is feasible to design and develop a bespoke-type distributed processing system using the company's existing hardware and software to provide a solution for calculation time limitations problems that are currently faced by the company. Even though the test results show considerable improvements in calculation time for particular applications, in real-time trading applications, a number of complex issues have to be addressed; these issues are investigated, and possible suitable solutions and recommendations are discussed in detail in Chapter 7 and 8.

The tested distributed processing approach has proved that using a new and innovative bespoke-type development that is particularly suited for the company's hardware infrastructure and the applications used in the company is a better solution rather than following the standard distributed processing approach that requires specialised hardware and software configurations. The approach that is taken to design and build a loosely coupled distributed processing system that uses existing hardware and familiar software and programming environment has many advantages compared with highly specific integrated distributed processing that requires specialised hardware, software,

and programming environments. Hence, the bespoke-type loosely coupled designed is proved to be the most suitable solution for the company in its current situation, and this approach is used across all the calculation clusters implemented within the company that are described in detail in the forthcoming chapters.

The calculation time improvements using a bespoke-type distributed processing system are as expected, and the investigation was performed using a single MS-Excel application with linear type of datasets. However, most of the applications used in the company have nonlinear application structures and datasets, and in addition, various limitations and business regulation-related issues that need to be considered for the distributed processing system to be fully implemented using live trading scenarios with different cluster configurations. One of the methods of improving calculation time for nonlinear dataset applications is to use the suitable load balancing techniques within the distributed processed calculation cluster, and this is discussed in detail in Chapter 5. Even though, the tests conducted are under control environment, the simplified implementation of distributed processing system using existing hardware and software with appropriate modification has proved to be the most suitable and innovative solution for the company. In addition, it has demonstrated the possibilities of further improvements that can be made to the existing system that will considerably improve the calculation efficiency across all the business areas in the company.

5 Implementation of Adaptive and Self-Tuning Task Scheduler and Load Balancing

5.1 Introduction

This chapter describes the original design techniques and methods used for distributed process load balancing in the distributed processing system that includes scheduling, resources monitoring and management. This is the second phase of the investigation that explained in section 1.4 in Chapter 1. A number of techniques, methods, and algorithms have been researched and successfully implemented in various types of distributed parallel processing systems for the last 20 years. However, some techniques are highly specialised and unique to particular systems such as operating system-level hyper-threading with multiple-core CPUs and parallel threads. For Northwest's distributed processing system, the methods have to be simpler, bespoke-type algorithms using certain types of well-tested robust techniques. Some variations of existing load balancing techniques have been applied to ensure that the algorithms are simple to implement and easy to modify for different types of applications that are used within the distributed processing system. This is a new unique approach to solve the complex load balancing within the distributed applications.

The static and dynamic balancing techniques were tested on the workstation cluster with varying hardware- and software-specific parameters. For Northwest's distributed processing system, the methods have to be bespoke-type load balancing algorithms that include some specific application-related parameters as well as hardware- and software-related parameters. Some variations of static and dynamic load balancing are applied with task allocations algorithms that are capable of supporting different types of applications that are currently used. Due to the critical nature of the calculations involved and the high reliability requirements of the Northwest infrastructure, the static load balancing algorithms performed well compared with dynamic load balancing, which is mostly applied to protect from unpredictable events within the distributed processing cluster. The dynamic load balancing is highly suited for fine-grained processes, and coarse-grained processes are less suited for efficient dynamic

load balancing. Hence, for the Northwest systems, the static load balancing is the primary load balancing system due to its robustness and reliability, and the dynamic load balancing is used as a secondary load balancing mechanism to safeguard against calculation node failure during the execution phase. Detailed tests are performed for certain applications for comparing the performance of various methods and the techniques that can be applied to the system. These tests are performed to check different methods that include application-specific parameters to evaluate the best possible combination that can provide a better solution for the company's applications. The test results show the expected improvement in calculation times, and it is possible to implement an efficient static and dynamic load balancing for the company's applications with appropriate data and functional decomposition methods.

5.2 Load Balancing Software

The Northwest distributed processing system is based on bespoke design; hence the load balancing mechanism has to be a bespoke-type system. Consequently, the software that controls the load balancing mechanism is designed as part of the distributed processing controller software. Different types of schemes are implemented for load balancing and task allocation using the following methods:

- Appropriate control systems used to manage the load balancing systems.
- Current and historical load index measures are used.
- Local and global load balancing with hierarchical system organisations with local load information.
- Incorporation of a set of tools at the operating system level and implementation of different load balancing policies depending on the system architecture and application requirements.
- Hardware, software, and application-related parameters incorporation for the load balancing algorithms design.

The control systems used for managing the load balancing is explained in Appendix A. Figure 5.1 shows the distributed processing controller with incorporated load balancing.

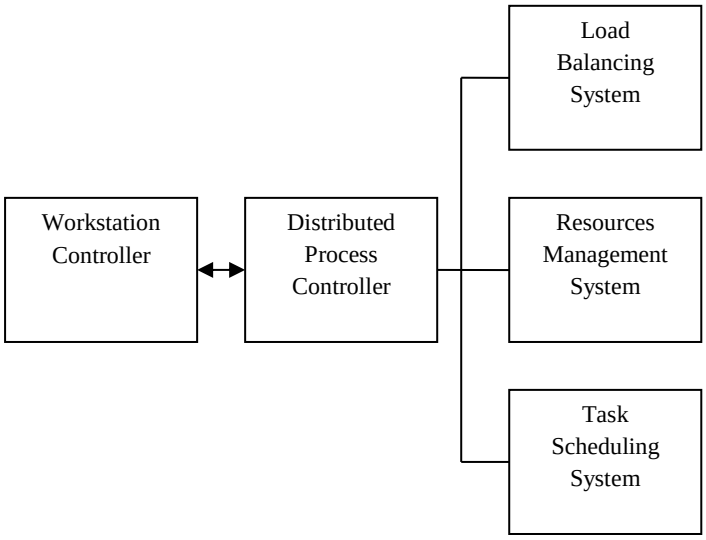


Figure 5.1: Distributed processing controller design with load balancer

5.3 Load Balancing Process

The load balancing process is managed by distributed process controller in coordination with load balancing system. Load balancing system allocates tasks to each calculation node based on defined algorithms. Two type of load balancing system is used, Static Load Balancing and Dynamic Load Balancing. In addition, two methods are used for task allocations, Data Decomposition and Process Decomposition. For dynamic load balancing, two extra calculations are added to each calculation cluster as auxiliary calculation nodes to protect against the calculation node failures. These auxiliary nodes are fully capable of replicating any of the nodes in case of failure of any of the cluster nodes. Static load balancing is used as primary load balancing and dynamic load balancing is used as a secondary load balancing. Each calculation node is loaded with calculation node controller that communicates with distributed process controller to update the status of the calculation node. The distributed process controller instructs the load balancing systems to allocate tasks to each calculation node based on various task allocation algorithms. How these techniques are implemented is discussed in detail in the forthcoming sections. Figure 5.2 illustrates the task allocation process to each calculation node.

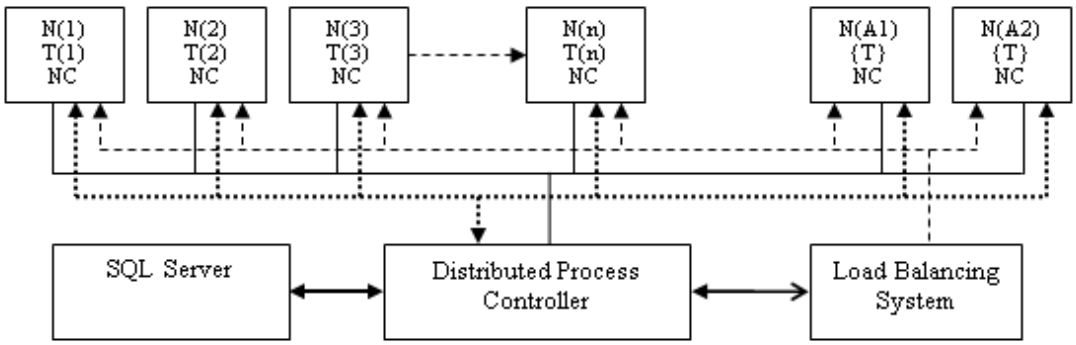


Figure 5.2: Task allocation process

where

n	Number of nodes in the calculation cluster
NC	Calculation node controller
N(i)	Calculation node i
T(i)	Task allocated to calculation node i
N(A1), N(A2)	Auxiliary calculation node 1 and 2

{T}=[*T(1), T(2),(T3)…… T(n-1),T(n)*]

5.4 Auxiliary Calculation Nodes

Two methods are implemented to protect against calculation node failures: the task relocation method that transfers the failed task to another available calculation node, and the auxiliary calculation node on standby that takes over the failed task. For dynamic load balancing and improvement of the robustness of the calculation cluster, two auxiliary calculations are introduced to the calculation cluster. The auxiliary node is loaded with all the process instructions and associated data for all the processing nodes; hence, if any of the calculation nodes fail, the auxiliary node will act as the failed node during the distributed processing execution phase with minimum delays. Hence, to avoid process transfer delays, two auxiliary workstations are placed on standby with all the processes and the required data sets loaded, and these workstations can replicate any of the calculation nodes during the execution phase.

The setup has the following advantages:

- Greatly reduces the complexity of rescheduling the tasks among the active calculations nodes.
- Improves the calculation efficiency and overall calculation time.
- Improves the reliability and robustness of the calculation cluster.

However, these workstations will only be used when an unexpected event happens during the execution phase, and if no disturbance during the processing, then these two workstations will be in a redundant state. The main purpose of the auxiliary node-based configuration rather than the rescheduling-based approach is that the reliability and the robustness of the system are far more important than the cost of maintaining two extra workstations within the cluster. For testing how the context switching algorithms transfer the process to the auxiliary calculation node from a failed calculation node, three types of simulations are used, as follows:

- Using the calculation node controller to send a message to the distributed processing controller notifying that the local process has exceeded the lapsed time limit.
- Shutdown one of the calculation nodes randomly by using a VBScript application; this will trigger a calculation node failed event process by the distributed process management controller.
- Using a VBScript application in the calculation node to simulate the high CPU and memory usage and, in effect, slow down all the processes running in the calculation node. This will cause the allocated task to take longer to complete and will trigger a process time lapsed event. The process time lapsed event is captured by the calculation node controller and a message is sent to the distributed processing controller notifying that the local process has exceeded the lapsed time limit.

Once the distributed processing controller has received a message from the calculation node controller, it activates the dynamic load balancing mechanism that will activate the calculation node’s local task in the auxiliary calculation node.

5.4.1 Process and Dataset Mapping

Every calculation task has an associated dataset mapped by the distributed processing controller as part of the static load balancing process, and the mapping may not be changed during the execution phase. Table 5.1 lists each calculation node's data and process mapping, and Table 5.2 lists auxiliary node data and process mapping.

Calculation node set: $N = \{N_1, N_2, ...N_n\}$

Process set: $P = \{P_1, P_2, ...P_n\}$

Data set: $D = \{D_1, D_2, ...D_n\}$

Table 5.1: Process and data mapping for each calculation node

Node	Process	Data Set
N_1	P_1	D_1
N_2	P_2	D_2
N_3	P_3	D_3
.	.	.
.	.	.
.	.	.
$N_{(n-1)}$	$P_{(n-1)}$	$D_{(n-1)}$
Nn	Pn	Dn

Table 5.2: Auxiliary calculation node process and data mapping

Node	Process	Data Set
N_a	P_1	D_1
N_a	P_2	D_2
N_a	P_3	D_3
N_a	.	.
N_a	.	.
N_a	.	.
N_a	$P_{(n-1)}$	$D_{(n-1)}$
N_a	Pn	Dn

For each calculation node: $N_i \rightarrow P_i, D_i$

For auxiliary calculation node: $N_a \rightarrow \{P_1, P_2, ...P_n\}, \{D_1, D_2, ...D_n\}$

For example, if calculation node *m* failed, then the distributed processing controller sends a message to the auxiliary calculation node to activate the process *m*. During this process, the node *m* will be redundant and will not be used for any calculation during the entire batch processing. The time delays caused by a process failure within a single calculation node for a given batch of calculations is considerably small compared with the overall calculation time for the batch concerned. However, if the process failure is more frequent and process rescheduling is activated during the execution phase by the dynamic load balancing algorithms, then the calculation time delays become more complex and unpredictable. Due the complex and critical nature of calculations used within the calculation cluster, the unpredictable behaviour has to be eliminated as much as possible, and the expected calculation time delays during the operation should be able to be predicted. Hence, the cluster hardware and software have to be kept in good working order and maintained routinely to ensure that the failure rate is minimal. Figure 5.3 shows the auxiliary nodes process and data mapping, and Figure 5.4 shows the calculation nodes process and data mapping to a single auxiliary node.

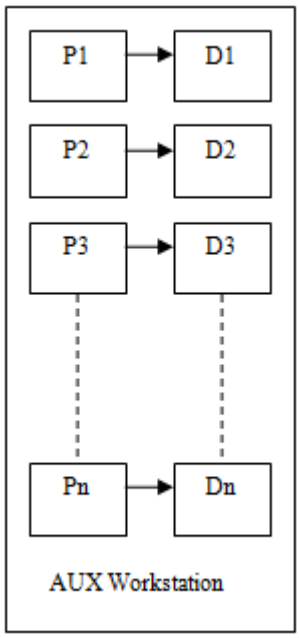


Figure 5.3: Auxiliary nodes process and data mapping

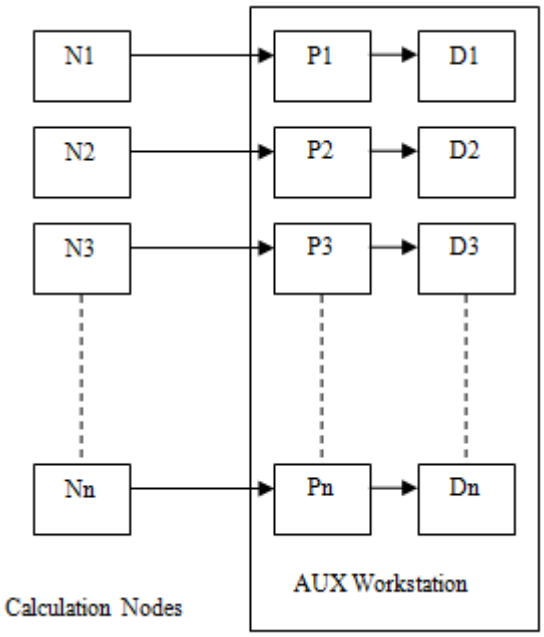


Figure 5.4: Calculation nodes process and data mapping to a single auxiliary node

5.5 Memory Use in Each Calculation Node

The memory usage of the each calculation node is monitored using WMI API in a predefined interval by the distributed processing control management software. The interval is set to every 10 minutes, and the collected memory usage data is saved in the cluster controller's management database. The collected data is used to analyse the memory usage profile for each calculation node to monitor the performance and the calculation efficiency of each node. As expected, three months average memory usage and randomly selected working day average memory usage have similar profiles, as shown in Figure 5.5. This is due to certain users employing particular applications as part of their job function and these applications consuming a certain amount of memory. However, certain users such as analysts and traders use many types of memory-consuming applications that consume larger amounts of memory compared with a typical user. Hence, certain workstations always use high amounts of memory, and other workstations use smaller amount of memory. The distributed processing management software has to monitor to ensure that the calculation node has enough

free memory for processing; otherwise, the workstation concerned should be excluded from the calculation cluster for particular batch processing. Figure 5.5 shows a typical memory usage profile for calculation nodes, and Figure 5.6 shows the typical memory usage on a working day on a single workstation.

The applications used in the calculation cluster are more CPU-intensive than memory-intensive; hence, the memory capacity has less impact than CPU speed. All the workstations used within the company have minimum memory of 12 GB and highly utilised workstations have 32 GB memory; hence, the available memory is always high enough for the program requirements within the distributed batch processing. However, due to the operating system's and software's utilisation of memory, this may cause memory overload to fill all the available memory, and this will result in the workstation freezing for a period of time. These types of scenarios have to be avoided to ensure that each workstation within the calculation cluster has enough memory to process a given task without memory bottlenecks.

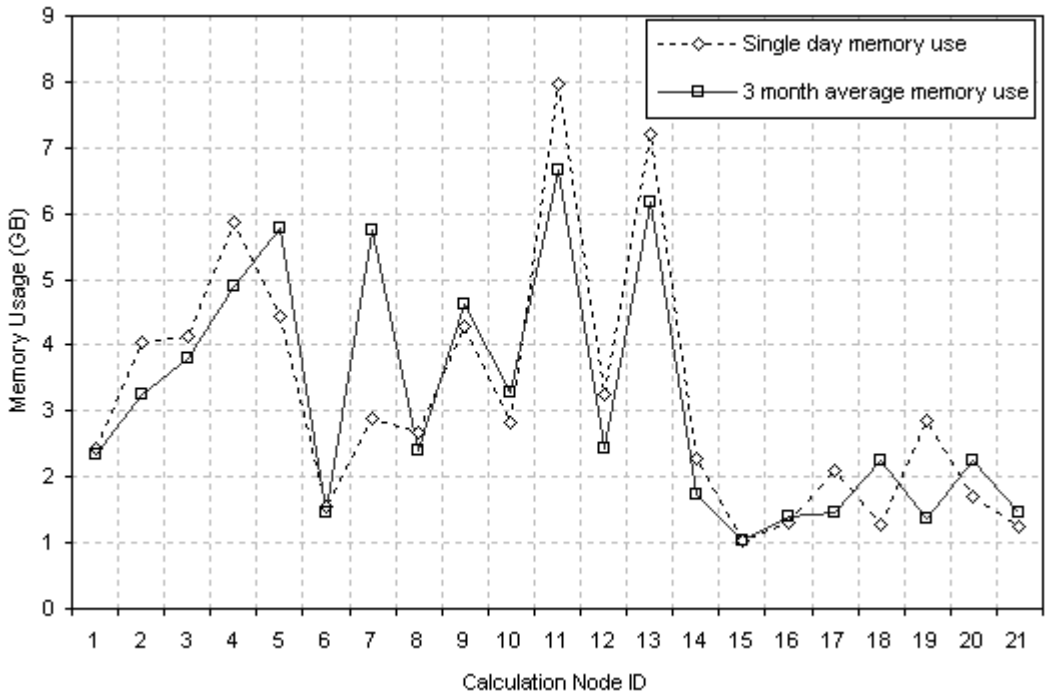


Figure 5.5: Memory usage profile for calculation nodes

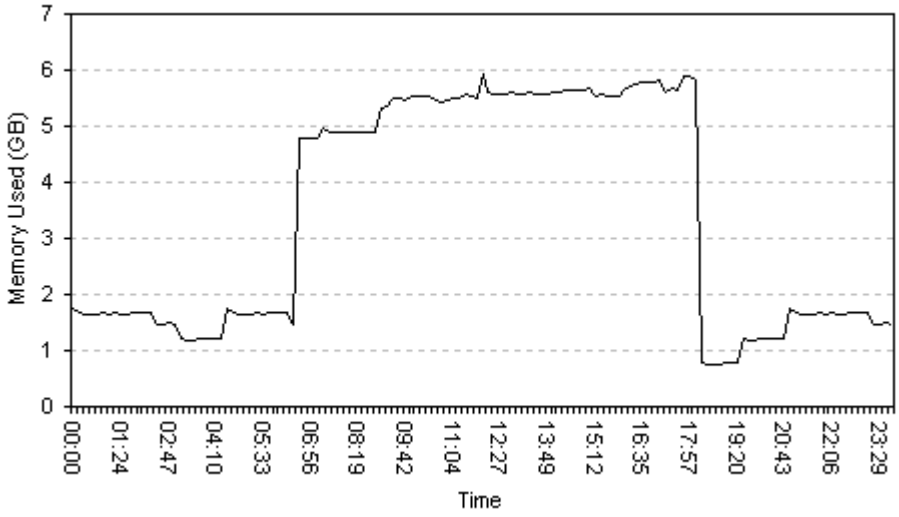


Figure 5.6: Typical memory usage on a working day (Node ID: NHKWST62)

5.6 CPU Use in Each Calculation Node

All the workstations used in the company have various profiles of CPU usage depending on how these workstations are used. In most cases, a majority of the workstations are highly utilised during the working hours, and these workstations are employed by the users in various capacities. However, certain CPU usage profile patterns are observed during the working hours, and these patterns are used for static load balancing algorithms. In addition, they are used for forecasting calculation time profiles using fuzzy logic rule-based methodologies. The CPU usage data is collected by the distributed processing management controller using WMI API at predefined intervals of every 10 minutes, and the collected data is stored in the cluster control's manager database for historical analysis and calculating weighted average CPU usage index. The distributed processing controller will allocate tasks accordingly or exclude workstations from the calculation cluster depending on the historical CPU usage index and current CPU usage index. Some of the workstations are permanently excluded from taking part in the calculation cluster due to their heavy usage during working hours, and these workstations are used for highly critical operations. However, these workstations are part of the calculation cluster when they are not employed by the users, such as during non-working hours.

Few workstations are used at high-level utilisation with high CPU usage, and these workstations are employed by analysts, traders, and fund managers for highly critical day-to-day operations. Hence, these workstations are most likely to be excluded from the calculation cluster nodes during the working hours by the distributed processing management software. The distributed processing management software monitors these workstations continuously to decide whether to include or exclude selected workstations depending on the workstation's CPU usage index thresholds values. Figure 5.7 shows the average CPU usage profile of the cluster, and Figure 5.8 shows the average CPU usage profile of heavily used workstations. Some workstations are used at a minimum level with considerably low CPU usage, and these workstations can be utilised within a distributed processing cluster to achieve maximum processing power for a given batch of tasks. The distributed processing control manager software monitors these workstations continuously to allocate more tasks to these workstations depending on current and historical CPU usage index. Figure 5.9 shows a typical CPU usage profile of a lightly used workstation.

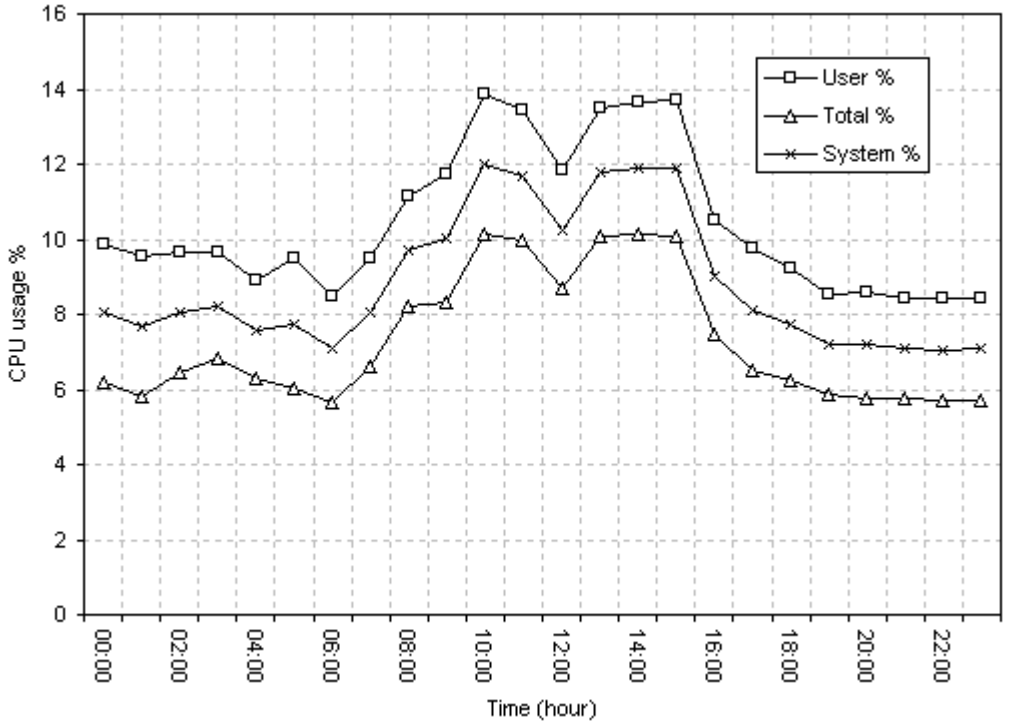


Figure 5.7: Average CPU usage of the calculation cluster

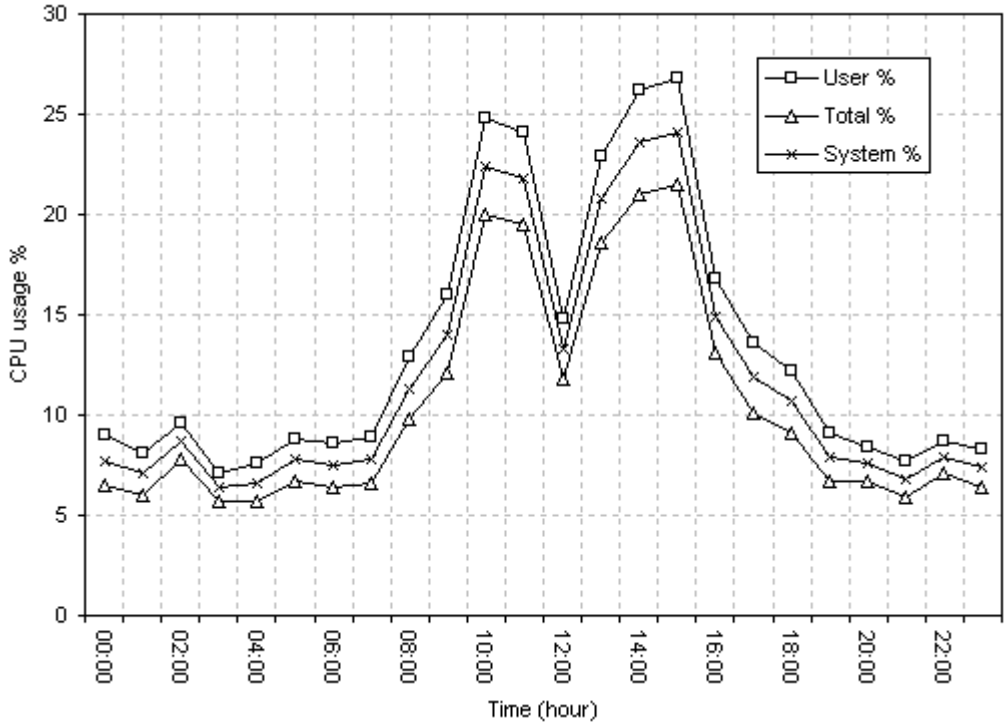


Figure 5.8: Average CPU usage of heavily used workstation

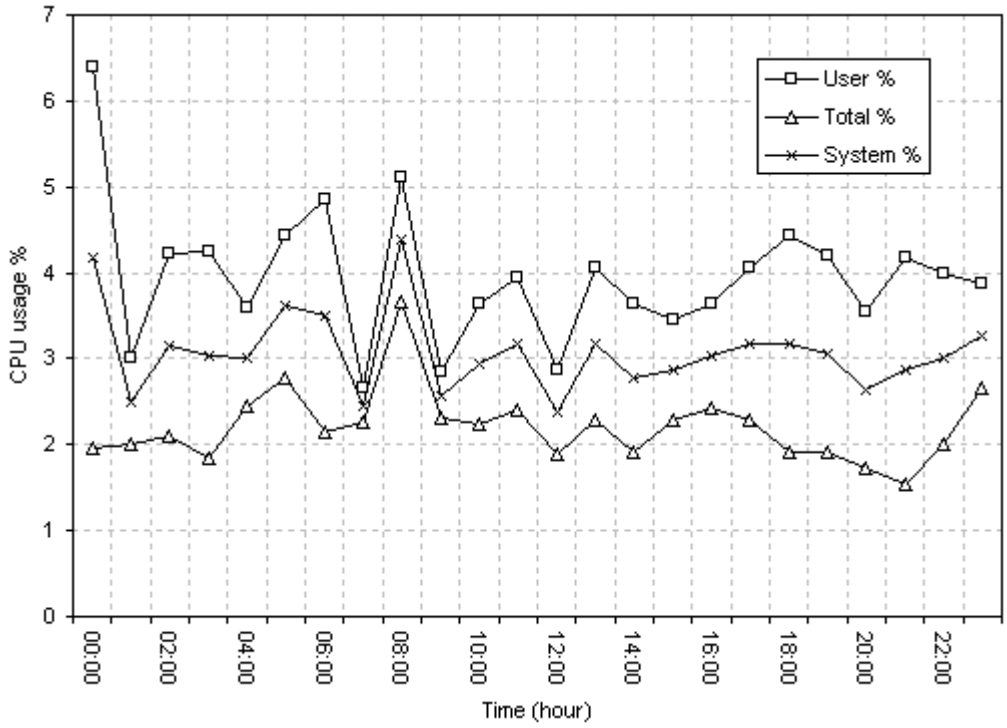


Figure 5.9: Average CPU usage of lightly used workstation

5.7 Static Load Balancing Implementation

The static load balancing techniques were tested on the Northwest system with various hardware and software configurations using the calculation cluster's hardware- and software-related parameters and including application-related parameters. For testing purposes, the Northwest risk calculation system is used with different portfolios that have varying calculation times. The calculation cluster-related hardware and software parameters are generic and are calculated using CPU speeds, memory availability, and workstation usage-related parameters. The application-related parameters are specific to company-related applications. Hence, the formulation of static load balancing algorithms has to incorporate these parameters to facilitate the cluster controller to allocate calculation tasks accordingly to each calculation node to ensure that the calculation cluster operates at load-balanced condition based on the calculation node's performance index, usage index, and application-related parameters.

5.7.1 Calculation Node's Performance Index

The calculation node's performance index is evaluated for each workstation within the cluster using a calibrating application to measure the time taken to complete the particular batch of calculations. The index mainly depends on the workstation model, memory size, and CPU speed. The index is measured when the workstation is in an idle state to ensure that each calculation node is measured at a similar calibration point.

5.7.2 Calculation Node's Usage Index

The node's usage index is measured by collecting data from each calculation node at a predefined interval, and the index is calculated using collected CPU and memory usage data for each calculation node. The usage index has two values for each calculation node: weighted average usage number and duration-based usage number.

The weighted average usage number is a single number for each calculation node; in contrast, the duration-based usage number is based on time duration usage of CPU and memory, and this can involve daily or hourly intervals. Hence, the availability index continuously changes for each calculation node depending on various factors such as how often the workstation is employed by the users and what type of applications are used. Due to the varying nature of this index, a user workstation may have a weighted average index of 3 (30% usage); during the peak hours, the duration based index may be 8 (80% usage); and during out-of-office hours it will be 0 (0% use). Hence, the weighted average index is an indicator to include or exclude a particular workstation from the calculation cluster for task allocations. However, the time duration-based values are used for allocating tasks to each calculation node depending on the time of the day that the batch process is executed.

5.7.3 Application-Related Parameter

The application-related parameter is based on what type of application is used in the calculation cluster. In certain applications, no need for the application parameter to be included in the static load balancing models. However, for the risk calculation application that is used for testing the static load balancing algorithms, an application-related parameter called the binomial tree-node number (ϕ) that is associated with each CB's theoretical value calculation using the binomial tree model. The value of ϕ depends on each CB's maturity date and the theoretical value calculation date; hence, the CBs that have same maturity dates will have the same ϕ values, and CBs with different maturity dates will have different ϕ values. Therefore, a portfolio that has multiple CBs with different maturity dates will have varying ϕ values, and this causes load imbalance in the number of calculations needed to complete each CB's theoretical value calculation. By implementing, each CB's ϕ values in the static load balancing algorithms reduce the calculation time imbalance between the calculation nodes. The binomial tree-node number (ϕ) is calculated using the CB model that described in section 4.2.1.

The ϕ is a benchmarking number that measures the number of tree-nodes required per derivative to complete the theoretical value calculation. The application parameter is an application-specific number that is related to each financial model calculation; for example, the binominal method and the trinomial method for the same derivative have different values. The tree-node number ϕ is calculated using reference models to find the number of calculations needed for each security. This number will vary daily and depends on a number of factors, and the tree-node number is used as part of the task allocation and load balancing algorithms.

5.7.4 Binominal Tree-Node Number

The binomial tree-node number is an application-specific parameter that is related to CB theoretical value calculation using binomial tree model. The number depends on what type of mathematical model is used for calculating the theoretical value for each CB, and the value is equal to the tree-node points needed for a selected CB to derive a theoretical value at a given calculation date. At each tree-node point, a number of calculations are performed, and each tree-node output will decide the next tree-node data inputs, and so on. Between each tree-node is called a step size. Figure 5.10 shows the tree-node for the binomial method with two steps.

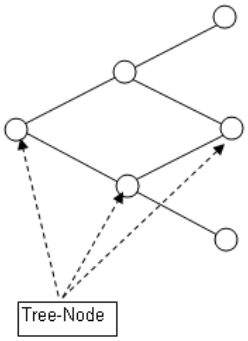


Figure 5.10: Tree-node in the binomial tree method

The tree-node value depends on specific parameters; however, for testing purposes, the following parameters are used: CB's maturity date, theoretical value calculation date, and binomial tree model. The binomial tree-based CB pricing model and the tree-node number calculation are shown in Equations (5.1) and (5.2), respectively.

$$\{J:1\rightarrow N+1\}+\{I:N\rightarrow 1\{2\times\{J:1\rightarrow I\}\} \tag{5.1}$$

$$\phi=\sum_{j=1}^{N+1}x(j)+\sum_{i=N}^{1}\sum_{j=1}^{i}x(j) \tag{5.2}$$

where

N Number of steps

x(j) Calculation function at binomial tree-node *j*

i and *j* are arbitrary variables

The correlation between step size and the total number of calculation nodes is exponential; hence, a nonlinear increase in calculation time as the CB's maturity date is further from the calculation date. Table 5.3 and Figure 5.11 show the correlation between step size and tree-node number. This is a simple example that demonstrates that varying the maturity date can cause a calculation imbalance within the cluster.

Table 5.3: Correlation between step size and tree-node number

Step size (N)	**Tree-node number (ϕ)**
100	10,201
200	40,401
300	90,601
400	160,801
500	251,001
600	361,201
700	491,401
800	641,601
900	811,801
1000	1,002,001

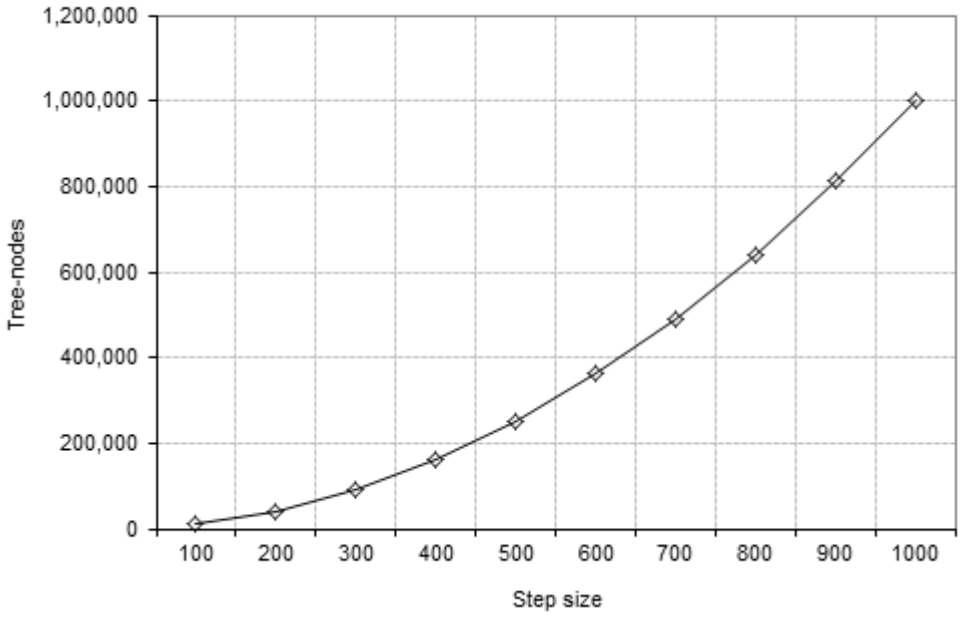


Figure 5.11: Correlation between step size and tree-node

For a given portfolio, the CB maturity date for each CB can vary from a few months to many years. This will cause calculation time discrepancies for a selected calculation date, and by implementing the tree-node number within the load balancing algorithms, the CBs can be distributed accordingly to complete the calculations at the same or similar time intervals. The distributed processing controller keeps track of each CB's tree-node number using static data from the portfolio database and distributed processing management database. For testing the nonlinear nature of tree-nodes within a portfolio, a test portfolio selected with 427 CBs, which has varying maturity dates. This portfolio is used as an input dataset to test the load balancing algorithms that described in detail in the forthcoming chapters. The CB parameter calculations and tree-node implementation methods that are used for load balancing tests in this chapter is described in detail in section 4.21 in Chapter 4. Table 5.4 lists the selected portfolio's maturity date distribution.

Table 5.4: Tree-node distribution within a selected CB portfolio

Years to maturity	Number of CBs	Total number of tree-nodes (ϕ)	CBs	Tree nodes
<=1	74	5,118,666	17.33%	0.55%
1 > and <=2	33	12,347,316	7.73%	1.32%
2 > and <=3	35	31,356,539	8.20%	3.34%
3 > and <=4	72	125,001,756	16.86%	13.33%
4 > and <=5	190	527,338,291	44.50%	56.24%
5 > and <=6	5	21,259,220	1.17%	2.27%
6 > and <=7	3	19,355,400	0.70%	2.06%
7 > and <=8	2	16,808,402	0.47%	1.79%
8 > and <=9	0	-	0.00%	0.00%
9 > and <=10	11	126,228,776	2.58%	13.46%
10>=	2	52,900,898	0.47%	5.64%

5.7.5 Fixed Step Size Calculation

In the fixed step size calculations method, each CB theoretical value calculation is similar regardless of its maturity date. This is due to how the binomial tree is constructed during the calculation, and the number of tree-nodes needed to derive the CB price for each CB mainly depends on step size. However, for highly complex CBs like Chinese CBs, the number of calculation nodes required depends on various factors and also depends on certain events such as call date that considerably affect the CB theoretical values. Equation (5.3) shows the calculation of the step size:

$$d = \frac{M - T}{N} \tag{5.3}$$

where

- d Number of days in a single step
- N Number of steps
- M Maturity date
- T Calculation date

Therefore, regardless of the maturity date of each CB, the fixed number of steps calculation method will adjust the stepping days to match the step size. In this method, calculation time can be reduced considerably and can be set to calculate using a linear method by employing batch processing. However, this method has some serious flows as far as financial mathematics is concerned. For more accurate theoretical value calculations, the calculation step size must be high enough, and the theoretical value accuracy increases with increase in the calculation step size. However, in a certain threshold point at which no improvement to the theoretical value accuracy occurs with increasing step size; that is, where the calculation step size forces the days in a single step to be less than 1 day. For example, if a CB has 1,000 days to maturity from the calculation date, and the calculation step size used is 100, then each step size is 10 days per step; that is, every period of 10 days is treated as a single logical day for calculation purposes. This will reduce the accuracy of calculation due to certain events that can happen during the 10-day period, such as put date, call date, or similar events, and these events can cause a considerable shift in each CB's theoretical value calculation. Therefore, to capture these events, the step size must be one day or less. Hence, the most mathematically suitable step size is one day per calculation step. Using one day per calculation step will produce the most accurate theoretical value calculation for a given model.

Currently, the company uses a fixed number of step sizes because of calculation time constraints, and usually, the step size of 100 is specified for all the CBs. This method has an advantage of reducing the calculation time for the entire portfolio. Hence, for fixed step size calculations, using the static load balancing technique is adequate to produce a desired calculation time reduction using the workstation cluster. For fixed step size calculations, the CBs are equally divided between calculation nodes using the average-based pro-rata method. Figure 5.12 shows the fixed step-size calculation method.

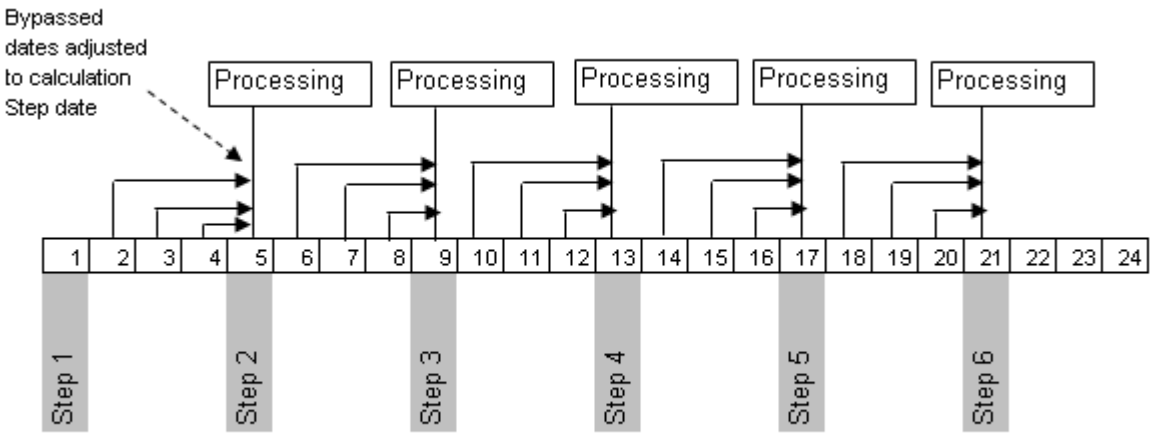


Figure 5.12: Fixed step-size calculation processing bypassing intermediate dates

5.7.6 Variable Step Size Calculation

In the variable step size calculations method, each CB has its own calculation step size to calculate the theoretical value based on the CB's different parameters. Hence, to complete the full calculation, each CB has to go through each calculation step to complete the calculation, and the time taken to complete the calculation depends on the CB's parameters for simply structured CBs. However, for complex-structured CBs, the number of steps depends on several factors such as underlying share price and other parameters. For testing purposes, Japanese CBs are used due to their simpler structure and ease to set up testing platforms to evaluate and compare the results using different static and dynamic load balancing techniques.

5.7.7 Maturity Date-Based Step Size Calculation

The simplest form of variable step size calculation is using the CB's maturity date and calculation date to evaluate the number of steps needed to calculate the CB's theoretical value. The difference between the maturity date and the calculation date is calculated as the number of days to maturity, and this is the number that is used to calculate the number of calculation nodes required to derive the CB's theoretical value for the selected calculation date. The calculation of number of days to maturity is shown in Equation (5.4):

$$D = INT(M - T) \tag{5.4}$$

where

D	Number of days to maturity
M	Maturity date
T	Calculation date
INT	Integer value round-down to floor

Hence, to make the financial models use a single day as each calculation step, the number of days to maturity is set equal to the calculation step size. The total calculation step size is calculated as shown in Equation (5.5):

$$N = D \tag{5.5}$$

where N is the calculation step size.

By substituting N in Equation (5.3), the value of *d* becomes equal to 1. Hence, to keep the single calculation step size equal to one day, the total step size will change depending on the calculation date and maturity date of the CB concerned. This will introduce complex nonlinearities for the selected portfolio that has many CBs with different maturity dates compared with the fixed step size method that is fixed for all the CBs regardless of their maturity dates, as explained earlier. Hence, the static load-balancing controller is designed to incorporate the variable steps in the form of an application parameter, that is, the binomial tree-node number (ϕ).

5.7.8 Hybrid Method

The hybrid method is a way of reducing calculation time by limiting the tree-node number (ϕ) for CBs with long maturity date. The following method is applied to calculate the number of tree-nodes required to calculate the theoretical value per CB. Using Equations (5.6) and (5.7), the hybrid method rule is set as follows.

$$N = D \tag{5.6}$$

$$D = M - T \tag{5.7}$$

having

If M > Y then N = (Y - T)

If M <= Y then N= D

where

N	Number of steps
D	Number of days to maturity
M	Maturity date
T	Calculation date
Y	Maturity date limit

The justification for this method is that the behaviour of CBs with long maturity date is different from that of CBs with short maturity date, and the impact of limiting the calculation step size is minimal for these CBs. Hence, using the hybrid method greatly reduces calculation time compared with the variable step size method while keeps the calculation accuracy to acceptable level.

5.8 Static Load Balancing Techniques Used in Northwest System

To test the static load balancing algorithms, the risk calculation system is used. This system uses the domain decomposition method to distribute data across cluster calculation nodes and uses coarse granularity for task division. Hence, more processing power is allocated for executing task calculations than for communication between the calculation node and controller. This method has the advantage of utilising maximum processing power for a given task; however, it has some disadvantage in applying particular load balancing algorithms commonly used in distributed processing clusters such as round-robin, first in first out (FIFO), and dynamic task allocations or similar. Therefore, the load balancing algorithms have to be designed in a certain way that is suitable for the particular scenario and application used in the calculation cluster. The generic form of a bespoke-type load balancing algorithm for a given scenario can be expressed as in Equation (5.8):

$$L = f(C, U, A, N) \tag{5.8}$$

where

- L Load balancing factor
- C Node's calculation speed-related load balancing parameter
- U Node's usage-related load balancing parameter
- A Application-related load balancing parameter
- N Number of calculation nodes within the selected cluster

The calculation performance index is a benchmarking number that measures the calculation speed of each calculation node for a given calculation scenario. The calculation index is evaluated when a workstation is in the idle state using a calibration program. Hence, the index value is measured to compare the speed of a particular workstation with other workstations within the cluster in an idle state using local data. Faster machines have higher calculation index values, and slower machines have lower index values. The calculation index can be calibrated in periodic intervals. For each workstation, the index is determined based on the following factors:

- Model of the workstation used as calculation node.
- CPU speed and number of cores in each CPU.
- Calculation node's memory size.
- Historical calculation time.

For the risk scenario system, the calculation index is formulated using Equation (5.9):

$$c = \frac{n}{t} \tag{5.9}$$

where

c	Calculation index
n	Number of tree-nodes
t	Time required for the calculation to complete

The usage index is a benchmarking number that measures the calculation node availability for a given period. The usage index is calculated using historical CPU and memory usage data per workstation. Hence, the usage index value is measured to compare the availability between each workstation for a given period. The most available workstations have a lower usage index, which is used as part of the task allocation and load balancing algorithms. Observation shows that CPU usage and memory usage in the user workstation are correlated, and this is due to the type of applications used. Hence, for evaluating usage index, both CPU usage data and memory usage data are employed. The usage index is calculated using Equation (5.10):

$$u_T = \frac{p + m}{2} \tag{5.10}$$

where

u_T	Usage index at interval T
p	CPU usage percentage averaged over the past 30 days
m	Memory usage percentage averaged over the past 30 days

At each binominal tree-node, a few calculations are performed; however, these calculations are common for each tree-node. Hence, the time required to calculate the functions of each tree-node is assumed equal. For example, the calculation time requirement for a five-year maturity CB is 25 times more than the one-year maturity CB. Hence, the tree-node calculation number is an important part of the load balancing algorithms, and the number of CBs allocated to each MS-Excel service is determined by the tree-node number (ϕ) in coordination with other appropriate load balancing parameters. Using the tree-node number, usage index, and calculation index, Equation (5.8) can be expressed as Equation (5.11):

$$L(k) = \left(\frac{1}{N}\right)\frac{u_T(k)}{C(k)}\sum_{i=1}^{m}\phi(i) \tag{5.11}$$

where

$L(k)$	Calculation node load balancing factor for node k
$C(k)$	Calculation index for node k
$u_T(k)$	Usage index at time range T for node k
$\phi(i)$	Tree-node number for security i
m	Number of securities in the batch
N	Number of calculation nodes in the cluster

Hence, the load balancing condition can be expressed as shown in Equation (5.12):

$$L(k) = L(k+1) \tag{5.12}$$

When the calculation is performed in a similar hardware-based workstation outside the office hours, the load balance factor becomes as expressed in Equation (5.13):

$$L(k) = \left(\frac{1}{N}\right)\sum_{i=1}^{m}\phi(i) \tag{5.13}$$

where

$$u_T(k) = u_T(k+1) = 1$$
$$C(k) = C(k+1) = 1$$

In practical implementations, the load balancing condition is shown in Equation (5.14):

$$L(k) = L(k+1) + e \tag{5.14}$$

where $\boldsymbol{e}$ is load balancing error.

The purpose of the load balancer is to make *e* become zero as in Equation (5.15):

$$e \to 0 \tag{5.15}$$

Each node's calculation time can be expressed as shown in Equation (5.16):

$$T(k) = C(k) \times \phi(k) \tag{5.16}$$

where

T(k)	Calculation time for node k
C(k)	Calculation index for node k
$\phi(k)$	Number of binomial tree-node number for node k

Hence, total calculation times for a given batch j of tasks is as shown in Equation (5.17):

$$T(j) = MAX\{T_1(j), T_2(j), ...T_N(j)\} + T_I(j) + T_F(j) \tag{5.17}$$

where

T_I	Initialisation time of batch j
T_F	Finalisation time of batch j
$T_k(j)$	calculation time of batch j at node k
N	Number of calculation nodes in the cluster
MAX	Maximum value of the dataset

Number of CBs allocated to each calculation node is determined using Equations (5.18), (5.19), (5.20) and (5.21):

$$P = \sum_{i=0}^{m} \phi_i \tag{5.18}$$

$$p(k) = P \times \left[\frac{\lambda(k) \times (1 - \beta(k))}{\sum_{i=1}^{N} \{\lambda(i) \times (1 - \beta(i))\}} \right] \tag{5.19}$$

$$\lambda(k) = \frac{C(k)}{MAX\{C(1), C(2), ...C(N)\}} \tag{5.20}$$

$$\beta(k) \ = MAX\{u_T(k), u_C(k)\} \tag{5.21}$$

where

$u_T(k)$	Historical time based usage index for node k
$u_C(k)$	Current usage index for node k
$C(k)$	Calculation index for node k
P	Total number of tree-node number within selected batch
$p(k)$	Number of calculations allocated to node k
m	Total number of CBs
N	Number of calculation nodes in the cluster
$\lambda(k)$	Rebased calculation index for node k
$\beta(k)$	Rebased usage index for node k
MAX	Maximum value of the dataset

The rebased calculation index values range from 0 to 1, where 1 means the most powerful workstation within the network or selected set, and 0 is assigned to the workstation not suitable for calculation. Because the workstations used in the Northwest network are similar models with similar configurations, the rebased calculation index ranges from 0.7 to 1.0. The rebased usage index value ranges from 0 to 1, where 1 indicates a fully used workstation that is usually not available for distributed processing within the network, and 0 is assigned to the workstation that is fully available for distributed processing. For the out-of-office-hour batch-processing scenario, all workstation usage indexes are set to zero; that is, all the workstations are fully available.

