Energy conservation in distributed heterogeneous computing environments using economic resource allocation mechanisms

by

Timothy Michael Lynar
BIT(Hons)

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

Doctor of Philosophy

The University of Newcastle

June, 2011

This thesis contains no material which has been accepted for the award of any other degree or diploma in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text. I give consent to this copy of my thesis, when deposited in the University Library, being made available for loan and photocopying subject to the provisions of the Copyright Act 1968.

Signature_____

Date_____

Abstract

Energy conservation in distributed heterogeneous computing environments using economic resource allocation mechanisms

by Timothy Michael Lynar

This thesis examines the question: can economic resource allocation mechanisms be used in distributed computing environments to reduce energy consumption whilst maintaining execution speed? This thesis investigates the use of several resource allocation mechanisms that take account of the power consumption and processing capacity of each available computing node within a distributed heterogeneous computing environment. Different economic resource allocation mechanisms have different attributes and allocate resources differently. The resource allocation mechanisms are evaluated to examine their effect on the time and energy required to process a workload of the sort that might be expected in a distributed computing system. Initial examination of the resource allocation mechanisms was conducted through the execution of artificial workloads on a simulated cluster. To further this research, a real cluster and grid environment was created from obsolete computers. An examination was undertaken of the use of obsolete computers in distributed computing environments and how the use of such systems may assist to mitigate electronic waste. The examination of resource allocation was continued on a cluster, and then on an institutional grid. The simulation model was then calibrated to the cluster and grid, which was then used to simulate the execution of real published grid workloads under each of the resource allocation mechanisms. The resource allocation mechanisms under consideration were found to have different characteristics that resulted in them being suited for different types of workload. It was also found that the choice of a resource allocation mechanism that takes account of the power consumption and performance of individual resources can make a significant difference, through leveraging the heterogeneous nature of resources, to the total system energy consumed and time taken in computing a workload.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

Thank you to my supervisors Dr. Ric Herbert and Dr. William Chivers, for their help and encouragement; to the Blue Gum Flats research group for their encouragement; to Simon for the help and direction he provided, and for assistance with English expression. Thank you to Steve Owers and Gary Maynard for their assistance in acquiring space and equipment. Thank you to Nic Croce and Maureen Townley-Jones from the Statistical Support Service from the Faculty of Science and Information Technology for their time and statistical advice. My sincere thanks to Rod Bell for providing invaluable feedback on this thesis. I acknowledge the help of a professional thesis editor Belinda Leskiw.

DEDICATION

To my wife Rachel,

who provided the most encouragement and support of all.

LIST OF ABBREVIATIONS

BA: Batch auction

CDA: Continuous double auction

CLUSTER: Computing cluster

CRA: Continuous random allocation

DVFS: Dynamic voltage and frequency scaling

EOE: End of execution

E-WASTE: Electronic waste

FLOPS: Floating point operations per second

FOSS: Free and open source software

GT: Grid trace

HPC: High-performance computing

HPDC: High-performance distributed computing

HPL: High-performance Linpack benchmark

MFLOPS: Millions of floating point operations per second

MIM: Marginal increase mechanism

MPI: Message passing interface

MWIPS: Millions of whetstone instructions per second

NODE: Computing node

PPBA: Pre-processed batch auction

PXE: Pre-execution environment

SPMD: Single process, multiple data

TTT: Test task time

UML: Unified modelling language

VFS: Voltage and frequency scaling

VOVO: Variable-on variable-off

WF: Workflow

WIPS: Whetstone instructions per second

ZIT: Zero intelligence trader

PUBLICATIONS RESULTING FROM THIS RESEARCH

Fully refereed journal articles

Lynar, T., Simon, Herbert, R. D., & Chivers, W. J. (2011). Resource allocation to conserve energy in distributed computing. International Journal of Grid and Utility Computing (IJGUC), 2 (1), 1-10.

Lynar, T. M., Herbert, R. D., Simon, & Chivers, W. J. (2010). Clustering obsolete computers to reduce e-waste. International Journal of Information Systems and Social Change, 1(1), 1-10.

Lynar, T. M., Herbert, R. D., & Simon. (2009). Auction resource allocation mechanisms in grids of heterogeneous computers, WSEAS Transactions on Computers, 10, 1671-1680.

Fully refereed conference proceedings

Lynar, T., Simon, Herbert, R. D., & Chivers, W. J. (2010). Reducing grid energy consumption through choice of resource allocation method. The 6th Workshop on High-Performance, Power-Aware Computing (HPPAC). Atlanta, GA, USA. April 19.

Lynar, T. M., Herbert, R. D., Simon, & Chivers, W. J. (2009). Impact of node ranking on outcomes of grid resource allocation. The 2009 International Conference on Grid Computing and Applications (GCA09). Las Vegas, NV, USA. July 13-16.

Lynar, T. M., Herbert, R. D., Simon, & Chivers, W. J. (2009). A grid resource allocation mechanism for heterogeneous e-waste computers. The 7th Australasian Symposium on grid computing and e-research (Ausgrid 2009). Wellington, New Zealand. January 20–23.

Herbert, R. D., & Lynar, T. M. (2009). A comparison of economic resource allocation mechanisms in grids of e-waste computers. The proceedings of the 9th WSEAS International Conference on Simulation, Modeling and Optimization (SMO'09). Budapest, Hungary. September 3–5.

Lynar, T. M., & Herbert, R. D. (2009). Allocating grid resources for speed and energy conservation. The 6th International Conference on Information Technology and Applications (ICITA09). Hanoi University of Technology. Hanoi, Vietnam. 9–12 November.

Lynar, T. M., Herbert, R. D., Simon, & Chivers, W. J. (2009). Why decide: Is a user's estimation of job completion time useful in grid resource allocation? The 18th World IMACS Congress and International Congress on Modelling and Simulation (MODSIM09). Cairns, QLD, Australia. 13–17 July.

Chapter 1

INTRODUCTION

1.1 Overview and purpose

Processor-intensive applications such as computer modelling have historically been processed on supercomputers, however, this high performance is costly (Schneck, 1990). Grid computing promises to be one way of approaching the computing power of a supercomputer at a relatively low cost.

Distributed computing environments such as grids consume considerable amounts of energy, and the environmental and financial costs of energy are substantial. Energy consumption is therefore a consideration of growing importance in this area of computing.

This thesis explores the question: can economic resource allocation mechanisms be used in distributed computing environments to reduce energy consumption whilst maintaining execution speed?

This research uses economic resource allocation mechanisms that take account of the power and performance data of nodes. By economic resource allocation mechanisms, it is meant mechanisms that incorporate economic principles for the allocation of resources; in this research auctions are used as the method of economic resource allocation.

Several resource allocation mechanisms are examined in relation to outcomes of energy consumption and speed of task execution for allocation of distributed resources. The mechanisms allocate according to the energy consumption and processing capacity of individual

nodes, with the aim of conserving energy through resource allocation alone. The mechanisms do not require exclusive control over individual resources in the distributed environment, nor do they require additional hardware. To examine the effect of these mechanisms on energy usage and the speed of execution, they are applied to various processing workflows through simulation and experimentation on a cluster and a grid.

The main original contribution of this thesis is the creation of a novel resource allocation mechanism that utilises economic resource allocation methods to conserve energy and maintain the speed of execution in distributed computing environments such as grids. Publications resulting from this research are listed on page xxi. Other key contributions of this thesis are described in Section 1.3.

Findings include the use of economic resource allocation mechanisms that take account of the power and performance of individual resources can make a significant difference to total system energy consumption; and that different economic resource allocation mechanisms have different characteristics that result in them being better suited to different types of workflow. The selection of the resource allocation mechanism can make a significant difference both to the time taken to execute a workflow and to the energy consumed in its execution.

1.2 Structure

The focus of this research is the use of economic resource allocation to conserve energy whilst maintaining speed in distributed computing environments. The distributed computing environments examined are those that are primarily used for their processing ability. For economic resource allocation to work, either nodes or tasks, or both, need values. In this research nodes are given a value based on their desirability. A node's desirability is based on its attributes. Chapter 3 examines different ways in which nodes can be valued and therefore ranked based on their attributes. Next, in Chapter 4, a simulation is analysed to examine the use of auctions as a form of resource allocation in distributed computing

environments. The simulation showed positive results; subsequently two distributed computing environments were constructed (Chapter 5) to test a real implementation of the effectiveness of these resource allocation mechanisms. Testing on a cluster and a grid environment is presented in Chapter 6. The initial simulation needed to be calibrated as it was made prior to the distributed resources being constructed. The calibration of the simulation is presented in Chapter 7. In Chapter 8 the calibrated simulation is used to extend the research through the examination of the resource allocation mechanisms through the simulated execution of workflows derived from well known grid traces.

The structure of each chapter will now be described in detail.

A review of the literature is presented in Chapter 2. It examines the work that has already been achieved in the field and concludes that there is a need to perform additional work in the area of energy conservation in distributed computing environments. This leads to the examination of the effect of economic resource allocation on energy consumption and the speed of execution in distributed computing environments. The literature review covers five areas of importance to this thesis.

The first topic in the literature review covers the use of agent-based models as a tool for inquiry. Agent-based modelling is a method of simulating a phenomenon from the bottom up, and is used extensively as a form of inquiry in this research. These models consist of many autonomous agents that interact with one another. It is through the study of these micro behaviours that we see the emergence of macro behaviours. The focus of this section of the literature review is on the merits of using agent-based models in research such as this.

The second topic examines the literature in the field of resource allocation mechanisms in distributed computing environments. This topic is covered with three focuses; economic resource allocation mechanisms for distributed computing environments, the use of agents in resource allocation, and the use of auctions to perform resource allocation.

The third topic looks at electronic waste (e-waste), that is electronic goods that have been

discarded or have entered a period of disuse. It discusses the growing problem that e-waste is to nations and the information technology industry. It examines what is known about the disposal methods of e-waste, the export of e-waste, the security issues associated with e-waste and the environmental consequences of e-waste. The distributed computing environments used in this research were constructed entirely from e-waste resources.

The fourth topic examines literature on computing projects that have utilised e-waste resources for their computational performance. The focus of this section is the Stone Souper-Computer, which was a cluster made from disused commodity computers, and then used for the processing of models.

The fifth section of this literature review examines literature on current methods of saving energy or reducing power consumption in distributed computing environments. The literature discusses why energy savings in such environments are important, the quantity of energy used, and various methods currently used to conserve energy. These methods are divided into uncoordinated methods that are employed individually on computing nodes in isolation, and coordinated methods that are employed in cooperation with other resources within the distributed computing environment.

Lastly, benchmarks are discussed. Benchmarks are often used as a method of ranking computing resources. The literature discusses the use of synthetic and non-synthetic benchmarks and the problems associated with them. Benchmarks are included in the literature review as the first research chapter of this thesis examines the impact of different node ranking methods on energy consumption and the speed of execution. In this thesis these methods are referred to as performance metrics.

The literature review identifies that some work has been done on energy conservation in distributed computing environments but there is room for further research. There has been little research on energy aware grid resource allocation, and the use of simple economic resource allocation mechanisms to conserve energy in distributed computing environments is a novel idea. This thesis examines the creation of a novel resource allocation mechanism

that utilises economic resource allocation methods to conserve energy whilst maintaining the speed of task execution in distributed computing environments such as grids.

With regards to the structure of the research chapters of this thesis; initially the examination of resource allocation occurs through simulation. For a resource allocation mechanism to prefer one resource over another it needs some metric by which to rank them. Prior to examining different resource allocation mechanisms, a simulation was created to examine the effect of different performance metrics used to rank nodes for resource allocation. The research presented in Chapter 3 introduces a novel performance metric, and uses a simulation to explore the impact of choosing between six different node performance metrics. It examines the effect that choice of metric can have on energy consumption and on the speed of execution. Initially the chapter discusses why a measure of performance of computing resources is needed. The need for a measure that accounts for both a node's power consumption and a node's computational ability are discussed at length. Current measures of a node's performance are discussed, including measures that account for power consumption and measures that account for computational ability. The performance measures that were taken for each node and the performance metrics that are based on these measures are described. The performance metrics are then analysed based on data collected from a number of real computing nodes. These performance metrics are examined in terms of the relative ranks they give to the different nodes. These ranks work by giving nodes that perform better higher values, that translate into higher ranks. Some metrics rank nodes purely on a measure of computational performance, some purely on power consumption, and others on both. An agent-based simulation is described as a method to examine the effect that the use of these different performance metrics can have on total energy usage and the time taken to execute tasks in a distributed computational environment. The simulation is constructed of a distributed computational environment with many nodes, these nodes are constructed based on the readings of real nodes. The effectiveness of the performance metrics is examined through the simulated execution of several synthetic workflows. A simple batch auction (BA) is used to allocate tasks to the available nodes with the highest performance metric value. This simulation is repeated for each workflow with each of

the described performance metrics. This chapter examines the means by which nodes are ranked for resource allocation; it examines what effect the selection of a performance metric has on energy consumption and the speed of execution.

A simulation study is presented in Chapter 4 to examine whether resource allocation alone can help to conserve energy in distributed computing environments such as grids and clusters. To examine the effect of resource allocation, a simulation was created of the allocation of a variety of workflows over a distributed resource, using a number of different economic resource allocation mechanisms. The focus of the study was on the effect that differing economic resource allocation mechanisms can have on energy consumption and speed of execution in high-performance distributed computing environments. Initially the chapter describes the agent-based simulation model that is used and how the model is set-up. The hardware that is simulated is then discussed. The method used to obtain readings of the node's computational ability and power consumption at different power states is described and it is explained how this information is incorporated into the simulation model.

The auctions that are used as the basis for resource allocation are described. These auctions are: the batch auction (BA), the continuous random allocation (CRA), and the pre-processed batch auction (PPBA). The BA sends a request to every node for a bid and waits until it receives a response from each resource before allocating a task; each node's bid is based on its performance metric value, with tasks allocated to the highest bidder. The CRA allocates tasks to a random node that is currently available. The PPBA, like the BA, asks all nodes for a bid, but unlike the BA it allocates based on historical information before it receives any responses. Different auctions have different characteristics; these characteristics include varying success at selecting the best available resource, and the speed with which it selects a resource. It is suspected that different auctions may be better suited to different workflows.

In addition to resource allocation mechanisms the use of variable-on variable-off (VOVO) is also described and included in the simulation. VOVO is used to turn idle nodes off during periods where the distributed resource is under-utilised. A simple implementation

of VOVO is included to examine if the methods can work collaboratively to reduce energy consumption whilst maintaining execution speed.

Once these resource allocation mechanisms are discussed, the tasks and workflows that are used to test the resource allocation mechanisms are described. The use of a number of synthetic workflows and a grid trace are justified. The methodology used for assessing the research question in this chapter is then presented.

In order to examine the use of economic resource allocation techniques in distributed computing environments, exclusive access is required to such environments. Chapter 5 examines the creation of distributed computing resources using computers that were to be discarded as e-waste, evaluating the creation of a cluster of computers that had reached their end of life and were otherwise going to be sent for recycling. Chapter 5 examines how clusters can help to mitigate e-waste, and the advantages and limitations of using such a system. E-waste resources were utilised in the distributed computing environments used in this research. Initially the chapter describes a cluster test platform that was constructed from e-waste resources. The hardware resources used are described, as is the set-up of the hardware and software resources. The environmental costs associated with e-waste are discussed and reasoning behind the use of e-waste resources in distributed computing environments is explored. Advantages of using such a system, such as the cost and the saving of resources, are weighed up against the disadvantages such as obsolescence and diminished reliability. It is possible, and indeed in some situations desirable, to use e-waste resources as computational grids or clusters. This chapter describes how such a resource was created and the reasoning behind its creation.

Chapter 6 describes the implementation of the previously simulated resource allocation mechanisms on real environments. Energy consumption and time are once again analysed. A cluster is used as a proof of concept pilot because it provided a controlled environment with exclusive access and on which accurate measures could be taken of the effects of altering the resource allocation mechanism. After the tests were performed on this pilot cluster environment, a grid of three clusters was created, and the same tests were performed.

Chapter 6 presents both sets of results and compares them. Firstly, a description of the resource allocation mechanisms as they are implemented is made, and the differences between the simulated model and the implementation are described. The performance metrics are briefly discussed at this point and analysed later in the chapter along with the resource allocation mechanisms. A technical description of the application used to implement resource allocation on the cluster and grid environments is presented. The methodology is then expounded, including a description of the synthetic workflows used, a discussion of the calibration of tasks, and a description of the testing equipment and testing procedure used. Both resource allocation mechanisms and performance metrics are examined. The effectiveness of the resource allocation mechanisms at reducing energy consumption and maintaining speed are explored on both a cluster and a grid environment.

Chapter 7 describes the calibration of the simulation described in Chapter 4. The simulation was initially of a cluster environment. This chapter examines the calibration of the simulation to both the cluster and the grid used and the results from this calibrated simulation are compared to the results observed from the real environments.

Initially the findings of the simulation presented in Chapter 4 are compared against the findings from the pilot cluster environment. There is then a description of a number of adjustments that were made to the model for it to become a better analogue to both the cluster and the grid environments. These adjustments include usage of identical workflows, calibration of the simulated tasks, and calibration of the simulated nodes. A methodology is then presented for determining if the simulation produces results that are the same as the results produced by the real cluster and grid environments. The results from the calibrated simulation are compared to the results of the cluster and grid environments to determine if the same phenomena could be observed on both, and to determine if there was a significant difference between the simulated and the observed results. The initial simulation model is now calibrated to the two environments used in the experiments. This simulation uses the same synthetic workflows and contains the same nodes for each environment. The analysis of this simulation suggests that although it is not a perfect analogue, the simulation produces the same phenomena.

The calibrated simulation is then examined further in Chapter 8. This chapter details and examines a number of extensions to the model to aid in the understanding of the use of resource allocation to conserve energy whilst maintaining execution speed. The simulation of a number of well known grid workflows is presented and additional resource allocation mechanisms are explored. To begin with, the chapter describes the addition of a further performance metric and a further resource allocation mechanism. Instead of allocating the task to the node with the greatest ratio of computational performance to power consumption, a new mechanism is implemented that allocates the task to the node that has the greatest ratio of performance to marginal power consumption; where marginal power consumption is the difference in a node's power consumption when idle to power consumption when processing.

An implementation of a continuous double auction (CDA) is also examined. The CDA as implemented is stochastic; but unlike the CRA, rather than allocating a task to the first node to bid, it waits until two nodes have bid and allocates the task to the node with the higher bid.

A methodology is then presented for the examination of all resource allocation mechanisms through the calibrated simulation. The calibrated simulation is set-up to simulate the execution of both synthetic workflows and a number of workflows derived from well known grid traces.

Chapter 9 summarises this thesis, discusses the key contributions of this research and discusses potential extensions of this research.

1.3 Key contributions

The prime focus of this thesis is on the use of economic resource allocation in distributed computing environments. The main original contribution of this thesis is the creation of a novel resource allocation mechanism that utilises economic resource allocation methods to conserve energy and maintain the speed of execution in distributed computing environments

such as grids. This thesis makes a number of key contributions to the fields of distributed computing, resource allocation, and energy conservation. These key contributions will be briefly described below, and described in further detail throughout the thesis.

A novel performance metric for heterogeneous computers and an examination of the effect that different node ranking methods can have on total energy consumption and the time taken to execute workflows

In order to use economic resource allocation that accounts for the heterogeneous nature of the computing resources, including energy consumed and computational performance, novel performance metrics were created and compared. These metrics use data regarding each node to objectively compare the nodes and rank them accordingly. The ranking was designed so that nodes with the most desirable attributes (high computational performance to energy consumption ratio) would receive the highest rank. This method ensures that the most desirable nodes have a bidding advantage in the resource allocation auctions, which thereby ensures that the most capable nodes will process the most tasks, hence optimising execution speed whilst minimising energy consumption of the cluster or grid. The findings of this thesis show that accounting for the individual attributes of heterogeneous nodes in resource allocation can make a significant positive difference to the time it takes to execute workflows and the energy consumption of the system.

A novel resource allocation mechanism, incorporating the attributes of individual nodes, that can apply different resource allocation strategies with a view to minimising both energy consumption and the time taken to execute workflows

Several auctions were studied and compared for the purpose of resource allocation on clusters and grids, through simulation and experimentation. The BA, CRA, and the PPBA were primarily used. Although the BA and PPBA auctions were found to be most suitable most often, each auction type was found to be most successful in specific circumstances.

This finding supports the use of resource allocation which accounts for the attributes of the nodes and also the workload to ensure the resource allocation is continuously up to date and achieving optimum allocation.

