

**A comparison between a Computational Grid and a High-end Multicore Server
in an academic environment.**

David Tatenda Risinamhodzi

(STUDENT NO. 25781146)

Dissertation submitted in fulfillment of the requirement for the degree

MSc. Computer Science

in the

SCHOOL OF INFORMATION TECHNOLOGY

in the

FACULTY OF ECONOMIC SCIENCES & IT

at the

NORTH-WEST UNIVERSITY (VAAL TRIANGLE CAMPUS)

Promotor: Mr P Jooste

Co-Promoters: Prof P Pretorious & Dr B Becker

VANDERBIJLPARK

2017

It all starts here TM



Acknowledgements

Firstly I would like to thank God almighty for seeing me through this journey. A special thanks is extended to my promoter and co-promoter, Mr Petri Jooste and Prof Philip Pretorius for their scholarship, guidance, leadership, dedication and commitment in making this study a success. Through their mentorship, I have emerged from this study as a prospective academic. My gratitude also extends to Bruce Becker who came through for me when I needed him most, thank you for the collaboration on this research and for granting me access to resources on the South African national grid. Lastly I would like to thank my family and friends for supporting me through it all, even when I seemed to lose hope. Special mention goes to my dad, Samuel, my surviving siblings Tatenda, Davison, Beauty, Miriam and Nyasha and my lovely nieces Larissah and Lateisha as well as my friend Ntombi. To God be the glory.

Dedication

This study is dedicated to Aleen Risinamhodzi and Rosemary Zimani, my mother and sister who died in a tragic road traffic accident on the 20th of May 2016. Having carried me through this process and encouraging me to soldier on regardless of the challenges, may their souls rest in eternal peace.

Abstract

This study compares a computational grid to a high-end multicore server in terms of pure performance and system management so as to establish the choices and trade-offs of using either of the two systems to provide academic researchers with high performance processing capacity whilst curbing underutilization of computing resources. We conducted an experiment by adapting a compute-intensive applications for processing on both systems and measured the job completion times of the application when executed on either of the systems. We recorded job completion times for every task ran on the two systems, these are used in our analysis of the results. The findings from this research suggest that the dedicated multicore server performs better than the grid in a case of pure performance and that the grid could be an alternative to the multicore server as it offers more benefits and computing resources than the multicore server. The findings offer insight to researchers in need of high performance processing and assists them in determining the best system to use for a specific scenario.

Contents

Acknowledgements	i
Dedication	iii
Abstract	iv
Contents	v
List of figures	viii
List of tables	ix
1 Introduction	1
1.1 Background	1
1.2 Problem statement	2
1.3 Specific focus of this study	3
1.3.1 The nature of applications used in this study	3
1.3.2 South African National Grid	4
1.3.3 Difference between this study and other comparisons	5
1.4 Study objectives	6
1.5 Research methodology	6
1.6 Layout of chapters	6
1.7 Summary	7
2 Literature review	8
2.1 Introduction	8
2.2 Towards using multiple processors	8
2.2.1 A brief history of microprocessors	8
2.2.2 Parallel computing	13
2.2.3 Peer-to-peer computing	13
2.2.4 Cluster computing	13
2.2.5 Grid and cloud computing	14
2.2.6 Origin of grid computing	16

2.2.7	Overview and architecture	18
2.3	Middleware used in grids	30
2.4	Performance on the grid.....	33
2.5	Classification of grids	34
2.6	Campus grids	35
2.7	Summary.....	36
3	Research methodology.....	37
3.1	Introduction	37
3.2	Research.....	37
3.2.1	Research paradigms	39
3.2.2	Qualitative and quantitative research	41
3.2.3	Mixed methods research	42
3.3	Research methodology	43
3.3.1	Research design.....	44
3.3.2	Research methods of this study	47
3.3.3	Systems being compared	49
3.3.4	Data analysis	50
3.3.5	Comparing computational grid versus multicore server	50
3.3.6	Determination of the choices and trade-offs of using the two systems ..	50
3.3.7	Finding and recommendations	51
3.3.8	Summary	51
4	Results and analysis.....	52
4.1	Introduction	52
4.2	Experiments	52
4.2.1	Support Vector Machine (SVM) application	52
4.2.2	SVM Data analysis	53
4.2.3	Overall system performance (real-life scenario)	53

4.2.4	Effect of the data staging time	57
4.2.5	Overall performance (with minimum data staging time for the grid).....	58
4.3	Second application: preparing a specific text corpus	61
4.3.1	Experiment with preparation of a specific text corpus.....	62
4.3.2	Results and analysis of text processing experiment	62
4.4	Narrative	63
4.4.1	Local Server	63
4.4.2	Computational grid	64
4.5	Summary.....	65
5	Conclusion and Reflection	66
5.1	Introduction	66
5.2	Summary of chapters	66
5.2.1	Chapter 1 – Introduction	66
5.2.2	Chapter 2 – Literature review	66
5.2.3	Chapter 3 – Research Methodology	66
5.3	Critical reflection.....	67
5.3.1	Findings in respect of objective 1	67
5.3.2	Findings in respect of objective 2	68
5.4	Contribution of Study.....	72
5.5	Recommendations	72
5.6	Limitations	73
5.7	Future work	73
5.8	Personal Reflection	73
	Reference list	75
	Addendum.....	83

List of figures

Figure 1-1: Generic representation of the South African National Grid	4
Figure 1-2: Comparisons in scientific computing	5
Figure 2-1: Five decades of Moore's Law (Scherer 2015).....	9
Figure 2-2: Multicore chips perform better than single-core processors based on Intel tests using SPECint2000 and SPECfp2000 benchmarks (Geer 2005) .	11
Figure 2-3: Architecture of a single-core vs a multicore processor (Akhter & Roberts 2006)	12
Figure 2-4: A small grid infrastructure	19
Figure 2-5: Grid architecture (Foster et al. 2003)	19
Figure 2-6: Grid architecture compared to internet architecture (Foster 2001).....	24
Figure 2-7: Grid architecture (Foster & Kesselman 2004)	24
Figure 3-1: A variation of the rational problem-solving process (Olivier 2009)	38
Figure 3-2: Two dimensions: Four paradigms (Adapted from Burrell & Morgan 1979)	40
Figure 3-3: Overview of the research methodology followed in this research	44
Figure 3-4: Typology of mixed research (Adapted from Leech & Onwuegbuzie 2009)	46
Figure 4-1: Overall performance of both systems	55
Figure 4-2: Variation in overall system performance	56
Figure 4-3: Overall performance when using minimum data staging time for the grid	60
Figure 4-4: Variation in overall performance when using minimum data staging time for the grid	61

List of tables

Table 1 Comparison of a grid and a conventional distributed system (cluster) (Németh 2003)	15
Table 2: Averages of all experiments (real-life scenario).....	54
Table 3: Overall performance of the grid vs local server	54
Table 4: Variance on the overall performance of the grid vs local server	54
Table 5: Averages of the grid vs local server for graph plotting.....	54
Table 6: Standard deviation values for the overall experiment on the grid vs local server	55
Table 7: 2K on maximum performance on the grid	57
Table 8: 2K maximum performance on the grid with minimum data staging time	57
Table 9: Averages of all experiments (minimum data staging time)	58
Table 10: Overall performance of the grid vs local server (minimum data staging) ..	58
Table 11: Variation on the overall performance of the grid vs local server	59
Table 12: Averages of the grid vs local server for graph plotting (minimum data staging)	59
Table 13: Standard deviation values for the overall experiment on the grid vs local server	60
Table 14: Job completion times for text corpus preparation	62
Table 15: Overall performance with second application	63
Table 16: 2K Max performance on CHPC	83
Table 17: 6K Max performance on CHPC	83
Table 18: 12K Max performance on CHPC	83
Table 19: 2K 8cores performance on UJ	84
Table 20: 6K 8 cores performance on UJ	84
Table 21: 12K 8 cores Performance on UJ	84
Table 22: 2K 8 cores on local server	85
Table 23: 6K 8 cores on local server	85
Table 24: 12K 8 cores on local server	85
Table 25: 2K 22 cores on local server	86
Table 26: 6K 22 cores on local server	86
Table 27: 12 K 22 cores on local server	86
Table 28: 2K Max performance on CHPC with minimum data staging time	87

Table 29: 6K Max performance on CHPC with minimum data staging time	87
Table 30: 12K Max performance on CHPC with minimum data staging time	87
Table 31: 2K on 8cores UJ with minimum data staging time	88
Table 32: 6K on 8 cores with minimum data staging time	88
Table 33: 12K on 8 cores with minimum data staging time	88
Table 34: Results on the Grid (UJ)	89
Table 35: Results on the Multicore Server	89

1 Introduction

1.1 Background

The rapid growth in quantity of compute-intensive applications in the fields of science, engineering, medicine and commerce has stimulated significant demand for computational power in the field of computing. The term computing here “includes designing and building hardware and software systems for a wide range of purposes like processing, structuring, managing various kinds of information and doing scientific studies” (Nagaraju & Anitha 2012).

The increasing demand for more computational resources in fields such as mathematical computation, scientific simulation and climate forecasting has driven the development of high-throughput performance computing (Sharma & Mittal 2013). One challenge in meeting this high computing demand has been the high cost of acquiring high-performance processors. On the other hand, organizations may have existing computing resources that are underutilized, as is the case on the North-west University (Vaal Triangle Campus). This type of disparity has led to research that explores the area of distributed systems in a bid to discover solutions to provide additional options for high-performance computing.

Prior to the inception of grid computing, high-performance computing has mostly been achieved by specialized parallel computers. Parallel computing is the utilization of multiple processors in which all processing units work in parallel and hence increase system throughput. However, parallel computing was first implemented in super computers (limited to a single machine). This mechanism for achieving high-throughput computing has proven to be effective, but extremely expensive in cost. High-end multicore servers have proven to be effective in providing the much needed computational power; however, the access of researchers to these systems has been limited because these servers are expensive to acquire.

The majority of the available computing resources at universities and institutions of higher learning, such as personal computers and office workstations, are vastly underutilized and remain idle most of the time. These idle computing resources can be used in solving the problem of continuous demand in high-performance computing. Hence we explore a distributed computing technique (Sharma & Mittal 2013), of grid computing as a way of aggregating and integrating these available resources into a

single high-throughput computer and making it available to researchers for solving demanding problems.

Grid computing may be defined as “a type of distributed computing that permits and ensures the sharing of aggregated resources across multiple administrative domains based on availability, capability, performance, cost and users’ quality-of-service requirements” (Sharma & Mittal 2013). A grid consists of loosely coupled, heterogeneous and distributed computers synergized for the purpose of providing high-throughput computing. Different grid types exist and we discuss the classification of grids in chapter 2 of this study.

When these synergized computing resources are owned by the same organization such as a university, it can be called a campus grid. A campus grid can be used to alleviate demand on high performance research computing platforms by assigning compute-intensive tasks to other computers that have idle cycles not used by the applications that normally run on them.

With the availability of both high-end multicore processors and computational grids, it is important that we investigate the performance and management trade-offs between the two technologies and discover effective ways of attaining high-throughput computing at an optimized cost for research purposes.

1.2 Problem statement

Though academic institutions continue to procure computing resources for use by students, staff and researchers in a bid to advance effective education and research, most of these computing resources have not been used to their full potential. The underutilization of computing resources in academic environments remains a challenge in our institution. While campus computing resources are underutilized, there is a continuously growing need for more computing capacity to solve compute-intensive problems in research.

Satisfying the research demand for high-end throughput computing poses an enormous challenge in the form of the high cost of acquiring more computing resources, because of budget constraints.

1.3 Specific focus of this study

1.3.1 The nature of applications used in this study

The research follows a pragmatic approach in trying to understand and solve a local problem experienced by researchers of the Multilingual Speech Technology (MuST) research group. The focus of experiments conducted is therefore on some applications used in pattern recognition and speech technology. However, lessons learnt from this study may also benefit other researchers that need to decide on the type of computing platform they need for computational work and collaboration.

As a research niche area of North-West University (Vaal Triangle Campus), MuST consists mainly of engineers and computer scientists actively involved in speech technology and pattern recognition research. It creates speech technologies for the less-resourced languages of the world, and tries to find new ways of doing this quickly and cost-effectively. In order to be able to build these systems, many questions have to be answered: How can computing systems be made to understand the many different accents within a single language? How do people pronounce proper names they have never heard before? How to capture and understand the essence of a language from a limited set of speech samples?

While creating and applying speech technologies within a multilingual context, MuST provides a focused, project-oriented learning environment to younger researchers, and provide senior researchers with significant freedom in choosing how they contribute to the group's activities. Initiated from the Faculty of Economic Sciences and Information Technology, Vaal Triangle Campus, the research activities include a small student presence at the CSIR in Pretoria, as well as a satellite research office in Hermanus, where group members and visiting scientists can spend time away from it all, in an environment that is conducive towards focused research.

The MuST research group owns a high-end multicore server that is used by researchers in executing compute-intensive applications. However the high demand for this computing hardware by researchers competing for processing time presents a challenge on the availability of this multicore server. This challenge creates a bottleneck in the research process and negatively impacts on timelines of research projects.

1.3.2 South African National Grid

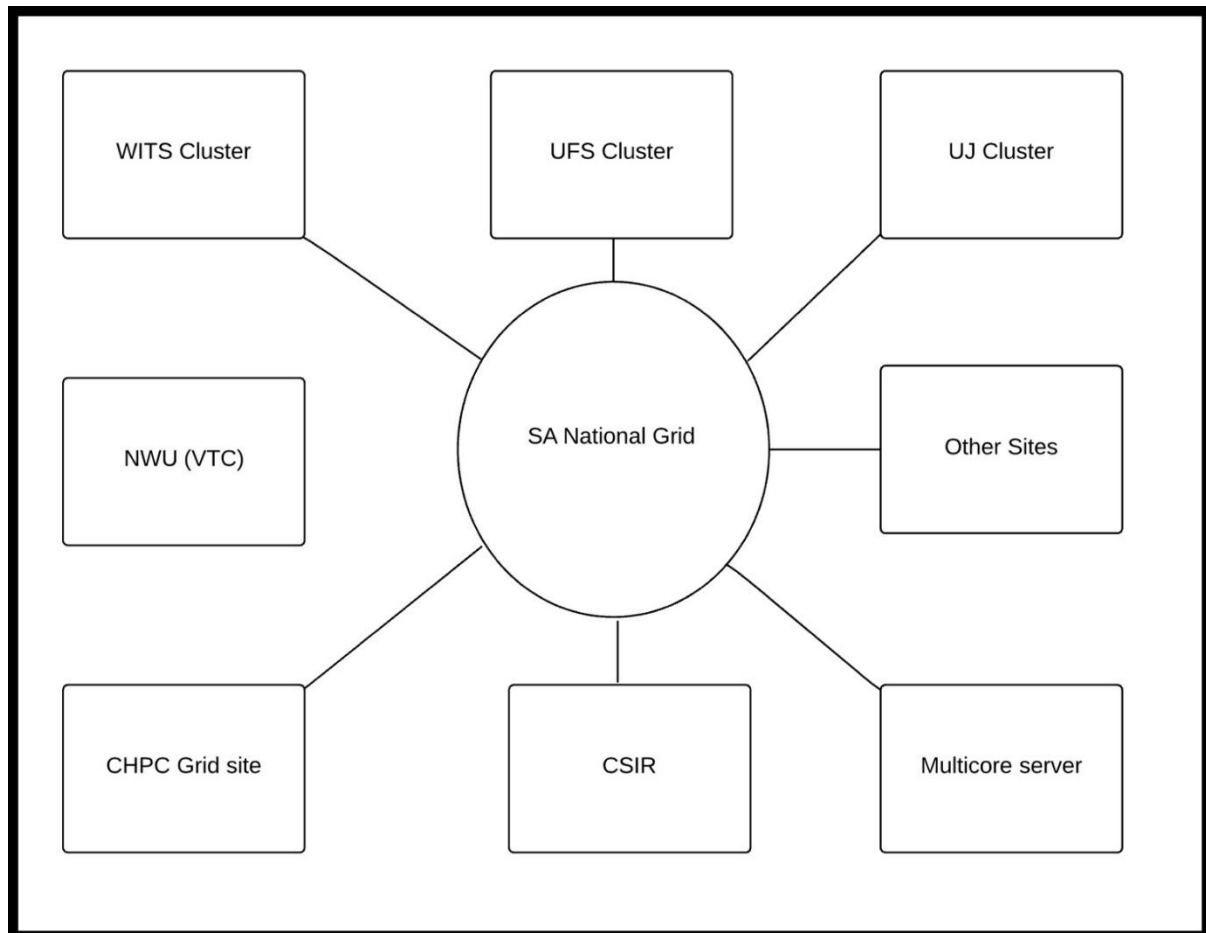


Figure 1-1: Generic representation of the South African National Grid

The diagram above shows a generic architecture of the South African National Grid and depicts several sites connected through the gLite middleware software stack. The NWU-VTC site is still being configured hence it is not connected to the South African National Grid. The above grid is used in this study and the sites selected for the execution of the experiments are shown in the diagram.

1.3.3 Difference between this study and other comparisons

Numerous studies on comparisons of the scaling of scientific applications on various computer platforms have been conducted by researchers in the field of computer science and engineering (Mabakane 2011). Many other types of comparisons are possible and can be added in the diagram below.

This study, not only compares two computing platforms based on pure performance in terms of job completion time, but also by observations made during the set-up and execution of experiments on the different computing infrastructures so as to draw conclusions on the benefits provided by one system over the other. The comparison is not solely based on pure performance as this would be an unfair comparison since we do not perform the standardization of hardware resources such as RAM, Flops, Storage, I/O bus speeds. The research exerts a major focus on the workflows of the two systems in analysing the advantages of one system over the other so as to ultimately outline factors to consider when trying to make a decision on which computing system a MuST researcher can utilise effectively

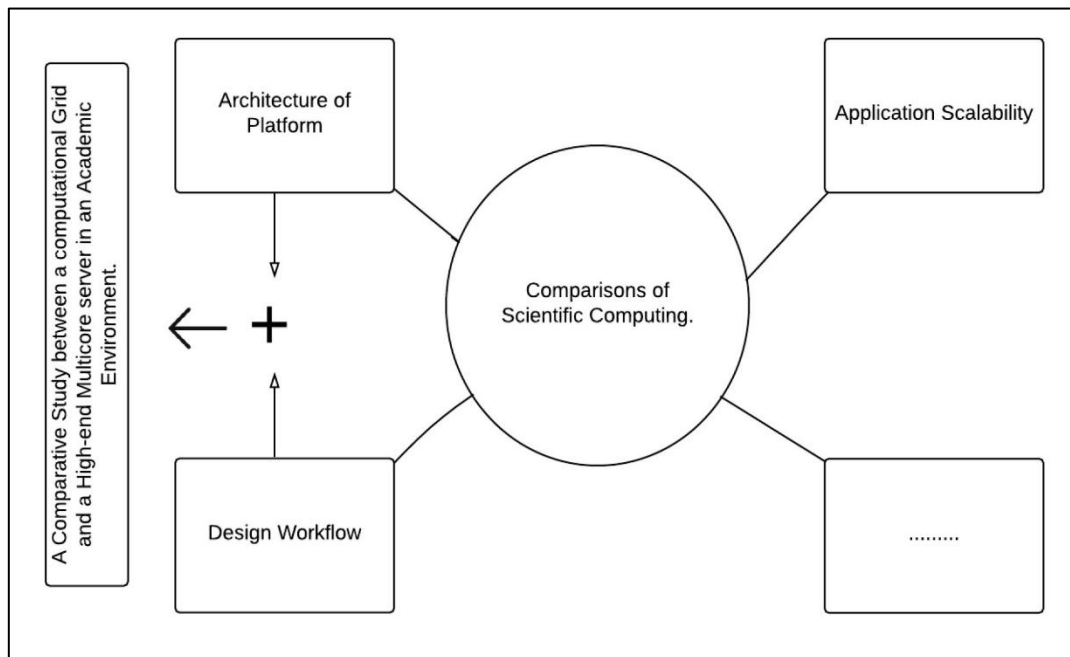


Figure 1-2: Comparisons in scientific computing

1.4 Study objectives

Although several studies comparing the performance of grid computing have been done, this study will consider a more specific case of comparing a campus grid or grid site in an academic environment with a local computational server, in view of the emerging demand for high-performance computing in academic environments. In particular, the objectives of this study are:

- Utilize an existing campus grid on the South African national grid and local multicore server to set up and run compute-intensive experiments.
- Determine the choices and trade-offs for performance and system management benefits of a campus computational grid and a multicore server.
 - The complexity of the administrative domains; ease of gaining authorization and access to other resources on campus and other sites.
 - The complexity of porting applications to the grid.
 - The overhead and availability of site administrators and their capacity to operate the services efficiently.

1.5 Research methodology

This research uses mixed methods, explained in detail in chapter 3, to compare a computational grid and a local multicore server. The research design of partially mixed concurrent domination status is used and the methodology of an experiment as well as a narrative were used to meet the objectives of the study. A comprehensive discussion on the research methods is in chapter 3 together with the research paradigms and methodologies used in this research to achieve the research objectives.

1.6 Layout of chapters

The study comprises five chapters outlined as follows:

Chapter 1 – Introduction: This chapter introduced the research; it focused on the introduction of the main concepts covered in this study, the research problem, study focus, objectives and research methodology.

Chapter 2 – Literature review: This chapter reviewed past work done in the research area and outlined the architecture of multicore processors as well as grid computing in detail.

Chapter 3 – Research Methodology: This chapter introduced the concept of scientific research and it continued to outline the research methods used in this study. Furthermore, it placed this study into a paradigm and discussed in detail the research design and methodology selected to meet the study objectives of the research.

Chapter 4 – Results Analysis: This chapter discussed the results obtained in the study and further narrated the researcher's experience in using the two computing infrastructures being compared. It then made findings based on this discussion.

Chapter 5 – Conclusion and Reflection: This chapter concluded the study by giving a summary of all the chapters discussed in the previous chapters. It then critically reviewed these findings in order to determine the contribution of the study, as well as make recommendations based on these findings. Possible future work is suggested and a personal reflection is given by the researcher.

1.7 Summary

Chapter 1 dealt with the background of the research area and defined some concepts regarding the study. The problem statement was clearly outlined and a focus of the study as well as the rationale behind the use of pattern recognition and speech technology applications was given. Study objectives were defined and the research methodology used to meet these objectives was briefly noted. The chapter concluded by giving the layout of the study in the form of chapter descriptions.