The total number of tree-nodes, P required for a selected batch is calculated using Equation (5.18), where ϕ is an application-related tree-node number for each CB and m is the total number of CBs in the selected calculation batch. The number of calculations tasks $p(k)$ allocated to node k is calculated using Equation (5.19), where $\lambda(k)$ is the rebased calculation index for node k and $\beta(k)$ is the rebased usage index for node k. The rebased calculation index $\lambda(k)$ for node k is a parameter that has values between 0 to 1 for calculation node k based on each cluster node's measured 30-day moving average calculation index within a selected cluster. The rebased calculation index $\lambda(k)$ is calculated using Equation (5.20). The rebased node usage index $\beta(k)$ for node k is a parameter that has values between 0 to 1 for calculation node k based on each cluster node's measured 30-day moving average usage index $u_T(k)$ and current usage index $u_C(k)$ within a selected cluster. The rebased node usage index $\beta(k)$ is calculated using Equation (5.21). If every calculation node within the selected cluster has the same hardware and software configurations, that is, the rebased calculation index for all calculation nodes is equal to 1 and the calculation nodes are fully dedicated, the rebased usage index for all calculation nodes is equal to 0. Hence, in this case, the load will be distributed equally to each calculation node and the number of calculations allocated to node k $(p(k))$ is equal to the total number of calculations P divided by the number of calculation nodes N.

5.8.1 Task Allocation Based on Fixed Step Size Method

In this method, the CBs are divided equally for each calculation node, regardless of the tree-node number (ϕ). Table 5.5 lists the calculation time per calculation node for the fixed step method. In this method, each calculation node completes its calculation simultaneously or within a similar timeframe if the hardware of the calculation node is similar. In this test case, step size is set to 100 for all the CBs in the batch. In this method, all 427 CBs in the test portfolio divided equally as possible and allocated to each calculation node. Hence, nine calculations node allocated 43 CBs and one calculation node allocated 40 CBs.

Table 5.5: Fixed step size tree-nodes and calculation time

Workstation Name	Number of CBs	Total Number of Tree-Nodes (ϕ)	Calculation Time (min)
NHKWST55	43	438,643	0.124
NHKWST57	43	438,643	0.124
NHKWST60	43	438,643	0.124
NHKWST61	43	438,643	0.124
NHKWST66	43	438,643	0.124
NHKWST75	43	438,643	0.124
NHKWST92	43	438,643	0.124
NHKWST93	40	408,040	0.115
NHKWST97	43	438,643	0.124
NHKWST99	43	438,643	0.124

5.8.2 Task Allocation Based on Variable Step Size Method

In this method, the CBs are divided equally for each calculation node, regardless of tree-node number (ϕ); however, the calculation step size is different for each CB depending on the CB's maturity date. Table 5.6 lists the calculation time per calculation node using the variable step size method. In this method, each calculation node has different calculation times that cause load imbalance within the system.

Table 5.6: Balanced CB-based allocation and calculation time for variable step size

Workstation Name	Number of CBs	Total Number of Tree-Nodes (ϕ)	Calculation Time (min)
NHKWST55	43	89,500,142	25.2
NHKWST57	43	16,726,252	4.8
NHKWST60	43	110,795,906	31.2
NHKWST61	43	135,947,170	38.4
NHKWST66	43	91,730,109	25.8
NHKWST75	43	88,325,001	25.0
NHKWST92	43	89,689,771	25.4
NHKWST93	40	115,970,501	32.8
NHKWST97	43	77,794,677	22.0
NHKWST99	43	121,235,735	34.2

5.8.3 Task Allocation Based on Balanced Tree-Node Method

In this method, the CBs are allocated to each calculation node using Equation (5.19) and based on the tree-node number (ϕ) per calculation node. Table 5.7 lists the balanced tree-node number and calculation time for the variable step size with balanced tree-node method. Hence, in this method, it is possible to achieve a load balanced condition that provides considerable improvement in calculation efficiency, and in addition, it improves the theoretical value calculation accuracy per CB. For imbalanced tree-nodes with variable step size calculation, the calculation index standard deviation is 9.18, and for the balanced tree-nodes with variable step size calculation, the calculation index standard deviation is 1.31. Therefore, the tree-node balanced method has better load balancing performance than the CB balanced method.

Table 5.7: Balanced tree-node-based allocation and calculation time for variable step size

Workstation Name	**Number of CBs**	**Total Number of Tree-Nodes (ϕ)**	**Calculation Time (min)**
NHKWST55	31	94,289,346	26.6
NHKWST57	82	93,782,160	26.4
NHKWST60	33	95,772,829	27.0
NHKWST61	28	96,715,936	27.2
NHKWST66	43	95,678,562	27.0
NHKWST75	49	95,772,458	27.0
NHKWST92	46	94,663,622	26.8
NHKWST93	26	81,068,088	22.8
NHKWST97	49	96,052,571	27.2
NHKWST99	40	93,919,692	26.4

To test the hybrid method performance that described in section 5.6.8, Equations (5.6) and (5.7) are used to allocate an appropriate number of calculation tasks to each calculation node. Table 5.8 lists the hybrid method CB allocations, and Table 5.9 lists the calculation time analysis for different methods used for load balancing and task allocations.

Table 5.8: Hybrid method CB allocation per calculation node, and time required to complete the calculations

Workstation Name	**Number of CBs**	**Total Number of Tree-Nodes (ϕ)**	**Calculation Time (min)**
NHKWST55	38	9,538,038	2.6
NHKWST57	65	9,523,025	2.6
NHKWST60	38	9,485,062	2.6
NHKWST61	39	9,345,519	2.6
NHKWST66	42	9,514,794	2.6
NHKWST75	48	9,472,377	2.6
NHKWST92	41	9,336,313	2.6
NHKWST93	32	8,032,032	2.2
NHKWST97	46	9,565,989	2.6
NHKWST99	38	9,538,038	2.6

Table 5.9: Distributed calculation time analysis for different calculation types

Calculation Type	**Maximum Time (min)**	**Minimum Time (min)**	**Average Time (min)**	**STDEV**
Fixed step size	0.124	0.115	0.123	0.003
Variable step size CB balanced	38.400	4.800	26.480	9.180
Variable step size Tree-node balanced	27.200	22.800	26.440	1.310
Hybrid	2.600	2.200	2.560	0.130

As shown in Figure 5.13, the hybrid method is able to reduce the calculation time considerably compared to CB-based or tree-node-based load-balancing methods. Even though fixed step size shows better calculation time performance, the method is an approximate method and is only suitable for certain type of derivative products.

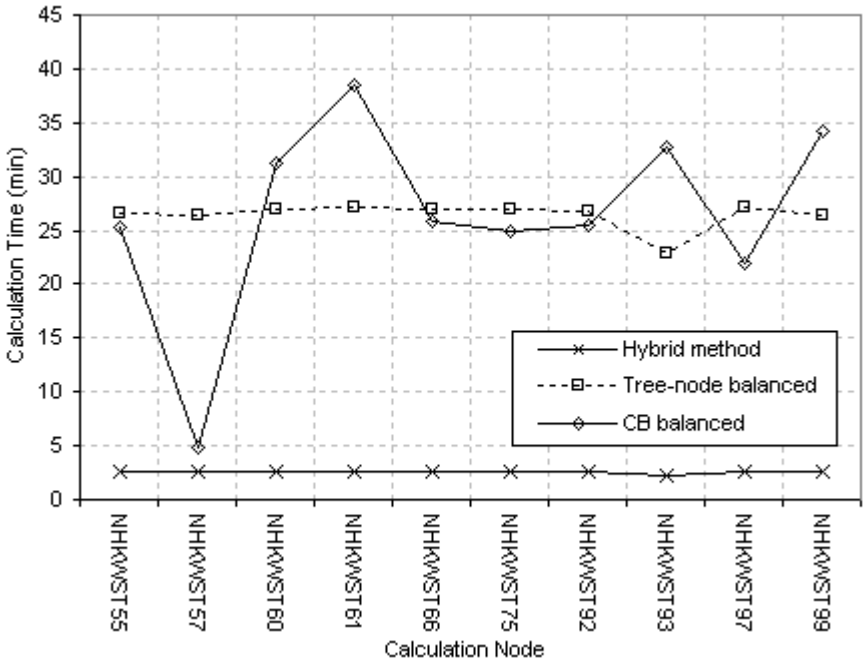


Figure 5.13: Calculation time per calculation node using hybrid, CB balanced, and tree-node number balanced methods

5.9 Dynamic Load Balancing Implementation

For testing dynamic load balancing techniques, most of the hardware, software, and applications setups are similar to the static load balancing tests. However, for testing the dynamic load balancing algorithms, some modifications are made to the hardware and software configurations. The application that used for testing the dynamic load balancing utilises the CB model that described in section 4.2.1 in Chapter 4. For testing, six calculation nodes are selected and allocated 10 CBs each; hence, 60 CBs are used to test the dynamic load balancing algorithms. Two techniques tested for implementing dynamic load balancing in the Northwest distributed processing system: process termination detection and context switching. The process termination detection and context switching algorithms are implemented using adaptive rule-based methods with task transfer and auxiliary processing. The dynamic load balancing is only activated after all the tasks are allocated to each calculation node using static load balancing algorithms and the distributed processing controller sends a message to each calculation node to start the calculation process. Hence, the dynamic load balancing is only active while the batch processing is in operation, and the distributed processing controller manages the dynamic load balancing within the calculation cluster while in the execution phase. The dynamic load balancing method uses an adaptive rule-based control, and these rules depend on the system configurations and application parameters; in addition, this is a logical set of rules and relatively easy to integrate with load balancing software. The rule set is generally constructed as follows:

$$[R]: \quad IF\ [X] = [A]\ THEN\ Y = [B]$$

where

R	Rule set
A	Input parameter set
B	Output parameter set
X	Input set
Y	Output set

Thus, *m* number of rules for *n* number of datasets is constructed as the following vector:

R(1): IF

X(1,1)=A(1,1), X(1,2)=A(1,2)….., X(1,n)=A(1, n)

THEN

Y(1,1)=B(1,1), Y(1,2)=B(1,2)….., Y(1,n)=B(1,n)

R(m): IF

X(m,1)=A(1,1), X(m,2)=A(1,2)….., X(m, n)=A(m, n)

THEN

Y(m,1)=B(m,1), Y(m,2)=B(m,2)….., Y(m, n)=B(m, n)

Linguistic-type rule sets are constructed as follows:

Rule 1: *IF Condition 1 TRUE THEN Activate x(1)*

Rule n: *IF Condition n TRUE THEN Activate x(n)*

Many rule sets *[R]* for different applications and cluster configurations depend on how they are implemented under varying conditions, in addition, the input and output parameters of each rule set are continuously modified by feedback data collection mechanisms. The adaptation rules are based on each calculation node's CPU, memory usage, and calculation performance for each application is used, and these parameters are captured in the distributed processing management SQL database for historical analysis. The captured historical data of each calculation node is used to construct the rule sets depending on the past performance of each calculation node under various conditions. To test the rule-based context-switched dynamic load balancing, six workstations are selected as a calculation cluster, and each workstation acts as a calculation node connected to the distributed processing controller. Each calculation node is given the same task to complete, and the time required to complete the given task is measured. Once the tasks are completed, then the task termination time limit is calculated using the measured calculation time. Using these data, the distributed processing controller allocates a task to each calculation node and monitors the performance using timer-based monitoring. If one of the calculation nodes takes more

than the allocated task termination time limit, then the distributed processing controller transfers the data and task to another available calculation node depending on the assigned set of rules. If more than one calculation node available, the distributed processing controller selects the most powerful calculation node within the available group and transfers the task to that node. To simulate the delay, a separate CPU and memory-intensive program is randomly executed while the task is running, and this will increase the calculation time on that particular calculation node. Eventually, the task termination time limit will lapse and the distributed processing controller will transfer the task and data to another available calculation node based on the established rules. If no calculation node failure, then the calculation time is calculated using Equation (5.22). If one of the calculation nodes fails, then the calculation time is formulated according to Equations (5.23), (5.24) and (5.25):

$$T_C = MAX\{T(1), T(2), ..., T(n)\} \tag{5.22}$$

$$T_N(i) = [T(i) \times K] + T_D + T_M \tag{5.23}$$

$$T_M = MIN\{T(1), T(2), T(j), T(j+1), ..., T(n)\} \tag{5.24}$$

$$j \neq x \tag{5.25}$$

where

T_C	Calculation time taken to complete the task
$T(i)$	Time taken to complete the task in calculation node i
T_D	Time taken to transfer task and data to calculation node
T_M	Minimum calculation time within given processing group
$T_N(i)$	Adjusted new time taken to complete the task in calculation node i
x	Failed calculation node
K	Task termination time multiplication factor
MAX	Maximum value of the dataset
MIN	Minimum value of the dataset

Table 5.10 lists measured values for distributed processing calculation times with and without calculation node failures using the task transfer method, and Table 5.11 lists the calculation time taken for different calculation scenarios. Figure 5.14 shows the minimum and maximum time taken for each failed calculation node. For testing, task termination time multiplication factor *K* is set 1.2, that is, if the particular calculation node crossover the time limit set, which is equal to (*T(i)* x *K*), the distributed process management controller will activate dynamic load balancing algorithms.

Table 5.10: Distributed calculation times with and without calculation node failure using the task transfer method

Calculation Node (i)	**T(i) (sec)**	**K**	**T(i) x K (sec)**	**T_D (sec)**	**T_M (sec)**	**T_N (sec)**
NHKWST68	46	1.2	55	2	36	93
NHKWST63	42	1.2	50	2	36	88
NHKWST66	39	1.2	47	2	36	85
NHKWST65	38	1.2	46	2	36	84
NHKWST62	36	1.2	43	2	36	81
NHKWST61	36	1.2	43	2	36	81

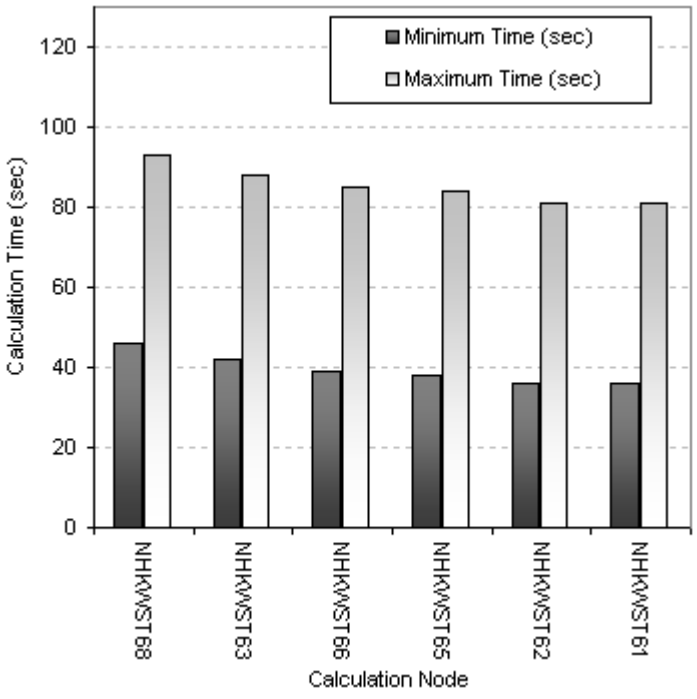


Figure 5.14: Minimum and maximum calculation times for failed calculation node

Table 5.11: Calculation time taken for different calculation scenarios

Calculation Condition	Calculation Time (sec)
Using a single most powerful workstation	216
Using a single least powerful workstation	276
Using cluster with no node failure	46
Using cluster with least powerful node failed	93
Using cluster with most powerful node failed	81

5.10 Auxiliary Processing

In this method, two ways of implementing the auxiliary calculations nodes: having one node as standby to take over if one of the nodes fails, and having replicated nodes for each calculation node executing in parallel. In the scenario of one node on standby, only one extra calculation node is needed, and if one of the main calculation nodes fails, then the standby calculation node takes over and completes the failed task. This will cause a certain time delay in the overall processing. In the replicated node scenario, exactly the same number of nodes is required, all the calculation nodes execute the tasks at the same time, and a single task is executed in two calculation nodes at the same time. By doing so, if one of the calculations has failed, it will not affect the overall calculation depending on how it is implemented. The main advantage of this method is that the overall failure rate is considerably lower than when running a single task on a single calculation node. However, for this method, the number of calculation nodes needed is doubled, for a one-to-two ratio; meanwhile, if the ratio is one-to-four, then the number of calculation nodes needed is four times more. This method is highly suited for the calculations that are a combination of time-critical and data-critical. For high reliability, a ratio of one-to-four is better suited, that is, the same task is executed in four separate calculation nodes at the same time. These types of implementations are investigated using CPU-core-level distributed processing, and they are discussed in detail in Chapter 7. Due to the availability of lower cost small form factor (SFF) computers, this method is highly feasible and relatively easy to implement compared with more complex dynamic load balancing algorithms.

For testing purposes, a single task executed in two separate calculation nodes at the same time. Six workstations are selected as calculation nodes, and the job is split into three parallel tasks that are executed at the same time. For example, if a task failure is 1 in 1,000 in a single calculation node, then, by executing the same task in two separate calculation nodes at the same time, the failure rate is reduced to 1 in 2 million for both calculation nodes that are executing same task. For reducing the calculation time, calculation nodes are paired as the most powerful with the least powerful calculation nodes within the selected cluster. Table 5.12 lists the calculation time for each calculation node using auxiliary processing, and Figure 5.15 shows the maximum and minimum calculation time for each group.

Table 5.12: Calculation time for auxiliary processing using two groups

Calculation Node(i)	T(i) (sec)	Task ID	Group ID
NHKWST68	92	1	1
NHKWST63	84	2	1
NHKWST70	78	3	1
NHKWST66	76	3	2
NHKWST65	72	2	2
NHKWST62	72	1	2

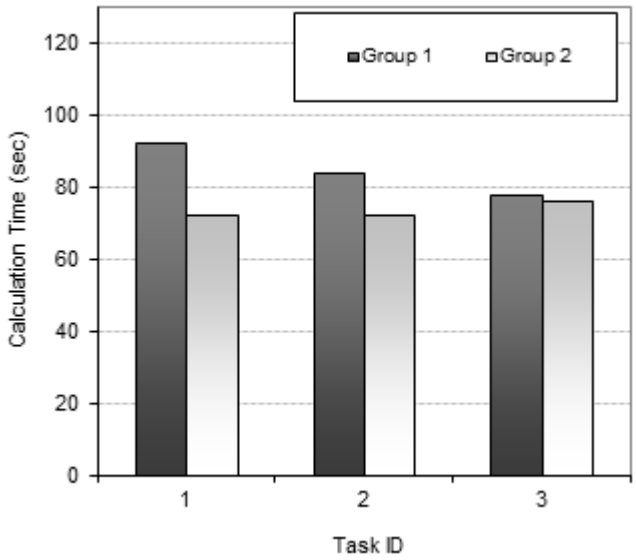


Figure 5.15: Minimum and maximum calculation time for each group and task

From the collected data, at the normal operating condition, that is, no calculation node failure, calculation time is 76 seconds; meanwhile, for the worst-case failure, assuming that two calculation nodes executing the same task that have not failed at same time, the calculation time is 92 seconds. Hence, the auxiliary processing method proved to be a very resilient and failproof method for data-critical and time-critical calculations.

For single standby auxiliary node testing, another workstation was selected to act as an auxiliary node with the calculation time of 29 seconds *(T_A)*. Hence, if any of the main calculation nodes failed, then it will take another 29 seconds to complete the task. In this case, using Equation (5.23), $T_D=0$, $T_M=T_A$, and $T_N= (T(i) \times K + T_A)$. Table 5.13 shows distributed calculation times with and without calculation node failure using the single auxiliary node method. As can be seen from the data, if no node failures, then the calculation completes in 46 seconds; meanwhile, if the most powerful node failed, then it will take 72 seconds to complete, and failure of the least powerful workstation requires 84 seconds to complete.

Table 5.13: Distributed calculation times with and without calculation node failure using the single auxiliary node method

Calculation Node (i)	**T(i) (sec)**	**K**	**T(i) x K (sec)**	**T_A (sec)**	**T_N (sec)**
NHKWST68	46	1.2	55	29	84
NHKWST63	42	1.2	50	29	79
NHKWST66	39	1.2	47	29	76
NHKWST65	38	1.2	46	29	75
NHKWST62	36	1.2	43	29	72
NHKWST61	36	1.2	43	29	72

5.11 Chapter Summary

This chapter has demonstrated the original contribution to the specific design methods and implementation of load balancing and task allocations techniques for complex applications. Due to the complex and bespoke nature of Northwest's software applications used in the distributed processing cluster, the load balancing techniques employed have to be highly specific for each application scenario. No single technique is suitable for all possible scenarios. Hence, the general load balancing algorithms widely used in known distributed system configurations are not suitable for Northwest's distributed processing system. Because of the critical nature of the calculations, the entire distributed processing has to be completed within the allocated time limits without any failures or has to operate with minimum failure rates. Therefore, the static load balancing is a crucial part of the overall load balancing implementations and must be able to predict the approximate total calculation time for given batch calculations; meanwhile, the dynamic load balancing acts as a safeguarding mechanism to improve the distributed processing efficiency.

The applications implemented within the distributed processing systems are mainly based on coarse-grained task distribution techniques, and coarse-grained processes are not suitable for efficient dynamic load balancing due to the longer processing time compared with the fine-grained processes that have shorter processing time. The fine-grained processes are highly suited for dynamic load balancing and perform well with different types of dynamic load balancing algorithms. Meanwhile, coarse-grained processes have very limited performance with dynamic load balancing; hence, very few dynamic load balancing techniques can be applied. Hence, in the case of Northwest, the use of a general type of dynamic load balancing has limited advantages compared with static load balancing. The company's network configurations and workstations are highly reliable and continuously monitored for hardware and software performance, hence, the failure rate of each calculation node is kept minimal. Therefore, the calculation time for each batch process within the selected workstation cluster can be predicted with a small margin of error and, in most cases, the static load balancing algorithms perform well without activating dynamic load balancing

algorithms. However, the dynamic load balancing acts as a safeguarding mechanism to protect the system when an unpredictable disturbance within the system. In addition, it is implemented as a part of the centralised load balancing mechanism and managed by the distributed processing controller, and is only activated by the controller if unpredicted events happen within the system.

Even though many techniques and algorithms can be applied for static and dynamic load balancing for different types of distributed processing systems, these significantly depend on what type of distributed processing systems are used and their implementations. Some of them are highly specific and others are generic in nature, and they depend on what type of applications, the scale of the distributed processing systems, and hardware and software used within the systems. As far as Northwest's distributed processing system is concerned, the load balancing techniques and algorithms have to be highly specific. It has to be simple to implement, and easy to manage with minimum disruption to the business. Because of the bespoke nature of the applications and software programs used within the distributed processing system, the load balancing algorithms have to be designed to work with the system concerned, and the fuzzy logic type rule-based algorithms are shown to perform effectively with these types of systems. The investigation on load balancing techniques has shown that a number of possible ways of achieving the fine-tuned load balanced condition within any type of distributed processing system to improve the processing efficiency of the calculation cluster. However, at the current stage, only a few of the bespoke techniques are suitable to implement within the company, and going forward, other techniques will be investigated as part of further research and development.

6 Dedicated Calculation Grid Design Using Peer-to-Peer Network for High-Volume Distributed Processing

6.1 Introduction

This chapter describes the original design, development, and testing of a dedicated distributed process calculator that acts as a calculation grid using SFF computers and PCs to build P2P network-based clusters with logically separated networks using Windows workgroups. This is the third phase of the investigation that explained in section 1.4 in Chapter 1. The purpose of building the dedicated calculation grid is to facilitate the research and development team and quantitative researchers to test and simulate various mathematical models and trading scenarios in real time and, in addition, to execute long-running batch processes as separate processes. These calculations require considerable amounts of processing power and cannot be implemented in a single PC or server that take many hours to complete. Hence, a requirement for a dedicated calculation grid that can be used for performing various calculations and can be employed 24x7. Unlike the workstation-based calculation cluster described in Chapter 4 that involves the user workstations, the dedicated calculation grid is isolated from user access and fully dedicated for certain applications and mathematical models that require extensive testing before being used in live-trading environments.

Two separate calculation clusters are built: one with small form factor (SFF) computers and the other with PCs. The PC cluster uses dedicated PCs as calculation nodes and these are spare PCs in good working conditions but not in use. The small form factor computer-based cluster is built using low-cost, single-board computers that are small in size and low in power consumption. Intel's NUC computers are used for this cluster. This is the main cluster that will be expanded by adding more calculation nodes in future developments. Meanwhile, the PC cluster that consists of spare PCs will only be used for testing purposes, and it will not be used in the production environment due to its high power consumption and requirement to be

located in a larger space. The distributed processing controller software and calculation node's controller software are modified accordingly to include a dedicated calculation grid cluster, and in addition, a number of changes are made to the SQL server database to incorporate the dedicated cluster nodes within the distributed processing control system. The message passing between calculation nodes and distributed processing controller is similar to the technique that is used and described in Chapter 4 with minor modifications to work with the P2P network.

A number of different tests are carried out on both clusters with and without load balancing conditions, and calculation time reduction using dedicated calculation clusters shows considerable improvements for compute-intensive calculations when used with distributed process-based calculation using both dedicated calculation grids. The NUC cluster is most suited for long-term development of dedicated calculation grids for the company due to its compact size, high reliability, lower power consumption, and considerably low cost per calculation node. Meanwhile, the PC cluster is most suited for testing load balancing algorithms and prototype mathematical models for distributed processing due to its nonlinearity. However, the PC cluster is not suitable for continuous operations due to its high power consumption, space requirements, noise, and unreliability of the hardware. Further analysis on collected data is discussed in detail in Chapter 8 where the results and discussions are presented.

6.2 Grid Calculation Nodes Configuration

The dedicated calculation cluster nodes are configured as a separate network grid using a P2P network configuration with logically separated clusters by using Windows workgroups. Two separate clusters are built; a NUC cluster that is built using Intel NUC computers and a PC cluster that is built using unused spare PCs and workstations in the company. All the calculation nodes are connected to the same gigabit switch, but logically grouped as two separate workgroups. Table 6.1 lists the network grouping for the PC and NUC clusters, Figure 6.1 shows each cluster node's configuration, and Figure 6.2 presents the logically separated P2P cluster configuration. Each dedicated calculation grid node's configuration procedure is similar to the workstation cluster configurations explained in Chapter 4. However, the grid nodes have different hardware and software, and primarily configured for executing MS-Excel VBA, VBS, CMD, and EXE applications. The NUC cluster calculation nodes are less powerful than workstations, and the workstations are optimised to calculate MS-Excel applications; however, the dedicated calculation grid can also be configured to execute MS-Excel applications if needed in the future with appropriate modifications.

Table 6.1: Northwest P2P network workgroups for PC and NUC clusters

Workgroup Name	Description
NWDP_WG_NUC	NUC cluster: 10 NUC PCs are connected as a cluster
NWDP_WG_WS	PC cluster: 11 PCs and workstations are connected as a cluster

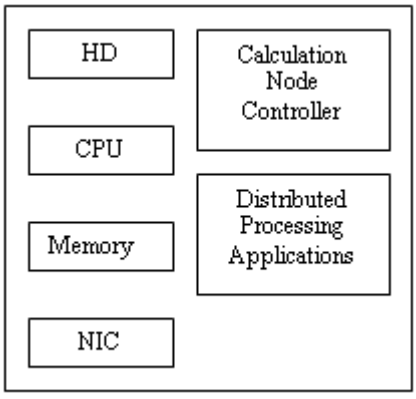


Figure 6.1: Configuration of each cluster node

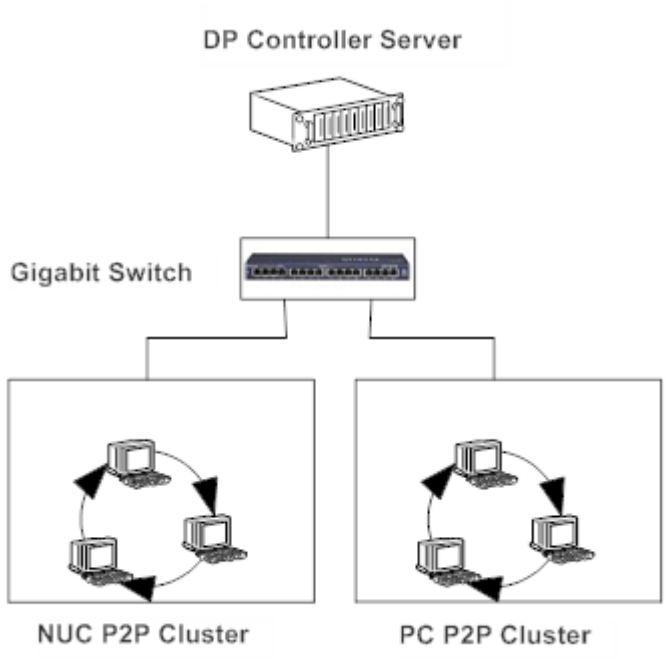


Figure 6.2: Northwest P2P logically separated workgroup cluster configuration

The main advantage of the dedicated calculation grid is that it is fully utilised for 24x7 operations and it is mainly used by a research and development team, and in addition, by quantitative developers for testing various scenarios and developing new trading strategies. However, the cluster consists of spare PCs and workstations used for testing purposes only and setup as a prototype cluster and it is highly useful for testing load balancing algorithms due to its non-linear hardware and software configurations. Due to its considerably high level of power consumption, bulkier size, and tendency to hardware and software failure, the PC cluster should not be used in the production environment. Hence, for future developments and enhancements of the dedicated calculation grid, development will be based on NUC-type small form factor computers due to their advantages mentioned earlier. Currently, the P2P network is physically separated from the main LAN for testing purposes, and in the future stages of the development, the dedicated calculation grid will be connected to the company's private LAN using two separate network interface cards (NIC) as shown in Figure 6.3. Figure 6.4 shows the NUC grid configuration diagram.

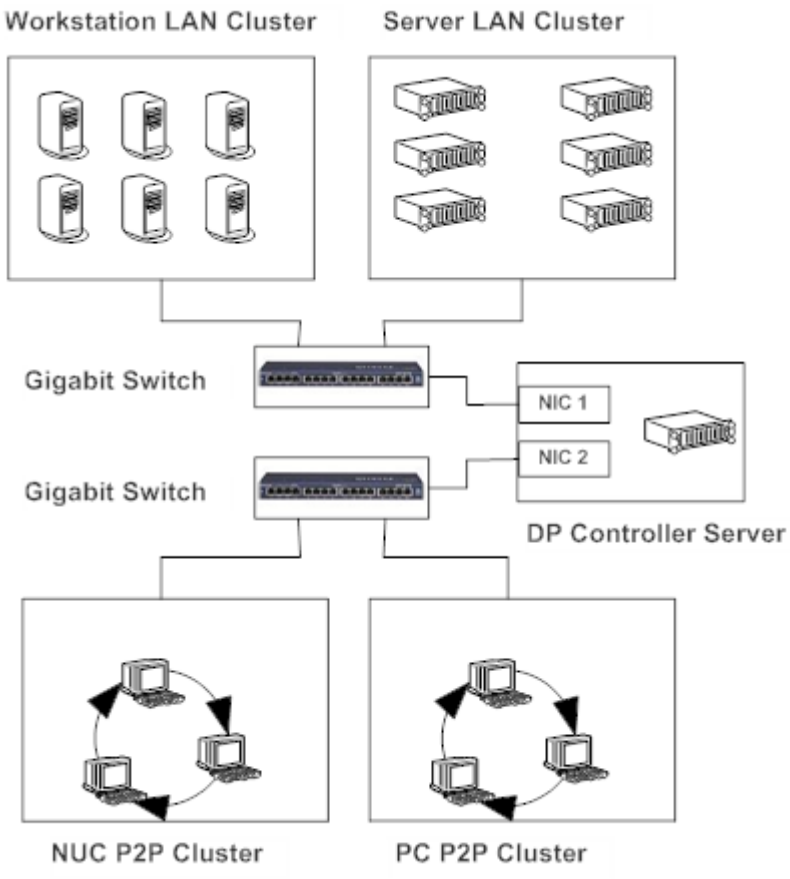


Figure 6.3: LAN and P2P workgroup combined cluster configuration

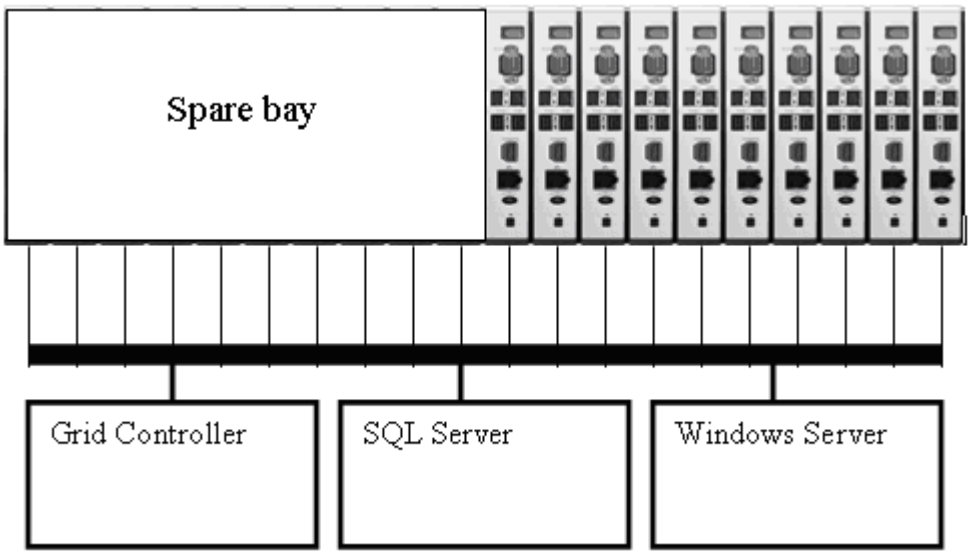


Figure 6.4: NUC grid configuration diagram

6.3 NUC Cluster

The NUC cluster consists of 10 NUC computers as calculation nodes and has the same hardware and software configuration as the other computers in the cluster. Each NUC computer is configured to act as a calculation node for the dedicated distributed processing cluster. The hardware and operating system for all the calculation nodes are the same, hence the distributed management controller only checks whether the particular calculation node is available before sending a message to execute the specific application. Due to the uniformity of the cluster hardware and software, hence, no need to check other parameters such as CPU usage and memory usage compared to the PC cluster. Table 6.2 lists the NUC computer calculation node's parameters. NUC cluster pictures are shown in Appendix C (Figure C.3-a, C.3-b).

Table 6.2: NUC computer hardware and software parameters

Parameter	Description
Model	DE3815TYKHE
CPU	1.46 GHz single-core Atom E3815
RAM	4GB SODIMM
Hard disk	30GB m-SATA
Operating system	Windows 7 (64)
Application	Microsoft Office Professional 2010
Power supply	35W

6.4 PC Cluster

The PC cluster is built using spare PCs and workstations currently not used by the company in daily basis. The company has a number of spare and unused PCs and workstations, and these are currently not used but in fully working condition with OEM software licenses. These PCs and workstations are refurbished and set as a separate workgroup cluster that is connected to the same gigabit switch but logically separated from the NUC cluster. These are the computers have been replaced with newer workstations and are in perfectly usable condition. In addition, these computers

are less efficient in various aspects, and their high power consumption makes these computers are unsuitable for long-term use as cluster nodes. Table 6.3 lists the PC cluster hardware configuration. For testing the load balancing algorithms, two newer models (Z400 and Z420) were added to the cluster. All the workstations and PCs are loaded with Windows 7 (sp1 64Bit) and Microsoft Office 2010. PC cluster picture is shown in Appendix C (Figure C.3-c)

Table 6.3: PC cluster hardware parameters

PC Name	PC Model	CPU speed	CPU Type	Number of Core	RAM	HD	Power Supply
WST35	Z400	2.67GHz	Xeon	4	12GB	125GB	600W
WST41	xw4100	3.20GHz	P4	2	2GB	80GB	475W
WST44	xw4400	2.67GHz	P4	2	2GB	250GB	475W
WST45	xw6200	3.40GHz	Xeon	2	4GB	160GB	475W
WST46	xw6200	3.40GHz	Xeon	2	4GB	160GB	475W
WST47	dc5800	2.93GHz	P4	2	2GB	160GB	240W
WST48	dx7300	1.80GHz	P4	2	3GB	250GB	240W
WST69	Z420	2.80GHz	Xeon	4	12GB	125GB	600W
WST83	wx6400	1.86GHz	Xeon	4	4GB	250GB	575W
WST84	wx6400	1.86GHz	Xeon	4	4GB	250GB	575W
WST96	wx6400	1.86GHz	Xeon	4	4GB	250GB	575W

As each calculation node in the dedicated calculation grid is configured similar to the workstation cluster node's configuration that was explained in Chapter 4. Hence, each node is configured to act as a calculation node within the cluster, and the following configuration is implemented:

- Perform regular hardware diagnostics and security settings.
- Regular operating system, application and utility software updates.
- Security software updates and password and access rights control.
- Database access rights for distributed processing database access.
- Remote access rights for remote management.
- Protected file share directories for local data and messages.
- Communication protocols for message passing.
- Local copy of distributed processing applications installed in each node.
- Calculation node controller installed in each node.

When the dedicated cluster calculation node boots up, the calculation node controller starts automatically, stays in the memory, and listens for messages from the cluster controller. When the message arrives from the cluster controller, the calculation node controller processes the message and executes instruction accordingly. Each calculation node has a directory path C:\NWDP\ in the local drive for distributed processing applications. It has subdirectories that store messages and data that are required for functioning as a calculation node within the cluster. This directory is protected for security reasons, and each calculation node has a shared directory.

6.5 Testing Procedure Configuration

To test the PC and NUC cluster calculation performance, a single CB from the current portfolio is used for the CB theoretical value calculation for different scenarios using Equation (6.1). The CB model that used for testing is described in detail in section 4.2.1 in Chapter 5.

The following configurations are used for testing:

- Single CB is selected from the current portfolio for scenario analysis.
- Financial calculation models used are binomial and trinomial methods.
- Four scenarios are calculated with varying parameters.
- The three programming methods used are Excel-VBA, VBScript, and EXE.

The risk scenario application is used for testing under various conditions and utilises the CB model that described in section 4.2.1. The following input parameters are used as variable for testing and the rest of the input parameters are kept as static.

- Parity (s)
- Implied Volatility (v)
- Interest Rate (r)
- Calculation step (N)

The following calculation scenarios are used for the selected CB:

- CB theoretical value profile
- Implied Volatility
- IV impact on CB theoretical value profile
- Interest Rate impact on CB theoretical value profile

The CB theoretical value calculation model used for testing is shown in Equation (6.1):

$$P = f(N, s, v, r, T, E) \tag{6.1}$$

where

P	Theoretical value
N	Number of steps
s	Parity
v	Implied volatility
r	Interest rate
T	Time to maturity
E	Redemption price

The test parameter ranges are selected to test the extreme case scenarios to prove that the distributed processing cluster is capable of calculating highly compute-intensive calculations within the acceptable timeframes. In addition, different programming methods are used to test how the existing MS-Excel-based mathematical models can be modified to improve the calculation speed. To test and analyse the calculation performance of both PC and NUC clusters using CB theoretical value calculations, binomial and trinomial tree methods with all VBA, VBS, and EXE programs are used. For testing the CB model calculations, only certain input parameters are used as variables and the rest of the input parameters are kept as static input parameters. Table 6.4 lists the input parameters used for scenario testing, and Table 6.5 presents parameters used for cluster testing.

Table 6.4: Parameters used for scenario testing

Parameter	Description
Parity	10 to 170 (increment size 1)
Maturity	5 years (fixed)
Interest Rate Shift	-2%, 0%, +2% and +4.5% (fixed)
Red Price	103 (fixed)
Implied Volatility	Varies between 1% to 95%, 45%, 55%, and 65%
Calculation Steps	100 to 1000 (increment size 100)

Table 6.5: Parameters used for PC and NUC cluster testing

Calculation Type	Cluster	Description
Discrete calculation at each node	PC	Step size: 500 Method: Binomial, Trinomial Program: MS-Excel VBA, VBS, EXE Number of calculations: 160
Distributed calculation across all nodes	PC	Step size: 500 Method: Binomial, Trinomial Program: MS-Excel VBA, VBS, EXE Number of calculations: Varies
Discrete calculation at each node	NUC	Step size: 100 to 1000 Method: Binomial, Trinomial Program: MS-Excel VBA, VBS, EXE Number of calculations: 16
Distributed calculation across all nodes	NUC	Step size: 500 Method: Binomial, Trinomial Program: MS-Excel VBA, VBS, EXE Number of calculations: 16

Hence, using Equation (6.1), scenario analysis calculations are performed using the following input parameters as variables: Parity (s), Interest Rate (r), Implied Volatility (v) and Calculation steps (N). The rest of the input parameters are kept as static parameters. Furthermore, for each scenario analysis, only one input parameter is used as variable parameter and the rest of the input parameters are kept as static parameters and these are explained in detail in forth coming sections. However, the calculation steps (N) is changed for each scenario calculation to compare calculation performance between NUC cluster and PC cluster.

6.6 Load Balancing

Only the PC cluster is used for testing the calculation index-based load balancing methods, and each calculation node within the PC cluster has a calculation index based on CPU power and memory capacity as explained in Chapter 5 where the adaptive load balancing techniques are discussed in detail. The calculation index is calibrated using a test program to execute under controlled conditions to measure the time taken to complete the task and that is based on each scenario's calculations. The NUC cluster consists of the same hardware and software for each calculation node; hence, the calculation index for each node is the same, hence the load is equally distributed. Each NUC cluster calculation node is allocated 16 calculations per scenario because the total calculations needed are 160 per scenario and the NUC cluster has 10 calculation nodes, hence 16 calculations per node are performed. However, the PC cluster has 11 calculation nodes, and the number of calculations per calculation node varies depending on the calculation index of the node concerned.

6.7 CB Theoretical Value Calculation

Mathematical models used for financial calculations are specifically designed for the company, and these financial calculation models are continuously modified and changed according to the need of the business and depending on the market conditions. For testing, two generally used CB pricing models are selected: the Binomial-Tree model and the Trinomial-Tree model. These two CB pricing models are used for various calculations in the company that are used for individual-level security analysis and for portfolio-level analysis. Fund managers and traders require different types of information about each security-level analysis and portfolio-level analysis of data to make trading decisions during trading hours. Any serious delays in critical analysis can cause unexpected losses due to incorrect data or misleading information. Therefore, the system that provides critical data that is depended upon for trading and fund management has to be robust and accurate. One of the methods of securing data accuracy and reliability is to execute the calculations in two separate processing units at the same time with the same input data, and this can be achieved by computing clusters. The trading desk is a highly busy environment, and traders and fund managers rarely perform detailed technical analysis themselves; most of the technical analyses are performed by quantitative analysts who provide the data to the trading desk in a simpler form that can be used with tables, charts, and graphs. The CB theoretical value profile analysis is based on three main parameters: CB theoretical value, bond floor, and parity. The CB price and bond floor are calculated using the company's own derivative pricing models, and the parity is calculated using a price feed from a financial feed provider, that is, the market price of the CB's underlying equity. Figure 6.5 shows a typical CB theoretical value profile analysis chart used by trading desks and quantitative analysts. The figure is a simple chart for illustrative purposes only, and it includes a few different tables and charts for compliance, reporting, and risk analysis. The trading desk uses various charting tools with a number of different functionalities; hence, the users have the ability to analyse the data efficiently and quickly. For visual representation, the chart has five regions for quick analysis for making trading decisions. A brief description of the regions and the corresponding technical analysis is shown in Table 6.6.

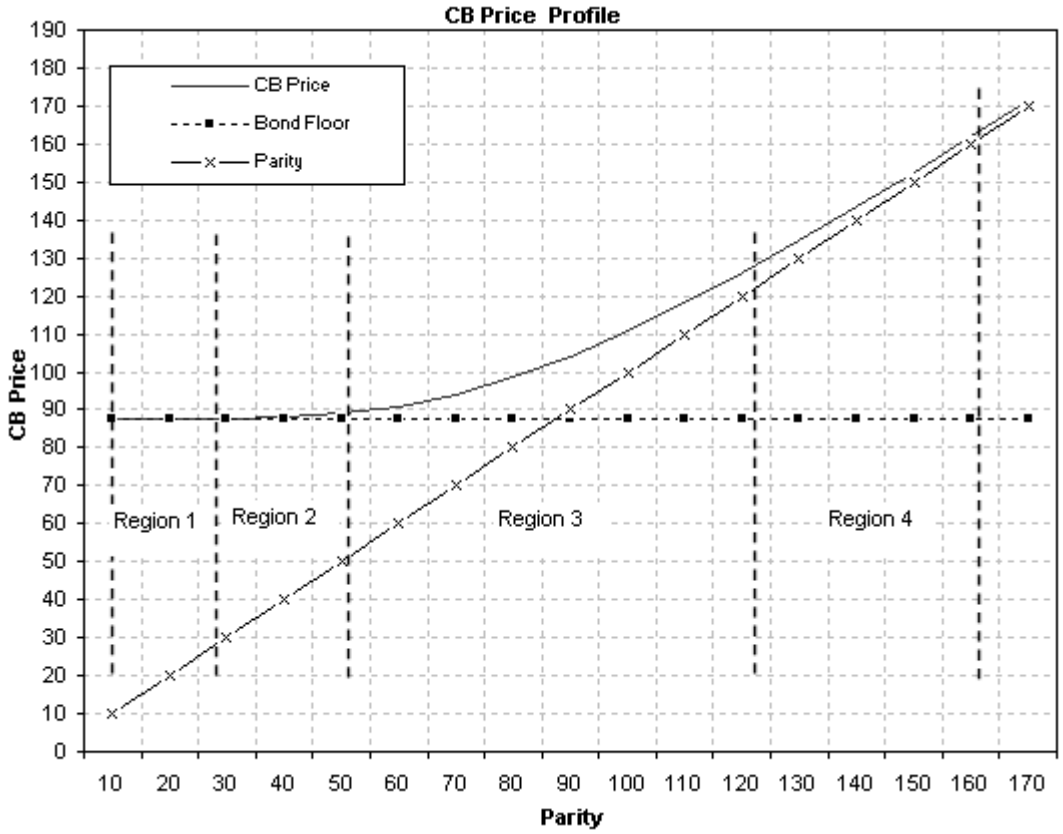


Figure 6.5: CB price profile and technical analysis regions (Source: Northwest)

Table 6.6: Technical analysis of CB price profile

Region	Primary Technical Analysis
Region 1	Default possibility Credit spread impact Interest rate impact Cheapness comparison Bond floor Asset-swapping
Region 2	Put provisions Cheapness comparison Upward and downward parity Bond floor Volatility impact
Region 3	Upward and downward parity Hedging parameters Arbitraging Volatility impact
Region 4	Conversion possibility Call provisions

For illustration, Table 6.7 shows the CB theoretical value differences using different derivative pricing models, with the difference shown in point value, and Table 6.8 shows the dollar value differences. These data are normally pre-calculated for quick lookups and analysis using various charts and tables by trading desk users. For example, if a trade worth 10 million dollars is executed with incorrect CB pricing calculation against mark-to-market price, it will cause a loss of $43,393 if the parity is 150 and implied volatility is 70%. These types of simplified lookup data constructs are useful for trading desk users; however, these calculations are compute-intensive and time-consuming to calculate for all the possible scenarios. Hence, the dedicated calculation cluster is one way of improving the calculation efficiency. For testing CB theoretical value calculations, the parameters used are shown in Table 6.9.

Table 6.7: CB price calculation model error in point value with varying parity and implied volatility

	Volatility						
Parity	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**
10	-0.0002	-0.0002	-0.0002	-0.0006	-0.0017	-0.0010	0.0051
20	-0.0002	-0.0002	-0.0009	-0.0049	0.0031	0.0000	0.0171
30	-0.0002	-0.0005	-0.0056	-0.0129	-0.0114	-0.0041	0.0232
40	-0.0002	-0.0022	-0.0164	-0.0012	0.0179	0.0502	0.0143
50	-0.0003	-0.0007	-0.0095	0.0250	0.0115	0.1287	-0.0023
60	-0.0010	0.0021	0.0187	-0.0346	-0.0198	0.1652	0.0981
70	-0.0059	-0.0024	0.0058	0.0173	0.0169	0.0792	0.2199
80	-0.0081	0.0136	0.0366	0.0390	0.0757	0.0688	0.1006
90	-0.0163	-0.0272	0.0100	-0.0172	0.0981	0.1491	0.2379
100	0.0156	0.0129	0.0149	0.0272	0.0579	0.1178	0.2203
110	0.0013	-0.0048	0.0345	0.0479	0.0817	0.1477	0.2611
120	0.0135	-0.0081	0.0354	-0.0139	0.1148	0.2041	0.3269
130	0.0038	0.0269	-0.0016	0.1100	0.0843	0.1016	0.2504
140	0.0018	0.0252	-0.0486	0.0067	0.1603	0.2237	0.3076
150	0.0018	0.0091	0.0414	0.0802	0.0343	0.2402	0.4339
160	0.0026	0.0011	0.0265	0.0044	0.2028	0.1073	0.4263
170	0.0029	-0.0056	-0.0142	0.0800	0.0302	0.2710	0.2613

Table 6.8: CB price calculation model error in $USD value with varying parity and implied volatility

	Volatility						
Parity	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**
10	-$24	-$24	-$23	-$63	-$174	-$105	$514
20	-$24	-$23	-$90	-$488	$311	-$5	$1,706
30	-$24	-$54	-$561	-$1,289	-$1,138	-$412	$2,323
40	-$24	-$217	-$1,642	-$118	$1,790	$5,016	$1,426
50	-$29	-$72	-$945	$2,503	$1,146	$12,875	-$231
60	-$96	$211	$1,867	-$3,461	-$1,978	$16,523	$9,811
70	-$587	-$243	$578	$1,727	$1,686	$7,922	$21,989
80	-$812	$1,361	$3,661	$3,905	$7,567	$6,884	$10,057
90	-$1,626	-$2,718	$563	-$1,722	$9,811	$14,908	$23,792
100	$1,564	$1,289	$1,491	$2,721	$5,792	$11,777	$22,034
110	$125	-$478	$3,449	$4,788	$8,168	$14,774	$26,108
120	$1,346	-$809	$3,542	-$1,393	$11,484	$20,406	$32,693
130	$378	$2,690	-$163	$11,003	$8,434	$10,164	$25,037
140	$179	$2,522	-$4,858	$673	$16,026	$22,366	$30,763
150	$179	$908	$4,144	$8,025	$3,435	$24,017	$43,393
160	$259	$107	$2,650	$436	$20,276	$10,730	$42,635
170	$287	-$560	-$1,416	$8,002	$3,017	$27,095	$26,131

Table 6.9: Test parameters of CB theoretical value calculations

Parameter	**Value**	**Description**
Parity	10 to 170	Variable (increment 1)
Calculation Step	100 to 1000	Variable (increment 100)
Interest Rate	2.5 %	Fixed
Implied Volatility	55%	Fixed
Maturity	5 years	Fixed
Red Price	103	Fixed

Hence, for CB theoretical value calculation testing, only Parity (*s*) and calculation steps (*N*) are variable inputs to the CB model and Maturity (*T*), Red Price (*E*), Interest Rate (r) and Implied volatility (*v*) are kept as static parameters. Hence, the Parity (*s*) minimum value is 10, and the maximum value is 170. The calculation steps (N) minimum value is 100, and maximum value is 1000.