A study of the effectiveness of the different resource allocation mechanisms through simulation and through experimentation on a cluster and grid environment

Simulation was used as an initial proof of concept. After positive results the resource allocation mechanisms were then implemented on a physical cluster as a further proof of concept, before being scaled up to a grid. A comparison of the results and associated statistical analysis shows that the results at each phase display similar phenomena and that use of simulation is an accurate and valuable method in research such as this. The results from the implementation on the grid were compared to the results from the cluster. The simulation model was then calibrated to both the cluster and the grid, and used to simulate real grid workflows. Related contributions were made as a result including: a comparative examination of a grid resource allocation mechanism on a heterogeneous cluster and on a grid environment; a calibrated simulation of the execution of a number of grid workflows; and an analysis of the simulated execution of a real grid workflow.

A study on the use of grids to mitigate e-waste and the advantages and limitations of such a process

All of the hardware used in this research was once e-waste, destined for recycling and obtained at no cost to the author. This research has shown that it is possible to create usable distributed computing resources from e-waste, and that in limited circumstances it may be viable to utilise e-waste clusters and even grids in place of clusters and grids made from new hardware.

1.4 Definition of key terms

There are various possible definitions for many terms used throughout this thesis. This section defines key terms in the manner they are used in this thesis.

1.4.1 Agent-based models

Agent-based models are used as the basis of the simulations used in this thesis. Agent-based models are computer-based simulations that model a situation or phenomenon from the bottom up using 'agents' (Gilbert and Terna, 2000). Agents are processes that are implemented on a computer and have the ability to interact with other agents and the environment, and can be both proactive and reactive (Gilbert and Terna, 2000).

1.4.2 Distributed computing environments

As referred to in this thesis, a distributed computing environment is a computing environment that contains many autonomous computing resources that can communicate with one another to achieve a common computational goal. In this research the computational goal is computing all tasks in a workflow, where each task can be processed in whole on a single computing resource. These environments include clusters, grids, and volunteer computing environments. The distributed computing environments that this thesis is concerned with are those that are primarily used for their computational ability rather than storage.

1.4.3 Economic resource allocation

Economic resource allocation mechanisms are resource allocation mechanisms that incorporate economic principles. In this thesis several auctions are examined.

1.4.4 Grid

Grid computing has been defined by many researchers. Foster et al. describes a computing grid conceptually as: "coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations" (Foster et al., 2001, p.200). Computational grids have also been likened to the power grid where users and resources are distributed and users have access to a dependable and consistent supply of computational power (Foster and Kesselman, 2001). Researchers such as Tarricone and Esposito note that grids can come in a variety of sizes from small local grids to large grids which span multiple geographic locations: "a grid can span domains of different dimensions, starting from local grids made up of nodes connected by LANs[local area networks], up to global grids, made up of heterogeneous nodes owned by different organizations and connected by the internet" (Tarricone and Esposito, 2004, p.15). Wells (2008) clarifies the issue by further defining grids, based on scope. A grid can either be a cluster grid, an enterprise grid, or a global grid. The view that a grid is a cluster of clusters is true in many academic environments. Plaszczak and Wellner Jr. state that: "Especially in the academic environment, grid systems often are based on a group of clusters, interconnected by network and certain middleware" (Plaszczak and Wellner Jr., 2006, p.61). Grids are not exclusively clusters of clusters, but can incorporate them, and so too does the definition that will be used throughout this research, which is as follows: a hardware and software resource that provides dependable, constant, pervasive and transparent access to distributed resources, that may be owned, shared, and utilised by multiple institutions and their users. The grid environment experimented on in Chapter 6 is a project grid as defined by Wells (2008). It is essentially a cluster of clusters that can be accessed in a decentralised manner.

1.4.5 Performance measure

A value that is obtained through an observation or test of performance. Performance measures used in this research include: power usage, flops, wips, and the time taken to execute a task.

1.4.6 Performance metrics

A system of measuring performance. The metrics used typically involve a calculation based on several performance measures.

1.4.7 Task

A task is a computer application that requires processing. The tasks used in this thesis are applications that run on a single processor but are executed multiple times, and can be executed with different input parameters. The primary task used in this research is a prime number search script. This thesis does not explore the scheduling of the individual threads of multithreaded applications, nor does it examine the allocation of tasks that require substantial input data.

1.4.8 Workflow

A workflow is a series of tasks that are submitted in a set sequence at set instances in time to computational resources for processing.

Chapter 2

LITERATURE REVIEW

2.1 Introduction

The literature review is divided into several different topics. First, agent-based models are reviewed, followed by resource allocation, with a particular focus on economic resource allocation methods. The issue of e-waste is then discussed, with a focus on disused personal computers, and a review of projects that incorporate e-waste resources. The literature review then focuses on energy consumption in distributed computing. Multiple techniques of reducing energy consumption in high-performance distributed computing (HPDC) environments are examined, and it is concluded that resource allocation has the potential to further reduce energy consumption. Finally, an examination of different benchmarks is performed. Substantial gaps in the literature are evident in the areas of high-performance energy-aware grid computing. This thesis aims to address some of these gaps.

2.2 Agent-based models

Agent-based models are computer-based simulations that model a situation or phenomena from the bottom up using 'agents' (Gilbert and Terna, 2000). An agent-based model simulates micro behaviours in order to produce emerging macro phenomena. Agents are processes that are implemented on a computer and have autonomy, the ability to interact with other agents and the environment, and are both proactive and reactive (Gilbert and Terna, 2000). Agents are not controlled by any external coordinating device (Gilbert and Terna, 2000; Tesfatsion, 2002).

The study of agent-based models presents two common themes. First, agent-based models can be accurate at representing a situation because they deal with individuals and their characteristics; second, agent-based models are complex and therefore processor-intensive applications.

Agent-based models can be accurate at representing social and economic phenomena, and in some situations have the potential to produce more accurate results than traditional models. This is because they can simulate complex social and economic situations at the micro level, and can often more easily account for heterogeneous variability (Chivers and Herbert, 2003; Gilbert and Terna, 2000).

Agent-based, statistical, and mathematical models all have the ability to represent extremely complex situations. Gilbert and Terna argue that although statistical and mathematical models have the ability to model complex phenomena they are "simply too complicated to be analytically tractable" (2000, p.3). They argue that agent-based models do not seem to have these limitations, that they can be quite complex while remaining analytically tractable (Gilbert and Terna, 2000). Chivers and Herbert observe that "The cost of the utility of the individual based approach is the complexity of the model itself" (2003, p.441). This complexity is something that needs to be contained. Mizuno and Nishiyama (2003) suggest that as a general rule it is best to keep the complexity of the model as simple as possible without sacrificing detail.

The complexity of the agent-based model can be seen as a product of the modelling of the micro behaviours in the system. Through the study of the micro behaviours we are able to see the emergence of macro behaviours, and the macro behaviours are easily tractable. Mizuno and Nishiyama state that "this model can, however, capture the essence of the macro behaviours that emerge from the interaction of the simple micro behaviours of agents" (2003, p.361). Chivers and Herbert appear to agree with Mizuno and Nishiyama (2003) and argue that this type of model has the ability to simulate not just a generalisation of a system's dynamics but the mechanics of the system itself. They argue that agent-based models can look at individual behaviour, learning, interactions, and environmental

differences. Tesfatsion (2001) also notes that agent-based models enable the designer to build a world that can incorporate evolutionary processes, and observe the created world without impacting upon it.

Chivers and Herbert argue that agent-based models can also take into account individual variability: "The details of the interaction between individuals and the different experiences of the individuals can have a significant effect on the overall system dynamics" (2003, p.442). Although agent-based models account for individual variability and are a useful form of modelling, they are more difficult than analytical models to replicate, communicate, and generalise results from (Kwaśnicki, 1999).

Agent-based models are able to accurately simulate a phenomenon, and may even simulate some situations more accurately than alternative techniques, because they not only look at the overall trend but also consider the micro interactions that occur, and account for individual variability. Macro behaviours emerge from this simulation of micro interactions. Because agent-based models simulate at the micro level, the complexity of these interactions can be great, though the observed macro interactions may still be simple and tractable.

2.2.1 Replication and reliability

Replication and reliability are two prominent issues in the creation of agent-based models. The ability to replicate an experiment is important; if an experiment cannot be replicated then the reliability of that experiment is put into question. Leon et al. point out that currently "there is no established methodology for the design of ABMs [Agent-Based Models]" (2003, p.332). Gilbert and Terna add that "There is no one recognised best way of building agents for a multi-agent system" (2000, p.9). One proposed solution to this problem of a lack of standards comes from Leon et al., who propose the use of a framework because, they argue, without one "it is difficult to validate the accuracy of these models" (2003, p. 331). Although many agree that a framework is a necessity for the construction of reliable models there are those who disagree. Gilbert and Terna argue that because an agent-based model

is a computer program it is inherently communicable to others and therefore testable, stating: "Once one has a model, the fact that it is formulated as a computer program will, in principle at least, allow it to be communicated to others" (2000, p. 6). However, they later concede that although a model is a computer program it does not guarantee portability, stating that "programs are often not easily portable from one machine and one operating system to another" (2000, p.6). This lack of portability creates an obstacle: if the models are not portable then they are not necessarily communicable, and as such require a way to make them portable. A clearly defined framework can assist in the reproduction of models. Two popular frameworks currently being developed and employed are Computational Laboratories (McFadzean et al., 2001) and SimBioSys (McFadzean and Tesfatsion, 1999). It is important for those who wish to create an agent-based model to address these two key issues of a) model replicability and b) the ability for others to examine the model for reliability issues. In order to do this a model must be portable.

2.2.2 Agent-based models are complex and processor-intensive

Agent-based models have the potential to be processor-intensive applications. One such application models the movement of sixteen thousand chickens in a pen. According to Goldstein the processor power required to run this simulation was "a chore that would tax a rack full of conventional servers" (Goldstein, 2007, p.37). Historically, processor-intensive applications such as computer modelling have been carried out on supercomputers, but this high performance is costly: "Supercomputers yield the highest performance feasible, although the cost of a supercomputer computation is greater than that of alternative systems" (Schneck, 1990, p. 14). The study of agent-based models has shown the enormity of computing power required to run models in a timely fashion.

2.2.3 Current examples of agent-based models in business

Agent-based models have many real-world applications, and have been used to model economic, social and ecological situations. This section examines recent agent-based models

and their outcomes.

Cartier (2004) conducted a study of two car makers, Renault and Ford, and their internal innovation strategies. An agent-based model was built, based on a genetic algorithm to simulate the innovation process in the organisations. The overriding purpose of the study was "to determine the link between the intensity of internal mechanisms of evolution and performance of the organization ...." (Cartier, 2004, p.147). Cartier discovered that a "high level of selection leads to regular improvement but also rapid fall of diversity" (Cartier, 2004, p.151). This study is a good example of how agent-based modelling can be used in real-world situations in order to find a solution to a common problem. The problem in this case was determining how much innovation, and what type of innovation, is beneficial to a company.

McFadzean and Tesfatsion (1997) describe an attempt to create a new framework for studying trade networks. The proposed framework incorporates an evolutionary trade network game, Prisoner's dilemma, and contains autonomous, endogenously interacting 'TradeBots'. Overall it is a very complex system that aims not just to look at economic outcomes but also to study the sociological, psychological, biological and physical conditions of the traders. This article provides a specific framework that has been developed for agent-based modelling of trade networks. McFadzean and Tesfatsion's (1997) model provided large amounts of information at the micro level so as to allow the modeller to view the formation of macro-level phenomena, and the framework presented offers promise to those who wish to model trade network games.

Herbert and Turton (2007) investigated the issue of which simple auction strategy was most effective in an auction. Three simple auction strategies were tested against one another, and they found that one auction strategy consistently won and another strategy consistently failed. Herbert and Turton's (2007) findings have important implications for the present project. An auction is a resource allocation mechanism like any other, but it is clear that in auctions strategies can play an important part in determining which resources are allocated to whom. As in the aforementioned work agent-based models have been utilised in this

research.

2.3 Resource allocation mechanisms in distributed computing environments

This section of the literature review examines different methods of allocating distributed resources. The focus is on economic resource allocation and agent-based resource allocation for HPC environments. Auctions and other methods of economic resource allocation are discussed, with a particular focus on their use in grid computing.

2.3.1 Economic resource allocation

A number of published resource allocation mechanisms incorporate economic principles (AuYoung et al., 2004; Buyya and Murshed, 2002; Buyya et al., 2005; Grosu and Das, 2004; Lai et al., 2005; Li and Li, 2004; Stuer et al., 2007; Yamamoto et al., 2006). Resources are typically given a value by the resource users, the resource providers, or both. This value is often related to the scarcity of the resource and/or the utility that a particular resource could provide to the bidding agent. In some papers it is not clear how the resources are valued, but they are given a value in monetary or barter terms. Buyya et al. identify a wide range of different economic models that have been used for grid resource allocation, including the Commodity Market Model, the Posted Price Model, the Bargaining Model, the Tendering/Contract-Net Model, the Auction Model, the Bid-based Proportional Resource Sharing Model, and the Community/Coalition/Bartering Model (Buyya et al., 2002).

A resource is typically allocated to a resource consumer through an auction, or through other means of negotiated settlement (Stuer et al., 2007). One of the primary benefits of economic based methods of resource allocation is that they can be decentralised, providing a good conceptual fit with grid resource allocation, which is also decentralised (Yamamoto et al., 2006).

2.3.2 Auctions as a means of resource allocation in grid computing

There has been extensive research into the use of auctions as a means of resource allocation for distributed computing environments (Zhao et al., 2006). This section examines the extent to which auctions have been used in grid resource allocation mechanisms, what types of auctions have been used in those mechanisms, and how and why auctions have been used.

Auctions have been used considerably, in both continuous and batch scenarios, for allocation of resources in grid computing environments. The auction model of resource allocation is a one-to-many negotiation that reduces the negotiating to a single value, which is typically a price (Buyya et al., 2002).

The literature includes a number of market-based resource allocation mechanisms that use auctions. There are many potential reasons for the interest researchers have shown in auctions as a means of grid resource allocation. Auctions have numerous benefits for resource allocation, although they also have some undesirable effects (Zhao et al., 2006). One benefit of using an auction in a resource allocation mechanism was stated by Zhao et al.: "auctions provide a convenient, straightforward mechanism for clearing the marketplace" (2006, p.269). However, they also note as a key disadvantage of auctions the uncertainty inherent in their use: "an auction is more unreliable in terms of both pricing and the ability to obtain a resource, and may therefore result in poor scheduling decisions and more inefficiency for consumers" (Zhao et al., 2006, p.270). Even so, the degree to which this statement holds true is no doubt dependent on the type of auction and the way the auction is being implemented.

Auctions as a method of resource allocation are used by a number of grid systems, including REXEC, CORA, and GridIS.

REXEC (Chun and Culler, 2000) is a secure remote execution environment which uses a semi-centralised bidding system. Users contact a server, which then contacts the nodes to see what resources are available. This is a semi-centralised system as resource requests to

the many nodes are made through a smaller number of servers (Chun and Culler, 2000). Clearing of the resources is performed by each of the servers through an auction.

The Coallocative, Oversubscribing Resource Allocation framework (CORA) also utilises auctions as a means of resource allocation (Bubendorfer and Thomson, 2006). The creators of CORA noted the many benefits of using auctions and indicated that auctions and other market-based allocation mechanisms are in common use for scalable resource allocation because "they are naturally decentralised, efficient and produce optimal allocations" (Bubendorfer and Thomson, 2006, p.1).

The GridIS system uses a continuous double auction (CDA). It is claimed that the benefit of using this auction as a resource allocation system is that it encourages both users and consumers to stay involved in the system through a monetary incentive (Xiao et al., 2005).

The literature shows that auctions have been used extensively in grid resource allocation mechanisms. There are many claimed benefits to using auctions as a means of resource allocation. There are also a variety of different auctions with different properties, characteristics, and benefits.

Types of auction

The many types of auction can be classified according to four main dimensions: winner determination, payment determination, the availability of information, and the mechanism used to perform bidding (Wooldridge, 2002).

Winner determination in auctions is not always to the participant that bids the highest amount; however, it is common for the winner to be the auction participant that bids the highest amount (Wooldridge, 2002).

Another feature that differs among auctions is the method of determining the amount to be paid by the winning participant. Two common types of auction are first price and second price auctions. In first price auctions the participant who bids the highest pays the

amount they bid, whereas in second price auctions, often referred to as Vickrey auctions, the participant who bids the highest then pays the amount of the next highest bid. There are also auctions in which the winning participant pays a price between the highest and second highest bids, or where the winning participant pays the amount they bid minus a specified discount (Byde, 2003; Wooldridge, 2002; Bubendorfer and Thomson, 2006).

The third dimension of an auction is the availability to participants of information related to the auction. Auctions are either 'open cry', where all participants are privy to the values of all bids, or 'sealed bid', where the values of bids are kept secret, so that bidders know only their private valuation of the resources, and only the auctioneer knows the value of each bid (Wooldridge, 2002).

The fourth dimension of an auction is the bidding mechanism. Bidding can typically be single shot, ascending, or descending. In a single shot auction each participant is allowed to bid only once. In ascending auctions the auctioneer sets the initial asking price low and incrementally increases the price. In descending auctions the price is initially set high and decreases incrementally until someone bids (Wooldridge, 2002).

Although it is not a dimension of an auction, the way in which the auction is executed is also important. Auctions can be run either in a continuous fashion, whereby bids and asks can constantly be made, or in a batch fashion, whereby there is a limited time for the auction to execute.

There are a variety of common types of auction explored in the literature as useful for grid resource allocation. These auctions include CDA, first price auctions, and Vickrey auctions.

Hajiaghayi (2005) tested a number of common auction types and assessed their usefulness in relation to the allocation of grid resources. Hajiaghayi's simulations indicated that the Vickrey-Clarke-Groves auction, sometimes referred to as the Vickrey auction, is suitable for use in a grid resource allocation mechanism. Hajiaghayi states that "no deterministic trueful online auction has revenue that is constant-competitive with that of the offline Vickrey-Clarke-Groves (VCG) mechanism" (2005, p.165). However, the Vickrey auction

would require batch processing of jobs rather than processing the auction in a continuous fashion.

In contrast, Porter (2004) constructed a formula for an auction in real time between self-interested agents. The objective of the auction was to allocate grid resources in a way that would maximise the number of jobs completed by their deadline. Porter's (2004) auction appears to achieve this goal in an efficient manner.

Other researchers have focused on comparing methods of resource allocation. Krawczyk and Bubendorfer tested an auction against various other forms of resource allocation to answer the question"what impact does the choice of resource allocation mechanism have on the target grid system?" Simulation was employed as a methodology to answer this question. A number of resource allocation mechanisms were simulated, including volunteer pooling, fair-share, and auctions. The auction chosen was the reverse first price sealed auction (RFPSA) (Krawczyk and Bubendorfer, 2008). The auction performed comparatively well: there was an initial delay caused by bidding, which was due to the batch nature of the auction, but the impact of this delay was lost after the jobs were distributed to their queues. Nevertheless, the authors recommended that "an auction should preferably be deployed in a continuous scenario" (2008, p.78). This study also showed that the particular auction chosen is effective at distributing jobs. Krawczyk and Bubendorfer state "This indicates that the queuing times at the resources are more consistent and that the load is significantly better distributed to resources than for the other two schedulers" (2008, p.78). It was discovered that auctions perform well against other forms of resource allocation; however,

> "In general auctions produced a slow start to a batch execution, although their turnaround times were very stable and useful if this was a QoS [Quality of service] parameter. This finding alone suggests that auction based resource allocation is best deployed in a continuous allocation scenario. In a burst scenario one of the other allocation mechanisms would return better overall utilisation" (Krawczyk and Bubendorfer, 2008, p.66).

Tan and Gurd (2007) appear to agree with Krawczyk and Bubendorfer's (2008) recommendation that auctions of grid resources should be run in a continuous fashion. Tan and Gurd argued that a CDA or similar is the only acceptable market based resource allocation mechanism that should be used for grid resource allocation, stating that:

> "Continuous Double Auction (CDA) is the most appropriate of the standard market models for Grid resource allocation. ... it is simple and robust, and yet achieves high market efficiency under a wide range of market conditions, with low communications and computation costs. More importantly, it offers continuous matching, which makes it flexible and fulfils the requirement for immediate allocation." (Tan and Gurd, 2007, p.283).

Tan and Gurd created a novel auction, a modified form of CDA, giving as their reason that "the traditional form of the CDA exhibits unnecessarily high price volatility" (2007, p.283). In contrast, they maintain that their modified CDA does not show this volatility but instead allows allocation of resources at stable prices in real time.

Grosu and Das (2004) investigated a first price auction, a Vickrey auction, and a double auction for grid resource allocation; they examined the auctions in terms of suitability for grid resource allocation, the economic efficiency of the auction, and system performance. They investigated these auctions in relation to the risk aversion of agents and discovered that the characteristics of the auctions resulted in one auction being preferred from a resource perspective, another being preferred from a resource consumer's perspective, and yet another being preferred when both were considered.