2 Literature review

2.1 Introduction

This chapter investigates existing literature in the field of high-end computing. It begins by looking at the history of microprocessors and underlying concepts in their design. The discussion then turns its focus to the different computing infrastructures such as, multicore computers, cloud computing as well as grid computing that can be utilized in high-end computing. The literature review then explores a computational grid as well as a multicore server which are the hardware infrastructures that are being compared in this study. A classification of computational grids is briefly outlined and particular attention is given to campus grids. The literature study concludes with a short summary which summarizes the chapter and establishes a base for the chapter that follows.

2.2 Towards using multiple processors

This section introduces ways in which processing power can be attained from multicore processors and grid computing.

2.2.1 A brief history of microprocessors

“Driven by a performance-hungry market, microprocessors have always been designed keeping performance and cost in mind” (Venu 2011). “Parallel processors have a long history, going back at least to the Solomon computer of the mid-1960s. The difficulty of programming them meant they were primarily employed by scientists and engineers who understood the application domain and had the resources and skill to program them” (Blake et al. 2009). Several companies created and offered parallel machines, or single-chip multicore microprocessors as they have been styled. Their evolution has been dramatic.

The fundamental guiding principle for the discussion of “computer architecture is known as Moore’s Law. In 1965, during an interview, Gordon Moore stated that the number of transistors on a chip will roughly double each year, later refining it in 1975 to every two years. However, what is often quoted as Moore’s Law is Dave House’s revision that computer performance will double every 18 months.” (Schauer 2008). The graph below depicts the evolution of microprocessors over five decades since the inception of Moore’s Law.

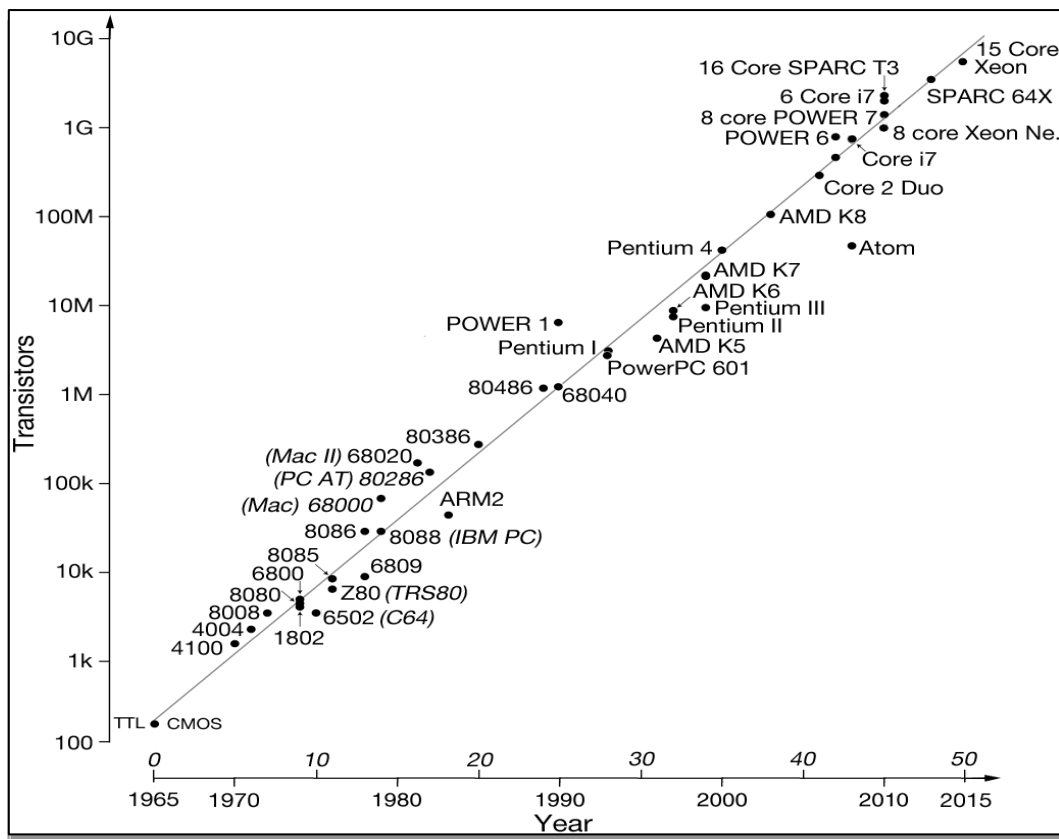


Figure 2-1: Five decades of Moore's Law (Scherer 2015)

The graph above shows the first microprocessors and their manufacturers. It further shows how microprocessors have evolved over 50 years, with every generation of processors growing smaller, faster and increasing in performance. Note the exponential scale of the y-axis. The graph shows that Moore's Law has reigned over a number of years, with the number of transistors roughly doubling every two years. Throughout the 1990s and early 2000s, microprocessor performance was directly linked to frequency (clock speed); "higher frequency meant a faster, more capable computer. The overall performance of a system in terms of power consumption and heat dissipation" (Schauer, 2008), was challenged by the inability to double the frequency of a single core. The result was that the increase in processor performance began slowing. "Chip performance increased 60 percent every year in the 1990s, but slowed to 40 percent per year from 2000 to 2004 and performance only increased by a further 20 percent in 2005, according to Linley Group president Linley Gwennap" (Geer 2005).

A new innovation termed "parallel processing" was suggested in the early 1990s to reduce power consumption (Chandrakasan et al. 1992). This innovation was well

received by architecture designers and therefore all processors nowadays exploit this innovation. Clock frequency had to be increased, which in turn generated more heat in a processor, which needed dissipation, therefore parallel processing could not on its own support the growing speed of a microprocessor (Roy et al. 2008). Parallel computing enables the full processing power of a multicore computer to be used by dividing a computationally intensive job into various tasks that are assigned to the available processors, which share memory, thereby reducing computation time (Varshney et al. 2012).

In seeking a solution to increase performance, the focus was directed to increasing the instructions per clock (IPC) to acceptable thermal dissipation levels. Furthermore, performance could be improved by reducing the number of instructions required to complete a task. The special technique, used to achieve this increase in performance is called single instruction multiple data (SIMD), was first introduced by Intel in 1996 as a 64-bit integer SIMD processor with MMX technology. Even though pure performance is important, the implications for power consumption can never be ignored when measuring performance (Gepner & Kowalik 2006).

This drive towards high computing performance paved the path for parallel computing. The onset of massive parallel computing was ignited by two unrelated fields: computational science and the video-game industry. Science constantly needs to solve large problems related to among others biology and geospatial information systems (Tapia & D'Souza 2009; Bernhardt et al. 2011), to mention only a few, in a reasonable amount of time and the video-game industry constantly needs to achieve real-time photo-realistic graphics.

Power4, a multicore processor that achieved greater performance through improved communication bandwidth, was introduced by IBM in 2001 (Tendler et al. 2002). This innovation by IBM would later ignite responses from several companies, with Intel reaching new levels of energy-efficient performance with their Intel Core™2 Duo processors designed as part of a mobile processor family. This new introduction to the Intel processor family was the first Intel mobile micro-architecture to use a chip multi-processor (CMP). The Intel Core™2 Duo, “was built to achieve high performance, while consuming low power and fitting into different thermal envelopes” (Gochman et al. 2006).

Because of the improvements in circuit technology and performance degradation in wide-issue, super-scalar processors, multicore technology had by 2007 become, “the

mainstream in CPU designs. It embeds multiple processor cores into a single die to exploit thread-level parallelism for achieving higher overall chip-level IPC” (Peng et al. 2007); (Hammond et al. 1997). The multicore concept was used in embedded systems for some time to execute specialized applications until Intel and AMD introduced the technology to produce commercially available multicore chips (Schauer 2008).

The biggest motivation to the adoption of the multicore processors was an attempt to address power and cooling challenges while delivering high performance. The diagram below shows a multicore processor outperforming a single-core processor, based on the benchmark reports of the Intel tests using SPECint2000 and SPECfp2000 (Roy et al. 2008).

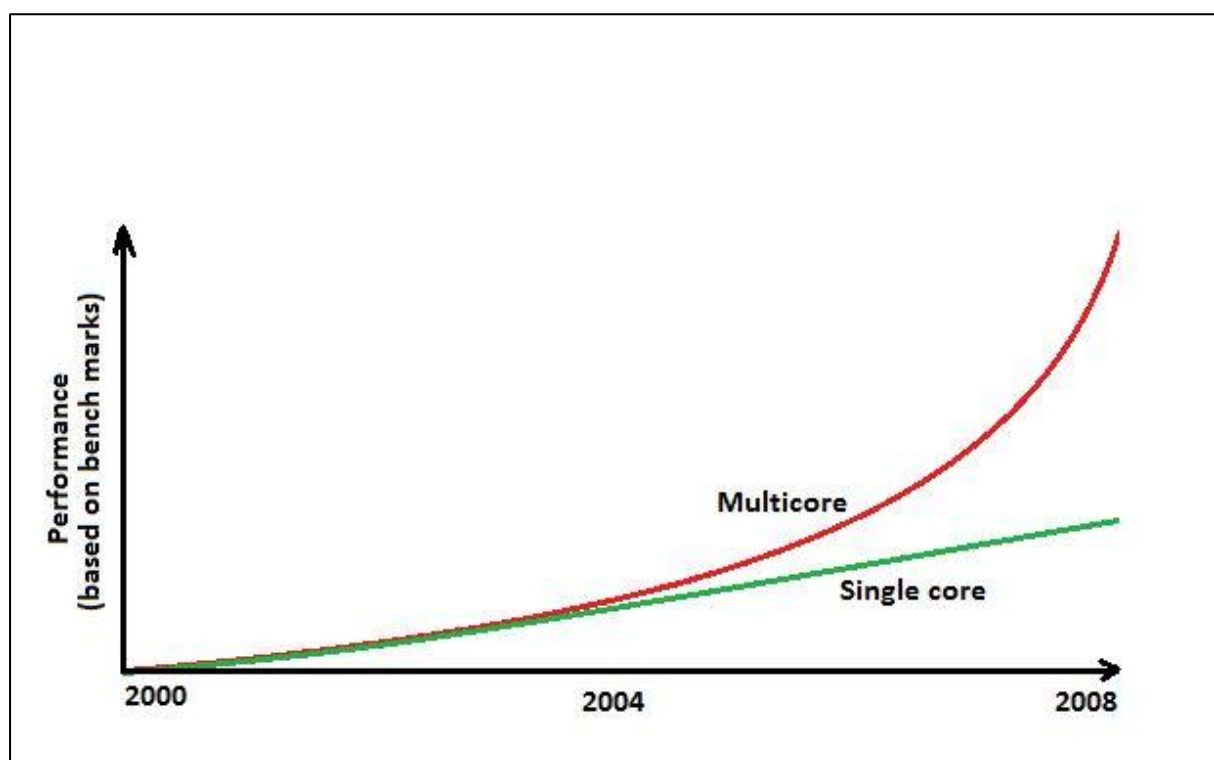


Figure 2-2: Multicore chips perform better than single-core processors based on Intel tests using SPECint2000 and SPECfp2000 benchmarks (Geer 2005)

The above diagram clearly depicts that multicore processors were projected to have an advantage in performance over single cores, in subsequent years.

“A multicore is typically a single core processor which contains several cores on a chip. The cores are fully functional with computation units and caches and hence support multithreading” (Wang et al. 2010). The different architectures of a single-core and a multicore processor are shown in the diagram below, clearly showing that a multicore system is an aggregation of single-core processors.

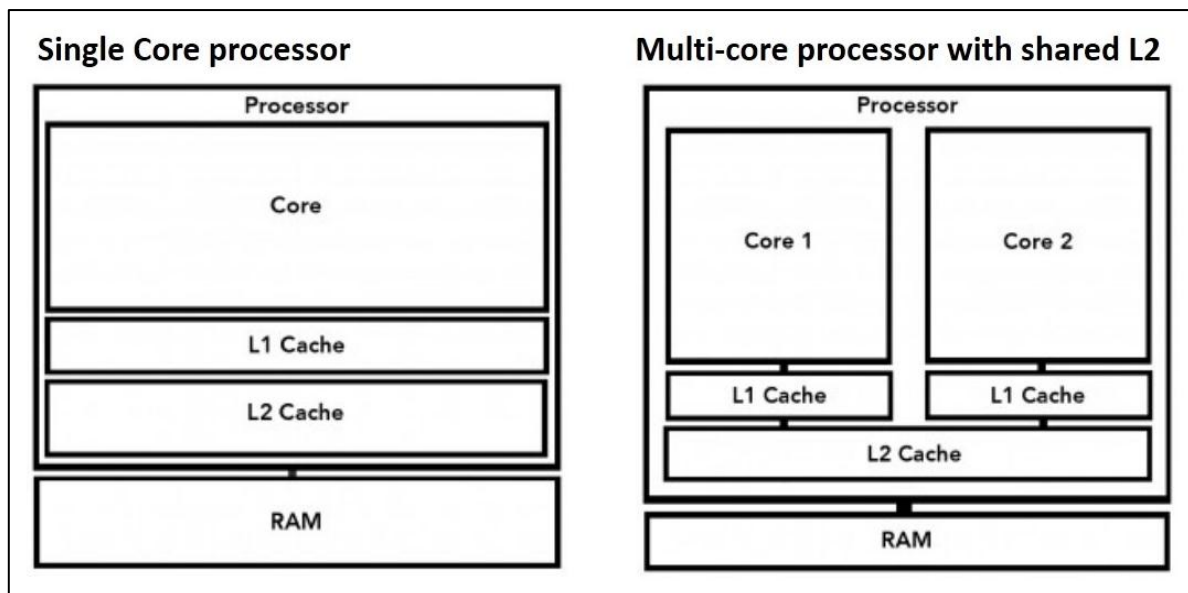


Figure 2-3: Architecture of a single-core vs a multicore processor (Akhter & Roberts 2006)

“A multicore chip-level processor aggregates more than one independent cores into a single die for example a dual-core processor contains two cores .A multicore processor implements multi-processing units on a single die.” (Roy et al. 2008) One basic distinction between a single-core and a multicore processor, as shown by the diagram above, is the level of cache dedicated to a processor core. As depicted in the diagram above, each independent processor core in a multicore system has an individual L1 cache and a shared L2 cache.

It is important to note that the hardware parallelism of multicore processors must be exploited by the use of parallel programming, otherwise they offer no performance benefit to the user (Asanovic et al. 2006). The individual cores on a multicore processor do not essentially run as fast as the maximum performing single-core processors, but handle more tasks in parallel thereby improving the overall system performance (Geer 2005). Applications do not automatically get faster as cores are added; software programs must be written to harness this parallel processing power. Microprocessors have continued to transform the world we live in and more resources have been invested in trying to improve their performance. Several techniques including data-level parallelism, instruction-level parallelism and hyper-threading already exist and these have significantly increased the performance of microprocessor cores and multicore processors (Goodacre & Sloss 2005).

2.2.2 Parallel computing

High-end processing for problem solving in the past was achieved through the use of supercomputers. The creation of programs in the supercomputing environment was not easy and hence other techniques were developed to satisfy the growing computational power needs of the computing community (Sharma & Mittal 2013). Some of the techniques that were developed are parallel computing, peer-to-peer computing, cluster computing, grid computing and cloud computing.

“Parallel computing is a form of computation which deals with hardware and software computation in which many calculations are carried out simultaneously” and this is achieved through the use of specialized hardware that supports parallelism. Multi-processor and multicore computers contain several processing elements within a single machine in which main memory is shared between all the processing units. This hardware capability is enabled by software programming languages and libraries that manage memory sharing and processing elements interaction. The increase in system throughput is achieved by the coordination of processing units working in parallel. (Schmidberger et al. 2009)

Parallel computing has drawbacks; for example, its effectiveness is limited to a single machine, shared resources must be used sequentially, problems of interdependency must be overcome and not every problem can be subdivided into smaller units and dealt with simultaneously (Sharma & Mittal 2013). These limitations led research to explore other techniques, such as peer-to-peer computing.

2.2.3 Peer-to-peer computing

Peer-to-peer computing is a computing technique in which computers in the network can act either as a client or server for other computers, this allows shared access to numerous resources such as files, peripherals and sensors without the need for a central server. It can also be described as a communication structure that allows the same capabilities and abilities to each and every part involved (Azeez & Abidoye 2011). The challenges arising from peer-to-peer computing are that the number of computing resources that can be connected are limited and homogeneous, hence another technique of cluster computing was born.

2.2.4 Cluster computing

A cluster computer is a parallel and distributed system made up of a number of stand-alone (similar/identical) computers interconnected through a high-speed local area

network (LAN) cooperating as a single integrated computing resource. Clusters are developed and deployed to increase availability and performance, unlike a single stand-alone computer. The technique of clusters also offers a cost-effective solution to the development of computer systems that provide more compute-intensive processing as opposed to a single computer (Gandotra et al. 2011). The drawback in increasing the cluster size in terms of computing nodes that can be added to it are that every computer must have the same hardware and operating system, job management and scheduling are centralized and computer clusters are often contained in a single geographic location, such as a single computer laboratory (Sharma & Mittal 2013).

2.2.5 Grid and cloud computing

The weaknesses of cluster computing stated above gave rise to the further development of grid computing as a technique for providing uniform and controlled access to non-identical computing resources and a seamless global aggregation. “The concept of Grid computing started as a project to link geographically dispersed supercomputers, but now it has grown far beyond that intent” (Baker et al. 2002). Grid computing platforms are also created from a number of underutilized computing resources connected over a wide area network (WAN) (Parashar et al. n.d.). The inception of grids introduced major benefits into the areas of data exploration, simulation science, high-throughput computing, collaborative science and engineering (Berman & Hey 2004). Challenges such as scheduling, security and lack of design enforcement in grid computing, as well as a need for more services on the computing platform, paved the way for a more recent computing technique known as cloud computing.

Grids	Conventional Distributed Environment (Cluster)
1. Virtual pool of resources	Virtual pool of computational nodes
2. Access to a resource may be restricted	Access to a node means access to all resources on the node
3. Resources span multiple trust domains	Nodes belong to a single trust domain
4. A user has access to the pool but not to individual nodes	A user has access (credentials) to all nodes in the pool

5. The user has little or no knowledge about each resource	The user is aware of the capabilities and features of the nodes
6. Elements in the pool are $\gg 100$ and are dynamic	Elements in the pool range from 10-100, and are static.

Table 1 Comparison of a grid and a conventional distributed system (cluster) (Németh 2003)

The table above tries to differentiate between the two prominent structures in distributed computing. This distinction is important in understanding grid computing and in what way it is better suited to meet the challenges of high computation demand in research today. It is also of vital importance to acknowledge conventional distributed environments, as this leads to a clear appreciation of the changes made on that infrastructure in enabling grid computing.

Cloud computing makes applications available in a flexible execution environment primarily located in the internet. Cloud computing is a parallel and distributed system made up of an aggregation of inter-connected and virtualized computers combined to provide a single unified computing resource, depending on service-level agreements obtained through negotiation between the cloud service providers and cloud service consumers. Cloud computing provides a tremendous benefit to small and medium business enterprises that seek to outsource their data-center infrastructure and to large companies that seek to balance their computational load during peak times. Cloud computing is an extension of grid computing in that the capabilities of business applications are provided over a network (Buyya et al. 2009).

Cloud computing is a combination of information technology (IT) services where the power of modern computers is utilized efficiently so as to modernize businesses and use IT as a service to satisfy business needs such as parallel batch processing, mobile interactive processes and compute-intensive business analytics that respond in real time in meeting the needs of the clients. This model allows real time business transactions to be executed and many scattered business applications to communicate efficiently over the network (Marston et al. 2011).

A simple definition of cloud computing denotes it as, “clusters of distributed computers (largely vast data centers and server farms) which provide on-demand resources and services over a networked medium such as the internet.”(Sultan 2010) Cloud computing surpasses grid computing by its provision of infrastructure as a service, through which products offered via this model include the remote delivery of full

computer infrastructure (e.g. servers, storage devices and virtual computers). A platform as a service allows the management of operating systems, databases and middleware to be managed remotely without the users having to be involved as in traditional computing. The software as a service layer enables software to be installed and maintained on the data centers; users simply access it via the network without any regard for the complex software and hardware management (Sultan 2010)

The preceding paragraphs make it clear that cloud computing is the latest innovation in distributed parallel computing techniques after grid computing. However, for the purposes of this research, grid computing has been selected over cloud computing, as it mainly focuses on providing researchers in academia with massive computation capabilities and data storage facilities as they conduct research. The business agility catered for by cloud computing services are not necessary for this particular research, hence grid computing will be used in aggregating the available computing resources in an academic environment so as to provide researchers with a high-throughput computational capacity.

2.2.6 Origin of grid computing

To fully comprehend grid computing, it is imperative that we first consider the term “grid”. The concept of the “grid” system is analogous to the “electricity grid system”. Grid computing therefore similarly aims to provide endless and universal access to expensive but high-quality computing resources to regardless of their physical location.

To understand the future of the grid, it helps to study the history of other infrastructures. Studying the development of other forms of infrastructure, such as railroads, telephones, the electricity grid, telegraphs and banking, which are also distributed, was critical in establishing concepts on grid computing. As we study these forms of infrastructure, which are all quite different on the surface, we begin to notice that they have in common a number of striking features that include connecting links, service providers at different sites, standardization and co-operation agreements and distributed end users expecting quality of service (Smarr 1998).

“Like many significant concepts and technologies that we now take for granted, Grid ideas have been inspired by, and were first applied to, problems faced by researchers tackling fundamental problems in science and engineering. Building on ideas first expounded in the 1960s and given concrete form by Grid pioneers in the 1990s, the

scientific community has continued to lead the development of Grid technologies that will act as a key enabler for twenty-first century science and society” (Berman & Hey 2004).

“The origins of a ‘grid’ to support scientific research can be traced back to the Internet pioneer J. C. R. Licklider” (Berman & Hey 2004). In 1962, while Licklider headed two Advanced Research Projects Agency (ARPA) departments, Behavioral Sciences and Command and Control. He brought to ARPA a vision of a future computer network that was inspired by an analysis made during a long time he spent in organizing and manipulating experimental data in his research (Berman & Hey 2004).

Licklider described his hypothetical network as follows: “If such a network as I envisage nebulously could be brought into operation we could have at least four large computers, perhaps six or eight small computers and a great assortment of disc files and magnetic tape units not to mention remote consoles and teletype stations-all churning away”(Waldrop 2000).

Licklider’s vision was instrumental in contributing to the formation of ARPANET; this invention that brought about the innovation of the internet has proved to be a major power in science and engineering. Without the internet it would not have been possible to invent technologies such as grid computing and hence with the internet becoming a reality, many distributed applications have been enabled and science and engineering have benefited immensely from these technologies. “The Grid is our latest and most promising attempt to realize Licklider’s vision” (Berman & Hey 2004).