6.7.1 Using PC and NUC Cluster for CB Value Calculation

Table 6.10 lists the calculation times for each calculation node using the PC cluster, and Table 6.11 lists the calculation times with varying step size using a single NUC computer. Table 6.12 lists the distributed calculation times using the PC cluster with imbalanced and balanced load conditions, and Table 6.13 lists the distributed calculation times using the NUC computer cluster with load balanced condition.

Table 6.10: Calculation time for each calculation node for CB theoretical value calculation using PC cluster

Calculation Node	Calculation Time (sec)					
	Binomial			Trinomial		
	VBA	VBS	EXE	VBA	VBS	EXE
WST69	15	22	4	28	35	4
WST35	15	23	4	29	36	4
WST47	18	25	5	35	42	6
WST44	20	28	6	39	46	7
WST96	24	37	9	50	60	10
WST83	24	37	9	54	60	10
WST85	26	37	9	55	64	10
WST48	28	38	10	56	65	11
WST45	28	41	11	59	65	12
WST46	29	42	11	65	66	12
WST54	30	43	12	66	73	14

Table 6.11: Calculation time using varying step size for CB theoretical value calculation using a single NUC computer

Step Size	Calculation Time (sec)					
	Binomial			Trinomial		
	VBA	VBS	EXE	VBA	VBS	EXE
100	2	3	1	5	7	2
200	9	13	3	17	24	4
300	20	27	7	36	51	8
400	34	48	14	63	88	14
500	53	76	20	99	138	23
600	77	109	26	141	199	30
700	103	147	33	190	268	40
800	135	194	41	244	351	48
900	169	244	52	323	444	59
1000	209	300	65	393	544	72

Table 6.12: Distributed calculation time for CB theoretical value calculation using PC cluster with load imbalanced and balanced conditions

Calculation Node	Calculation Time (sec)											
	Imbalanced						Balanced					
	Binomial			Trinomial			Binomial			Trinomial		
	VBA	VBS	EXE	VBA	VBS	EXE	VBA	VBS	EXE	VBA	VBS	EXE
WST69	1.4	2.1	0.4	2.6	3.3	0.4	2.1	2.8	0.7	4.0	4.6	0.7
WST35	1.4	2.2	0.4	2.7	3.4	0.4	2.1	2.9	0.7	3.8	4.5	0.7
WST47	1.7	2.3	0.5	3.3	3.9	0.6	2.0	3.0	0.6	4.2	4.7	0.7
WST44	1.9	2.6	0.6	3.7	4.3	0.7	2.0	3.0	0.6	4.1	4.9	0.7
WST96	2.3	3.5	0.8	4.7	5.6	0.9	2.0	3.0	0.6	4.1	4.9	0.7
WST83	2.3	3.5	0.8	5.1	5.6	0.9	2.0	3.0	0.6	4.1	4.9	0.7
WST85	2.3	3.2	0.8	4.8	5.6	0.9	2.0	3.0	0.6	4.1	4.8	0.7
WST48	2.5	3.3	0.9	4.9	5.7	1.0	1.9	2.9	0.6	4.2	4.9	0.7
WST45	2.5	3.6	1.0	5.2	5.7	1.1	1.9	2.8	0.6	4.1	4.9	0.7
WST46	2.5	3.7	1.0	5.7	5.8	1.1	2.0	2.9	0.6	4.1	5.0	0.7
WST54	2.6	3.8	1.1	5.8	6.4	1.2	2.1	3.0	0.7	4.1	4.6	0.7

Table 6.13: Distributed calculation time for CB theoretical value calculation using NUC cluster with load balanced condition

Calculation Node	Calculation Time (sec)					
	Binomial			Trinomial		
	VBA	VBS	EXE	VBA	VBS	EXE
NUC01	5.34	7.60	2.00	9.99	13.87	2.31
NUC02	5.32	7.60	2.02	9.96	13.92	2.32
NUC03	5.32	7.61	2.01	9.93	13.88	2.31
NUC04	5.33	7.60	2.01	9.90	13.84	2.31
NUC05	5.34	7.66	2.01	9.92	13.91	2.31
NUC06	5.34	7.65	2.01	10.00	13.89	2.32
NUC07	5.33	7.60	2.01	9.95	13.93	2.31
NUC08	5.33	7.63	2.00	9.98	13.83	2.31
NUC09	5.33	7.61	2.00	9.92	13.82	2.30
NUC10	5.31	7.64	2.01	9.99	13.90	2.30

6.8 Implied Volatility (IV) Calculation

To calculate the implied volatility, the derivative instrument price has to be calculated using the theoretical value model for number of times with varying volatility using the Newton-Raphson method. This process is the most time-consuming process within the distributed processing systems and can cause serious time delays during the operation. For testing, a simple static load balancing is used; however, to get the load balancing mechanism to adapt to the long delays in calculations, some type of approximation technique has to be used to predict the estimated time. This can be done by collecting historical data to predict the estimated calculation time. However, small changes in the derivate prices can cause the number of calculations needed to calculate the implied volatility to be increased significantly. Hence, to predict the estimated calculation time, a number of different data sets have to be maintained as lookup tables with fuzzy logic-based rules to estimate the calculation time during the operation. However, for distributed process performance testing, the linear type input dataset for the CB theoretical value model is selected to avoid unexpected financial model-related time delays that can cause incorrect results. Table 6.14 lists the IV calculation test parameters.

Table 6.14: IV calculation test parameters

Parameter	Value	Description
Parity	10 to 170	Variable (increment size 1)
Calculation Step	100 to 1,000	Variable (increment size 100)
Interest Rate	4.5%	Fixed
Implied Volatility	1 to 95	$f(P,v)$ (step varies)
Maturity	5 Years	Fixed
Red Price	103	Fixed

Hence, for Implied Volatility calculation testing, only Parity (s) and calculation steps (N) are variable inputs to the CB model and Maturity (T), Red Price (E), Interest Rate (r) and Implied Volatility (v) are kept as static parameters. Hence, the Parity (s) minimum value is 10, and the maximum value is 170. The calculation steps (N) minimum value is 100, and maximum value is 1000. However, in this scenario is a recursive calculation, hence, the output data, the Implied Volatility (IV) is also used as input data for the next calculation, that is, for $(i+1)^{th}$ calculation, $v(i+1)=IV(i)$.

6.8.1 Using PC and NUC Cluster for IV Value Calculation

Table 6.15 lists the calculation times for each calculation node using the PC cluster, and Table 6.16 lists the calculation times with varying step size using a single NUC computer. Table 6.17 lists the distributed calculation times using the PC cluster with imbalanced and balanced load conditions, and Table 6.18 lists the distributed calculation times using the NUC computer cluster with load balanced condition.

Table 6.15: Calculation time for each calculation node for IV value calculation using PC cluster

Calculation Node	Calculation Time (sec)					
	Binomial			Trinomial		
	VBA	VBS	EXE	VBA	VBS	EXE
WST69	60	28	9	61	73	14
WST35	61	28	8	60	73	18
WST47	73	38	11	73	91	21
WST44	81	42	12	81	101	30
WST96	94	61	18	94	140	30
WST83	95	61	18	95	141	30
WST85	104	57	18	104	138	31
WST48	113	58	18	113	139	35
WST45	113	61	18	113	145	35
WST46	116	60	18	116	147	33
WST54	122	62	19	122	154	39

Table 6.16: Calculation time using varying step size for IV value calculation using a single NUC computer

Step Size	Calculation Time (sec)					
	Binomial			Trinomial		
	VBA	VBS	EXE	VBA	VBS	EXE
100	25	17	3	28	27	7
200	38	47	6	51	87	13
300	62	99	12	91	189	24
400	94	164	23	147	323	39
500	134	229	34	216	480	57
600	184	334	47	306	687	73
700	244	438	64	410	872	98
800	300	537	84	523	1,183	128
900	396	672	105	698	1,496	151
1000	479	859	129	816	1,787	179

Table 6.17: Distributed calculation time for IV value calculation using PC cluster with load imbalanced and balanced conditions

Calculation Node	Calculation Time (sec)											
	Imbalanced						Balanced					
	Binomial			Trinomial			Binomial			Trinomial		
	VBA	VBS	EXE	VBA	VBS	EXE	VBA	VBS	EXE	VBA	VBS	EXE
WST69	5.6	2.6	0.8	5.7	6.8	1.3	7.9	4.0	1.3	8.0	10.0	2.3
WST35	5.7	2.6	0.8	5.6	6.8	1.7	8.0	4.2	1.3	7.9	10.5	2.4
WST47	6.8	3.6	1.0	6.8	8.5	2.0	8.2	4.3	1.2	8.2	10.2	2.4
WST44	7.6	3.9	1.1	7.6	9.5	2.8	8.1	4.2	1.3	8.1	10.1	2.4
WST96	8.8	5.7	1.7	8.8	13.1	2.8	8.2	4.2	1.2	8.2	10.5	2.4
WST83	8.9	5.7	1.7	8.9	13.2	2.8	8.3	4.2	1.2	8.3	10.6	2.4
WST85	9.1	5.0	1.6	9.1	12.1	2.7	7.8	4.3	1.2	7.8	10.4	2.3
WST48	9.9	5.1	1.6	9.9	12.2	3.1	7.8	4.4	1.2	7.8	10.4	2.4
WST45	9.9	5.3	1.6	9.9	12.7	3.1	7.8	4.2	1.2	7.8	10.0	2.4
WST46	10.2	5.3	1.6	10.2	12.9	2.9	8.0	4.1	1.2	8.0	10.1	2.5
WST54	10.7	5.4	1.7	10.7	13.5	3.4	8.4	4.3	1.3	8.4	10.6	2.4

Table 6.18: Distributed calculation time for IV value calculation using NUC cluster with load balanced condition

Calculation Node	Calculation Time (sec)					
	Binomial			Trinomial		
	VBA	VBS	EXE	VBA	VBS	EXE
NUC01	13.42	23.12	3.42	21.75	48.04	5.75
NUC02	13.46	23.01	3.42	21.73	48.00	5.76
NUC03	13.50	23.00	3.43	21.81	48.20	5.75
NUC04	13.51	22.94	3.40	21.62	48.09	5.75
NUC05	13.49	22.94	3.40	21.79	48.17	5.74
NUC06	13.53	22.91	3.43	21.79	48.18	5.72
NUC07	13.43	23.00	3.41	21.65	48.09	5.73
NUC08	13.52	23.00	3.43	21.72	48.42	5.75
NUC09	13.46	23.00	3.43	21.66	48.30	5.73
NUC10	13.47	23.11	3.41	21.81	48.34	5.74

6.9 CB Theoretical Value IV Sensitivity Analysis

CB price is correlated with underlying share price, and the underlying share price depends on the recorded ninety days volatility. The implied volatility (IV) is calculated to find the volatility that returns the theoretical value equal to the current market CB price. By using calculated IV, various scenarios' analysis is performed on a particular CB or portfolio as a whole. One of the scenario analyses is to vary the IV and examine how the CB will behave under various conditions using the current CB price and calculated theoretical value. The CB's sensitivity to the IV depends on the current CB price and in what region the CB is located in the CB price profile chart that is shown in Figure 6.5. Hence, traders, fund managers, and quantitative analysts continuously monitor each CB and the whole portfolio to forecast 'what-if' scenarios for changing IV. The IV impact analysis data is also used for risk reporting, trade decision-making, and portfolio analysis, and quantitative analysts perform various calculations on saved data such as autocorrelation and cross-correlations. These types of scenario analyses are highly compute-intensive tasks and require considerable amounts of processing power. The implementation of the dedicated calculation cluster improves the overall calculation time considerably for these calculations. Generally, CB theoretical value increases with increase in IV and vice versa, as shown in Figure 6.6; however, the IV impact on the CB depends on where the CB price is located in the CB price profile. As expected, the testing results show that using both the PC cluster and the NUC cluster improved the calculation time considerably for IV impact analysis compared with using a single workstation or single PC. Table 6.19 lists the input parameters for the IV impact calculations. Where parity is set 1 to 170 with increment of 1. Calculation step is set 100 to 1000 with increment of 100. Implied volatility is set as 3 values, 45%, 55%, and 65%. The interest rate, maturity and red price are fixed.

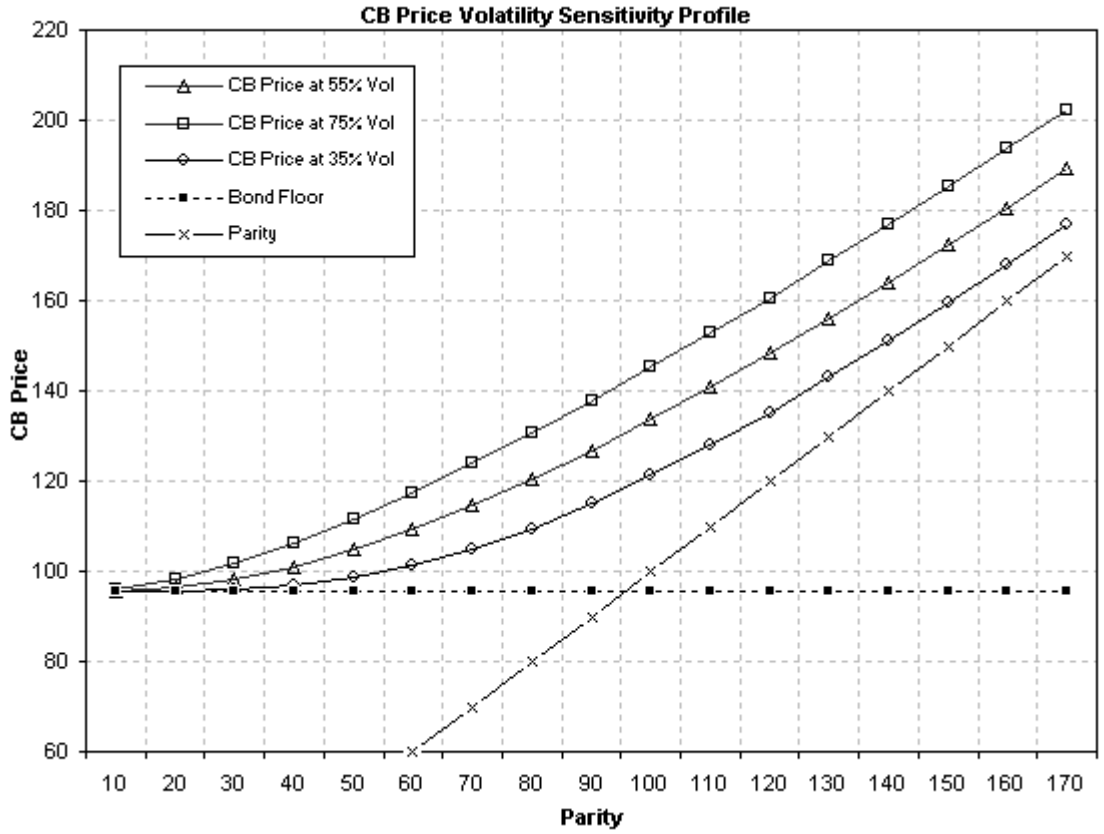


Figure 6.6: Implied Volatility (IV) impact on CB theoretical value

Table 6.19: IV impact on test parameters of CB theoretical value calculations

Parameter	Value	Description
Parity	10 to 170	Variable (increment size 1)
Calculation Step	100 to 1000	Variable (increment size 100)
Interest Rate	2.5 %	Fixed
Implied Volatility	55% ± 10%	65%, 55%, 45%
Maturity	5 Years	Fixed
Red Price	103	Fixed

Hence, for IV sensitivity analysis testing, only Parity (*s*) and calculation steps (*N*) are variable inputs to the CB model and Maturity (*T*), Red Price (*E*), and Interest Rate (r) are kept as static parameters. Hence, the Parity (*s*) minimum value is 10, and the maximum value is 170. The calculation steps (*N*) minimum value is 100, and maximum value is 1000. However, three values 65%, 55% and 45 are used for Implied Volatility (*v*) for testing IV sensitivity analysis.

6.9.1 Using PC and NUC Cluster for IV Sensitivity Calculation

Table 6.20 lists the calculation times for each calculation node using the PC cluster, and Table 6.21 lists the calculation times using varying step size using a single NUC computer. Table 6.22 shows the distributed calculation times using the PC cluster with imbalanced and balanced load conditions, and Table 6.23 lists the distributed calculation times using the NUC computer cluster with load balanced condition.

Table 6.20: Calculation time for each calculation node for IV sensitivity calculation using PC cluster

Calculation Node	Calculation Time (sec)					
	Binomial			Trinomial		
	VBA	VBS	EXE	VBA	VBS	EXE
WST69	36	68	12	85	109	13
WST35	36	68	12	85	109	18
WST47	44	77	17	104	127	21
WST44	49	88	18	115	138	30
WST96	63	110	26	141	175	30
WST83	62	111	27	144	178	30
WST85	62	111	27	148	189	31
WST48	68	113	26	159	195	33
WST45	68	115	27	161	196	36
WST46	69	125	28	164	202	36
WST54	73	131	29	172	219	42

Table 6.21: Calculation time using varying step size for IV sensitivity calculation using a single NUC computer

Step Size	Calculation Time (sec)					
	Binomial			Trinomial		
	VBA	VBS	EXE	VBA	VBS	EXE
100	9	10	3	15	16	5
200	26	37	10	50	53	12
300	53	82	19	109	114	23
400	89	145	32	190	199	38
500	133	226	50	294	311	57
600	188	329	71	424	443	80
700	251	445	96	576	602	109
800	324	580	124	749	784	140
900	405	732	156	947	989	177
1,000	496	904	194	1168	1223	218

Table 6.22: Distributed calculation time for IV sensitivity calculation using PC cluster with load imbalanced and balanced conditions

Calculation Node	Calculation Time (sec)											
	Imbalanced						Balanced					
	Binomial			Trinomial			Binomial			Trinomial		
	VBA	VBS	EXE	VBA	VBS	EXE	VBA	VBS	EXE	VBA	VBS	EXE
WST69	3.4	6.4	0.4	8.0	10.2	1.1	4.7	9.4	0.7	11.2	13.6	2.0
WST35	3.4	6.4	0.4	8.0	10.2	1.1	4.7	8.9	0.7	11.7	14.3	1.9
WST47	4.1	7.2	0.6	9.8	11.9	1.6	5.0	8.7	0.7	11.7	14.3	1.8
WST44	4.6	8.3	0.7	10.8	12.9	1.7	4.9	8.8	0.7	11.5	14.7	1.8
WST96	5.9	10.3	0.9	13.2	16.4	2.4	4.7	8.9	0.7	11.5	14.2	1.8
WST83	5.8	10.4	0.9	13.5	16.7	2.5	5.0	9.0	0.7	11.7	14.5	1.9
WST85	5.4	9.7	0.9	13.0	16.5	2.4	5.0	9.0	0.7	11.1	14.2	1.9
WST48	6.0	9.9	1.0	13.9	17.1	2.3	5.1	8.5	0.7	11.9	14.6	1.8
WST45	6.0	10.1	1.1	14.1	17.2	2.4	5.1	8.6	0.7	11.1	14.7	1.9
WST46	6.0	10.9	1.1	14.4	17.7	2.5	4.7	8.6	0.7	11.3	13.9	1.9
WST54	6.4	11.5	1.2	15.1	19.2	2.5	5.0	9	0.7	11.8	13.7	1.8

Table 6.23: Calculation time profile for IV sensitivity calculation using NUC cluster under load balanced condition

Calculation Node	Calculation Time (sec)					
	Binomial			Trinomial		
	VBA	VBS	EXE	VBA	VBS	EXE
NUC01	13.38	22.81	5.01	29.57	31.32	5.73
NUC02	13.42	22.82	5.03	29.40	31.35	5.72
NUC03	13.35	22.64	5.02	29.60	31.30	5.70
NUC04	13.36	22.69	5.02	29.57	31.25	5.71
NUC05	13.40	22.82	5.04	29.44	31.22	5.76
NUC06	13.31	22.72	5.01	29.41	31.14	5.73
NUC07	13.42	22.71	5.05	29.52	31.17	5.74
NUC08	13.37	22.65	5.05	29.53	31.28	5.73
NUC09	13.43	22.61	5.03	29.65	31.37	5.75
NUC10	13.36	22.67	5.03	29.48	31.34	5.73

6.10 CB Theoretical Value IR Sensitivity Analysis

CB is highly sensitive to the interest rate (IR) of the CB currency and underlying share currency. The IR impact analysis data is used for risk reporting, trade decision-making, and portfolio analysis. The quantitative analysts perform various calculations on saved data such as autocorrelation and cross-correlations across various CBs and IR moves. The CB pricing mathematical models use yield curves for analysing IR impact on CB theoretical value, and two types of IR: risk-free rate and risky rate. The implementation of IR impact on CB theoretical value is called parallel shifts, that is, shifting the yield curve by same value as shown in Figure 6.7. For testing purposes, the risky rate is used, and generally, CB theoretical value increases with reducing interest rates and vice versa as shown in Figure 6.8.

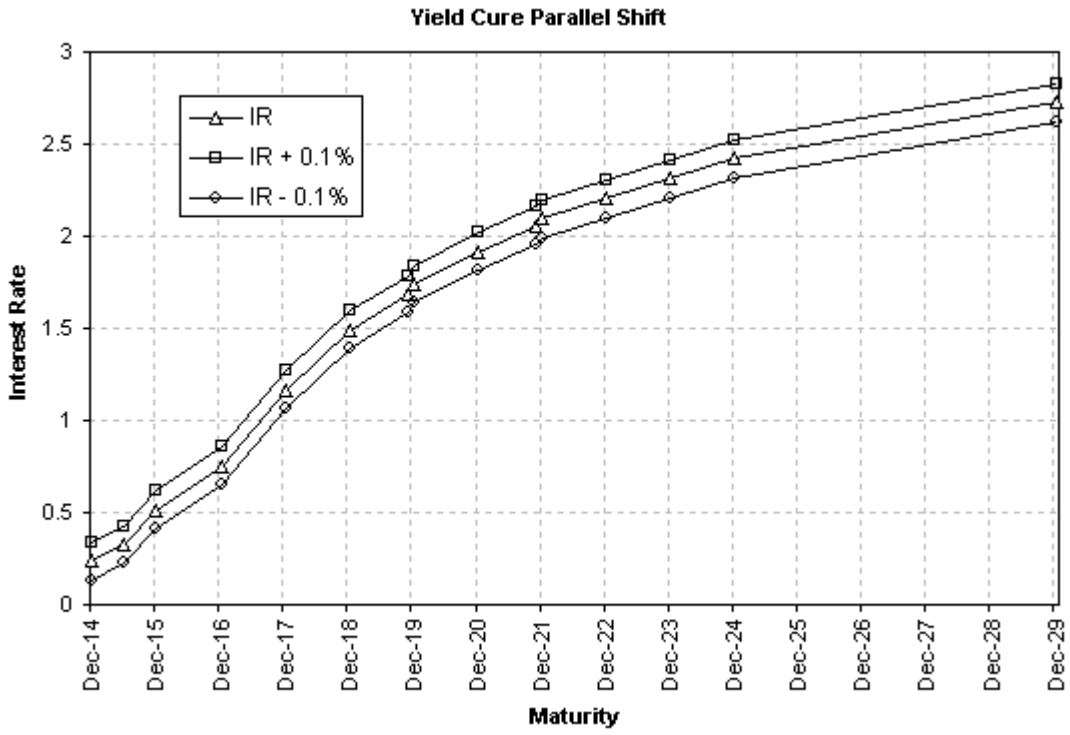


Figure 6.7: Yield parallel shift for USD (Data source: Northwest)

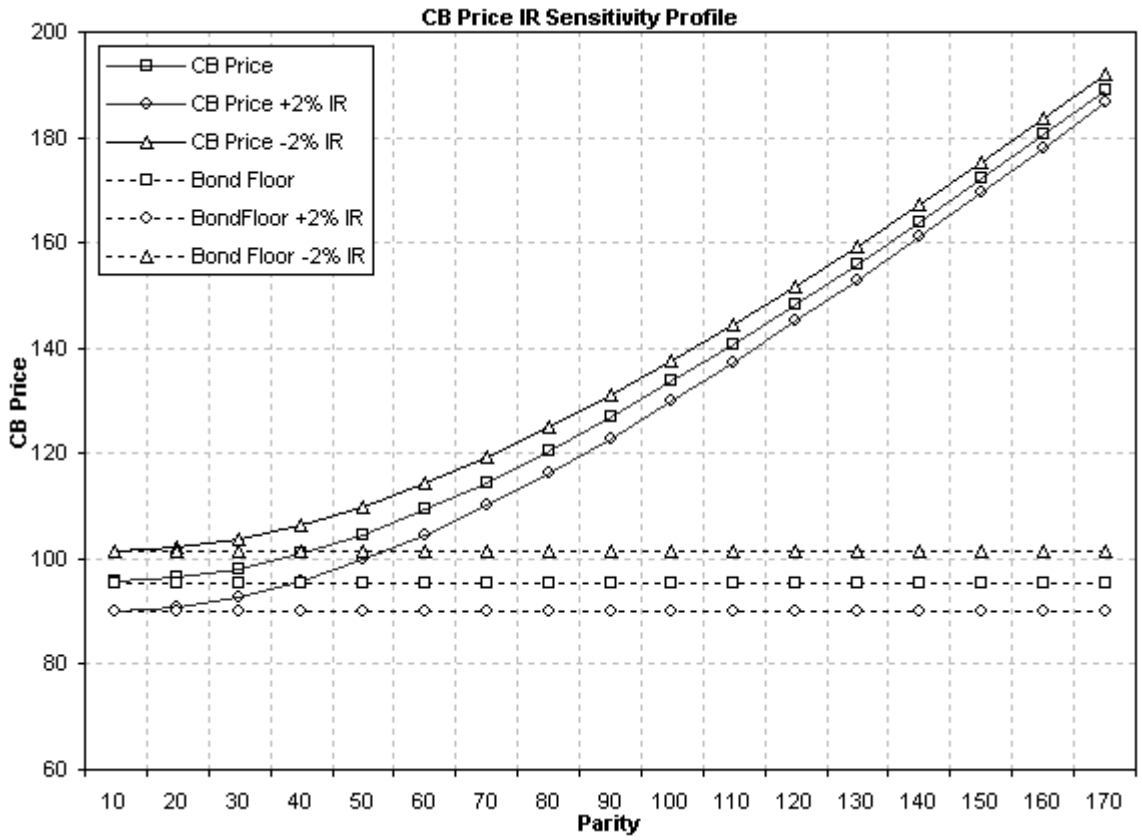


Figure 6.8: IR impact on CB price and bond floor (Source: Northwest)

Hence, for IR sensitivity analysis testing, only Parity (*s*) and calculation steps (*N*) are variable inputs to the CB model and Maturity (*T*), Red Price (*E*), and Interest Rate (r) are kept as static parameters. Hence, the Parity (*s*) minimum value is 10, and the maximum value is 170. The calculation steps *(N*) minimum value is 100, and maximum value is 1000. However, three values -2%, 0% and +2% are used for Interest rate (*r)* for testing IR sensitivity analysis. Table 6.24 lists the input parameters for IR impact on CB theoretical value calculation.

Table 6.24: Parameters used for IR impact on CB theoretical value calculation

Parameter	Value	Description
Parity	10 to 170	Variable (increment size 1)
Calculation step	100 to 1,000	Variable (increment size 100)
Interest Rate shift	± 2%	-2%,0%,+2%
Implied volatility	55%	Fixed
Maturity	5 years	Fixed
Red Price	103	Fixed

6.10.1 Using PC and NUC Cluster for IR Sensitivity Calculation

Table 6.25 lists the calculation times for each calculation node using the PC cluster, and Table 6.26 lists the calculation times using varying step size using a single NUC computer. Table 6.27 presents the distributed calculation times using PC cluster with imbalanced and balanced load conditions, and Table 6.28 lists the distributed calculation times using the NUC computer cluster with load balanced condition.

Table 6.25: Calculation time for each calculation node for IR sensitivity calculation using PC cluster

Calculation Node	Calculation Time (sec)					
	Binomial			Trinomial		
	VBA	VBS	EXE	VBA	VBS	EXE
WST69	35	67	11	82	105	13
WST35	37	67	12	85	106	13
WST47	43	76	16	104	123	18
WST44	47	84	18	115	137	21
WST96	61	108	26	140	178	30
WST83	62	109	26	142	180	30
WST85	63	111	26	148	191	31
WST48	66	115	27	159	192	31
WST45	66	115	27	161	192	36
WST46	67	125	28	164	202	36
WST54	71	131	29	172	217	42

Table 6.26: Calculation time using varying step size for IR sensitivity calculation using a single NUC computer

Step Size	Calculation Time (sec)					
	Binomial			Trinomial		
	VBA	VBS	EXE	VBA	VBS	EXE
100	9	9	3	14	16	6
200	26	36	10	49	52	11
300	53	81	18	107	112	24
400	89	142	33	185	197	38
500	133	223	50	288	305	57
600	190	320	70	414	437	81
700	250	434	96	562	592	109
800	325	567	124	733	773	141
900	405	717	157	925	1,000	177
1000	496	884	194	1,159	1,202	218

Table 6.27: Distributed calculation time for IR sensitivity calculation using PC cluster with imbalanced and balanced load conditions

Calculation Node	Calculation Time (sec)											
	Imbalanced						Balanced					
	Binomial			Trinomial			Binomial			Trinomial		
	VBA	VBS	EXE	VBA	VBS	EXE	VBA	VBS	EXE	VBA	VBS	EXE
WST69	3.3	6.3	1.0	7.7	9.8	1.2	4.8	9.2	1.9	11.8	15.1	2.2
WST35	3.5	6.3	1.1	8.0	9.9	1.2	4.9	8.8	1.8	11.2	13.9	2.1
WST47	4.0	7.1	1.5	9.8	11.5	1.7	4.8	8.6	1.8	11.7	13.8	2.1
WST44	4.4	7.9	1.7	10.8	12.8	2.0	4.7	8.9	1.8	11.5	14.6	2.1
WST96	5.7	10.1	2.4	13.1	16.7	2.8	5.0	8.8	1.8	11.4	14.5	2.1
WST83	5.8	10.2	2.4	13.3	16.9	2.8	4.7	8.9	1.8	11.5	14.6	2.1
WST85	5.5	9.7	2.3	13	16.7	2.7	4.7	8.3	1.8	11.1	14.3	2.1
WST48	5.8	10.1	2.4	13.9	16.8	2.7	5.0	8.6	1.9	10.9	14.4	2.1
WST45	5.8	10.1	2.4	14.1	16.8	3.2	5.0	8.6	1.9	11.1	14.4	2.3
WST46	5.9	10.9	2.5	14.4	17.7	3.2	4.6	8.6	1.8	11.3	13.9	2.3
WST54	6.2	11.5	2.5	15.1	19.0	3.7	4.9	9.0	1.8	11.8	13.6	2.1

Table 6.28: Calculation time profile for IR sensitivity calculation using NUC cluster under load-balanced condition

Calculation Node	Calculation Time (sec)					
	Binomial			Trinomial		
	VBA	VBS	EXE	VBA	VBS	EXE
NUC01	13.37	22.48	5.05	29.10	30.74	5.76
NUC02	13.33	22.38	5.01	28.85	30.72	5.78
NUC03	13.31	22.62	5.05	29.19	30.60	5.71
NUC04	13.43	22.39	5.04	29.05	30.76	5.72
NUC05	13.41	22.31	5.03	29.08	30.77	5.71
NUC06	13.38	22.37	5.01	29.01	30.64	5.71
NUC07	13.33	22.50	5.07	29.06	30.51	5.77
NUC08	13.33	22.50	5.07	29.11	30.63	5.78
NUC09	13.39	22.58	5.07	28.84	30.52	5.77
NUC10	13.35	22.48	5.06	28.97	30.58	5.75

Table 6.29 to Table 6.32 lists the average calculations time analysis for both PC and NUC clusters using all four scenarios' calculations. Test results have shown considerable calculation time improvements across all the scenario calculations when the load balanced distributed processing method is used compared with the serial calculation method.

Table 6.29: Average calculation time analysis for CB theoretical value calculation

Method	Average Calculation Time (sec)					
	Binomial			Trinomial		
	VBA	VBS	EXE	VBA	VBS	EXE
Serial PC	23.36	33.91	8.18	48.73	55.64	9.09
Imbalanced PC	2.08	3.00	0.72	4.27	4.89	0.80
Balanced PC	2.00	2.93	0.62	4.08	4.81	0.70
Serial NUC	53.00	76.00	20.00	99.00	138.00	23.00
Balanced NUC	5.33	7.62	2.01	9.95	13.88	2.31

Table 6.30: Average calculation time analysis for IV calculation

Method	Average Calculation Time (sec)					
	Binomial			Trinomial		
	VBA	VBS	EXE	VBA	VBS	EXE
Serial PC	93.82	50.55	15.18	93.82	122.00	28.73
Imbalanced PC	8.25	4.48	1.35	8.25	10.78	2.52
Balanced PC	8.01	4.21	1.23	8.01	10.28	2.39
Serial NUC	134.00	229.00	34.00	216.00	480.00	57.00
Balanced NUC	13.48	23.00	3.42	21.73	48.18	5.74

Table 6.31: Average calculation time analysis for IV sensitivity calculation

Method	Average Calculation Time (sec)					
	Binomial			Trinomial		
	VBA	VBS	EXE	VBA	VBS	EXE
Serial PC	57.27	101.55	22.64	134.36	167.00	29.09
Imbalanced PC	5.06	8.96	0.80	11.87	14.68	2.00
Balanced PC	4.89	8.84	0.70	11.47	14.30	1.87
Serial NUC	133.00	226.00	50.00	294.00	311.00	57.00
Balanced NUC	13.38	22.71	5.03	29.52	31.27	5.73

Table 6.32: Average calculation time analysis for IR sensitivity calculation

Method	Average Calculation Time (sec)					
	Binomial			Trinomial		
	VBA	VBS	EXE	VBA	VBS	EXE
Serial PC	56.18	100.73	22.36	133.82	165.73	27.36
Imbalanced PC	5.08	9.11	2.02	12.11	14.96	2.47
Balanced PC	4.83	8.75	1.83	11.39	14.28	2.15
Serial NUC	133.00	223.00	50.00	288.00	305.00	57.00
Balanced NUC	13.36	22.46	5.05	29.03	30.65	5.75

6.11 Chapter Summary

The main goal of implementing the dedicated calculation grid is to facilitate the research and development team and the quantitative analysts and researchers to test and simulate different compute-intensive internally developed financial models and trading scenarios in real time. Currently, this process is performed by using a single workstation or a server, and in certain cases, the full analysis takes many hours to a few days to complete. Hence, some of the analysis is carried out as a batch process during nights and weekends, and in addition, sampling frequency is reduced to improve the calculation time. By implementing the dedicated calculation cluster grid for compute-intensive calculations, the test results have shown considerable improvement in reducing the overall calculation time. The tests and simulations that were carried out on the dedicated calculation clusters are similar to the scenario analysis carried out by the researchers and quantitative analysts, but on a reduced scale with small sets of test data. The test result has proved that implementing the dedicated calculation cluster with appropriate application- and hardware-specific load balancing algorithms greatly improves the calculation efficiency for many of the bespoke mathematical models used in the company. Hence, this chapter has demonstrated the original contribution to the design of specific dedicated calculation clusters to improve the calculation efficiency of particular type of compute-intensive applications.

The advantage of the dedicated calculation cluster grid is that unlike the workstation cluster that utilises the user workstations, it can be fully utilised 24x7. The dedicated calculation grid is isolated from user access and fully dedicated for certain applications that require extensive testing before being used in the live-trading environment. The test have shown that the NUC cluster has many advantages compared with the PC cluster; even though the NUC computers are less powerful than the PCs used in the PC cluster, the NUC computer has various advantages such as low power consumption, smaller size, and lower cost. However, the PC cluster is useful for testing different types of load-balancing algorithms due to its nonlinear hardware configurations, but the PC cluster is not suitable for continuous use in production environments due to its

high power consumption, requirement for larger space, and higher rate of failure. Meanwhile, the NUC cluster be expanded and improved by adding more calculation nodes in the future for long-term research and development in the company. Both clusters are undergoing continuous improvement to fine-tune their performance for various calculation scenarios, and more tests and simulations are being performed on these clusters. A number of changes have been made to the distributed processing controller software and calculation node's controller software to include a dedicated calculation grid cluster, and in addition, there have been a few changes made to the SQL server database to incorporate the cluster nodes within the distributed processing control system. However, further improvements required to the distributed processing controller software and to the SQL server database to incorporate different types of calculation cluster configurations. Meanwhile, the message passing between calculation nodes and the distributed processing controller is similar to the technique used in workstation cluster that explained in Chapter 4 with a few modifications and proved to be adequately support the communications requirements for coarse-grain process-based calculations. Further analysis of both PC and NUC clusters using captured data is presented in Chapter 8 where the results and discussions are presented in detail.

7 Hybrid Processing Using Multiple Calculation Clusters

7.1 Introduction

This chapter describes the original design and implementation of distributed processing using multiple physical computing devices and logical CPU cores as calculation nodes to form a hybrid type of calculation cluster to perform a combination of distributed processing and parallel processing. This is the fourth phase of the investigation that explained in section 1.4 in Chapter 1. This method is one of the approaches to investigate the feasibility of hybrid processing to create multiple calculation clusters using available processing devices within the company. These calculation clusters can be configured to work as a single cluster or a combination of clusters as groups depending on the applications used, and are able to be utilised as a single cluster with data decomposition-based processing, multiple clusters with functional decomposition-based processing, or a combination of both depending on the requirements. The main purpose of the hybrid type of distributed processing investigation is to design high-throughput distributed processing cluster systems that utilise all the available processing devices within the company such as servers, workstations, PCs, and single-board small form factor computers in the main office as intelligent processing devices to facilitate the complex and time-consuming calculations. In addition, the goal is to be able to incorporate the disaster recovery (DR) site's hardware and software to be integrated as part of the distributed processing system that facilitates compute-intensive applications to be executed within manageable time scales and to maximise the processing capacity.

The clusters are designed based on consolidating various processing devices into an intelligent calculation cluster that can be used for various operations with high throughput and utilises an adaptive and self-tuning control mechanism that continuously fine-tunes the cluster's performance to permit it to function autonomously. The hybrid cluster is designed in such a way that it can be modified whenever required with minimum impact to the business. In addition, implementing

different methods and techniques improves the efficiency of the distributed processing cluster using a combination of distributed processing and parallel processing with CPU-core-level program executions and static and dynamic load balancing techniques employing multi-core processors. For testing the hybrid cluster performance, some of the currently used applications are employed, and these applications are developed using Windows-based programming languages. Hence, the testing is performed to evaluate the suitability and usability of these applications within the distributed system that have multiple hybrid types of clusters.

Different types of distributed processing approaches are applied to the existing systems to solve complex and compute-intensive calculations, and some tests are performed to evaluate the data processing capabilities of the distributed processing systems that use a combination of workstations, servers, and SQL servers. Two applications used to test the performance improvements by using distributed calculation methods: the dispersion trading system that uses the market volatility to trade certain derivative instruments and the AH trading system that uses security-pricing discrepancies within different exchanges. Both systems are computationally intensive and require continuous calculations with high accuracy for executing live trades during the volatile market conditions. Hence, combinations of distributed processing methods are tested and the results show considerable improvements in calculation time. The tests are performed using prototype systems with real-time data feeds from financial data feed (FDF) providers. However, due to the complex nature of the trading involved in these types of live-trading systems, it requires further rigorous testing under stressed conditions to ensure that the proposed distributed systems are capable of withstanding the highly volatile market conditions.

A number of changes have been made to the distributed process management software to include different processing devices as calculation nodes. In addition, the required modifications are made to the distributed processing controller SQL database to support the database-level changes needed to incorporate the hardware and software as a single distributed processing unit or logically segregated multiple calculation

clusters. Further investigations are carried out for implementing dedicated calculation clusters using small form factor multi-core and many-core computers with lower power consumption and smaller size with low heat dissipation characteristics. Alongside that development, small-scale dedicated calculation grid-type clusters are implemented to facilitate the research and development team and quantitative researchers to test and simulate financial models and trading scenarios in real time. A number of tests and simulations are carried out on different types of processing clusters as a single cluster and also as a combination of clusters to evaluate their calculation performance with and without load balancing. In addition, the testing also provides detailed information regarding various parameters that are used for comparing the cluster performances and their advantages and disadvantages under different conditions. The test results show that using multiple clusters with varying configurations coordinated as a hybrid distributed processing cluster has many advantages for complex and compute-intensive applications. Furthermore, it is possible to implement these types of clusters for real-time trading applications.

7.2 Hybrid Distributed Processing

The hybrid processing system design is based on utilising both distributed processing and parallel processing systems using existing hardware within the company. The consolidation of different processing units such as workstations, virtual servers, spare PCs, and servers forms a distributed calculation cluster that is capable of processing at CPU-core-level calculation nodes. Virtualisation and multi-threading techniques are used in conjunction with discrete CPU-core-level processing to design and build calculation clusters for research and development in the company. The design approach is to consolidate various processing units into an autonomous self-managed calculation cluster that can be used continuously or for on-demand operations with high throughput. The hybrid system utilises all the processing devices available in the main office, and in addition it has the capability to incorporate the processing devices in the disaster recovery (DR) site to form intelligent processing devices to facilitate the complex and time-consuming calculations to be executed within manageable time scales and to maximise the processing capacity.

7.3 Cluster Configurations

An alternative approach is made to design a high-throughput distributed processing cluster computer system that utilises a combination of all the processing devices. It uses the workstations, PCs, different types of servers, and small form factor computers as separate, mutually exclusive clusters that can work as intelligent processing devices to facilitate the compute-intensive calculations to be executed within manageable time scales and maximise the processing capacity. The calculation clusters are configured in a way that they can act as a single cluster or as part of a multiple-cluster system depending on how the distributed processing controller manages these clusters. By consolidating different clusters into an intelligent calculation cluster that can be used during out-of-office hours when most the resources are free to be used at full capacity, this is highly useful for batch processing tasks that take many hours to complete in a single server. An adaptive and self-tuning control mechanism is implemented that continuously fine-tunes the cluster performance and is able to function autonomously with varying parameters. The varying parameters can be hardware, software, or business rules. The cluster design approach has various degrees of flexibility due its bespoke-type design; hence, it can be modified with minimum impact to the business. The consolidated calculation cluster configurations are based on the following:

- Workstations are configured as virtual machines using VMware software.
- Noncritical servers are set up as calculation nodes.
- Surplus conventional servers are set up as calculation nodes.
- Cluster controllers are modified for coordinated process management.
- SQL servers are configured for distributed data and query processing.
- Clusters are designed using adaptive processing for efficient and robust operation.
- CPU cores are used as calculation nodes with load balancing techniques.
- Historical data are collected for adaptive load balancing and task scheduling.
- Static and dynamic load balancing algorithms are implemented.
- Various methods and techniques are implemented to improve the efficiency of the distributed processing.

The following tests are performed on consolidated calculation clusters:

- Various applications are used to monitor hybrid cluster performance.
- Various tests are performed, and collected data is used to analyse cluster performance under different load conditions.
- CPU core-level process executions using different applications and static and dynamic load balancing.
- Control techniques applied to manage the process execution within calculation nodes to avoid process deadlocks.
- Virtualisation techniques are used for workstations and servers.
- Multi-core processing and multi-thread techniques are employed.
- Parallel processing is performed in a single workstation using MS-Excel applications.
- Virtualised servers are used as dedicated calculation nodes.
- Distributed data and query processing are implemented using local SQL database servers and linked SQL servers.
- A combination of hardware is consolidated as physical and logical calculation clusters.

The followings are investigated using the consolidated hybrid clusters:

- The systems that are currently used have been developed using VB6, VBA, VBScripts SQL server, and batch processing systems. How these applications can be incorporated within the distributed processing system at no extra cost or at minimal cost is investigated.
- Implementation of reliable, robust, and efficient distributed processing system for different type of processor-intensive calculations at minimal cost. Currently, these calculations are performed as batch overnight and weekend processes.
- Incorporating Disaster Recovery (DR) site hardware and software as part of the overall distributed processing system.
- Use the distributed processing system in the real-time trading environment.

The Northwest distributed systems must have certain characteristics, hence, the overall design and development is focused on complying with the following characteristics:

- The system should be scalable with a number of calculation nodes and able to add extra calculation clusters with minimum disruption to the business operations.
- The system should be robust enough to any node failures of various types and should be able to manage the failure gracefully and the system continues to operate and delivers required critical calculations in the presence of node failures.
- The system should be extensible for different types of data and applications that are used within the systems.
- The system should have minimal management overheads and manual configurations.
- The system should be portable to a variety of applications that are used in the company and able to upgrade the hardware and software with minimum disruption to the systems.
- The system should provide low per-node overheads for all scarce computational resources including CPU, memory, I/O, network bandwidth, power consumption, and space required.

Figure 7.1 shows the high-level diagram of multiple cluster configurations. The multiple cluster-based distributed processing systems developed is partly being used for the following applications, and it will be improved and fully implemented to various compute-intensive applications that are currently used by the company once the distributed processing system is fully tested under strict conditions.

- Used for pricing of different types of existing and new derivative instruments.
- Used for calibration of existing financial models and developing new financial models.
- Used for risk data calculation, stress testing, and scenario analysis.
- Used for valuation of portfolios using varying data for scenarios analysis.
- Used for research and development for bespoke system designs.

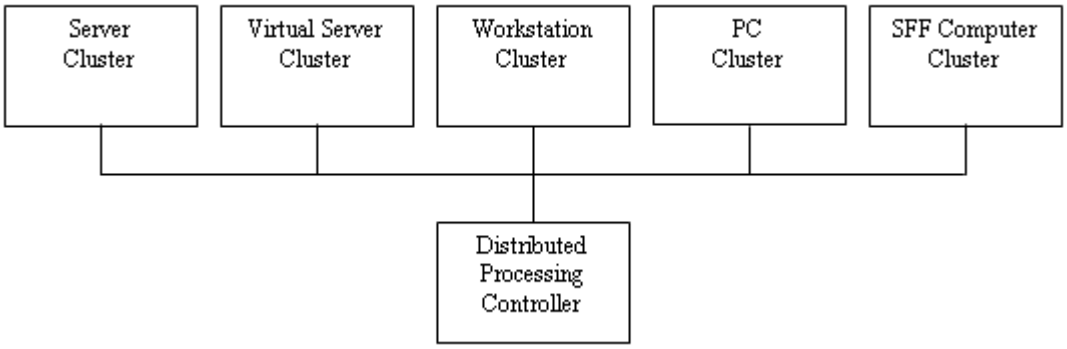


Figure 7.1: High-level diagram of multiple cluster configurations

7.4 Server Cluster

The server cluster is built using surplus conventional servers that are not used as part of the company network servers; these servers were used in the past and replaced with later versions of blade servers. However, these servers are in fully working condition with all the required server software licenses. Hence, these servers can be used as a dedicated calculation cluster for performing batch process tasks. The original proposal was to build a server-based calculation cluster using servers and the server cluster was to be located in the disaster recovery site and housed in a separate server rack. However, test results shows that SFF computers with many cores are highly suitable for building cost-effective and highly efficient calculation clusters. Meanwhile, the server cluster is useful for testing static and dynamic load balancing techniques by utilising various hardware and software configurations and testing the CPU core-level distributed processing. Hence, the server cluster is used for testing purposes only and is not to be used in the production environment. For testing, five of these servers are used as calculation nodes connected via gigabit Ethernet switch to the distributed processing controller, and each server is configured as a single node. Figure 7.2 shows the server cluster configuration. Table 7.1 lists the parameters of each server node. All severs nodes are loaded with the Windows 2008 operating system and Office 2010 software for testing. Server cluster's pictures are shown in Appendix C (Figures C.1-a, C.1-b, and C.1-c).

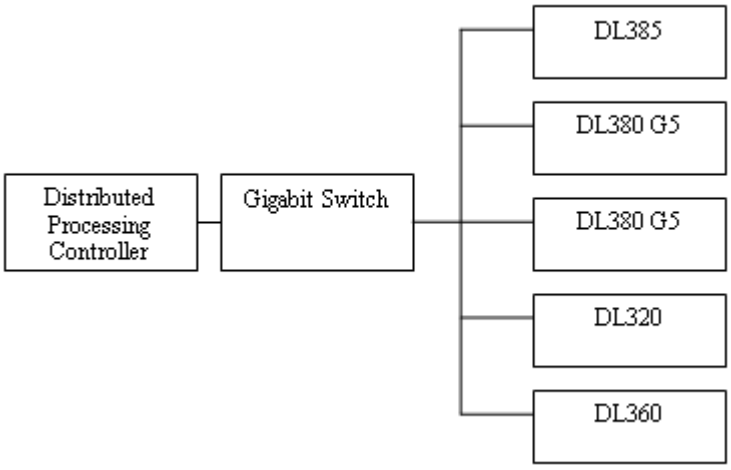


Figure 7.2: Server cluster configuration

Table 7.1: Server cluster node parameters

Server	CPU	Memory (GB)	Size (H x W x D) cm	Weight (Kg)	Power (Watts)	Cooling (BTU Hour)
DL 385	1x AMD 2.6 GHz Dual core	1	8.59 x 44.54 x66.07	28	575	2,500
DL 380 G5	2x Xeon 3.5 GHz Dual core	12	8.59 x 44.54 x66.07	27	980	3,350
DL 380 G5	2x Xeon 3.5 GHz Dual core	12	8.59 x 44.54x 66.07	27	980	3,350
DL 320 G2	1x P4 3.06 GHz Single core	4	4.24 x 48.30 x 55.6	12	225	1,000
DL 360 G2	1x P3 1.4 GHz Single core	4	4.19 x 42.55 x 63.5	11.5	200	1,000

To test the server cluster performance, one of the scenario-based calculation systems is used. For scenario analysis calculations, an asset-swap calculation system is used and 500 asset-swaps are selected from the current portfolio for testing the calculation speed improvements using the server cluster. Asset-swap is a special type of CB that behaves like a normal CB with certain constrains and uses the same financial model that described in the section 4.2.1 in Chapter 4. For all 500 asset-swaps, 15 scenarios are calculated. Table 7.2 and Table 7.3 list each server node's calculation time and server cluster's calculation time analysis, respectively. Using a single DL360 G2 server, calculating 500 asset-swaps with 15 scenarios takes 6 hours 38 minutes to complete. Using a single DL360 G5 server, calculating 500 asset-swaps with 15

scenarios requires 2 hours 8 minutes to complete. Meanwhile, by distributing the calculation using static load balancing techniques using the domain decomposition method, the calculation time is reduced to 35 minutes and 15 seconds. Figure 7.3 shows each node's calculation time for load imbalanced and load balanced conditions.