There are numerous types of auction available for grid resource allocation, and both continuous and batch auctions have been shown to offer considerable benefits over other resource allocation mechanisms. Various authors have used different metrics for examining the desirability of different auctions.

The use of auctions in grid resource allocation mechanisms has been studied extensively in the literature. There is a considerable variety of auctions used in the literature, with many

of the studied articles indicating that auctions are a convenient and effective method of allocating resources. Researchers such as Grosu and Das (2004) have found circumstances in which one auction has produced a more desirable allocation than another.

2.3.3 The use of agents

This section of the literature review examines the extensive literature on the involvement of agents in the grid resource allocation process.

Agents have been used in many resource allocation mechanisms (Cao et al., 2003; Chun and Culler, 2000; Fu et al., 2003; Negri et al., 2005; Tan and Gurd, 2007; Xiao et al., 2005). These agents are typically employed inside market-based resource allocation mechanisms, or as part of the resource allocation mechanism itself.

Agents are involved in a number of different resource allocation mechanisms that are used in a variety of ways. SHARP (Fu et al., 2003) is one such mechanism designed to allocate resources in wide area networks. One of the key features of SHARP is its distribution of resource allocation: within the SHARP framework resources are controlled by agents that produce claim tickets to the resources that they have control over, and distribute these tickets to clients who request them (Fu et al., 2003). Interestingly, SHARP agents may choose to oversubscribe their resources. It would seem to be counter-intuitive to allow this to happen in a system, but the authors observe that "Oversubscribed claims can improve resource utilization by statistical multiplexing" (Fu et al., 2003, p.136)

Porter (2004) uses agents in a completely different manner by incorporating flexibility. Porter created a formula for a grid resource allocation mechanism that allows self-interested agents to modify the algorithm to best allow their jobs to be completed over the grid (Porter, 2004).

Cao et al. have developed another agent-based system, in which agents are used to control resource discovery and to make resource allocation decisions based on their own internal

logic and not on any overall resource allocation policy. Each agent in this system is aware only of its neighbours, and requests are processed only among local agents (Cao et al., 2003). This gives the system great potential for scalability as it overcomes many of the load limitations on systems with a centralised auctioning environment. This system incorporates agents into every facet of resource allocation:

> "The scheduler introduced in this work is distributed for grid computing using an agent-based methodology, where agents are used to control the query process and to make resource discovery decisions based on internal logic rather than relying on a fixed-function query engine." (Cao et al., 2003, p.2).

A more conservative approach to the use of agents was elaborated on by Tan and Gurd (2007), whose system employs a CDA as an allocation mechanism for grid resources. Within this auction there are a number of agents, each of which uses one of three fixed strategies in order to maximise its benefit.

Negri et al. (2005) propose a new agent-based grid system called GAIN. The developers of the GAIN system have enthusiastically incorporated agents into their resource allocation system in a similar fashion to Cao et al. (2003). The result is a complex system consisting of six different types of agent, which are involved in all aspects of development and execution of the grid applications run over the GAIN system. Such extensive use of agents is justified by the developers because of the proven benefits of incorporating agents into the resource allocation system. "The use of agents have [sic] been proved a good means for intelligent task distribution and for supporting users in both on-line and off-line activities" (Negri et al., 2005, p.1). Unfortunately, this extensive use of agents was not supported by results, as despite the enthusiasm of the developers no results were presented.

Agents have been extensively incorporated into grid resource allocation systems, both as part of the resource allocation system and as part of the resource allocation decision-making process. There has been considerable justification for the extensive incorporation of agents

as part of a grid resource allocation mechanism. However, agents can be computationally intensive (Goldstein, 2007; Lynar et al., 2007), so extensive implementations of agents in a resource allocation mechanism must be justified by a benefit.

2.3.4 Predictive scheduling in resource allocation mechanisms

In order for resource allocation mechanisms to successfully schedule jobs, the execution time of the application is often required in advance. A number of grid resource allocation mechanisms require the user to guess or know the length of time their application will take to execute over unknown hardware (Germain-Renaud et al., 2008). Some systems have attempted to solve this problem using various methods to make an informed guess as to how long a task will take. This section of the literature review examines why this is an issue and reviews the various methods that have been used in the attempt to overcome the problem. Accurate knowledge of the execution time for a task to be run on a distributed resource can be important. If a user is bidding for resources it is important that the user knows how much resource time they must bid for. In some instances if they do not bid for enough compute time their task may be terminated before it finishes. If the user is not required to input the estimated time for the completion of a task, it is important that the resource allocation mechanism knows how long the task will take so that the resource allocation mechanism can successfully schedule tasks. Accurately predicting the execution time of a task is becoming increasingly important due to the growing demand for Quality of Service (QoS) agreements (Spooner et al., 2003). Job execution time impacts scheduling, which can affect QoS performance metrics. Many authors have examined the issue of job execution time (Anderson, 2004; Anderson, 2007; Chapman et al., 2007; Huang et al., 2005; Scriven et al., 2008; Spooner et al., 2003).

Many authors believe that there will be a growing emphasis on QoS agreements in the future (Spooner et al., 2003; Foster et al., 2002; Foster et al., 2004; Germain-Renaud et al., 2008). Spooner et al. state that "quality-of-service (QoS) contracts will become an integral element of grid scheduling, where the perceived value of a service will steadily decrease as

key service metrics such as deadlines are not met" (2003, p.87). There is a real and present need for accurate estimates of application runtime in grid resource allocation environments, which is emphasised by the need to meet QoS agreements.

Many grid resource allocation systems have attempted to incorporate a method of estimating the time a task takes to execute, and subsequently to schedule tasks based on this information. There may be potential to conserve energy through different scheduling techniques. However, the research presented in this thesis makes no attempt at such scheduling, instead using resource allocation strategies that have no requirement for estimates of runtime.

Many innovative resource allocation mechanisms have been created for the distributed computing community, often with a particular use in mind. Despite this, few have been designed in a manner that attempts to both minimise energy consumption and maximise the throughput of tasks.

This section of the literature review has examined the current state of grid resource allocation mechanisms. It has examined the use of auctions as a mechanism for grid resource allocation, reviewed the use of agents in resource allocation mechanisms, and investigated the issue of predicting the job execution time of an application over unknown hardware. The following section will examine e-waste and why it is a problem.

2.4 Electronic waste

Electronic waste (e-waste) is a growing concern and is a major focus of this research. This section looks at what e-waste is, what problems are currently associated with e-waste, current methods of dealing with e-waste, and how computational grids could be part of an e-waste solution.

E-waste is a recent term referring to electronic goods that have been discarded or have entered a period of disuse. The Australian Bureau of Statistics (ABS) describes e-waste

as "obsolete electronic waste" (ABS, 2006, p.4). E-waste can include many items such as televisions, electronic audio equipment, white goods, mobile phones, and computer systems (ABS, 2006; Terazono et al., 2006). However, the definition of e-waste and the means of calculating the quantity of e-waste differ between organisations and between countries (Terazono et al., 2006).

Terazono et al. state that "there is no standard definition of e-waste, and the methods used to estimate e-waste generation are not compatible among countries" (2006, p.3).

The component of e-waste that concerns this thesis is computers, particularly personal computers constructed out of ordinary commodity parts.

E-waste is increasingly becoming an issue for both developing and developed nations for a number of reasons, including the large and growing quantity of e-waste, the methods of disposal and deconstruction of e-waste, the quantity of non-renewable resources in e-waste, the potential environmental and health effects of incorrect disposal of e-waste, and the potential for damage to reputations of companies and countries. For all of these reasons, e-waste has recently become an issue of concern in Australia and in the rest of the world.

There is already a large and growing quantity of e-waste in Australia and throughout the world. An ABS report (ABS, 2006) discusses e-waste in Australia, noting that e-waste is a rapidly growing source of waste in Australia, of which only a small percentage is being recycled.

> "E-waste is one of the fastest growing waste types. Very little of the increasing amount of e-waste generated in Australia is being recycled, with most of it ending up in landfill" (ABS, 2006, p.4).

Australians purchase 2.4 million personal computers each year (ABS, 2006). Presumably all of which will eventually become obsolete, and this is a growing problem: "It is estimated that there are currently around nine million computers, five million printers and two million scanners in households and businesses across Australia" (ABS, 2006, p.19). Personal

computers particularly have a short life span: "The duration of the product's first life is estimated to be between 2 and 4 years for corporate users and between 2 and 5 years for domestic users"(Ahluwalia and Nema, 2007, p.793).

Although a considerable number of new electronic products are purchased every year, more concerning is the growth rate of e-waste in comparison to other forms of waste in Australia: "E-waste in Australia is growing at over three times the rate of general municipal waste" (ABS, 2006, p.50).

Australia is not the only country to experience growth in e-waste. Developing countries such as China and India are also experiencing a boom in the quantity of electronic items that are becoming obsolete.

While Australia's problem with e-waste might appear large, it is dwarfed by other countries. In 2003, China domestically consumed 30.7 million new personal computers, and it was estimated that by 2010 China would have over 70 million obsolete personal computers (Li et al., 2006). India also has a substantial problem, with an estimated 2.25 million computers becoming obsolete in 2005, growing to an estimated 8 million obsolete computers by 2010 (Ahluwalia and Nema, 2007).

The volume and rate of growth of e-waste in Australia, China, and the rest of the world is concerning, but so too are the methods of disposal which are employed, as discussed in the next section.

2.4.1 Disposal methods

Four basic methods are employed for dealing with e-waste: reuse the item, recycle the item, destroy/dump the item, or store the item.

Landfill and storage are the two main methods employed in Australia, where a significant number of computers are disposed of every year. The ABS (2006) observes that:

> "It has been estimated that in 2006 there will be around 1.6 million computers disposed of in landfill, 1.8 million put in storage (in addition to the 5.3 million already gathering dust in garages and other storage areas and 0.5 million recycled in Australia alone)" (ABS, 2006, p.20).

Some researchers have suggested that the alternatives to landfill may not be environmentally preferable (Williams, Kahhat, Allenby, Kavazanjian, Xu and Kim, 2008). Although the destruction of computers in landfill is unsophisticated, studies have shown that it poses little threat to the environment: "our analysis suggests that the risk of leaching of toxic materials in computers from well-managed sanitary landfills is very small" (Williams, Kahhat, Allenby, Kavazanjian, Kim and Xu, 2008, p.6446).

China has a very different set of problems with e-waste disposal. Although China has large quantities of obsolete computers, Li et al. indicate that due to tradition and cultural norms the Chinese people are unlikely to dispose of e-waste in landfill: "According to the traditional economical custom, Chinese seldom discharge their used electrical and electronic products, even if these products are out of date or broken" (Li et al., 2006).

Recycling of e-waste can take many forms. At the low technology end of the spectrum of recycling, people are employed to remove components manually and sort parts into piles of like materials based on their physical appearance (Krikke, 2008).. At the high technology end of the spectrum, computers are first shredded into uniform sized pieces and then sorted in machines (Krikke, 2008). A variety of machines can be used to sort the shredded material. There are machines that use magnets, x-rays, and alternating electrical currents to separate metals, and there are machines that can sort plastics based on their density (Krikke, 2008). Well managed recycling is not necessarily the most environmentally friendly disposal method for computers; Lu et al. note that "recycling for some components actually leads to greater negative environmental impacts than the alternatives" (2006, p.13). However, much e-waste is exported each year to developing nations, and ends up in the unofficial recycling industry where low technology methods of recycling and deconstruction are generally used (Li et al., 2006). Williams, Kahhat, Allenby, Kavazanjian, Kim and Xu

add that practices used in the unofficial recycling industry can actually generate additional toxins: "Backyard recycling processes both release these toxins as well as generate new ones" (2008, p. 6446).

Export of e-waste

Some organisations and individuals deal with the issue of e-waste simply by exporting it. The export of e-waste has itself become an issue, and in many nations its export and import are illegal. While many individuals believe that their e-waste is being recycled, it is possible it is actually being exported to another country for processing. Cairns states that

> "A substantial quantity of the equipment returned for recycling, more than half by some estimates, may actually be exported for disposal in other countries where environmental and occupational health protections are weak and landfills are not properly controlled." (2005, p. 237)

Many developing nations have imported or are importing e-waste. It is illegal to import e-waste into China (Li et al., 2006), but it is believed that e-waste is still exported to China, and to other developing nations, under different names. E-waste is often transformed into metal scrap, which is then sold overseas for further processing. Terazono et al. (2006) explained that in Korea "crushed E-waste is categorized as scrap that is needed to be recycled overseas. This e-waste is exported under the name of mixed metal scrap" (2006, p.7). This method of exporting e-waste appears common in many nations. Krikke (2008) states that "The US exports more than $1 billion of scrap to China annually" (2008, p.53). The total value of US exports of scrap, waste, and used goods exceeded $15 billion dollars in 2005 (Williams, Kahhat, Allenby, Kavazanjian, Kim and Xu, 2008) – although it should be noted that it cannot be determined what quantity of this scrap metal is a result of e-waste. The export of e-waste is not limited to Asia; there is a growing trend to export e-waste to Africa, where the disposal of e-waste is typically different from that in Asia (Schmidt, 2006).

In Africa there is generally little or no attempt to recover materials from the e-waste, so it is often piled up and burnt (Schmidt, 2006), or resold or used for refurbishing. Many of the waste items may have come in as legitimate donations; others may have been purchased, with the cost being paid for by the transportation of useless components which are expensive to dispose of in the West, such as Cathode Ray Tube (CRT) displays (Schmidt, 2006).

The literature indicates a number of reasons why e-waste is being exported for processing. E-waste is typically exported by Western nations to developing nations with less stringent environmental and occupational laws. This is presumably because it is seen as a profitable enterprise by those on both sides of the deal (Williams, Kahhat, Allenby, Kavazanjian, Kim and Xu, 2008).

Li et al. (2006) state that there are three economic reasons to export e-waste from developed countries to their developing neighbours: the lower cost of recycling waste, lower labour costs, and larger demand for the secondary materials that can be recovered from the waste. E-waste is rich in valuable metals including gold and copper. Velte et al. state that "one ton of computer scrap yields more gold than 17 tons of gold ore" (2008, p.128).

Li et al. indicate that because of its less stringent laws and cheap labour, China is a prime target for the import of e-waste, although this import is illegal: "lax environmental and occupational laws and regulations make China one of the destinations to which some developed countries export their e-waste" (2006, p.13). It is also considered that recycling e-waste is difficult: "the treatment and disposal of e-waste is problematic"(Li et al., 2006, p.13) although it can be profitable. There is a growing demand for natural resources in China and some of the metals that can be recovered from the e-waste are valuable; electronic goods typically have small quantities of precious and nonferrous metals (Tong, 2004).

The Basel Convention is an international agreement that prohibits the international movement of certain hazardous wastes; e-waste is prohibited under Annex VIII List A of the Convention (*Basel Convention on the Control of Transboundary Movements of Hazardous Wastes and Their Disposal*, 2008). The preamble of the Basel Convention document makes

it clear that the prohibition is due to the potential negative effects of certain waste products on the environment and on human health.

Some groups disagree with the limitation of trade in e-waste. Krikke (2008) states that among groups that oppose the Basel Convention, the Liberty Institute in Delhi is in opposition because it "opposes any kind of trade restrictions by state agencies" (Krikke, 2008, p.55). This is just one of the many arguments that are used by supporters of the waste trade. Other arguments in favour of the e-waste trade include that e-waste processing provides much needed employment in developing countries (Krikke, 2008) and that what is considered to be waste in one country could be considered to be a usable good in another country (Terazono et al., 2006).

The export of e-waste continues because its movement from developed nations to developing nations is typically profitable. However, it is a concern because the environmental and occupational laws of the developing nations to which e-waste is transported are typically less stringent than the laws of the nations from which the waste has come. The risk is that the electronic goods have the potential to cause more harm than benefit to the people and environment of the nation that imports it.

Toxins, health and environmental effects

The production of electronic goods involves numerous metals and chemicals, some of which can bring about serious consequences for health and the environment. E-waste, particularly of computers, is both valuable and hazardous. Personal computers contain a large variety of different materials including toxic and valuable metals: "A typical PC consists of 23 percent plastic, 32 percent ferrous metals, 18 percent nonferrous metals, and 12 percent electronic boards (gold, palladium, silver, and platinum)" (Krikke, 2008, p.53). However, e-waste also contains many toxic materials, including heavy metals such as lead, mercury, arsenic, and cadmium, and chemicals such as brominated flame retardants (Terazono et al., 2006). Modern Liquid Crystal Displays (LCDs) also contain a number of hazardous chemicals.

"The LCD contains the chemicals ... cyclohexane, pyrimidine and biphenyl, as well as heavy metals, and thus is considered a prime source of contaminants after disposal" (Lu et al., 2006, p.18) The valuable materials in e-waste can be difficult to extract; the methods of disposal or recycling can dictate the environmental and health effects that the waste will have.

Waste management appears to be one of the primary concerns relating to e-waste, but there are many others, as outlined in the following sections.

Security

Privacy and security issues can dictate the method by which a company or individuals will dispose of their waste. Cairns states that "security of personal information embedded in equipment that is recycled or reused may also deter consumers from relinquishing old units" (2005, p.240). The security of data or the perception of security of data can be very important in determining what a user will do with their disused electronic equipment. Individuals and organisations may not wish to discard or recycle their equipment due to fear that data could be stolen from their computer's hard disks. The issue of security and privacy may partially explain why so many disused computer systems are being stored rather than disposed of.

Reputation and legality

Another driving factor in the e-waste issue is the threat to the reputation of organisations and individuals. Carlson (2003) illuminates the issue of e-waste in relation to the reputation of an organisation responsible for the equipment and also in relation to the possibility of criminal charges being laid against individuals as a result of unacceptable disposal practices. Interviewing an information technology manager at an American university, Carlson concludes that "He fears that he could be fined if one of his computers later turned up in a dump, or abandoned by the side of the road, or put on a ship headed for Asia" (2003,

p.A34). Many companies and individuals are concerned about their reputation, and about the potential of legal consequences if they do not dispose of their waste in a legal manner.

This section of the literature review has summarised the issue of e-waste, by discussing what e-waste is, why e-waste is a problem, and current programs that are in place to resolve the e-waste problem.

2.5 Computing projects that utilise e-waste

This section reviews past and present research projects that have utilised e-waste to form HPC environments.

In 1994 NASA created the first cluster of personal computers and named it Beowulf (Hargrove et al., 2001). Thomas Sterling and Donald Becker hypothesised that supercomputer-like processing power could be obtained by networking inexpensive commodity computing systems, which they called Beowulf clusters (Gropp et al., 2003). Since that time many such clusters have been produced using personal computers.

A cluster computing project named the Stone SouperComputer was constructed entirely of disused computers (Hargrove et al., 2001). It has been reported that due to a planning mistake there were no funds allocated to computing for the project, so it was decided to create a Beowulf cluster out of disused workstations (Gropp et al., 2003). The name was taken from the stone soup fable, where a soldier stops in a village and informs everyone that he will make soup out of a rock. The soldier eventually encourages everyone to add just a little to his soup, until eventually he has a proper soup (Hargrove et al., 2001). The Stone SouperComputer cluster was constructed at the Oak Ridge National Laboratory exclusively from disused computers.

The creators of the Stone SouperComputer cluster were able to obtain parts for no cost. This particular cluster eventually contained 133 nodes of varying power and hardware (Hargrove et al., 2001). While many projects incorporated Beowulf clusters, the particular point of

interest with this project is that it created a usable cluster resource solely from old disused computers. The resource was used successfully for a number of tasks (Hargrove et al., 2001).

The Stone SouperComputer is a good example of what can be achieved with reused computers.

2.6 Green high-performance computing and resource allocation

This section of the thesis examines the literature on energy-aware, power-aware, HPC. The literature reviewed here examines the methods by which energy and power savings are made, and examines the differences in methods used for heterogeneous and homogeneous computing environments. The literature on energy-aware computational grids is sparse. This review generally covers literature on comparable technologies, whose methods of energy conservation could be modified to be implemented on grids of heterogeneous machines. The idea of treating energy as a primary concern in operating environments was proposed a decade ago (Vahdat et al., 2000), but the concept of energy awareness is relatively new to the HPC domain (Hsu and Feng, 2005*b*). This literature review examines why energy savings are important, what potential exists to save energy, and the current methods employed for saving energy in comparable environments.