In the near future, technology will continue to provide more potential capability and capacity and will need to be integrated into the grid technologies. This vision will lead to the establishment of de facto standards to manage the ever evolving technological infrastructure. “Today Grids seek to use common infrastructure and standards to promote interoperability and reusability, and to base their systems on a growing body of robust community software” (Berman et al. 2003).

The accelerated development of grid computing systems had significantly placed them as the promising next generation computing platforms since the early 1990s (Foster & Kesselman 2004). They enable, “sharing, selection, and aggregation of geographically distributed resources for solving large-scale problems in science, engineering and commerce” (Abramson et al. 2002). “The grid can be thought of as a distributed system with non-interactive workloads that involve a large number of files. What distinguishes

a grid from conventional high-performance computing systems such as cluster computing is that grids tend to be more loosely coupled, heterogeneous and geographically dispersed. Although a single grid can be dedicated to a particular application, commonly a grid is used for a variety of purposes” (Nagaraju & Anitha 2012).

Grid computing provides a way of efficiently utilizing computer resources in an organization. It decreases the computing cost and turnaround time needed to complete a job and increases available computing resources. The use of a grid middleware layer in communicating with heterogeneous hardware and databases is essential to the operation of the grid. This layer provides an interface between the grid resources and the applications that need to use the grid. (Sharma & Mittal 2013).

Grids are often classified as either compute grids or data grids. Compute grids emphasize the aggregation of computational resources which are specialized for high performance computing. On the other hand data grids support federation, integration and mining of data resources, the nodes on data grid have need for high-bandwidth network access. “Other classifications, such as a hallway, campus, enterprise and global grid, indicate the scale or other properties of the grid. In practice, grids tend to display characteristics of various functional classifications, and their scale tends to increase over time, so the distinctions are often blurred” (Srinivasan & Treadwell 2005).

2.2.7 Overview and architecture

The popularity of the internet, coupled with the establishment of high-speed networks as well as high-end processing computers, has had a huge influence on the way in which we interact with computers today. Together these developments have led to the design and development of a distributed parallel computing structure that is a single, unified computing resource now known as grid computing (Baker et al. 2002). “The term Grid is chosen as an analogy to a power Grid that provides consistent, pervasive, dependable, transparent access to electricity irrespective of its source” (Chetty & Buyya 2002).

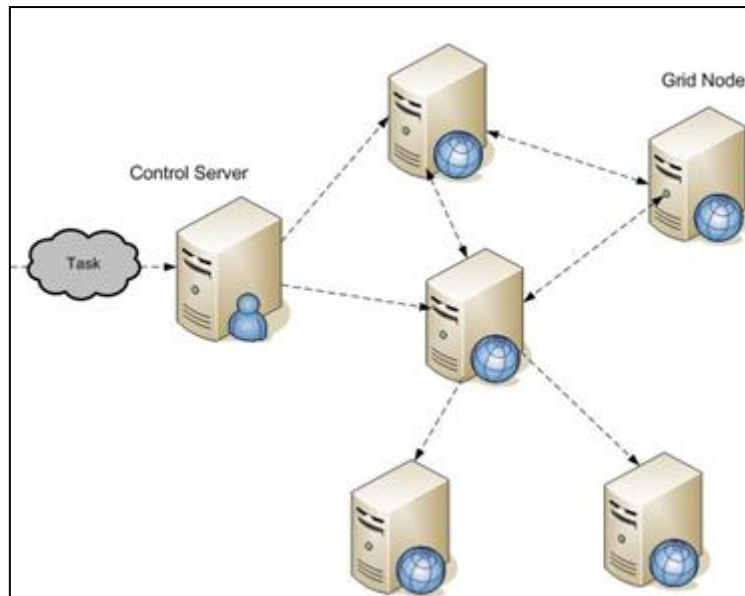


Figure 2-4: A small grid infrastructure

The figure above shows an overview of a grid computing platform where distributed computing resources are connected to provide high-performance computing from a centrally controlled access point.

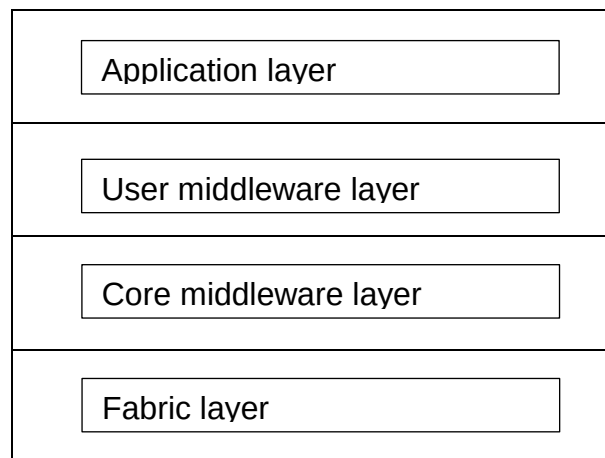


Figure 2-5: Grid architecture (Foster et al. 2003)

The diagram above depicts the internal infrastructure of a grid that has four categories, namely the application layer, user level middleware layer, core middleware layer and lastly the fabric layer. “The application layer is the topmost layer of the architecture. It includes applications in science, engineering, business, medicine, etc. Users of the grid interact with the application layer. The user middleware layer consists of tools such as libraries, debuggers and language compilers. It consists of resource brokers that are used for resource management. The core middleware layer is responsible for the management of processes, resource allocation, accessing the storage, secure

access of information and registry information”(Sharma & Mittal 2013). Lastly, the fabric layer delivers resources which are pooled by the grid, such as CPU time, storage, and sensors.

The dominant area in grid computing research is the resource management and task scheduling. This is mainly because one of the greatest distinctions between a grid and other parallel processing schemes lies with the middleware that manages resource brokering in a grid. Middleware scheduling in grid computing is used to schedule tasks in such a way that perfect resources are allocated to the task submitted by a user. Certain schedulers must provide for, “but not limited to advanced resource reservation, service-level agreement validation and enforcement as well as job and resource policy management and enforcement for best turnaround times.” (Kumar & Jangra 2013).

Arguably the most important component of the computational grid is the network, which can easily link up distributed computing resources and aggregate them to a single platform where they can coordinate in task execution. At the center of the grid one has networks that effectively support the integration of computers and form a virtual machine that can be used to support execution of a single task or run a single application on a distributed system (Berman et al. 2003). The faster and more reliable the network is, the more effectively the performance of the grid can be exploited, and therefore networks can be a factor that has an impact on the performance of the grid. Schwiegelshohn et al. (2010) agree with the above statement on the importance of networks in the world of computing. They further justify the value of networks by stating that there are only a few sites housing supercomputers around the world and without networks, users of these systems would have to migrate close to these sites to access these computational resources. This would negatively affect researchers who cannot migrate. The sharing of information with other networked devices and the cooperation of computing resources in achieving a single task are made possible by the network. The paper goes on to expose the strength of the networks in linking many companies and disparate research sites in order to achieve a common purpose. This leads the researcher to acknowledge the importance of networks in the system being explored fully.

The performance and reliability of our IT networks has enabled the linking of remote computing sites, bringing many computer users together on a shared platform. This has been compared with an electrical grid, particularly the characteristics of bringing power to many users, even in remote sites, with the help of an effective network

cabling system. An important factor contributing to the success of the internet is a stable infrastructure that provides ways of rendering services that are reliable, easy to access and relatively cheap as technology advances. Such an observation has formed a strong base for grid computing (Schwiegelshohn et al. 2010).

Distributed applications, such as the internet, consist of many collaborating processes that utilize resources of loosely coupled computer systems. Applications may be distributed for the sole purpose of gaining access to remote resources and boosting performance, whereas distributed computing in high-performance computing may be attained by the use of traditional environments such as a parallel virtual machine (PVM) and message-passing interface (MPI) or with the emerging software infrastructure called computational grids (Németh 2003).

There are many grid definitions available and in this study we have selected just a few to indicate our understanding of what a grid is. "The grid is defined as, a system that is concerned with the integration, virtualization and management of services and resources in a distributed, heterogeneous environment that supports collections of users and resources (virtual organizations VOs) across traditional administrative and organizational domains (real organizations)." (Foster et al. 2001)

Foster et al. (2003) define a computational grid as, "a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities." In a chapter on the anatomy of the grid, Foster further adds to the above definition by stating: "A computing grid provides coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations" (Czajkowski et al. 2001).

The grid is not only an infrastructure but a model of aggregating virtual resources such as computers, data, software and people for solving large-scale problems. This model cannot be easily distinguished from a new approach of organizing several separate sites into virtual organizations (VOs) and this leads to another configuration of virtual research organizations designed to link researchers efficiently (Foster et al. 2003).

Stockinger (2007) agrees with the previous definition by considering the grid as "a combination of distributed, high throughput and collaborative systems for effective sharing and distributed coordination of resources which belong to different control domains. Another view is that a Grid is a very large scale resource management system." The grid is further defined by Schwiegelshohn et al. (2010) as "a

computational grid is a distributed system that supports a virtual research environment across different institutions.”

The study is targeted to discover ways of linking researchers together and providing them with a high-end computing platform and for the purpose of this study, Foster et al.'s (2003) definition is preferred, as it focuses on providing high-end throughput computing. This is because the researcher seeks to provide more high-end throughput computing to researchers in academic environments.

The scalability of grid computing with different applications, ranging from science, engineering, humanities and commerce to arts, has motivated its use and acceptance. The worldwide acceptance of the internet has largely been because of many web applications and the diversity these give to users (Blanke et al. 2009). Grid computing has the potential to follow a similar trend with the development of more grid applications to be used in fields other than science and engineering. For this reason, researchers have realized the necessity of intensive research in grid computing and how it can become a much sought after computing resource, just like the internet.

Foster, Kesselman and Tuecke in the article, “The Anatomy of the grid,” state that the grid is concerned with, “coordinated resource sharing and problem solving in a dynamic, multi-institutional virtual organization.” (Foster et al. 2001) It is important to note that the sharing is not just simple file transfer, but the provision of a pervasive computing platform bringing together data, software, people and computers in solving problems in many different fields of research. It should, however, be noted that this coordinated sharing of resources is defined clearly within the bounds and limits of access and a set of standards/rules to be followed when a grid is used (Foster et al. 2002).

The architecture of the grid, according to Schwiegelshohn et al. (2010), is made up of critical components, ranging among others from hardware resources such as networks, processors and storage and domain-independent software to manage the various grid resources such as the grid middleware to lastly application software dedicated to the particular need of the target group using the grid; for example, the Multilingual Speech Technology (MuST) group at the North-west University (NWU) would need application software such as Libsvm, HTK or Kaldi. This assertion agrees with many grid architectures and only differs in terms of the arrangements of these grid elements and the different configurations brought about by different hardware and middleware from one grid to another.

The article, “Describing the elephant: The different faces of IT as service” by Ian Foster and Steve Tuecke, describes the term grid infrastructure as an “important aspect of the grid space – namely, a horizontal infrastructure integration layer”. VOs intend to share resources and execute a range of distributed applications, but the associated workload managers do not integrate well at the infrastructure level due to different configurations. A solution to this drawback, is introducing a mutual horizontal layer which defines and supports a consistent set of abstractions and interfaces for access to shared resources. This horizontal resource integration layer is termed the grid infrastructure, and it enables the decoupling of applications and hardware (Foster & Tuecke 2005).

Grid architecture categorizes fundamental system components, specifying the purpose and function of each and every component of the system and indicates how these components interact together to achieve a common goal. Enabling VO demands that proper sharing relationships be defined, hence at the center of the grid architecture one has interoperability. Interoperability is achieved by defining common protocols, hence our grid architecture is governed by predefined protocols, which define the basic mechanisms and policies used by VOs in exploiting sharing relationships. (Czajkowski et al. 2001).

Protocols define and specify how distributed systems cooperate with each other so as to achieve a specified behavior, and manage the structure of information exchange during this interaction. Protocols manage the collaboration between components, but not the implementation of the components. A service is defined solely as by the protocol that it speaks and the behavior that it implements. As part of the grid architecture we also consider application programming interfaces (APIs) and software development kits (SDKs), which are also significant on the grid architecture (Czajkowski et al. 2001).

The grid architecture, just like the internet structure, is a layered architecture organizing components into layers. Within each layer components share mutual characteristics and build their strength from capabilities provided by the lower layer. The layered structure provides for different components to interact with one another and hence form component classes that interact with one another, enabling a grid computing architecture. This layered structure of the grid architecture compared to that of the internet is shown in the diagram below. This diagram also clearly depicts the

grid architecture as comprising component layers such as fabric, connectivity, resource, collective and application (Foster & Kesselman 2004).

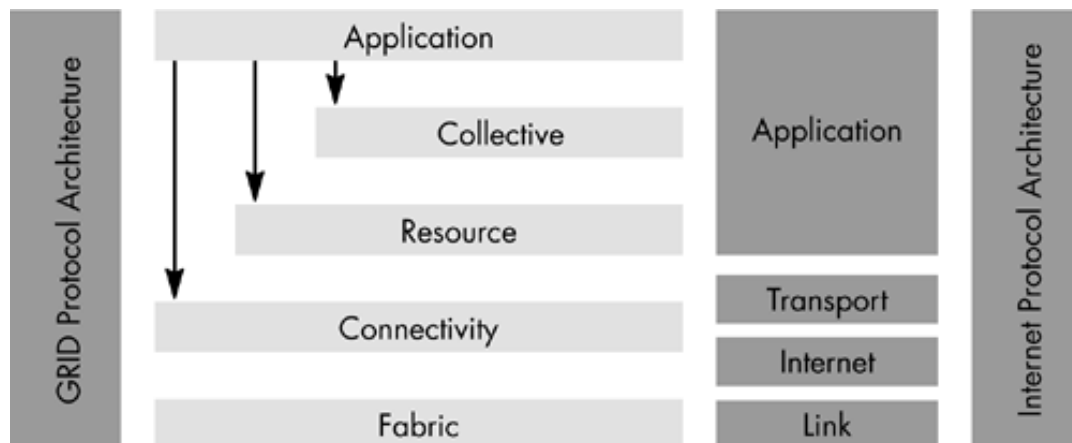


Figure 2-6: Grid architecture compared to internet architecture (Foster 2001)

Foster and Kesselman (2004) give an hourglass model that combines connectivity and the resource layer, unlike the diagram above, becoming one layer, as will be shown in the diagram below. However, the other layers remain unchanged

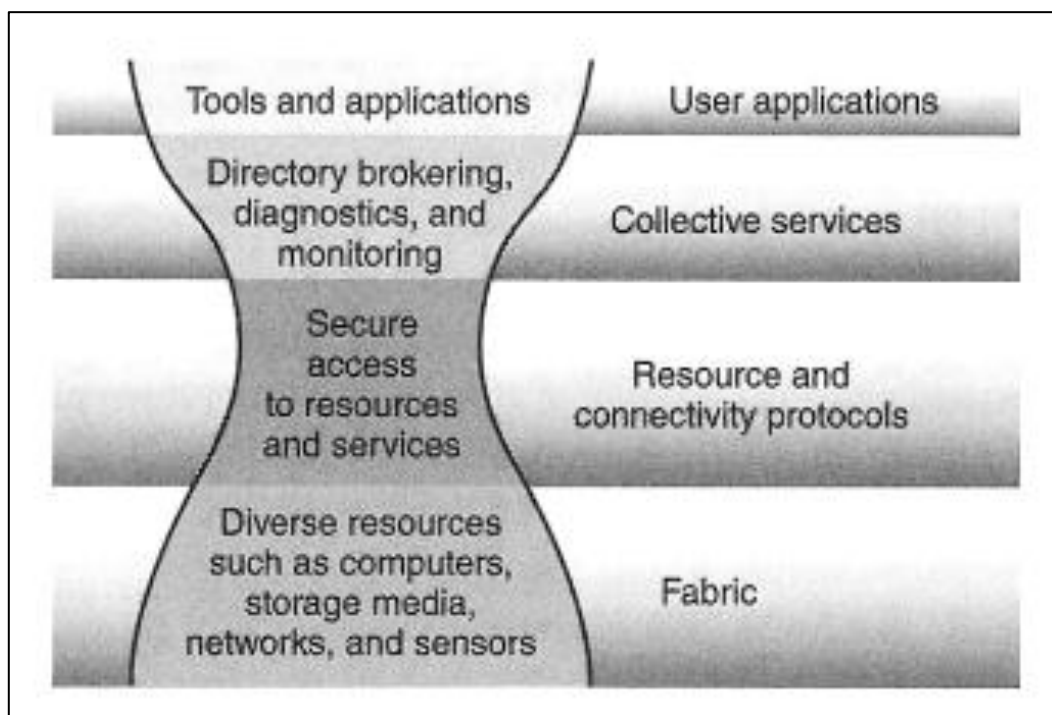


Figure 2-7: Grid architecture (Foster & Kesselman 2004)

“Our grid architecture is based on the principles of the (hourglass model). The narrow neck of the hourglass defines a small set of core abstractions and protocols (e.g. TCP and HTTP), onto which many different high-level behaviors can be mapped (the top of

the hourglass), and which themselves can be mapped onto many different underlying technologies (the base of the hourglass). By definition the number of protocols defined at the neck must be small”(Foster & Kesselman 2004).

In the architecture given by the figure above, the neck of the hourglass houses resource and connectivity protocols, which handles the sharing of individual resources. A full description of the grid architectural layers are given below.

2.2.7.1 Fabric layer

“The fabric layer provides the resources to which shared access is mediated by grid protocols, for example, computational resources, storage systems, catalogs, network resources and sensors. A ‘resource’ may be a logical entity, such as a distributed file system, computer cluster, or distributed computer pool; in such cases, a resource implementation may involve internal protocols (e.g. IP-networked storage protocols such as the NFS storage access protocols or a cluster resource management system’s process management protocol), but these are not the concern of our grid architecture” (Foster & Kesselman 2004).

“Fabric components implement the local, resource-specific operations that occur on specific resources (be they physical or logical) as a result of sharing operations at higher levels” (Trnkoczy et al. 2006). More functionality required from the fabric, for example support for advance resource reservation makes the deployment of grid infrastructure very difficult. Fabric resources implement introspection mechanisms in order to enable the discovery of their structure, state and capabilities, as well as resource management mechanisms that ensure quality of service (QoS), as in the following examples(Foster et al. 2001):

- **Computational resources**, which require mechanisms for initiating programs as well as monitoring and controlling the execution of the resulting jobs. Management mechanisms that determine and enforce QoS and introspection functions for determining hardware, software and critical information pertaining to the current system load and queue state for scheduler-managed resources.
- **Storage resources**, which require mechanisms for storing and accessing files. Management mechanisms that enable resources to transfer effectively are useful introspection functions for ascertaining system characteristics and relevant load information.

- **Network resources**, which require management mechanisms that provide control over the resources allocated to network transfers, as well as introspection functions that determine the network load. These are critical in enabling good fabric functionality in our grid architecture.
- **Catalogs**, which are a specialized storage resource, require mechanisms for implementing catalog query and update operations.

2.2.7.2 Connectivity layer: communications

“The connectivity layer defines core communication and authentication protocols required for grid-specific network transactions. Communication protocols enable the exchange of data between fabric layer resources. Authentication protocols build on communication services to provide cryptographically secure mechanisms for verifying the identity of users and resources. Communication requirements include transport, routing and naming” (Foster & Kesselman 2004). Currently it is safe to assume that most grid communication is achieved through protocols such as TCP/IP protocol stack, specifically the internet (IP and ICMP), transport (TCP, UDP), and application (DNS, HTTP, file transfer protocol [FTP]), which are layers of internet layered protocol architecture (Foster & Kesselman 2004).

Networks are continuously evolving, which means that in the near future grids may be required to use different protocols. A major concern on the connectivity layer is security and hence we base our solution on solid, already existing standards as much as possible. Authentication for VO environments should be characterized by the following (Foster & Kesselman 2004):

- **Single sign-on**; this seeks to enable users to have universal and pervasive access to the grid resources defined in the fabric layer by authenticating or logging on just once.
- **Delegation**; this enables a user to provide a program with the ability to execute tasks on behalf of the user accessing resources that the user has authorized access to on the user’s behalf. The running program should be able to delegate a subset of rights to a process or another program; this is referred to as restricted delegation.
- **Integration with local security solution**; this allows each site or resource provider in a heterogeneous grid to have local security solutions that inter-operate with grid security solutions. This must be achieved without wholesale

replacement of local security solutions but reasonably allow a mapping into the local environment.

- **User-based trust relationships;** for a grid user using multiple resources from several service providers, the security systems of different service providers must not require interaction in order to cooperate and achieve the functions required by the user.

It is vital that grid security solutions provide flexible support for communication protection and enable control over authorization decisions, including the ability to restrict the delegation of rights in various ways. For the purposes of this research, communication and grid security is not a major concern, since the campus grid to be built lies within a single network and security domain and all the grid fabric resources are already protected by a firewall. The major issue will then be authentication, as the grid is used by a varied set of users, for example, students, lecturers and researchers; however, these authentication issues are handled by our grid middleware software.

2.2.7.3 Resource layer: sharing single resources

“The resource layer builds on the connectivity layer communication and authentication protocols to define protocols (and APIs and SDKs) for the secure negotiation, initiation, monitoring, control, accounting, and payment of sharing operation on individual resources. Resource layer implementations of these protocols require fabric layer functions to access and control load resources. Resource layer protocols are concerned entirely with individual resources and hence ignore issues of global state and atomic actions across distributed collections, such as the issues and concern of the collective layer”(Foster et al. 2001).

Two primary classes of resource layer protocols can be distinguished:

- **Information protocols,** which are used to acquire information about the structure and state of a resource, for example its current load of resources, and ignore usage policies such as the cost of resource usage, since all resources are locally owned by the NWU.
- **Management protocols;** “these are used to negotiate access to a shared resource, specifying resource requirements, for example advanced resource reservation and QoS, and the operations to be performed, such as process creation, or data access. Since management protocols are responsible for

initiating sharing relationships, they must serve as a policy application point, ensuring that the requested protocol operations are consistent with the policy under which the resource is to be shared” (Baltopoulos 2005).

Though a number of “protocols can be imagined, the resource and connectivity protocol layers form the neck of our hourglass model and as such should be limited to a small and focused set” (Foster & Kesselman 2004).