Table 7.2: Calculation time analysis for each server node

Node	Sever	Single Node Calculation Time (hours)	Distributed Calculation Time (min)	Distributed Calculation Time with Load Balanced (min)
1	DL 385	3.08	37.67	33.75
2	DL 380 G5	2.21	27.48	34.25
3	DL 380 G5	2.13	26.87	35.03
4	DL 320 G2	2.56	32.14	34.75
5	DL 360 G2	6.64	80.57	35.25

Table 7.3: Distributed calculation time analysis for server cluster

	Load Imbalanced Calculation Time (min)	Load Balanced Calculation Time (min)
Maximum	80.57	35.25
Minimum	26.87	33.75
Average	40.95	34.61
STDEV	22.57	0.61

Even though the calculation time improvement is considerably high, the practical implementation of these types of clusters is not a suitable for long-term usage due to the following reasons:

- Power consumption is high.
- Larger space is required.
- Dedicated server rack is needed.
- Operating noise level is high.
- Dedicated temperature controlled server room is required.
- Cost of spare parts is high and unavailability is common.
- Dedicated hardware and software management is required.
- Cost of operations and maintenance is high.

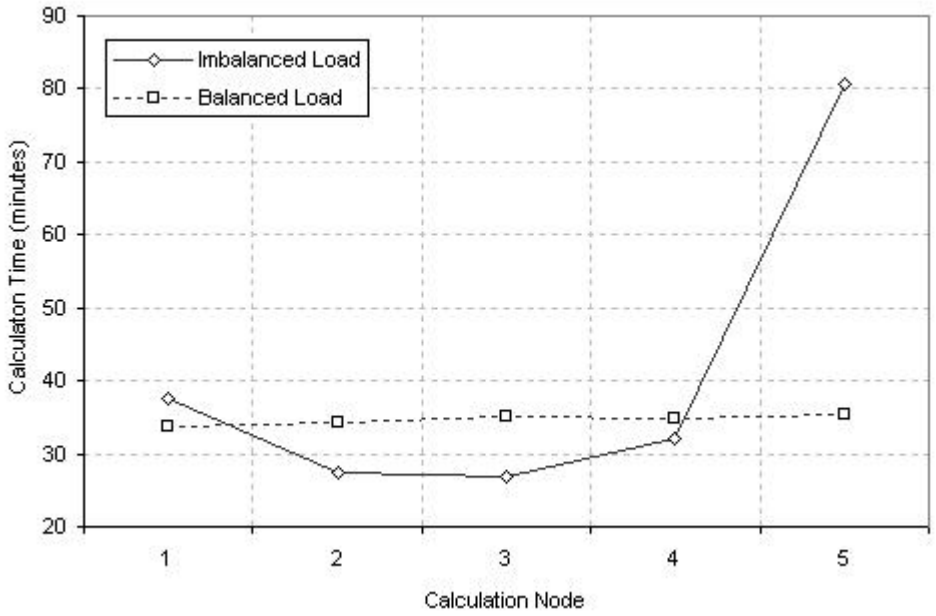


Figure 7.3: Each node's calculation time for load imbalanced and load balanced conditions

Due to various restrictions and limitations of using surplus servers and PC calculation clusters, these clusters are not viable for long-term use. Hence, these clusters are only to be used to test the distributed processing methodologies and load balancing techniques that can be utilised for distributed processing; therefore, these clusters are set up as prototype test clusters and will not be developed further. Meanwhile, small form factor computer (SFF)-based clusters will be developed further to improve the cluster performance by using various distributed processing and load balancing techniques. In addition, a number of dedicated calculation clusters will be built using SFF type computers specifically for analysts and researchers.

7.5 Virtual Server Cluster

The virtual server cluster is built using the virtual servers that are currently used in the company. These servers span across multiple physical servers and share the hardware resources as a pool, and the company has a combination of physical and virtualised servers for various dedicated functions. Apart from SQL and application servers, all other servers consume minimal CPU and memory for their operations. Hence, these servers are good candidates for use as server-based calculation nodes for logically separated calculation clusters and can be fully utilised during out-of-office hours. Because the server hardware is more reliable and powerful than the workstations, these servers are useful for setting up as logically separated calculation clusters that are mainly used for calculation-intensive batch processing tasks. The company has a number of blade servers configured as a blade server cluster within two blade server cases. These blade servers are connected to two storage area network (SAN) systems. Currently, 14 virtual servers are running on a blade server cluster and a number of virtual servers dynamically changeable depending on the requirements. The blade server cluster consists of 13 HP BL460C G8 blade servers, Table 7.4 lists the server parameter, and Figure 7.4 shows the blade server configuration as virtual servers. Virtual server cluster's picture is shown in Appendix C (Figure C.1-d).

Table 7.4: Virtual server names and their roles

Server Name	Server Role	Model	CPU	Memory
NHKSRV02	Email Server	BL460C G8	Xeon 2.9GHz	32 GB
NHKSRV04	SQL Server	BL460C G8	Xeon 2.9GHz	32 GB
NHKSRV05	File Server	BL460C G8	Xeon 2.9GHz	32 GB
NHKSRV06	SQL Server	BL460C G8	Xeon 2.9GHz	32 GB
NHKSRV07	Terminal Server	BL460C G8	Xeon 2.9GHz	32 GB
NHKSRV08	SQL Server	BL460C G8	Xeon 2.9GHz	32 GB
NHKSRV09	Application Server	BL460C G8	Xeon 2.9GHz	32 GB
NHKSRV14	SQL Server	BL460C G8	Xeon 2.9GHz	32 GB
NHKSRV15	Domain Controller	BL460C G8	Xeon 2.9GHz	32 GB
NHKSRV16	FTP Server	BL460C G8	Xeon 2.9GHz	32 GB
NHKSRV17	Terminal Server	BL460C G8	Xeon 2.9GHz	32 GB
NHKSRV19	Web Server	BL460C G8	Xeon 2.9GHz	32 GB
NHKSRV28	Management Server	BL460C G8	Xeon 2.9GHz	32 GB
NHKSRV34	Backup Server	BL460C G8	Xeon 2.9GHz	32 GB

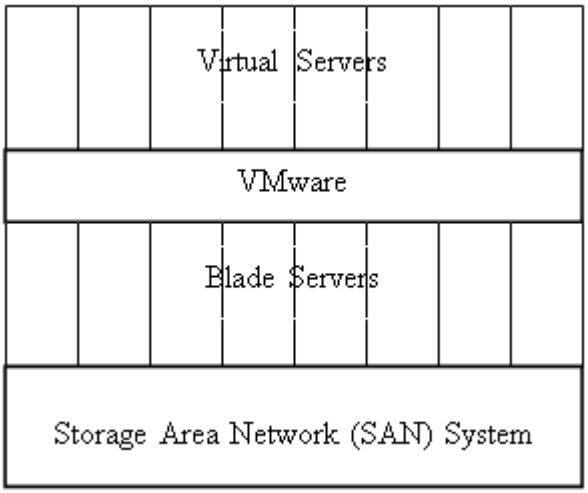


Figure 7.4: Northwest's blade server configuration

For testing the distributed calculation performance on the virtual server cluster, 560 asset-swapped positions are used for distributed calculation. The calculation setup is similar to the conventional server cluster that was described earlier in section 7.4; however, the number of scenarios is limited to a single scenario rather than all 15 scenarios. The blade servers are live production servers, and any errors that cause the servers to slow down or require reboot will cause serious business disruption during office hours. Therefore, a single scenario is used with all 560 asset-swapped positions, and the load distribution is based on the number of asset-swaps rather than scenarios that are implemented in the conventional server cluster. Each server is used for the calculation of all 560 asset-swapped positions. This is intended to test the server performance compared to different virtual severs.

- Split the asset-swap positions to different servers on a *pro rata* based to evaluate the calculation time reduction by using multiple servers as a cluster.
- Allocate each server an appropriate number of asset-swap positions using static load balancing techniques.
- Perform calculation tests during office hours and out-of-office hours to compare server performance.

Table 7.5 lists each server's node calculation times for full load and distributed load conditions, and Table 7.6 lists the virtual server cluster's calculation time analysis. Figure 7.5 and Figure 7.6 show calculation times using balanced and imbalanced loads respectively. The behaviour of the virtual server is different from the conventional server; this is due to the virtualisation software and operating system continuously monitoring the server CPU and memory usage, and allocating memory and CPU processing capacity accordingly from a shared pool of memory and CPU cores.

Table 7.5: Calculation times for each server node for the full load, distributed load without load balancing, and distributed load with load balancing

Calculation Node	**Server Name**	**Single Node Calculation Time (min)**		**Distributed Calculation Time (sec)**		**Distributed Calculation Time with Load Balanced (sec)**	
		T1	**T2**	**T1**	**T2**	**T1**	**T2**
1	NHKSRV02	8.05	7.33	34	31	37	29
2	NHKSRV04	9.45	6.89	40	28	38	29
3	NHKSRV05	8.53	7.85	36	33	38	30
4	NHKSRV06	10.57	6.11	42	25	38	29
5	NHKSRV07	11.2	7.12	48	29	37	29
6	NHKSRV08	10.72	6.57	44	28	37	29
7	NHKSRV09	9.62	7.06	40	30	38	29
8	NHKSRV14	11.68	6.88	48	28	37	29
9	NHKSRV15	7.05	6.91	29	28	38	29
10	NHKSRV16	7.53	6.82	32	28	38	29
11	NHKSRV17	10.73	7.55	46	31	38	29
12	NHKSRV19	8.5	7.43	36	30	38	28
13	NHKSRV28	8.15	7.63	34	32	37	30
14	NHKSRV34	7.25	7.1	31	30	37	29

where

T1 Calculation time during office hours

T2 Calculation time during out-of-office hours

Table 7.6: Virtual server cluster's calculation time analysis

	Office Hours Calculation Time (sec)		Out-of-Office Hours Calculation Time (sec)	
	Imbalanced	**Balanced**	**Imbalanced**	**Balanced**
Maximum	48	38	33	30
Minimum	29	37	25	28
Average	38.57	37.57	29.36	29.07
STDEV	6.36	0.51	2.06	0.47

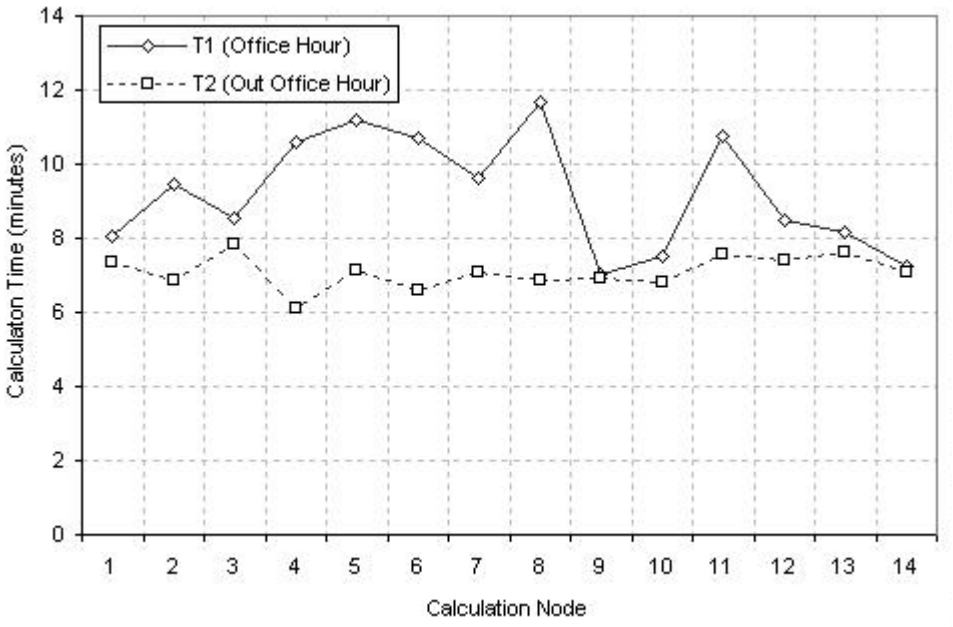


Figure 7.5: Node calculation times of each server for all 560 assert swaps during office hours (T1) and out-of-office hours (T2)

The calculation time improvement is observed as expected, and the out-of-office hour (T2) calculation is more stable due to all the hardware and software resources being fully available for distributed processing. Meanwhile, the office hour (T1) calculation is less efficient due to the virtualisation software and operating system dynamically allocating various hardware resources to virtualised servers.

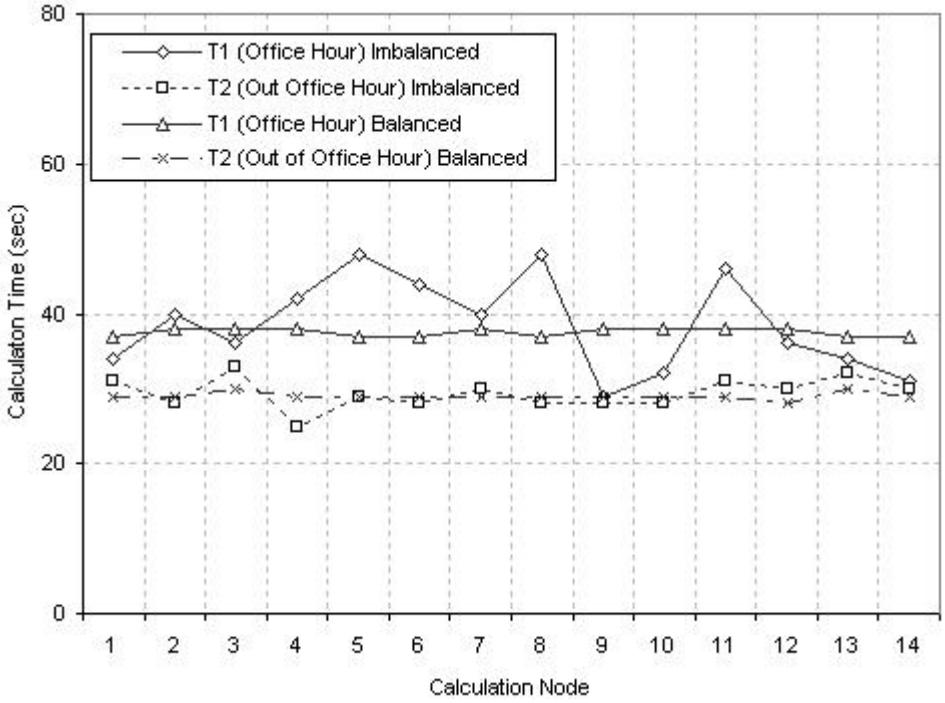


Figure 7.6: Node calculation times of each server for distributed loads during office hours (T1) and out-of-office hours (T2)

7.6 Distributed Data Processing

Distributed data processing is one of the requirements for the company's distributed processing systems to be used to process various data feeds within the shortest time possible. The primary reason for this requirement is due to the recent data-feed license changes that were introduced by the financial data feed (FDF) company. The primary financial data feed provider has introduced various data protection and data usage terms and conditions. This has had a profound impact on the company's day-to-day business and requires some sophisticated solutions to overcome the limitations imposed by the FDF without violating the terms and conditions of data usage. The company has 11 FDF terminals, and each costs $20,000 USD per year. These terminals are used by traders, fund managers, risk and quantitative analysts, researchers, and back office and middle office staff for various business functions using the FDF-provided data. In recent years, the FDF has been introducing continuous changes to its data usage terms and conditions to restrict the use of each FDF terminal's data to various degrees. The following two terms and conditions were recently introduced by the FDF and are main concerns that need to be addressed:

- Each FDF terminal must be used by the assigned FDF user, and any data used or downloaded must be kept as local data and must not be shared.
- A defined data download limit for each FDF terminal for each 24-hour period of use, and in case the data downloaded exceeds the limit, there will be an extra fee based on the number of data points downloaded.

Hence, due to these current restrictions, downloaded data cannot be saved in the common database and a limitation on the number of data points can be downloaded each day. The reason behind these data usage changes is based on each FDF terminal being licensed to each user, not to the company as a whole. Hence, if the company wants to do any companywide-consolidated data processing, then another data license scheme provided by the FDF for the company and that will cost $200K USD per year. For a small company like Northwest, these types of licenses are not needed in general; however, for research and development purposes, a need for some type of data consolidation. Therefore, it is necessary to consolidate the FDF data without violating the data usage terms and conditions. One method to overcome the data usage problem imposed by the FDF is to process the data in each FDF terminal and save it in the local hard disk by using a distributed data processing cluster that consists of all the FDF terminals as a single cluster. Hence, all the processing that is related to each terminal is executed on one particular terminal only. In addition, the terminal consolidates the results using the distributed query method. Due to on-going negotiation with the FDF regarding data saving and data usage terms and conditions, the distributed data processing configurations are currently set up as prototypes and only used for testing purposes. Once the terms and conditions are negotiated, then the systems will be activated for live trading operations.

For testing, followings setup is used:

Number of security:	700
Number of fields per security:	11
Number of days:	3500

The FDF terminal service has various data limitations, such as, how long the terminal applications can be used on a single session, how much historical data can be allowed per 24-hour period. In addition, FDF applications have their own built-in functions that are linked to the FDF APIs, and these functions are commonly used with MS-Excel sheets. The data limitation is calculated depending on various factors such as whether the data requested is the current data or historical data, direct market feed, or FDF-calculated data. Hence, each data processing node has to be checked for available data points before distributing the required calculation data points to each data processing node. By implementing load balancing, data overload is avoided; data overload can cause serious problems for the node concerned, and it can lock the FDF terminal service for the next 24 hours on that particular node. Meanwhile, different FDF terminals have different data usage profiles depending on type of application is used and the FDF functions used within the applications. Hence, to avoid data overload, a limitation factor *k* is introduced to each data processing node depending on the criticality of the FDF terminal. For example, traders' terminals are highly critical and operation staff members' terminals are less critical. The value of *k* is set to 0.5 for highly critical terminals, that is, only 50% of the available data points will be used, and for less critical terminals, *k* is set to 0.8, that is, only 80% of the available data points will be used. Therefore, by implementing a limitation factor, a safety buffer zone will be created that protects the data overload scenario. For testing, five user workstations with FDF terminals are selected to form a data processing cluster with each workstation set up as a data processing node and connected to the distributed processing controller. Figure 7.7 shows the distributed data processing cluster configuration using FDF terminals, and Figure 7.8 shows the configuration of each FDF terminal.

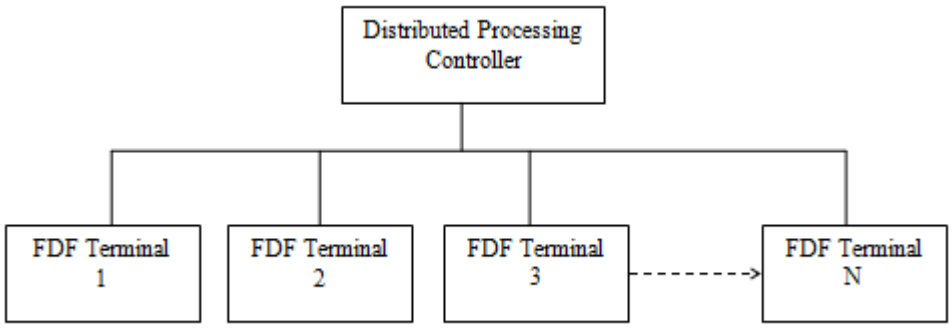


Figure 7.7: Distributed data processing FDF terminal configuration

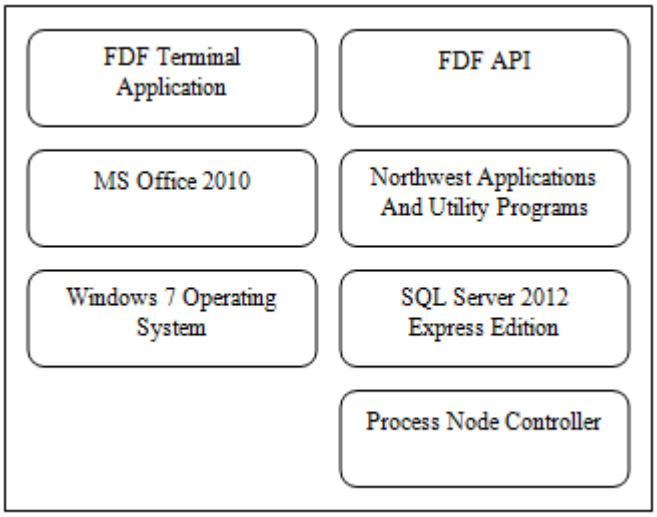


Figure 7.8: Terminal configuration of each FDF

Total number of data points for a historical data analysis is shown in Equation (7.1), and ticker allocation for each data processing node is evaluated using Equations (7.2) and (7.3).

$$P = T \times D \times F \tag{7.1}$$

where

P Total data points needed for calculation

T Number of security tickers

D Number of historical days

F Number of FDF fields per security ticker

$$T(i) = \frac{P}{(D \times F)} \times \left[\frac{k(i) \times (B_M(i) - B_U(i))}{\sum_{i=1}^{N} k(i) \times (B_M(i) - B_U(i))} \right] \tag{7.2}$$

$$B_A = \sum_{i=1}^{N} k(i) \times (B_M(i) - B_U(i)) \tag{7.3}$$

where

$T(i)$	Number of tickers allocated to node i
$B_M(i)$	Maximum usable data points within 24 hours in node i
$B_U(i)$	Used data points within 24 hours in node i
$B_A(i)$	Available data points within 24 hours in node i
$k(i)$	Data limitation factor for node i
N	Number of data processing nodes in the cluster
P	Total data points needed for calculation

If $B_A < P$ then $P = B_A$ and $P' = P - B_A$, where P' is the reminder of the data points that will be carried on to the next day, and P' becomes P on the next day and so on until all the data points are filled. This process may take a few days for large sets of data points, and for smaller sets of data points, the entire process can be completed within a day. The distributed processing controller allocates a number of tickers to each data processing node based on Equation (7.2). The same method can be used for distributing data using days or number of data fields per data processing node; however, for testing purposes, ticker-based distribution is used. Once the ticker-set is allocated to each data processing node, then the distributed processing controller sends a message to each data processing node to start the data processing. For each ticker, three processes are performed:

(1) Fetch data from FDF for given date and field using FDF functions.

(2) Calculate various parameters using fetched data using Northwest programs.

(3) Save the calculated data to local SQL database and local XML file.

These processes repeat until all the tickers have been calculated. By using a set of tickers to download the data using a single FDF terminal with 100% use, seven days are required to download all the required data. However, employing distributed data processing using five FDF terminals with 80% download limits takes 2.5 days, and 11 FDF terminals with 50% download limits only requires 1.5 days. Even though, 4.7 times improvement for 11 terminals used, data feed delays are also involved and that is not included in the calculations; hence, this will also affect the download speed.

7.7 Distributed Query Processing

Distributed query processing technologies have been implemented in various data processing centres using processing clusters in different forms. However, Northwest's distributed query processing has to be simple to use and easy to implement within the nodes. The reason for these requirements is that currently all the historical data based query processing is executed during weekend or out-of-office hours. This is due to the large amount of saved historical data that is kept in various SQL server databases, and these data have to be linked using a TCP/IP network to perform complex calculations. This is causing considerable time delays, SQL server connection timeouts, SQL server deadlocks, and network traffic delays. To solve these issues, two options are investigated:

- Consolidate all the historical data into one SQL server database and separate the historical data in a dedicated SQL server. This will require a major overhaul of company's database structures and require considerable time, developer resources, and redesign costs.
- Use the existing databases and implement distributed query process techniques to distribute the data across dedicated and non-dedicated calculation nodes and execute queries on each set of data in parallel.

At the current state, the distributed query processing is most suited for the company due to a number of constraints in deploying new database systems. The company uses Microsoft SQL Server Enterprise edition as the main database system, and a number of different editions of SQL server available with different licensing structures. One of them is SQL Express edition, and that is free to use. The core part of the SQL server engine is the same for all the SQL server editions, and only a number of user limitations and certain add-ons are different for each edition. Therefore, each calculation node within the distributed processing cluster is installed with SQL Server Express edition and set up as a data processing node with no extra licensing cost.

Two ways of distributing the data between distributed databases: the distributed data method and the linked database method. Each has its own advantages and disadvantages depending on how the query is implemented.

7.7.1 Distributed Data Method

In the distributed data method, the master data is replicated with the distributed databases. This method is most suited for producing reports using snapshot data from the master database or historical data analysis.

Advantages:

- Queries execute faster due to the local databse utilisation.
- Minimum network-related delays.
- Most suited for historical data analysis.

Disadvantages:

- Data has to be replicated continously and synchronised with the master database.
- Possible data discrepancies between replicated data and master data.
- Not suitable for live data processing.

7.7.2 Link Database Method

In the link database method, the master database is linked to each distributed database. This method is most suited for producing reports using current data from the master database or live data analysis.

Advantages:

- Direct access to the master data and no need for replication.
- Mostly suited for live data processing.

Disadvantages:

- Queries execute slowly due to network latency and data paging.
- Complex nested subqueries are not suitable due to long delays and timeouts.
- Network-related delays that can cause deadlock on larger datasets.

Currently, most of the processing for various anlysis done by quantitative researchers is based on historical data, that is, saved data that does not change during the day. Hence, the distributed data method is most suitable for this type of processing. However, a number of situations in which live and current data are needed produce analysis reports for various reasons; for this purpose, up-to-date data are required. Therefore, each calculation node is set up with a single instance of SQL Server Express edition and two separate local databases, one with linked tables that is linked to a master database and another with local tables that are replicas of the master tables. For a distributed data configuration, another local database is used for the local tables to be partitioned using the master table for distributed query processing. Table partitioning can be vertical, that is, spliting table columns, or horizontal, that is, spliting table rows, or both. The company has many historical data tables that hold various levels of historical data for the last 18 years; the number of rows exceeds 40 million while the table columns are fewer then 255. Hence, the horizontal partitioning is the most suitable solution for distributing data across the processing nodes. By using dynamic SQL views in each processing node, the same SQL stored procedure can be used in the master database or in the processing node. Dynamic SQL views are managed by the distributed processing contoller using data structure rules and metadata tables. For example, a master table data can be divided for distributed processing by using SQL view to logically divide the data or using partitioned tables to physically divide the data.

Therefore, by implementing table partitioning and programmable views in each processing node, it is possible to execute queries and SQL stored procedures that are currently used within the company without changing the structure of these objects. This approach has many benefits, such as, no need to alter the exsting SQL server

objects in the master database and only the distributed processing part of the development has to be modified to fit the current SQL server applications. Most part of the distributed query processing is managed by the distributed processing management controller coordinating with the distributed processing SQL database that maintains the partitioned tables' metadata for each processing node. This is a simple example to illustrate the concept of distributed query processing using replicated and distributed data. However, for complex queries, the benefit of using distributed data query processing has better performance enhancement.

To test the time improvements of executing an existing SQL stored procedure in a distributed query peocessing cluster, 10 workstations are selected and configured as processing nodes and the distributed process controller database is modified to maintain the partitioned table metadata details for managing the query processing in each processing node. Each processing node is installed with SQL Server Express edition and has three local databases: one is the linked database in which all the tables are linked to the master database, another has local tables that are copies of the master tables, and the third one has corresponding partitioned tables and programmable views. The advantage of keeping a copy of the master tables and all the partitioned tables in every processing node is that all the processing nodes have an exact copy of the master tables and corresponding partitioned tables. Hence, the distributed processing controller can allocate query processing tasks to each processing node depending on their availability and CPU load parameters regardless of their data status. The data updates and synchronisation are managed by the distributed processing controller's SQL server in coordination with the master SQL database, and this process is set as a daily overnight scheduler process. However, the data synchronisation process can be on demand or continous as live synchronisation, and the data update and synchronisation process replicates the master table's data for each processing node. This method is well suited for smaller datasets with few cluster nodes; however, if the data is considerably large and the number of cluster nodes is also high, then this method is not a viable solution due to the time taken to update and synchronise the data across the processing cluster.

Table 7.7 lists distributed processing local database details, Figure 7.9 shows the distributed query processing cluster configuration, and Figure 7.10 illustrates the query-processing node's configuration.

Table 7.7: Distributed processing local database details

Database Name	Description
NW_SQL_DB1	Linked database and the data tables are linked to master database.
NW_SQL_DB2	Local copy database and all the data tables are replicated with master database.
NW_SQL_DB3	Local partitioned table database and all the data tables are synchronised with master database.

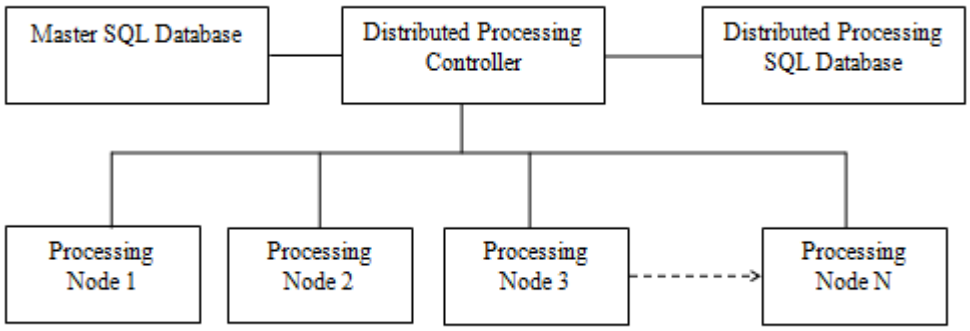


Figure 7.9: Distributed query processing cluster configuration

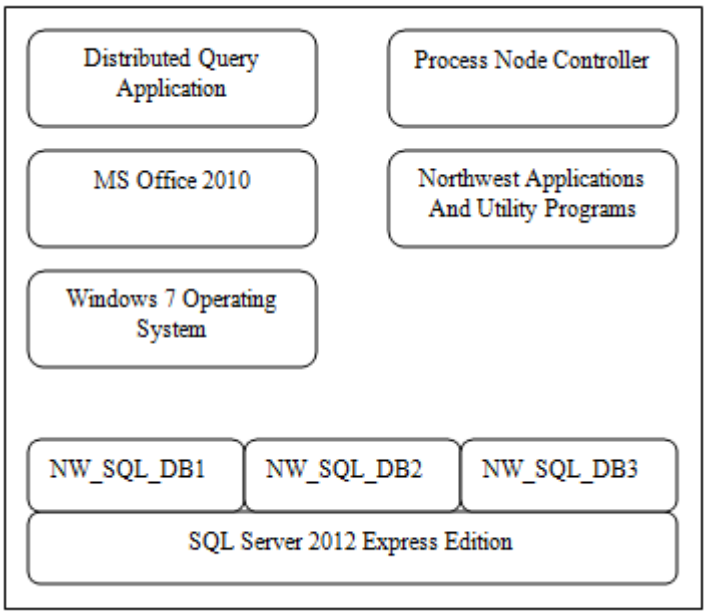


Figure 7.10: Distributed query processing node configuration

For testing, three separate SQL databases in the single instance of the SQL Server, and the security settings for each SQL object are managed by administrator-level authentications. Table 7.8 and Table 7.9 list the distributed and replicated query processing times for each processing node and cluster-level analysis, respectively.
For testing the query processing, the following data tables are used:

- Price table with 20 million rows and 12 columns
- Security table with 20,000 rows and 125 columns
- Trade table with 700,000 rows and 182 columns

Table 7.8: Distributed and replicated data query processing time for each node

Partitioned Table	Data Year	Processing Node	Distributed Processing Time (sec)	Replicated Processing Time (sec)
Table(1)	2004	NHKWST58	252	765
Table(2)	2005	NHKWST59	250	759
Table(3)	2006	NHKWST62	261	777
Table(4)	2007	NHKWST63	263	754
Table(5)	2008	NHKWST65	275	784
Table(6)	2009	NHKWST68	271	785
Table(7)	2010	NHKWST70	289	769
Table(8)	2011	NHKWST77	294	758
Table(9)	2012	NHKWST89	303	763
Table(10)	2013	NHKWST98	309	781

Table 7.9: Processing time analysis for distributed and replicated data query for each processing node

	Distributed Processing Time (sec)	Replicated Processing Time (sec)
Maximum	309	785
Minimum	250	754
Average	276.70	769.50
STDEV	21.02	11.47

7.8 Multi-Core Distributed Processing

A new type of CPU core-level distributed processing approach is investigated to find a solution for implementing multi-core program execution within the single physical CPU using existing Windows-based technologies. Various ways of utilising multi-core CPUs for parallel and distributed processing using threaded programming, and these types of programs require complete redesign of the existing applications and systems. Hence, this option is not suited for using existing applications and systems that are currently used in the company. The method that investigated is simple to implement within the company's existing infrastructure with minimum change. Hence, no need for complex programming or using third-party software, and it only uses existing functionalities within the operating system. The main advantage of this method is that it is using the existing Windows utilities that are part of the Windows network infrastructure.

Advantages of using the multi-core processing method:

- Implementation is simple within the existing Windows infrastructure.
- For software implementations, no need for specialised programs.
- Programs are written using tools provided by the operating system with no extra costs.
- Troubleshooting is easy during its operations.
- It is highly portable within the Windows environment.
- It is highly suitable for coarse-grain-type task processing.

Disadvantages of using the multi-core processing method:

- It is specific for the company's applications and software.
- It is not suitable for general types of applications and software.
- It can only be used in the Windows network environments.
- It is suitable for fine-grain task-based distributed processing.

A number of modifications are made to the distributed processing controller to incorporate CPU core-level distributed processing, and the calculation node controller allocates tasks to each CPU core depending on the following parameters: Number of physical CPUs, calculation nodes, number of cores in a single CPU, and CPU usage index.

7.8.1 CPU Core Usage Rule

In a normal program execution scenario, the operating system is responsible for allocating tasks to each CPU core depending on the availability; context switching and threads are used. Normally, context switching is activated when all the CPU cores are fully utilised by executing programs; however, it is possible to activate context switching by utilising programs that simulate 100% CPU core usage that will cause the operating system to activate context switching. In the MS Windows operating system, version 7 onwards, the CPU core allocation for each process is handled in a sequential manner, and once all the CPU cores are fully utilised then context switching will be activated by the operating system. Hence, it is possible to segregate each CPU core for dedicated processing without the complex programming that had been used in earlier versions of the MS Windows operating system. To test how the operating system allocates the processes to each CPU core, a VBScript program is used to simulate a CPU-intensive task and every instance of this program execution will fully utilise each CPU core; this process is managed by the operating system. To test how the operating system manages the program execution, the VBScript program is used to simulate high CPU usage and the use of program memory and hard disk usage is negligible. Table 7.10 lists the test workstation specification and Table 7.11 lists CPU core utilisation by operating systems with different numbers of execution instances of the VBScript program. This shows how the operating system is managing the cores that are utilised efficiently using thread management and context switching. Where *a, b, c* are threads used by the operating system for context switching.

Table 7.10: Workstation specification

Parameter	Description
Model	HP Z420
CPU	Four-core Intel Xeon 2.80 GHz
Memory	12 GB
OS	Windows 7 (64)

Table 7.11: CPU core utilisation and context switching details

Program Instance	CPU Utilisation	CPU Core Utilisation	Context Switching
1	25%	4	No
2	50%	4	No
3	75%	4	No
4	100%	4	No
5	100%	4(a, b)	Yes
6	100%	4(a, b, c)	Yes

The test results show that the operating system is managing the program execution using its own rules. Hence, to execute programs in a particular CPU core, the program has to be executed using a shell type of program that uses threading and execution encapsulation or similar. These types of program designs are comparatively complex and require C++, #.Net, or a similar type of programming environment. Therefore, need to implement a solution that is simple to use, easy to manage, and able to support existing applications with minimum changes. Number of third party tools available, but all of them require some sort of programming and also altering the existing programs to work with these tools. However, a tool available within the Windows SDK toolsets called *PsTools.exe* that is part of the system administration tools. Many utilities are available with *PsTools.exe*; particularly, a program called *PsExec.exe* that is used for remote execution within a Windows network; this program is mainly used by network administrators for remote administration. Furthermore, it has a method that is capable of executing programs in a particular CPU core using the processor affinity assignments method.

An enterprise-level remote execute utility available called *RemoteExec.exe* that is an agent-less software solution allowing the execution of predefined remote actions through a graphical interface. It can remotely install applications, execute programs and scripts, and update files and folders on Windows systems throughout the network. *RemoteExec.exe* is a feature-rich enterprise software solution that meets the performance and security requirements of IT professionals managing small to large Windows networks. Meanwhile, *PsExec.exe* is a utility that can be useful for managing a small-sized Windows network and has minimal feature and security requirements. Hence, for testing CPU core-level execution within a single physical CPU or within a small-scale distributed processing cluster, *PsExec.exe* is more suitable. Thus, by using the *PsExec.exe* utility, executable programs can be executed in remote computers or local computers, and the main advantage of using the *PsExec.exe* utility is that it is a part of the Windows network infrastructure.

Advantages of using the *PsExec.exe* utility:

- It is fully supported within the Windows network infrastructure.
- It is easy to implement within the existing network infrastructure.
- It is a lightweight program, can be executed using command line utilities.
- It is a free utility from Microsoft and has no extra cost to implement.
- It executes programs under system accounts.

Disadvantages of using the *PsExec.exe* utility:

- Designed for administrator-level usage; hence, cannot be employed at user level.
- Security features are less credible for network usage.
- Only works with Windows network environments.

The security concern of using *PsExec.exe* is not a serious problem for the company's distributed processing cluster, because the cluster network is set up inside the firewall; hence, no network access to unauthorised users and no remote login from outside the network. To avoid unauthorised remote execution, the *PsExec.exe* program is installed in each calculation node and managed by the calculation node controller using local

administrator rights. The distributed process management controller allocates tasks to each calculation node depending on the available CPU cores per calculation node, and the calculation node controller allocates a single task to each CPU core using the *PsExec.exe* program. To test the CPU-level execution, a simple executable program called *CPUCoreText.exe* is used.

7.8.2 CPU Core Testing Application

This is a test program designed using VB6 to execute CPU-hogging calculations to monitor how the CPU performs under stressed conditions and to execute on the assigned CPU core by using the *PsExec.exe* program. Initial test results show that programs can be executed exclusively in a selected CPU core without affecting other cores. Hence, once the program execution is assigned to a particular CPU core, it will continuously executed on the assigned core only, does not affect the other CPU cores. If the distributed processing is activated during office hours when users are employing their workstations, the operating system will avoid CPU cores already occupied by the distributed processing programs and use the available CPU cores for user-level programs. However, if the user-level programs require more CPU power, then the operating system will activate the context switching to use all the CPU cores, and this will affect all the programs on that particular node; it will slow down the distributed processing programs as well as user programs. By utilising the load balancing technique that incorporates each CPU core usage with its algorithms, it is possible to allocate tasks depending on the CPU core load parameters. Hence, each CPU core acts as a single logical calculation node. Consequently, a single workstation that has four CPU cores can be set up as four separate logical calculation nodes if fully utilised. However, if a physical workstation is employed by an assigned user, then it depends on CPU core usage index and how many logical calculation nodes are available for to be used. The distributed processing controller in coordination with the local calculation node controller manages this process. Table 7.12 and Table 7.13 list CPU core utilisation by operating system and using the CPU core usage profile-based processor affinity assignments method.

Table 7.12: CPU core usage profile when operating system manages the program execution in a single CPU that has four cores

Program Instance	CPU Utilisation	Core 1 Used	Core 2 Used	Core 3 Used	Core 4 Used
1	25%	Yes	Yes	Yes	Yes
2	50%	Yes	Yes	Yes	Yes
3	75%	Yes	Yes	Yes	Yes
4	100%	Yes	Yes	Yes	Yes

Table 7.13: CPU core usage profile when *psexec.exe* executes the program in a single CPU that has four cores

Program Instance	CPU Utilisation	Core 1 Used	Core 2 Used	Core 3 Used	Core 4 Used
1	25%	Yes	No	No	No
2	50%	Yes	Yes	No	No
3	75%	Yes	Yes	Yes	No
4	100%	Yes	Yes	Yes	Yes

During office hours, the workstations and servers are continuously used, and the CPU usage index for each calculation node is employed to evaluate the available CPU cores per node. When the workstations and servers are not used during out-of-office hours, all the CPU cores are available for distributed processing. The number of available CPU cores for distributed processing is calculated using Equations (7.4) and (7.5):

$$c(i) = INT[m(i) \times (1 - p(i))] \tag{7.4}$$

$$p(i) = MAX\{p_C(i), p_A(i)\} \tag{7.5}$$

where

- $c(i)$ Number of available core(s) for distributed processing in node i
- $m(i)$ Number of CPU cores in calculation node i
- $p(i)$ Selected CPU usage percentage of calculation node i
- $p_A(i)$ Average CPU usage percentage of calculation node i
- $p_C(i)$ Current CPU usage percentage of calculation node i
- *MAX* Maximum value of the dataset
- *INT* Integer value round-down to floor

Table 7.14 lists the calculation node average CPU usage percentage and available CPU cores for distributed processing. The number of total available CPU cores within a selected calculation cluster is calculated using Equation (7.6).

Table 7.14: Available CPU cores per calculation node with varying CPU cores

Number of CPU Core	**Average usage of CPU**					
	20%	**30%**	**40%**	**50%**	**60%**	**70%**
2	1	1	1	1	0	0
4	3	2	2	2	1	1
8	6	5	4	4	3	2
12	9	8	7	6	4	3

$$n = \sum_{i=1}^{N} c(i) \tag{7.6}$$

where

n Total number of available CPU cores within a selected calculation cluster.

N Total number of calculation nodes within a selected calculation cluster.

A simple test is performed to test one of the risk calculation scenarios that described section 4.2.1 in Chapter 4, using a Zotac Pico computer to illustrate that the CPU core-level calculation can be performed on small form factor (SFF) computers. The result shows that using a single core, the calculation time is 83 seconds, and using four cores as distributed calculation, the time is reduced to 22 seconds. The behaviour of calculation time reduction for the execution of a particular program is similar to using discrete PCs and workstations. Hence, the CPU core-level calculation can be utilised with various hardware such as servers, workstations, PCs, and SFF computers to form a logical CPU core cluster, and the CPU core cluster can span across the entire spectrum of hardware. In addition, the CPU core-based logical cluster can be separated for various calculation requirements, such as 10 four-CPU-core workstations being used as a single 40-CPU-core node cluster or logically separated clusters such as four separate clusters with 10 CPU cores each.

7.8.3 CPU Core-Level Auxiliary Processing

The auxiliary processing method is investigated to reduce the failure rates of any given task per batch execution. Even though the failure rate of a calculation node's hardware is considerably low due to its high reliability and continuous maintenance. However, other factors also involved in distributed processing, such as the network and operating system that affect the performance. Hence, by implementing the auxiliary processing using the CPU core coupling method for each process, the task failure rate per batch process is reduced. The calculation process coupling is done by executing a single process in two or more logical CPU cores that are located in different physical calculation nodes, and how the task is allocated to each CPU core is determined by the distributed process management controller by a defined set of rules. For CPU core-level task processing, a single physical calculation cluster that consists of multiple CPU cores can be configured as multiple logical clusters with CPU cores as processing nodes. Figure 7.11 shows the four workstations with a physical CPU that has four logical cores in the logical cluster using CPU cores.

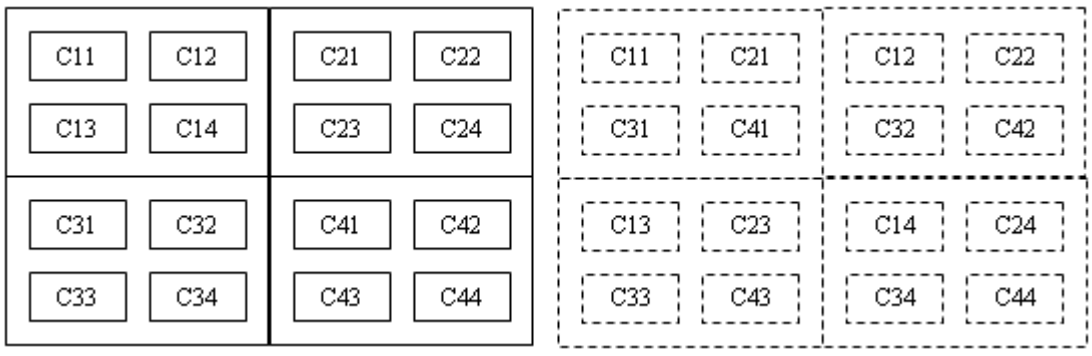


Figure 7.11: Physical and logical CPU core grouping: (a) Four workstations with 16 logical CPU cores, (b) Four logical CPU cores-based auxiliary processing groups

For example, if a single process or a single workstation failure is one in 1,000 per operation, then for four CPU cores, coupled task failure probability per operation is one trillion to one, and for a two-CPU core coupled task, failure probability is one million to one. To get a similar calculation time for each auxiliary task execution, the

CPU core speeds should be the same, and this is normally true if the chosen calculation nodes have the same hardware configurations, especially the CPU core speed. Figure 7.12 shows the use of physical processing devices as separate logical CPU core clusters.

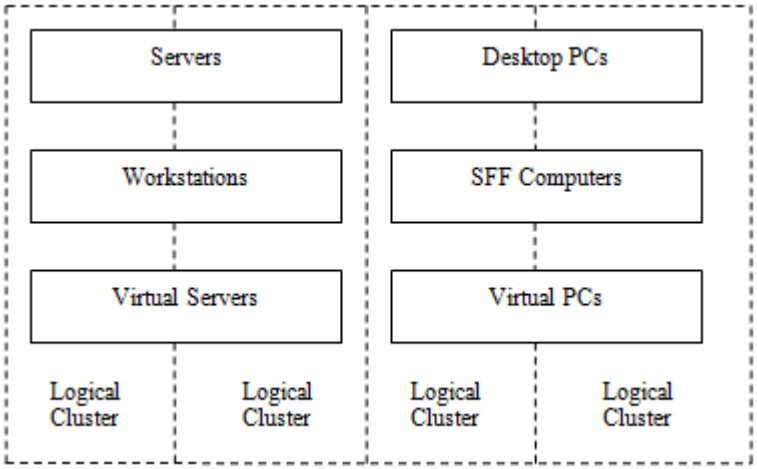


Figure 7.12: Using physical processing devices as separate logical CPU core clusters

7.9 Distributed Processing Implementation for Dispersion Trading

Dispersion trading is a type of portfolio management strategy that buys or sells index options premiums and sells or buys the underlying share's premiums based on structured portfolio-level delta hedging. The dispersion trading system that was used in the past by the company is based on the trading strategy that uses an index and its constituents' options for producing a considerable amount of profit against the benchmark index for a given period. It was designed in a way that the system operates like a program trading type of system. However, it was a partially automated system with various manual overrides and facilities for traders to make required adjustments to trading algorithms when needed. Due to the complex nature of the strategy used and the computing power needed to calculate all the required parameters for profitable trading, the system came to a halt after a few months of usage. The system proved to be highly complicated to use in the conventional serial calculation method using dedicated servers, especially during high-volatility market conditions. Hence, to solve the calculation requirement for this system, various tests are performed on the distributed-process-based calculation method to evaluate whether is it feasible to calculate all the required parameters for the dispersion trading strategy during highly volatile market conditions where the data changes occur rapidly. To test the distributed process-based calculations for the dispersion trading strategy, three types of calculation clusters are used as a combined hybrid cluster. In addition, both data decomposition-based and functional decomposition-based calculations are used to minimise the calculation time for the entire portfolio that used for testing. As shown in Table 7.15, three separate clusters are configured with 10 calculation nodes for each cluster: the Virtual server cluster, Workstation cluster, and NUC cluster.

Table 7.15: Calculation cluster configuration parameters

Parameter	Server Cluster	Workstation Cluster	NUC Cluster
Number of Nodes	10	10	10
Node Model	HP BL480	HP Z420	Intel NUC
Node CPU	Xeon 2.8 GHz	Xeon 2.8 GHz	Atom 1.4 GHz
Node CPU core	4	4	2
Node Memory	12 GB	12 GB	2 GB
Node Power	120W	270W	35W

To evaluate the calculation time improvements against different clusters, the following test scenarios are performed:

- All required calculations in a single calculation node of each cluster.
- Distributed calculations used for static load balancing with data decomposition.
- Distributed calculations used for static load balancing techniques with both data and functional decomposition.

Different types of calculations need to be performed continuously to facilitate the traders, fund managers to make trading, and portfolio management decisions. The calculations are nonlinear, and some of the calculations take longer than other calculations. Hence, the distribution is done by splitting calculation tasks by combination of data and functions across multiple clusters to maximise the calculation efficiency. The following processes are distributed across three calculation clusters:

- Implied Volatility Analysis
- Dispersion Analysis
- Correlation Analysis
- Tracking Error Calculation
- Profit and Loss (P&L) Calculation
- Risk Analysis
- Delta Hedge Calculation

To test the dispersion trade simulation using distributed processing techniques, the Hong Kong Index (HSI Index) and 50 of its constituent shares as a share basket that have higher 90-day realised volatility that is considerably higher than the HSI index 90-day volatility are chosen to create a dispersion trade test portfolio. The tests that carried out are mainly focused on how the automated trading process can be improved using distributed processing techniques. To test whether the dispersion trading strategy can produce enough positive cash flow for the portfolio, the program-based autonomous trading system has to be simulated on the following scenarios.

- Various market conditions are used.
- Different indexes are used.
- Varying share baskets are used.
- Different delta hedging algorithms are used.
- Different dispersion and correlation algorithms are used.