2.6.1 The importance of saving energy

The energy used in HPC is significant (Hsu and Feng, 2005*a*), and much of this energy could be saved. No statistics have been found for the energy consumption of computational grid installations, but it is believed that their energy needs are substantial. In 2006 the electricity used by data centres in the United States of America accounted for approximately 1.5% of total United States electricity consumption (Velte et al., 2008). High energy usage has numerous negative effects, one of the most obvious being an increase in heat production. "When cluster nodes consume and dissipate more power, they must be spaced out and aggressively cooled; otherwise, un-dissipated power causes the temperature

to increase rapidly" (Hsu and Feng, 2005*a*, p.1). Increases in server room temperature have other negative effects, such as an increase in erroneous results and hardware failure (Hsu and Feng, 2005*b*). "for every 10°C increase in temperature the failure rate of a system doubles" (Hsu and Feng, 2005*b*, p.1). Other researchers note that a reduction in server room power consumption has many beneficial effects on the operating environment, including increased reliability: "First, decreasing power consumption leads to greater reliability and maintainability. It also has significant secondary benefits because of the corresponding reduction in heat generated" (Springer et al., 2006, p.230). The corresponding reduction in server room temperature can even result in an extension of the life of the hardware (Springer et al., 2006).

The other negative effect of high power consumption is an increase in the cost of cooling. Hsu and Feng (2005*a*) note that the cost of cooling server rooms in some research institutions is as high as 70% of the energy used to power the equipment that is being cooled: "For every watt of power consumed at LLNL, 0.7 watts of cooling is needed to dissipate the power" (2005*a*, p.1).

There is, however, a limit to the quantity of energy that can be consumed, and many of these centres are running out of capacity. It has been predicted that up to 50 percent of data centres in the United States of America have insufficient capacity (Velte et al., 2008).

High energy and power consumption will result in high energy bills, but may also result in other negative effects, including increased heat and reduced hardware reliability. The need to conserve energy for computing is clear, and there have been a variety of different approaches explored to achieve this.

2.6.2 Potential to save

While the energy used in HPC is substantial, so too is the potential to save energy. Springer et al. note that HPC installations rarely reach peak performance: "because HPC applications rarely achieve peak efficiency, much of the power that is used is actually wasted"

(2006, p.230). The utilisation of HPC installations is typically low, and the idle power consumption of computing nodes is typically high as a percentage of their peak power consumption. For example Chase and Doyle found that "all servers we measured draw 60% or more of their peak power even when idle" (2001, p.1). The CPU is not the only component of a node that needs to be addressed, but it is major and can account for as much as 55% of overall system energy usage (Freeh et al., 2005). There is much potential to save energy. For example, significant savings could be made at the level of individual nodes merely by turning off resources when they are not in use, or by reducing the power consumption of idle components. At the cluster level there is great potential to save energy through coordinated efforts, load balancing, and leveraging the heterogeneity of the nodes. Many methods of saving energy have been discovered in the literature and will now be discussed.

2.6.3 Method of saving

The energy-saving techniques described in the literature include processing voltage and frequency scaling (VFS), placing nodes in various power states including completely turned off, and resource allocation. The methods employed can be broadly categorised as either uncoordinated, adjusting individual nodes on the basis of the internal information of those nodes, or coordinated, examining the whole system prior to making a change.

Uncoordinated methods

The uncoordinated methods described in the literature are based on the power saving mechanisms of processors and other hardware. These methods, which are employed individually and in an uncoordinated manner, are available with the majority of new computers and have been shown to substantially reduce the energy consumption of a cluster (Elnozahy et al., 2003; Hsu and Feng, 2005*a*; Hsu and Feng, 2005*b*). The most common method employed is VFS, which some projects have employed independently on nodes (Elnozahy et al., 2003; Freeh et al., 2005; Hsu and Feng, 2005*a*; Hsu and Feng, 2005*b*). VFS involves adjusting the voltage and frequency of a processor, which as a consequence alters the amount

of power the processor consumes. This can result in substantial overall system savings, as the CPU can account for as much as 55% of system energy consumption (Freeh et al., 2005). VFS has shown savings of between 19% and 29% of system energy consumption (Elnozahy et al., 2003; Hsu and Feng, 2005*a*).

Freeh et al. (2005) have explored manual VFS, but note that it can also be performed dynamically (Freeh et al., 2005, p.1). Many researchers have explored dynamic voltage and frequency scaling (DVFS) (Elnozahy et al., 2003; Hsu and Feng, 2005*a*; Hsu et al., 2005; Hsu and Feng, 2005*b*; Khan, 2009). This method uses system software, such as Intel's SpeedStep (Khargharia et al., 2008), to dynamically alter the voltage and frequency of the CPU during runtime (Hsu and Feng, 2005*a*). The results of DVFS have been promising, but both DVFS and VFS result in a tradeoff between performance and energy consumption (Hsu and Feng, 2005*a*). Results show that DVFS can achieve a substantial saving in energy consumption at the same time as maintaining a high level of performance: "a 5% performance slowdown can be traded off for an average of 19% system energy savings and 24% system power reduction" (Hsu and Feng, 2005*a*, p.1).

The drawback of these methods is that the energy savings are achieved at the cost of speed, as the use of processor scaling methods involves a tradeoff between speed of execution and energy consumption. Harada et al. state that "Lowering the clock speed in order to reduce energy consumption, however, leads to QoS degradation. Thus, there is a trade-off between energy consumption and system performance" (2006, p.1). In the majority of the literature DVFS has been implemented as an uncoordinated method; however, Khan (2009) utilised game theory to implement DVFS in a coordinated fashion in a grid environment.

Other promising uncoordinated methods for energy saving do exist. One simple and promising method is dynamic power management (DPM) of non-processor devices such as hard disks, network interface cards, and random access memory (Khargharia et al., 2008).

Although uncoordinated methods are of interest to this research, coordinated methods of energy saving are more relevant, as they operate on the cluster level, with potential to be scaled for use on a grid.

Coordinated methods

The coordinated methods are methods that can be employed cluster-wide, using data collected from the nodes and the resource allocator to make decisions on resource allocation, distribution, and the employment of power saving measures. It is these decisions that result in a reduction of the cluster's overall energy consumption. This section will explore various coordinated methods including load unbalancing, coordinated voltage and frequency scaling, Variable On / Variable Off (VOVO), and resource allocation.

Load unbalancing is the strategic use of load balancing to concentrate processing jobs on fewer servers while completely unloading other servers. The conservation of energy is achieved by the unloaded computers idling, being switched to a low power state, or being turned off (Pinheiro et al., 2001; Verma et al., 2008). Load unbalancing is a form of resource allocation that can achieve reasonable reductions in energy usage, even with homogeneous nodes; for example, Pinheiro et al. (2001) were able to achieve a 43% reduction of energy consumption (2001). Verma et al. (2008) have also attempted load unbalancing, but used virtual machines to achieve this, allowing for the added benefit of live migration of running applications. However, energy consumption statistics were not published for this innovation.

VFS and DVFS have been discussed as an uncoordinated method that can be of some benefit. Coordinated voltage and frequency scaling is a more sophisticated form of this method, which uses data obtained from other nodes to coordinate actions with those nodes (Elnozahy et al., 2003; Ge et al., 2007; Springer et al., 2006). Elnozahy et al. discovered that coordinated voltage scaling resulted in slightly greater energy savings than independent voltage scaling. However, in the opinion of the authors this saving was not sufficient to warrant the additional complexity of the implementation of coordinated VFS over independent VFS: "this benefit is probably not sufficient to justify the increased implementation complexity" (Elnozahy et al., 2003, p.180). More recently a simulation study by Beloglazov and Buyya (2010) has examined the use of resource allocation to conserve energy with the placement of virtual machines in data centres. They achieved savings of up to 83% when utilising DVFS with resource allocation of virtual machines (Beloglazov and Buyya, 2010).

VOVO is a coordinated method of saving energy by dynamically switching nodes on and off, or into different power states (Elnozahy et al., 2003). Chase and Doyle propose that "servers are an appropriate granularity for power management in clusters" (2001, p.1). Many authors have since proposed the use of VOVO in one form or another (Elnozahy et al., 2003; Rusu et al., 2006; Khargharia et al., 2008; Aziz and El-Rewini, 2009). Node VOVO has been described as follows:

> "Node vary-on/vary-off takes whole nodes offline when the workload can be adequately served by a subset of the nodes in the cluster. Machines that are taken off-line may be powered off or placed in a low power state. Machines that are taken off-line are placed back online should the workload increase. Node vary-on/vary-off is an inter-node power management mechanism" (Elnozahy et al., 2003, p.181).

VOVO has shown substantial energy savings when used alone, and greater savings when used in combination with other methods: "On average, we measured energy savings of 17% using DVS, 39% using On/Off [VOVO], and 45% using both schemes" (Rusu et al., 2006, p.8). VOVO's potential to turn computers off to save energy could be particularly useful in grid environments that are under-utilised, as some computers may rarely process a task. While these computers would normally sit idle for extended periods of time, VOVO would offer the possibility of switching them off, saving substantial amounts of energy.

Most of the coordinated efforts to reduce energy consumption rely in some way on resource allocation. Some efforts use resource allocation alone as a method of energy conservation. Heath et al. (2005) used simulated annealing to come up with a schedule for the resource allocation of web server jobs to heterogeneous nodes. They claim that this can result in a 42% energy saving compared with a system that does not account for the heterogeneous nature of the nodes. Unfortunately, they give no indication of how this substantial saving was achieved.

The coordinated and uncoordinated methods examined offer the potential for substantial energy and power reductions for a variety of distributed computing environments.

2.6.4 Type of hardware

The characteristics of the hardware used in cluster and grid environments are also important. Most systems assume that the hardware in the system is entirely homogeneous, but this is rarely the case. The importance of accounting for the heterogeneous nature of hardware should not be overlooked. Some environments are constructed using heterogeneous hardware, and some are constructed using homogeneous hardware. However, homogeneous environments may and inevitably do become heterogeneous over time: "Real-life clusters are almost invariably heterogeneous in terms of the performance, capacity, and power consumption of their hardware components." (Heath et al., 2005, p.1)

Heterogeneity of resources creates a different set of problems but also a different set of opportunities for energy saving through resource allocation:

> "Heterogeneity raises the problem of how to distribute the clients' requests to the different cluster nodes for best performance. Furthermore, heterogeneity must be considered in cluster reconfiguration for energy conservation, raising the additional problem of how to configure the cluster" (Heath et al., 2005, p.1).

There appears to have been little research on utilising the heterogeneous nature of server rooms. A number of studies examine heterogeneity in relation to load-balancing web-servers (Heath et al., 2005; Rusu et al., 2006). Lubin et al. (2009) present a novel market-based mechanism for heterogeneous data centres that attempts to maximise performance to the user while at the same time minimising energy and other costs to the seller.

2.6.5 Types of system

The research reviewed thus far has dealt with web-server and high-performance cluster environments. There are important differences between these systems on the one hand and geographically distributed grid systems on the other. The literature discovered on energy-aware grid computing has been very limited. Patel et al. (2002) discuss utilising the heterogeneous nature of geographically dispersed data centres to reduce energy consumption, taking advantage not only of the different designs of the data centres but also of the differences in weather at the different locations. Shah and Krishnan (2008) further develop the same concept and explore both the environmental and economic consequences of this approach. Costa et al. (2009) propose a framework that utilises resource allocation and VOVO at the grid level to reduce the energy consumption of a grid, while Subrata et al. (2010) describe a power-aware game theoretical scheduling solution that accounts for both ownership and incentives of grid resource providers.

2.6.6 The need for energy savings in HPC environments

There is a great need to reduce the overall energy consumption of computing, particularly in server rooms, where a great deal of energy is used to cool equipment that uses substantial energy and produces substantial heat. Most of the literature reviewed elucidates energy saving techniques for web servers and assumes homogeneous resources. This review shows a great gap in the literature: there has been little research done for HPC environments, grid environments, and environments that are heterogeneous in nature.

This section of the literature review has discussed methods to achieve reductions in energy usage in distributed computing environments. One of these methods was resource allocation – choosing which computing node will execute each incoming task. When a distributed resource contains nodes of differing performance abilities, there are many ways to differentiate and rank them. The next section of the literature review discusses benchmarks as a method of ranking heterogeneous nodes.

2.7 Benchmarks

The literature on economic resource allocation was examined earlier in this literature review. Economic resource allocation mechanisms rely on some form of valuation of a resource. Energy-aware resource allocation was also reviewed. To value a resource based on its computational ability and energy consumption, measures of both are required. Scriven et al. state that "clock speeds are no longer a valid way of comparing the performance of different CPUs"(2008, p.68). To enable the comparison of different computers, benchmarks were developed. There are many benchmarks available, but they are not always indicative of the true performance of the system. This section of the literature review examines the use of benchmarks, and some common benchmark applications available.

2.7.1 Benchmarks explained

Speed is an important characteristic of computers, and benchmarks are used for comparing the speeds of different computers (Curnow and Wichmann, 1976). There are two broad categories of benchmark: synthetic and non-synthetic (Weicker, 1990). The speed of an application on a particular computer is most accurately discovered by executing that application on the computer in question; this is a non-synthetic benchmark. Weicker states "It has been said that the best benchmark is the user's own application. But this is often unrealistic, since it is not always possible to run the application on each computer in question" (1990, p.66). Synthetic benchmarks become useful when it is not possible to run the application on every computer that needs testing. Synthetic benchmarks can produce useful comparisons, but they do have potential disadvantages that need to be considered.

2.7.2 Disadvantages of synthetic benchmarks

There are a number of potential issues related to synthetic benchmarks: benchmarks may be optimised for a particular architecture; the benchmark may not be widely used; and,

the benchmark may not be representative of a particular application type. A number of attempts to create synthetic benchmarks, taking the above issues into account, will now be reviewed.

2.7.3 Common benchmark applications

The Whetstone benchmark, created by H.J. Curnow and B.A. Wichmann (1976), is the first synthetic benchmark that appears in the literature (Weicker, 1990). The benchmark was first written in ALGOL 60 but later translated into FORTRAN (Curnow and Wichmann, 1976). Weicker (1990) identifies a number of characteristics of the Whetstone benchmark, including the high use of floating-point operations, a high proportion of execution time spent executing library functions, sparse use of local variables, substantial use of global variables, and attempts to prevent code optimisation. These characteristics need to be taken into account when utilising the Whetstone benchmark application.

Dhrystone is another popular synthetic benchmark, with characteristics quite different to the Whetstone benchmark (Weicker, 1990). Dhrystone uses few floating-point operations, calls many string functions, incorporates few loops but many binary selection statements, and makes no attempt to prevent code optimisation by the compiler (Weicker, 1990).

2.7.4 Benchmark output

Synthetic benchmarks are used to assist in the ranking and comparative analysis of the performance of systems. Many benchmarks output a comparative measure of performance as a result of their execution, and one such measure is Millions of Instructions Per Second (MIPS). While MIPS is one measure of a computer's processing ability, there are many reasons why it should not be considered a good indicator of a computer's ability to compute a task.

Factors other than CPU clock speed have importance in relation to the speed of the execution of an application. The size of memory and processor cache can be important, as too

can the language, the compiler, and the related library functions called by the application. In addition, "cache size can have a considerable effect" (Weicker, 1990, p.73). There are other reasons to question the relevance of MIPS, including the way in which the measure is derived, and differences in processor architecture.

A MIPS value provided by a manufacturer can have a number of different meanings. The number may be made based on some calculation. It could be 'Native MIPS' – in which case it is important to know what application was used to derive the number; 'Peak MIPS', which "is largely irrelevant, since it equals the clock frequency for most processors" (Weicker, 1990, p.66); or even MIPS derived using a particular application. In the search for comparability, VAX MIPS define the speed of carrying out particular tasks relative to their speed of execution on a specific VAX computer from the 1970s.

It is apparent that MIPS itself is not an ideal indicator of performance, so it is generally better just to compare the execution speed of a particular application over the various computer systems.

Unfortunately, it is sometimes not possible to execute the same application over different machines, so it has been decided, for this research, to use common benchmarks to test and compare the performance of various nodes. The use of common benchmarks gives a good general indication of the performance capability of hardware resources.

2.8 Concluding remarks

The review of the literature has examined many different facets of environmental grid computing and related areas. It has examined the use of agent-based models, and different methods of economic resource allocation used in distributed computing environments such as clusters and grids. The problem of e-waste was discussed in terms of its impact on the environment; current attempts at resource allocation designed to minimise energy usage were assessed; and finally, different methods currently employed to measure a computing node's processing ability were examined.

The research presented in the remainder of this thesis explores energy conservation by way of resource allocation that uses the characteristics of individual nodes. Chapter 3 examines the use of different performance metrics for nodes in relation to energy and time (Lynar et al., 2009). Chapter 4 describes the construction and examination of a simulation model with a number of different resource allocation mechanisms (Lynar and Herbert, 2009). Chapter 5 examines the creation of distributed computing resources using computers that were to be discarded as e-waste (Lynar, Herbert, Simon and Chivers, 2010). Chapter 6 describes the use of a real institutional grid to examine a controlled environment on which a number of resource allocation mechanisms can be tested (Lynar, Simon, Herbert and Chivers, 2010). Chapter 7 deals with the calibration of the simulation model and Chapter 8 examines the calibrated model by examining the results of simulating the execution of real world grid workloads.

The main original contribution of this thesis is the creation of a novel resource allocation mechanism that utilises auctions to conserve energy and maximise the speed of task execution in distributed computing environments such as grids. Publications resulting from this research are listed on page xxi.

Chapter 3

PERFORMANCE EFFICIENCY METRIC: AN ANALYSIS OF THE USE OF DIFFERENT METRICS TO RANK NODES

3.1 Introduction

In order for a resource allocation mechanism to prefer some nodes over others, a performance metric is required to rank each node based on its performance relative to the performance of other nodes. In this chapter a number of metrics and their effects are examined in relation to energy consumption and time. Many of the performance metrics examined in this chapter account for both a node's computational ability and its power consumption. These metrics are used to identify the most desirable nodes by ranking them, so that the resource allocation mechanism can allocate tasks to the most desirable node at any given time.

This chapter has two aims: to introduce a number of potential performance metrics and to discover what effect, if any, the use of differing performance metrics for node evaluation has on the overall energy consumption and speed of a grid environment. It is hoped that through the informed selection of a performance metric to rank resources the total energy consumption of the grid may be minimised while having a minimal impact on mean task execution time.

The research presented in this chapter has been published in the proceedings of the 2009 International Conference on Grid Computing & Applications (GCA 2009) (Lynar et al., 2009).

The key contribution of the research presented in this chapter is a novel performance metric for the ranking of heterogeneous computers and a simulation study analysing the impact on energy consumption and execution time of using different performance metrics to rank nodes for task allocation in distributed heterogeneous environments.

3.1.1 Why a measure of performance efficiency is needed

Many measures of a node's performance are currently available, including both synthetic and non-synthetic benchmarks. While the prime focus of HPC has historically been performance, there is an increasing focus on energy conservation (Hsu et al., 2005). Computers have variability in their power needs and their computational performance. Any performance metric for such computers should therefore take account of a node's power consumption as well as its processing ability. The aim of using a metric as described in this chapter is to indicate, through ranking, which nodes have the most desirable combination of power consumption, which is ideally low, and computational performance, which is ideally high. There are few current mechanisms available that perform this function.

As discussed in Section 2.7, benchmarks are an important measure of performance and allow for the easy comparison of one node against another. However, computational performance alone does not give the resource allocation mechanism enough information to identify the resources that best satisfy the goals of this research. The resource allocation mechanism must be able to rank nodes according not just to their computational performance but also to their power consumption.

In this chapter the Whetstone benchmark[1] was used to give an indication of the computational performance of eight nodes that were taken as a cross-section of the nodes available in the grid. The results, obtained by executing the benchmark over each node, are presented

[1] The Whetstone benchmark was obtained from `http://homepage.virgin.net/roy.longbottom/oldones.htm` and `http://homepage.virgin.net/roy.longbottom/dualcore.htm` The executables used in these tests were the `whetcod.exe` and `whets32MP.exe` executables.

as Mwips (Millions of Whetstone Instructions Per Second) and Mflops (Millions of Floating Point Operations Per Second) in Table 3.1.

3.1.2 Other methods of measuring a node's performance

There are many current measures of a node's performance that could be used to rank the different nodes in the grid. These measures include power consumption, Mflops, or any synthetic or non-synthetic benchmark that can be executed over each node. Few mechanisms that account for both power consumption and the node's computational performance were discovered in the literature. Measures such as $\frac{flops}{watts}$ have been used by many projects as a measure of power to performance (Eadline, 2008; Adams and Brom, 2008; Hsu et al., 2005), but these measures are possibly less than optimal, as noted in Section 2.7, because flops on its own may not be a good measure of a node's performance. In order to satisfy this need a generic performance efficiency metric has been designed.

3.2 Description of performance efficiency metrics

The performance efficiency metric of each node i includes the node's computational performance C_i and its power requirement P_i.