2.2.7.4 Collective layer: coordinating multiple resources

“The collective layer contains protocols and services not associated with any specific resource but instead capturing interactions across collections of resources. Because collective components build on the narrow resource and connectivity layer ‘neck’ in the protocol hourglass, they can implement a wide variety of sharing behaviors without placing a new requirement on the resources being shared, for example” (Foster & Kesselman 2004):

- **“Directory services** allow VO participants to discover the existence and/or properties of VO resources. A directory service may allow its users to enquire about resources by name and/or by attributes, such as type, availability, or load.”(Foster & Kesselman 2004)
- **“Co-allocation, scheduling and brokering services** allow VO participants to request the allocation of one or more resources for a specific purpose and the scheduling of tasks on the appropriate resources.
- **Monitoring and diagnostics services** support the monitoring of VO resources for failure, adversarial attack (‘intrusion detection’), overload and so forth.
- **Data replication services** support the management of VO storage resources to maximize data access performance with respect to metrics such as response time, reliability and cost.
- **Grid-embedded programming systems** enable familiar programming models to be used in grid environments, using various grid services to address resource discovery, security, resource allocation and other concerns.”(Foster et al. 2001)
- **Software discovery services** ascertain and choose the best software operation and execution platform.
- **Collaborating services** support the synchronized interchange of data within possibly large user communities.

The preceding paragraphs have described the grid architecture based on sound scientific and engineering principles and a comprehensive description has been given. It is, however, important to understand the grid user perspective in attempting to develop a system from which users will obtain benefit so that we can interest users in using our grid system. From an end user perspective, grids are anticipated to provide services such as:

- **Computational services**, which are primarily concerned with providing secure services for execution of applications that are individually or collectively executed on this distributed computing resource. This type of grid, known for providing computational services, is mostly known as a computational grid (Baker et al. 2002) Examples of this computational grid include NASA IPG (Johnston 1999), World Wide Grid and XSEDE (Anon n.d.).
- **Data services**, which aim to provide secure pervasive access of distributed data and the management of the data on this platform. “Scalable storage and access to data to be replicated, catalogued and stored in distributed locations creating an illusion of mass storage”(Baker et al. 2002) is provided for by this infrastructure. Data grids are made possible by combining computational grid services and data services, so as to accommodate data-intensive applications such as high-energy physics (Chervenak et al. 2003) and astronomical and biological services that generate large quantities of data (Antonioletti et al. 2005).
- **Application services**, which are concerned with the provision of remote software and the management of applications and libraries. This service is supported by computational and data services provided for by the grid (Foster et al. 2003).
- **Information services**, which are aimed at extracting and presenting data with meaning by use of computational, data and/or application services. The basic services provided are representation, storage, access, sharing and maintenance of information (Baker et al. 2002)
- **Knowledge services**, which are concerned with the acquisition, use, retrieval, publishing and maintenance of knowledge to achieve a user’s goal, solve problems and/or execute a decision. A good example is data mining for building new knowledge (Baker et al. 2002).

2.3 Middleware used in grids

Grid computing as a “federation of computer resources from multiple administrative domains to reach a common goal” (Nagaraju & Anitha 2012) is highly dependent on the middleware used to create the grid, which is the most prominent component of the architecture. Hardware and application programs are valuable components of the grid, but more focus is placed on the middleware, since it is the key management software layer of the grid computing platform (Priol 2005). Many versions of middleware software have been developed and are developing and this is a major area of focus on the grid, hence it has received most of the attention in grid evolution and development.

The grid middleware allows communicating processes to exchange messages through sockets and Message Passing Interfaces (MPI`s). These processes are mostly sustained by Application Programming Interfaces (API`s). “Grid computing middleware also allows the logical interconnection of various clusters which do not necessarily have the same services” (Georgiou et al. 2006). This functionality support engine of the grid provides for heterogeneity and can be executed in many different application-specific domains; however, most of these middleware programs support open grid service architecture (OGSA) (Schwiegelshohn et al. 2010). In light of this exposition it is important to note that many versions of middleware software can easily be used on many different platforms; for example, Sun Grid Engine (SGE), Unicore, Legion and Globus can be used in many different domains with different application-specific software.

At the core of the grid middleware is interoperability, which can be defined as follows: “Interoperability is the native ability of grids and grid technologies to interact directly via common open standards” (Blanke et al. 2009). In the article, “Grid interoperability: The interoperations cookbook” (Field & Schulz 2008) agree with the above definition by stating: “Interoperability is the ability to exchange information and to use what has been exchanged.” They further suggest an interoperability matrix used to depict the critical interfaces to be considered when dealing with many different infrastructures. The matrix basically focuses on major areas such as information services, security, job management and data management. To attain interoperability on the grid, the above-mentioned critical areas need to be studied carefully so as to understand the differences and then craft methods to enable interoperability.

The importance of interoperability can be clearly understood if we consider resource interoperability, which allows users from diverse VOs to share resources without demanding resource providers to install and maintain different grid middleware packages (Schwiegelshohn et al. 2010). This view is critical because it enables two or more VOs with different middleware stacks, such as Legion, Unicore and Globus, to share resources effectively without any installation overhead.

It is significant to note that grid interoperability can be fostered by using the same middleware across multiple VOs. However, the problem of grid interoperability cannot be eliminated because of different middleware choices by different entities (Field & Schulz 2008). Interoperability is relatively easy to achieve on small-scale grids such as campus grids, as they may have the same requirements and hence settle for the same middleware, but as the grid expands, the problem of interoperability becomes challenging to eliminate.

Middleware is critical in distributed systems, as it joins different parts of this system and creates a unified convenient environment. Therefore, at the center of the management software of the grid lies the grid middleware and it repurposes most if not all of the following components (Priol 2005):

- Communicating processes, which enable an interchange of messages, for example sockets, MPI and PVM. These processes are usually supported by APIs.
- Remote procedure calls, which relocate control between processes.
- “Objects within the distributed object-oriented architecture, enabling transparent access to objects within a distributed system, as each of these implements a programming language entity.
- Components using the component oriented architecture, which allow modular composition, configuration and deployment. These are also supported by programming language entities.
- Services within a service-oriented architecture that provide platform-independent functionality, as such services do not directly map to programming language entities.” (Priol 2005)

The above components are provided for and already built into grid middleware such as Globus Toolkit (GT), and they form a major part of the grid computing management. “Globus Toolkit is a community-based, open architecture, open source set of services

and software libraries that support Grids and Grid applications. It addresses issues of security, information discovery, resource management, data management, communication, fault detection, and portability” (Foster et al. 2003).

The GT is concerned with implementing grid protocols and as such bases its implementation on the assumption of existing appropriate software on fabric elements for local CPU scheduling, storage and network connectivity (Foster et al. 2003). The GT has been designed to use (principally) existing fabric components. However, if a vendor does not provide necessary fabric-layer functionality, the GT includes the missing functionality, which positions it as a reliable middleware service (Czajkowski et al. 2001).

GT uses the public-key based grid security infrastructure (GSI) protocol to achieve easy and secure grid connectivity and communication. GSI protocols are used for authentication (Foster & Kesselman 1999), communication protection and authorization. This security protocol extends the transport layer security (TLS) to address issues such as single sign-on, delegation, integration with existing local security solutions such as Kerberos and user-based trust relationships. GSI uses public-key-infrastructure certificates as a credential format for achieving most of the secure connectivity (Czajkowski et al. 2001).

GT further provides a set of protocols and standards for resource discovery, remote invocation of computation, monitoring and data transport (Foster et al. 2003). A grid resource access and management (GRAM) protocol, which is based on HTTP, is used for distribution of computational resources and for monitoring and regulating computation on these particular resources (Czajkowski et al. 1998). The “monitoring and discovery service (MDS) provides a uniform framework for discovery and accessing configuration and status information such as compute server configuration, network status and the capabilities and policies of services”. (Foster & Kesselman 2004)

Grid resource information protocol (GRIP) is used to define a standard resource information protocol and associated information model. An extended version of the FTP in grid environments is known as the GridFTP protocol. FTP is adopted as a base data transfer protocol because of its support for third-party transfers and because its separate control and data channels facilitate the implementation of sophisticated servers. To facilitate interoperability with other grid services, GridFTP services use the widely deployed grid security infrastructure for robust and flexible authentication,

integrity and confidentiality (Foster et al. 2003). GridFTP protocol is used as a, “management protocol for data access; extensions include the use of connectivity layer security protocols, partial file access and management of parallelism for high-speed transfers”. (Allcock et al. 2002)

“The Globus Toolkit further defines client-side C and Java APIs and SDKs for each of these protocols. Server-side SDKs and servers are also provided for each protocol, so as to facilitate the integration of various resources (computational, storage, network) into the grid. An important server side element of the overall toolkit is the ‘gatekeeper,’ which provides what is in essence a GSI-authenticated ‘inetd’ that speaks the GRAM protocol and can be used to dispatch local operations.”(Foster et al. 2001)

GT “software distribution provides only a limited number of collective layer capabilities: the DUROC resource co-allocation library is one example” (Czajkowski 1999). Many collective layer services and libraries that are compatible with the GT have already been discussed in the paragraphs above and most of these build on Globus connectivity and resource protocols (Foster et al. 2003).

The middleware briefly discussed above forms a major part of the grid structure and, “it enables an open set of standards and protocols known as Open Grid Services Architecture (OGSA)” (Bose et al. 2004). GT has been widely used by many researchers and has proven to be a robust grid middleware, hence it is considered and discussed ahead of a variety of other available grid middleware (Foster & Kesselman 1999).

2.4 Performance on the grid

Performance measurement on the grid is very important, as it encourages users of the grid to assess the performance benefits of using the grid computing infrastructure. Although grid middleware has been developed extensively over the past years, grid users still struggle to understand the grid because of the lack of performance measurements. A major challenge in evaluating grid performance is what performance metrics are to be used. This is caused by the high complexity and dynamic nature of the grid (Stoelwinder 2004).

Benchmarking is a widely used standard for testing and evaluating the performance of a grid system. Grid benchmarks enable users to evaluate the grid system’s capabilities against other similar infrastructure and help grid users to improve on the performance of the grid. “NAS Grid Benchmarks (NGB) have been proposed as grid

benchmarks based on widely used NAS Parallel Benchmarks.” (Frumkin & Wijngaart 2002)

Major focus is normally placed on a) **the turnaround time**, which is the time between starting a job/task and obtaining the resulting data, and b) **throughput**, which is the submission capacity possible without negatively impacting the turnaround time (Frumkin & Wijngaart 2002). CPU utilization is another way of determining grid exploitation by determining the total proportion of consumed CPU cycles and the obtainable computational resources (Snaveley et al. 2003).

The research will limit the grid performance measurements to, turnaround time and system utilization as the campus grid is implemented. The performance metrics selected will assist researchers and all campus grid users to make certain scheduling decisions and select an appropriate system for use in their research.

2.5 Classification of grids

Grids are often classified as either **compute grids**, which primarily focus on the shared use of computational resources to achieve high-throughput computing, or **data grids**, which place their focus on supporting the federation, integration and mining of data resources. These distinctions also determine the types of hardware needed to build the different grids. As such, a computational grid is expected to be built out of high-speed computers, as opposed to a data grid, which is built from a number of high-capacity storage computing devices (Srinivasan & Treadwell 2005).

The classification of grids is often made according to the grid functions, but this classification is not always agreed upon. Stockinger (2007) attempts to express the main ideas behind grid classification. People generally distinguish between pure Computational Grids and enhanced Data Grids. There exist other grid distinctions as outlined by (Stockinger 2007), and these are:

- “Collaborative grids, involve multiple organizations (institutions) and individuals, security domains, protocols and discovery mechanisms. It is important to note that they are widely distributed, virtual organizations (VO), and are enabled by service agreements and commercial partnerships and as a business model they increase overall revenue.
- Enterprise grids, are usually technically more complex, as collaborative grids encompass the complete life cycle of service deployment, provision management and decommissioning. These production grids of major data

centers are enabled by virtualization of enterprise resources and applications, aggregation and centralized management and as a business model they reduce the total cost of ownership.

- Cluster grids. Aimed at high-performance/throughput computing, these grids are mostly workload scheduling environments. They tend to be static rather than dynamic as collaborative and enterprise grids. Their services are mostly generic in nature, such as job processing and they normally provide the same service all the time. However, if clusters themselves are not connected to other clusters, they do not form a grid.

2.6 Campus grids

The field of science and engineering has a vast need for large numbers of CPUs so as to solve compute-intensive problems encountered in scientific research and engineering. This enormous need for more computational power has invoked an implementation of campus grids in many academic institutions as a response to the needs of the academic community. “Campus grids have the unique property that consumers of the system must always defer to the needs of the resource providers. For example, if a desktop computer that forms (Moretti et al. 2010) part of the campus grid is being used by a visiting job because it is idle, the visiting job will be preempted as soon as the desktop is used locally.

Campus grids are used, “by multiple teams in one organization sharing several cluster grids and these are mostly single-site to enterprise-wide grids. Campus grids are mostly used by engineers and scientists, associations, corporations, environmental, training and educational organizations. They are dedicated to on-demand computing, high-throughput computing, data-intensive computing, collaborative computing and supercomputing” (Gentzsch 2002) Campus grids typically contain multiple computer clusters that are independently administered (Weitzel 2011).

Several campus grids already exist; examples are the University of Virginia campus grid (Humphrey & Wasson 2005), University of Oxford campus grid (Wallom & Trefethen 2006) and Purdue University campus grid (Smith et al. 2008). It is against this background that the researcher seeks to utilize an existing campus grid on the South African national grid to test the performance of a campus grid and consider the feasibility of designing and building a campus grid for the NWU’s Vaal Triangle Campus (NWU-Vaal).

2.7 Summary

The literature review informed the study on the computing infrastructures being explored and provided an understanding of the different computing infrastructures, their history, operation, as well as the benefits that can be obtained from previous studies in the field of computing. The chapter gave a solid foundation regarding the nature and operation of the computing systems of multicore servers and computational grids, being explored by the study. The literature study clearly provided an understanding of the field of computing the researcher intended to conduct the study in. Critical concepts of high-end computing as well as distributed computing were discussed as the study seeks to address the needs of compute-intensive applications at the MuST research group. The researcher explored microprocessors, uncore, multicore, cloud and grid computing infrastructures, in order to establish a clear understanding of the field the study falls into. The work done in this chapter ushers the study into the discussions that follow in the research methodology chapter, as the study attempts to meet its objectives.

3 Research methodology

3.1 Introduction

This chapter explores the research design as well as methodology used in this study to fulfil the study objectives of the research. The researcher briefly investigates different research methodologies as well as research paradigms that can be used in problem solving. The chapter further discusses research methods that can be used in conducting scientific research, research methods such as quantitative, qualitative as well as mixed methods. The chapter then deals with the research design and methodology selected for this study, which is a partially mixed concurrent dominant status research design. Lastly the chapter concludes with a summary on research methodology and lays a foundation for the analysis of results obtained during the experiment as well as the narration of the observed workflows in the experiment set-up and execution.

3.2 Research

This section explains the nature of research as seen by a number of scholars. It goes on to outline research paradigms, followed by a brief introduction to quantitative and qualitative methods, as well as mixed methods.

Burrell and Morgan (1979) note that research commences when researchers are conscious of their domain of research, aware of the boundaries that define their perspective, and are able to undertake an intellectual crossing to other research domains.

Olivier (2009) states: “Research is an investigation to discover facts”. Note that this definition does not constraint the researcher to the discovery facts that have not previously been discovered by someone else. Research therefore does not have to be original, but then it does not really make sense to do a lot of work to ‘discover’ information available from easily accessible sources. Furthermore, on a practical note, if one is doing the research for academic purposes to get some qualification, “the point to be proven is that you are capable of doing research.”

There are three primary strategies for conducting research as stated by (Olivier 2009), and these are:

- Compile information on some topic, where the bits and pieces have already been discovered (often by other researchers), but have not yet been integrated into a single coherent body of knowledge.
- Solve a problem for which no known (or apparent) solution exists.
- Look with new eyes at existing knowledge (or standard ways of doing things).
In other words, find a better solution for a problem that has previously been solved.”

These above-mentioned strategies are integrated into various research methods and used to conduct effective research in a bid to discover and suggest solutions to problems identified by the researcher (Olivier 2009).

The diagram below shows the iterative process of rational problem-solving that is usually undertaken in problem-solving by employing research as a problem-solving tool.

Research can be viewed as a rational problem-solving process where a problem area is discovered, thereby igniting an interest in trying to solve the problem. After a researcher discovers a problem area, the focus shifts to ascertaining the present situation and describing the unsolved aspect of the problem. All this is achieved

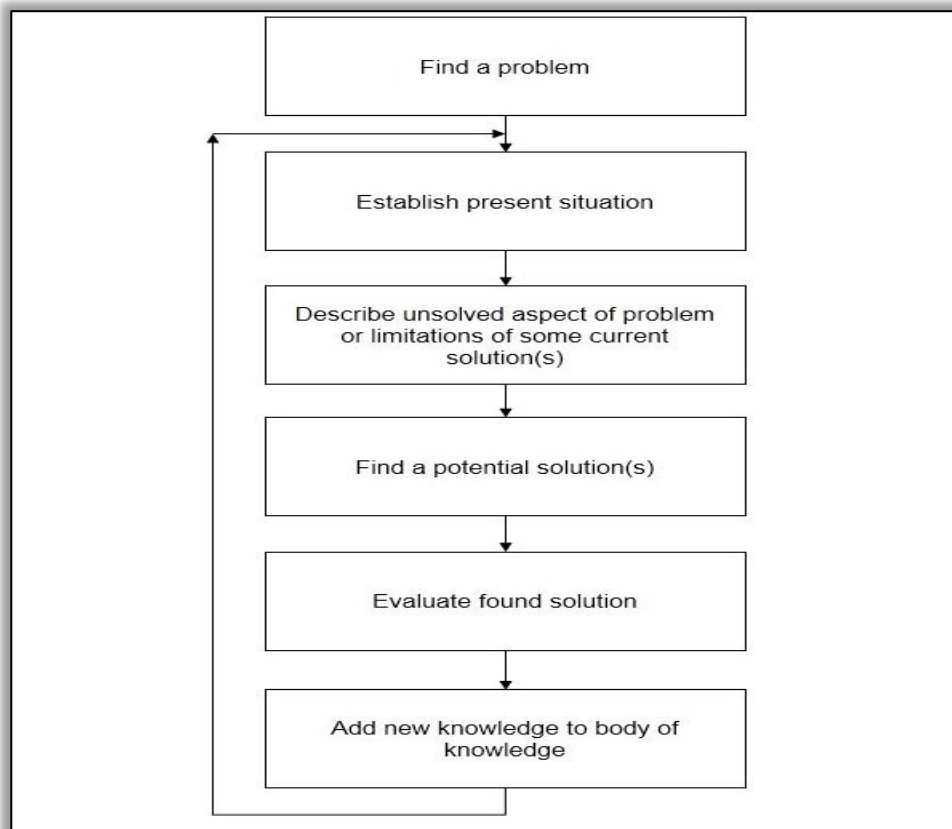


Figure 3-1: A variation of the rational problem-solving process (Olivier 2009)

through literature study. To find a solution to the identified problem, a researcher then chooses a method suitable for the particular research from various research methods available. The solution obtained from the application of the research method of choice is evaluated through critical analysis and the knowledge attained may then be added to the body of knowledge. It is important to note that problem-solving is an iterative process by nature (Olivier 2009).

3.2.1 Research paradigms

This section gives a brief outline of research paradigms and it further asserts the paradigm this research falls into. De Villiers (2005) describes a constructive framework called a paradigm as “a primary point of departure” for research. A research paradigm is alternatively defined by (Goede et al. 2011) as “a description of a research methodology according to a specific set of assumptions.” Kuhn (1976) views “a paradigm as a general concept including a group of researchers having a common education and an agreement on examples of high quality research or thinking.” Research that shares research paradigms conforms to research practices that are committed to the same rules and standards.

As shown in the diagram below, (Burrell & Morgan 1979) the model of sociological paradigms comprises four paradigms (humanist, structuralist, interpretivist and functionalist), according to the sociologies of regulation and radical change. The following section gives a brief overview of the humanist, structuralist and interpretivist paradigms and a more detailed explanation of the functionalist paradigm.

Research methodologies are often placed in the horizontal dimension between subjective research and objective research. Considering the second dimension of sociological differences, one can then populate four research paradigms, which generally encompass all the research methods that can be used to conduct research systematically. Each paradigm provides guidelines for viewing a problem based on different meta-theoretical assumptions (Burrell & Morgan 1979).

Two paradigms widely used in information systems research are positivism (also known as structuralism) and interpretivism (Pather & Remenyi 2004).

Burrell and Morgan (1979) view *structuralism* (i.e. positivism) as the traditional research paradigm used by researchers to direct their research. The philosophy of

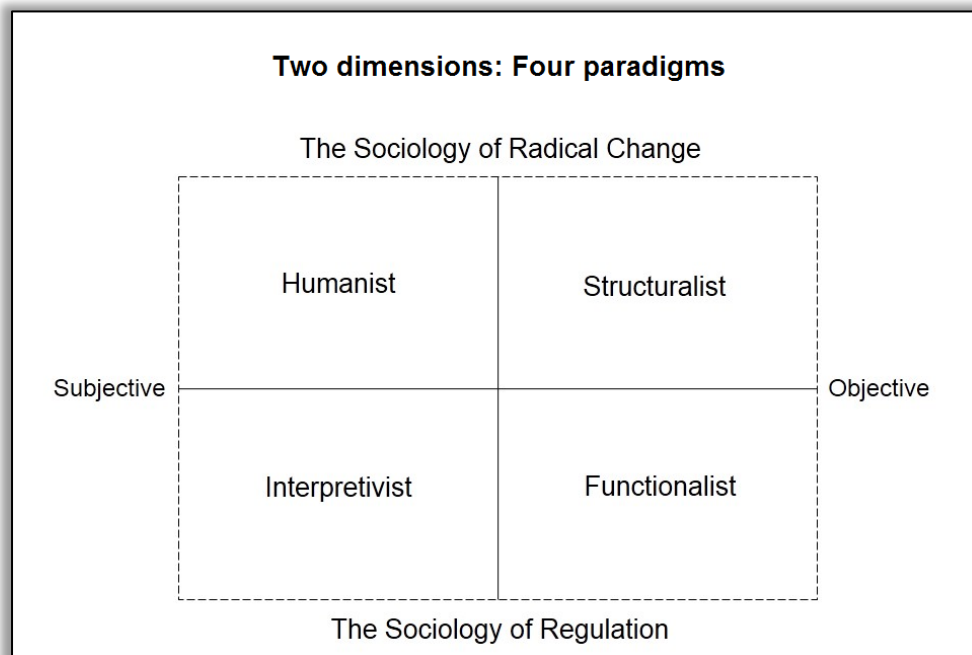


Figure 3-2: Two dimensions: Four paradigms (Adapted from Burrell & Morgan 1979)

positivism adopts the empiricist view that knowledge stems from human experience and observation. The purpose of positivistic research is often to measure the effect of specific variables in a situation objectively (Goede et al. 2011). The perspectives important to this study are identified by Giddens and Giddens (1974) as:

- “Science constitutes a framework by which any form of knowledge can be determined.
- There is a fundamental distinction between fact and value. Science deals with facts, while values belong to an entirely different order of discourse beyond the remit of science.”