However, these types of simulations are mainly conducted by the quantitative analysts and researchers. Hence, these simulations can only be carried out when the system is fast enough to calculate the required parameters for dispersion trading. The test shows that by using a single HP Z420 workstation to calculate dispersion trade algorithms, it takes around 11 minutes to complete. Hence, in theory, trades can be placed at 11 minutes intervals, and this is not a suitable solution for a real-time trading environment. Ideally, a total calculation time of less than 15 seconds per calculation cycle is the minimum requirement for the dispersion trading strategy to be fully implemented for the live-trading environment. In addition, calculation reliability, repeatability, and accuracy are also important factors of the calculation process, and these can be achieved by distributing the calculation process across many calculation clusters with auxiliary calculation nodes. The calculation clusters can have combinations of multiple physical and logical nodes that span across multiple physical clusters, and this will increase the cluster reliability.

In a typical trading day, the first hour after the market opens and the last hour before the market close is the most volatile periods of the day, and the dispersion trading activities are high during these time-periods. Furthermore, during volatile market conditions, the whole day can be highly volatile and the dispersion trading activity can be high throughout the day. Hence, to eliminate trade execution delays during highly volatile markets, every calculation that updates the dispersion trade table keeps the next 10 consecutive trade execution data values as predictive trading data for downside move and upside move. In theory, every calculation cycle creates a single trade execution data set for each security; however, in practical implementation, this method can cause serious execution order delays due to unforeseen calculation delays. Therefore, introducing multiple datasets for trade execution is the safest option during

highly volatile market conditions, and the method used is shown in Equations (7.7) and (7.8):

$$Y(i, j, T) = f(X(i, j)) \tag{7.7}$$

At calculation period T, the output dataset is reset as shown in (7.8):

$$Y(i, j, T) = Y(i, j, T_P) \tag{7.8}$$

where

X	Input dataset
Y	Output dataset
T	Current calculation dataset's timestamp
T_P	Previous successfully completed dataset time stamp
m	Number of datasets *(j= 1, 2, …., m-1, m)*
n	Number of calculations *(i=1, 2, …, n-1, n)*

Thus, every calculation cycle, the next 10 (m=10) consecutive trade execution data values are available for the trade execution engine to process the trades. Currently, five for downside market move and five for upside market move are used; however, these can be fixed numbers or can vary based on adaptive methods depending on the algorithms used. This is a safety protection mechanism to avoid serious trading delays during highly volatile market conditions, and the trade execution engine is able to submit the trades to the electronic trading system in case of serious delays in calculations. Currently, the number is set to 10 for testing; however, this number can be large or small depending on the calculation time required to complete all the calculations needed for producing trade execution data sets. Smaller numbers produce better results in highly volatile market conditions. In low-volatility market conditions, they have less impact. In highly volatile markets, the number of trades required is relatively high compare with low-volatility market conditions; hence, the dispersion trading system must be capable of adapting to changing market conditions. In highly volatile market conditions, trade execution might be required every 15 seconds, and in low-volatility market conditions, there will possibly be a few trades per day, and in

certain days, there will be no trades. Hence, the system has to be capable of adapting to changing market conditions and able to be fast enough to calculate required datasets within set time limits. The dispersion trading strategy is more profitable during the highly volatile market conditions; therefore, the calculation speed is crucial for the implementation of dispersion trade strategies. For time-critical calculations using simultaneous calculation methods, the first returned output is selected as a result and the rest of the data is discarded. Hence, the fastest CPU core result will be used for the preceding calculations, and so on. The following example shows how four calculation nodes are used to calculate the same function with the same data sets simultaneously and the first returned output is used for the calculations. For selecting the appropriate output datasets, output dataset *Y* as a function of input dataset *X* is represented as in Equation (7.9) as a generic form:

$$Y = f(X) \tag{7.9}$$

Hence, output dataset Y for the calculation node i rewritten as Equation (7.10):

$$Y(i) = f(X, i) \tag{7.10}$$

The first completed output dataset and selected output dataset are shown in Equations (7.11) and (7.12), respectively:

$$Y(i, T(i)) = f(X, i, T(i)) \tag{7.11}$$

$$Y_S = Y(i, T_M(i)) \tag{7.12}$$

where

$T_M(i) = MIN\{T(1), T(2), T(3)....T(N-1), T(N)\}$

N Number of calculation nodes

T(i) Calculation time taken to complete the calculation in node *i*

Y_S Selected output dataset

MIN minimum value of the dataset

For data-critical calculations, multiple calculations have to be performed and the output data compared to each other to ensure that all the calculations' results are the same before moving to the next stage of the calculations. Equation (7.13) shows how the output dataset is selected where *p* is an arbitrary value that is set to 1 or 0 depending on the resultant datasets:

$$Y_S = p \times Y(i) \tag{7.13}$$

where

Y(i) Output dataset from calculation node i

Y_S Selected output dataset

p=0 when $Y(i) \neq Y(i+1)$; *p=1* when $Y(i)=Y(i+1)$

To eliminate the time delays due to calculation time discrepancies, the suitable solution is to use the same speed CPU cores with identical memory allocations. Hence, all four calculations will be completed at the same time or with minimal time delays, assuming that no failure during the calculation. Figure 7.13 and Figure 7.14 show the simultaneous calculations using CPU core as a calculation node and output dataset selection based on the rules using Equations (7.12) and (7.13), respectively.

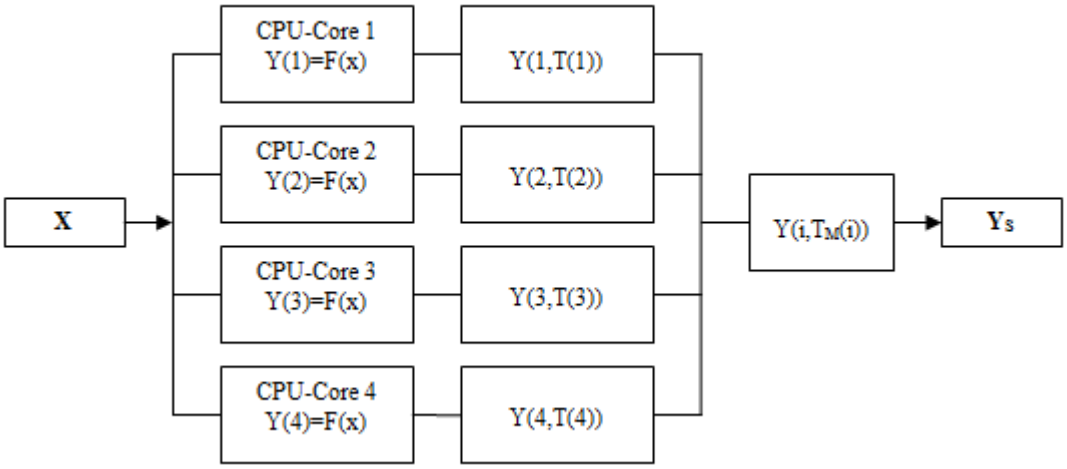


Figure 7.13: CPU core-level distributed calculation for first output dataset selected

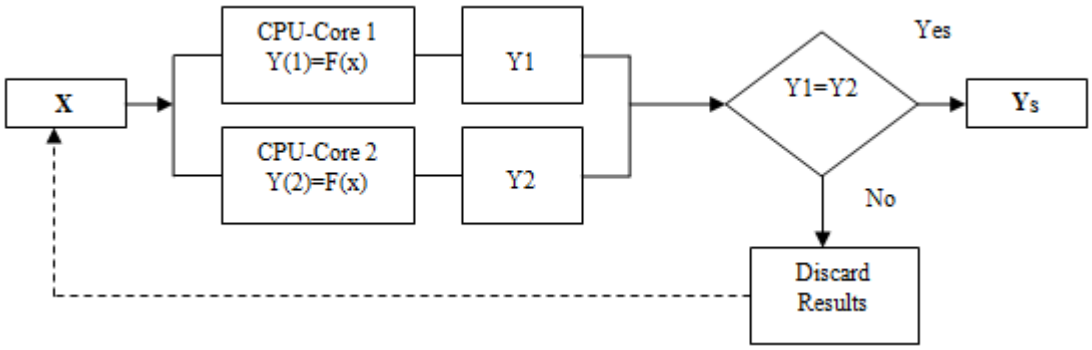


Figure 7.14: CPU core-level distributed calculation for output dataset selected when both outputs are equal

Using input datasets for performing the same calculations that uses eight CPU cores; it is possible to achieve the minimal requirement for both time-critical and data-critical scenarios. Using 16 CPU cores gives an optimal performance for both time-critical and data-critical requirements. However, to implement this level of data- and time-resilient calculations requires larger numbers of calculation nodes or multiple and separate calculation clusters working together as a single cluster. Hence, for example, if a process requires 10 calculation nodes in a conventional distributed processing cluster, then it requires 160 nodes for data- and time-critical calculations. Therefore, these types of calculations are better suited for CPU cores set up as each calculation node; for example, a workstation with a CPU that has 16 cores can be set up as 16 separate calculation nodes where each node is managed by the distributed processing controller. For complex trading operations, such as dispersion trading that consists of many hundreds of securities in the basket, more than 1,000 calculation nodes are required for calculating all the necessary trading parameters within the acceptable time scales. Even though the calculation node number is high, it is possible to build these types of calculation clusters using small form factor (SFF) computers with multiple CPU cores, and the initial cost of these types of computers and the running cost of these types of clusters are relatively small compared with workstations or server clusters. Table 7.16 lists the cluster type and allocated calculations types, and Figure 7.15 and Figure 7.16 show functional decomposition methods and both domain and functional decomposition methods, respectively.

Table 7.16: Process distribution for each cluster using functional decomposition

Cluster Name	Cluster Type	Calculation Type
CL_1	Server Cluster	Risk Analysis
CL_2	Workstation Cluster	P&L Calculation Implied Volatility Analysis
CL_3	SSF Computer Cluster (NUC Cluster)	Delta Hedge Calculation Tracking Error Calculation Dispersion Analysis Correlation Analysis

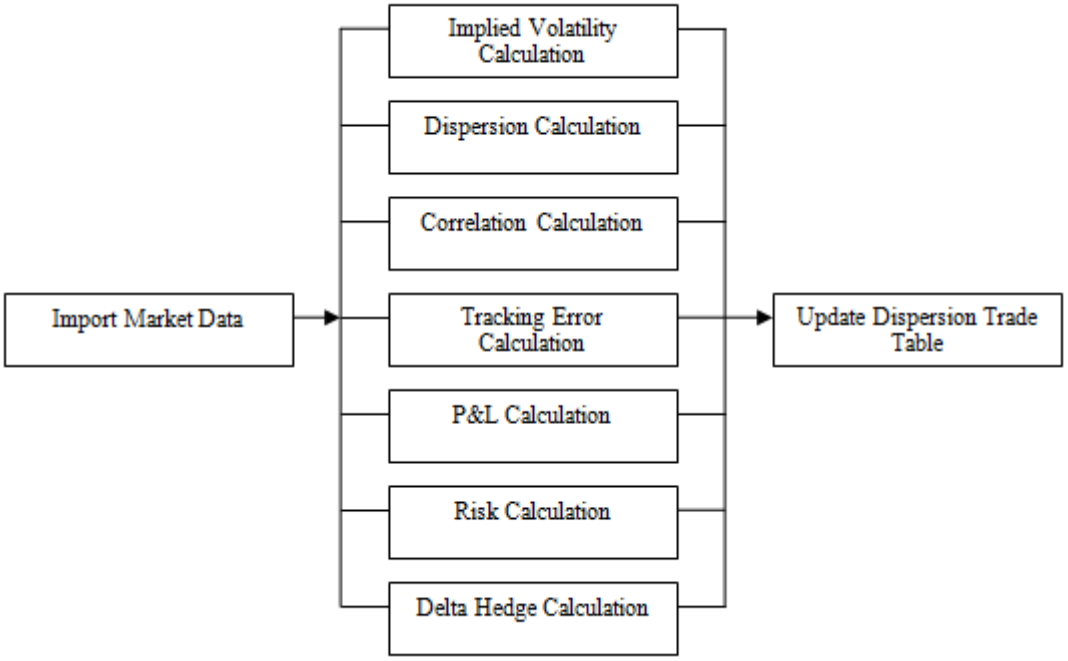


Figure 7.15: Distribution of dispersion trade calculations using functional decomposition

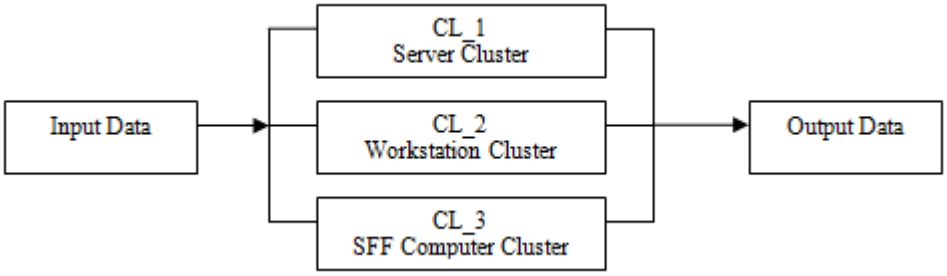


Figure 7.16: Implementation of distributed calculation cluster for domain and functional decomposition

Table 7.17 lists calculation requirements for options and shares, and Table 7.18 lists calculation time for each batch using a single HP Z420 workstation.

Table 7.17: Calculation requirement for options and shares

Calculation Type	Index Option	Share Option	Share
Implied Volatility Analysis	Yes	Yes	No
Dispersion Analysis	Yes	Yes	No
Correlation Analysis	Yes	Yes	No
Tracking Error Calculation	Yes	Yes	No
P&L Calculation	Yes	Yes	Yes
Risk Analysis	Yes	Yes	Yes
Delta Hedge Calculation	Yes	Yes	Yes

Table 7.18: Calculation time taken for each dataset in a single HP Z420 workstation

Calculation Type	Calculation Time (sec)
Delta Hedge Calculation	15
Tracking Error Calculation	17
Dispersion Analysis	19
Correlation Analysis	27
P&L Calculation	67
Implied Volatility Analysis	128
Risk Analysis	158

For calculation based on distributed processes, both domain and functional decomposition are used. The domain decomposition is used within each cluster; hence, the same calculation is used with different datasets for each cluster calculation node. The functional decomposition is used across different clusters; hence, each cluster is using same dataset but processing different sets of calculations. The domain and functional decomposition method is shown in Equation (7.14):

$$Y = \sum_{i=1}^{n} \sum_{j=1}^{m} f_i(X_j) \tag{7.14}$$

where

X_j Input data for dataset j

Y Output data

n Number of processes

m Number of datasets

f_i Calculation function for process i

To demonstrate to the company the possibility of using multiple clusters for complex calculations that can be performed within acceptable and specific time limits, the following parameters are used: One Index option, 50 Share options, and 50 Shares. The number of securities is selected to ensure that the total calculation time completes within 15 seconds and that every 15 seconds, the dispersion trade executions data table is updated. However, if the number of securities increases, then a need for more calculation nodes to ensure that the total calculation time remains below the acceptable period of 15 seconds. Hence, a dedicated cluster with a large number of calculation nodes is the most suitable solution for real-time trading scenarios, and this will be investigated further once the dispersion trading strategy is finalised by the company.

Table 7.19: Number of calculations and dataset required for each security type

Trade Sec Type	Required Data Set (X)	Total Data Set (Xn)
Index Option	1	10
Share Option	50	500
Share	50	500

Table 7.20: Distributed calculation times for each cluster type with load balanced condition

Calculation Node	Workstation Cluster Calculation Time (sec)	NUC Cluster Calculation Time (sec)	Server Cluster Calculation Time (sec)
1	10	14	12
2	11	13	14
3	12	13	10
4	11	12	12
5	14	12	13
6	12	13	10
7	13	13	10
8	14	14	13
9	14	14	12
10	12	13	13

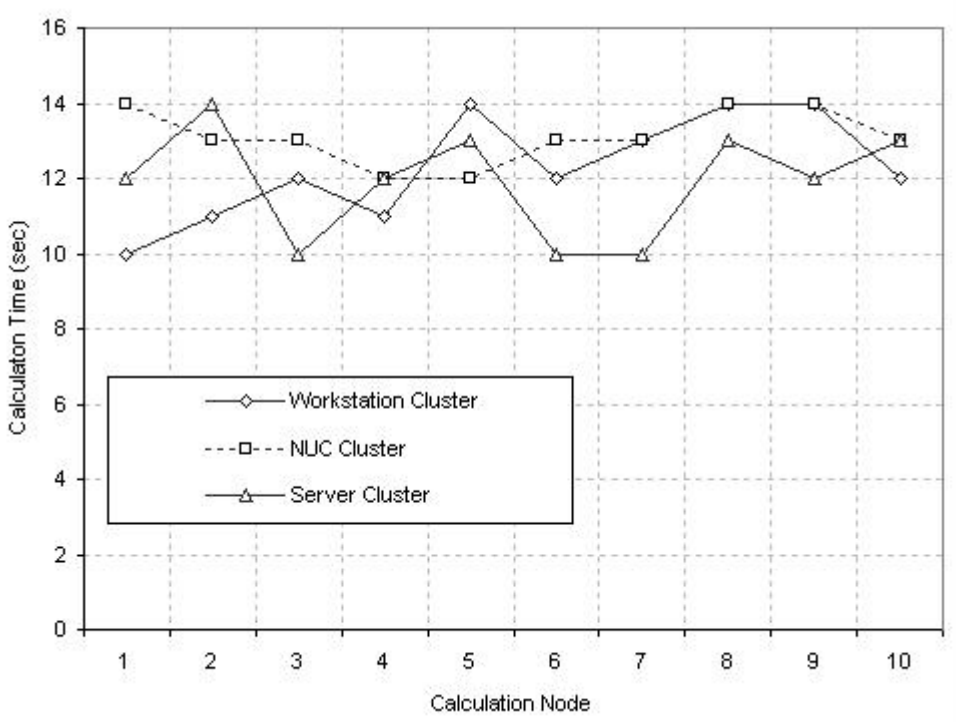


Figure 7.17: Distributed calculation times for each node for all three clusters under load balanced condition

Implementing three physical clusters with data and functional decomposition methods to distribute the calculations across each calculation node within each cluster using static load balancing techniques has reduced the overall calculation time to be within the 15-second time limit. Even though the test is performed in a prototype environment, it has proved that it is possible to implement suitable distributed process-based calculation solutions for time-critical and data-critical real-time applications. For the real-time trading environment, the better solution is to implement a fully dedicated cluster with standby auxiliary calculation nodes to cover the node failures, and in addition, using application-specific static and dynamic load balancing to further improve the calculation time. Due to successful prototype testing and promising results, currently, the company is considering to implements dedicated blade server based calculation clusters for real-time trading systems and various type of cluster configuration proposals are under investigation.

7.10 AH Trading System

The company has been trading AH shares for more than seven years using a simple AH trading system. In the past, the number of trades per day was relatively low, and the full calculation takes up to approximately a minute to complete. Because of the small number of trades per day, the calculation time was not a critical factor, and the fund managers were able to use the system in real-time trading. However, the calculation time became a major issue due to the following factors:

- The introduction of Stock Connect (SC) between Hong Kong and mainland China: the trade volumes have increased considerably in both the mainland China and Hong Kong markets.
- Due to the limited number of shares that can be traded per day using the SC, the volatility of the share prices becomes considerably high compared with pre-SC.
- The company has increased the number of positions within the AH trading strategy portfolio.
- The AH trading strategy portfolio now has both Long-Short and Short-Long AH pair trades in comparison with only Long-Short in the past.

Stock Connect (SC): A new platform introduced by mainland China for trading shares in the mainland market and in the Hong Kong market using an electronic trading system with limited quotas per day. Hence, the aforementioned factors have affected the calculation time, and now it takes more than ten minutes to process all required calculations to be completed. In addition, the number of trades per day has also risen considerably to keep up with the high level of market volatility, and this is causing calculation time constraints. Because the AH trading system was originally designed to be used within a single workstation and intended for single user-based operation, hence, its data and programming structure is not suitable for distributed processing with separated hardware nodes in its current state. However, the program codes, data structures, and calculations can be modified to use multi-threading techniques, but this requires major modifications to the existing systems. This type of modification may be implemented at later stages when the further developments have been approved by

the company. Therefore, one of the best possible ways of distributing the calculation is to implement distributed processing within a single workstation using CPU cores as calculation nodes. The workstation that is used for the AH trading system has a 12-core CPU, and these CPU cores are used as calculation nodes using VBScript, Windows Management Instrumentation (WMI), and *psExec.exe* tools. In addition to local calculations, a SQL stored procedure that provides data to the AH system on demand, and this is a single procedure that runs on a single SQL server. This is another area of improvement made to reduce the calculation time by using four linked servers to distribute the data processing load across all four servers at the same time. The AH system is based on MS-Excel and it has the following configurations: MS-Excel built-in functions, user-defined VBA functions, local XML data storage, and data from the SQL server.

MS-Excel has built-in functionality to utilise all the CPU cores in parallel to calculate the entire workbook. However, how MS-Excel distributes the calculations and the calculation order cannot be controlled by the user, except by writing a sophisticated operating system-level program to control the threads. Meanwhile, if MS-Excel built-in functions are used, then the calculations will be managed at the thread level by the operating system, but if user-defined functions or nested functions are used, and then the calculations become unpredictable. Therefore, the better solution is to execute the MS-Excel calculations in a serial mode and to distribute the user-defined function calculations using distributed processing with CPU cores as calculation nodes. The advantage of using bespoke-type distributed processing within the single workstations is that this is fully manageable, unlike threads that are mainly controlled by the operating system. The efficient way of improving local calculation speed is to use parallel threads. However, to implement safe threading, the entire application structure has to be modified, and this is a major development process involving C++, .Net, or similar programming languages. The challenge is to develop simple and easy-to-use distributed processing systems that use existing programs and tools provided by the operating system. One of the major advantages of this method is that the system can be used in simple serial calculation-based mode or distributed calculation-based mode without changing the core part of the system. Figure 7.18 shows a distributed AH

system implementation, and Table 7.21 lists the improvements made to the system using distributed calculations with CPU cores and distributed data processing using linked SQL servers. For testing the AH system, the followings are used: a calculated SQL data table that has 30,000 rows and 87 columns, and an Excel application that has 1500 rows and 155 columns. The SQL table processing is managed by SQL server RDMS and Excel user-defined functions calculated by VBScript programs.

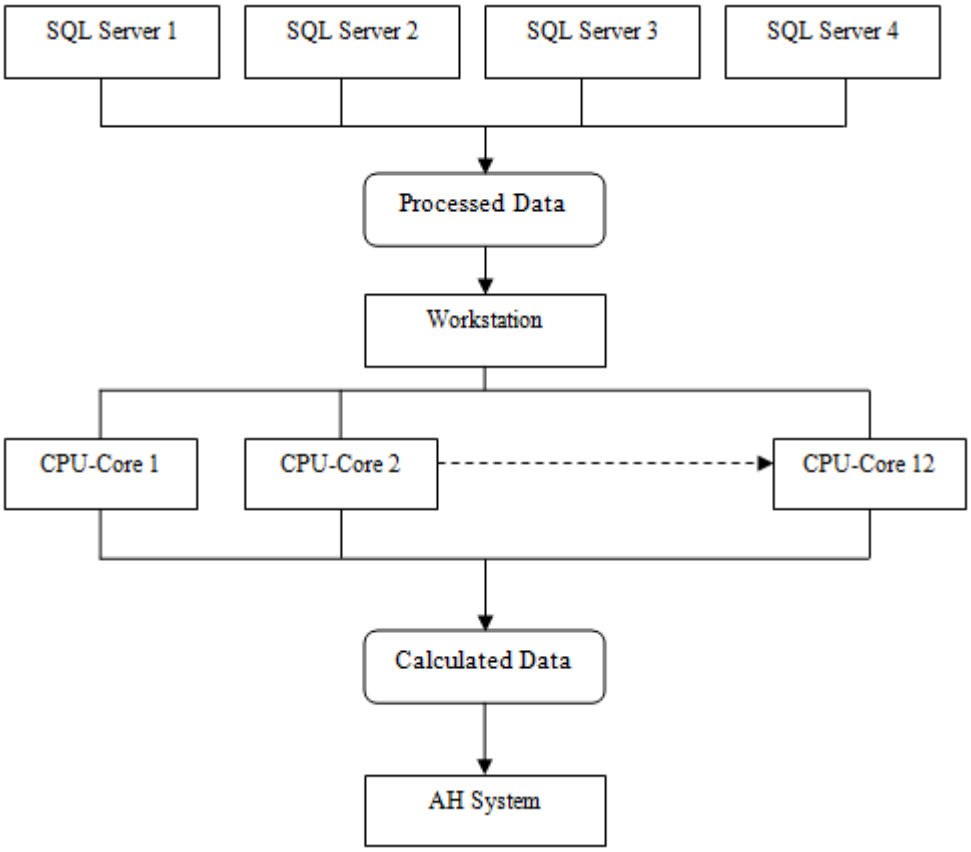


Figure 7.18: Distributed processing implementation for the AH trading system

Table 7.21: AH trading system serial and distributed calculation times

System	Improvement made	Serial Method Time (sec)	Distributed Method Time (sec)
SQL Stored procedure	Used four linked SQL servers for distributed data processing	12	4
Local VBA programs and MS-Excel functions	Used VBScripts, WMI, and PsExec.exe tools for distributed processing within a single workstation	124	15

7.11 Distributed Query Processing Using SQL Server

For distributed SQL query processing, four SQL servers linked as distributed SQL query-processing clusters for SQL server-based applications. However, these servers are not part of the distributed processing cluster, and these servers are linked at the SQL server level rather than at the Windows server level. Hence, the security authentication for executing distributed SQL queries is based on SQL server user-level security settings, and SQL programs are written using Microsoft Transact-SQL to perform the distributed SQL query processing across all four servers. The query execution and load balancing is managed by the SQL server engine; therefore, once the execution process has started, the SQL server engine is responsible for managing the CPU and memory management. In database query processing, two types of distributed processing: SQL server-managed distributed processing and user-managed distributed processing. In SQL server-managed distributed processing, how the data is processed is managed by the SQL server's relational database management system (RDBMS) engine. This process is managed by the SQL server engine in coordination with the server operating system for efficiently utilising the CPU, memory, and data storage access. The efficiency of the RDSMS engine also depends on how the SQL programs are written to handle the database data; hence, the SQL programs are structured in particular ways to facilitate the RDBMS engine to perform efficient operations on the database. The SQL server has various tools to monitor the RDBMS engine's performance and to facilitate the fine-tuning of the SQL programs.

Linked SQL servers are extremely useful for data-intensive processing and complex calculations, and the task distribution is simple to implement where the business data is highly structured. Because all the applications and databases are internally developed, and the data structures are well organised to suit the business, modifying the data structure to work with distributed SQL processing is relatively easy within the company's business model. However, this approach may not be suitable for other businesses that rely on off-the-shelf products that have no access to the internal workings of the systems. In general, systems cannot be modified; in most cases, these applications are fully compiled and no access to the code or data structures is possible.

Most of the companies do not allow for linking SQL servers due to security concerns; however, for Northwest, all the SQL servers are within the Northwest secure network and only accessed by authorised users. In addition, the network and security are internally managed. Hence, the security issues are not a serious concern for linked SQL servers that are located within the Northwest network. In server-side distributed processing, the linked SQL server RDBMS engines manage the calculation processes using multiple SQL stored procedures, and the client side only executes a single command on one of the linked servers; the server that received the client request will take control of how the process is executed across all linked servers. In client-side distributed processing, each linked server is connected separately using the asynchronous SQL connection method. Hence, each linked SQL server is treated as a separate calculation node, and the client sends commands to each server to execute a defined SQL stored procedure. This process is managed at the client level in coordination with the distributed processing SQL database. For testing, same data tables are used that described in section 7.7. Figure 7.18 shows the SQL server crosslink method, Table 7.22 lists the server-side and client-side process times, and Table 7.23 presents the distributed process time analysis.

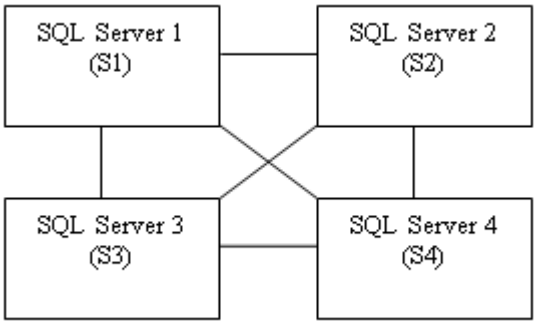


Figure 7.19: Linked SQL server interconnection between four SQL servers

Table 7.22: Server-side and client-side distributed process time for each node

Processing Node	Server-side Processing Time (min)	Client-side Processing Time (min)
S1	3.8	4.2
S2	4.4	4.6
S3	4.2	4.4
S4	4.1	4.5

Table 7.23: Time analysis of server-side and client-side distributed process calculations

	Server-Side Processing Time (min)	**Client-Side Processing Time (min)**
Maximum	4.4	4.6
Minimum	3.8	4.2
Average	4.13	4.43
STDEV	0.25	0.17

Because the test data used for processing requires a small set of data, the processing time difference between server-side and client-side processing is relatively small. However, certain historical data processing requires large sets of data to be shared between linked servers; hence, server-side processing is most suited for these types of distributed processing. Therefore, the implementation of either server-side or client-side processing depends on the application used for the processing, and some applications use small amounts of data, whereas other applications use large amounts of data such as batch processing applications. The tests are carried out to illustrate the possible implementation techniques within the company using existing SQL servers and applications as part of the distributed processing clusters. The SQL server is highly optimised for processing large amounts of relational data. Hence, utilising the SQL servers as a linked server cluster for data processing has the following advantages:

- Resources are managed by the SQL server in coordination with the operating system.
- The Relational Database Management System (RDBMS) engine takes control of processing data in the most efficient way possible.
- A number of highly optimised built-in functions are available for various calculations and data processing.
- SQL server security is closely linked with server operating system security.
- Many tools are available for incorporating Microsoft products and programming languages, such as MS-Excel, Access, VBA, .Net, or similar types of programming languages.

7.12 Portfolio Calculations Using Distributed Calculation Method

Currently, the company has more than 1,000 security positions within all managed fund accounts, and these positions are analysed using various calculations at each position level and at the portfolio level. The number of calculations needed depends on various parameters that are related to the security types. Some calculations are simple, and some of them highly complex; the calculation time varies from a fraction of a second to a few minutes per calculation, and certain portfolio-level calculations can require more than a few hours. These calculations are performed using multiple applications at various time intervals. Hence, currently, the portfolio-level calculations are grouped based on security type, data type, exchange feeds data, and broker feeds data. The general calculation formula is shown in Equation (7.15), and the sampling time *T* varies for different groups of datasets:

$$Y = f(X_1(T_1), X_2(T_2), X_2(T_3)....X_n(T_n)) \tag{7.15}$$

where

X	Input parameter
Y	Output parameter
T	Sampling frequency
f	Calculation function

Ideally, all the input parameters must be sampled at the same time *T* for accurate calculations; thus, Equation (7.15) can be reformulated as (7.16) where the sampling frequency for all the input data is the same:

$$Y = f(X_1(T), X_2(T), X_2(T)....X_n(T)) \tag{7.16}$$

By implementing distributed process calculations, it is possible capture all the input and output data at same time at a fixed sampling frequency. In addition, the data can be saved in the SQL server database for historical analysis and auditing purposes with calculation date and time stamps. The advantages of using distributed processing for portfolio-level calculations are as follows:

- Input and output data is centralised; hence, reduced data discrepancy between each user performing separate calculations.
- Number of position- and portfolio-level calculations is reduced considerably.
- Data sampling is centralised and uniform across all the portfolios and calculations.
- User-requested on-demand calculations' performance is improved.

For implementing a dedicated calculation cluster for portfolio-level calculation that should be highly effective, a single CPU core is dedicated to each security position that the company holds. Hence, each CPU core has to be configured as a calculation node within the dedicated calculation cluster to perform as a series of calculations for a given position ID. Therefore, to achieve maximum performance for the entire portfolio, the company requires a 1,000-node dedicated cluster. Even though, in theory, the number of calculation nodes can be less than the number of positions in the whole portfolio, the calculation time needed to calculate all the parameters will be higher, and in some cases, it takes longer to calculate the required parameters and is not suitable for real-time use. Meanwhile, due to the recent development of multi-core small form factor (SFF) computers and license-free Windows 10 operating system available for these types of computers, it is possible to build a 1,000-core calculation node dedicated cluster that costs approximately $12,500 USD. This improves the real-time calculation efficiency that can be implemented in the company, and the recommendation is made to the company. To test the proposed method of distributed portfolio calculation using distributed processing, the following are used:

- Ten NUC computers set up as a calculation cluster.
- Each computer is configured to act as a calculation node.
- XML files are used as input/output data storage.
- VBScript is used for calculations.
- A calculation node controller is installed on each calculation node.

For testing, 100 existing positions are selected with linear calculations to test the calculation times using distributed processing with an SFF computer-based calculation cluster. The primary purpose of testing portfolio-level calculations using distributed processing is to investigate and demonstrate that it is possible to implement dedicated and cost-effective calculation clusters to perform real-time calculations for various position- and portfolio-level calculations. The tests carried out are relatively simple and just use less complex portfolio positions with simpler data composition. Because all the NUC computers used in the cluster have similar hardware and software, the load is distributed evenly to each calculation node. In real-time full-scale calculation, there will be various challenges to be reckoned with, such as non-linear complex calculations, time-varying parameters, and large numbers of calculations per portfolio. In full-scale implementations, adaptive load balancing techniques must be used to minimise the calculation time using full portfolios for real-time calculations. Table 7.24 lists each node's calculation time, Table 7.25 lists distributed calculation time analysis, and Figure 7.20 shows the distributed calculation time for each calculation node.

Table 7.24: Distributed calculation time for each calculation node

Calculation Node	Calculation Time (sec)
NUC-1	32
NUC-2	32
NUC-3	33
NUC-4	32
NUC-5	34
NUC-6	32
NUC-7	34
NUC-8	33
NUC-9	33
NUC-10	32

Table 7.25: Distributed calculation time analysis

Maximum Time (sec)	Minimum Time (sec)	Average Time (sec)	STDEV
34	32	32.70	0.82

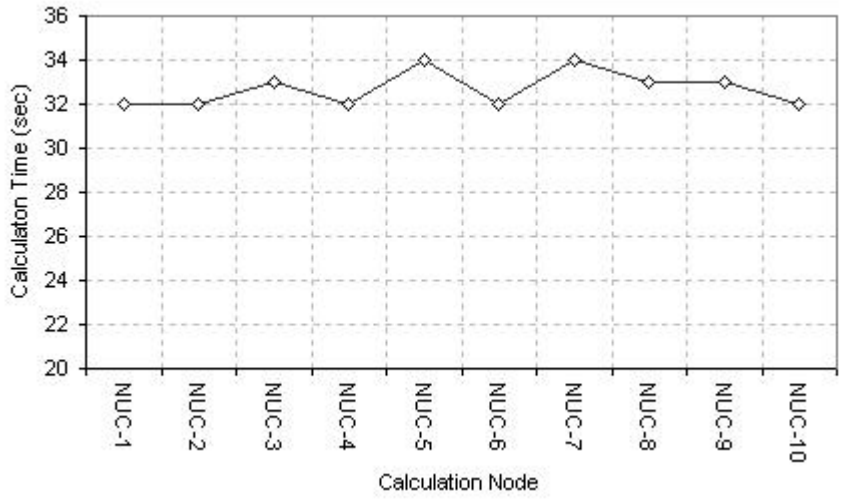


Figure 7.20: Distributed calculation time for each calculation node

For real-time calculations, dedicated calculation services using distributed processing cloud based computing are provided by various companies for financial institutions. The service they provide is based on the number of calculations used, how often the calculations are performed, and the fee structure based on usage parameters. The major advantage of use these types of utility computing is they are highly powerful and no need to maintain in-house hardware. All the calculations are performed on the distributed cloud that is scalable depending on the demand. Hence, the calculation time is considerably shorter than that of a fixed distributed processing cluster. The downside of this approach, particularly for Northwest, is the cost and need to change the applications to those that are compatible with cloud programming languages. Due to these restrictions, the utility-type computing option is not suitable for the current situation. Therefore, implementing an internally developed dedicated distributed processing cluster that is cost-effective with low power consumption using small form factor (SFF) computers with multiple cores is an alternative approach to improve the calculation efficiency for portfolio-level calculations.

7.13 Distributed Processing in a Single Workstation

A number of applications are used by the company that employ MS-Excel for calculating various complex financial algorithms and derivative pricing, and these applications are developed as bespoke development within the company. Most of these applications are used by quantitative analysts and researchers, and are continuously modified to suit the business requirements. These applications have the following characteristics:

- Applications are self-contained.
- Primarily used by a single user on a particular workstation.
- Input and output data are part of the application and are managed by the users.
- In certain cases, uses FDF to get data in addition to manual data entry.
- Data analysis, charts, and graphs are part of the application.
- User-defined functions are written in VBA.

Hence, these types of applications need different types of distributed processing methods that require minimum modification to the applications to reduce the overall calculation time. One of the approaches is to implement a distributed processing method within a single workstation using multiple instances of MS-Excel applications. This method is only suitable for multiple-core CPU-based computers, and all the high-end workstations that are used by analysts and researchers have 12 processing cores in a single CPU. Hence, these types of workstations are highly suited for distributed calculation within a single computer using a single CPU core as a calculation node. However, instantiating multiple MS-Excel applications will consume considerable memory space, but this is not a serious issue because the high-end workstations have 32 GB memory installed. The distributed processing method that is tested uses technologies that are simple to implement with minimal modification to the existing applications that are currently being used. Hence, the method tested has the potential to be applied to new applications that will eventually be developed by analysts and researchers in the future as the business grows and the business requirements change.

MS-Excel has a certain type of built-in distributed calculation functionality using multithreading methods; however, these functionalities only apply to MS-Excel's own built-in functions and for user-defined functions (UDF), these methods do not apply. To implement multithreading within a single MS-Excel application incorporating user-defined functions, threaded programming is involved and has to be designed as executable and used as an add-in to each application. Even though employing an executable solution is the most efficient way of implementing parallel and distributed processing, this method is time-consuming and requires considerable development using high-level languages such as C++. Due to continuous changes made by the users to the MS-Excel applications that are currently used, this method is not suitable in its current state; instead, a simpler and adaptable distributed processing implementation is most suited for these types of applications. The method tested is mostly suited for certain types of MS-Excel applications used within the company and may not be suitable for other types of applications. Table 7.26 lists the steps needed to implement the distributed calculation process within a single workstation.

Table 7.26: Distributed processing steps for using a single workstation

Step	Process Description
Step 1	Find number of CPU cores in the workstation using WMI
Step 2	Evaluate total number of input and output data using the main MS-Excel application
Step 3	Create input XML data files in local drive
Step 4	1. Create MS-Excel instances according to the CPU cores 2. Open a distributed calculation application in each instance 3. Allocate appropriate number of input data to each application instance 4. Calculate each instance of the application 5. Save output data to XML file 6. Close all the MS-Excel instances
Step 5	Read the output data XML and load data to main MS-Excel application dictionary

The distributed MS-Excel-based calculation method is suitable for implementing within a single computer with multiple CPU cores, and in addition, it can be modified to suit the different application types depending on whether domain decomposition or functional decomposition is possible within the application. In certain application types, the distributable part of the calculations can be used for distributed processing and the rest the calculations uses a serial calculation method. Therefore, the

combination of distributed and serial calculations is the best suited for complex MS-Excel applications. Equation (7.17) shows the total calculation time taken in a single MS-Excel application instance. Equations (7.18) and (7.19) show how the load is distributed across all MS-Excel application instances. Because all the calculations are the same for each dataset and CPU core speed is the same, the load is distributed equally as shown in Equation (7.20). Equation (7.21) shows the total distributed calculation time for a given calculation task.

$$T(i) = \sum_{j=1}^{m(i)} t(i, j) \tag{7.17}$$

$$M = \sum_{i=1}^{N} m(i) \tag{7.18}$$

$$m(i) = f(i, N, L(i)) \tag{7.19}$$

$$m(i) = \frac{M}{N} \tag{7.20}$$

$$T = MAX\ (T(1),\ T(2) \ldots T(N\text{-}1),\ T(N)) + T_D \tag{7.21}$$

where

T	Total distributed calculation time for a given task
M	Total number of calculations
N	Number of CPU cores
$m(i)$	Number of calculations allocated to MS-Excel instance *i*
$T(i)$	Time taken to complete the calculations in MS-Excel instance *i*
T_D	Read, write, and application load time delays
$t(i,j)$	Time taken to calculate a single dataset (j) in MS-Excel instance *i*
$L(i)$	Load balancing factor for MS-Excel instance *i*
MAX	Maximum value of the dataset

Figure 7.21 and Figure 7.22 show how multiple instances of the MS-Excel application are implemented using local XML data as input and output for the application. The followings programming methods are used for implementing distributed processing within a single workstation:

- Windows 7 WMI process
- Command shell process
- VBScript programming
- VBA programming
- MS-Excel (Using the multiple instantiation method)

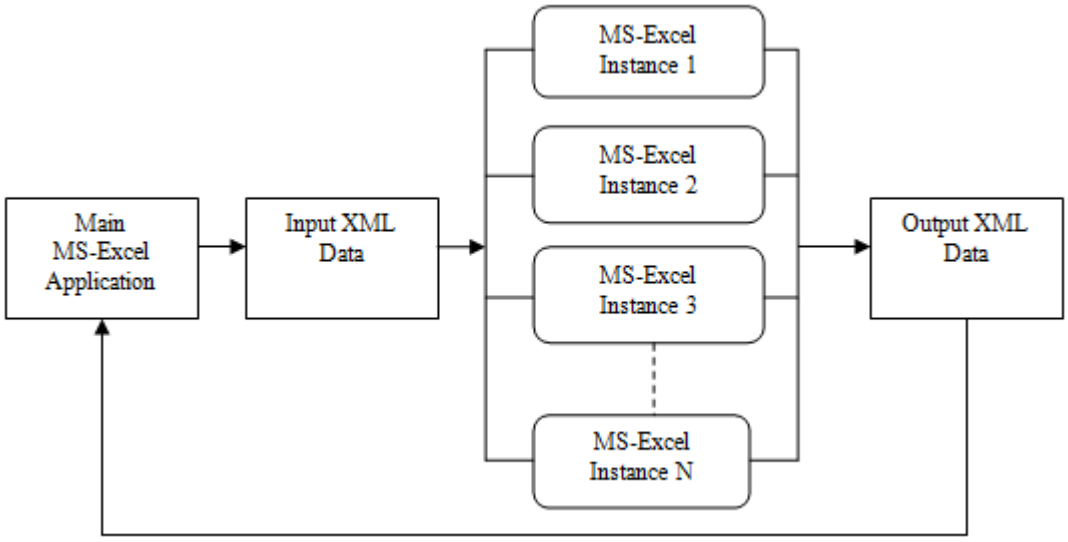


Figure 7.21: MS-Excel application multiple instantiation using XML data as input/output

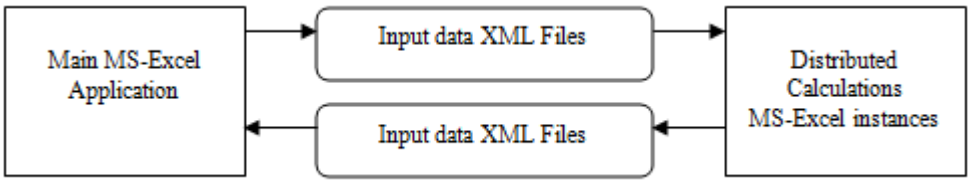


Figure 7.22: MS-Excel application multiple instantiation using consolidated XML data as input/output

7.13.1 Testing Configuration

The application used for testing the single workstation-based distributed processing techniques is called call trigger shudder fixing using propriety design based method. The application goes through multiple recursive calculations to find the best solution using binomial and trinomial tree methods. This process is highly compute-intensive and takes between a few minutes to a few hours depending of the number of CB positions used. One of the major problems for the CB price profile calculation that has call provision and these types of CBs can cause undesirable price discontinuity called 'call shuddering'. This effect is caused by the inherent nature of the binominal and trinomial tree models used to calculate the CB price and the shuddering effect can be reduced by increasing the calculation steps. However, increasing the calculation step will increase the calculation time due to the exponential nature of the binominal and trinomial tree structures. Meanwhile, another method that developed by the company uses multiple superimposed trees to calculate the CB price during the call trigger periods. This method also takes considerable time to calculate compared with the single binary tree method; however, it requires less time than using the increased step size method. For testing and comparing the calculation time improvements using distributed calculation within a single computer, two workstations are used with different hardware configurations: one with a 4-core CPU and another with a 12-core CPU. For testing, an Excel application is used that has 120 CBs and the CB model calculation is explained in section 4.2.1 in Chapter 4.

The workstation parameters are shown in Table 7.27, Table 7.28, and Table 7.29 lists distributed calculation time analysis for 4-core and 12-core workstations. Figure 7.23 and Figure 7.24 show application instances against calculation time for a 4-core workstation, while Figure 7.25 and Figure 7.26 show application instances against calculation time for a 12-core workstation. The test result has shown that if number of MS-Excel instances exceeds the number of CPU cores in the workstations, the calculation time reduction is inconsistence due to operating system related context switching. Hence, to get the optimal calculation time improvement for a given batch of calculation, the MS-Excel instances must be equal to the number of CPU cores.

Table 7.27: Test workstation's parameters

Workstation Model	CPU	CPU Core	Memory	OS
HP Z420	1 x Xeon w3520 2.67 GHz	4	12 GB	Windows 7 (64)
HP Z820	1 x Xeon E5-2620 2.00 GHz	12	32 GB	Windows 7 (64)

The main MS-Excel application that is employed by the users as a user interface-based application is modified to execute multiple instances of MS-Excel calculation applications to serve as a calculation service for the main application. This is done by using the following steps:

- Collect hardware configuration and CPU core details using WMI objects.
- Create VBScript files dynamically based on number of CPU cores.
- Create input/output XML files dynamically.
- Instantiating calculation application instances depending on number of cores.
- Allocating data to each application for distributed calculation.
- Use system timer to monitor calculation completions.
- Consolidate the distributed processing of output data.