The computational performance C_i of a node can be measured in several ways. Three ways considered here are

1. the Whetstone processing power measured in Mwips (W_i)

2. the processing power measured in Mflops (F_i)

3. the time taken for a node to complete a benchmark task (T_i).

To ensure that a better computational performance is reflected by a larger number, the third measure is the reciprocal of the time taken on the benchmark task. Therefore C_i, the computational power of node i, can be measured as W_i, the measured Mwips of the node, F_i, the measured Mflops of the node, or $1/T_i$, the inverse of the number of seconds the node takes to complete the benchmark task.

The performance efficiency E_i for node i is then given by:

$$E_i = \frac{C_i}{P_i} \tag{3.1}$$

Because E_i will be calculated using all three measures of performance, for the remainder of this chapter the performance metric will be described as Ewips when using W_i as a measure of computational performance, Eflops when using F_i as a measure of computational performance, and Ettt when using T_i as a measure of computational performance.

3.3 Analysis of metrics

A cross-section of the available nodes included in the grid were examined and their various performance measures calculated. Both the Mwips and the Mflops measures of computational performance were calculated using the Whetstone benchmark. Three performance metric values were then calculated, one for each of the performance measures. The results are shown in Table 3.1. Examination of the table shows that the nodes are in comparable pairs; for example, nodes 5 and 6, which are newer computers, outperform the other six nodes on each of the three performance measures, but also draw more power than the other six nodes.

The performance measures derived from the performance metrics Ewips, Eflops, and Ettt will obviously produce different values for the same nodes. However, this research is interested not so much in the absolute values as in the ranks, the relative order in which the values place the nodes. If each measure ranks the nodes identically, the measures are functionally equivalent, and a resource allocation mechanism would always prefer the same

Table 3.1: Power, three measures of performance, and three corresponding performance measures for eight real nodes

Node	Node ID	Watts	Mwips	Ewips	Mflops	Eflops	Test task time	Ettt$\times 10^4$
Node 1	001	60	760	12.7	237	4.0	52	3.2
Node 2	002	62	760	12.3	234	3.8	44	3.7
Node 3	CN1	63	867	13.8	287	4.6	27	5.9
Node 4	CN2	63	868	13.8	286	4.5	28	5.7
Node 5	HN1	83	5485	66.1	1544	18.6	18	6.7
Node 6	HN2	103	6106	59.3	1795	17.4	18	5.4
Node 7	HPN1	79	966	12.2	318	4.0	25	5.1
Node 8	HPN2	80	966	12.1	318	4.0	24	5.2

node regardless of which measure was being used. Table 3.2 shows how each node is ranked according to each column of Table 3.1: the power drawn by the node, the traditional performance measures, and the new measures obtained from the new performance metrics.

Node 6, the fastest node by all three performance measures, would be ranked last if selected purely on the basis of minimising power consumption. Node 5, which is comparable to node 6 on many measures but draws less power, consequently ranks first on all three performance metric values. Interestingly, while node 6 ranks second on the Ewips, on Ettt it ranks only fourth, displaced by two nodes that process far more slowly but draw far less power.

It should be clear from the derivation of the performance efficiency metrics and from the subsequent analysis that nodes with low power consumption and high computational performance (however this is measured) will rank highest according to these metrics, that nodes with high power consumption and low computational performance will rank lowest, and that nodes with intermediate characteristics will rank between these extremes.

The implication of this finding is that the use of a different ranking metric has the potential to influence a resource allocation mechanism to select different resources, which may in turn

Table 3.2: Ranks given to eight real nodes by power, three performance measures, and the corresponding performance metric values

Node	Watts	Mwips	Ewips	Mflops	Eflops	Test task time	Ettt
Node 1	1	8	5	7	7	8	8
Node 2	2	7	6	8	8	7	7
Node 3	3	6	4	5	3	5	2
Node 4	3	5	3	6	4	6	3
Node 5	7	2	1	2	1	2	1
Node 6	8	1	2	1	2	1	4
Node 7	5	4	7	3	5	4	6
Node 8	6	3	8	4	6	3	5

alter the total energy consumed by the grid and the mean time taken to execute tasks on the grid.

3.4 Methodology

A simulation was created to further analyse the implications of using different performance measures and performance efficiency metrics. The simulation was of a grid containing the eight heterogeneous nodes described in Table 3.1. The simulation was run six times, once for each of six different measures for ranking nodes. During each run all elements of the simulation remained the same, with the exception of the metric used to rank the resources. The results from the simulation are non-stochastic and deterministic; it is certain that the same outputs will be observed when the same inputs are used. Inferential statistical tests are neither required nor appropriate to determine the properties of this population, as each individual result is, in effect, the whole population (McClave and Dietrich, 1991).

The research hypotheses is that changing the metric which is used to rank nodes will have an effect on the total energy consumed and on the total time taken for the grid to compute a set of tasks. In this research these sets of tasks are referred to as workflows. The null

hypothesis is therefore that irrespective of the performance metric used to rank nodes, the energy consumed and time taken to process the tasks will be the same.

3.4.1 Operational definitions

Several operational definitions were used:

Metric: the measure used to rank each node against other nodes

Energy used: energy measured in watt hours

Time taken: time measured in theoretical time steps that progress in sequence with an even spacing between steps.

3.4.2 Assumptions and intentions

The simulation incorporates several simplifying assumptions, including that:

- the power and performance readings taken from the real nodes are accurate;
- tasks will never fail during execution;
- there will be no failures in the hardware or the software;
- any task assigned to a node of the grid will begin execution immediately and will execute using all of the node's computational ability until the task is complete;
- all nodes can perform at their measured performance for every task irrespective of the way the task is programmed;
- all tasks require equal amounts of processing, and the processing that a task requires can be provided by any resource.

The intention of the research in this chapter is to rank nodes using different metrics – power itself, performance measures, and performance efficiency metrics – and to examine what effect this has on the total consumption of energy and the total time taken to complete the tasks. The intention is not to test the underlying resource allocation mechanism, the means by which these metrics are applied.

3.4.3 Experimental design

The inputs to the simulation are the attributes of each node, the attributes of the tasks submitted to the grid, and the metric to be used. The attributes of each node, which were collected from real nodes, include a flops measure, power usage at 100% processing, power usage while idle, a wips measure, and the time taken to execute a test task. Details of the time taken to execute a test task on each node are also input into the simulation. All the attributes of each node remain the same for each run. The performance metric to be used is an input of the simulation, and the choice of performance metric to be used is the only difference in inputs between runs of the simulation. In each run the performance metric is used to rank nodes against one another. Six different performance-related metrics were evaluated: power, flops, wips, test task time, Eflops, and Ettt.

The simulation assigns nodes to tasks using a simple batch auction, a mechanism that is explained in depth in Chapter 4. This mechanism was chosen for this test as it is a well understood economic method of resource allocation. The nodes are given scores based on the chosen performance metric, so that the node with the highest performance metric receives the highest score and is allocated the first task. Each resource is able to execute only one task at a time.

The model used to perform this simulation is an earlier version of the model described in Chapter 4 Section 4.2. For this experiment the model was set to use a variety of different performance metrics. The source code of this model is presented in Appendix A.1. An extended description of the model used in this chapter is described in Appendix E. To

determine the simulated rate at which a node can compute tasks, the model uses the node's measure of test task time.

At each time step the simulation outputs the total energy consumed and the mean execution time of tasks, and these outputs are recorded and archived for analysis.

3.4.4 Research instruments

A number of research instruments were used in the experiment. These tools and the reasons for using them are briefly described here.

Java: Java version 1.5.0-16 was used to construct and run the simulation.

Energy meter: Power Tech plus Multifunction Energy Meter Model MS-6115 was used to gain a measure of power consumption from the real nodes that were simulated.

Test task application: The test task application was to compile the Linpack HPC benchmark software (Petitet et al., 2008). The time taken to execute this task was measured using the GNU/Linux `time` command.

Flops application: The Whetstone benchmark was used to produce this reading in millions of floating point operations per second (Mflops).

Wips application: The Whetstone benchmark was used to produce this reading in millions of Whetstone instructions per second (Mwips).

3.4.5 Description of data collection and analysis techniques

This section details the procedures used to collect the data on each physical node. Two readings for power consumption were collected for each node, one when the node was idle

Algorithm 1 Data collection procedure

1: Reset energy meter
2: Plug computer in
3: Boot computer
4: Wait 5 minutes
5: Record observed watt usage (idle usage)
6: Insert benchmark disk
7: Run benchmark tool
8: Record watt reading whilst running (processing usage)
9: Save output file to disk
10: Remove disk
11: Turn off computer
12: Unplug cable

and another when it was processing a computationally intensive task. The procedure used to collect readings is displayed in Algorithm 1.

Computational performance measurements were taken on all nodes that were simulated in this experiment. To determine the time taken to execute a test task the High Performance Linpack Benchmark 1.0a 2004 was compiled and executed in a GNU/Linux operating system[2]. The `time` command was used to record the time taken to compile the benchmark on each node. The 'real time' output from this command was recorded.

The analysis examined the total energy consumed through computation and the total time it took to process all tasks.

3.4.6 Limitations

There are limitations to this methodology. The primary constraint is that the methodology employs a simulation and therefore has all the limitations of a simulation. These limitations

[2] The pelican live CD version 1.3 can be obtained from `http://pareto.uab.es/mcreel/PelicanHPC/`

include: the modelling and analysis of simulations can be time consuming and expensive; it is difficult to communicate a simulation to someone else so that they can repeat or verify the experiment; and, simulation utilises simplifying assumptions (Banks, 1998). Further limitations of this particular simulation are that a task can only start at the start of a time step, and that a node processes for a whole number of time steps, meaning that if a task finishes processing partway through a time step, the node will still be considered in use for the entire time step. However, this can also happen in a real system, as nodes that complete tasks would have to wait until the next task was assigned to them, and during this time the nodes would be on and consuming power but not performing useful processing.

3.4.7 Simulation validation

Chapter 6 presents the research of an examination on a real cluster of the effects that varying the performance metric has on the time taken to process tasks and the energy used to process them.

3.4.8 Simulation execution

A simulation was conducted to analyse the performance metrics. The execution of fourteen workflows were simulated. Three variables are used to describe each run: the length of the run; the length of the interval between tasks; and, the number of tasks that are submitted to the system at each interval. Fourteen different runs have been designed to test the effects of these variables on the results. For each workflow the simulation was executed six times, once for each of the performance metrics used. The resulting energy usage and average time taken to execute a task were recorded for each performance metric, and are presented in Table 3.4.

The fourteen workflows that were chosen to examine the effect of modifying the three variables are described in Table 3.3[3]. Workflows 1–5 and 12 examine the results of modifying

[3] All tasks require 10000 units of simulated computation

Table 3.3: Parameter values for test workflows showing; length of workflow (simulated time steps), gap between tasks (simulated time steps), and the number of tasks submitted each time

Workflow	Total time steps	Submission gap	Tasks submitted each time
1	100	50	1
2	10000	10	1
3	10000	100	1
4	10000	10000	1
5	10000	25	1
6	10000	50	1
7	10000	50	2
8	10000	50	3
9	10000	50	4
10	10000	50	8
11	20000	50	1
12	84600	2	1
13	84600	50	1
14	84600	50	2

the time gap between task submissions. Workflow 4 is of special note as it examines a single task being submitted only once to the system. Workflow 12 is also of special note as it examines what would happen if tasks were constantly submitted over a prolonged period. Workflows 6–10 examine the result of modifying the number of tasks that are submitted at each submission time. Workflows 1, 6, 11, and 13 examine the result of modifying the number of time steps the tasks run for. The average time taken and the total energy usage of the grid when using each metric in each run of the model are shown in Table 3.4.

3.5 Results

The results from fourteen workflows of runs of the simulation are presented in Table 3.4. The results display the readings for total energy consumed and average time taken to execute a task.

Table 3.4: Simulation results showing mean time taken and energy used when using each performance metric

Workflow	Power		Flops		Wips		TTT		Ettt		Eflops ($\frac{flops}{watts}$)	
	$\bar{x}$ time	Energy used	$\bar{x}$ time	Energy used	$\bar{x}$ time	Energy used	$\bar{x}$ time	Energy used	$\bar{x}$ time	Energy used	$\bar{x}$ time	Energy used
1	35.00	14.87	36.00	14.70	35.00	14.70	36.00	14.70	36.00	14.46	35.00	14.46
2	57.64	1583.23	42.89	1553.24	42.88	1553.24	42.88	1553.24	43.91	1560.02	43.40	1555.06
3	96.71	1463.85	35.00	1453.11	35.00	1453.11	35.00	1453.11	36.00	1441.25	36.00	1441.25
4	105.00	1436.64	35.00	1436.28	35.00	1436.28	35.00	1436.28	36.00	1436.16	36.00	1436.16
5	70.34	1510.38	35.50	1480.37	35.50	1480.37	35.50	1480.37	45.10	1474.04	35.50	1480.33
6	82.94	1481.44	35.00	1470.11	35.00	1470.11	35.00	1470.11	36.00	1446.39	36.00	1446.39
7	70.56	1503.12	35.50	1480.39	35.50	1480.39	35.50	1480.39	45.18	1474.10	35.50	1480.39
8	63.47	1527.02	39.97	1499.83	39.64	1507.61	39.64	1507.61	44.01	1504.72	41.78	1508.10
9	57.00	1548.06	41.97	1527.05	41.97	1527.05	41.97	1527.05	43.47	1531.44	43.47	1531.44
10	51.55	1647.09	51.55	1647.09	51.55	1647.09	51.55	1647.09	51.55	1647.09	51.55	1647.09
11	82.81	2962.80	35.00	2940.22	35.00	2940.22	35.00	2940.22	36.00	2892.78	36.00	2892.78
12	51.65	13935.33	51.63	13935.34	51.63	13935.33	51.63	13935.34	51.63	13935.32	51.63	13935.34
13	82.71	12532.53	35.00	12437.14	35.00	12437.14	35.00	12437.14	36.00	12236.45	36.00	12236.45
14	70.44	12715.60	35.50	12524.09	35.50	12524.09	35.50	12524.09	45.24	12470.96	35.50	12524.09

3.6 Discussion

The null hypothesis states that irrespective of the performance metric used to rank nodes, the energy consumed and time taken to process the tasks will remain the same. The results support the rejection of the null hypothesis. With the exception of workflows 10 and 12, results show some change in the output for different performance metrics.

Workflow 14 can be used to illustrate this point. Workflow 14 shows differences in both the mean time taken to execute a task and the total energy consumed during processing when the nodes are ranked using different performance metrics. The results show a marked difference between the results of different metrics, particularly the Power and Ettt performance metrics that result in execution times that are different from the other metrics.

In addition to having different mean execution times the total energy consumed differed considerably. In workflow 14 the minimum total energy consumed during processing was 12471Wh when using Ettt and the maximum amount of total energy consumed was 12716Wh when using power as the performance metric. The total energy used differed substantially depending on the performance metric used. Figure 3.1 displays the total energy used for each performance metric relative to that used by Ettt.

The null hypothesis is therefore rejected, as both the average execution time and the total energy used differed depending on which performance metric was used to rank nodes in the grid.

The two performance efficiency metrics used in this simulation, Ettt and Eflops, produced no difference in either energy consumption or time taken for workflows 1, 3, 4, 6, 9, 10, 11, 12, and 13. On the other hand, workflows 2, 5, 7, 8, and 14 did show a difference in the results between these two metrics. Considering the parameter setting for the different workflows, as shown in Table 3.3, it can be seen that the workflows that showed a difference had a greater computational workload than the workflows that did not show a difference, with the notable exceptions being workflows 9 and 10 which had an extremely high workload.

From these results it can be extrapolated that when using the nodes used in this research, choosing between these two performance efficiency metrics can make a substantial difference when a moderate number of nodes, but not all of the available nodes, are occupied in the processing of tasks.

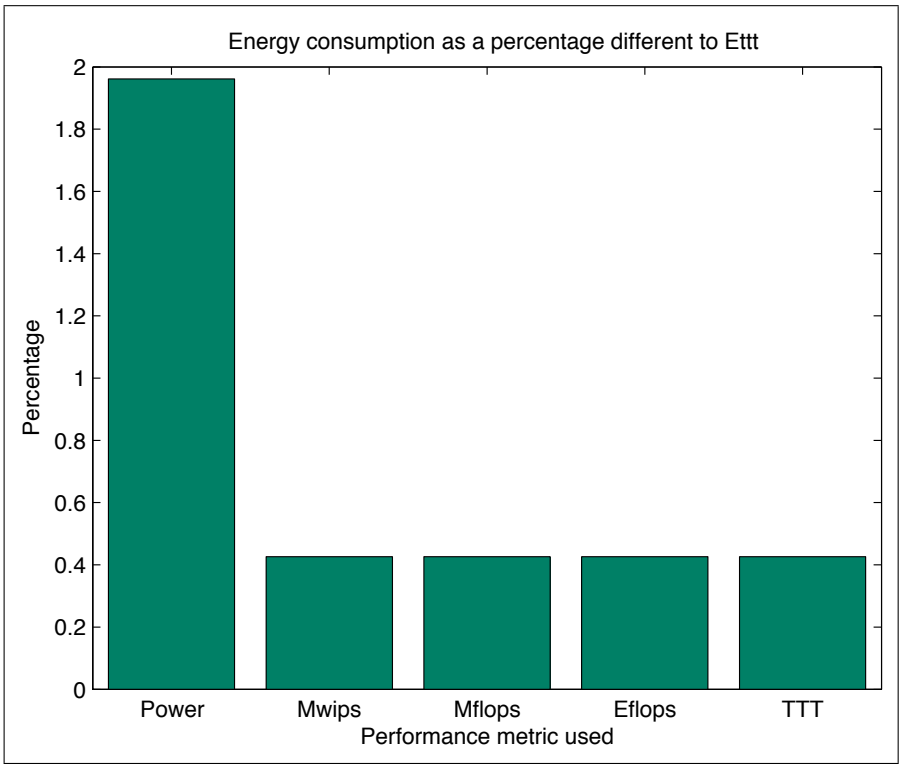

Figure 3.1: Energy consumption for workflow 14, showing the energy consumed using each metric relative to the energy consumed using the Ettt metric

3.6.1 Analysis of parameters in relation to sensitivity of results

Section 3.6 has shown that in some circumstances the choice of performance metric can result in a difference in the average time taken to execute tasks and in the total energy consumed in the grid's operation. This section examines the effect on the results of modifying the parameter values, that is, modifying the different types of workflow submitted to the simulated grid.

The time gap between tasks being submitted to the grid has a potentially considerable effect on the outcome. Workflows 1–6 and 12 examine the effect of modifying this gap. As the gap decreases, so too does the difference between the results. Workflow 12 has the smallest gap, and illustrates what happens if tasks are submitted constantly. Table 3.4 shows that there is no difference between results of any of the performance metrics in this scenario. Workflow 4 is also of special note as it examines what happens if the time gap is so large that only one task is ever submitted to the system. In this scenario only one node, the node identified by the performance metric as being the most desirable resource, processes any tasks. Although there is a difference between the mean time taken to execute tasks, the difference is not between all groups. A substantial difference is seen when using power as a metric, and this is not surprising. While it might appear to be logical in considering power alone as a metric when seeking to minimise energy usage, the nodes that draw least power (nodes 1 and 2 in Table 3.1) are also the nodes that take the longest time to process a task. If one of these nodes is selected, its lower power consumption is more than offset by the fact that it needs to run for longer to process the same task.

The number of tasks submitted to the grid at each interval can also have a marked effect on the results. Workflows 6–10 examine the result of modifying the number of tasks that are submitted at each interval. As the number of tasks increases, the difference between the results decreases. Workflow 6 (Figure 3.2), 7 (Figure 3.3), and 8 (Figure 3.4) show the diminishing proportional difference in energy consumption as the number of tasks per interval increases.

Modifying the number of time steps can also affect the outcome, although not in any systematic way. Workflows 1, 6, 11, and 13 examine the result of modifying the number of time steps the simulation runs over. In workflow 1 the difference between the maximum and minimum watt hours used is some 7%. In workflows 6 and 13 the difference is about 2.4%, while in workflow 11 it is less than 1%.