Positivistic research, from the methodology viewpoint, is conducted in situations where well-organized experiments are executed by examining well-defined and quantifiable environmental variables in a closed system. In a research project the researcher will define a scenario and use experimental procedures to determine results. Scientific and statistical methods are used to obtain results that are considered, in order to suggest a solution to the problem being investigated, therefore this type of research is quantitative (Goede & de Villiers 2003). In this paradigm, the objectivity of the research is high and the control is high, as usually seen with quantitative research methods (Burrell & Morgan 1979).

The *interpretivist* paradigm integrates human interest into the research and researchers have to interpret elements of the study. The interpretivist paradigm adopts a method that is implicit rather than explicit. Interpretivists tend to be nominalist, anti-

positivist, voluntarist and ideographic (Burrell & Morgan 1979). Interpretivists believe that *we can only know what we experience* (Patton 2005) and there is no objective experience (Merriam & Tisdell 2015). The aim of interpretivists is providing an explanation of the phenomena by interpreting the data and understanding the information that is derived (Goede & de Villiers 2003). Research in the interpretivists paradigm uses qualitative research methods to gather and analyze data; the subjectivity of research in this paradigm is high and the control is low (Burrell & Morgan 1979).

The *humanist paradigm* focuses on the sociology of change, modes of domination, emancipation and deprivation, and potentially takes a subjectivist point of view. Humanists are inclined to be nominalist, anti-positivist, voluntarist, and ideographic in nature. It is believed that people are confined in organizations made and maintained by society. A pathway has to be created that enables humans to break free from existing social patterns, which trap them in social organization, and then to alter the social world by adapting modes of cognition and consciousness. The humanist acknowledges the ontological position of the social world and has some attributes in common with the interpretivist. In this paradigm the subjectivity and control of the research is high and the focus is on criticism of existing systems, with no focus on proposing a solution to the problem (Burrell & Morgan 1979).

The *functionalist* paradigm originated in the sociology of regulation and is mainly used to study organizations and behavior of academic sociology from an objectivist point of view. The stance here is predominantly pragmatic and focuses on understanding society from a problem-oriented approach. The major aim here is gaining constructive knowledge so as to provide practical solutions to solve practical problems. This paradigm uses either quantitative or qualitative research methods or mixed methods (which combine both quantitative and qualitative methods) as a researcher tries to solve a problem in a real-life situation (Burrell & Morgan 1979).

3.2.2 Qualitative and quantitative research

The terms quantitative and qualitative research are widely used in research whenever research methods are discussed. However, confusion sometimes arises from the fact that these terms may be used to distinguish the nature of data (Myers 1997) and sometimes they distinguish the research approach (Bryman 2006). It is important that

we clearly distinguish these two research approaches in this study so that it becomes clear at mention what they are.

To distinguish between the research approaches, “quantitative purists believe that the observer should be objective and separate from entries that are observed. They should remain emotionally detached and uninvolved and this is in accordance to the positivism paradigm. Qualitative purists reject this and argue for idealism, humanism, and hermeneutics. They are characterized by a dislike of detached writing and prefer detailed and rich description” (Johnson & Onwuegbuzie 2004).

The distinction between qualitative and quantitative research in terms of the nature of data is captured as follows: When *natural phenomena* with quantifiable attributes are studied, then quantitative research uses hypothesis formulation, experiments, observation and statistical analyses to obtain correlations and make predictions with *numerical data*. “Qualitative research, on the other hand, uses data from interviews, documents, observations and stories, usually to understand and explain *social phenomena*. In qualitative research the emphasis is on processes and meaning that cannot always be measured in terms of quantity” (Goede et al. 2011) of attributes (e.g. amount or frequency). Recorded data include words, images and sound, which are analyzed by means of qualitative techniques. (Myers 1997)

3.2.3 Mixed methods research

“Since the early 1960s mixed methods research has grown more popular in many disciplines, including information systems” (Leech & Onwuegbuzie 2009). The term *mixed methods research* refers to “research that involves collecting, analysing, and interpreting quantitative data and qualitative data in a single study or a series of studies that investigate the same underlying phenomenon” (Johnson & Onwuegbuzie 2004). “A study that combines quantitative and qualitative techniques to any degree in the study no longer utilizes a monomethod design, but mixed methods” (Leech & Onwuegbuzie 2009). Monomethods involve the exclusive use of either quantitative or qualitative research techniques in the study. A mixed methods research design uses either, “a fully mixed design or a partially mixed design in the study. When undertaking a mixed methods study, the researcher uses qualitative research methods for one part of a research study and quantitative methods for the other part of the research study. Thus, qualitative and quantitative research studies are conducted either concurrently or sequentially”. (Leech & Onwuegbuzie 2009)

This research uses the term *quantitative* to refer to experiments and numerical methods and *qualitative* to refer to observations arising from the collection and use of quantitative data. Hence, this study follows a mixed method research design and this will be expanded on in the research method section.

3.3 Research methodology

This section outlines how the research was conducted, it discusses the research design and research method, and describes the two systems (*computational grid and multicore server*) that are used in this comparative study. Furthermore, it outlines the process used in data analysis in comparison of the two systems and then discusses the rationale behind the determination of choices and trade-offs for performance and system management benefits of using a computational grid against a multicore server. The organization of the rest of this section follows the way in which the research was carried out and is depicted in Figure 3-3 below.

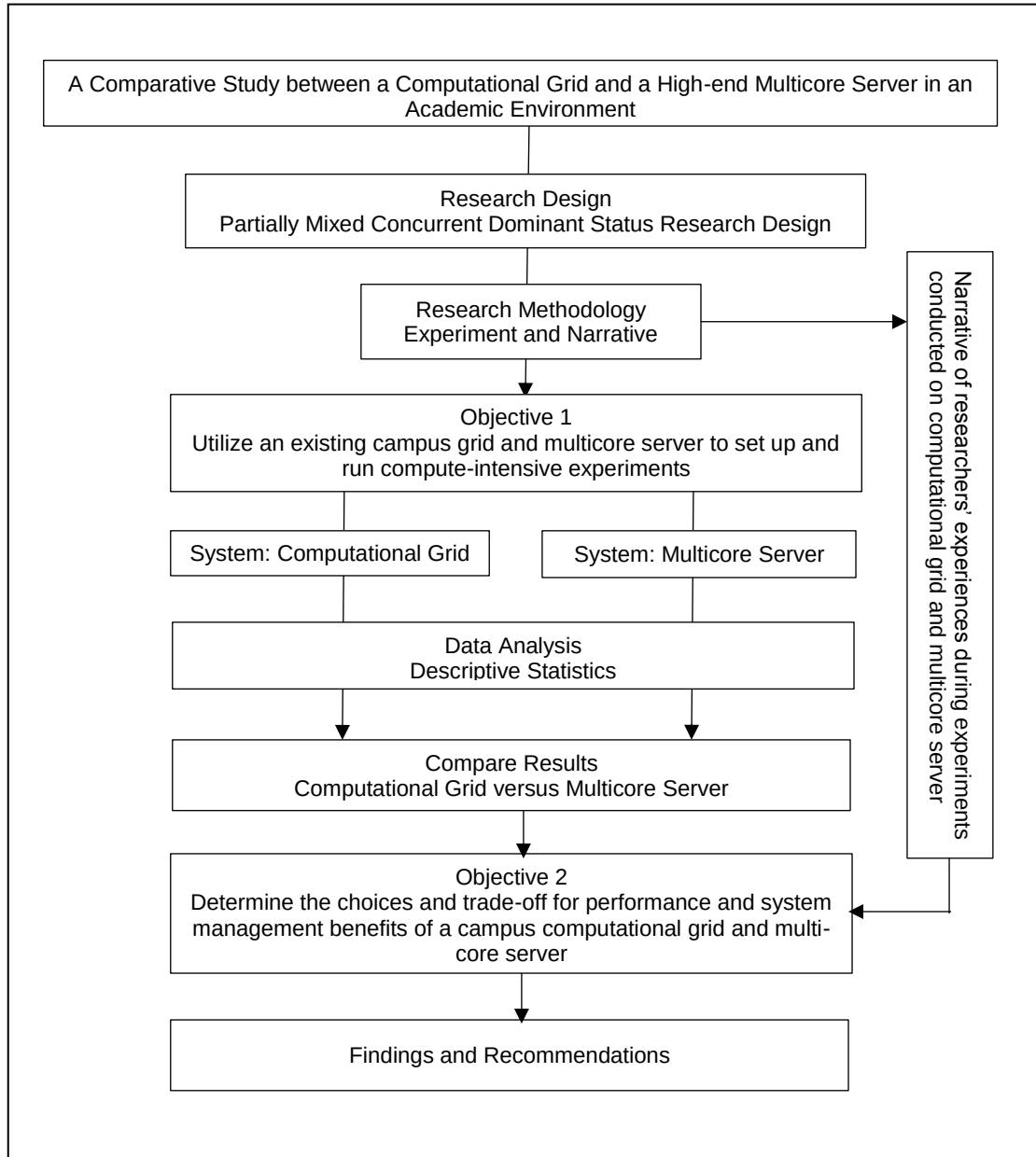


Figure 3-3: Overview of the research methodology followed in this research

3.3.1 Research design

This research uses the functionalist paradigm and takes a pragmatic view in trying to understand the problem of underutilization of computing resources in **our academic** institution against the increase in computing performance need that grips academic researchers. In this study the computing performance capability of a computational grid versus a multicore server was investigated, with the aim to compare their performance and workflows in order to determine if a campus computational grid could be used in trying to alleviate the high demand for computational power on the local multicore server used by academic researchers.

This research is no exception, as it uses mixed methods of combining quantitative methods in the form of experiments and qualitative methods in the form of a narrative gathered from using the computational grid as well as the multicore server. “Mixed methods research generally represents research that involves collecting, analyzing and interpreting quantitative and qualitative data in a single study or a series of studies that investigate the same underlying phenomenon”. (Johnson & Onwuegbuzie 2004) “The major difference between partially mixed methods and fully mixed methods is that whereas fully mixed methods involve the mixing of quantitative and qualitative techniques within one or more stages of the research process or across these stages, with partially mixed methods, the quantitative and qualitative phases are not mixed within or across stages. Instead, with partially mixed methods, both the quantitative and qualitative elements are conducted either concurrently or sequentially in their entirety before being mixed at the data interpretation stage”. (Leech & Onwuegbuzie 2009) In this research we follow a partially mixed concurrent dominant status research design, which is explained in the section below.

3.3.1.1 Partially mixed concurrent dominant status research design

The research question in this study seeks to discover if a campus computational grid can be used in alleviating the pressure of computation on the local multicore server while fully utilizing the several computing resources provided by the university (NWU). The multi-mode approach used in this research is deemed viable for the purposes of: (i) experiment configuration and execution of both the computational grid and the multicore server so as to compare their performance, as well as (ii) analysis and reflection of the researcher’s experience in successfully setting up and executing this experiment on both systems.

(Leech & Onwuegbuzie 2009) conceptualize a *three-dimensional typology of mixed methods designs* to apply to this study and then proposes, “a function of the following three dimensions: (i) level of mixing (partially mixed versus fully mixed); (ii) time orientation (concurrent versus sequential), and (iii) emphasis of approaches (equal status versus dominant status)”.(Leech & Onwuegbuzie 2009) The *level of mixing* refers to the balance between quantitative and qualitative research methods. *Time orientation* captures the stages in which the quantitative and qualitative methods are utilized, either at the same time (concurrent) or successively (sequential). *Emphasis on the approaches* deals with the levels of priority, e.g. whether the quantitative

method has a higher priority over the qualitative one or whether they bear the same weight.

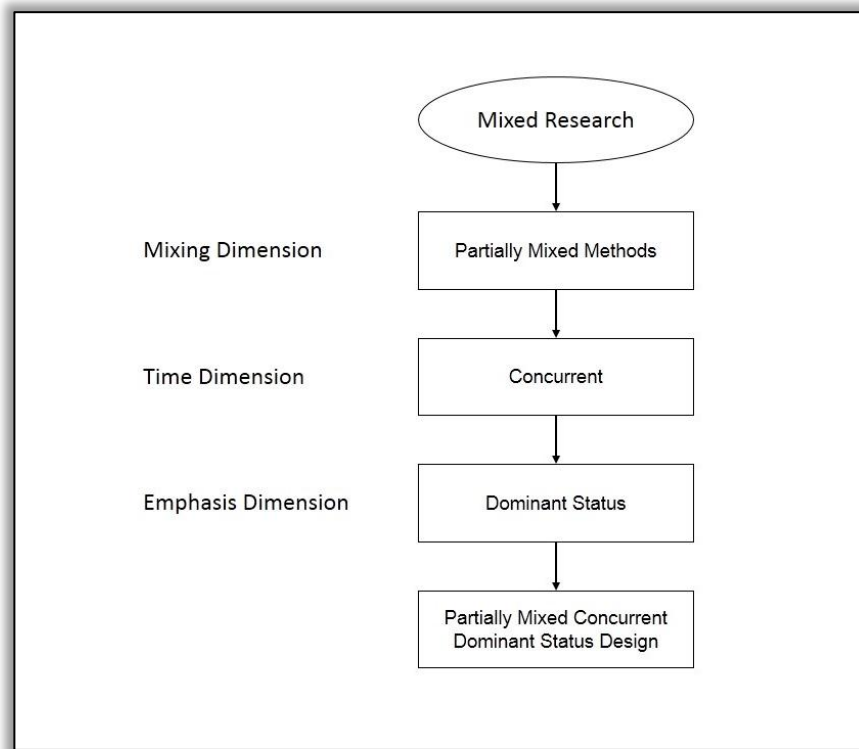


Figure 3-4: Typology of mixed research (Adapted from Leech & Onwuegbuzie 2009)

The diagram above gives an overview of the approach this study followed in dealing with the mixing of methods. Partially mixed methods are selected as the mixing dimension in this research. This selection is justified by the fact that this research focuses mainly on computing performance comparison of the two systems, which is achieved by a quantitative research method of conducting experiments on both systems, while observations are made in the process of experiment configuration and execution. The time dimension chosen is concurrent in that the stage of conducting experiments cannot be separated from the observation stage, as both happen at the same time. The choice of a partially mixed concurrent dominant status design in the emphasis dimension guides the researcher in grounding the research in a more quantitative method of experiments in order to compare the performance of the two systems being tested, with little emphasis on the qualitative method of observing the lessons learnt in the quantitative process.

3.3.2 Research methods of this study

The research methods followed in this study are of a quantitative and qualitative nature. The quantitative method was achieved by performing an experiment and the qualitative method by the observations derived from using the computational grid against the multicore server in carrying out the experiment. Firstly we turn our focus to the quantitative research method as we seek to outline the process carried out using this method.

“Experiments are performed by investigators in virtually all fields of inquiry, usually to discover something about a particular process or system. Literally, an experiment is a test. More formally, we can define an experiment as a test or series of tests in which purposeful changes are made to the input variables of a process or system so that we may observe and identify the reasons for changes that may be observed in the output response”. (Montgomery 2008)

(Olivier 2009) asserts that an experiment is conducted with the following goals in mind (in increasing order of ambitiousness):

- “To see if one can find something interesting (in other words, an exploratory experiment)
- To ‘test’ a theory
- To prove a theory.”

The first goal outlined above, “to see if one can find something interesting (in other words, an explanatory experiment”) was chosen for this research. This choice by the researcher is informed by the particular experiment (a real-world scenario) which was not very structured, as the researcher had to experiment with a specific idea of comparing the real-life performance of two existent systems (computational grid and multicore server) and see what could be learnt from this process.

In the second and third case one will conduct a limited experiment to see whether the theory holds for some specific cases and sometimes to prove whether the theory is correct, to refine the theory and/or to justify a full-scale experiment.

In following the first goal, we set up an experiment for testing the performance of a computational grid against that of a multicore server in order to compare the turn-around time of the two systems in a case of pure performance.

3.3.2.1 Experiment set-up

A compute-intensive application for support vector machine training was identified and the computational data and a number of scripts that execute the data were adopted from an experiment by some researchers in the MuST research group at the NWU. This research, however, uses the same experiment for a different purpose of comparing the performance of a computational grid versus a multicore server and the computational times of each single run are recorded.

The dataset used in the experiment is described in section 3 of the article, “Language identification of individual words with joint sequence models” (Giwa & Davel 2014). This is a real-life research scenario with the actual experiment and data used in the discipline of human language technologies (HLT). However, in this case it was used for a different purpose, namely measuring the time of computation with varying execution environment parameters.

The experiment is set up in similar ways on both the computational grid and multicore server as follows:

3.3.2.1.1 Computational grid

- Choice of a grid site on the South African grid
- Data staged on the metadata server
- Application ported to CODE-RADE
- Libsvm library compiled
- Perl and Python modules compiled
- Job description language (Jdl) files authored to execute the scripts on the grid.
- gLibrary installed for timing data collection.

3.3.2.1.2 Multicore server

- Data copied to the server storage
- Scripts that run the experiment arranged in several folders
- Libsvm library complied
- Python and Perl modules complied
- Changes to scripts made in order to adapt them for parallel processing.
- Experiment structured and organized in terms of input and output files from the experiment runs

- Bash script written to repeat the run and return turnaround time, CPU percentage, file system inputs (fs-inputs) and file system outputs (fs-outputs), as well as memory information
- Experiment executed with different dataset sizes and results collected.

3.3.2.2 Process narrative

We further compare the two systems using lessons learnt in the process of conducting this experiment. It is important to note that our experiment is a real-world scenario and hence we do not focus on the system standardization to ensure an absolutely fair comparison, but we focus on determining what the trade-offs between two existing systems in a real-world scenario are.

The researcher notes and records advantages and disadvantages of one system over another as the research experimentation process unfolds. These notes will guide the researcher in outlining his experience on the use of both systems.

3.3.3 Systems being compared

A brief description of the two platforms is given (the computational grid and the multicore server), to provide the reader with a background of the environment in which the study was conducted.

3.3.3.1 Computational grid

A selection of sites of the South African grid was used for this study in general. Specific sites were used for performance timing of experiments. The sites of the University of Johannesburg (UJ) and Center for High Performance Computing (CHPC) were used to run a few experiments and timing results were collected for the purpose of comparing them with the local server at the NWU-Vaal. The grid site specifications are as follows:

UJ CE: glite-ce.grid.uj.ac.za:8443/cream-pbs-sagrid

- LogicalCPU	208
- FreeCPUs	183
- PhysicalCPU	52
- CPUVendor	amd
- ClockSpeed	2600

CHPC CE: grid-ce.chpc.ac.za:8443/cream-pbs-sagrid

- LogicalCPU	4032
- FreeCPUs	1392
- PhysicalCPU	168
- CPUVendor	Intel
- ClockSpeed	2800

The command to get the above information on the grid is: `lcg-info --list-ce --vo sagrid --attrs 'LogicalCPU,FreeCPUs,PhysicalCPU,CPUVendor,ClockSpeed'`

3.3.3.2 Multicore server

The specifications of the server machine at the NWU-Vaal, used in this comparison are as follows:

- HP ProLiant ML360 G6 (Tower)
- 6 CPUs x 2 sockets = 12 CPUs
- 24 logical cores or 24 threads
- 96GB RAM
- Intel Xeon X5690 Processor (Anon n.d.)
- 12M Cache
- 3.465 GHz Processor Base Frequency.

The server specified above was released to the market in 2011 and has since been retired from the HP product offerings, but for the purposes of this comparison it was used because it was available locally at NWU-Vaal. This server is used for processing compute-intensive research by the MuST research group and it is fully controlled by server administrators on the NWU-VTC.

3.3.4 Data analysis

We analyze the data obtained from running the experiment on the computational grid and multicore server. Descriptive statistics are used – the analysis of data so as to summarize data in a meaningful way so that patterns might emerge from the data. Inferential statistics will be used to make inferences about similar data that may need to utilize the systems used in this research. These two statistical methods will be used in analyzing this study so as to draw conclusions based on this analysis.

3.3.5 Comparing computational grid versus multicore server

A comparison of the two systems will be outlined from the data analyzed in the section above, as well as the observation on the use of both systems.

3.3.6 Determination of the choices and trade-offs of using the two systems

We discuss issues that should guide users who intend to choose between the two systems when conducting similar experiments.

3.3.7 Finding and recommendations

We report on our findings in chapter 4 and make recommendations in chapter 5 for researchers interested in extending this study in future.

3.3.8 Summary

This chapter on research design and methodology exposed the process taken in fulfilling the study objectives of this research. The researcher displayed an understanding of fundamental concepts in research and this guided the study in choosing an appropriate research design and methodology. Critical concepts of the study have been explored in this chapter and a holistic view of how the research is to be conducted has been given. It should be noted that the discussions on partially mixed concurrent dominant status research design and the methodology that uses a combination of an experiment and a narrative to make findings and recommendations, formed a backbone to the study. The chapter that follows, presents a critical analysis of the results obtained from the experiment as well as the narrative stemming from the current chapter, in order to draw inferences and recommendations based on the results obtained as well as the experience of the researcher when conducting this study.

4 Results and analysis

4.1 Introduction

The current chapter on results and analysis, begins with outlining the experiments conducted in this study, and these are speech technology applications executed on the multicore server as well as the computational grid. It then reports on the results obtained as a result of the experiments set-up and executed in the research methodology chapter. The researcher uses descriptive statistics as a tool to analyse the results obtained and visual cues in the form of graphs of this data are used to make it easy to analyse these results. It further gives a narrative on the workflows of using the two different computing platforms. The narrative given is as a result of the observations made when interacting with the multicore server as well as the computational grid during the process of set-up and execution of experiments conducted in this study. The chapter concludes with a summary of the analysis done in this chapter and lays a foundation for the discussions that follow in the next chapter.

4.2 Experiments

Two applications were executed with several options and their job completion times recorded.

4.2.1 Support Vector Machine (SVM) application

The experiment used as test case for this research deals with language identification of individual words using joint sequence models. It classifies words for four South African languages: Sesotho, isiZulu, Afrikaans and English. This classification is done using 15 000 unique words per language and the training data are obtained from the NCHLT-inlang dictionaries (Giwa & Davel 2014).