Table 7.28: Distributed calculation time analysis for 4-core CPU workstation

Number of MS-Excel Instances	Number of Calculations	Maximum Calculation Time (sec)	Minimum Calculation Time (sec)	Average Calculation Time (sec)	STDEV
2	60	81	80	80.50	0.70
4	30	42	38	40.00	1.82
8	15	36	20	26.75	7.16

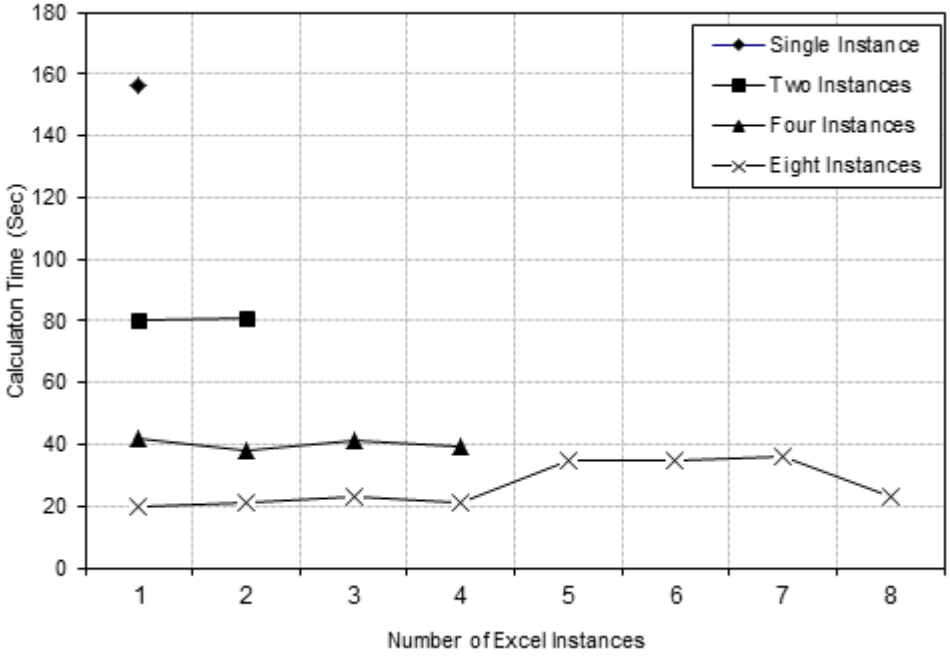


Figure 7.23: Number of MS-Excel application instances against total calculation time for 4-core CPU

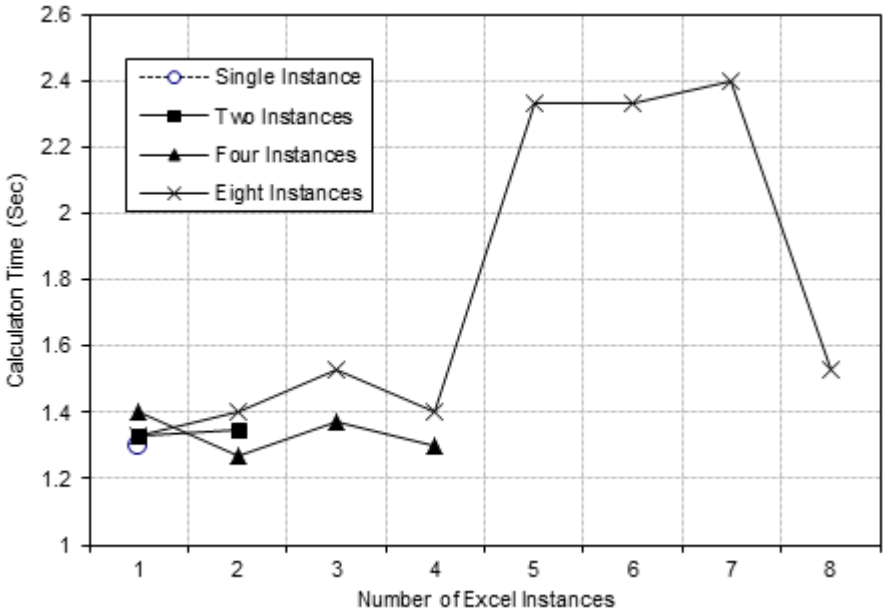


Figure 7.24: Number of MS-Excel application instances against the calculation time of each dataset for 4-core CPU

Table 7.29: Distributed calculation time analysis for 12-core CPU workstation

Number of MS-Excel Instances	Number of Calculations	Maximum Calculation Time (sec)	Minimum Calculation Time (sec)	Average Calculation Time (sec)	STDEV
2	60	90	90	90.00	0.00
4	30	47	46	46.50	0.58
8	15	24	23	23.25	0.47
12	10	16	15	15.75	0.45
24	5	34	8	15.71	7.94

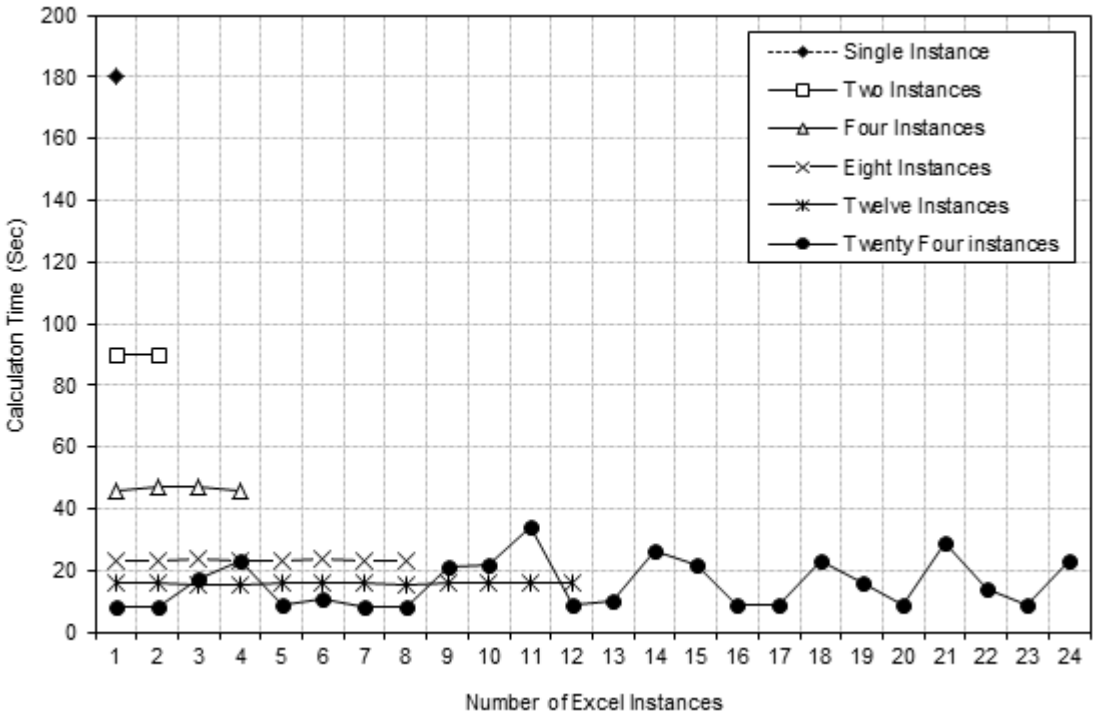


Figure 7.25: Number of MS-Excel application instances against total calculation time for 12-core CPU

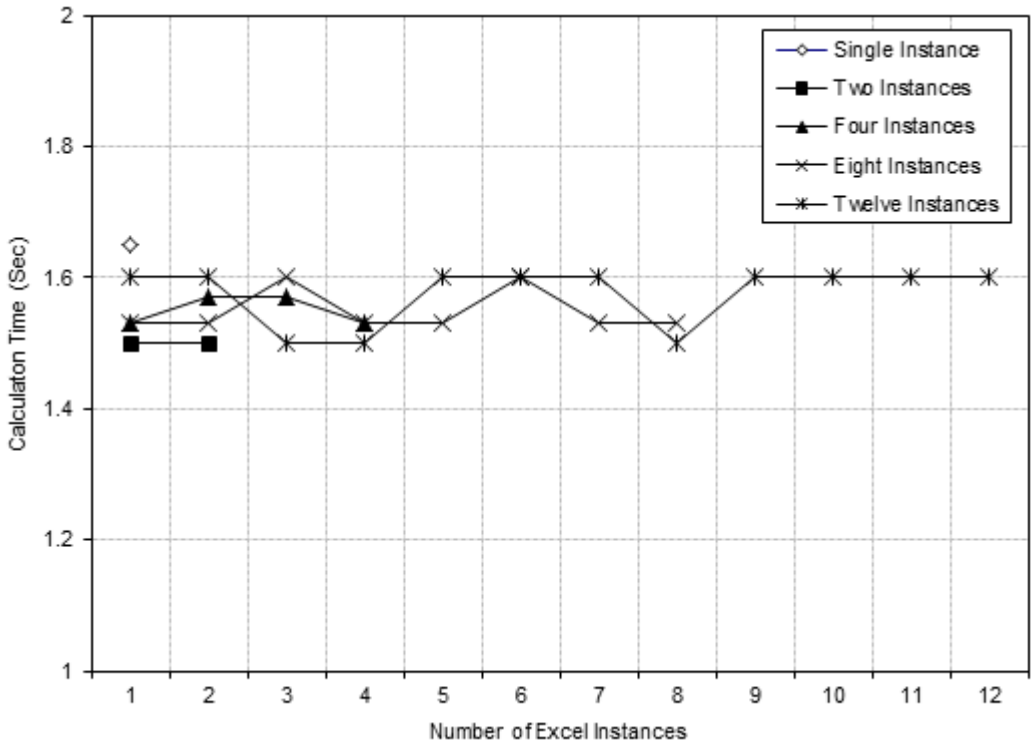


Figure 7.26: Number of MS-Excel application instances against the calculation time of each dataset for 12-core CPU.

The suitable option for implementing distributed processing in a single computer is to use multithreading methods by using compiled programs. However, to implement this method requires careful planning and extensive programming using compiled programs and necessitates complete redesign of the existing applications. This process is time-consuming and laborious; therefore, this approach is not suitable for the following reasons:

- The applications used are prototypes for testing various scenarios and may not be used in the long term or modified after a few months of usage.
- The applications are mainly used by researchers and quantitative analysts who continuously add, remove, and modify user-defined functions.
- All the user-defined functions are based on MS-Excel VBA.
- The applications use cross-referenced built-in VBA program libraries.

Meanwhile, the tested approach has the potential to be used in the company for certain type of MS-Excel based applications and further investigations are in progress.

7.14 Chapter Summary

The results from the application testing using different types of cluster configurations for compute-intensive tasks have shown that the implementation of different types of clusters for distributed process-based calculation can improve the overall real-time calculation times for different types of applications used in the company. Furthermore, it is possible to implement consolidated hybrid-type calculation clusters for real-time trading applications and multivariable applications. These types of multi-level calculation cluster configurations facilitate the analysts, traders, and fund managers to test the trading algorithms in a real-time environment as well as to perform various financial model simulations and research. In the traditional way of testing, a single server or a workstation is used that usually takes a few days to complete the full tests. If any errors are detected or modifications are needed, it will take even longer to complete the tests. However, by implementing distributed process-based calculation using various combinations of calculation cluster configurations, the time taken to complete the full testing process is reduced to within an hour or less. Further calculation time reduction is achieved by implementing improvements on static and dynamic load balancing and, in addition, using separate dedicated calculation clusters. The advantage of using multiple cluster configurations is that each cluster can act as a single fully isolated cluster or part of the consolidated cluster and this process is fully managed by the distributed processing controller. In addition, the clusters can be physically or logically separated depending on the applications used and the time of the day they are used, such as during office hours or out-of-office hours. However, the clusters that are built using surplus unused servers for testing the load balancing algorithms are not suitable as production systems due to various limitations such power consumption, space requirements, and maintenance costs. Meanwhile, these clusters are useful for testing load balancing algorithms due to their nonlinear nature of software and hardware configurations. Various lessons have been learned about how to build different types of clusters for distributed calculations using different types of processing devices. Due to the recent development of power-efficient, cost-effective small form factor (SFF) computers with highly reliable single-board designs, these are better suited for dedicated calculation cluster design rather than using conventional workstations or PCs.

As far as the company's application is concerned, certain types of applications are not suitable for distributed process-based calculations, and some of them are partially or fully suited for distributed process-based calculations. Hence, to get the maximum benefit of calculation efficiency for a given system requires the implementation of a combination of serial, parallel, and distributed processing techniques depending on the systems used. Most of the MS-Excel-based applications that are developed with SQL server databases and VBA user-defined functions are relatively easy to modify to work with the distributed processing systems; in addition, certain user-defined functions can be converted to VBScript or XLA types of programs that can be used in CPU core-level logical calculation clusters. However, certain types of MS-Excel applications that are mostly designed by quantitative analysts and risk analysts require considerable modifications. Because these types of applications originally designed to be used as standalone applications for a particular user, no structure or order to the application concerned. Hence, application usability within the different types of calculation cluster configurations is also a major factor for consideration regarding cluster operations.

For a certain types of compute-intensive MS-Excel applications that are designed to be used in a single computer that tested for distributed calculation tests using both 4- and 12-core workstations, the test results show that by implementing memory-ridden MS-Excel calculation application instances using the MS-Excel object instantiation technique, it is possible to reduce the overall calculation time. However, the optimum number of MS-Excel instances must be equal to the number of CPU cores, and if the MS-Excel instances are more than the number of CPU cores available in the workstation, then the performance will be reduced due to context switching applied by the operations system. The main advantage of this technique is that the distributed calculation is self-contained within a single workstation; in addition, it is simple to implement compared with network-based distributed processing. Hence, to get the maximum benefit of the distributed calculation using the multiple MS-Excel application instantiation technique, the number of MS-Excel application instances must be equal to the number of CPU cores available in the workstation concerned. For testing purposes, all the CPU cores are used; however, when the workstation is employed by users, it is safe to use certain number of CPU cores for distributed

calculation rather than using all the available cores in order to avoid the workstation freezing due to possible 100% CPU activities at peak. Therefore, in most cases, such as in a 12-core workstation, it better to utilise 10 cores for distributed calculation and leave the other two cores for the operating system. By doing so, the users can carry on using the workstation as normal with minimum interruption while the distributed calculation is running in the background.

The tested distributed processing methods are only suitable for certain types of applications and are not suitable for general types of distributed processing methods. However, these methods are highly suitable for most of the company's applications and provide solutions for reducing calculation time for compute-intensive applications. Furthermore, various tests performed using multiple clusters and different applications under different conditions have proved that it is feasible to use hybrid-type clusters within the company by recruiting all of the available processing devices as consolidated cluster configurations. However, a number of advantages and disadvantages of using certain types of cluster configurations, and these are discussed in detail in Chapter 8 where the results and discussions are presented. The hybrid method is one of the distributed calculation techniques that the company can utilise along with other types of distributed processing methods for improving number of applications and maximise the utilisation of processing devices that are available in the company. Hence, this chapter has demonstrated the original contribution to the design methods and implementation techniques of hybrid calculation clusters using multiple processing devices that are available in the company to improve the calculation efficiency of compute-intensive applications.

8 Research Evaluation

8.1 Introduction

This chapter discusses the experimental results and analyses the data captured during extensive tests and simulations carried out on distributed processing clusters and the application of load balancing techniques within Northwest's applications. The focal point of the discussion is how the bespoke-oriented design has proved to be a successful distributed processing approach that facilitates the company to solve highly complex compute-intensive applications that are currently used in the company. Furthermore, it will analyse and discuss the use of multiple clusters as hybrid-type clusters and their advantages in use for possible real-time trading environments. In addition, this chapter compare the different types of clusters and their advantages and disadvantages. Different types of comparisons are analysed using captured data for the dedicated calculation grid clusters under various scenarios and conditions as well as for the application of load balancing techniques. In addition, discuss the advantages and disadvantages of different types of hardware and software configurations to build the calculation clusters and how these differ, from general types of clusters to the company-specific implementations.

Further investigation of data collected during the testing of phase 1 that described in Chapter 4 and the improvement in using distributed processing method using MS-Excel application is discussed in section 8.4. Distributed processing cluster using a network of workstations for MS-Excel applications show considerable improvements in calculation time for various MS-Excel applications compared to serial calculation method. Furthermore, it possible to design, develop, and implement a cost-effective and high-performance distributed processing cluster system that uses Windows network topologies and existing networked workstations to perform time-critical and data-critical calculations. The investigation has demonstrated that it is feasible to implement batch process-based distributed processing without any major changes to the existing systems or with minimal change.

Load balancing system that implemented in phase 2 of the investigation the described in Chapter 5 is further analysed and discussed in detail in section 8.5. Bespoke load balancing system that applied has proved that it is feasible to implement a certain type of bespoke type load balancing algorithms for the distributed processing system. Further investigation on captured data shows that static load balancing technique with varying hardware- and software-specific parameters are highly suitable for critical systems. Moreover, the dynamic load balancing system proved to be highly useful for protecting against calculation node failures during the operation. Hence, for the Northwest systems, the static load balancing is the primary load balancing system due to its robustness and reliability; and the dynamic load balancing is used as a secondary load balancing mechanism to safeguard against calculation node failure during the execution phase. Furthermore, the test results show that certain applications are performed better under certain condition such as when application-specific parameters are incorporated within load balancing algorithms.

Detail analysis on captured data during various tests conducted on dedicated calculations clusters that described in Chapter 6 has proved that for batch processing type applications, SFF cluster is highly suitable. However, the PC cluster has certain advantages, such as testing load-balancing algorithms due its non-leaner nature of hardware and software configurations. A number of different tests are carried out on both clusters with and without load balancing conditions, and calculation time reduction using dedicated calculation clusters shows considerable improvements for compute-intensive calculations when used with distributed process-based calculation using both dedicated calculation grids. The results are analysed using different categories such as cluster overall performance, size, cost, power consumption, and reliability. The test results have proved that the dedicated clusters have improved the calculation efficiency of complex mathematical calculations consistently and both PC and NUC clusters have their own advantages that are useful for future research and development. Detail analysis of SFF cluster and PC cluster comparisons are presented in section 8.6.

The hybrid cluster design described in Chapter 7 that based on consolidating various processing devices into an intelligent calculation cluster that can be used for various calculation intensive operations. Further analysis on test results show that it is a highly suitable solution to maximise the processing power within the network. Furthermore, it is a high throughput cluster design and utilises an adaptive and self-tuning control mechanism that continuously fine-tunes the cluster's performance to permit it to function autonomously and designed in such a way that it can be modified whenever required with minimum impact to the business. The test results show that using multiple clusters with varying configurations coordinated as a hybrid distributed processing cluster has many advantages for complex and compute-intensive applications and it is possible to implement these types of clusters for real-time trading applications.

In section 8.8, a number of recommendations are made related to how to make improvements and enhancements to these clusters and the improvements required for the applications that are used in the distributed processing system based on the company's business point of view. Finally, some critical points are discussed regarding the long-term benefits of utilising the bespoke-type distributed processing system using cost-effective hardware components with loosely coupled design that facilitate the company to move forward with appropriate enhancement to the system in the future. Hence, this approach will help the company to be consistent with developments in hardware and software that continuously improve the processing efficiency of the systems used.

8.2 Distributed Processing Concept Analysis

In general, distributed processing is a method of distributing tasks to multiple processing units to reduce the overall processing time. This is the simplest form of task distribution. However, for practical applications, a number of factors determine how efficient the distributed processing is compared to serial processing methods. In distributed process-based calculations, reducing the overall calculation time for a given task is the critical factor. Thus, the general form of calculating the distributed process-based calculation time is shown in Equations (8.1) and (8.2):

$$T(i) = \frac{T_S}{N} + \Delta T(i) \tag{8.1}$$

$$T_D = MAX\{T(1), T(2), ...T(N)\} \tag{8.2}$$

where

$T(i)$	Time taken to complete the calculation task in node i
T_S	Serial process-based calculation time
T_D	Distributed process-based calculation time
N	Number of calculation nodes
$\Delta T(i)$	Distributed process-based calculation time delay in node i
MAX	Maximum value of the dataset

In an ideal situation where all the calculation nodes are equal, ΔT becomes 0; however, due to load imbalance, network delays, and hardware and software inconsistencies, ΔT always has some nonzero value. In its simplest form, to improve the distributed processing efficiency, that is reducing the overall distributed processing time; two parameters need to be improved: Increase the calculation nodes N and reduce the calculation time delay ΔT. The number of calculation nodes N can be increased by adding more calculation nodes to the calculation cluster, and the calculation time delay can be improved by implementing efficient load balancing algorithms and linear hardware configurations for each calculation node in the cluster. Increasing the calculation nodes will not achieve further improvements in calculation

time after a certain value due to management process overhead. However, these limits are only applicable to large-scale distributed systems. The system used in the company is comparatively small, and these limitations are not applicable. However, the applications tested have minimum calculation time related limitations that discussed in Chapter 4. As far as calculation time delay is concerned, the best possible way to improve is to implement a cluster with similar hardware and software configurations, and in effect, this reduces the load balancing complexity. Therefore, further developments regarding distributed processing within the company will be highly focused on these factors. Meanwhile, if the company decided to expand and incorporate the latest technologies as part of its IT infrastructure, then it will be possible to expand the cluster configurations further to incorporate fast communications technologies. These can be, MPI, PVM, Infiniband, and 100GbE, and in addition, fibre-optic cluster interconnects can be considered and dedicated multiple rack-based clusters can be used as private calculation clouds.

The initial phase that described in Chapter 4 was carried out to investigate the distributed processing implementation possibilities within the company's network using existing workstations has proved that not only is it feasible to set up a distributed process-based calculation cluster using available workstations, but also that this considerably improves the calculation speed for compute-intensive applications. The investigation's test results and collected data show that the cluster performance is highly acceptable for use as a batch processing calculation cluster for various MS-Excel-based applications. Even though the calculation cluster was only tested with risk scenario and gamma calculations with appropriate modifications, the concept is proved that employing the users' workstations, as a calculation cluster for the company is feasible. Furthermore, the tested applications use one of the company's most compute-intensive derivate financial models; hence, other similar types of compute-intensive MS-Excel-based applications can modified to be utilised within the distributed processing cluster. The tests conducted using the workstation cluster have proved that distributed process-based calculation considerably reduced the calculation times for certain compute-intensive applications; the wall clock calculation time improvements are shown in Table 8.1.

Table 8.1: Distributed calculation time improvements using workstation cluster

	Number of Calculation Nodes	Risk Scenario Calculation (Nonlinear)	Risk Scenario Calculation (Linear)	Gamma Calculation
Serial	1	289 seconds	133 seconds	67 minutes
Distributed	16	65 seconds	10 seconds	5.38 minutes
Improvement X		4.45 times	13.30 times	12.58 times

Further investigation carried out on the workstation cluster to test the bespoke-type load balancing and task allocation algorithms has shown calculation time improvements and a certain way of protecting against calculation node failures during the cluster operation phase. In addition, it has proved that it is possible to include company-specific parameters as part of the load balancing algorithm alongside general hardware- and software-related parameters as discussed in Chapter 5. By using application-related parameters as a part of the static load balancing algorithm, the distributed calculation time is reduced to 27.2 minutes from 38.4 minutes, and the standard deviation from 9.2 to 1.3. Furthermore, applying the hybrid method reduced the calculation time to 2.6 minutes and standard deviation to 0.13. The fixed step size method is the fastest; however, this method is an approximate calculation method, and for long-maturity date derivative products, this calculation method is less accurate than other variable step size methods. Meanwhile, the hybrid method is the most suitable method for long-maturity date derivate products due to its calculation accuracy and time taken to calculate. The most accurate calculation method is variable step size calculation, and the step size is related to each derivative product's maturity date; however, this method is also the most time-consuming method as far as calculation is concerned.

The dedicated distributed processing clusters using separate P2P network, PCs, and small form factor (SFF) computers that explained in Chapter 6 show the benefits of having a dedicated calculation cluster for certain types of applications that mostly require 24x7 operations and, in addition, time-critical and data-critical applications. Furthermore, detailed tests conducted by investigating different parameters between NUC clusters and PC clusters have proved that certain types of calculation cluster configurations are highly suited for particular applications. The main finding of these

investigations shows that the most efficient way of building dedicated calculation clusters is to use cost-effective compute nodes with uniform hardware and software configurations. Even though the clusters that were constructed using SFF computers are less powerful than those built from PCs, the overall benefit they offer outweighs the PC cluster in many aspects.

The investigation carried out on the hybrid type of clusters using multilevel cluster setups employing workstations, PCs, virtual servers, and CPU cores as calculation nodes has the potential to be utilised in the company for heavy-duty applications and batch processing tasks that require as much power as possible from all the processing devices. The hybrid clusters are mostly useful for time-critical and data-critical applications that require failproof and robust operations. These clusters can be used for real-time trading systems; however, from the test data collected and various analyses assessed on cluster performance, it is highly advisable to utilise well-designed and fully dedicated calculation clusters for critical applications rather than using hybrid types of calculation clusters for real-time trading systems. These approaches are still under investigation to evaluate the cost-benefit analysis and the regulatory and compliance issues that are associated with live-trading systems. While these investigations are underway to decide, what types of hardware should be used as calculation nodes, various tests are carried out to investigate whether the proposed distributed processing system is capable of handling extreme market conditions during its operations. The simulation and tests are conducted using historical data and captured market conditions; however, the real market conditions in the future cannot be predicted. Hence, the behaviours of these systems have to be controlled under such a condition based on different types of logical rules and using circuit breaker type control mechanisms to shut down the system completely in case of emergency situations.

Due to the nature of the company's business, the distributed system has to comply with existing infrastructure and systems. Hence, the system has to work with the existing systems without interrupting the day-to-day business and has to prove that it can considerably improve the calculation time of various systems that are used in the company. To satisfy the company's requirements, a solution must be implemented that uses existing technologies in alternative and innovative ways to design and develop systems that will facilitate the business within the current development model rather than using new technologies or third-party systems. The solution has to be best suited for the company's current requirements, and in addition, must be able to adapt to the business changes in the future. The bespoke-type development approach can be used fully or partly with similar or other types of business environments. The implemented distributed processing system has the following characteristics:

- All the required components for the distributed processing were developed using existing technologies that are currently used in the company.
- The calculation clusters were built using existing hardware and software in the company.
- Dedicated calculation clusters are built using cost effective SFF computers.
- A transparent development method was used, and complied codes are only used for system controllers; the business logic parts of the programs are fully transparent.
- Distributed processing applications have manual overrides for monitoring and troubleshooting or switching back to manual operations in case of emergency.

The company has a dedicated disaster recovery (DR) site situated in a different location than the main building and connected through a high-speed network. In case the company decided to expand the distributed processing capabilities in the future, the DR site can be utilised for constructing cluster farms for distributed processing with minimum cost. Currently, the DR site's hardware is used for prototype clusters testing and eventually, these clusters become part of the consolidated hybrid cluster for various application developments and testing.

8.3 Cluster Configuration Overview

Tests were performed on various types of clusters, and the data collected show that different types of cluster configurations are possible to implement within the company's requirements and within hardware and software constraints. However, the different distributed processing clusters that were designed and tested have shown that some types of clusters are more suitable compared to others depending on their usage. In addition, the applications used, data criticality, time criticality, and usability also influence how these clusters are utilised within the company. The implemented distributed processing systems consist of multiple types of cluster configurations as shown below, and each of them has its own advantages and disadvantages.

- Workstation cluster based on employing the users' workstations.
- Server cluster based on using virtual servers.
- P2P network-based dedicated cluster using small form factor computers.
- P2P network-based dedicated cluster using PCs.
- P2P network-based dedicated cluster using conventional servers.
- Hybrid clusters using all the processing devices within the company.
- Logical clusters using CPU cores as nodes across multiple processing devices.

8.3.1 User Workstation and Virtual Server Cluster

Using existing workstations and servers as calculation nodes for distributed processing clusters has many advantages. However, a few disadvantages due to the fact, these workstations and servers are part of the company's network infrastructure and are utilised for various functions. Furthermore, these workstations and servers are continuously used by company staff and system-level operations. The advantages of distributed processing using existing servers and workstations include:

- No specific hardware and no specific software required.
- Existing bespoke software can easily be adopted with little or no modifications.
- Using of already existing servers and workstations as calculation nodes.

- Different types of cluster configurations flexibility.
- Loosely coupled cluster design that can be modified or improved with minimum effort.
- The parallel virtual machines can run on the workstations and servers to act as calculation nodes.
- No extra hardware and software cost.
- Combinations of dedicated and non-dedicated workstations and servers can be used.

The disadvantages of distributed processing using existing servers and workstations include:

- Full capacity usage is limited due to continuous use by users and systems.
- Certain workstations and servers cannot be used as calculation nodes due to their usage for system-level critical applications.
- Need to implement complex load balancing algorithms due to the nonlinear nature of hardware, software, and usage parameters.

8.3.2 PC and Conventional Server Cluster

The clusters that are built using spare, unused PCs and conventional servers as calculation nodes for dedicated distributed processing clusters have a few advantages. These clusters are only used as prototype for testing different types of load balancing algorithms and distributed processing controllers. These clusters are not suitable for continuous operations due to many disadvantages as shown below.

The advantages of using spare PCs and servers for dedicated calculation clusters include:

- No cost for hardware and software that is used.
- Existing bespoke software can easily be adopted with little or no modifications.
- Different types of cluster design configuration flexibility.
- Loosely coupled cluster design that can be modified or improved with minimum effort.

The disadvantages of using spare PCs and servers for dedicated calculation cluster are:

- Considerably high power consumption.
- Require dedicated temperature-controlled environment.
- Require large space to house the cluster.
- Operating noise levels are considerably high.
- Cluster overall weights are considerably high.
- Reliability is low due to the employed hardware and high level of failure rates.
- Need to implement complex load balancing algorithms.

8.3.3 Small Form Factor (SFF) Computer Cluster

Using SFF computers for building dedicated calculation clusters has many advantages compared with dedicated calculation clusters built from spare unused PCs and servers; however, a few disadvantages as shown below.

The advantages of using SFF computers for dedicated calculation clusters include:

- Fully integrated design provides high reliability.
- Low cost, smaller size, and no audible noise.
- Considerably low power consumption.
- Considerably low heat dissipation.
- Cluster weight is considerably low.

The disadvantages of using SFF computers for dedicated calculation clusters are:

- Processing power is low compared with workstations or servers.
- Network interconnect is slower than workstations or servers.

8.3.4 CPU Core-Based Cluster

The CPU core-based cluster that utilises a single CPU core as a processing node is another approach tested that has many advantages to be used as a calculation cluster for certain types of applications; however, a few disadvantages as shown below.

The advantages of using CPU core clusters are:

- Single hardware device such as workstation or server that has many-core CPUs that can be used as multiple calculation nodes.
- Multiple logical clusters can be configured using various processing units, such as servers, workstations, and SFF computers.
- Clusters can be set up as mutually exclusive to each other.
- Cluster nodes' configuration flexibilities are based on requirements.
- Possible to set up robust and resilient calculation clusters using multiple auxiliary nodes.
- Different type of hardware can be utilised partially.

The disadvantages of using CPU core clusters are:

- Processing power is low compared with workstations or server-based clusters.
- Need to implement complex load balancing algorithms.
- Require certain modifications to the applications to work with the logical calculation cluster.

8.3.5 Cluster Security

Cluster security is governed by the company network's access security policies; hence, no need for separate security setups for the cluster security management. The network is protected by multiple firewalls and external access through VPN connections with three-level password protections. Only the workstations that are used as calculation nodes can be accessed by users under the user access policy, and the dedicated calculation clusters have only administrator-level access rights. In addition, a process-level security policy to protect the cluster management and process control to isolate the user interface from cluster operations within each calculation node.

8.4 Using MS-Excel Applications for Distributed Processing

Most of the compute-intensive applications used in the company are based on MS-Excel, and these applications were originally designed many years ago and continuously modified according to the business requirements. Hence, these applications have certain types of system and data structures that are not optimised and not suitable for structured operations such as distributing data or processing at the current state. However, by implementing certain types of add-on programs, these applications can be modified with minimum changes to be used as part of the distributed processing systems that are managed by the distributed processing controller and used as the user interface mode; that is, the users can employ the application in the usual way they are accustomed to. Therefore, no need for the complete redesign of these applications, and in addition, the company can continuously develop prototype applications using MS-Excel for testing various trading strategies and portfolio analyses. These applications can utilise the distributed processing functionality available in the company for speeding up the calculation process. In the financial industry, specifically in small hedge fund management companies like Northwest, the use of MS-Excel for developing various applications is a common practice. Hence, MS-Excel usage in the business application development will continue within the company. Therefore, a few structural changes are in place to ensure that the applications are capable of adapting to the distributed processing

system with minimal changes. The advantages of using MS-Excel for application developments include:

- Many built-in functions are available for various business-related calculations.
- Able to implement UDFs using VBA, or add-ins such as DLL, ActiveX EXE, or similar types of COM-compliant add-ins.
- Able to instantiate mutually exclusive multiple instances of using a single license.
- Object hierarchy is highly compatible with all the Microsoft products.
- Data partitioning is easy to perform due to grid-based data structures.
- Rapid Application Development (RAD) tools can be used to develop business-specific applications on shorter time scales.

8.5 Load Balancing Implementation Overview

The Northwest infrastructure has high efficiency with minimum failure rates due to high levels of maintenance by dedicated network administrative staff. Hence, the individual workstation failure or network failure rate is low, and the hardware and software used are highly reliable for day-to-day financial management of business operations. Due to its high reliability, a considerably low probability of calculation node failure at any point in time, and the auxiliary calculation node configuration is capable of handling two calculation node failures simultaneously. In most cases, whether a calculation is executed during office hours or out-of-office hours, the crucial factor that dictates the load balancing is the static load balancing algorithms, and most of the time, the dynamic load balancing is on standby mode, in case of any unexpected events happening during the execution phase. Hence, due to the highly specific and bespoke nature of applications used within the calculation clusters and the high reliability of the network infrastructure, and in addition, the tightly coupled control systems used within the calculation cluster, adaptive static load balancing is more suitable compared with dynamic load balancing, especially when critical financial calculations are involved. However, for particular systems, dynamic load balancing is more suited than static load balancing, and both static and dynamic load balancing techniques have advantages and disadvantages depending on how they are being

implemented and what type of system is used. This investigation has also proved that the system reliability and calculation node uniformity are the crucial components when predicting the expected calculation time for each batch of calculations prior to the distributed processing calculation starts.

For fine-grained processes, the dynamic load balancing is well suited, and the process of transferring from one processor to another is efficient and fast due to the process time and communication time ratio being small. Various methods are available for these types of implementations. However, for coarse-grained processes like Northwest applications that are more complex than just a simple calculation, any disruption during the distributed processes' execution can introduce serious time delays in the overall calculation time. Hence, the dynamic load balancing within the Northwest calculation cluster acts as a secondary load balancing mechanism to protect the distributed processing system from unexpected events that might cause the distributed processing system to experience a deadlocked state, which can cause serious delays.

Further investigations are being carried out for the multiple auxiliary process approach to reduce the process failure rates in order to maximise the calculation efficiency for a given application. This approach is implemented using a multi-core CPU with segregated processing within a single workstation; that is, the same calculation process be executed in parallel, in two separate CPU cores in different workstations simultaneously. Hence, both processes failing simultaneously are low compared to a single process failure within a single workstation, and this will greatly improve the calculation efficiency of the overall distributed processing cluster. Various load balancing techniques and algorithms are under investigation using real-time risk calculation data under different conditions. This may introduce possible challenges that have to be addressed, and solutions to be found and work-around must be put in place to ensure that the system is stable and robust. A few implementation-related challenges that need to be fixed that are related to certain complex derivative-product models. Three known problems need to be addressed: the first one is that for some convertible bonds such as Chinese CBs, the number of calculations per CB depends on

various factors beyond from the maturity date. This will cause some challenges to incorporate the number of calculations per CB to allocate the number of CBs per calculation node based on the load-balanced condition. Hence, a need for multivariate types of task allocation systems that can cater to these types of complex derivative models.

The second problem is that to calculate the implied volatility of a derivative instrument product, the derivative instrument price has to be calculated number of times using varying volatility. This is the most time-consuming process within the distributed processing systems that use risk analysis applications and can cause serious time delays during the operation. To get the load balancing mechanism to adopt a strategy to compensate for the long delays in calculations, some type of approximation techniques have to be used to predict the estimated calculation time. This can be performed by collecting historical data to predict the estimated calculation time. However, small changes in the derivate prices or underlying prices can cause the number of calculations needed to calculate the implied volatility to be increased significantly. Hence, to predict the estimated calculation time, a number of different data sets have to be maintained as lookup tables with fuzzy logic-based rules to estimate the calculation time during the operation.

The third problem is that the financial feed-enabled workstations have poor performance as calculation nodes while users are logged into the workstations. However, if the workstations are not employed by users, then the workstation performance is normal as expected. Even though memory and CPU usage are low while using financial feed software, the calculation performance is relatively poor. This is caused by background processes using various resources to maintain the live-data feed from financial data feed applications. This can cause bottleneck scenarios during other processes trying uses the same resources. Hence, for financial feed-enabled workstations, it is possible to introduce another load balancing parameter that is related to each workstation and that can be included in the task allocation formula discussed in Chapter 5.

8.6 Dedicated P2P-Based PC and NUC Calculation Cluster Analysis

The data collected during the tests conducted on the PC cluster and the NUC cluster that were described in Chapter 6 shows that the implementation of a P2P network dedicated distributed process-based calculation cluster has considerably improved the calculation time for different types of compute-intensive applications. Further analysis performed on the collected data shows that a consistent improvement for all the scenario calculations. The PC cluster is more powerful than the NUC cluster; however, the PC cluster has calculation nodes that have nonlinear hardware and software configurations and require load balancing using hardware and software parameters. Meanwhile, the NUC cluster has uniform calculation nodes; hence, the calculation tasks are split equally. The analysis is based on calculation time data collected for the calculation step size of 500 with the following categorisations:

- Cluster type: PC cluster and NUC cluster
- Program type: MS-Excel VBA, VBS, and EXE
- Financial model type: Binomial and Trinomial
- Calculation type: CB value, Implied volatility (IV), IV impact, and IR impact

Table 8.2 to Table 8.5 list distributed calculation time analysis for CB theoretical value calculations. Table 8.6 to Table 8.9 list distributed calculation time analysis for implied volatility (IV) calculations. Table 8.10 to Table 8.13 list distributed calculation time analysis for implied volatility (IV) sensitivity calculations. Table 8.14 to Table 8.17 list distributed calculation time analysis for interest rate (IR) sensitivity calculations.

8.6.1 CB Theoretical Value Calculation Analysis

Table 8.2: CB value calculation time for each PC cluster node

	Calculation Time (sec)					
	Binomial			Trinomial		
	VBA	VBS	EXE	VBA	VBS	EXE
Maximum	30	43	12	66	73	14
Minimum	15	22	4	28	35	4
Average	23.36	33.91	8.18	48.73	55.64	9.09
STDEV	5.54	7.87	2.93	13.76	13.35	3.36

Table 8.3: Distributed CB value calculation time for PC cluster with imbalanced load condition

Imbalanced Condition	Calculation Time (sec)					
	Binomial			Trinomial		
	VBA	VBS	EXE	VBA	VBS	EXE
Maximum	2.6	3.8	1.1	5.8	6.4	1.2
Minimum	1.4	2.1	0.4	2.6	3.3	0.4
Average	2.13	3.07	0.75	4.41	5.03	0.84
STDEV	0.45	0.65	0.25	1.14	1.09	0.28

Table 8.4: Distributed CB value calculation time for PC cluster with balanced load condition

Balanced Condition	Calculation Time (sec)					
	Binomial			Trinomial		
	VBA	VBS	EXE	VBA	VBS	EXE
Maximum	2.1	3.0	0.7	4.2	5.0	0.7
Minimum	1.9	2.8	0.6	3.8	4.5	0.7
Average	2.01	2.94	0.63	4.08	4.79	0.70
STDEV	0.07	0.08	0.05	0.11	0.16	0.00

Table 8.5: Distributed CB value calculation time for NUC cluster with balanced load condition

Balanced Condition	Calculation Time (sec)					
	Binomial			Trinomial		
	VBA	VBS	EXE	VBA	VBS	EXE
Maximum	5.3	7.7	2.0	10.0	13.9	2.3
Minimum	5.3	7.6	2.0	9.9	13.8	2.3
Average	5.33	7.62	2.01	9.95	13.88	2.31
STDEV	0.01	0.02	0.01	0.04	0.04	0.01

8.6.2 Implied Volatility (IV) Calculation Analysis

Table 8.6: IV calculation time analysis for each PC cluster node

	Calculation Time (sec)					
	Binomial			Trinomial		
	VBA	VBS	EXE	VBA	VBS	EXE
Maximum	62	122	19	122	154	39
Minimum	28	60	8	60	73	14
Average	50.55	93.82	15.20	93.82	122.00	28.73
STDEV	13.76	22.24	4.24	22.24	31.00	7.77

Table 8.7: Distributed IV calculation time analysis for PC cluster with imbalanced load condition

Imbalanced Condition	Calculation Time (sec)					
	Binomial			Trinomial		
	VBA	VBS	EXE	VBA	VBS	EXE
Maximum	10.7	5.7	1.7	10.7	13.5	3.4
Minimum	5.6	2.6	0.8	5.6	6.8	1.3
Average	8.47	4.56	1.38	8.47	11	2.6
STDEV	1.79	1.18	0.37	1.79	2.62	0.65

Table 8.8: Distributed IV calculation time for PC cluster with balanced load condition

Balanced Condition	Calculation Time (sec)					
	Binomial			Trinomial		
	VBA	VBS	EXE	VBA	VBS	EXE
Maximum	8.4	4.4	1.3	8.4	11	2.5
Minimum	7.8	4.0	1.2	7.8	10	2.3
Average	8.05	4.22	1.24	8.05	10.30	2.39
STDEV	0.21	0.11	0.05	0.21	0.23	0.05

Table 8.9: Distributed IV calculation time for NUC cluster with balanced condition

Balanced Condition	Calculation Time (sec)					
	Binomial			Trinomial		
	VBA	VBS	EXE	VBA	VBS	EXE
Maximum	13.5	23.1	3.4	21.8	48.4	5.8
Minimum	13.4	22.9	3.4	21.6	48.0	5.7
Average	13.48	23.00	3.42	21.73	48.18	5.74
STDEV	0.04	0.07	0.01	0.07	0.14	0.01

8.6.3 Implied Volatility (IV) Sensitivity Calculation Analysis

Table 8.10: IV sensitivity calculation times analysis for each PC cluster node

	Calculation Time (sec)					
	Binomial			Trinomial		
	VBA	VBS	EXE	VBA	VBS	EXE
Maximum	73	131	29	172	219	42
Minimum	36	68	12	85	109	13
Average	57.30	102.00	22.60	134.00	167.00	29.10
STDEV	13.60	22.40	6.55	31.80	39.20	8.55

Table 8.11: Distributed IV sensitivity calculation times analysis for PC cluster with imbalanced load condition

Imbalanced condition	Calculation Time (sec)					
	Binomial			Trinomial		
	VBA	VBS	EXE	VBA	VBS	EXE
Maximum	6.4	11.5	1.2	15.1	19.2	2.5
Minimum	3.4	6.4	0.4	8.0	10.2	1.1
Average	5.18	9.19	0.84	12.20	15.10	2.05
STDEV	1.11	1.81	0.28	2.57	3.18	0.56

Table 8.12: Distributed IV sensitivity calculation times analysis for PC cluster with balanced load condition

Balanced Condition	Calculation Time (sec)					
	Binomial			Trinomial		
	VBA	VBS	EXE	VBA	VBS	EXE
Maximum	5.1	9.4	0.7	11.9	14.7	2.0
Minimum	4.7	8.5	0.7	11.1	13.6	1.8
Average	4.90	8.85	0.70	11.50	14.30	1.86
STDEV	0.17	0.25	0.02	0.29	0.38	0.07

Table 8.13: Distributed IV sensitivity calculation times analysis for NUC cluster with balanced load condition

Balanced Condition	Calculation Time (sec)					
	Binomial			Trinomial		
	VBA	VBS	EXE	VBA	VBS	EXE
Maximum	13.4	22.8	5.1	29.7	31.4	5.8
Minimum	13.3	22.6	5.0	29.4	31.1	5.7
Average	13.38	22.71	5.03	29.52	31.27	5.73
STDEV	0.04	0.08	0.02	0.08	0.08	0.02

8.6.4 Interest Rate (IR) Sensitivity Calculation Analysis

Table 8.14: IR sensitivity calculation times analysis for each PC cluster node

	Calculation time (sec)					
	Binomial			Trinomial		
	VBA	VBS	EXE	VBA	VBS	EXE
Maximum	71	131	29	172	217	42
Minimum	35	67	11	82	105	13
Average	56.2	101.0	22.4	134.0	166.0	27.4
STDEV	13.10	23.00	6.74	32.20	40.30	9.72

Table 8.15: Distributed IR sensitivity calculation times analysis for PC cluster with imbalanced load condition

Imbalanced Condition	Calculation Time (sec)					
	Binomial			Trinomial		
	VBA	VBS	EXE	VBA	VBS	EXE
Maximum	6.2	11.5	2.5	15.1	19	3.7
Minimum	3.3	6.3	1.0	7.7	9.8	1.2
Average	5.08	9.11	2.02	12.1	15.0	2.47
STDEV	1.06	1.86	0.58	2.61	3.31	0.83

Table 8.16: Distributed IR sensitivity calculation times analysis for PC cluster with balanced load condition

Balanced Condition	Calculation Time (sec)					
	Binomial			Trinomial		
	VBA	VBS	EXE	VBA	VBS	EXE
Maximum	5.0	9.2	1.9	11.8	15.1	2.3
Minimum	4.6	8.3	1.8	10.9	13.6	2.1
Average	4.83	8.75	1.83	11.4	14.3	2.15
STDEV	0.14	0.25	0.05	0.3	0.44	0.08

Table 8.17: Distributed IR sensitivity times analysis for NUC cluster

Balanced Condition	Calculation time (sec)					
	Binomial			Trinomial		
	VBA	VBS	EXE	VBA	VBS	EXE
Maximum	13.4	22.6	5.1	29.2	30.8	5.8
Minimum	13.3	22.3	5.0	28.8	30.5	5.7
Average	13.36	22.46	5.05	29.03	30.65	5.75
STDEV	0.04	0.10	0.02	0.11	0.13	0.03

8.6.5 Calculation Time Improvement Analysis

The calculation time analysis shows that considerable improvements in calculation time for both balanced and imbalanced load conditions compared with the serial calculation method. In addition, as expected, the balanced load condition shows further improvements consistently for all the test conditions used for all applications. Table 8.18 lists the calculation time improvement in number of folds against the least powerful PC within the PC cluster for different types of scenario calculations. Table 8.19 lists the number folds calculation time improvement against the most powerful PC within the PC cluster for different types of scenario calculations. In both cases, considerable improvement in calculation time is accomplished using the PC cluster for distributed process calculation. Table 8.22 list the average calculations time improvements for the most powerful and least powerful PCs within the PC cluster.

Notations used:

BL Load balanced condition

IBL Load imbalanced condition

VBA Excel VBA program

VBS VB Script program

EXE Executable (compiled) program

Table 8.18: Calculation time improvements in number of folds for each type of calculation against the least powerful PC within the PC cluster

Test Type	**Binomial**						**Trinomial**					
	VBA		**VBS**		**EXE**		**VBA**		**VBS**		**EXE**	
	IBL	**BL**	**IBL**	**BL**	**IBL**	**BL**	**IBL**	**BL**	**IBL**	**BL**	**IBL**	**BL**
CB Price	12	14	11	14	11	17	11	16	11	15	12	20
IV	11	15	11	14	11	15	11	15	11	15	11	16
IV Impact	11	14	11	14	12	15	11	14	11	15	11	17
IR Impact	11	14	11	14	12	15	11	15	11	14	11	18

Table 8.19: Calculation time improvements in number of folds for each type of calculation against the most powerful PC within the PC cluster

Test Type	Binomial						Trinomial					
	VBA		VBS		EXE		VBA		VBS		EXE	
	IBL	BL	IBL	BL	IBL	BL	IBL	BL	IBL	BL	IBL	BL
CB Price	6	7	6	7	4	6	5	7	5	7	3	6
IV	6	7	5	6	5	6	6	7	5	7	4	6
IV Impact	6	7	6	7	5	6	6	7	6	7	4	5
IR Impact	6	7	6	7	4	6	5	7	6	7	4	6

Table 8.20: Average calculation time improvement in number of folds for the most powerful and least powerful PC within the PC cluster

PC Cluster Node	IBL	BL
Low-power PC	11	15
High-power PC	5	7

8.6.6 Cluster Comparison

During the various tests conducted on the PC cluster that was built from PCs and workstations, a number of issues were encountered, particularly with power consumption and heat dissipation. Hence, most of the testing was carried out during the weekends when the office overall power consumption is low. This is a major downside for using the PC cluster as a viable long-term distributed processing solution; therefore, the PC cluster is only to be used for testing and applying various load balancing algorithms. The PC cluster is a suitable candidate for load balancing algorithm testing due to its nonlinear nature of hardware configurations. Even though the PC cluster is being used during the testing phase, it will not be used in the production environment due to its high power consumption and requirement of a large amount space to set up the cluster. Meanwhile, the NUC cluster is uniform with low power consumption and is compact in size; it has considerable advantage against the PC cluster, and it will be used in the production environment. Further improvements and enhancements will be implemented to improve the cluster performance in future developments.

Due to its compact size, comparatively low cost, low power consumption, and high reliability, it is also possible to have multiple clusters using NUC-type SFF computers for dedicated user groups or application groups. Hence, more emphasis will be placed on building clusters using NUC-type SFF computers for future distributed processing developments for company-wide applications. The total power consumption of the PC cluster is considerably high, as expected; the NUC cluster's power consumption is low, and each NUC computer consumed the same amount of power because all the NUC computers have similar hardware specifications. Meanwhile, the PC cluster consists of various old PCs and workstations, and the power consumption of each node varies depending on the computer model. The total cluster power consumption during the peak performance is a crucial factor for calculating the energy efficiency of the cluster and the PC cluster's energy efficiency is relatively poor compared with the NUC cluster. The NUC cluster is the most energy-efficient cluster for the company, and for future cluster implementations, NUC-type computers are highly recommended due to their better performance on heat management and power consumption. The NUC computer's idle measured power is 5.5 watts and measured peak power is 7 watts for each calculation node.

The cluster operating noise levels are measured using Decibel Meter; the PC cluster produces more noise during the operations, and no measurable noise for the NUC cluster. For the PC cluster, most of the noise comes from PC fans, as each PC has multiple fans for cooling the CPU and power supply. Because the PCs used within the PC cluster are a few years old and due to possible mechanical faults, some PCs are noisier than other ones, and in addition, due to the high level of heat dissipation, all the fans operate at optimum levels to keep the system cool enough to operate continuously. Hence, the PC cluster is considerably noisier than the NUC cluster that has no noise because no moving parts in the NUC computer. The NUC cluster's measured noise level is 35 dB that consists of the ambient noise.

The space requirement to set up the cluster is also an important consideration due to the operational cost of the space per year per square metre. The space required to set up the PC cluster is considerably large compared with the NUC cluster, and this is one of the major advantages of SFF type computer clusters. The PC cluster requires approximately 24 times more space than the NUC cluster; however, due to heat dissipation and ventilation requirements for the PC cluster, it may require around 50 times more space than the NUC cluster in a production environment.

Heat dissipation data is calculated using the computer specification provided by the manufacturer, and each computer's heat dissipation is defined by thermal design power (TDP) and measured in watts. Maximum and minimum TDP ratings are assigned depending on how the computer is used. For workstations and PCs, the TDP values are considerably higher than small form factor (SFF) computers such as NUCs. The NUC TDP is only 5 watts maximum, and the PC cluster calculation node PC TDP varies depending on the model. Heat produced by the overall cluster is one of the important factors when setting up the cluster. Due to the cooling requirements and dedicated space required for managing the heat, the PC cluster requires a dedicated and temperature-managed environment; however, for the NUC cluster, no need for cooling due to its considerably low heat dissipation. Hence, the NUC cluster can operate at room temperature.

The total weight of the cluster is one of the important factors for setting up an appropriate server rack or computer table rack, and in addition, it is important for ease of relocation of the cluster. The PC cluster is considerably heavier than the NUC cluster at 20 times the weight of the NUC cluster, and the weights are calculated using the computer manufacturer's data. Table 8.21 lists PC and NUC cluster comparison.

Table 8.21: PC and NUC cluster comparison

Parameter	PC Cluster	NUC Cluster	Notes
Idle power (watts)	1,060	55	Measured
Maximum processing power (watts)	1,570	70	Measured
Idle noise (dB)	46	35	Measured
Maximum processing noise (dB)	51	35	Measured
Space required (m^3)	0.2890	0.0122	Calculated using manufacturer-provided parameters
Heat dissipation (watts)	890	50	Calculated using manufacturer-provided parameters
Weight (Kg)	139.5	7	Calculated using manufacturer-provided parameters

8.6.7 Program Performance Analysis

For testing different program platforms to investigate the calculation time taken to complete the CB theoretical value, the same mathematical functions are used for testing with same input parameters with minor changes in the programming code to match with each platform-specific programming instance. Currently, MS-Excel-based mathematical models are used by Northwest; however, these models can be modified to work with scripting languages such as VBS or any executable programs like .NET languages, C++, or similar. The speed tests were carried out using the CB theoretical value calculation model using the binomial method. The results show that for small calculation steps, the calculation time difference is small for each platform compared with large step size, and this is due to the exponential nature of the mathematical

model that uses the binomial method. For speed comparison testing, the following three programming platforms are used: MS-Excel 2010 (VBA + MS-Excel functions), VBScript (VBS), and VB6 (EXE). Figure 8.1 shows calculation times for varying step sizes using MS-Excel-VBA, VBS, and EXE programs and Table 8.22 lists programming platform comparisons. As expected, VBS is the slowest in the group; however, VBS has many advantages such as system-level programming and simple programming interface. VBS is not suitable for heavy program executions compared with fully compiled EXE-type programs. Meanwhile, the MS-Excel and EXE program shows a small difference within small step size ranges, and for larger step size calculations, the fully compiled EXE program has better calculation performance compared with MS-Excel and far better calculation performance than VBS. Hence, for compute-intensive financial calculation models, compiled software development is better suited, and this is under investigation as part of the software development strategies within the company that are in progress.