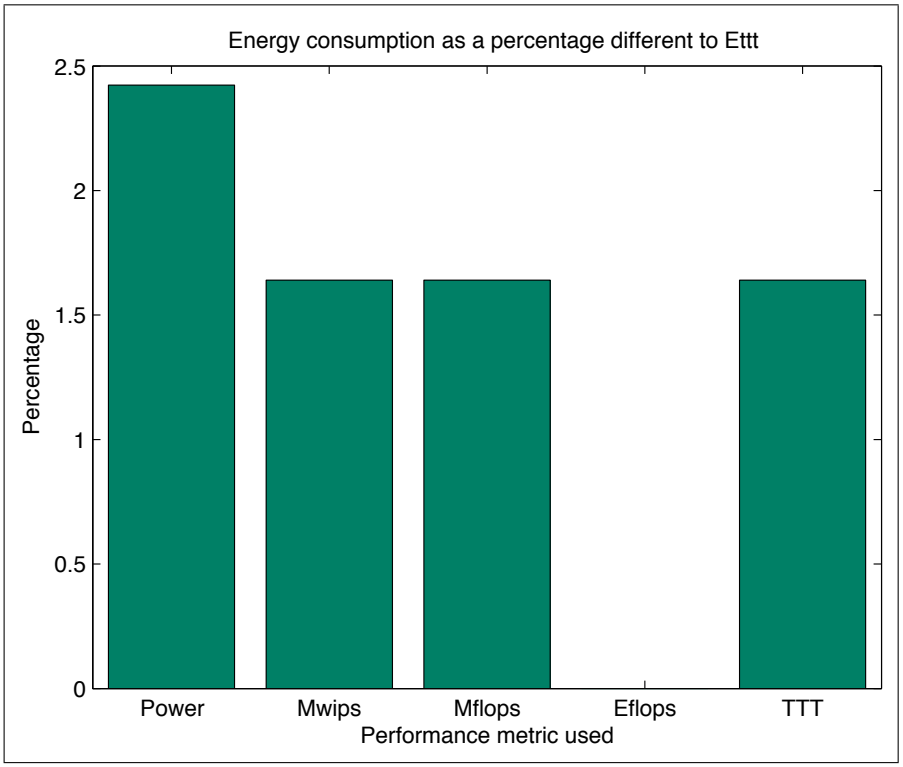

Figure 3.2: Energy consumption for workflow 6, showing the energy consumed using each metric relative to the energy consumed using the Ettt metric

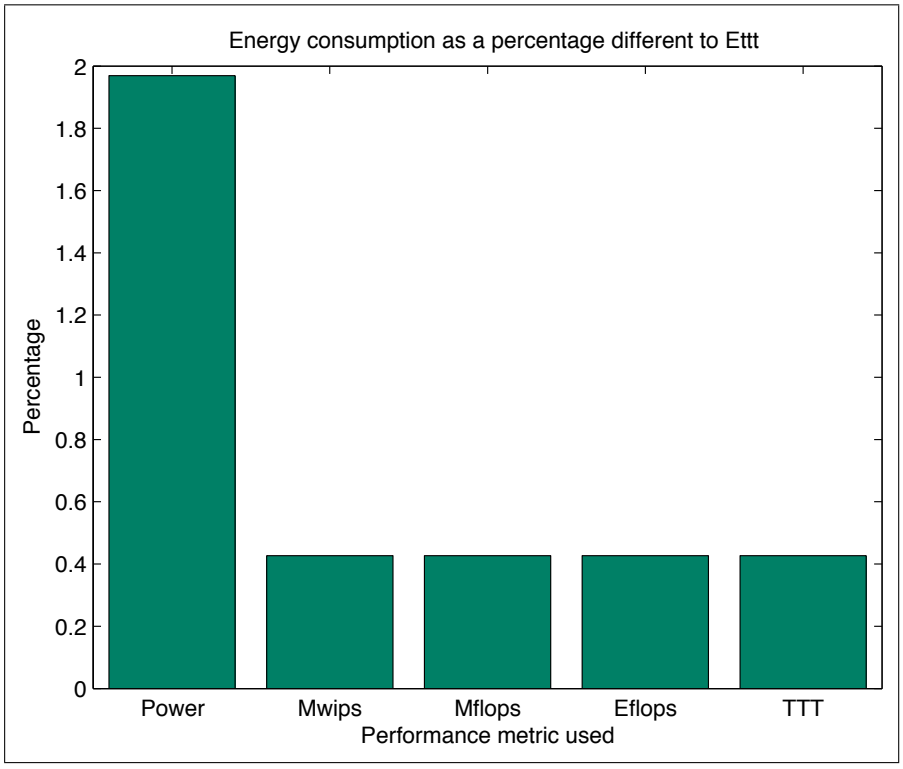

Figure 3.3: Energy consumption for workflow 7, showing the energy consumed using each metric relative to the energy consumed using the Ettt metric

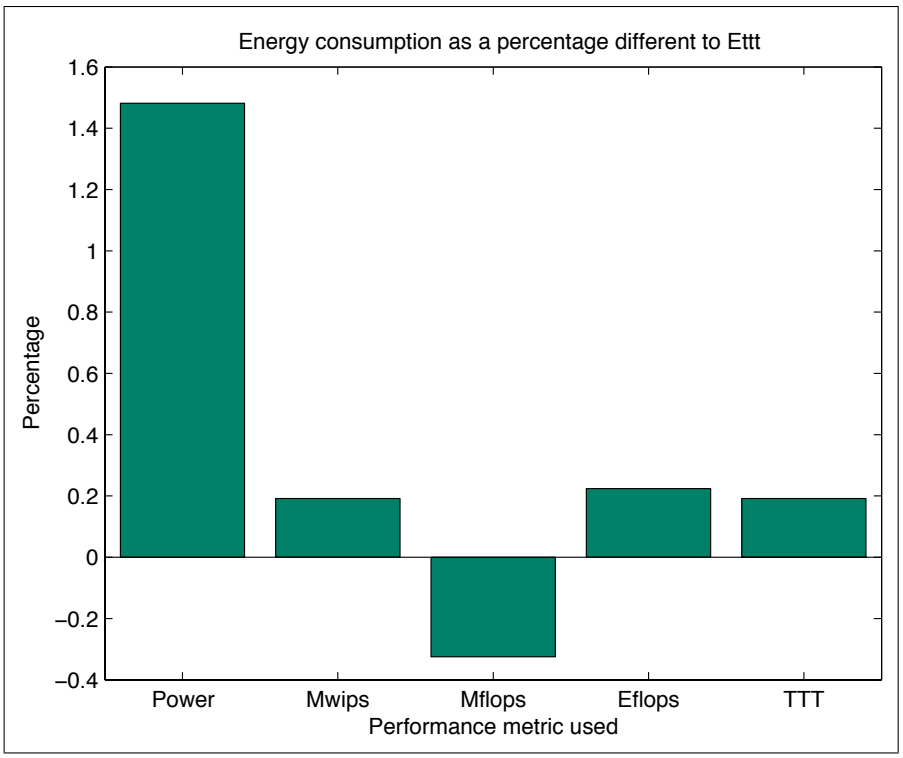

Figure 3.4: Energy consumption for workflow 8, showing the energy consumed using each metric relative to the energy consumed using the Ettt metric

3.7 Concluding remarks

The results from the simulation have shown that the choice of performance metric can make a difference to the total energy used and time taken to process tasks in a distributed heterogeneous computing environment.

The use of metrics that incorporated both a node's computational ability and power usage consistently resulted in allocations that consumed less energy than were realised when using alternate means to rank nodes. The most power efficient node is not always the most energy efficient node. In this simulation use of the power metric consistently resulted in the most energy being consumed and the longest time to execute tasks.

The node's test task time was used as the node's measure of computational performance in this simulation, as such the TTT and Ettt metrics used a perfect measure of a nodes computational ability. It is therefore not surprising that the use of the Ettt metric consistently resulted in allocations that performed better than the other metrics.

This simulation has shown that modifying the time gap between tasks, the number of tasks submitted at each gap and the total number of tasks can all have an effect on the results of using different performance metrics. In most instances there was a difference between the outcomes for some performance metrics, and the difference between the performance metrics was most pronounced when not all nodes were being used to capacity. As the grid comes closer to full capacity, the differences diminish (as shown in Table 3.4).

The simulation model presented in this chapter examined one aspect of resource allocation: the ranking of nodes.

A simulation has been employed to investigate the effect that selection of performance metrics can have on total energy used and mean task execution time. The overall energy consumption and the mean time taken to execute tasks on a small heterogeneous grid were recorded. The simulation showed that in most cases the choice of performance metric had an impact on the total energy consumed and on the mean time taken to execute a

task. This chapter has presented a performance metric that incorporated a node's power usage and processing performance, which was then used to rank each node based on its performance relative to the performance of other nodes. The metric came in various forms according to which measure of processing performance was used. The Ettt performance metric generally resulted in the least total energy consumption, but generally at a slight expense in execution time. It is clear that the workflow itself plays a substantial role in the effectiveness of resource allocation. No attempt was made to examine any other method of energy reduction, such as adjusting the power states of nodes.

3.7.1 Key findings

Key findings from the research presented in this chapter are:

- In most cases the choice of performance metric has an impact on the total energy consumed and on the mean time taken to execute a task.
- When ranking nodes for resource allocation, the use of a different performance metric results in a difference to the total energy consumed and in the time taken to execute tasks.

Chapter 4

ENERGY AWARE RESOURCE ALLOCATION: A SIMULATION STUDY OF THE USE OF ECONOMIC RESOURCE ALLOCATION TECHNIQUES TO REDUCE ENERGY CONSUMPTION

4.1 Introduction

Energy consumption is increasingly becoming an issue for high-performance grid computing. There has been substantial research on grid resource allocation, but little on energy-aware resource allocation. This chapter explores the notion of varying the resource allocation mechanism used by a grid, examining the effect of this variation on total energy consumption and on time taken to execute tasks. There are many economic resource allocation mechanisms, each with different attributes. This thesis examines three such mechanisms: the batch auction (BA), the continuous random allocation (CRA), and a pre-processed batch auction (PPBA). A simulation was created of a high-performance heterogeneous grid environment. A number of different grid workflows were devised, both artificial and actual, and each was run several times over the simulated grid, with the only difference being the resource allocation mechanism used. The total energy consumed and the time taken to execute tasks were analysed.

The research presented in this chapter has been accepted by the International Journal of Grid and Utility Computing (IJGUC) (Lynar et al., 2011). Part of the research presented in this chapter has been published in the 6th International Conference on Information Technology and Applications (ICITA09) (Lynar and Herbert, 2009).

The key contributions of the research presented in this chapter are: a novel resource allocation mechanism, incorporating the attributes of individual nodes, that can apply different

resource allocation strategies with a view to minimising both energy consumption and the time taken to execute workflows; and a simulation study of the effectiveness of the different resource allocation strategies aforementioned.

4.2 Simulation details

To examine the effect of varying the resource allocation mechanism on total energy consumption and on time taken to execute tasks, a simulation was created. The simulation model constructed was an agent-based model written in the computer programming language Java[1]. It was an extended version of the model used in Chapter 3. The computing resources simulated in this chapter are the same as those simulated in Chapter 3. For the purposes of the research in this chapter the CRA mechanism had its random number generator seeded. A class diagram of the simulation model is displayed in Figure 4.1.

The model incorporates a number of hardware resources, tasks, and resource allocation mechanisms. A stream of tasks is read into the model; the tasks are submitted to the resource allocation mechanism over time; and those tasks are allocated, using the preselected resource allocation mechanism, to available resources. In simulating the execution of a task on a resource, the time spent processing is calculated on the basis of the amount of processing involved in the task and the processing ability of the resource. For each time step, data were collected on the energy used and on the submission and completion of tasks. The structure of the simulation is described in Algorithm 4, and the source code of the model is presented on the attached media (see Appendix A.1). Algorithm 4 can be broken down into a number of steps: define resources, define workflow, choose resource allocation mechanism, switch nodes on and off, assign tasks, analyse resources, process tasks, and record statistics. The 'define resources' process creates new resources in the simulation with set characteristics for processing performance, power usage, and boot time. The 'define workflow' process creates the simulated tasks to be executed, in typical use a

[1] Java 1.5.0-16 was used for this simulation. (Arnold et al., 2000)

workflow is read in from a file, where each line in the file represents a task requiring a set number of units of processing, submitted at a particular point in time. An alternative use is to input the required computation of tasks, the number of tasks, and the time over which the tasks are to be executed over the simulated environment. The process then creates a list of tasks to be simulated. The process 'choose resource allocation mechanism', selects the resource allocation mechanism to use throughout the processing of the simulation; the resource allocator selected is defined by user input prior to the commencement of the simulation. The process 'switch nodes on and off' switches nodes on or off based on the logic described in Algorithm 8. Essentially this process attempts to ensure that the minimum number of nodes required are on at any given point in time. The 'assign tasks' process allocates tasks to nodes for processing. This process is implemented by each of the resource allocation methods. The algorithm for this process when the BA is used is described in Algorithm 5, when the CRA is used is described by Algorithm 6 and when the PPBA is used is described by Algorithm 7. The 'analyse resources' process determines how much energy a resource used during the prior simulated time step. This method is described by Algorithm 2. The 'process tasks' process is described by Algorithm 3. It examines simulated tasks and adjusts the task's required processing, based on the computational ability of the resource that was assigned to process the task. The process 'record statistics' records for each time step the number of tasks submitted, the number of tasks completed, the average time taken to complete a task, and the energy used.

A resource allocation mechanism allocates tasks to resources on the basis of some metric by which the resources are ranked. In Chapter 3 a number of performance metrics were examined, some of them novel, in a simulation that used a single resource allocation mechanism. In this chapter the intention is to examine a number of different resource allocation mechanisms, so the performance metric will remain fixed. The performance metric chosen for this chapter is Eflops, the performance metric based on a node's power and the Mflops reading obtained from the node. Eflops was chosen above the other measures examined in Chapter 3 because it is a performance metric that performed well in the prior chapter and utilises a well understood measure for computational performance.

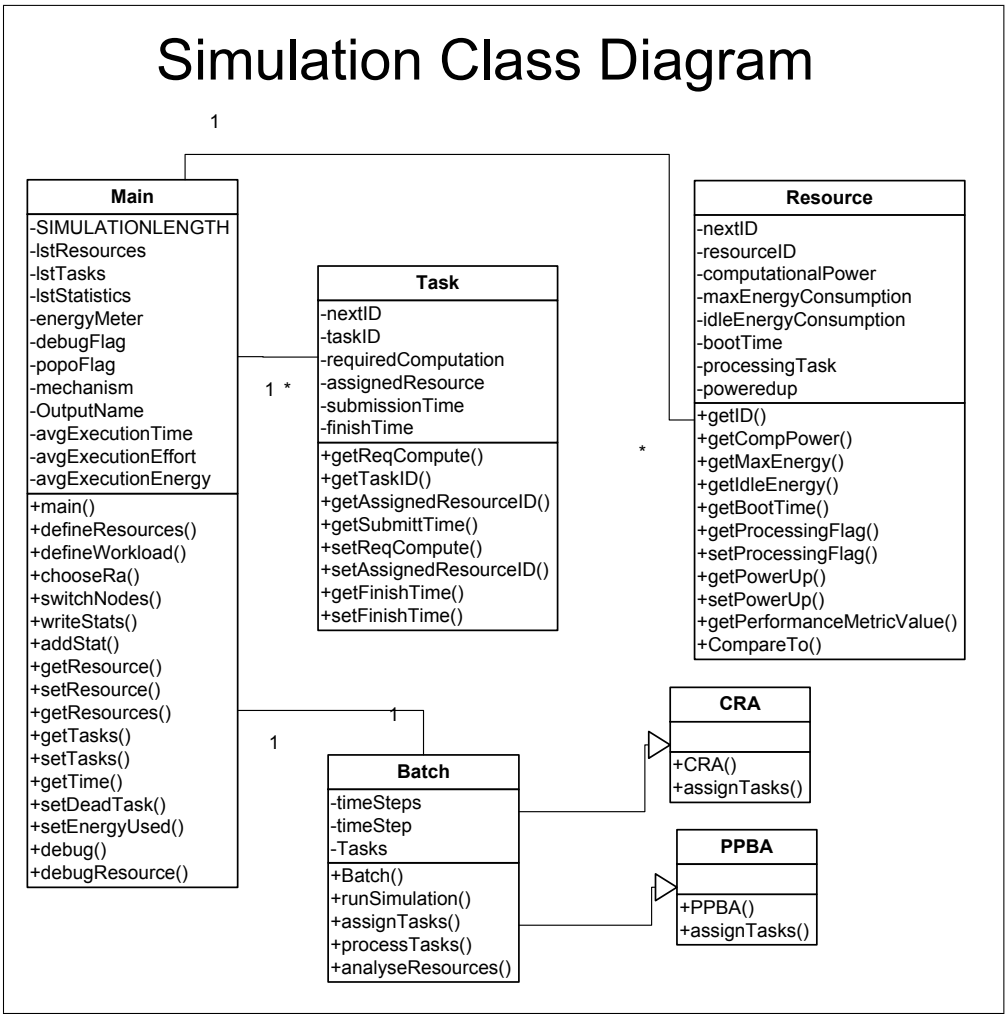

Figure 4.1: A class diagram of the simulation model

4.2.1 Hardware

The nodes used in this simulation are modelled on real nodes, from a real grid, and represent a cross-section of the available nodes in that grid. The eight nodes used in this simulation are described in Table 4.1. These eight are pairs of two of each kind of computer made available to this project. The real nodes were interrogated and readings recorded for power drawn when idle, power drawn while processing at 100% of capacity, and floating point operations per second (flops). The power readings were obtained using a Power Tech plus

Algorithm 2 Analyse resources

```
for all Resources do
    if The resource is on and The resource is processing then
        Account for the resource's energy use over the time step based on its processing power usage
    else if The resource is on and The resource is not processing then
        Account for the resource's energy use over the time step based on its idle power usage
    else
        Account for no energy usage
    end if
    Add the resource's power usage for this time step to the total energy used
end for
```

Multifunction Energy Meter Model MS-6115; the flops measure was obtained by executing the Whetstone benchmark, which produces a reading in millions of floating point operations per second (Mflops).

4.3 Description of resource allocation mechanisms

The simulation uses three economic resource allocation mechanisms, all of which are auctions. These auctions were chosen because auctions are well understood and studied mechanisms for resource allocation. The chosen auctions were the BA, the CRA, and the PPBA. In each of these auctions the agents (the computing nodes) bid for the task, and the resource allocation mechanism allocates the task on the basis of the bids. In the terminology of auctions, the bids are single shot and sealed; that is, the agents bid only once for a given task, and do not see each other's bids. Each agent's bid is based upon known data of its energy consumption and its processing ability. The greater the agent's ratio of processing ability to energy usage, the greater its bid. There are different methods of calculating this,

Algorithm 3 Process tasks

```
for all Tasks do
  if The task is assigned to a resource and the task has not finished processing then
    Set the required processing of the resource to its current required processing - the
    processing the resource can perform each time step
    if The new processing requirement of the task ≤ 0 then
      Flag the resource as not processing
      Record the finish time for the execution of the task
    end if
  end if
end for
```

all bids have been calculated using the Eflops performance metric. An agent always bids the same amount unless it is already processing a task, in which case it returns a bid of zero, indicating that it has no spare capacity with which to execute the task. With both the BA and PPBA, the tasks are allocated to the most energy efficient of the available nodes.

Batch auction (BA)

The BA is a first price sealed bid auction. It asks resources to provide a bid, waits for the resources to respond, sorts the resources, and then assigns incoming tasks to resources. The most desirable resource, the available node with the highest bid, is therefore given the first task, and so on. The BA can guarantee that an incoming task will be assigned to the most desirable resource at any given point in time; however, before allocating a task it must wait for all available resources to bid, which in large environments might prove to be a serious delaying factor. The BA used in this thesis is described in Algorithm 5.

Algorithm 4 The simulation

```
Define Resources
Define Workflow
Choose resource allocation mechanism
for all Time steps do
  if VOVO then
    Switch nodes on and off
  end if
  Assign tasks
  Analyse resources
  Process tasks
  Record statistics
end for
```

Continuous random allocation (CRA)

The CRA mechanism randomly asks nodes to bid and allocates the task to the first node that responds with a matching bid. In this simulation any task can be computed by any resource, so any bid is considered a match. The CRA cannot guarantee that the task will be allocated to the most efficient resource; indeed, it is equally probable that the task will be assigned to the least efficient resource. This approach is also known as a Zero Intelligence Trader (ZIT), described by Gode and Sunder (1993, p.121) as a trader that "has no intelligence, does not seek to maximise profits, and does not observe, remember, or learn." The CRA auction used in this thesis is described in Algorithm 6.