The training is done on partitions of the data named with labels such as 2K, 6K, 12K. In this research we measured the turnaround time of multiple runs of the experiment for each partition using the local server and the computational grid. The data sets (partitions) labelled 2K, 6K and 12K are of size 190MB, 361MB and 602MB respectively and they represent:

- 2K: input based on 2 000 words per language for the four languages
- 6K: input based on 6 000 words per language for the four languages
- 12K: input based on 12 000 words per language for the four languages

The experiment was repeated at least five times for all the datasets. The results of job completion times were recorded for the two systems and attached in the addendum. In the section below we simply refer to the data by the labels 2K, 6K and 12K.

4.2.2 SVM Data analysis

We use descriptive statistics to analyze the data obtained from the experiment so as to answer objective 1: Utilize an existing campus grid on the South African grid federation and local multicore server to set up and run compute-intensive experiments. The averages and standard deviation of five repeated experiments for each dataset are used in the analysis of the experiment that is described below.

All the job completion time data obtained from the experiment were recorded and placed in the addendum of this document. The following coding is used in the analysis:

- System (X1):
1 represents the **grid computer**.
-1 represents the **local server**.
- Number of cores (X2):
1 represents **max cores** used on both the grid (48 cores) and local server.
-1 represents both the grid and local server restricted to use **only eight cores**.
- Dataset label (X3):
1 represents the 12K dataset.
0 represents the 6K dataset
-1 represents the 2K dataset

4.2.3 Overall system performance (real-life scenario)

The table below shows 12 experiments each consisting of five repetitions for each dataset (2K, 6K and 12K) with a varied number of cores as noted above. The averages and standard deviation were calculated and recorded for the local server and the grid.

Experiment	System (X1)	Number of cores (X2)	Dataset size (X3)	Average (s)	Standard Deviation (s)
1	1	1	-1	638.87	452.440
2	1	1	0	945.51	422.658
3	1	1	1	16124.91	424.230
4	1	-1	-1	229.28	1.795
5	1	-1	0	738.14	40.352
6	1	-1	1	7265.82	316.366
7	-1	-1	-1	170.78	3.266
8	-1	-1	0	1399.12	12.301
9	-1	-1	1	6227.58	86.336
10	-1	1	-1	106.92	6.992
11	-1	1	0	956.73	8.471

12	-1	1	1	4268.31	37.746
----	----	---	---	---------	--------

Table 2: Averages of all experiments (real-life scenario)

The table above is used to determine the overall performance of the local server against the grid and this analysis is depicted in the tables below by using the overall averages and standard deviation for the two systems.

Average	Seconds	Minutes	Hours
X1= +1 (Grid)	4323.7554	72.06259	1.201043
X1= -1 (Local Server)	2188.2397	36.470661	0.607844
Effect	2135.5157	35.591929	0.593199

Table 3: Overall performance of the grid vs local server

The table above shows that the local server outperformed the grid by 49% on this particular experiment in a real-world scenario.

The variation of the overall system performance on the grid against the local server is shown by the table below.

Standard deviation	Seconds	Minutes	Hours
X1= +1 (Grid)	276.30731	4.60512	0.07675
X1= -1 (Local server)	25.85212	0.43087	0.00718
Effect	250.45518	4.17425	0.06957

Table 4: Variance on the overall performance of the grid vs local server

The table above shows that the effect in variation on the grid is approximately 10 times more than the local server in terms of the overall performance of the system.

The graphs that follow give a graphical representation on the overall performance of the local server against the grid.

Dataset	Local server, 8C	Local server, Max	Grid, 8C	Grid, Max
2K	170.78	106.92	229.28	638.87
6K	1399.12	956.73	738.14	945.51
12K	6227.58	4268.31	7265.82	16124.91

Table 5: Averages of the grid vs local server for graph plotting

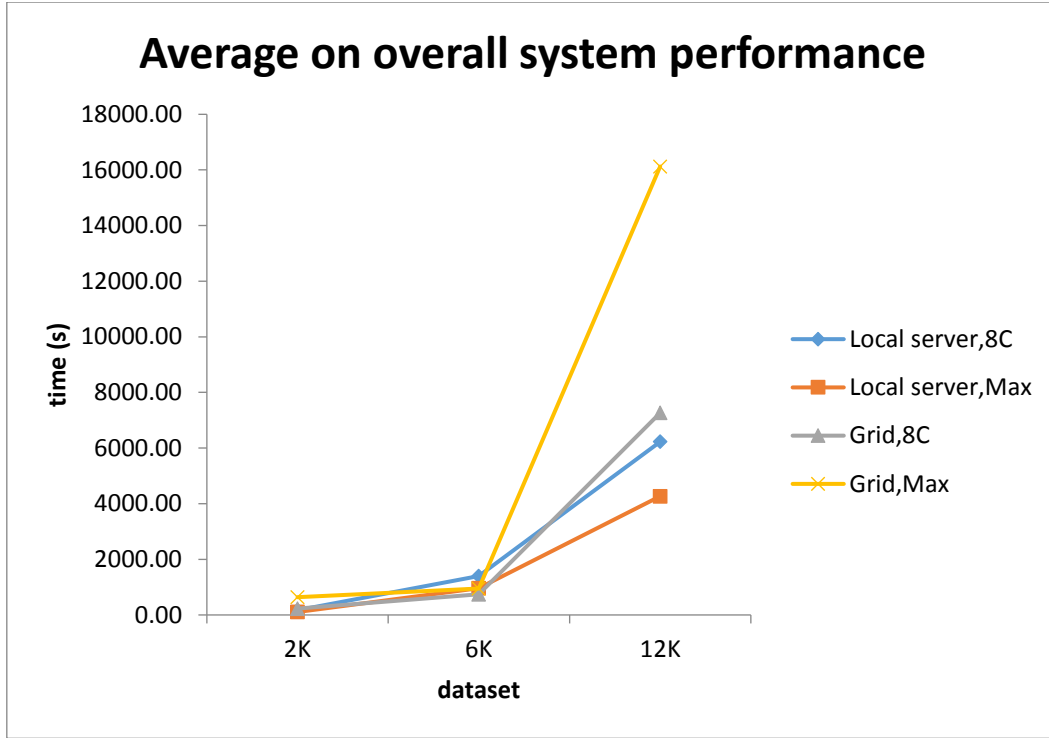


Figure 4-1: Overall performance of both systems

The figure above shows that both the grid and local server performed relatively well on the smaller datasets of 2K and 6K, with the local server outperforming the grid on the 2K dataset. Interestingly the grid outperformed the local server on the 6K dataset, showing that the local server does not always outperform the grid. The local server performed four times better than the grid on the bigger dataset of 12K with maximum available 48 cores on the grid and 22 cores on the local server. In this experiment we observe that the grid, when restricted to eight cores, performs better than it does with 48 cores. The local server, however, on the contrary performs differently in that its performance increases as one fully utilizes the server. We conclude that on average the local server overall outperforms the grid in a real-life scenario used in this research, regardless of the better performance observed with the 6K dataset on the grid.

The table below shows standard deviations used to represent the variance of the overall system performance graphically.

Dataset	Local server, 8C	Local server, Max	Grid, 8C	Grid, Max
2K	3.27	6.99	1.80	452.44
6K	12.30	8.47	40.35	422.66
12K	86.34	37.75	316.37	424.23

Table 6: Standard deviation values for the overall experiment on the grid vs local server



Figure 4-2: Variation in overall system performance

The figure above shows the variation in performance of the grid and local server. The local server's variation in overall performance of the application decreases with the increase in the number of cores being used to execute the application, except for the 2K scenario, with the smaller datasets of 2K and 6K showing low variation in system performance compared to the 12K dataset. On the 12K dataset the variation in performance is lower when the local server is used to maximum capacity, while an increase in the variation of the application performance is observed when using a local server restricted to eight cores. The variance in performance on the grid is relatively low for the smaller dataset when restricted to eight cores and it steadily increases when the 6K dataset is used. When running the experiment using the 12K dataset, a steep increase in the variation time is exhibited by the grid running on eight cores. On the contrary, when the grid is running on its full capacity of 48 cores, the variation observed is higher in the smaller dataset and steadily decreases when the bigger dataset of 12K is used.

The figure above also shows that the grid has more variation in performance than the local server. This means that there is more uncertainty in job completion time when using the grid compared to using the local server, for which the job completion time can be predicted based on a baseline.

4.2.4 Effect of the data staging time

We investigated the variation in performance on the grid running on 48 cores by selecting the 2K results on the grid and discovered that data staging time is a huge factor influencing the total turnaround time of each individual run shown as iterations in the table below. The table below shows a significant difference in the data staging time of iteration 1, which took **14.64802** seconds and iteration 5, which staged data in **932.91560** seconds. This difference directly affects the total turnaround time, which we use to calculate the average and standard deviation for the 6K dataset running on the grid containing 48 cores. The total processing time also varies depending on the CPU capacity of the worker node being used on the grid, but this variation, as seen in the table below, is less than that of the data staging time.

Iteration	Staging time (s)	Total processing time(s)	Total job time (s)
1	14.648016952	113.692899896	134.389706272
2	62.276286041	121.249677058	190.282781666
3	560.105986520	182.988365593	786.785846882
4	735.090274302	188.412845198	949.113712915
5	932.915595533	162.777407396	1133.790020130
Mean			638.8724136
STDEV			452.4397597

Table 7: 2K on maximum performance on the grid

We then consider a scenario with a constant minimum data staging time observed for the above table and we populate the table below:

Iteration	Staging time	Total processing time	Total job time
1	14.648016952	113.692899896	128.3409168
2	14.648016952	121.249677058	135.8976940
3	14.648016952	182.988365593	197.6363825
4	14.648016952	188.412845198	203.0608622
5	14.648016952	162.777407396	177.4254243
Mean			168.472256
STDEV			34.636338

Table 8: 2K maximum performance on the grid with minimum data staging time

The table above shows a significant reduction in the average and standard deviation times when we use an ideal case of constant minimum data staging time. This explains the big variation in performance on the grid. So, in the next section we reconsider the

overall system performance when a constant minimum in data staging time is used, to see how the grid will compare to the local server with more efficient data staging.

4.2.5 Overall performance (with minimum data staging time for the grid)

The table below shows 12 experiments, each consisting of five repetitions for each dataset (2K, 6K and 12K), with a varied number of cores as noted above, and their averages based on minimum data staging times. Standard deviations were calculated and recorded for the local server and the grid.

Experiment	System (X1)	Number of cores (X2)	Dataset size (X3)	Average (s)	Standard deviation
1	1	1	-1	168.472	34.6363380
2	1	1	0	524.139	46.8873646
3	1	1	1	15800.792	251.1536568
4	1	-1	-1	223.029	2.8542898
5	1	-1	0	691.847	0.6259693
6	1	-1	1	6971.539	255.0074642
7	-1	-1	-1	170.784	3.2662639
8	-1	-1	0	1399.122	12.3006675
9	-1	-1	1	6227.582	86.3364493
10	-1	1	-1	106.918	6.9922471
11	-1	1	0	956.726	8.4707101
12	-1	1	1	4268.306	37.7464254

Table 9: Averages of all experiments (minimum data staging time)

The table above is used to determine the overall performance of the local server against the grid with a special case of minimum data staging time and this analysis is depicted in the tables below by using the overall averages and standard deviation of the two systems.

Average	Seconds	Minutes	Hours
X1= +1 (Grid)	4063.30315	67.72171915	1.12869532
X1= -1 (Local server)	2188.23967	36.47066111	0.60784435
Effect	1875.06348	31.25105804	0.52085097

Table 10: Overall performance of the grid vs local server (minimum data staging)

The local server outperforms the grid by 46% when the minimum data staging time is used in the comparison between the grid and the local server.

The variation in the overall system performance on the grid against the local server is shown by the table below.

Standard deviation	Seconds	Minutes	Hours
X1= +1 (Grid)	98.5275138	1.64212523	0.02736875
X1= -1 (Local server)	25.8521272	0.430868787	0.00718115
Effect	72.6753866	1.211256443	0.02018761

Table 11: Variation on the overall performance of the grid vs local server

The table above shows that the variation on the grid is approximately four times more than the local server in terms of the overall performance of the system.

The graphs that follow give a graphical representation on the overall performance of the local server against the grid computer with minimum data staging time.

Dataset	Local server, 8C	Local server, Max	Grid, 8C	Grid, Max
2K	170.78	106.92	223.03	168.47
6K	1399.12	956.73	691.85	524.14
12K	6227.58	4268.31	6971.54	15800.79

Table 12: Averages of the grid vs local server for graph plotting (minimum data staging)

The figure below shows that the turnaround time taken to complete the task on the grid, using an ideal case of minimum data staging time, decreases compared to the observed real-world scenario. The performance of the local server does not change from what we observed in the real-life scenario, since the data are stored before runtime, and so we consider the performance of the grid. The grid running 48 cores with minimum data staging time performs relatively well at approximately 1.5 times that of the local server. It is interesting to note that the grid running on eight cores and 48 cores outperformed the local server when executing the application with the 6K dataset, but the time taken to run the application with the 12K dataset is still very long when the grid runs on 48 cores. Looking at all these observations, we note that the local server still outperforms the grid even with an ideal case of minimum data staging time.

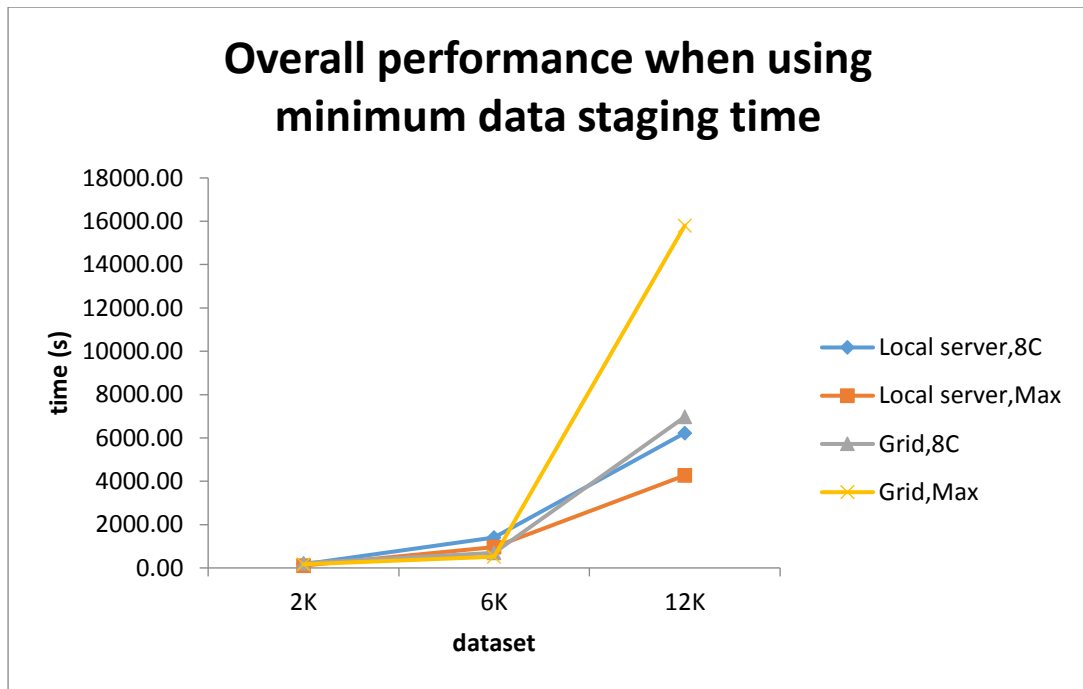


Figure 4-3: Overall performance when using minimum data staging time for the grid

The table below shows standard deviations used to represent the variance of the overall system performance graphically.

Dataset	Local server, 8C	Local server, Max	Grid, 8C	Grid, Max
2K	3.27	6.99	2.85	34.64
6K	12.30	8.47	0.63	46.89
12K	86.34	37.75	255.01	251.15

Table 13: Standard deviation values for the overall experiment on the grid vs local server

The figure below shows that the data staging time significantly influences the variation in total job time on the grid, with the smaller datasets of 2K and 6K being affected most by this inconsistent data staging. The variation in performance on the grid running on eight cores, on the two smaller datasets of 2K and 6K, is less than what we observe on the local server. However, the overall variation in time taken to complete a task is still higher on the grid compared to the local server and this affirms that the grid is unpredictable in terms of pure performance and the levels of uncertainty in total job time are higher on the grid compared to the local server.

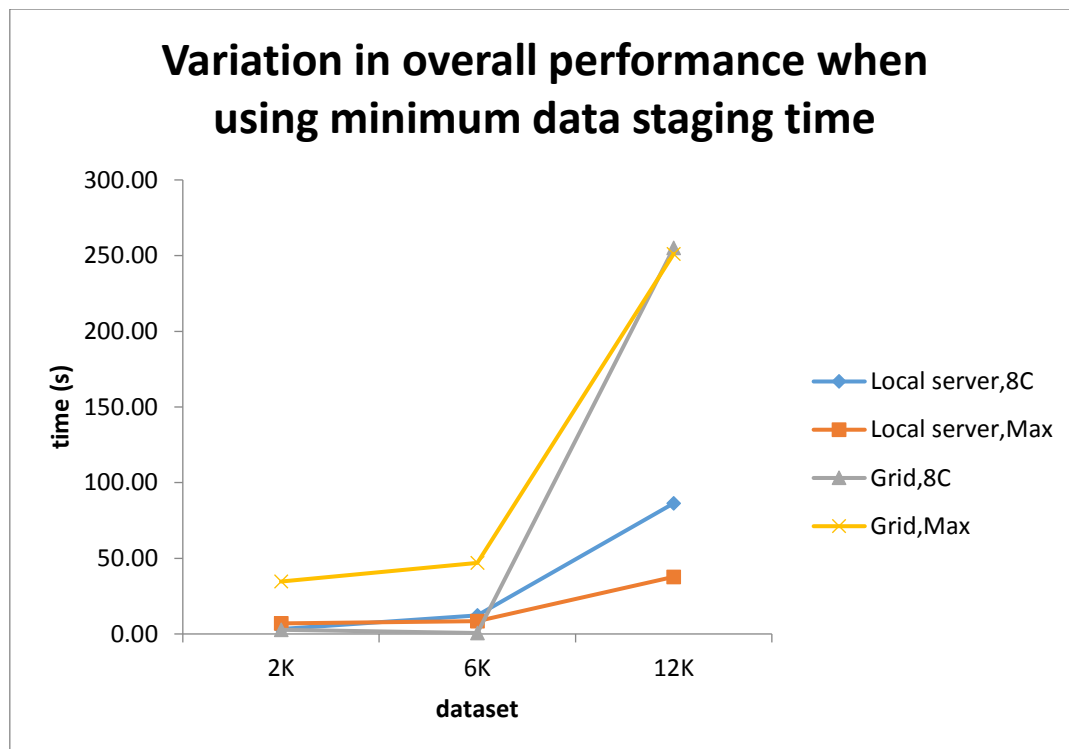


Figure 4-4: Variation in overall performance when using minimum data staging time for the grid

4.3 Second application: preparing a specific text corpus

An automatic speech recognition process requires a language model. This language model is built from a text corpus which consists of documents appropriate for the language model. In a lecture transcription task the researcher wanted to investigate the effects of document selection and data preparation of a text corpus on the usefulness of the language model built from it for lecture transcription. Different corpora were to be built from university study guides. Step 1 was to convert each source document from the format in which it was available (doc, docx or pdf) to plain text. This step was only performed once on the collection of study guides (more than 50 GB) and produced text files totaling about 5 GB. The subsequent steps were designed to clean up and normalize the text. Different approaches were possible here and since the choices here would impact on the corpus and the language model built from it, the researcher wanted to be able to repeat this process to test the resulting changes. This study used the following two steps (step 2 and 3) of the corpus clean-up and normalization process as an application for comparing the performance of the multicore server and the computational grid. Step 2 cleans up the text by removing debris left over from the automatic text extraction processes of step 1. This debris had

a big impact on the language models which relies on frequency of words and sequences of words. The challenge is to remove as much debris as possible while retaining as much useful text as possible. Step 3 normalized the text in terms of punctuation, capitalization and representation of accents and other diacritical markings. These steps typically implies a linear relationship between data size and processing time required. Regular expressions are used extensively for this process.

4.3.1 Experiment with preparation of a specific text corpus

For this timing experiment steps 2 and 3 as described above were used as a single task. Since the text to be processed was given as several thousand individual files, this granularity was used for implementing parallelism. Five iterations were executed on both the local server and the grid.

4.3.2 Results and analysis of text processing experiment

Iteration	Local Server	Grid
1	851.61	2171.798312
2	851.24	2176.849005
3	846.83	2219.274833
4	849.35	2019.26736
5	852.15	2150.585398
Mean	850.236	2147.554982
STDEV	2.176299612	75.9218566
Coefficient of variation %	0.255964181	3.535269516

Table 14: Job completion times for text corpus preparation

Calculating the coefficient of variation

$$CV = (STDEV/mean) * 100\%$$

The table above gives the mean, standard deviation and coefficient of variation between the local server and the computational grid. The standard deviation shows that there is more reliability in terms of the overall processing time on the local server than on the computational grid. The coefficient of variation of 0.26% is observed on the local server against 3.54% on the computational grid, this means that the predictability in processing time on the local server is higher than on the computational grid.

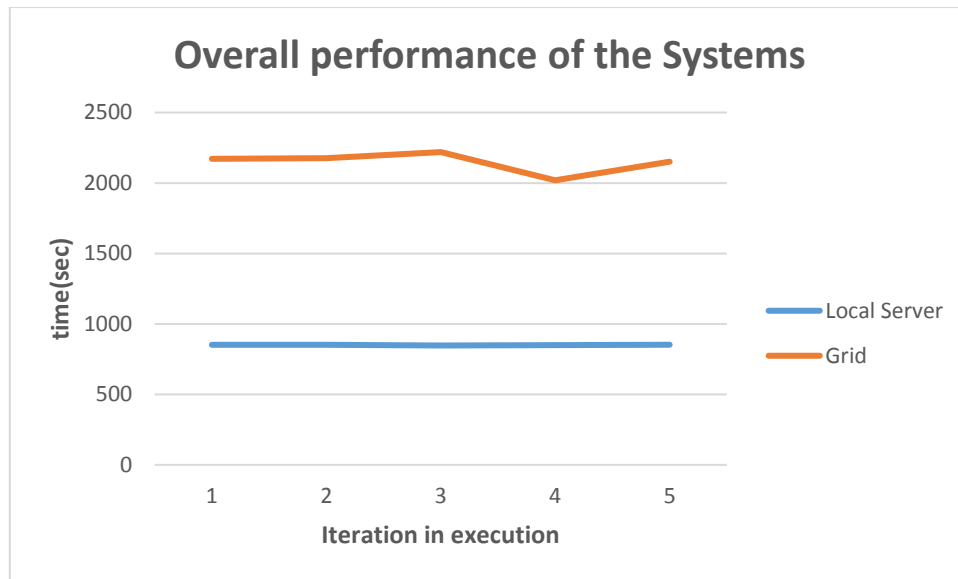


Table 15: Overall performance with second application

The diagram above shows that the local server performs better than the computational grid for the experiment performed hence in the case of pure performance the local server is a better choice than the computational grid.