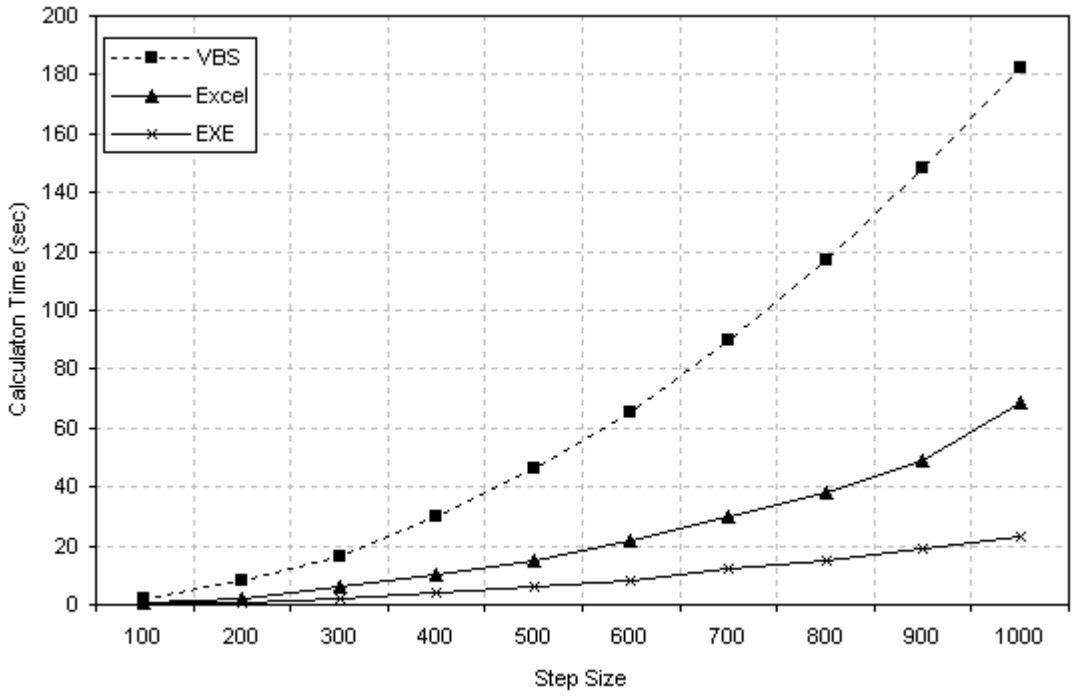


Figure 8.1: Calculation speed comparison for programming platforms

Table 8.22: Programming platform comparison for Northwest

Platform	Advantages	Disadvantages
Excel VBA	• Fully integrated with business suites and applications • Users are familiar with MS-Excel applications • Advanced users and quant developers can write programs using VBA • Many business-related built-in functions • Used as Rapid Application Development Tool • Changes can be made quickly • Easy to troubleshoot • Can make changes on the fly • Supports DLL and ActiveX	• Unsecure programs • Prone to coding and logical errors • Pseudo-compiled code • VBA has various limitations • Program has limited portability • Nonuniform and discrepancy programming • No programming structure • Difficult to audit the code • Slow execution time compared with fully compiled EXE programs
EXE	• Execution speed is high • Structure programming • Auditable code • Secure code • Securely distributable • Run as a standalone application • Highly portable	• Requires VB programming knowledge • Cannot make changes on the fly • Difficult to troubleshoot • Compiled code
VBS	• Fully integrated with Windows operating system • Easy to program • Many system-level management built-in functions • Used as scheduling, monitoring, and administration tool • Changes can be made quickly • Easy to troubleshoot • Can make changes on the fly • Highly portable	• Requires VBScript programming knowledge • Unsecure programs • Prone to coding and logical errors • Code not compiled • Various coding limitations • Slow execution time compared with fully compiled EXE programs

8.7 Research Impact Analysis

The research conducted is to investigate and implement an innovative and suitable distributed process-based calculation cluster within the company's network infrastructure to facilitate the company in using compute-intensive systems and applications. The investigation and test results have shown that it is feasible to implement a bespoke-type distributed processing system within the company using different types of processing devices such as workstations, PCs, conventional servers, and virtual servers. In addition, investigation of small form factor (SFF)-based computers has proved that these types of computing devices are the most effective solution for dedicated calculation clusters due to their small size, low cost, low power consumption, and high reliability. Hence, these types of clusters are better suited for dedicated calculations and testing, especially for research and development that requires large amounts of dedicated computing power for 24x7 operations. Different types of tests are performed using the company's compute-intensive applications that are used for various scenario analyses and stress testing. The test results show that consistent improvement in calculation time can be achieved by using distributed calculation rather than using a single server or workstation to calculate the same datasets. Currently, the following applications are benefited by using distributed calculations.

8.7.1 Scenario Analysis and Stress Testing

This application is used for various calculations on managed portfolios at the fund level, book level, and position level. In addition, it is used for predicting "what-if" scenarios based on expected market condition movements and using certain derivative products to see what effect they will have on the company's managed portfolios. Hence, the calculation time varies depending on the type of analysis performed, and by using a single HP Z420 workstation; it usually takes around one to three hours. By introducing workstation cluster-based calculation, the total calculation time has been reduced to in the region of 5 to 15 minutes. Even better performance is observed if the calculations are performed as a batch process during out-of-office hours.

8.7.2 Derivative Price-Related Data Calculation

This application is used for calculating different types of derivative price-related calculations during the trading hours to facilitate the traders, fund managers, and analysts. Hence, this application calculates the required data every 2 hours using a system timer and the full calculation takes just over an hour using a single HP Z420 workstation; the calculation time depends on the portfolio size. By introducing the workstation cluster, the calculation time has been reduced to 5 minutes or less depending on the time of the trading day. Furthermore, now it is possible to reduce the calculation frequency, and in effect, the data accuracy is improved by using up-to-date data from different financial feeds.

8.7.3 End-of-day Profit and Loss (PL) Calculation

This process is performed by a group of applications in coordination with various data sources. Each trading day, various end-of-day processes have to be completed as part of the business functions, and one these processes is the calculation of daily, weekly, monthly, and yearly profit and loss for all the managed funds. This process is a highly time-consuming task, and using the HP Z420 workstation, the total calculation time is around an hour. The calculation process is only performed after the close of the trading day when the market is closed, that is, after 4:30 PM. Hence, if any errors are found in price marking or trade booking, the PL calculations have to be recalculated, and this will cause further delays in daily reporting. Implementing the workstation cluster in coordination with the server cluster has linked SQL servers, and in addition, due to some changes made to the PL calculation systems, the overall calculation time has been reduced to less than 3 minutes. Hence, even if any erroneous data is identified after the calculation, the recalculation process is within the acceptable time limits to complete the end-of-day process.

8.8 Recommendations

A few recommendations are made to the company based on the investigations carried out during this research. These recommendations are mainly focussed on the possible improvements that can be made to the existing IT infrastructure and systems and what can be done to improve the calculation efficiency of future development of compute-intensive applications and systems. Some recommendations are made based on practical improvements. How these improvements can be brought to the systems currently used, and in addition, to future developments of the systems.

8.8.1 Structured System Design

All the applications and systems that are used in the company have been developed using Rapid Application Development (RAD) and the Dynamic System Development Method (DSDM). These applications and systems are developed as bespoke propriety systems. Some of the applications were developed by quantitative analysts for their use as prototype systems using MS-Excel, and these systems are continuously modified and employed as live systems. Furthermore, these systems are intended to be used as prototype systems for short time periods for testing. Hence, no structure is in place within the systems to support larger datasets or other critical functionalities that are required for live systems. Therefore, it is highly recommended that the users should closely work with the IT team before they start to work on the applications or systems that they are supposed to use for their day-to-day work or for the purpose of analysis and research. The systems used and the business models are highly specific to the company; hence, the distributed processing system that was developed is specifically tailored to the company's business model. The IT infrastructure and systems are designed in such a way that the company can expand the business if the market conditions are favourable and able to downsize to run the business profitably during stressed market conditions. This type of flexibility is a crucial part of the business' success and continuity during stressed business and market conditions.

8.8.2 Technology and Programming Improvements

Most of the current applications are based on the VBA programming model and are limited to certain Microsoft applications such as MS-Excel and MS Access. Even though it is relatively easy to implement, this method has a major downside, that is, it has various programming limitations on improving the programs to utilise the latest programming techniques available for parallel processing or to incorporate specific hardware like GPUs for improving the processing speed. Hence, for high-speed processing models, it is highly advisable to use the programs that are capable of supporting the parallel processing method using threads and multi-core processing, and incorporating specific hardware such as co-processors and GPUs. The following implementations will improve the development process and facilitate the company's business operations:

- Using programming languages such as the .Net framework and C++ for develop specific parallel applications.
- Using specific packages for research and developments, such as MATLAB, Simulink, and Mathematica or similar. These software packages have built-in parallel processing libraries that can be used for development.
- Converting user-defined VBA codes to compiled binary code programs that can be used as add-ins to MS-Excel applications. This can facilitate that the compiled program to be used with different software development environments such as .Net framework programs.
- Replace servers and workstations with multi-core and many-core CPUs with lower power consumption. By replacing the hardware with low power consumption and less powerful CPU cores configured as multi-core and many-core architectures, the cost will also be reduced compared with the hardware featuring more powerful CPUs.
- Building a larger-scale dedicated cluster with more than 100 calculation nodes using low-cost and low power consumption SFF computers that have multi-core CPUs for research and development. The cluster can be housed in the company's remotely located disaster recovery site with high-speed internet links and can be available for 24x7 operations.

8.8.3 Improvement on Bespoke Design

When the company's business started 18 years ago, in the early days, there were several attempts made to use the third-party systems, especially using derivative pricing models for various scenario calculations and risk profile analyses. However, the solutions supplied by the third-party providers had the following problems:

- The models used were generic and not suited for certain specific instruments that were traded by the company. Building these types of financial models takes time, and in addition, costs extra, and these types of specialised models have to be designed by the providing company and to be incorporated within their systems.
- The cost structure of using third-party systems was complex and expensive for a small company like Northwest.
- Implementations and modifications were highly complex and time-consuming, and have to comply with the provider's methods and data structures.
- Often, the expected results are not consistent or are incorrect and require multiple calculations to produce an acceptable result set, which adds extra cost and time.

Hence, the company decided to build its own financial models and scenario analysis system using existing technologies such as MS-Excel and VBA that are familiar to most of the analysts and researchers. Bespoke-type proprietary financial models have various advantages as follows:

- Financial models and their applications are particularly designed for the business requirements and unique to the company's business model.
- Used existing and familiar technologies to design and develop required financial models and applications that are needed for day-to-day analysis. Hence, the development process, improvements, and maintenance are less complex to manage.

- Financial models and applications are used in the company, and no need to pass the sensitive data to a third-party company; hence, higher levels of data security and business process security.
- Due to the use of RAD and DSDM methodology for the development process, it is comparatively faster to implement new solutions compared with solutions provided by third parties.

Having numerous advantages of using bespoke-type systems that are developed internally with required business logics, a serious downside to this approach is that the internally developed financial model-based applications are highly compute-intensive and more difficult to scale. This is due to the financial models being based on multiple layers of nested interactive loops and being inherently nonlinear. Hence, to overcome the time limitations, reduced sampling periods and minimal scenarios are used to manage the calculation time requirements. To improve the calculation speed, the applications require a faster and more powerful server, and this is not a suitable approach for long-term operations due to increasing cost and power limitations. This is one of the primary reasons for the initiation to investigate the possible alternatives; that is, the bespoke-type distributed processing system that can be used to improve the calculation speed of these applications.

Another area of business operations is executing batch-type processing to calculate certain numbers that are used for business-wide internal reporting and external reports for investors and brokers on a daily basis. This is a number crunching process using a number of SQL databases, financial feeds from various providers, and CSV, TXT, and XML files. The entire process takes a few hours to complete, and the batch process is executed at night by the schedulers. Because many of the processes in the batch are interdependent, if one of them fails, then all of the dependent processes have to be recalculated the full batch process to be completed. This is a highly time-consuming task and causes serious delays. This is another business-critical process that needs to be improved and that is benefited by implementing distributed process-based batch processing.

8.9 Suggestions for Cluster Improvements

The results from the application testing data have shown that the implementation of various clusters for distributed calculation can improve the overall real-time calculation time. This will facilitate the analysts, traders, and fund managers to test the trading algorithms in real-time environments. The traditional way of testing is to test certain scenarios with limited parameters; it usually takes a few days to complete the full test, and if any errors are detected or modifications are needed, then it will take even longer to complete the test. However, by implementing distributed calculation using various calculation cluster configurations, the time taken to complete the testing process is reduced to within an hour or less. Further calculation time reduction can be accomplished by implementing static and dynamic load balancing and in addition, using dedicated calculation clusters. Even though the calculation clusters using workstations, servers, and SFF computers are highly suitable for testing and batch processing scenario analysis types of applications, for real-time trading scenario calculations, the current distributed processing setup is not suitable due to the following reasons:

The clusters are nonlinear and require complex load balancing algorithms.

- Cannot be reset or calibrated during trading hours.
- Clusters are part of the company network and can cause calculation delays.
- Server cluster resources continuously change due to server virtualisation.
- Network traffic will affect the calculation time.

Hence, for the real-time trading environment, it is highly advisable to have a dedicated calculation cluster with the following characteristics:

- Cluster must be fully separated from the user networks.
- Cluster must be fully available 24x7.
- Cluster must have redundancy-cover calculation nodes (auxiliary nodes).
- Cluster must be built using the same hardware and software configurations for all the calculation nodes.
- Each calculation node must have a multi-core CPU.

Building a fully separated dedicated calculation cluster for real-time trading is the most suitable and cost-effective solution for the company. The following recommendations are made for building the dedicated cluster:

- Use low-cost SFF-type computers to design systems with multi-core CPUs.
- Use exactly the same hardware and software configurations for the calculation cluster to achieve hardware and software uniformity that in effect reduces the load balancing complexities.
- Configure the cluster in a separate network setting with tightly controlled security settings.
- Configure multiple logical clusters within a single physical cluster with multiple auxiliary calculation nodes for high reliability and improved robustness.

General types of distributed computing systems that use various devices as computing nodes across multiple networks introduce many new challenges that are not usually considered an issue in a single machine or a private network. The following are some of the concerns:

- Security is the main concern in widely distributed systems.
- Node failure rates are higher than in a dedicated parallel machine.
- Nodes may not be consistent and can vary.
- Nodes and their connections in the processing network may not be homogeneous.
- Some algorithms simply do not scale well over slower networks.

For distributed process communications, even gigabit-speed networks are slower in speed in comparison with processors existing on a purposely-built bus. This limitation is requires serious consideration when designing a distributed computing system and one has to take account of the nature of the computing job that will be executed within the system. However, the aforementioned issues have less impact in the case of Northwest's distributed processing system due to its full integration with the company's network as dedicated computer clusters on a private network.

8.10 Chapter Summary

Number of advantages and disadvantages are found when using the different types of calculation clusters. The dedicated calculation cluster grid that described in Chapter 6 has an advantage that it can be fully utilised for 24x7 operations. Unlike the workstation cluster that was described in Chapter 4, it utilises the user workstations, and the dedicated calculation grid is isolated from user access and fully dedicated for certain applications. The NUC cluster has many advantages compared with the PC cluster; even though the NUC computers are less powerful than the PCs that are used in the PC cluster, the NUC computer features low power consumption, smaller size, and lower cost. However, the PC cluster is useful for testing different types of load balancing algorithms due to its nonlinear hardware configurations, but the PC cluster is not suitable for continuous use in production environments due to its high power consumption, requirement for larger space, and higher rate of failure. Hence, the SFF type of computer-based clusters will be expanded and improved by adding more calculation nodes, and in addition, building separate clusters in the future for long-term research and development within the company.

The main purpose of the dedicated calculation grid is to facilitate the research and development team and quantitative researchers to test and simulate different types of financial models and trading scenarios in real time. Currently, this process is done by quantitative analysts and researchers using single workstations or servers, and in certain cases, the full analysis takes many hours to a few days to complete. Hence, some of the analysis is carried out as batch processes during nights and weekends and sampling frequency is reduced to improve the calculation time. Regarding the implementation of the dedicated calculation cluster grid for processor-intensive calculations, the test results have shown considerable improvement in reducing the overall calculation time for different scenario calculations. The tests and simulations that were carried out on the dedicated calculation cluster are similar to the scenario analysis carried out by the researchers and quantitative analysts. The test results have proved that implementing the dedicated calculation cluster with appropriate software- and hardware-specific load balancing algorithms greatly improves the calculation

efficiency for many of the bespoke mathematical models used in the company. Both clusters are undergoing continuous improvement to fine-tune their performance for various calculation scenarios, and more tests and simulations are being carried out; these results and analyses show possible alternatives for calculation cluster designs. A number of changes have been made to the distributed processing controller software and calculation node controller's software to include dedicated calculation grid cluster nodes, and in addition, a few changes are made to the SQL server database to incorporate the cluster nodes within the distributed processing control system. The message passing between calculation nodes and distributed processing controller is similar to the technique that was used in Chapter 4 with a few modifications.

The load balancing algorithms tested have proved that it is possible to incorporate company-specific application-related parameters with the general hardware- and software-related parameters that were investigated in Chapter 5. In addition, implementing a calculation index and usage index for each calculation node has also proved that the bespoke design-based distributed processing controller software is capable of handling different types of calculation clusters with varying calculation node parameters. Furthermore, the controller software is capable of selecting clusters as physical clusters, logical clusters, or both depending on process requirements and calculation time criticality. Even though the controller software efficiently manages both static and dynamic load balancing algorithms, the implementation of dynamic load balancing algorithms is in its simplest forms by using a set of well-defined rules. However, for static load balancing, the controller software works in coordination with the SQL server database to collect various data continuously from each calculation node for adaptively fine-tuning the static control algorithms.

The load balancing algorithm tests that were evaluated on the different calculation clusters show that the static load balancing is well suited for the company-wide applications compared with dynamic load balancing. However, dynamic load balancing has its own benefits regarding protecting the calculation cluster in case of node failures and unexpected calculation delays. Hence, dynamic load balancing acts

as a safety protection mechanism and only be activated by the distributed processing control manager software if any problems are detected during the calculation phase based on a given set of rules. The static load balancing is the primary load balancing mechanism that is used to allocate calculation tasks to each calculation node based on various parameters. Some of the parameters are static and others vary with the applications used and time of the day that the calculation is executed. Using distributed processing systems for critical applications, the complexities must be reduced to the minimum to avoid serious errors or calculation delays. Hence, using dedicated clusters with each calculation node having the same hardware and software configurations is the preferable option for critical applications; in addition, this will reduce the static load balancing complexity due to uniform calculation nodes used in the cluster.

The server cluster configuration that uses the spare conventional servers has the same problems encountered that are similar to the PC cluster using spare PCs, such as high power consumption, higher-level noise, larger space requirements, the requirement of a dedicated cooled room, and low reliability. Hence, this type of server cluster has the same problems faced by the PC cluster as mentioned earlier. However, these types of server clusters are useful for testing the load balancing algorithms due their nonlinear nature of hardware and software similar to the PC cluster. Therefore, the spare PC-based and spare server-based clusters are used for testing the various load balancing algorithms and distributed processing control software. Hence, due to the aforementioned problems are related to these types of clusters, the original proposal to include these types of clusters as part of the company's distributed processing system will not be examined further. Meanwhile, users' workstations and virtual servers continue to be used as multiple calculation clusters, especially for out-of-office-hour batch processing tasks and on-demand compute-intensive tasks, and for building dedicated clusters, SFF computers should be used. The recent development in multi-core and many-core architecture microprocessors for smaller devices such as mobile phones is driving the processing power improvements and, going forward, these devices will power the small form factor computers with multi-core and many-core architectures that, in effect, improve the processing power.

9 Conclusion

The investigations conducted in this research have demonstrated the original contribution to the distributed processing technologies that uses highly specific design methods of building hybrid types of calculation clusters as distributed processing based calculation systems. The research conducted herein is an innovative and original research by investigating distributed processing technologies that contribute to the sponsoring company, their practical implementations in the company, and detailed understanding of applicable techniques. Furthermore, a substantial body of knowledge is acquired in distributed processing technologies that are relevant in the forefront of this academic discipline and in the area of professional practice. In addition, the original approach of system design is presented that utilises different types of hardware and software in the company as an intelligent distributed processing system that is unique to the company. However, this new approach has the potential to be implemented in any company that has multiple processing devices such as servers, workstations, and laptops or similar processing devices. Furthermore, the new methods of hybrid cluster implementation that demonstrated are unique and adaptable to changing business requirements. This unique method is based on bespoke design that is capable of incorporating existing legacy applications as well as supporting new application developments.

The development and implementation of new methods and approaches that are based on the original concept of distributed and parallel computing to implement a distributed processing system for the company proved to be considerably successful. Furthermore, using Windows network topologies with simplified cluster architecture to perform compute-intensive calculations proved that it is feasible within the business environment and applications used in the company. This approach has greatly reduced the computing time for a number of compute-intensive applications used in the company, and performing various simulations using proprietary mathematical models. It has also proved that the bespoke-type distributed processing approach has the potential to be implemented in the real-time trading and risk calculation and portfolio management scenarios within the company. In addition, it is highly useful for

facilitating the company's internal research and development and for developing proprietary financial mathematical models for testing new trading strategies. The distributed processing-based calculation system implemented is relatively simple to adopt with minimal changes to the existing systems and applications and with little or no extra cost.

The initial investigation to test the calculation time improvements by using a workstation cluster for certain types of MS-Excel applications has shown that it is possible to build a simple and easily manageable distributed processing cluster utilising existing hardware and software without complex programming or third-party software. The test results have proved that it is feasible to build a practical Beowulf-class distributed processing system using existing hardware and software within the company and also it is possible to design and develop bespoke-type software for managing the cluster. In addition, the distributed processing cluster system is compatible with existing applications and systems that are used in the company. The workstation cluster is based on loosely coupled design and can be used as a dedicated or non-dedicated distributed processing system depending on the requirements, and it can easily be configured accordingly. Another advantage of the loosely coupled design is that since it uses the existing network infrastructure technologies, no need for specialised networks or interconnects; hence, the calculation nodes can be added or removed by just attaching to or removing from the existing network. It is also possible to allow individual workstations to join dynamically as cluster nodes and leave the cluster without disruption to the overall operations; this whole process is managed by the distributed processing controller. The main advantage of this type of loosely coupled cluster design is the ease of the configuration, as well as the management of the cluster hardware and software, and it has been proved that any company that has a number of PCs or workstations can set up these types of loosely coupled clusters with minimum effort.

The detailed investigations have shown some promising results that the company's bespoke applications can be modified and implemented with the developed distributed processing system. The test results confirm that the new and original approach has provided the expected calculation time improvements within a controlled environment. In addition, it has proved that it is possible to continuously design and develops a bespoke-type distributed processing system using the company's existing hardware and software to provide solutions for calculation time limitations problems currently faced by the company. Even though the test results show considerable improvement in calculation time for particular applications, in the real-time trading applications, many complex issues have to be addressed; these issues are under investigation, and suitable solutions need to be implemented. However, the initial investigations have successfully demonstrated the possibility of implementing a solution for improving calculation efficiency for compute-intensive calculations using existing technologies in the company. In addition, this research aimed to find an innovative, cost-effective, and most efficient way to utilise the workstation cluster for distributed processing with easy-to-use and simple-to-manage functionalities for long-term research and development.

During the various investigations carried out to modify the existing application to work with the designed distributed processing clusters, fundamental problems have been identified regarding data structures, programming structures, and input and output methods. These are due to the fact that the applications were never intended to be used in parallel or distributed processing environments in the first place, and were only meant be used in a single server or workstation. Some of these applications were created as prototype applications with small datasets, and the applications were developed by quantitative analysts and risk analysts for their own use with MS-Excel and VBA. Eventually, these applications became production applications with continuous modifications occurring with many years of usage, and the data consumption became large. All these factors have a serious impact on the application performance due to inconsistent data and programming structures.

While investigating performance analysis of different types of cluster configurations, a few undesirable effects were found for certain types of clusters such as large amounts of power consumption, heat dissipation, and space requirements, and these clusters are built using spare workstations, PCs, and servers. It was proposed to the company in the early stages of the investigations that these clusters would be part of the distributed processing systems; however, due to many disadvantages, these types of clusters are not suitable to be used in a production environment and should only be used as part of the testing and prototyping clusters. Meanwhile, the clusters that are built from SFF computers show considerable advantage over other types of clusters such as workstations and virtual server clusters in terms of cost, power consumption, and the overall cluster size. Even though the workstation and server clusters have more processing power per calculation node compared with SFF computer calculation nodes, for building a dedicated calculation cluster, SFF computers are the most suitable option. Because of recent developments in SFF computers with more processing power and, in addition, utilising multi-core and many-core architectures with less power consumption, smaller size, and reduced cost, it is possible to build multiple dedicated calculation clusters for specific use. It is also possible to upgrade the clusters with newer and improved computers as technology progresses.

The investigation on hybrid clusters was performed by using a combination of multiple cluster configurations utilising various processing devices that are available in the company such as currently used workstations, servers, and unused spare PCs and servers and additionally utilising SFF computers as dedicated clusters in P2P networks; this has shown considerable improvement in compute-intensive applications. The resultant data collected from various applications testing using a combination of clusters shows that the implementation of a combination of various clusters for distributed process calculations can improve the overall real-time calculation time. This will eventually facilitate the analysts, traders, and fund managers to test the trading algorithms in real-time environments. In the traditional way of testing, the tests are performed on certain scenarios with limited parameters, and it usually takes a few days to complete the full test, assuming that no errors, and if any errors occur, or modifications are needed, then it will take longer to complete the

test. However, by implementing distributed process-based calculations using a combination of cluster configurations and static and dynamic load balancing, the time taken to complete the full testing process is reduced to within an hour or less. In addition, the test results show that consolidating various processing units into an intelligent hybrid calculation cluster that can be used for 24x7 operations with high throughput is another possible way of improving batch-processing operations.

An application used in the company that is used for risk scenario calculation on a daily basis to process various calculations at each position level and portfolio level. It is usually takes between two to three hours when used in a single server depending on the number of open positions and type of derivative products in the portfolio, and the calculation process is performed as an overnight process. Hence, the calculated data is available for users before the market opens. However, if any errors in the calculation are due to incorrect data or to software or hardware failure, the whole calculation has to be repeated, and this is a time-consuming task during office hours. Hence, the previous day's data will be used, and this is not an ideal solution. Using the distributed process-based calculations method with domain decomposition has reduced the calculation time to between 10 and 15 minutes. This is a considerable time improvement regarding overall calculation time, and if any errors occur during the overnight calculation process, then the whole calculation process can be completed within 20 minutes during office hours. This is one of the major improvements made to an existing application using distributed processing.

The calculation clusters using workstations, servers, and SFF computers are highly suitable for testing purposes of internal research and development, various types of batch processing, and scenario analysis. However, for real-time trading scenarios, the current distributed processing setup is not suitable due to tighter controls being in place by compliance and regulation to safeguard against program-based trading errors. Meanwhile, these clusters are used to calculate various parameters in real time using trading support applications that facilitate the traders to make trading decisions under highly volatile market conditions. Furthermore, when the company decides to embark

on a program trading strategy, then it is possible to build highly efficient and dedicated distributed process-based calculation clusters from what has been learned from this research. For real-time trading environments, it is highly advisable to have dedicated calculation clusters with multiple standby calculation nodes for safeguarding against possible calculation node failures. In addition, for high-speed trading situations, using dedicated multiple auxiliary calculation clusters will minimise the calculation delays and data inconsistencies.

The distributed processing cluster usually has many processing nodes, and these processing nodes are bound to have a component failure or become temporarily inaccessible across a network at some point during their operational lifetime. The probability of a single node failure relatively is high compared with multiple node failures simultaneously; hence, need to have a certain type of redundancy protection such as task migration techniques and using auxiliary standby nodes. In the case of Northwest, the workstation and server hardware is highly reliable and maintained at a high standard as part of the regular IT infrastructure management. In addition, operating systems and software used are also maintained with appropriate updates and security checks. Hence, the processing node failure is kept minimal by regular checks and maintenance procedures.

In recent years, small hedge fund management companies like Northwest have been facing serious challenges in attracting larger institutional investors for long-term investments due to the negative effects caused by the serious financial crises that happened during the last eight years. During this period, most of the small hedge fund management companies were closed down and all or most of the investments were lost, and even some of the larger investment banks also closed down with billions of dollars lost. Hence, in recent years, after 2008, the financial regulatory bodies and the institutional investors have introduced tightly controlled compliance tests and due diligence investigations to ensure the hedge fund management companies are fully compliant with current regulations and restrictions. This has initiated various internal processes within the company to comply with newly introduced compliance

requirements and regulations. One of the due diligence processes that were introduced as part of this, is the IT due diligence that is used in larger organisations where the IT and systems are usually managed by dedicated internal departments or by specialised third-party companies. However, due to the recent introduction of cyber security procedures by various government organisations, the IT due diligence procedures is applied across the entire investment finance industry regardless of company size. This, in effect, provides strong indications to the investors and compliance authorities that the companies concerned fall within the strict compliance requirements and recommends these companies accordingly.

Hence, the company has to maintain the IT infrastructure and systems at a high standard with appropriate supporting documents to satisfy the routine compliance and due diligence tests. Another area of investigation conducted by investors and prospective investors on a regular basis is to check whether the company has adequate measures to protect the investments under stressed market conditions, investment strategies used, and "what-if" scenarios used to predict the outcomes using various financial models. This process is laborious and is spread across all the business areas; in larger companies, these types of processes are outsourced to highly specialised third-party companies with confidentiality agreements. However, for a small company like Northwest, using third-party companies is far too expensive; hence, all the analyses are performed using bespoke-type financial models and scenario calculations that are highly compute-intensive and take a long time to complete on a single workstation or server. Consequently, this is one of the main areas of improvement made by implementing bespoke-type distributed processing technologies in the company. Therefore, the implementation of distributed processing has positive impact on the hedge fund management technologies, particularly for small to medium sized hedge funds that have limited IT resources. It facilitates the business to perform various compute-intensive calculations that are impossible to perform within the required timeframes using serial calculation methods, and in addition, provides the opportunity to carry out in-house business-specific quantitative research and development. Furthermore, it promotes the competitive edge to small hedge fund management companies like Northwest to attract larger institutional investors.

Distributed processing research and development over the past 20 years has facilitated many businesses and universities to have a new way of implementing supercomputing for their own requirements at lower cost compared with traditional supercomputers. Distributed processing methods have opened up a new direction of research to improve the computing power using loosely coupled low-cost commodity components and multi-core single-chip-based parallel processing. Another factor that is driving parallel design is the limitation of miniaturisation due to the current physical laws; hence, chip-manufacturing companies are moving towards distributed processing in a single physical chip using many-core architectures. For loosely coupled distributed processing systems, the improvements in technology related to commodity components have a greater direct impact than the fabricated technology-based systems. The primary focus of the conducted research is based on designing distributed processing systems using loosely coupled processing nodes. Hence, this will have the advantage of replacing the processing nodes easily and adding more nodes when required with minimal alteration to the existing systems. This research provides an alternative approach to designing distributed processing systems that utilise the existing hardware and software that are used in the company at no additional cost or at minimal cost. Therefore, the outcome of the research directly benefits the sponsoring company by providing extra computing power for many calculation-intensive financial software programs and applications that are currently used and for future developments.

Another concern regarding innovative technology implementation within the investment finance industry is that the industry is highly regulated with various compliances both internally and externally. Due to this type of tighter regulatory compliance, the investment finance industry is slow to catch up with technology advancements and implementations; as a result, a high level of legacy systems are used in the investment finance industry compared with other industries. In addition, the investment finance industry tightly controls its internal research and development and rarely shares its research with other researchers in the field or other businesses. In comparison to larger investment banks, smaller hedge fund management companies like Northwest have more flexibility in technology and system implementations;

however, any changes made to the IT and systems, especially related to trading, portfolio management, risk management, and similar types of critical systems, have to be rigorously tested and fully documented for auditing. Even small companies like Northwest has undergo regular inspections and due diligence checks by investors and external compliance authorities. Hence, certain investigations are carried out in this research, such as dispersion trading and data processing using finance data feeds that were described in Chapter 7; these are not to be implemented until fully tested with tightly controlled test environments. Meanwhile, some internal applications used in the company that are compute-intensive have already started to be used within the distributed processing cluster in various forms and have shown considerable calculation time improvement compared with using a serial method. The most improved applications are batch process applications that are executed during out-of-office hours where all the processing units are fully available for use as calculation nodes and calculation time reduction from a few hours to a few minutes is achieved by utilising all the calculation nodes.

Most of the research in distributed and parallel processing is moving towards supercomputing-level implementations, and most of the recent supercomputer designs are based on certain types of combinations of distributed processing and parallel processing technologies. Hence, various improvements are made in using combinations of multi-core and many-core processors with co-processors for vector calculations, and in addition, a number of improvements have been made to processors, network interconnects, and high-speed data access technologies to improve the overall system speed. Because of the complexities involved in interconnects and data access in large-scale clusters, new methods of building cluster-based supercomputers that use a small-scale cluster as a single processing node. Hence, each cluster acts as a processing node and it has its own management systems that control the processing, communications, data storage, and memory management. Furthermore, large-scale cluster computing systems use large amounts of power and produce large amounts of heat; hence, need for efficient power management, air-cooling, and utilisation of power-aware processing techniques. Meanwhile, many areas of research utilise the custom-built Beowulf type loosely coupled clusters using commodity computers for their own research and development; in particular, most universities

take advantage of these types of clusters for their research. However, most of these types clusters are specifically built for certain types of research-oriented utilisation and use Linux platforms. In industry-based custom-built cluster utilisation, the benefits to the business are rarely published in the public domain due to safeguarding the business interests and business competitive edge. However, as demonstrated in this research, it is possible to utilise an original concept with a business-specific design that uses a new and original approach for a highly specific distributed processing system.

Distributed processing using commodity hardware and software has proved be a technological breakthrough and the concept of parallel and distributed computing is here to stay and is the future of computing. The key factors in the design of these types of supercomputing level of systems is that these are scalable, high-bandwidth, low-latency networks with low-overhead network interfaces. In addition, tightly coupled with the global operating system layer, the high-speed network allows users to see the resources on the network like processors, memory, and storage device disks as shared devices within a single system. The most important part of this technology is the opportunity for large-scale computing within everyday commodity components-based framework of interactive computing. The distributed processing system designed in this research is based on the bespoke-type first principle approach that is particularly suited for the company's infrastructure; hence, it can be modified as and when required with minimum impact to the business. Furthermore, bespoke-type utilisation of adaptive and self-tuning control mechanisms continuously fine-tunes the cluster performance and is able to function autonomously with varying parameters, and these parameters can be hardware, software, or business rules. The fundamental method and approach that is used in this research can be adopted by other small to medium-sized companies for their own distributed processing, regardless of the business type or area of business.

To implement successful and efficient distributed processing cluster systems, fault tolerance, load balancing, and resource management techniques are important components of the system implementation to ensure its robustness. Distributed processing using multiple processing devices that are available in the company such as processing nodes to configure company-specific processing clusters has positive impact on the hedge fund management technologies. This approach can facilitate the business to perform various compute-intensive processes that are currently impossible to perform within the required timeframes and in addition, provide the opportunity to carry out in-house business-specific quantitative research. Furthermore, it promotes the competitive edge for small hedge fund management companies like Northwest to attract larger investors by proving that they have the required computing capacity to produce the required financial analysis demanded by investors, prospective investors, and compliance authorities. Distributed process-based computing research holds significant promise, and much potential for researchers and businesses that can harness these technologies to reap the benefits it offers at various levels.

Bibliography

1. Kronenberg, N.P., H.M. Levy, and W.D. Strecker, *VAXcluster: a closely-coupled distributed system.* ACM Transactions on Computer Systems (TOCS), 1986. **4**(2): p. 130-146.
2. Leblang, D.B. and R.P. Chase Jr. *Computer-aided software engineering in a distributed workstation environment*. in *ACM Sigplan Notices*. 1984. ACM.
3. Baker, M. and R. Buyya, *Cluster computing: The commodity supercomputing.* SOFTWARE—PRACTICE AND EXPERIENCE, 1988. **1**(1): p. 1-4.
4. Kung, H., et al. *Network-based multicomputers: An emerging parallel architecture*. in *Proceedings of the 1991 ACM/IEEE conference on Supercomputing*. 1991. ACM.
5. Dolezal, R., et al., *HPC cloud technologies for virtual screening in drug discovery*, in *Intelligent Information and Database Systems*. 2015, Springer. p. 440-449.
6. Fischer, J., et al. *XCBC and XNIT-Tools for Cluster Implementation and Management in Research and Training*. in *Cluster Computing (CLUSTER), 2015 IEEE International Conference on*. 2015. IEEE.
7. Yeo, C.S., et al., *Cluster computing: high-performance, high-availability, and high-throughput processing on a network of computers*, in *Handbook of nature-inspired and innovative computing*. 2006, Springer. p. 521-551.
8. Luther, A., et al. *Alchemi: A. NET-based Enterprise Grid Computing System*. in *International Conference on Internet Computing*. 2005.
9. Becker, D.J., et al. *BEOWULF: A parallel workstation for scientific computation*. in *Proceedings, International Conference on Parallel Processing*. 1995.
10. Sterling, T.L., *Beowulf Cluster Computing with Windows*. 2002: MIT Press.
11. Amdahl, G.M. *Validity of the single processor approach to achieving large scale computing capabilities*. in *Proceedings of the April 18-20, 1967, spring joint computer conference*. 1967. ACM.
12. Gustafson, J.L., *Reevaluating Amdahl's law.* Communications of the ACM, 1988. **31**(5): p. 532-533.
13. Cox, J.C. and S.A. Ross, *The valuation of options for alternative stochastic processes.* Journal of financial economics, 1976. **3**(1-2): p. 145-166.
14. Connolly, K.B., *Pricing convertible bonds*. 2001: Wiley.
15. Opera_Report. *Opera_Report*. [cited 2016 March 16]; Available from: http://www.theopenprotocol.org/top/home
16. SFC. *Securities and Futures Commission (SFC)*. [cited 2016 March 2016]; Available from: http://www.sfc.hk/web/EN/index.html.
17. Connolly, K. and T. Kumar, *CB Financial Model Development* 2012: Northwest Investment Management (HK) Ltd.
18. Baker, M. and R. Buyya, *Cluster computing: the commodity supercomputer*. Software-Practice and Experience, 1999. **29**(6): p. 551-76.
19. Buyya, R., *High Performance Cluster Computing: Architecture and Systems, Volume I.* Prentice Hall, Upper SaddleRiver, NJ, USA, 1999. **1**: p. 999.
20. HFSB. *Hedge Fund Standards Board*. 2016 [cited 2016 June 2016]; Available from: http://www.hfsb.org/.

21. Elliott, L. *Global financial crisis: five key stages 2007-2011* 2011 [cited 2016 March 2016]; Available from: http://www.theguardian.com/business/2011/aug/07/global-financial-crisis-key-stages.
22. Beynon-Davies, P., et al., *Rapid application development (RAD): an empirical review*. European Journal of Information Systems, 1999. **8**(3): p. 211-223.
23. Stapleton, J., *DSDM, dynamic systems development method: the method in practice*. 1997: Cambridge University Press.
24. Faulk, S., et al., *Measuring high performance computing productivity.* International Journal of High Performance Computing Applications, 2004. **18**(4): p. 459-473.
25. Dell. *Dell HPC*. 2016 [cited 2016 March 2016]; Available from: http://www.dell.com/learn/us/en/25/business~solutions~whitepapers~en/documents~hpc-right-choice.pdf.
26. HP. *HP HPC*. 2016 [cited 2016 March 2016]; Available from: http://www8.hp.com/us/en/products/servers/scalable-systems/clusterplatform.html?jumpid=go/clusterplatforms&404m=rt404Mb.
27. Buyya, R. and S. Venugopal, *A gentle introduction to grid computing and technologies.* database, 2005. **2**: p. R3.
28. Buyya, R., et al., *Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility.* Future Generation computer systems, 2009. **25**(6): p. 599-616.
29. Kim, J.-S., et al. *Creating a robust desktop grid using peer-to-peer services*. in *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*. 2007. IEEE.
30. Sterling, T., D.J. Becker, and D.F. Savarese, *How to Build a Beowulf: A Guide to the Implementation and Application of PC Clusters (Scientific and Engineering Computation).* 1999.
31. Qin, X., et al., *Dynamic load balancing for I/O-intensive tasks on heterogeneous clusters*, in *High Performance Computing-HiPC 2003*. 2003, Springer. p. 300-309.
32. Shi, J., C. Meng, and L. Ma. *The Strategy of Distributed Load Balancing Based on Hybrid Scheduling*. in *Computational Sciences and Optimization (CSO), 2011 Fourth International Joint Conference on*. 2011. IEEE.
33. Liang, S., V. Holmes, and I. Kureshi. *Hybrid Computer Cluster with High Flexibility*. in *Cluster Computing Workshops (CLUSTER WORKSHOPS), 2012 IEEE International Conference on*. 2012. IEEE.
34. Assunção, M.D., et al., *Big Data computing and clouds: Trends and future directions.* Journal of Parallel and Distributed Computing, 2015. **79**: p. 3-15.
35. Corp, I. *Small Form Factor Computers*. [cited 2016 March 2016]; Available from: http://www.intel.co.uk/content/www/uk/en/nuc/overview.html.
36. Ltd, Z. *Small Form Factor Computers*. [cited 2016 March 2016]; Available from: http://www.zotac.com/en/z-zone/zbox-pico.html.
37. Cusick, J.J., et al., *Design, Construction, and Use of a Single Board Computer Beowulf Cluster: Application of the Small-Footprint, Low-Cost, InSignal 5420 Octa Board.* arXiv preprint arXiv:1501.00039, 2014.
38. Abrahamsson, P., et al. *Affordable and Energy-Efficient Cloud Computing Clusters: The Bolzano Raspberry Pi Cloud Cluster Experiment*. in *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*. 2013. IEEE.

39. Anderson, T.E., D.E. Culler, and D.A. Patterson, *A case for NOW (networks of workstations).* Micro, IEEE, 1995. **15**(1): p. 54-64.
40. Ridge, D., et al. *Beowulf: harnessing the power of parallelism in a pile-of-PCs*. in *Aerospace Conference, 1997. Proceedings., IEEE*. 1997. IEEE.
41. Brunner, R.K. and L.V. Kalé. *Adapting to load on workstation clusters*. in *Frontiers of Massively Parallel Computation, 1999. Frontiers' 99. The Seventh Symposium on the*. 1999. IEEE.
42. Milutinovic, V., et al., *Guide to DataFlow Supercomputing: Basic Concepts, Case Studies, and a Detailed Example*. 2015: Springer.
43. Cook, J.S. and N. Gupta, *History of Supercomputing and Supercomputer Centers.* Research and Applications in Global Supercomputing, 2015: p. 33.
44. Andrews, C., *The future of weather forecasting [Communications Met Office Supercomputer]*. Engineering & Technology, 2015. **10**(2): p. 65-67.
45. Wang, B., et al., *Modern Gyrokinetic Particle-In-Cell Simulation of Fusion Plasmas on Top Supercomputers.* arXiv preprint arXiv:1510.05546, 2015.
46. Tian, X., et al., *Latency critical big data computing in finance.* The Journal of Finance and Data Science, 2015. **1**(1): p. 33-41.
47. CRAY. *Supercomputing*. [cited 2016 March 2016]; Available from: http://www.cray.com/.
48. IBM. *Supercomputing*. [cited 2016 March 2016]; Available from: www.IBM.com.
49. Korpela, E., et al., *SETI@ HOME—massively distributed computing for SETI.* Computing in science & engineering, 2001. **3**(1): p. 78-83.
50. Anderson, D.P. *Boinc: A system for public-resource computing and storage*. in *Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on*. 2004. IEEE.
51. Yigitbasi, M.N., *Understanding and Improving the Performance Consistency of Distributed Computing Systems*. 2012: TU Delft, Delft University of Technology.
52. TOP500. *TOP500*. [cited 2016 March 2016]; World Top 500 Supercomputers]. Available from: http://www.top500.org/.
53. Kogge, P.M. and T.J. Dysart. *Using the TOP500 to trace and project technology and architecture trends*. in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*. 2011. ACM.
54. Booth, G.M. *Distributed information systems*. in *Proceedings of the June 7-10, 1976, national computer conference and exposition*. 1976. ACM.
55. Woodward, P.R., *Perspectives on supercomputing: Three decades of change.* Computer, 1996. **29**(10): p. 99-111.
56. Emma, P.G. and E. Kursun, *Is 3D chip technology the next growth engine for performance improvement?* IBM journal of research and development, 2008. **52**(6): p. 541-552.
57. Loh, G.H., Y. Xie, and B. Black, *Processor design in 3D die-stacking technologies.* Ieee Micro, 2007(3): p. 31-48.
58. Strohmaier, E. *20 Years Supercomputer Market Analysis*. in *International Supercomputer Conference*. 2005.
59. Coulter, S.K. and J.E. Martinez, *Introduction to InfiniBand.* 2015.
60. Boden, N.J., et al., *Myrinet: A gigabit-per-second local area network.* IEEE micro, 1995(1): p. 29-36.

61. Upadhyay, J., V. Varavithya, and P. Mohapatra. *Routing algorithms for torus networks*. in *Intl. Conf. On High Performance Computing*. 1995. Citeseer.
62. Barker, K.J., et al. *Entering the petaflop era: the architecture and performance of Roadrunner*. in *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*. 2008. IEEE Press.
63. Ia, V., *Supercomputing's exaflop target.* Communications of the ACM, 2011. **54**(8).
64. Sterling, T., et al. *An assessment of Beowulf-class computing for NASA requirements: initial findings from the first NASA workshop on Beowulf-class clustered computing*. in *Aerospace Conference, 1998 IEEE*. 1998. IEEE.
65. Feng, W.-c., M. Warren, and E. Weigle. *The bladed beowulf: A cost-effective alternative to traditional beowulfs*. in *Cluster Computing, 2002. Proceedings. 2002 IEEE International Conference on*. 2002. IEEE.
66. Adams, J.C. and T.H. Brom. *Microwulf: a beowulf cluster for every desk*. in *ACM SIGCSE Bulletin*. 2008. ACM.
67. Barczak, A.L., C.H. Messom, and M.J. Johnson, *Performance characteristics of a cost-effective medium-sized beowulf cluster supercomputer*, in *Computational Science—ICCS 2003*. 2003, Springer. p. 1050-1059.
68. Love, P., et al., *Parallel processing of radiotherapy Monte Carlo simulations on a remote Beowulf cluster*, in *The Use of Computers in Radiation Therapy*. 2000, Springer. p. 409-410.
69. Yang, C.-T., et al. *Using a Beowulf cluster for a remote sensing application*. in *Paper presented at the 22nd Asian Conference on Remote Sensing*. 2001.
70. Dmitruk, P., et al., *Scalable parallel FFT for spectral simulations on a Beowulf cluster.* Parallel Computing, 2001. **27**(14): p. 1921-1936.
71. Bennett III, F.H., et al. *Building a Parallel Computer System for $18, 000 that Performs a Half Peta-Flop per Day*. in *GECCO*. 1999.
72. Microsoft_Research. *Microsoft HPC Servers*. [cited 2016 March 16]; Available from: http://www.microsoft.com/hpc/en/us/product/cluster-computing.aspx.
73. Wilbertz, B., *GPGPUs in computational finance: Massive parallel computing for American style options.* Concurrency and Computation: Practice and Experience, 2012. **24**(8): p. 837-848.
74. Owens, J.D., et al., *GPU computing.* Proceedings of the IEEE, 2008. **96**(5): p. 879-899.
75. Sanders, J. and E. Kandrot, *CUDA by Example: An Introduction to General-Purpose GPU Programming, Portable Documents*. 2010: Addison-Wesley Professional.
76. Yang, X.-J., et al., *The TianHe-1A supercomputer: its hardware and software.* Journal of Computer Science and Technology, 2011. **26**(3): p. 344-351.
77. Chrysos, G., *Intel® Xeon Phi™ Coprocessor-the Architecture.* Intel Whitepaper, 2014.
78. Casavant, T.L. and J.G. Kuhl, *A taxonomy of scheduling in general-purpose distributed computing systems.* Software Engineering, IEEE Transactions on, 1988. **14**(2): p. 141-154.
79. Xu, C., *Load balancing in parallel computers: theory and practice*. 1997: Springer.
80. Choi, S., et al. *Volunteer availability based fault tolerant scheduling mechanism in desktop grid computing environment*. in *Network Computing and*

Applications, 2004.(NCA 2004). Proceedings. Third IEEE International Symposium on. 2004. IEEE.
81. Kim, C. and H. Kameda, *An algorithm for optimal static load balancing in distributed computer systems.* IEEE Transactions on Computers, 1992(3): p. 381-384.
82. Overeinder, B.J., et al., *A dynamic load balancing system for parallel cluster computing.* Future Generation Computer Systems, 1996. **12**(1): p. 101-115.
83. Yan, K.-Q., et al., *Towards a hybrid load balancing policy in grid computing system.* Expert Systems with Applications, 2009. **36**(10): p. 12054-12064.
84. Litzkow, M.J., M. Livny, and M.W. Mutka. *Condor-a hunter of idle workstations*. in *Distributed Computing Systems, 1988., 8th International Conference on*. 1988. IEEE.
85. McLaughlin, D., S. Sardesai, and P. Dasgupta. *Preemptive scheduling for distributed systems*. in *11th International Conference on Parallel and Distributed Computing Systems*. 1998.
86. Borkar, S. *Thousand core chips: a technology perspective*. in *Proceedings of the 44th annual Design Automation Conference*. 2007. ACM.
87. Keyes, R.W., *Fundamental limits of silicon technology.* Proceedings of the IEEE, 2001. **89**(3): p. 227-239.
88. Gepner, P. and M.F. Kowalik. *Multi-core processors: New way to achieve high system performance*. in *Parallel Computing in Electrical Engineering, 2006. PAR ELEC 2006. International Symposium on*. 2006. IEEE.
89. Kruger, J. and R. Westermann. *Acceleration techniques for GPU-based volume rendering*. in *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*. 2003. IEEE Computer Society.
90. Seiler, L., et al. *Larrabee: a many-core x86 architecture for visual computing*. in *ACM Transactions on Graphics (TOG)*. 2008. ACM.
91. Mattson, T.G., R. Van der Wijngaart, and M. Frumkin. *Programming the Intel 80-core network-on-a-chip terascale processor*. in *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*. 2008. IEEE Press.
92. Howard, J., et al., *A 48-core IA-32 processor in 45 nm CMOS using on-die message-passing and DVFS for performance and power scaling.* Solid-State Circuits, IEEE Journal of, 2011. **46**(1): p. 173-183.
93. Wolf, W., A.A. Jerraya, and G. Martin, *Multiprocessor system-on-chip (MPSoC) technology.* Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 2008. **27**(10): p. 1701-1713.
94. *A network on chip architecture and design methodology*. in *VLSI, 2002. Proceedings. IEEE Computer Society Annual Symposium on*. 2002. IEEE.
95. Feng, W.-c., et al. *Optimizing 10-Gigabit Ethernet for networks of workstations, clusters, and grids: A case study*. in *Proceedings of the 2003 ACM/IEEE conference on Supercomputing*. 2003. ACM.
96. Liu, J., et al. *Performance comparison of MPI implementations over InfiniBand, Myrinet and Quadrics*. in *Supercomputing, 2003 ACM/IEEE Conference*. 2003. IEEE.
97. Alverson, B. *Cray high speed networking*. in *Proceedings of the 20th Annual Symposium on High-Performance Interconnects (HOTI)*. 2012.
98. Pfister, G.F., *An introduction to the infiniband architecture.* High Performance Mass Storage and Parallel I/O, 2001. **42**: p. 617-632.
99. InfiniBand. *InfiniBand Trade Association*. 2016]; March 2016]. Available from: http://www.infinibandta.org/.