Pre-processed batch auction (PPBA)

A PPBA is in many ways a compromise between the BA and CRA. The PPBA remembers the prior responses of resources to asks, and assumes those prior responses to remain valid. The resource continually updates its response and there is always a response available for

Table 4.1: Data from real nodes showing power when processing ($Watts_{max}$) and idle ($Watts_{min}$), performance ($Mflops$), and Eflops performance efficiency rank

Node	$Watts_{max}$	$Watts_{min}$	$Mflops$	$Eflops\ rank$
Node 1	60	42	237	7
Node 2	62	61	234	8
Node 3	63	53	287	3
Node 4	63	55	286	4
Node 5	83	78	1544	1
Node 6	103	86	1795	2
Node 7	79	72	318	5
Node 8	80	70	318	6

the auction to sort by. The PPBA cannot guarantee that the task is allocated to the most desirable resource, as it is potentially using redundant historical data, but the advantage is that it responds instantly to requests. The PPBA is described in Algorithm 7.

4.3.1 Variable on variable off

The model examined in this simulation has the capability to dynamically switch nodes on and off based on the incoming workload. This approach to energy conservation is referred to through the literature (see page 43) as Variable On / Variable Off (VOVO). This research does not attempt to discover the most efficient algorithm for VOVO but instead uses a simple algorithm to switch nodes. VOVO is incorporated into the model to examine whether its use makes any difference to the effect of varying the resource allocation mechanism. Algorithm 8 describes the VOVO logic used in this model.

4.3.2 Description of workflows

To examine the differing characteristics of the resource allocation mechanisms employed in this model, the energy consumption and speed of the resource allocation mechanisms are

Algorithm 5 Batch auction (BA)

```
loop
  Ask all resources for a bid
  Wait a set time for the bids to be returned
  Sort resources so that the resource with the highest bid is first
  for all Submitted tasks in order of submission do
    for all Resources in order of bid value do
      if Resource is not assigned a task & resource is on & task has not already been
      assigned then
        Assign task to resource
      end if
    end for
  end for
end loop
```

recorded under different computational loads. Each mechanism is employed in 18 scenarios that includes seven different basic workflow scenarios (Table 4.2), where the tasks are submitted at random intervals within the total number of time steps; three further scenarios with constant streams of tasks (Table 4.3); and eight single-hit scenarios (Table 4.4), where all tasks are submitted in the first time step.

The seven basic workflows (Table 4.2) explore a variety of scenarios. Workflow 1 examines a small number of large tasks; workflow 2, a small number of small tasks; workflow 3, a large number of large tasks; workflow 4, a large number of small tasks; workflow 5, a small number of very large tasks; workflow 6, a large number of very large tasks; and the last, workflow 7, examines a large number of very small tasks. The high and low values were chosen to fully test the range of computational abilities of the actual nodes modelled in the simulation. A task with a simulated computational requirement of 200 can be computed in one time steps by the least powerful node and in less than one time step by the most powerful node; a task with a computational requirement of 2000 takes over eight time steps

Algorithm 6 Continuous random allocation (CRA)

```
Shuffle resources
Ask all resources for a bid
for all Submitted tasks (in order of submission) do
  for all Resources do
    if Resource is not assigned a task & resource is on & task has not already been
    assigned then
      Assign task to resource
    end if
  end for
end for
```

on the least powerful node and over one time step on the most powerful node; a task with a computational requirement of 200000 clearly places great demands on any of the nodes, while a task with a computational requirement of 20 is trivial for all of the nodes. The simulated computational ability of nodes was mapped to the nodes mflops reading. A node with a mflop reading of one was simulated to allow one simulated unit of processing per time step.

Workflows 8–10 explore a constant stream of jobs (Table 4.3); that is, the same number of jobs is submitted at each time step. Workflow 8 explores the scenario where there are fewer jobs being submitted per time step than can be processed; in workflow 9 the number and size of the jobs is increased to exactly match the amount that can be processed by the grid; and workflow 10 examines a scenario where the jobs submitted at each time step require more computation than the grid can perform in a single time step.

Workflows 11–18 (Table 4.4) explore a different scenario, in which a number of homogeneous tasks are submitted at time step one and processed immediately. Workflow 11 submits only a single task, workflow 12 submits two tasks, and so on until workflow 18 submits eight tasks.

Algorithm 7 Pre-processed batch auction (PPBA)

```
loop
  Assume the last bid from each resource to be accurate
  Sort resources so that the resource with the highest bid is first
  for all Submitted tasks in order of submission do
    for all Resources in order of bid value do
      if Resource is not assigned a task and resource is on and task has not already
      been assigned then
        Assign task to resource
      end if
    end for
  end for
  Ask all resources for a bid
end loop
```

Grid trace

The workflows described above, designed to explore the consequences of manipulating specific variables, are necessarily artificial. To fully explore the capabilities of a resource allocation mechanism a real world[2] workflow should be used in its examination (Frachtenberg and Feitelson, 2005). To more adequately examine the effect of the differing resource allocation mechanisms in a real-world scenario, a workflow[3] has been derived from a trace obtained from the DAS-2 grid at the Advanced School for Computing and Imaging at the Delft University of Technology. The trace was published as part of the Grid Workloads Archive Project (Anoep et al., 2009). For its input into the model, the workflow was transformed into tasks requiring a set amount of computation being submitted at a set instance in time. The workflow is the trace of the activity of a real grid over 56,914,643 seconds, that

[2] A workflow created from a trace of a production research grid's workload

[3] The DAS-2 grid trace and the workflow created from the grid trace are available on the accompanying media. This trace was obtained from the Grid Workloads Archive Project (Anoep et al., 2009)

Algorithm 8 VOVO mechanism

```
Count resources on = number of resources currently turned on
if (1 + number of resources currently processing) > number of tasks requiring processing then
    required nodes = (1 + number of resources currently processing)
else
    required nodes = number of tasks requiring processing
end if
if numberofresources < requirednodes then
    required nodes = number of resources
end if
for all resources in reverse order of their bid values do
    if resources required < resources on and resource is on and resource is not processing
    then
        turn resource off
        decrement resources on
    end if
end for
for all resources in order of their bid values do
    if resources required > resources on and resource is off then
        turn resource on
        increment resources on
    end if
end for
```

is, 658.73 days, and it contains 1,124,772 individual tasks. These tasks are heterogeneous and vary dramatically in terms of the computation required to process them. The trace of DAS-2, a research grid, was used because of the completeness and quality of the data in the trace, as well as its applicability to this research. This work entails the examination of a research grid, and the DAS-2 trace gives an idea of the utilisation of another research grid that is in everyday use. Furthermore, the use of a widely available workflow increases the reproducibility of the experiments conducted for this thesis. The DAS-2 trace indicates when tasks were submitted and how long those tasks took to execute. The tasks in the

Table 4.2: Basic computational workflows with homogeneous tasks

WF	Computation size	Number of tasks	Time steps	Description of computation
1	2000	200	10000	Large
2	200	200	10000	Small
3	2000	2000	10000	Large
4	200	2000	10000	Small
5	200000	200	10000	Very large
6	200000	2000	1000000	Very large
7	20	2000	10000	Very small

Table 4.3: Workflows with a constant number of identical jobs being submitted at each time step

WF	Computation
8	Less processing demand than capacity
9	Equal processing demand and capacity
10	Greater processing demand than capacity

trace are anonymised, and there is no indication of the resource on which they ran, so it is not possible to determine their exact computational requirements. Instead, a plausible computational requirement for each task was calculated based on the recorded processor time the task used and the number of processors the task executed over. These computational requirements were then converted into a real pattern of task submission. A number of figures have been produced to visualise the distribution of jobs in this trace. Figure 4.2 shows the number of tasks that were submitted per day in the DAS-2 trace, and Figure 4.3 shows the total computation required to process the tasks submitted on each day of the trace.

Table 4.4: Single-hit workflows where tasks are submitted only at the first time step

Workflow	Number of tasks
11	1
12	2
13	3
14	4
15	5
16	6
17	7
18	8

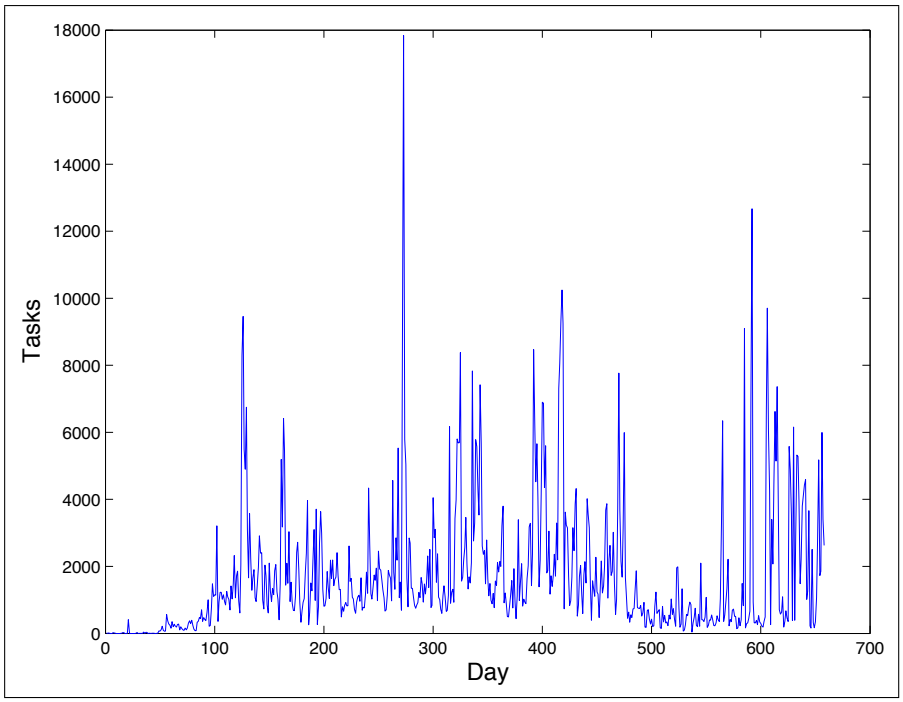

Figure 4.2: Number of tasks submitted per day in the DAS-2 grid trace

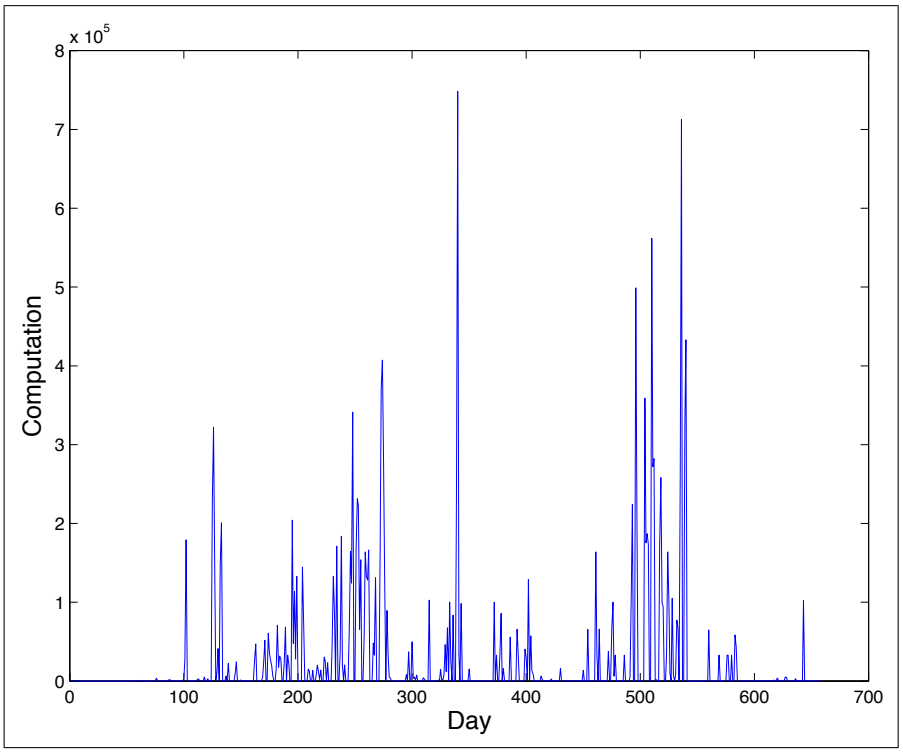

Figure 4.3: Computation required to complete tasks per day in the DAS-2 grid trace

4.4 Methodology

The research presented in this chapter involves the creation and execution of a simulation. A number of questions were asked to examine whether the use of different resource allocation mechanisms would result in differences in output. The questions are:

Q1: Will there be a measurable difference in the total energy consumption of the grid depending on the resource allocation mechanism chosen?

Q2: Will there be a measurable difference in the time it takes to execute tasks depending on the resource allocation mechanism chosen?

Q3: Will switching off idle nodes make a measurable difference to the total energy consumption of the grid?

Q4: Will switching off idle nodes make a measurable difference to the time it takes to execute tasks?

4.4.1 Assumptions

In the creation of the simulation a number of simplifying assumptions were made:

- The resources submit bids for tasks in a random order.
- When all the resources in the grid are asked to bid for tasks, a set time must be allowed to elapse by the resource allocator to ensure that all resources intending to respond have done so. This time will be measurable in seconds.
- Any resource can process any task.
- Each task processes on only one resource, not over multiple resources at once, and only requires that one resource to process.

- A resource can process only one task at a time, and it processes that task with all of its processing ability.

- There is no foreknowledge of how long it will take to execute an incoming task.

- Resources are able to be turned on and off; it is assumed that the time to turn on a resource is exactly 100 time steps and the time to turn off a resource is a single time step.

- There will be no other stream of tasks after the stream of tasks processing. That is, the nodes will remain idle after they have finished processing the last task, until the end of the simulation.

- All agents bid truthfully. Agents do not exaggerate bids to gain advantage.

4.5 Results

The results in this section will generally present task execution time and energy consumption for the BA and PPBA resource allocation mechanisms as a percentage of the values for the CRA mechanism. This reflects a default assumption of random allocation of tasks to resources. The raw results, absolute values of energy and time, are presented in Appendix A.4. It should be noted that the values for energy were measured from the start of execution of a workflow to the end of execution of the workflow, not the end of execution of the last task in a workflow.

Table 4.5 displays energy consumption and time taken to execute each workflow as a percentage of the time and energy taken by the CRA mechanism; these runs did not use VOVO. Table 4.6 displays time and energy usage for each of the three mechanisms when using VOVO, as a percentage of the time and energy taken by the same mechanisms without VOVO.

Table 4.5: Time and energy performance of BA and PPBA relative to CRA for 18 workflows and the grid trace (GT); each value is the percentage difference from the corresponding CRA performance

Workflow	Time BA	Energy BA	Time PPBA	Energy PPBA
1	0.0	-0.2	0.0	-0.2
2	0.0	0.0	0.0	0.0
3	0.0	-0.8	-0.1	-1.4
4	0.1	0.0	0.0	-0.1
5	2.4	0.0	-1.1	-0.2
6	-0.1	-1.5	-0.1	-1.5
7	0.1	0.0	0.0	-0.1
8	407	0.2	0.0	0.0
9	119	0.7	-1.9	0.0
10	800	0.0	0.0	0.0
11	-16.4	-3.2	-17.5	-3.2
12	-16.4	-3.4	-17.5	-3.4
13	-16.8	-2.4	-17.8	-2.4
14	0.6	0.2	-0.5	0.2
15	1.1	0.0	0.0	0.0
16	-16.8	-2.4	-17.8	-2.4
17	0.6	0.2	-0.5	0.2
18	1.1	0.0	0.0	0.0
GT	0.0	0.0	0.0	0.0

It should be noted that the results of the resource allocation mechanisms examined are deterministic; it is certain that the same result will be produced each time when the inputs are the same. Each individual test is, in effect, the whole population. Inferential statistic tests such as the Kruskal-Wallis test are neither required nor appropriate to determine the properties of a population (McClave and Dietrich, 1991).

Table 4.6: Time and energy for each mechanism executing workflows 1–7 with VOVO compared to the same workflows without VOVO

Run	Mechanism	Energy % non-VOVO	Time % non-VOVO
1	BA	-84.4	0.0
1	CRA	-82.3	1.3
1	PPBA	-84.5	0.0
2	BA	-84.7	0.0
2	CRA	-88.2	0.0
2	PPBA	-84.9	0.0
3	BA	-35.1	0.6
3	CRA	-36.0	0.5
3	PPBA	-80.0	0.0
4	BA	-31.3	0.5
4	CRA	-50.4	0.0
4	PPBA	-84.2	0.0
5	BA	-13.9	-0.1
5	CRA	-14.2	2.6
5	PPBA	-15.8	-0.3
6	BA	-80.5	0.0
6	CRA	-73.3	0.0
6	PPBA	-80.5	0.0
7	BA	-31.3	0.5
7	CRA	-50.4	0.0
7	PPBA	-84.2	0.0

4.5.1 Analysis of the grid trace

The grid trace used in this thesis was discussed in Section 4.3.2 on page 83. Because the grid trace used was large, it was decided to concentrate on a small subset of the grid trace as well as examining the grid trace as a whole. A specific 24-hour period was selected for this detailed examination. Figure 4.4 shows the computation that was requested throughout the day[4]. Figure 4.5 shows the first hour of this day, where it can be seen that there were

[4] This day started 10730399 seconds from the start of the grid trace. Usage of the grid resources has ramped up by this point.

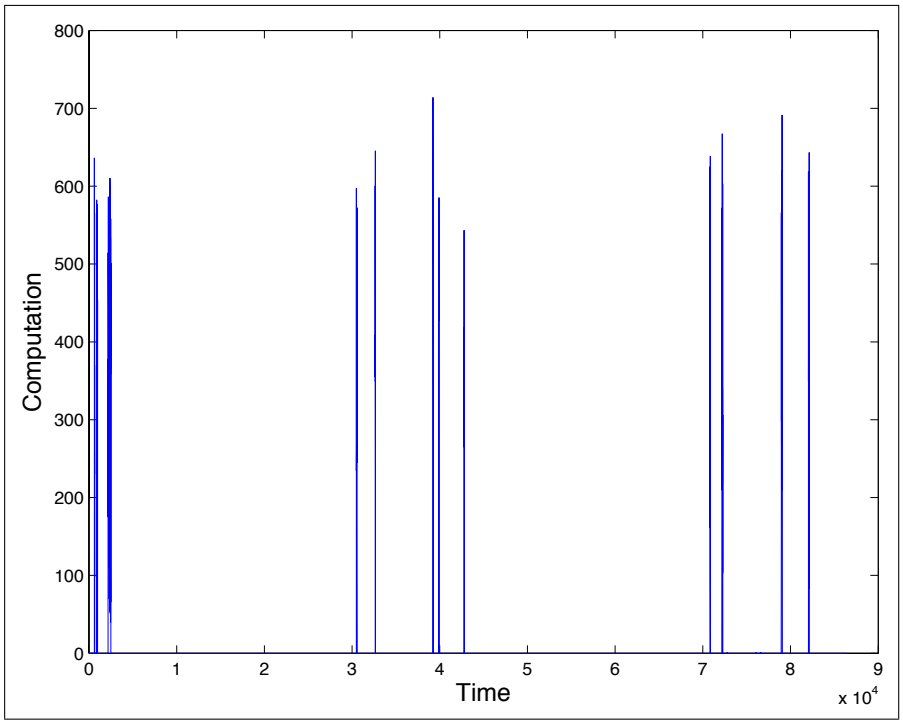

Figure 4.4: Computation required in a one-day section of the DAS-2 grid trace

several spikes in requested computation. Figure 4.6 shows a plot of total task submission and completion based on the auction used in the 24-hour period, and Figure 4.7 shows the cumulative energy used by each mechanism relative to the PPBA in the 24-hour period.

It might seem reasonable to assume that the BA mechanism will generally be slower than the other mechanisms because of the delay involved in waiting for resources to bid[5]. However, Figure 4.7 shows that in certain circumstances the BA does perform better than the other mechanisms. The BA's inbuilt delay sometimes results in a more desirable allocation than if there were no delay. If the most desirable resources are processing tasks when a new task is allocated, that task would go to the next most desirable resource. In some cases it would be more efficient if that task's allocation were delayed so that it could run on a more efficient resource. This situation occurs in the first hour of the day presented in Table 4.7

[5] In this simulation the batch auction waited 10 time steps.