4.4 Narrative

The study upon investigating literature and determining the appropriate methodology for the research, turned its focus on understanding the architecture of the existing computing infrastructures to be used in this research. A look at the local server and the computational grid was performed so as to determine their hardware architectures, software stack as well as their usage policies and usage patterns. The sections that follow, reflects the researcher's experience in using the local server as well as the computational grid.

4.4.1 Local Server

In order to guide the decisions of the researcher in using the local server, the usage policy of the local server was carefully studied and access rights to the local server were requested and subsequently granted by the server administrator. Upon the granting of access to the researcher, the study carefully examined the hardware, software, network and storage capabilities of the local server and all the relevant information to the study was noted. The researcher further observed the usage patterns on the local server so as to determine, the load patterns of the local server in order to comply with the usage policy set out by the local server administrator.

The study proceeded to look at the software stack installed on the local server and discovered that a Ubuntu Linux operating system was installed and the repositories that are used by the server were not updated to the latest release. The researcher after identifying all the relevant software needed for the experiments to be ran, compiled the relevant software binaries and their dependences in configuring the system for the experiment to be conducted. To set-up the application to be executed, the researcher placed the scripts in the appropriate directories on the local server and the data consumed by the application was moved to the storage disks of the local server. A test run was performed on the local server so as to determine if bugs existed and in the case of these, debugging was done. After the test run completed successfully, focus turned to efficiently conducting the experiment and the researcher had to abide by the usage policy as well as patterns in successfully conducting the experiments. Data generated as well as timing results of the experiment were stored on the local server.

It should be noted that the above process came with its challenges as the server was unavailable to the researcher due to network failures caused by power failures as well as the requirement of finding time when the load on the server was not minimal. The researcher mitigated all these factors to ensure a successful completion of the experiments on the local server.

4.4.2 Computational grid

The study explored a computational grid and in this case the South African National Grid so as to understand as well as use sites on this grid. The researcher carefully studied this computational grid in order to understand its operation and usage. The usage policy on the grid was considered and certain restrictions imposed by other sites that form part of this grid were noted. The researcher requested access into this grid from the grid administrators but failed to get authorization since he is not part of a federation with the South African National Grid. This computational grid authenticates and authorizes by using identity federations through their proxy servers that are authorized access into the South African national grid. The researcher finally gained access through a trust certificate awarded to him by the administrators as a result of collaborating with other researchers and administrators on the grid.

The researcher started to investigate the internal operations of this grid and the grid middleware of gLite used on this grid as well as setting-up other open source project

handling systems such as git which are used with the grid components such as the Continuous Delivery of Research Applications in a Distributed Environment (CODE-RADE) project used in building applications. This process enabled the researcher to collaborate easily but was very time consuming in learning all the new technologies. The researcher then turned his focus on building the applications to be tested on the grid and stored their data on the metadata service and a Logical File Catalogue (LFC) used to stage the data at runtime on the grid. After the application compiled successfully they were deployed onto the available sites and Job Description Language (JDL) files that define how a task is executed and the sequence of execution were written and submitted on the grid Computing Element (CE). The results obtained were placed on the Open Access Repository (OAR), which made it easy to share the results and collaborate on them.

During the course of setting-up and executing the applications, the researcher paid cognizance of the redundancy capabilities of the grid when other sites failed, jobs were simply automatically reallocated within a given space of time to the next available site. The unavailability of site administrators and their overhead was also noted during the process. Furthermore, the lack of total control of the grid by the researcher as well as the unpredictability of job completion times were also noted.

4.5 Summary

The researcher used empirical data obtained from the research process to analyze and draw inferences based on these trends of the results. The selection and use of descriptive statistics benefited the study immensely, as it established the difference in pure performance of the multicore server against that of the computational grid tested for two different applications. The narrative on the workflows of the two systems being compared further asserts important lessons that guides the MuST researcher in determining the computing platform to be used when such a choice is to be made. It should be noted that the work done in this chapter builds from the work done in the previous chapters of literature review and research methodology as it seeks to inform the discussion on critical reflection, contribution of the study and recommendations made as a result of this research in the final chapter.

5 Conclusion and Reflection

5.1 Introduction

This chapter that concludes the study begins with a summary of the chapters, where the work done in all the preceding chapters is outlined. A critical reflection of the study is then conducted in order to realize the contribution of the study as well as make recommendations based on this reflection. The research then turns the focus to possible future work that may build on top of this research and briefly outlines the limitations noted during the research. A summary of the chapter concludes this study.

5.2 Summary of chapters

5.2.1 Chapter 1 – Introduction

The background of the research and the concepts regarding the study were defined. The problem statement was clearly outlined and a focus of the study as well as the rationale behind the use of pattern recognition and speech technology applications was given. The study objectives were defined and the research methodology used to meet these objectives was clearly stated.

5.2.2 Chapter 2 – Literature review

The literature review informed the study on the computing infrastructures being explored and provided an understanding of the different computing infrastructures, their history, operation, as well as the benefits that can be obtained based from previous studies in the field of computing. The chapter gave a solid foundation regarding the nature and operation of the computing systems of multicore servers and computational grids, being explored by the study

5.2.3 Chapter 3 – Research methodology

This chapter on research design and methodology exposed the process taken in meeting the study objectives of this research. The researcher displayed an understanding of fundamental concepts in research and this guided the study in choosing an appropriate research design and methodology. Critical concepts of the study were explored in this chapter and a holistic view of how the research is to be conducted has been given. Partially mixed concurrent dominant status research design and the methodology of an experiment and a narrative were used to make findings and recommendations.

Chapter 4 – Results analysis

The researcher used empirical data obtained from the research process to analyze and draw inferences based on these trends of the results. The selection and use of descriptive statistics benefited the study immensely, as it established the difference in pure performance of the multicore server against that of the computational grid tested for two different applications. The narrative on the workflows of the two systems being compared further asserted important lessons that guides the MuST researcher in determining the computing platform to be used when such a choice is to be made.

5.3 Critical reflection

In this section the researcher reflects on the findings of the study in respect to its objectives.

5.3.1 Findings in respect of objective 1

- **Objective 1: Utilize an existing campus grid on the South African national grid and local multicore server to set up and run compute-intensive experiments.**

This objective was met by selecting two compute-intensive experiments for processing on the grid and the local server. The scope of the experiments and the datasets they use was identified and optimized for the local server and the computational grid. The experiment was run while collecting the timing data of each run on both systems and the results were recorded for use in comparing the timing performances of the grid to that of the local server.

During this process, several factors that influence the choice of researchers who are considering which of the two systems suits their needs were observed and lessons learnt in the process of successfully running a compute-intensive experiment on the grid and the local server were discussed in the design workflow analysis.

The study discovered from the timing results analysis done in chapter 4 that the local server on average outperforms the grid in turnaround time and that the variability in job completion time on the grid is higher than that of the local server. Therefore the study concludes that the local server (when dedicated to the experiments) outperforms the grid in a case of pure performance for the given task.

5.3.2 Findings in respect of objective 2

- **Objective 2: Determine the choices and trade-offs for performance and system management benefits of a campus computational grid and a multicore server.**

In this section the experience of the researcher while working on both the local multicore server and the computational grid in terms of advantages and disadvantages of one system over the other is discussed. The design workflows and system management benefits of the two computing platforms are explored, the discussion begins with the researcher's observations on the local server and on the computational grid used in the study.

5.3.2.1 Advantages of the local server

The local server ensured quick and easy data distribution and experiment setup, as well as easy software installation, since we simply had to utilize the Linux repositories to download the necessary software. The speed of data transfer and the ease of experiment configuration and software installation ensured that the experiments on the local server could be done easily and quickly, therefore the researcher could achieve a lot of work in a short period of time.

It was observed during the experiment that the researcher had full control of the local server in terms of access and configuration, which made it easier to configure and adapt the experiment on the local server. Access on the local server is locally controlled, ensuring that confidentiality, integrity and authenticity of data are preserved. The local server's security is locally controlled and the system is shared by local researchers. This ensures better enforcement of security policies on the local server.

The local server's performance was easy to predict and this made it easy for the researcher to run all the experiments effectively. The local server's load statistics were readily available to the researcher and this facilitated easy planning and execution of the experiments.

Although the results of the experiments in this study showed that the local server outperformed the grid, it should be noted that measurements were taken (by design) when the local server was dedicated to these experiments. When a researcher runs several experiments at the same time, or indeed when several researchers use the

server at the same time, it may not be so easy to predict how long an experiment will run.

Service level agreements on how to share the local server's resources and how to use the local server effectively were easy to attain, simply because the researchers who use this system were readily available for consultation. The patterns of use on the local server and the demand for resources in a given period of time were easy to collect and factor into the planning of the experiments.

The researcher had access to the hardware of the local server in cases where there were network problems and power failures. This assisted the researcher in resolving these problems in good time. The backing up of data on the local server was also easy to achieve, since the system is locally available.

5.3.2.2 Disadvantages of the local server

The researcher experienced challenges such as outdated software versions, which affected the process of setting up experiments on the local server. This was mainly due to the fact that there was no full-time server administrator and hence maintenance was an issue on the local server. This could be mitigated by a full-time server administrator to handle the support needed on this platform.

The local server does not have redundancy and this affected the researcher at times when the local server experienced power and network failure. On several occasions the experiments had to be put on hold to resolve the problems on the local server, and this had a negative impact on the planning of the researcher.

The performance of the local server is affected negatively when it is overloaded. It has no inbuilt mechanisms to deal with an overload and hence the researcher had to deal manually with the concerns of system load, which wasted time. Some of the mechanisms the researcher used to mitigate the problem of system overload were to prioritize all the jobs being submitted whenever the system was overloaded and to submit jobs overnight on the local server.

The local server does not promote easy sharing of data, scripts and other resources because of local access control and restrictions on who is allowed to use the system. This presented the researcher with a challenge in terms of collaborating the research with other researchers, as well as sharing the data and scripts that needed to be used to complete the study successfully.

The local server used in this research does not abstract workflows to execute them on any other stand-alone machine if the need arises. This means that the researcher had to restart the process of configuring an application as well as setting up the experiments if the server being used was changed. Therefore the local server did not provide for permanent application porting.

5.3.2.3 Advantages of the computational grid

The computational grid used in this study allows abstraction of workflows so they can be executed at any available site without any need for the researcher to change the application. This specific grid uses a built-in service called CODE-RADE (**C**ontinuous **D**elivery of **R**esearch **A**pplications in a **D**istributed **E**nvironment) that makes porting applications easier and ensures that the ported application can be accessed from anywhere. The researcher abstracted workflows and used CODE-RADE to port the application and submit jobs on several grid sites without having to repeat the process of application porting.

The researcher selected the sites to be used in this study and the subset of resources to be used in a particular setting. This was a major benefit for the researcher in that the study could be conducted in the desired way and with the resources that needed to be used, such as the sites selected and not the whole grid. This flexibility in choice of services on the grid was very important in this research and would be a major benefit for other researchers who may want to undertake the same research.

The computational grid used allows abstraction of the data location and this service was used to achieve efficient data distribution. The data were placed in a metadata service and a logical file catalogue was used in the distribution of data. This was beneficial to this research in that data were readily available at the site that needed to use it without physically transferring the data there, which helped ensure the integrity of the data that were physically stored in one location.

The services provided on the South African grid are free of charge and the researcher used these services with no regard for cost, from which other researchers can also benefit. The grid provided redundancy and better fault handling during the experiment. The grid used in this study provided for easy collaboration of research, as the researcher found it easy to collaborate with some of the site administrators on the South African national grid, such as Bruce Becker and Sakhile Masoka (Becker et al. 2016). These site administrators made the application porting easy for the researcher

and they assisted the researcher with usage policies agreed upon for the South African national grid.

This computational grid provides an underlying security infrastructure based on an X.509 certificate and a trust anchor of global certificate authorities, ensuring robust security for the services on the grid. Personal certificates are not easily issued on this grid, as experienced by the researcher when trying to obtain a personal certificate. This ensures better security of resources on the grid.

The researcher had several back-end services at his disposal during the study and there was no need for concern about accounting and information services. Services such as the Logical File Catalogue, gLibrary metadata service and CREAM were accepting and wrapping jobs for submission.

The campus grid sites used, such as UJ and Wits, made it easier to negotiate for more control of the sites in terms of submitting jobs with particular demands and this made the comparison in this study possible. The researcher benefited from this flexibility and control in that the proxy renewal policies on these sites had to be altered to make it possible for the experiments to be run to completion.

5.3.2.4 Disadvantages of the computational grid

The computational grid used in this study lacks documentation of its structure, services and usage policies, hence it was very difficult for the researcher to understand the services and benefits of using this system. The researcher also struggled to gain access to the grid owing to lack of documentation, as well as the restriction imposed on personal certificates. The researcher tried numerous times but failed to gain access to the grid because he did not belong to a virtual organization that is part of the South African national grid. The NWU is not currently part of the South African national grid, therefore its proxy could not be used to gain access to the grid services. This presents individual researchers with a challenge in terms of access to this grid.

The overheads and unavailability of site administrators at times made it difficult for the researcher to progress effectively with the study, as he needed site administrators with full access to port applications and submit jobs for processing. The researcher discovered that first-time users on this grid would struggle to use it without first consulting any of the experts of the South African national grid.

The inconsistencies in data staging time experienced on the grid made the system unpredictable and affected the performance of the grid in a real-life scenario. The data

staging time experienced could be due to factors such as network support provided by the South African National Research and Education Network and the configuration of services on the sites.

The adaptation of the experiments from the local server to the grid was a challenging process in terms of writing job description language files to be used in submission of jobs on the grid. The researcher had to spend a lot of time trying to become proficient in the use of this grid with little or no documentation at all. The researcher at times had to rely on some site administrators for information and guidance on policies and practice in using this grid.

The researcher faced challenges of submitted jobs being aborted before completion owing to proxy renewal settings and problems with the input sandbox and this adversely affected the progress of the study.

5.4 Contribution of the study

The study informed MuST researchers of the factors that arise when deciding which computing platform to use beyond their personal computer. It makes apparent the differences between performance and most importantly the differences in the workflow design of the two computing infrastructures.

5.5 Recommendations

Based on our findings and experience gathered during this study, we make the following recommendations to both the institution and researchers regarding the use of a grid in alleviating the pressure currently imposed on the local server:

- **Institution**

The institution could hire a grid site administrator to set up and maintain a campus grid as well as port applications to make it easy for researchers to use the grid. The institution could also join a federation such as the South African Grid Network and benefit from using other grid sites on this federation at no cost.

- **Researchers**

Researchers could use the grid if their needs are as follows:

- Ease of collaboration with other researchers.
- Mobility, as the grid allows the researcher to access and use the grid from anywhere.

- Redundancy, since a failure of one worker does not have a heavy impact on the completion of the task being executed.
- Long-term use, as applications ported to CODE-RADE are permanent unless changed or removed.
- Continuously increasing need of more computational power beyond what they have on their local server.

5.6 Limitations

The study was limited by the complexity of joining the South African National Grid, the insufficiency of documentation that describes the use of the grid. Furthermore the fact that NWU is not a federated identity presented a challenge in joining the grid. The load on the local server also limited the amount of time the researcher had to utilize the server.

5.7 Future work

The researcher noted the following interesting factors that affect the performance of the grid and in future more research will have to be done in these areas:

- Investigate the network performance and how it affects the grid used.
- Investigate the best methods of staging data to curb the variability in data staging times observed in this study.
- Investigate the administrative issues of using other computing infrastructures such as cloud computing and cluster computing.
- Investigate the administrative issues of joining a federation such as the South African Grid Network.
- Compare the performance of other applications in other scientific fields such as Astrophysics and Climate Forecasting.

5.8 Personal Reflection

This study was a success, as it achieved both objectives set out in the beginning of the study. The researcher benefited immensely from this study and has learnt several invaluable lessons in parallel computing, distributed computing, research methods and critical problem-solving skills that will be put to good use in future. The research process was not always smooth, as it had its ups and downs, but the researcher

acquired resilience in dealing with problems and finding solutions in effective problem-solving. Finally, the researcher hopes that this research will benefit the institution in fully utilizing its campus computing resources and other researchers in trying to determine what processing platform suits their needs, as well the implications of these choices.

Reference list

- Abramson, D., Buyya, R. & Giddy, J. 2002. A computational economy for grid computing and its implementation in the Nimrod-G resource broker. *Future Generation Computer Systems*, 18(8), pp.1061–1074. Available at: <http://linkinghub.elsevier.com/retrieve/pii/S01677339X02000857>.
- Akhter, S. & Roberts, J. 2006. *Multi-core programming* (Vol. 33). Hillsboro: Intel press.
- Allcock, B., Bester, J., Bresnahan, J., Chervenak, A.L., Foster, I., Kesselman, C., Meder, S., Nefedova, V., Quesnel, D. & Tuecke, S., 2002. Data management and transfer in high-performance computational grid environments. *Parallel Computing*, 28(5), pp.749-771.
- Anon, Intel® Xeon® Processor X5690 (12M Cache, 3.46 GHz, 6.40 GT/s Intel® QPI) Specifications. Available at: http://ark.intel.com/products/52576/Intel-Xeon-Processor-X5690-12M-Cache-3_46-GHz-6_40-GTs-Intel-QPI [Accessed May 8, 2016].
- Antonioletti, M., Atkinson, M., Baxter, R., Borley, A., Chue Hong, N.P., Collins, B., Hardman, N., Hume, A.C., Knox, A., Jackson, M. & Krause, A. 2005. The design and implementation of Grid database services in OGSA-DAI. *Concurrency and Computation: Practice and Experience*, 17(2-4), pp.357-376.
- Asanovic, K., Bodik, R., Catanzaro, B.C., Gebis, J.J., Husbands, P., Keutzer, K., Patterson, D.A., Plishker, W.L., Shalf, J., Williams, S.W. & Yelick, K.A. 2006. The landscape of parallel computing research: A view from Berkeley (Vol. 2). Technical Report UCB/EECS-2006-183, EECS Department, University of California, Berkeley.
- Azeez, N. & Abidoye, A., 2011. Peer-to-Peer Computing and Grid Computing: towards a better understanding. *Pacific Journal of Science and Technology*, 12(1), pp.270–276. Available at: <http://repository.uwc.ac.za/xmlui/handle/10566/962> [Accessed March 14, 2014].
- Baker, M., Buyya, R. & Laforenza, D., 2002. Grids and Grid technologies for wide-area distributed computing. *Software: Practice and Experience*, 32(15), pp.1437–1466. Available at: <http://doi.wiley.com/10.1002/spe.488> [Accessed June 4, 2014].
- Baltopoulos, I.G., 2005. *Web Service Composition in the Grid Environment*.
- Becker, B., Masoka, S. & Risinamhodzi David, 2016. CODE-RADE: Libsvm Release. Available at: <http://dx.doi.org/10.5281/zenodo.51591>.

- Berman, F., Fox, G. & Hey, A. 2003. The Grid: past, present, future. [Wiley Online Library]. doi: 10.1002/0470867167.ch1.
- Berman, F. & Hey, T., 2004. The scientific imperative. (In Foster I. & Kesselman, C. eds. *The Grid 2: Blueprint for a New Computing Infrastructure*. 2nd ed. San Francisco, CA: Morgan Kaufmann.)
- Bernhardt, A., Maximo, A., Velho, L., Hnaidi, H. & Cani, M.P. 2011. Real-time terrain modeling using CPU-GPU coupled computation. In *24th SIBGRAPI Conference on Graphics, Patterns and Images*. August. (pp. 64-71). IEEE.
- Blake, G., Dreslinski, R.G. & Mudge, T. 2009. A survey of multicore processors: A review of their common attributes. *IEEE Signal Processing Magazine*, 26(6), pp.26–37.
- Blanke, T., Hedges, M. & Dunn, S. 2009. Arts and humanities e-science—Current practices and future challenges. *Future Generation Computer Systems*, 25(4), pp.474–480. Available at: <http://linkinghub.elsevier.com/retrieve/pii/S0167739X08001684> [Accessed July 2, 2014].
- Bose, A., Wickman, B. & Wood, C. 2004. Mars: A metascheduler for distributed resources in campus grids. In *Grid Computing. Proceedings. Fifth IEEE/ACM International Workshop on Grid Computing*, 8 November (pp. 110-118). IEEE. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1382822 [Accessed July 24, 2014].
- Bryman, A. ed. 2006. *Mixed Methods*. SAGE. Available at: <https://uk.sagepub.com/en-gb/afr/mixed-methods/book228130> [Accessed May 10, 2016].
- Burrell, G. & Morgan, G. 1979. *Sociological paradigms and organisational analysis*, London: Heinemann.
- Buyya, R. et al. 2009. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, 25(6), pp.599–616. Available at: <http://linkinghub.elsevier.com/retrieve/pii/S0167739X08001957> [Accessed July 9, 2014].
- Chandrakasan, A.P., Sheng, S. & Brodersen, R.W., 1992. Low-Power CMOS Digital Design. *IEICE TRANSACTIONS on Electronics*, E75-C(4), pp.371–382.
- Chervenak, A., Deelman, E., Kesselman, C., Allcock, B., Foster, I., Nefedova, V., Lee, J., Sim, A., Shoshani, A., Drach, B. & Williams, D. 2003. High-performance remote access to climate simulation data: a challenge problem for data grid technologies. *Parallel Computing*, 29(10), pp.1335-1356.