100. Gutiérrez-Castrejón, R., L. Schares, and M. Duelk, *SOA nonlinearities in 4× 25 Gb/s WDM pre-amplified system for 100-Gb/s Ethernet.* Optical and quantum electronics, 2008. **40**(13): p. 1005-1019.
101. Zenios, S.A., *High-performance computing in finance: The last 10 years and the next.* Parallel Computing, 1999. **25**(13): p. 2149-2175.
102. Menkveld, A.J., *High frequency trading and the new market makers.* Journal of Financial Markets, 2013. **16**(4): p. 712-740.
103. Wood, R., J. Upson, and T.H. McInish, *The flash crash: Trading aggressiveness, liquidity supply, and the impact of intermarket sweep orders.* Liquidity Supply, and the Impact of Intermarket Sweep Orders (July 2013), 2013.
104. Zhang, N., et al. *Cpu-gpu hybrid parallel binomial american option pricing*. in *Proceedings of the International Multiconference of Engineers and Computer Scientists*. 2012.
105. Suo, S., et al. *GPU option pricing*. in *Proceedings of the 8th Workshop on High Performance Computational Finance*. 2015. ACM.
106. V., A.Y. Grama, and N.R. Vempaty, *Scalable load balancing techniques for parallel computers.* Journal of Parallel and Distributed Computing, 1994. **22**(1): p. 60-79.
107. Wilkinson, B. and M. Allen, *Parallel programming*. Vol. 999. 1999: Prentice hall Upper Saddle River, NJ.
108. Kshemkalyani, A.D. and M. Singhal, *Distributed computing: principles, algorithms, and systems*. 2011: Cambridge University Press.
109. Hwang, K., J. Dongarra, and G.C. Fox, *Distributed and cloud computing: from parallel processing to the internet of things*. 2013: Morgan Kaufmann.
110. Lee, S.-Y. and C.-H. Cho. *Load balancing for minimizing execution time of a target job on a network of heterogeneous workstations*. in *Job Scheduling Strategies for Parallel Processing*. 2000. Springer.
111. Shirazi, B., A.R. Hurson, and K. Kavi, *Introduction to scheduling and load balancing.* Scheduling and Load Balancing in Parallel and Distributed Systems, 1995.
112. Leland, R. and B. Hendrickson. *An empirical study of static load balancing algorithms*. in *Scalable High-Performance Comput. Conf*. 1994.
113. Schlagenhaft, R., et al. *Dynamic load balancing of a multi-cluster simulator on a network of workstations*. in *ACM SIGSIM Simulation Digest*. 1995. IEEE Computer Society.
114. Hsu, T.-S., et al., *Task allocation on a network of processors.* Computers, IEEE Transactions on, 2000. **49**(12): p. 1339-1353.
115. Sinnen, O., *Task scheduling for parallel systems*. Vol. 60. 2007: John Wiley & Sons.
116. Tørresen, J. and K.A. Vinger. *High Performance Computing by Context Switching Reconfigurable Logic*. in *Esm*. 2002.

Appendix A

A.1 Control Systems for Load Balancing

To maintain the stable load balancing mechanism in the distributed processing system, three types of control methods were investigated. These methods are simplified models of complex control systems that are widely used within various industries. A number of load balancing algorithms are used for static and dynamic load balancing within numerous distributed processing systems, and these methods are highly suited for particular types of distributed systems. However, the Northwest distributed processing system, which is designed as a bespoke type system, requires bespoke type load balancing algorithms and control mechanisms, and has to be simple to implement and easy to manage. In addition, it must be robust during the system's operation. The control system is based on a discrete sampling method to capture various data during system execution; hence, the sampling frequency is also crucial for fine-tuning the control mechanism. For fine-grained processing, this method is not suitable, and for Northwest's system that utilises coarse-grained batch processing type task calculations, the sampling frequency can be lower depending on the application used.

A.1.1 Linear Model-Based Control

The linear control method used for static load balancing is similar to the open-loop controller and uses existing data, current state, and the defined model of the system to compute its input parameters into a system using only the current state. The characteristic of the open-loop controller is that it does not use feedback to determine whether its output has achieved the desired target of the input. This means that the open-loop system does not observe the output of the processes that it is controlling. Hence, the linear open-loop controller method is most suitable for static load balancing and simpler to implement within the known calculation cluster systems. Meanwhile, various parameters are collected during the system's operation, and these parameters are used to refine the static algorithms employed to allocate tasks to each

processing unit. Therefore, the static algorithms will continuously adapt to perform better for the next process execution within the system concerned. However, during the process execution, the model remains static, and all the input parameters remain the same. Hence, the linear open-loop controller method is most suitable for implementing within the known systems. Figure A.1 and Figure A.2 show the liner model-based control system schematic and implementation diagrams.

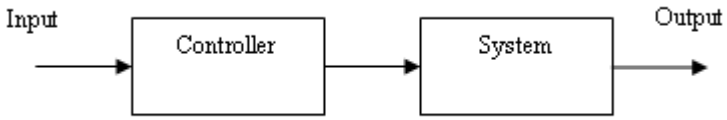


Figure A.1: Linear control system schema

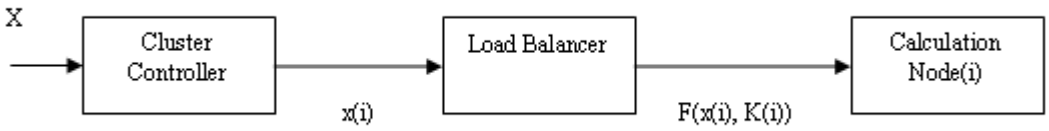


Figure A.2: Linear control system implementation

A.1.2 Self-Tuning Model-Based Adaptive Control

Self-tuning model-based adaptive control is used for dynamic load balancing, and it is similar to the feedback control system. It uses the defined models and current state of the system, as well as the feedback data from the output parameters and continuously changes the input parameters during the process execution. A characteristic of the closed-loop controller is that it continuously changes the input parameters using various feedback parameters and model-based parameters to determine whether its output has achieved the desired target of the input. Hence, the system has to observe the output of the processes that it is controlling, and this makes the system slower than static load balancing. Similar to the static load balancing system, the dynamic load balancing system also collects data continuously, and the collected data are used by the distributed processing controller to fine-tune the load balancing algorithms. Figure

A.3 and Figure A.4 show the Self-Tuning Model-Based Adaptive Control system schematic and implementation diagrams.

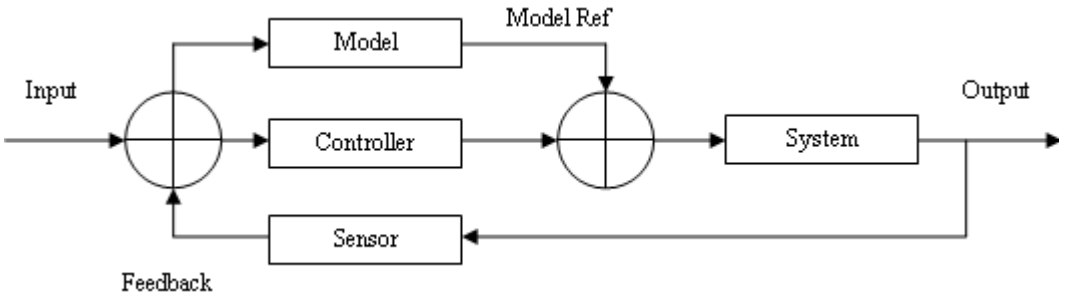


Figure A.3: Self-Tuning Model-Based Adaptive Control system schema

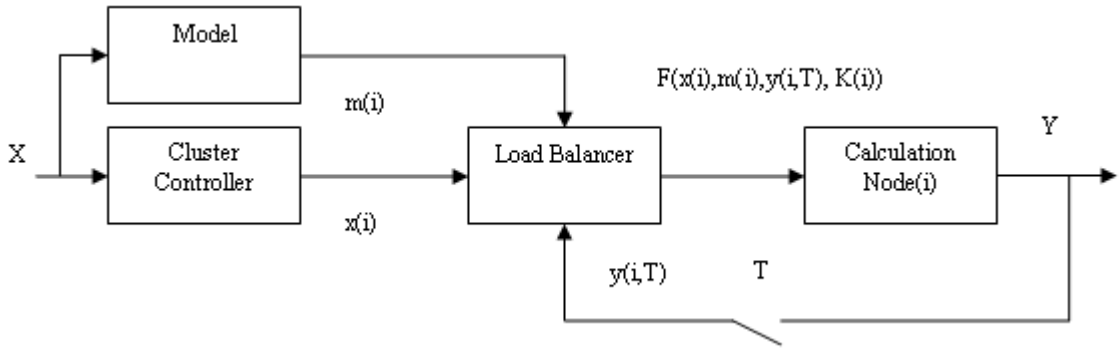


Figure A.4: Self-Tuning Model-Based Adaptive Control implementation

A.1.3 Fuzzy Logic Control

In comparison with conventional control techniques, fuzzy logic is best used for complex processes that can be controlled by a rule-based controller without prior knowledge of their underlying dynamics. The fundamental idea behind the fuzzy logic control technique is to incorporate the past performance of a system. The design of the controller involves controlling a process the input/output relationship of which is described by collection of fuzzy control rules involving weighted mean value-based variables rather than complicated dynamic models. The utilisation of weighted mean variables, fuzzy control rules, and approximate reasoning provides a means to incorporate logical reasoning in designing the controller. The controller is strongly based on the concepts of fuzzy sets, weighted mean variables, and approximate

reasoning. This type of rule-based fuzzy logic algorithm is used in Northwest's distributed processing system as a part of dynamic load balancing algorithms. Figure A.5 and Figure A.6 show the Fuzzy Logic-based control system schematic and implementation diagrams.

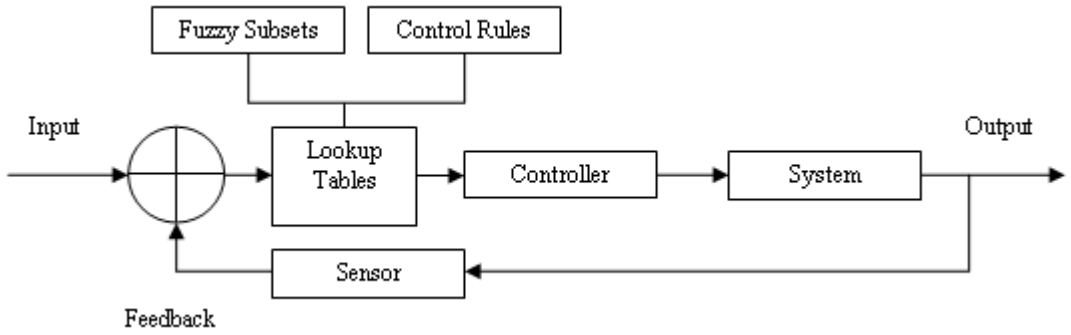


Figure A.5: Fuzzy logic-based control system schema

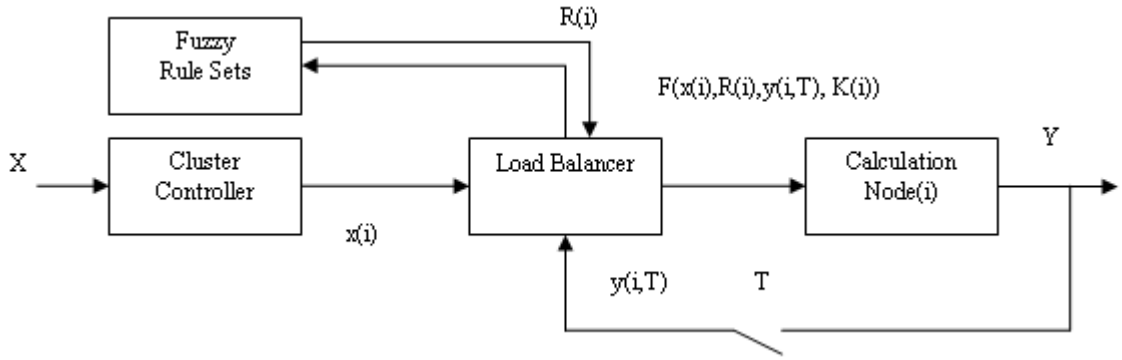


Figure A.6: Fuzzy logic-based control implementation

where

x(i)	Input parameter for node i
K(i)	Load balancing parameter for node i
y(i,T)	Output status for node i at sampling time T
m(i)	Model parameter for node i
R(i)	Control rule set for node i
T	Sampling period

A.2 Termination Detection

Termination detection is a component of task allocation and load balancing mechanisms, and is used for detecting when a computation is completed. Termination detection becomes a significant issue when the computation is distributed and various algorithms have to determine whether a distributed computation being performed in a system has terminated. The distributed computation being performed is known as the basic computation, and the inter-process messages used for implementing it are known as the basic messages. For termination detection, an additional computation known as the control computation is superimposed on the basic computation, and the messages used to implement the control computation are known as the control messages. A process is in the active state if it is currently performing the basic computation or in passive otherwise. Hence, an active process can send or receive basic messages, create other processes, or become passive. Meanwhile, a passive process can only receive basic messages, and become active-state. Therefore, the distributed computation is said to have terminated when all of its live processes are in the passive state. This is called the distributed termination condition (DTC), and the control computation performed by the processes for detecting the termination is called the termination detection algorithm. The implementation of termination detection algorithms depends on the types of distributed processing systems, such as:

- Tightly coupled distributed processing systems.
- Loosely coupled distributed processing systems.
- Centrally controlled distributed processing systems.
- Model-based distributed processing systems.
- Mixed-mode based distributed processing systems.

Depending on what type of distributed processing system is implemented, the complexity of the termination detection process varies accordingly. The Northwest distributed processing system mainly consists of a loosely coupled system, and the termination detection is managed by the distributed processing management controller.

A.2.1 Process Completion Time Limits

Each process submitted to the calculation node has a calculation completion time limit parameter, and the parameter is an estimated value that is calculated based on the following factors:

- Past calculation-time based on the same calculation node or another that has similar hardware and software configurations.
- CPU and memory use parameters of the calculation node that are based on current data and saved historical data.
- Application parameters mainly depend on each application used within the distributed processing system.

Hence, the process completion time limit is used by the distributed processing management controller to make decisions based on these parameters regarding whether the process terminated and whether it should be assigned to another available calculation node.

A.2.2 Termination Rules

The process termination is determined by the following three conditions: Process completed, Process calculation time lapsed, and Calculation node is not responding.

Process Completed: This condition is met when the allocated calculation process completed by the calculation node and the calculation node controller send a message to the distributed processing controller. Once the ‘process completed’ condition message has been received, the distributed processing controller allocates another process to the calculation node, and so on. This is a seamless process assuming no disruption to the calculation node and that the calculation node performs as expected by the static control algorithms.

Process Time Lapsed: This condition only happens when the calculation process exceeds the predefined time limit for the calculation node concerned. When this condition is met, the calculation node controller sends a message to the distributed processing controller to notify the status of the calculation process. At this point, the distributed processing controller activates the dynamic load balancing algorithms to relocate the processes to another calculation node or to activate the standby node using the context switch method. This will cause time delays in the overall calculation due to various processes involved in transferring calculations from one calculation node to another calculation node dynamically.

Node Error: This condition happens when no response from the calculation node concerned, and this is mainly due to hardware or software error. In this case, no communication between the distributed processing controller and the calculation node controller, and the calculation node is permanently excluded from the calculation cluster by the distributed processing controller. In this case, the distributed processing controller activates the dynamic load balancing algorithms to reassign the processes to another active or standby calculation node using the context switch method. Figure A.7 shows task messages and times related to termination processes. Equations (A.1) and (A.2) formulate the calculation times for process terminations.

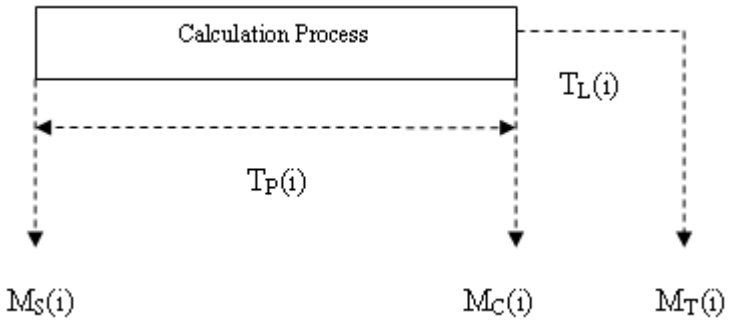


Figure A.7: Task termination process time limits and associated messages

$$T_P(i) = T_C(i) + T_I(i) + T_F(i) \quad \text{(A.1)}$$

$$T_L(i) = K(i) \times T_P(i) \quad \text{(A.2)}$$

where

$T_P(i)$	Allocated total process time for node i
$T_C(i)$	Total calculation time taken by node i
$T_I(i)$	Initialisation time for node i
$T_F(i)$	Finalisation time for node i
$T_L(i)$	Lapsed time for node i
$M_S(i)$	Process start message for node i
$M_C(i)$	Process complete message for node i
$M_T(i)$	Process terminate message for node i

The value of T_P depends on the applications and types of batch processes used within the calculation cluster. The calculation node controller sends messages to the distributed processing management controller to notify the status of the calculation process. Each calculation process has different T_P values depending on various factors such as the application used, calculation node's hardware parameters, and historical data. The estimated T_P values are maintained in the distributed processing management controller SQL database, and T_P is continuously modified by adaptation algorithms to reduce the T_P using rule-based fuzzy logic techniques. Figure A.8 shows the task termination and relocation process. Here, *T1* and *T2* are arbitrary values for time delays, and the values depend on each batch process.

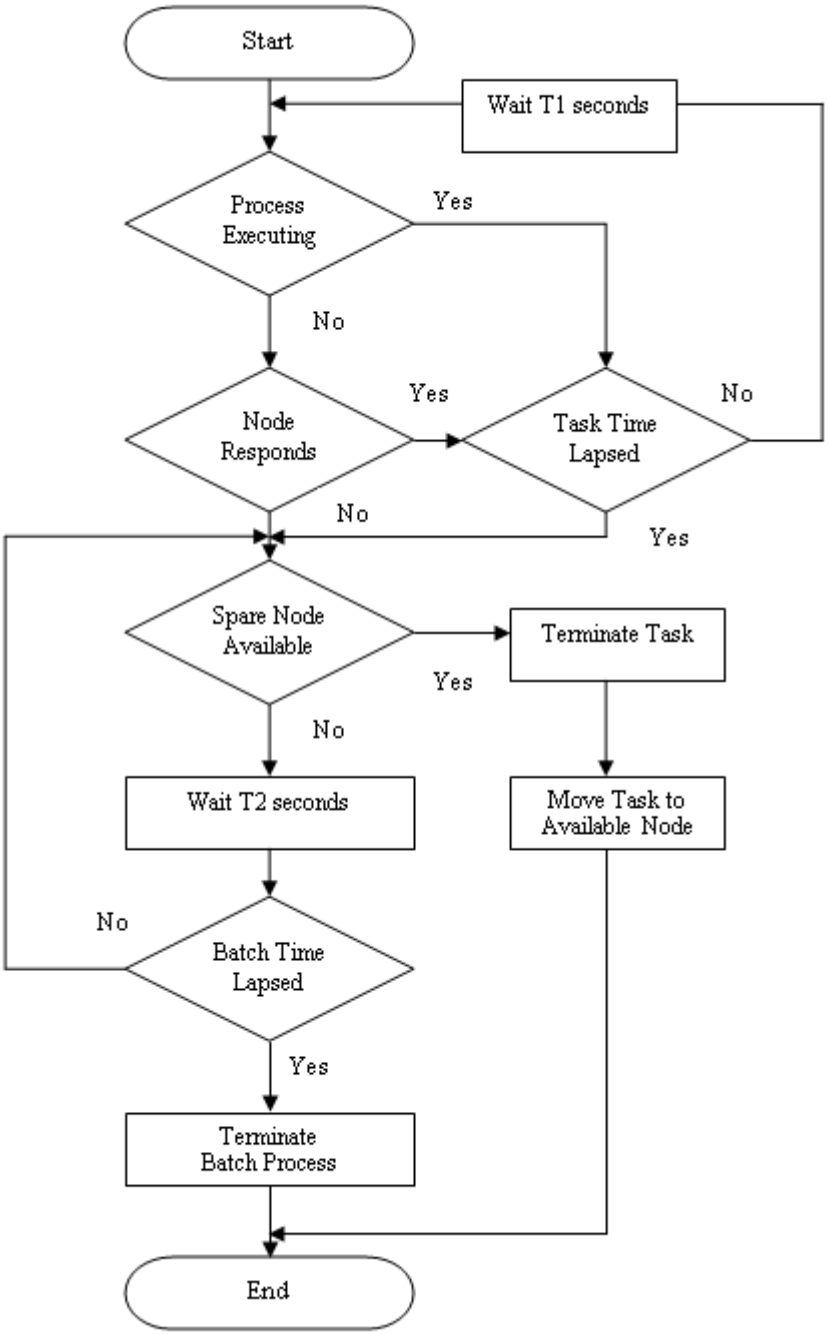


Figure A.8: Task termination and relocation process

A.3 Distributed Processing Time Delays

Various time delays within the system will affect the calculation speed and increase the calculation time. These time days have to be reduced to a minimum to achieve a better performance from the system. The network latency-related time delays are also important in the distributed processing systems; however, for the system that is used within Northwest's network, the network-related time delays are small compared with application- and data-related time delays. However, in the Northwest distributed processing system, time delays that are part of the communication and synchronisations are small compared with the overall computing time. This is because the applications used for distributed calculations are based on the coarse-grained method, and the calculation time for each distributed segment is considerably higher than the communication and synchronisation time, which has been observed to be in the range of 1,000 to 1 ratio. Hence, communication and synchronisation time delays are not considered in any analysis and only calculation times are used, that is, calculation time that includes other types of time delays. The current applications used in the distributed processing system are highly compute-intensive and this is one of the reasons that the communication and synchronisation times become comparatively small. However, if the company decided to use different types of applications in the future, then communication and synchronisation time delays have to be included in the distributed processing calculation analysis.

A.4 MS-Excel Usage in the Financial Industry

Microsoft Office MS-Excel is widely used in the financial industry and especially by the small organisations and hedge fund management industries. For a small company like Northwest, MS-Excel has become the core component of the applications within the business. The primary reason for the success of using MS-Excel applications is related to its many advantages compared with other similar products. It has been a widely adopted spreadsheet application, and it has replaced Lotus 1-2-3 as the industry standard for spreadsheets. MS-Excel is a valuable tool for portfolio managers, traders, and analysts. Various management reports and risk management tools can be created and executed in MS-Excel with simple implementations. MS-Excel has created useful functionalities, especially for the finance and accounting industries, and some companies fully manage their entire range of business operations just using MS-Excel applications. Accordingly, it is a useful tool for financial use in small to medium organisations. The main drivers for using MS-Excel in small organisations that it is relatively cost effective, easy to use, easy to learn, and provides most of the solutions to the various needs of an organisation. However, MS-Excel has limitations in terms of performance, security, and consistency, and these all depend on the size, needs, and nature of the business. However, MS-Excel is used by 90% of financial applications, and the reason for this is that it provides reliable financial application software design.

A.4.1 Advantages

Financial usage: MS-Excel is a customised financial application. All necessary financial formulas are installed with it, and specialised financial functions are available those are provided by Microsoft or through various third-party vendors.

High usage rate: The high usage rate, which is around 90% in the industry, and all external devices being compatible with Windows, and consequently with MS-Excel, have resulted in growing demand. Using MS-Excel increases the chances of

accomplishing effective understanding between users or organisations, making communication easier.

Prototyping Tool: MS-Excel can be used for developing applications within shorter time scales using Rapid Application Development (RAD) methods, and can be improved and adapted according to the business requirements using Dynamic System Development Methods (DSDM).

Highly Integrated: MS-Excel highly integrated with other Microsoft applications. Hence, the data manipulation and data flow are easier to perform.

Upgradability: MS-Excel is upgraded regularly, which improves it and makes it more consistent with the needs of organisations.

Cost-Effective: Off-the-shelf software is relatively cheaper for smaller organisations.

Easy to Use: MS-Excel's features and structure make the usage experience user-friendly, despite a steep learning curve for first-time users.

Security: MS-Excel can be password-protected with various levels of access rights, making it more secure for organisations.

Portability: MS-Excel is a portable application; its documents can be sent through e-mail and can be synchronised with other applications and matched with different devices.

A.4.2 Disadvantages

Viruses: The greatest risk with MS-Excel is the possibility of spreading viruses that can be costly for organisations.

Possible Data Loss: Data could be lost when the application is broken down into many files. To avoid data inconsistency and data loss, the data has to be regularly managed.

Non-scalability: MS-Excel is a file-based application, and it tends to bloat and corrupt when it becomes larger with large amounts of data.

Security: Built-in security features are weak and limited data and code security is available.

A.5 Distributed Processing Using MS-Excel Applications

Since MS-Excel version 2007, MS-Excel has supported multithreading workbook calculation functions that act as parallel thread calculations. MS-Excel uses multithreaded recalculation (MTR) of worksheets; this can be configured to use up to 1,024 concurrent threads when recalculating, regardless of the number of CPUs or CPU-cores in the computer. If the computer has multiple CPU or CPU-cores, the operating system takes responsibility for allocating the threads to the processors in the most efficient way. An operating system overhead associated with each thread, hence a trade-off between number of threads and operating system resources. However, parallel thread calculation of MS-Excel workbooks is not well integrated with the MS-Office system, and it is not a seamless process, but the possibility to be utilised still remains. For simple applications that use built-in MS-Excel functions, it is possible to achieve better performance in a single computer that has a multi-core CPU. The MS-Excel application itself is not capable of executing tasks in parallel. Hence, external applications and frameworks are required to implement distributed processing using MS-Excel applications. Typically, these external applications are computational grids, like Microsoft HPC Server, Platform Symphony, or similar platforms.

A.5.1 Microsoft HPC Server

Several solutions are available from Microsoft to run distributed MS-Excel on HPC, and has no generic framework, because every user should implement his or her own user-defined functions (UDF) using a preferred language. The custom UDFs have many advantages, including effective parallelisation, using API or HPC server directly, and programming environments such as .NET or C++ instead MS-Excel's VBA. The second approach, which allows computing with MS-Excel workbooks on the cluster, is represented by HPC MS-Excel services. To make this solution work, SharePoint MS-Excel services must be installed on every compute-node on the cluster. This solution uses an MS-Excel add-in to specify job information, input data, and output data mappings. HPC MS-Excel services support only lists as input values for parallelisation and lists for output values. Documentation is extensive, and sample models are described well. An interesting feature of the solution is that it supports submitting models from the SharePoint portal. However, implementing these types of solutions for the company's distributed processing requires various changes to existing systems and necessitates the purchase of certain hardware and software products. Hence, this approach is not a suitable solution for the company given its current requirements. Figure A.9 shows the Microsoft HPC Cluster configuration.

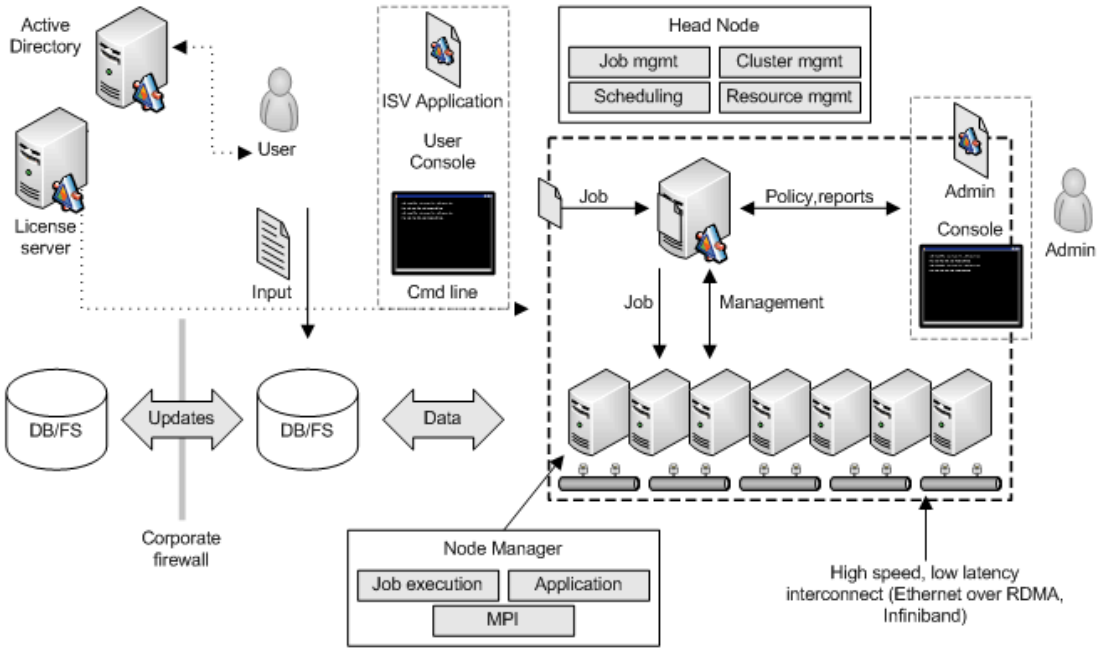


Figure A.9: Microsoft HPC Cluster configuration (Source: Microsoft)

A.5.2 Platform Symphony

This platform also supports both approaches using UDFs, which implement actual parallelisation logic, and computing MS-Excel workbooks on compute-nodes. No generic-purpose framework for this, and everything should be customised for MS-Excel models. Useful for heavy MS-Excel calculations and grid vendor support available to distributed MS-Excel to some extent. Several approaches available to run MS-Excel workbooks on a grid, from using MS-Excel just as UI and accessing a grid using an API from UDFs, to calculating actual workbooks with VBA code on compute nodes. However, this approach also require extra work on IT hardware and software to implement this method; hence, this approach not a suitable solution for the company given its current requirements.

A.5.3 Bespoke Design

Another approach is to execute MS-Excel worksheets themselves on the specifically designed cluster. In this case, a workbook contains both UI and computation logic in cell formula or VBA as UDF. Almost all solutions support distributing a workbook by data (domain decomposition). The framework is to split these values into equal parts, copy MS-Excel documents with partial inputs to a cluster, execute documents on compute nodes on a cluster, and return the results to a main document that had initiated the computation. The computation logics can be implemented in the MS-Excel workbook itself with formulas or VBA functions. This approach is a general-purpose framework and can be used in certain application scenarios.

A.5.4 Northwest Implementation of MS-Excel

No specific approaches are available to utilise the MS-Excel applications as part of distributed processing systems. Hence, a bespoke method has been used to implement a distributed processing system for the company's applications that employs existing MS-Excel built-in functionalities. MS-Excel has the following functionalities that are useful for implementing distributed processing:

- Grid type of data structure can easily be portioned vertically and horizontally.
- Able to use user-defined functions (UDF) using VBA.
- Able to use various COM add-ins such as DLL, Active-X EXE.
- Able to create mutually exclusive MS-Excel instances using a single MS-Excel license.

By using these functionalities, it is possible to build a distributed processing cluster that can use MS-Excel applications as MS-Excel services to improve the calculation time for compute-intensive applications. This approach is discussed in detail in Chapter 4.

Appendix B

B.1 Cluster Groups

The cluster grouping is included in the cluster management controller's software for selecting different cluster groups for particular batch process calculations. Calculation nodes physically and logically are separated for grouping as separate calculation clusters, and these clusters are managed by the management controller software. Depending on which cluster is selected for the given calculation, all of the available nodes within the selected cluster will be used by the management controller to allocate tasks. Table B.1 lists the calculation cluster groups.

Table B.1: Calculation cluster groups

Cluster Name	Description
NUC_Grid	Workgroup NUC PC cluster
PC_Grid	Workgroup PC cluster
WS_Office	Office network user workstations
SRV_Office	Office network servers
SRV_DRSite	Disaster recovery site network servers
VWS_DRSite	Disaster recovery site virtual workstations
Cluster_A	NUC_Grid + WS_Grid
Cluster_B	WS_Office + SRV_Office
Cluster_C	UC_Grid + WS_Grid + WS_Office
Cluster_D	UC_Grid + WS_Grid + WS_Office + SRV_Office
Cluster_E	VWS_DR-Site
Cluster_F	VWS_DR-Site + SRV_DR-Site

B.2 Distributed Process Controller Interface

Figures B.1 to B.6 show the distributed processing management controller's user interface.

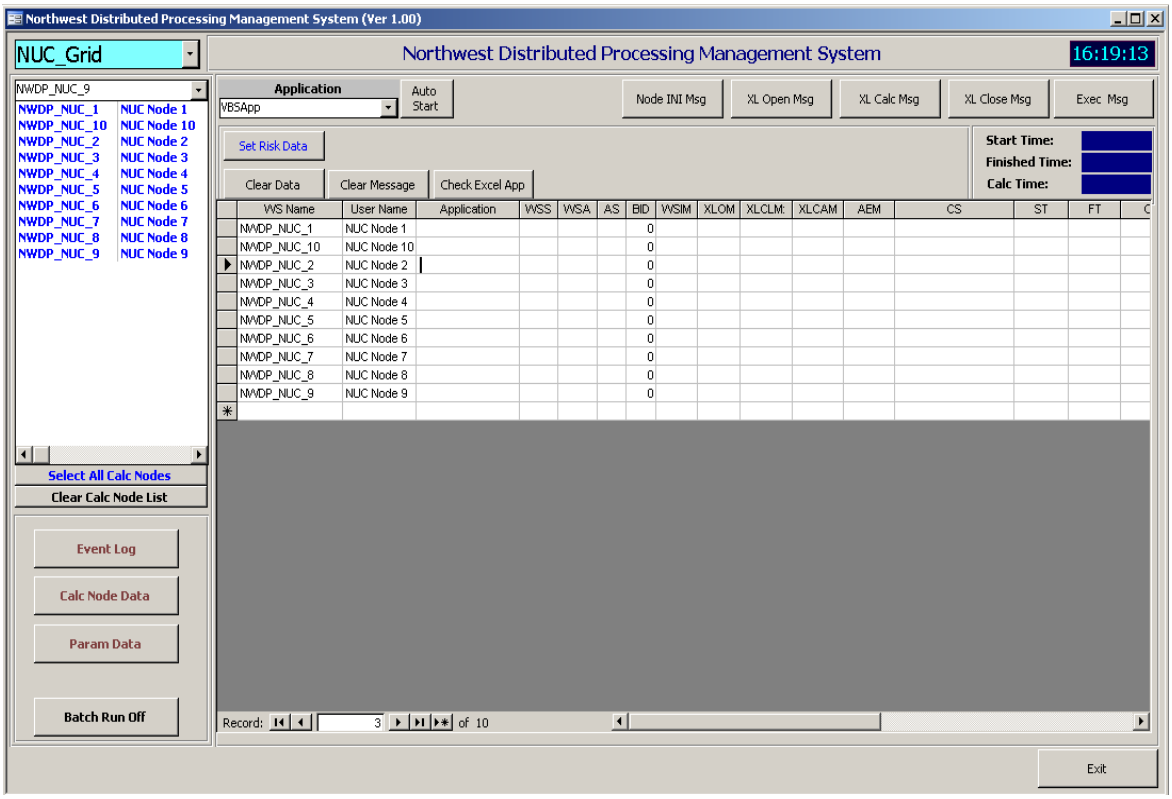


Figure B.1: Main control panel

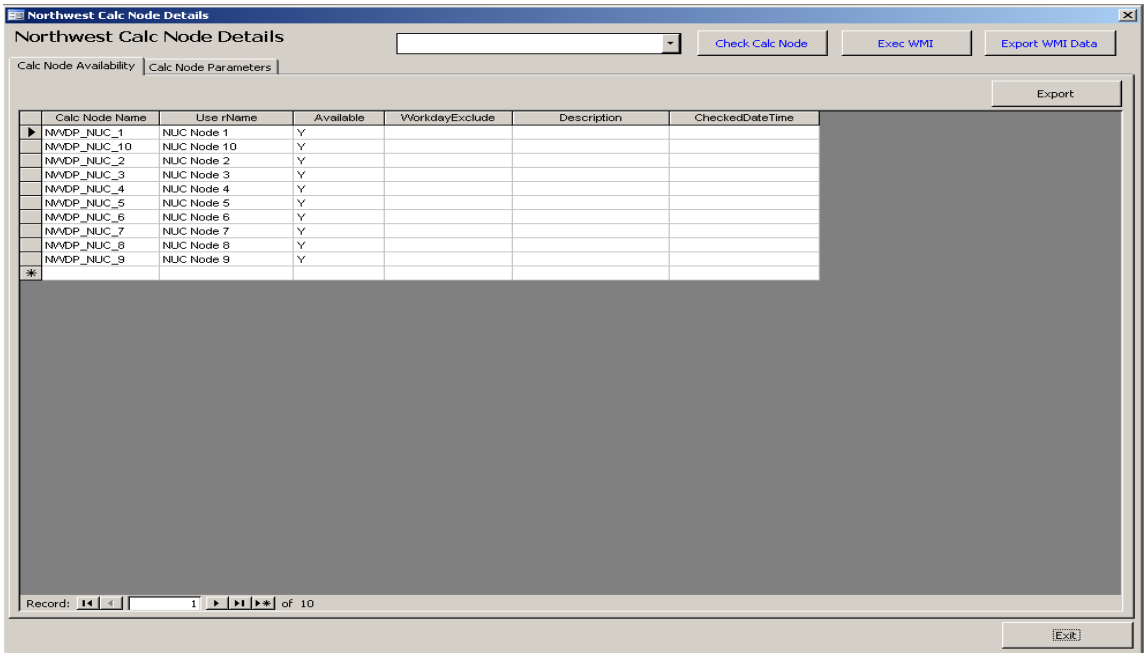


Figure B.2: Calculation node's parameter form

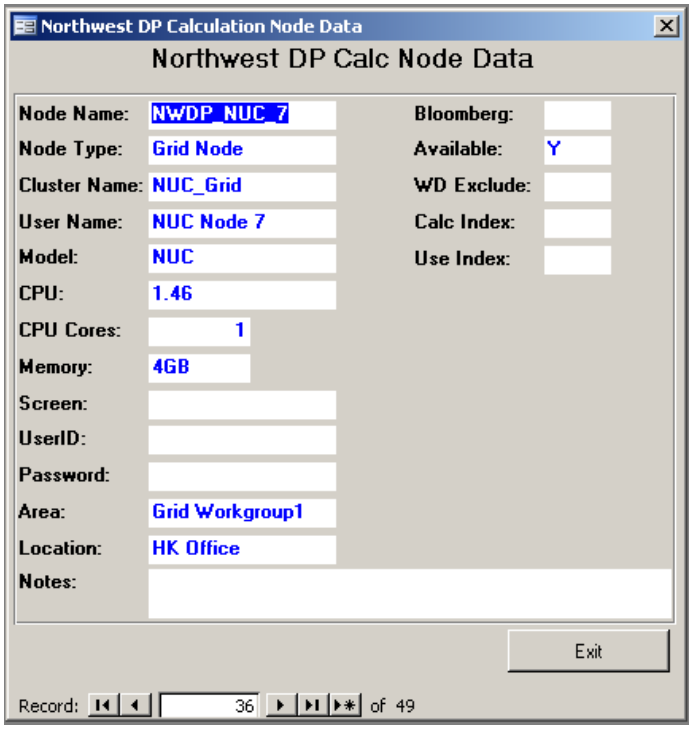


Figure B.3: Each calculation node's parameter form

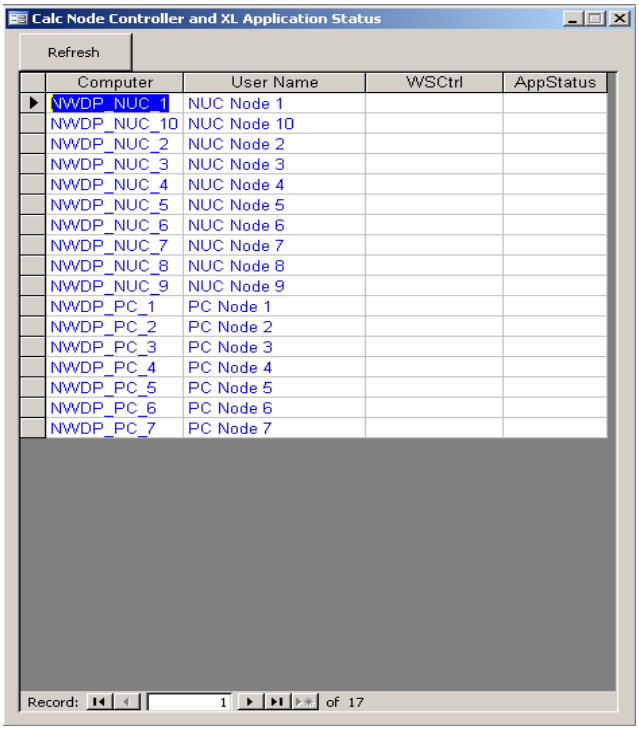


Figure B.4: Calculation node controller status monitoring form

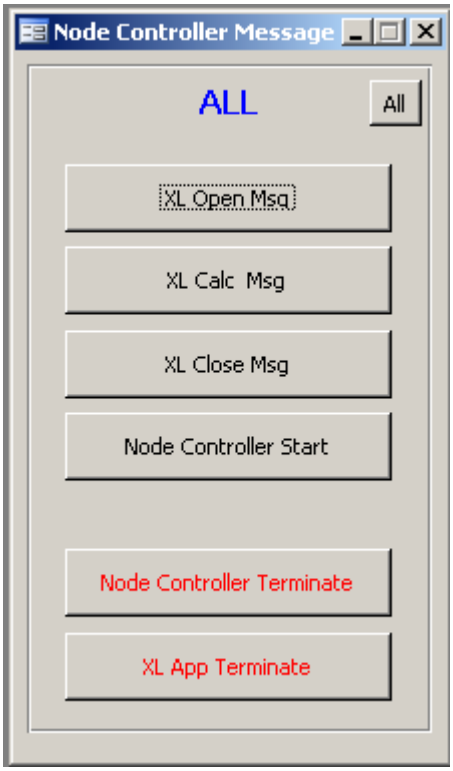


Figure B.5: Calculation node controller message processing panel

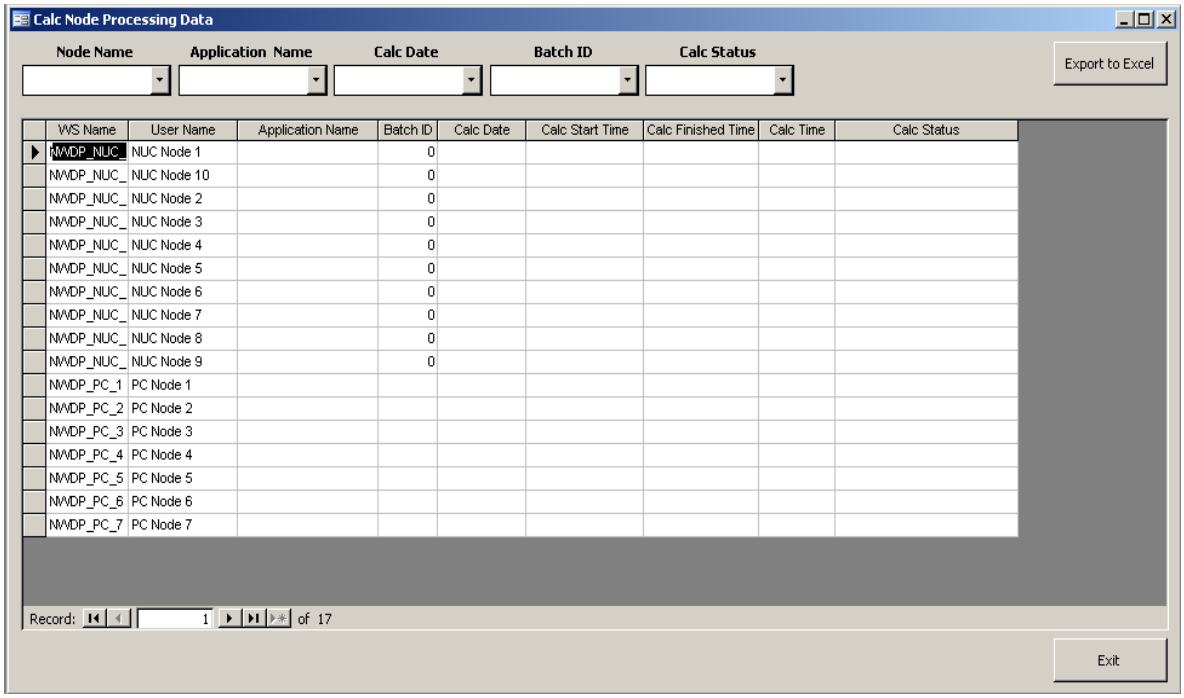
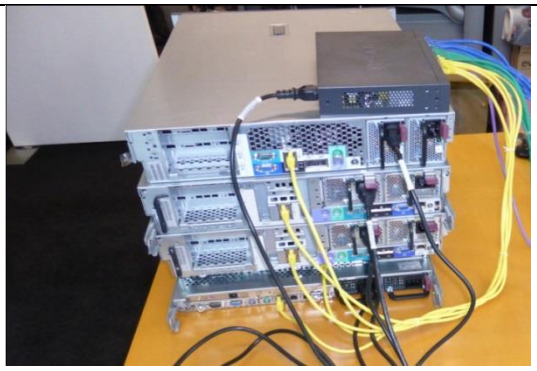


Figure B.6: Calculation node process monitoring form

Appendix C

This section presents the pictures of the server clusters, PC cluster, and NUC cluster configurations.

(a) Front view



(b) Back view

(c) Side view

(d) Blade server cluster

Figure C.1: Server Clusters

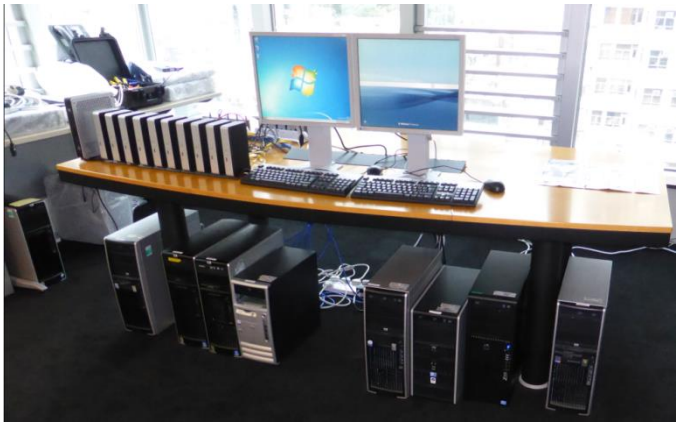
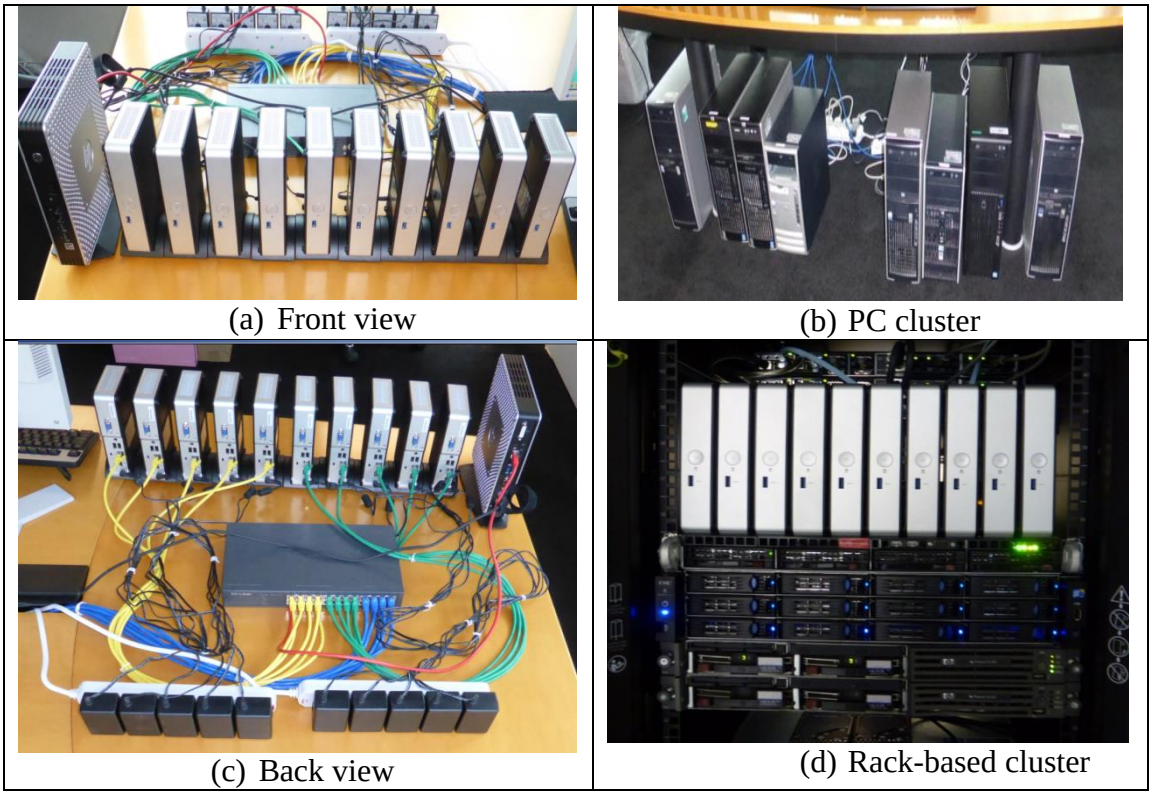


Figure C.2: P2P workgroup-based PC and NUC clusters



(a) Front view

(b) PC cluster

(c) Back view

(d) Rack-based cluster

Figure C.3: PC and NUC clusters