- Chetty, M. & Buyya, R. 2002. Weaving computational grids: how analogous are they with electrical grids? *Computing in Science & Engineering*, 4(4), pp.61–71.
- Czajkowski, K., Fitzgerald, S., Foster, I. & Kesselman, C. 2001. Grid information services for distributed resource sharing. In *High Performance Distributed Computing, 2001. Proceedings. 10th IEEE International Symposium on* (pp. 181-194). IEEE.
- Czajkowski, K., Foster, I. & Kesselman, C. 1999. Resource co-allocation in computational grids. In *High Performance Distributed Computing, 1999. Proceedings. The Eighth International Symposium on* (pp. 219-228). IEEE.
- Czajkowski, K., Foster, I., Karonis, N., Kesselman, C., Martin, S., Smith, W. & Tuecke, S. 1998. A resource management architecture for metacomputing systems. In *Workshop on Job Scheduling Strategies for Parallel Processing. March* (pp. 62-82). Springer Berlin Heidelberg. Available at: <http://link.springer.com/chapter/10.1007/BFb0053981> [Accessed July 23, 2014].
- De Villiers, M.R. 2005. Three approaches as pillars for interpretive information systems research: development research, action research and grounded theory. In *Proceedings of the 2005 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries. July* (pp. 142-151). SAICSIT.
- Field, L. & Schulz, M. 2008. Grid interoperability: the interoperations cookbook. *Journal of Physics: Conference Series*, 119(1), p.012001. Available at: <http://stacks.iop.org/1742-6596/119/i=1/a=012001?key=crossref.71410a138e494316184b468c9025b7f6> [Accessed July 3, 2014].
- Foster, I., Kesselman, C., Nick, J.M. & Tuecke, S. 2002. Grid services for distributed system integration. *Computer*, 35(6), pp.37–46. Available at: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1009167> [Accessed June 4, 2014].
- Foster, I., Kesselman, C., Nick, J.M. & Tuecke, S. 2003. The physiology of the grid. In *Grid computing: making the global infrastructure a reality*, pp.217-249.
- Foster, I. & Kesselman, C. 1999. The Globus project: a status report. *Future Generation Computer Systems*, 15(5-6), pp.607–621. Available at: <http://www.sciencedirect.com/science/article/pii/S0167739X99000138> [Accessed June 4, 2014].
- Foster, I. & Kesselman, C. eds. 2004. *The Grid 2: Blueprint for a New Computing Infrastructure*. 2nd ed. San Francisco, CA: Morgan Kaufmann.
- Foster, I., Kesselman, C. & Tuecke, S. 2001. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications*, 15(3), pp.200–222.

- Foster, I. and Tuecke, S. 2005. Describing the elephant: The different faces of IT as service. *Queue*, 3(6), pp.26-29.
- Frumkin, M. & Van der Wijngaart, R.F. 2002. Nas grid benchmarks: A tool for grid space exploration. *Cluster Computing*, 5(3), pp.247-255.
- Gandotra, I., Abrol, P., Gupta, P., Uppal, R. & Singh, S. 2011. Cloud computing over cluster, grid computing: a comparative analysis. *Journal of Grid and Distributed computing*, 1(1), pp.1-4.
- Geer, D. 2005. Industry trends: Chip makers turn to multicore processors. *Computer*, 38(5), pp.11–13.
- Gentzsch, W. 2002. Grid Computing, A Vendor's Vision. *2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'02)*, pp.290–290. Available at: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1540476>.
- Georgiou, Y., Leduc, J., Videau, B., Peyrard, J. & Richard, O. 2006. A tool for environment deployment in clusters and light grids. In *Proceedings 20th IEEE International Parallel & Distributed Processing Symposium*, April (pp. 8-14).
- Gepner, P. & Kowalik, M.F. 2006. Multi-Core Processors: New Way to Achieve High System Performance. *International Symposium on Parallel Computing in Electrical Engineering (PARELEC'06)*, pp. 0–4.
- Giddens, A. & Giddens, A. 1974. *Positivism and sociology*, Heinemann London.
- Giwa, O. & Davel, M.H. 2014. Language identification of individual words with joint sequence models. *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, (September), pp.1400–1404.
- Gochman, S., Group, M. & Corporation, I. 2006. Introduction to Intel® Core™ Duo Processor Architecture. *Intel Technology*, 10(2), pp.89–98.
- Goede, R., Taylor, E. & van Aardt, C. 2011. *Combining Research Paradigms to Improve Poor Student Performance*, IGI Global. Available at: <http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-4666-2491-7.ch013>.
- Goede, R. & de Villiers, C. 2003. The applicability of grounded theory as research methodology in studies on the use of methodologies in IS practices. In *Proceedings of the 2003 annual research conference of the South African institute of computer scientists and information technologists on Enablement through technology*. pp. 208–217.
- Goodacre, J. & Sloss, A.N. 2005. Parallelism and the ARM instruction set architecture. *Computer*, 38(7), pp.42–50.

- Hammond, L., Nayfeh, B.A. & Olukotun, K. 1997. A single-chip multiprocessor. *Computer*, 30(9), pp.79–85.
- Johnson, R.B. & Onwuegbuzie, A.J. 2004. Mixed methods research: A research paradigm whose time has come. *Education researcher*, 33(97), pp.14–26.
- Johnston, W.E., Gannon, D. and Nitzberg, B. 1999. Grids as production computing environments: The engineering aspects of NASA's Information Power Grid. In *High Performance Distributed Computing, 1999. Proceedings. The Eighth International Symposium on* (pp. 197-204). IEEE.
- Kuhn, T.S. 2011. The essential tension: Selected studies in scientific tradition and change. Chicago,IL : University of Chicago Press. pp.30–65.
- Kumar, A. & Jangra, A. 2013. Genuine Character Foundation of Scheduling and its Challenges in Grid Computing. *Journal of Advanced Research in Computer Science*, 4(9).
- Leech, N.L. & Onwuegbuzie, A.J. 2009. A typology of mixed methods research designs. *Quality and Quantity*, 43(2), pp.265–275.
- Mabakane, M.S. 2011. Scaling of scientific software applications on CHPC Clustering Environment. Tshwane: TUT. (Thesis M-Tech)
- Marston, S., Li, Z., Bandyopadhyay, S., Zhang, J. & Ghalsasi, A. 2011. Cloud computing - The business perspective. *Decision support systems*, 51(1), pp.176-189..
- Merriam, S.B. & Tisdell, E.J. 2015. *Qualitative research: A guide to design and implementation* 4th ed., Hoboken, NY: John Wiley & Sons.
- Montgomery, D.C. 2008. Design and analysis of experiments. Hoboken, NY: John Wiley & Sons.
- Moretti, C., Bui, H., Hollingsworth, K., Rich, B., Flynn, P. & Thain, D. 2010. All-pairs: An abstraction for data-intensive computing on campus grids. *IEEE Transactions on Parallel and Distributed Systems*, 21(1), pp.33-46..
- Myers, M.D. 1997. Qualitative research in information systems. *Management Information Systems Quarterly*, 21(2), pp.241-242.
- Nagaraju, M. & Anitha, B. 2012. Performance Analysis of Various New Technologies for Computing. *Journal of Engineering, Computers and Applied Sciences*, 1(2).
- Németh, Z.V. 2003. Characterizing Grids: Attributes, Definitions, and Formalisms. *Journal of Grid Computing*, 1, pp.9–23.
- Olivier, M.S. 2009. Information technology research: a practical guide for computer science and informatics. Pretoria: Van Schaik.

- Parashar, M. & Lee, C.A. 2005. Grid computing: introduction and overview. *Proceedings of the IEEE, special issue on grid computing*, 93(3), pp.479-484.
- Pather, S. & Remenyi, D. 2004. Some of the philosophical issues underpinning realism in Information Systems: From Positivism to Critical Realism. *Fulfilling the Promise of ICT. Proceedings of SAICSIT*.
- Patton, M.Q. 2005. *Qualitative research*, Wiley Online Library.
- Peng, L., Peir, J.K., Prakash, T.K., Chen, Y.K. & Koppelman, D. 2007. Memory performance and scalability of Intel's and AMD's dual-core processors: a case study. In *IEEE International Performance, Computing, and Communications Conference*. April. (pp. 55-64). IEEE.
- Priol, T. 2005. Objects, Components, Services for Grid middleware: pros and cons. In *Keynote talk at European Grid Conference, Science Park*. Amsterdam, The Netherlands. Available at: <http://web.eecs.utk.edu/~dongarra/Lyon-2004-slides/Wednesday08-Thierry-Priol.pdf> [Accessed July 22, 2014].
- Roy, A., Xu, J.X.J. & Chowdhury, M.H. 2008. Multi-core processors: A new way forward and challenges. *2008 International Conference on Microelectronics*, pp.454–457.
- Schauer, B. 2008. Multicore processors – a necessity. *ProQuest discovery guides*, September, pp.1-14.
- Scherer, T. 2015. 50 Years of Moore's Law. *Elektor.POST*. Available at: <https://www.elektormagazine.com/articles/moores-law> [Accessed April 27, 2016].
- Schmidberger, M., Morgan, M., Eddelbuettel, D., Yu, H., Tierney, L. & Mansmann, U. 2009. State-of-the-art in Parallel Computing with R. *Journal of Statistical Software*, 47(1).
- Schwiegelshohn, U., Badia, R.M., Bubak, M., Danelutto, M., Dustdar, S., Gagliardi, F., Geiger, A., Hluchy, L., Kranzlmüller, D., Laure, E. & Priol, T. 2010. *Perspectives on grid computing. Future Generation Computer Systems*, 26(8), pp.1104-1115.
- Sharma, D. & Mittal, P. 2013. Grid Computing An Overview. *Journal of Science, Engineering and Technology Research*, 2(3).
- Smarr, L. 1998. Grids in context. San Francisco, CA: Morgan Kaufmann. pp.1–13.
- Smith, P.M., Hacker, T.J. & Song, C.X. 2008. Implementing an industrial-strength academic cyberinfrastructure at Purdue University. *2008 IEEE International Symposium on Parallel and Distributed Processing*, pp.1–7. Available at: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4536447> [Accessed July 24, 2014].

- Snavey, A., Chun, G., Casanova, H., Van der Wijngaart, R.F. & Frumkin, M.A. 2003. Benchmarks for grid computing: a review of ongoing efforts and future directions. *ACM SIGMETRICS Performance Evaluation Review*, 30(4), pp.27-32.
- Srinivasan, L. & Treadwell, J. 2005. An overview of service-oriented architecture, web services and grid computing. *HP Software Global Business Unit*, 2..
- Stockinger, H. 2007. Defining the grid: a snapshot on the current view. *The Journal of Supercomputing*, 42(1), pp.3–17.
- Stoelwinder, A. 2004. Performance evaluation in computational grid environments. *Proceedings. Seventh International Conference on High Performance Computing and Grid in Asia Pacific Region*, pp.54–62. Available at: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1324016>.
- Sultan, N. 2010. Cloud computing for education: A new dawn? *International Journal of Information Management*, 30(2), pp.109–116. Available at: <http://linkinghub.elsevier.com/retrieve/pii/S0268401209001170> [Accessed July 10, 2014].
- Tapia, J.J. & D’Souza, R. 2009. Data-parallel algorithms for large-scale real-time simulation of the cellular potts model on graphics processing units. *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, (October), pp.1411–1418.
- Tendler, J.M., Dodson, J.S., Fields, J.S., Le, H. & Sinharoy, B. 2002. POWER4 system microarchitecture. *IBM Journal of Research and Development*, 46(1), pp.5-25.
- Trnkoczy, J., Turk, Ž. & Stankovski, V., 2006. A grid-based architecture for personalized federation of digital libraries. *Library Collections, Acquisitions, and Technical Services*, 30(3-4), pp.139–153.
- Varshney, S., Srivastava, L. & Pandit, M. 2012. Electrical Power and Energy Systems ANN based integrated security assessment of power system using parallel computing. *International Journal of Electrical Power and Energy Systems*, 42(1), pp.49–59. Available at: <http://dx.doi.org/10.1016/j.ijepes.2012.03.016>.
- Venu, B. 2011. Multi-core processor overview. *CoRR*, abs/1110.3. Available at: <http://arxiv.org/abs/1110.3535>.
- Waldrop, M., 2000. Computing’s Johnny appleseed. *MIT Technology Review*, January(1). Available at: <https://www.technologyreview.com/s/400633/computings-johnny-appleseed/> [Accessed June 22, 2014].

- Wallom, D. & Trefethen, A. 2006. OxGrid, a campus grid for the University of Oxford. *Proceedings of the UK e-science all hands meeting*. Available at: <http://www.allhands.org.uk/2006/proceedings/papers/625.pdf> [Accessed July 24, 2014].
- Wang, L., Tao, J., von Laszewski, G. & Marten, H. 2010. Multicores in cloud computing: research challenges for applications. *Journal of Computers*, 5(6), pp.958-964.
- Weitzel, D. 2011. *Campus Grids: A Framework to Facilitate Resource Sharing*. Lincoln, NE: University of Nebraska - Lincoln (Dissertation - PhD). Available at: <http://digitalcommons.unl.edu/computerscidiss/21/> [Accessed June 30, 2014].

Addendum

Max cores (48) used on Grid

Iteration	Staging time	Total Processing time	Total Job time
1	14.648016952	113.692899896	134.389706272
2	62.276286041	121.249677058	190.282781666
3	560.105986520	182.988365593	786.785846882
4	735.090274302	188.412845198	949.113712915
5	932.915595533	162.777407396	1133.790020130
Mean			638.8724136
STDEV			452.4397597

Table 16: 2K Max performance on CHPC

Iteration	Staging time	Total Processing time	Total Job time
1	732.138294725	489.379387394	1238.636366441
2	924.596851649	448.186319369	1389.519343943
3	562.715839639	503.876388457	1105.641998798
4	77.127773825	448.463137541	537.665857348
5	69.488488353	383.345892504	456.111389546
Mean			945.5149912
STDEV			422.6575049

Table 17: 6K Max performance on CHPC

Iteration	Staging time	Total Processing time	Total Job time
1	466.446504783	16001.928172227	16472.814080685
2	384.813577897	15911.438079291	16299.626238755
3	343.565054715	15910.621815498	16278.608465701
4	420.391403971	15737.474193674	16184.242215690
5	14.261718784	15371.187989658	15389.253053097
Mean			16124.90881
STDEV			424.2339234

Table 18: 12K Max performance on CHPC

8 Cores on UJ (Grid)

Iteration	Staging time	Total Processing time	Total Job time
1	14.930868454	210.421376801	230.390644800
2	15.245452582	209.901692908	230.806945858
3	14.976755125	206.737182843	229.273777935
4	15.617630616	203.780793360	226.249926905
5	14.869308311	209.959639654	229.663094066
Mean			229.2768779
STDEV			1.795354389

Table 19: 2K 8cores performance on UJ

Iteration	Staging time	Total Processing time	Total Job time
1	54.548771848	675.372869936	753.263541980
2	65.432459274	675.146960162	743.115160372
3	16.272719677	676.610776763	696.439644467
4	24.500533189	675.065086586	702.647447734
5	116.468272804	675.677136144	795.244431261
Mean			738.1420452
STDEV			40.35167503

Table 20: 6K 8 cores performance on UJ

Iteration	Staging time	Total Processing time	Total Job time
1	966.023306990	5925.446322814	6911.257096033
2	612.597920816	6366.120373200	7003.149128368
3	788.654551040	6466.835540786	7274.221020404
4	935.625138339	6448.123629323	7469.638355364
5	1030.945942726	6588.181649263	7670.820653140
Mean			7265.817251
STDEV			316.3656381

Table 21: 12K 8 cores performance on UJ

Performance on 8 Cores Local Server

Iteration	%Real_hrs	%real_sec	%System	%User	%CPU	%Max_res	%fs-inputs	%fs-outputs
1	2:55.53	175.53	9.89	883.18	508%	37104	944	24048
2	2:47.05	167.05	9.67	866.82	524%	37068	0	24928
3	2:52.42	172.42	10.10	882.29	517%	37156	24	24056
4	2:49.35	169.35	9.90	855.17	510%	37124	0	24064
5	2:49.57	69.57	10.16	886.63	528%	37172	0	24064
Mean		170.784						
STDEV		3.266263921						

Table 22: 2K 8 cores on local server

Iteration	%Real_hrs	%real_sec	%System	%User	%CPU	%Max_res	%fs-inputs	%fs-outputs
1	23:34.66	1414.66	74.43	7518.55	536%	279572	2632	68968
2	23:03.97	1383.97	72.36	7363.56	537%	279252	0	68920
3	23:11.24	1391.24	72.56	7456.06	541%	279704	0	69016
4	23:18.18	1398.18	71.87	7434.18	536%	279316	16	68936
5	23:27.56	1407.56	72.41	7504.16	538%	279684	0	68912
Mean		1399.122						
STDEV		12.30066746						

Table 23: 6K 8 cores on local server

Iteration	%Real_hrs	%real_sec	%System	%User	%CPU	%Max_res	%fs-inputs	%fs-outputs
1	1:42:50	6170.07	181.88	31804.76	518%	507312	0	137096
2	1:42:06	6126.97	184.05	31644.51	519%	504660	0	135032
3	1:44:06	6246.52	205.40	32870.94	529%	505308	8	134712
4	1:44:00	6240.87	224.67	32460.80	523%	503752	0	134664
5	1:45:53	6353.48	221.63	32716.57	518%	504208	0	139768
Mean		6227.582						
STDEV		86.33644926						

Table 24: 12K 8 cores on local server

Performance on 22 Cores Local Server

Iteration	%Real_hrs	%real_sec	%System	%User	%CPU	%Max_res	%fs-inputs	%fs-outputs
1	1:59.42	119.42	12.49	945.15	801%	37148	1232	23944
2	1:44.13	104.13	11.79	939.15	913%	37140	80	23952
3	1:43.54	103.54	11.93	944.11	923%	37148	48	23944
4	1:43.82	103.82	11.88	944.45	921%	37072	32	23952
5	1:43.68	103.68	12.35	943.04	921%	37176	56	28032
Mean		106.918						
STDEV		6.99224714						

Table 25: 2K 22 cores on local server

Iteration	%Real_hrs	%real_sec	%System	%User	%CPU	%Max_res	%fs-inputs	%fs-outputs
1	16:10.83	970.83	105.80	8947.17	932%	279332	2744	69136
2	15:52.31	952.31	100.51	8890.88	944%	279604	24	68856
3	15:48.51	948.51	98.78	8781.22	936%	279464	0	68944
4	15:55.96	955.96	101.82	8849.70	936%	279692	0	68968
5	15:56.02	956.02	102.21	8823.33	933%	279684	0	68912
Mean		956.726						
STDEV		8.470710124						

Table 26: 6K 22 cores on local server

Iteration	%Real_hrs	%real_sec	%System	%User	%CPU	%Max_res	%fs-inputs	%fs-outputs
1	1:11:47	4307.65	285.73	39065.62	913%	506052	0	134256
2	1:11:33	4293.55	288.36	39029.15	915%	504512	56	134248
3	1:11:24	4284.04	256.60	38892.31	913%	504588	1736	134472
4	1:10:32	4232.65	240.21	38570.39	916%	504772	0	143264
5	1:10:23	4223.64	245.22	38647.54	920%	503432	0	143680
Mean		4268.306						
STDEV		37.7464254						

Table 27: 12 K 22 cores on local server

Max (48 cores) performance on Grid with minimum staging time

Iteration	Staging time	Total Processing time	Total Job time
1	14.648016952	113.692899896	128.3409168
2	14.648016952	121.249677058	135.897694
3	14.648016952	182.988365593	197.6363825
4	14.648016952	188.412845198	203.0608622
5	14.648016952	162.777407396	177.4254243
Mean			168.472256
STDEV			34.63633802

Table 28: 2K Max performance on CHPC with minimum data staging time

Iteration	Staging time	Total Processing time	Total Job time
1	69.488488353	489.379387394	558.8678757
2	69.488488353	448.186319369	517.6748077
3	69.488488353	503.876388457	573.3648768
4	69.488488353	448.463137541	517.9516259
5	69.488488353	383.345892504	452.8343809
Mean			524.1387134
STDEV			46.88736463

Table 29: 6K Max performance on CHPC with minimum data staging time

Iteration	Staging time	Total Processing time	Total Job time
1	14.261718784	16001.928172227	16016.18989
2	14.261718784	15911.438079291	15925.6998
3	14.261718784	15910.621815498	15924.88353
4	14.261718784	15737.474193674	15751.73591
5	14.261718784	15371.187989658	15385.44971
Mean			15800.79177
STDEV			251.1536568

Table 30: 12K Max performance on CHPC with minimum data staging time

Performance on 8 Cores with minimum data staging time

Iteration	Staging time	Total Processing time	Total Job time
1	14.869308311	210.421376801	225.2906851
2	14.869308311	209.901692908	224.7710012
3	14.869308311	206.737182843	221.6064912
4	14.869308311	203.780793360	218.6501017
5	14.869308311	209.959639654	224.828948
Mean			223.0294454
STDEV			2.854289818

Table 31: 2K on 8cores UJ with minimum data staging time

Iteration	Staging time	Total Processing time	Total Job time
1	16.272719677	675.372869936	691.6455896
2	16.272719677	675.146960162	691.4196798
3	16.272719677	676.610776763	692.8834964
4	16.272719677	675.065086586	691.3378063
5	16.272719677	675.677136144	691.9498558
Mean			691.8472856
STDEV			0.62596939

Table 32: 6K on 8 cores with minimum data staging time

Iteration	Staging time	Total Processing time	Total Job time
1	612.597920816	5925.446322814	6538.044244
2	612.597920816	6366.120373200	6978.718294
3	612.597920816	6466.835540786	7079.433462
4	612.597920816	6448.123629323	7060.72155
5	612.597920816	6588.181649263	7200.77957
Mean			6971.539424
STDEV			255.0074642

Table 33: 12K on 8 cores with minimum data staging time

Application: preparing a specific text corpus

Iteration	Staging time	Total Processing time	Total Job time
1	265.8665020	1917.607075626	2171.798312
2	283.8684163	1887.281186185	2176.849005
3	297.8667864	1898.400678980	2219.274833
4	287.8693516	1844.824940968	2019.26736
5	294.8651648	1877.172556372	2150.585398
Mean			2147.554982
STDEV			67.90657275

Table 34: Results on the Grid (UJ)

Iteration	%Real_hrs	%real_sec	%System	%User	%CPU	%Max_res	%fs-inputs	%fs-outputs
1	14:11.61	851.61	150.36	14852.82	1761%	49084	6545136	15079440
2	14:11.24	851.24	150.54	14859.62	1763%	49088	152	15079488
3	14:06.83	846.83	151.58	14877.02	1774%	49092	120	15079480
4	14:09.35	849.35	149.10	14872.14	1768%	49068	168	15079480
5	14:12.15	852.15	150.16	14866.74	1762%	49032	1000	15079472
Mean		850.236						
STDEV		1.9465415						

Table 35: Results on the local server