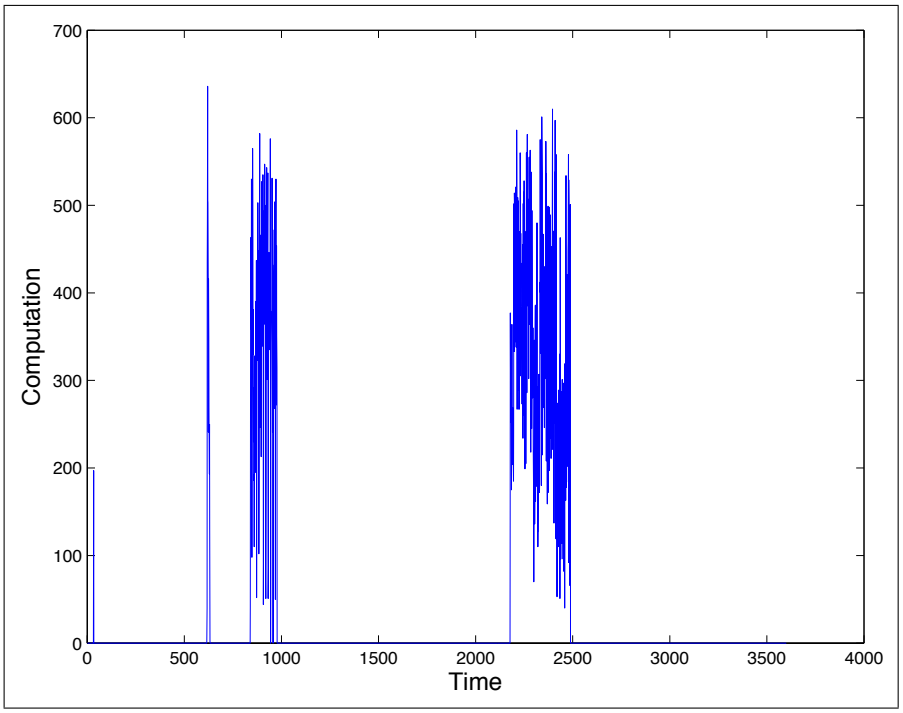

Figure 4.5: Computation required in the first hour of the one-day section of the DAS-2 grid trace

– which, as shown in Figure 4.5, has several spikes in required computation.

A further advantage of the BA in these circumstances is that it is prepared to wait for accurate current bids, while the PPBA relies on the last known bid from each resource. If tasks are being allocated at a high rate, the historical bids used by the PPBA are more likely to be inaccurate. In short, the PPBA would immediately assign the new tasks to what was the best available resource when bids were last received, while the BA, in waiting for all resources to respond, is more likely to find a node that is actually available to execute the task.

4.6 Discussion

The first question (Q1) asks: "Will there be a measurable difference in the total energy consumption of the grid depending on the resource allocation mechanism chosen?"

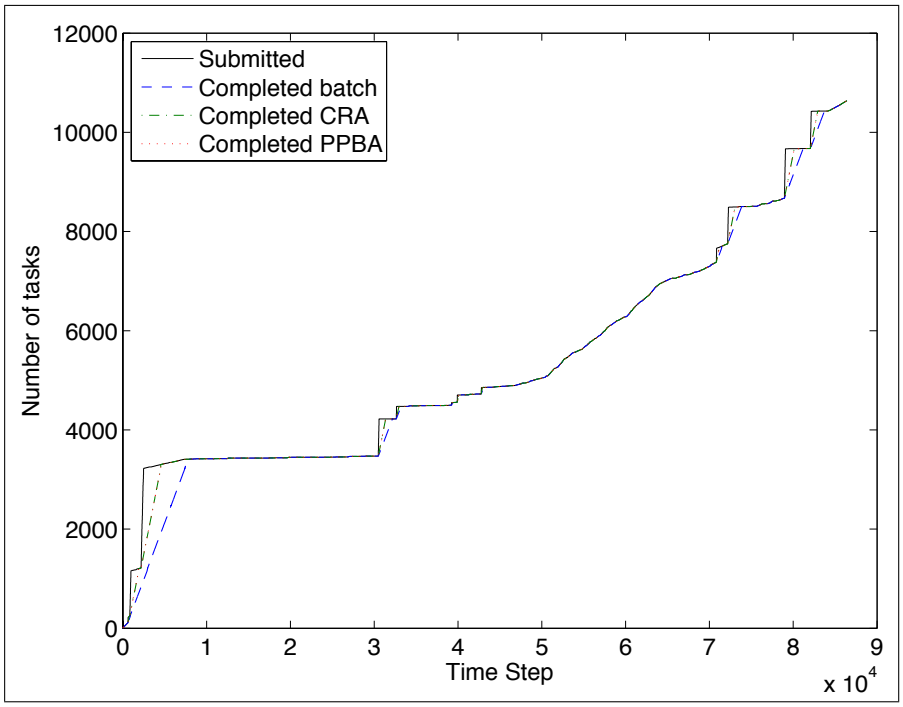

Figure 4.6: Task submission and completion in the one-day section of the DAS-2 grid trace; the lines for CRA and PPBA substantially overlap on this figure

The simulation shows that there are measurable differences in the energy consumed when using different resource allocation mechanisms. Workflows 5 and 6 highlight these differences as they compute the largest jobs. In workflow 6 the CRA mechanism uses the most energy, 146,407Wh, and the BA uses the least energy, 144,157 Wh; the difference is a substantial 2,250Wh or 1.5%. Although Table 4.5 shows no difference between mechanisms for the grid trace, there is in fact a small difference. The least energy efficient allocation mechanism for this workflow is the BA, which consumes 8,197,120 Wh. The most efficient is the PPBA, which consumes only 8,195,985 Wh. A difference of 0.01% might appear insignificant, but in absolute terms the saving is a not inconsiderable 1,135 Wh.

The overall impact on energy consumption of changing the resource allocation mechanism appears to be small. The largest energy saving, realised in workflow 12, is 3.4% of the energy consumed. The average saving across all workflows is only 1%, and the grid workflow trace

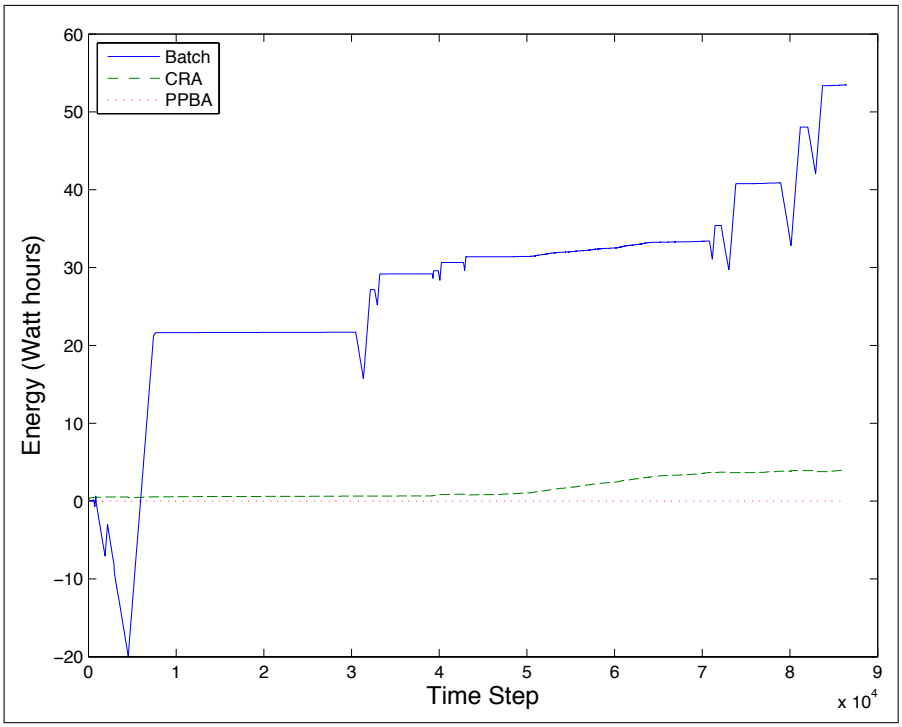

Figure 4.7: Energy consumption relative to PPBA in the one-day section of the DAS-2 grid trace

resulted in an energy saving of only 0.01%. This result may be due to the bursty nature of task submission in the trace. However, while the energy savings as a percentage are not large, the realised energy savings over the lifetime of the grid may be. Grid resources typically run for several years, and have many more than eight nodes. A one percent saving in energy over the life of many grids could be substantial.

In summary, when the tasks are large and the time taken to process the tasks is considerable, circumstances that are highly likely to be found in a grid environment, the choice of resource allocation mechanism can make a measurable difference to the consumption of energy by the grid.

The second question (Q2) asks: "Will there be a measurable difference in the time it takes to execute tasks depending on the resource allocation mechanism chosen?"

The results show that in some workflows there are measurable differences. In workflow 5,

one of the longest-running workflows, the BA takes 2.4% longer than the CRA to execute the workflow, while the PPBA is 1.1% faster than the CRA. While the differences in total task execution time are generally small, there are sometimes far greater differences in the average time taken to execute a task. This is because the total execution time includes time spent waiting for resources to bid and tasks to be allocated, whereas the average time taken to execute tasks is based solely on the time actually spent executing tasks. In workflow 6 the average time taken to execute tasks ranged from 138 time steps for the PPBA to 555 time steps for the CRA. These differences were also seen in the results of the grid trace workflow, in which both the CRA and the PPBA took a mean time of 251 time steps to process a task, while the BA averaged 504 time steps to complete a task. This indicates that the time tasks take to execute can be measurably altered by the resource allocation mechanism selected.

The third question (Q3) asks: "Will switching off idle nodes make a measurable difference to the total energy consumption of the grid?" All of workflows 1–7 were tested with and without switching off idle nodes (see Table 4.6). In every one of these workflows the total energy consumption was substantially lower when running VOVO than it was for the same workflow, running the same mechanism, without the use of VOVO. For example, workflow 1 using the BA consumed 1437Wh with VOVO turned off, compared with only 224Wh with VOVO turned on, a saving of nearly 85%. In workflow 3 the results were similar, although less pronounced. Without VOVO processing the BA consumed 1455 Wh, while with VOVO the energy consumption was 944 Wh, a saving of 35%. These results show that switching off idle nodes results in substantial reductions in energy consumption.

The fourth question (Q4) asks: "Will switching off idle nodes make a measurable difference to the time it takes to execute tasks?" Of the 21 runs (seven workflows run on three mechanisms) in workflows 1–7 and 11 showed no difference in overall execution time when VOVO was turned on or off. This is possibly because even when VOVO was on in those runs, there were never any idle nodes to turn off. Of the runs that did produce different results, seven took longer to execute with VOVO and three took less time than with VOVO. Considering again the average time to execute tasks within the workflow, there are once

more greater differences. In 14 of the 21 runs, the mean time taken to execute tasks was greater with VOVO than without it. The results suggest that switching off idle nodes will make a measurable difference to the time it takes to execute tasks, and in general that difference will be that execution takes longer.

Both the time taken to execute tasks and the energy utilised in execution have been shown to alter substantially with choice of resource allocation mechanism. These differences are accentuated when there are many tasks that require substantial computation, and are much smaller as the size and the number of tasks diminishes. In summary, the choice of resource allocation mechanism can make a measurable difference to the time taken to execute tasks and the energy used in the execution of tasks.

The results highlight a number of interesting phenomena. A zero intelligence approach to resource allocation may be appropriate in some circumstances; as the task size and the number of tasks increased, the differences between auctions were amplified; when a point of saturation was met the difference between the auctions was negated; and different auctions performed better under different circumstances.

In some instances the CRA performed as well as other auctions. The CRA mechanism presented in this model essentially allocated tasks to random resources, as the resources were shuffled prior to allocation in order to simulate the effect of resources responding to requests at different times. The results suggest that in some circumstances the use of a zero intelligence approach to resource allocation, one in which tasks are randomly allocated, can result in an allocation that is as good as competing mechanisms (workflows 2, 10, 15, 18 and the grid trace). This is particularly the case when there are many small tasks to be processed or the resources are saturated with tasks.

The results of this simulation study suggest that in some circumstances, resource allocation that incorporates the energy efficiency of the nodes can on its own make a difference to the total energy consumption of the distributed environment. This was shown in the simulation results where both the BA and PPBA mechanisms consistently resulted in lower consumption of energy when processing the same workflow as the CRA mechanism.

However, once saturation occurs, the selection of resource allocation mechanisms makes no difference. Workflow 15 was unique in that it examined what happens if saturation occurs, that is, when the number of tasks submitted is greater than or equal to the number of available nodes. In workflow 15 the number of tasks submitted exactly matched the number of available nodes. Saturation resulted, with each task allocated to a resource and all resources were in use. The tasks used were homogeneous, so the allocation made no difference to the energy consumption or the time taken to execute the tasks. From workflow 15 it can be seen that when the grid is perfectly saturated, altering the resource allocation mechanism will make no difference to the outcomes of the system.

With regard to energy consumption it can be seen that the PPBA performed best in most circumstances, but there were workflows for which other auctions performed as well as or better than the PPBA. The performance of the CRA has already been discussed: it performed well when tasks required little computation. The BA has also performed well in a limited number of circumstances. In workflow 6, the BA achieved the lowest energy consumption of any mechanism. When there was a large number of large tasks, it became increasingly important for tasks to be allocated to the most efficient resource. In this simulation, where resources never failed, the BA had two advantages over the PPBA: the time spent waiting for resources to respond, which could allow enough time for more desirable resources to complete processing; and the fact that the bids are accurate currently, rather than historically. The BA performed best when there are a large number of tasks that required substantial computation.

4.7 Concluding remarks

This simulation study showed that in some circumstances energy may be saved through the use of resource allocation alone. The use of different resource allocation mechanisms offers the opportunity to save energy and time. The results presented have shown that in different situations the use of different resource allocation mechanisms results in different allocations. Altering these allocation mechanisms can make a difference to the time tasks take to execute,

and over time can make a substantial difference to the total energy consumption of the grid. The results discovered in this chapter suggest that when using resource allocation alone, the savings in time and energy are small. Later in this thesis a real implementation of these resource allocation mechanisms are examined on a cluster and grid environment.

4.7.1 Key findings

Key findings from the research presented in this chapter are:

- The use of a resource allocation mechanism that incorporates the attributes of individual nodes can conserve energy and minimise execution time.
- Different resource allocation mechanisms are suited to different workflows.
- VOVO offers the possibility to accentuate the energy savings of resource allocation.

Chapter 5

CREATING THE TEST PLATFORMS: DISTRIBUTED COMPUTING ENVIRONMENTS CONSTRUCTED FROM ELECTRONIC WASTE

5.1 Introduction

Research institutions routinely use HPC equipment, either to run experiments or for training and teaching purposes. Historically, supercomputers have been employed in processor-intensive applications such as computer modelling, but their high performance comes at a cost. While supercomputers yield the highest performance feasible, they also have historically had the highest cost of any system. Grid computing promises to be one way of approaching the computing power of a supercomputer at a relatively low cost (Sterling et al., 1999). Modern computing power, parallel programming and clustering techniques allow for the creation of HPC environments at a relatively low cost, but this lower cost is still too much for many institutions, and typically involves the purchase of new computers that will later require disposal.

This chapter examines the creation of a cluster test platform out of computers that had previously been written off and were going to be sent for recycling. The cluster was made of entirely recycled components that were obtained at no cost to the author. The operating system and all of the software used were also obtained at no cost to the author. The chapter goes on to explore how the creation of grids or clusters can help to mitigate e-waste, and what limitations and advantages there are to using such a system. The research performed later in this thesis examines energy conservation in distributed computing environments through resource allocation. To perform this research a number of dedicated distributed computing environments were required. In the absence of the availability of modern cluster

resources for exclusive use, e-waste resources were utilised as a tool in this research. The systems examined in this chapter form part of a cluster and grid environment that was used to examine different economic resource allocation techniques for distributed computing, as discussed later in this thesis.

An argument is made that in some circumstances the use of e-waste resources could be of benefit. Exactly where and when they would be of benefit is a topic for future research.

The research presented in this chapter has been published in the International Journal of Information Systems and Social Change (IJISSC) (Lynar, Herbert, Simon and Chivers, 2010).

The key contribution of this chapter is a study on the use of grids to mitigate electronic waste and the advantages and limitations of such a process.

5.2 The cluster test platform

The cluster used in this chapter consists of eight computing nodes. All computing nodes are standard commodity desktop computers aged between 4 and 6 years old. All computers contain 100Mb/s network interface cards and are connected together with a 100Mb/s switch. Each system used in the cluster has a single Pentium 4 processor with a single core and a clock speed of between 2.4 and 2.66 Ghz. Table 5.1 lists the computer systems used[1], which are representative of the type of commodity hardware that is commonly disposed of in landfill or sent to recycling programs. The final item listed in the table is a reasonably new computer that will be compared with the cluster in Section 5.7.

The OpenMPI implementation of the Message Passing Interface (MPI) was used to enable the execution of parallel applications on the Debian operating system (Software in the Public Interest Inc, 2007). The operating system was created with the Debian live project (Baumann, 2008) so that it could be run without interacting with the hard disk drives of

[1] Technical details on the nodes used can be found in Appendix F on page 221.

Table 5.1: Nodes used in the e-waste cluster

Node name	Node ID	Processor	Memory (MiB)
Computer 1	CN1	Pentium 4 2.4	256
Computer 2	CN2	Pentium 4 2.4	256
Computer 3	CN3	Pentium 4 2.4	256
Computer 4	CN4	Pentium 4 2.4	256
Computer 5	CN5	Pentium 4 2.4	256
Computer 6	CN6	Pentium 4 2.4	256
Computer 7	HPN1	Pentium 4 2.66	512
Computer 8	HPN2	Pentium 4 2.66	512
New Computer	NN2	Athlon 64X2 2.8	1024

any computers in the cluster. This operating system[2] is loaded from a CD during boot, and subsequent nodes are booted from the same CD across the network. The use of a live CD which is capable of booting nodes makes it simple to add nodes to the cluster or to remove them. To add a node, its basic input/output system (BIOS) settings are adjusted to boot from the network. Once this is set, the node is plugged into power and the network switch and turned on, and when it has finished booting the operating system it becomes part of the cluster.

5.3 Electronic waste and environmental costs

E-waste is increasingly becoming an issue for both developing and developed nations. There are a number of reasons for this including the large and growing quantity of e-waste, the methods of disposal and deconstruction, the quantities of non-renewable resources in e-waste, the potential environmental and health effects of incorrect disposal, and the potential for damage to the reputations of companies and countries. Current research on these issues has been discussed in Chapter 2. The increasing magnitude and importance of e-waste has

[2] The custom Debian live make script is available on the attached media.

led the author to research novel ways of productively reusing personal computers from the e-waste stream.

Apart from reducing the quantity of e-waste it is believed that it's desirable to increase the life of older computers in-order to mitigate the environmental costs associated with the creation and subsequent disposal of new hardware. The manufacture and pre-manufacture of computers and computer components can make up a considerable proportion of the total environmental impact of a product. A life-cycle assessment of a desktop computer made in Korea in 2001, excluding the monitor, has revealed that over 95% of the environment impact of the product was made during the pre-manufacturing process (Choi et al., 2006, pg. 125). Other manufacturers have claimed that the environmental impact of the manufacturing process is a much lower percentage of the total usage. A modern server model MB449 is said to use only 10% of its life-cycle "total greenhouse gas emissions" during production (Inc, 2009). Although it should be noted that in this instant 10% is 416 kg CO_2e (Inc, 2009). Other consumer products by the same manufacturer have quite different environmental footprints. A new desktop computer (model MC508) is said to use only 1240 kg CO_2e of total greenhouse gas emissions, 39% (399 kg CO_2e) of which are created during production (Inc, 2010). Ultimately the majority of the difference in estimated green house gas emissions between the two can be attributed to estimated usage. As usage increases so to does total emissions but the percentage of emissions used in production diminishes. Extending the life of computers will further diminish the environmental costs of production as a percentage of total environmental costs. However, the estimated CO_2e emissions from usage component of the life-cycle CO_2e emissions assumes that all electricity generated produces CO_2e emissions. It could be argued that it is possible to power distributed resources from renewable or low CO_2 emitting energy sources.

5.4 High-performance computing from e-waste resources

HPC resources of commodity computers can be constructed out of commodity e-waste components to form usable computing resources, which can then be used for the same

applications as an HPC resource not built from e-waste.

Streicher-Porte et al. (2005) trace the entire life-cycle of the personal computer in Delhi, from production, through sale and consumption, reuse and refurbishment, to the material recovery in the mainly informal recycling industry. The study reveals that prolonging the lifespan of a personal computer creates value by means of refurbishing and upgrading activities, and slows down the flow rate of the whole system. They conclude that life-prolongation is one of the simplest ways of preventing an uncontrolled increase in environmentally hazardous emissions by the recycling sector (Streicher-Porte et al., 2005).

The use of e-waste computers is beneficial in two ways. First, it has the potential to defer the construction of new computer systems. If a cluster or grid would otherwise be constructed of new computers, these new computers, like all computers, will eventually require disposal, adding to the growing e-waste issue. The use of e-waste as a substitute for new computers in a cluster could reduce the construction of computers that would ultimately end up as waste. Second, the construction of clusters from e-waste saves money, as e-waste clusters can be set up for little or no money. Most or all of the required resources can be freely obtained. All components of the cluster constructed in this research were obtained at no cost except for labour.

It must be acknowledged that there are a number of limitations to using e-waste computer systems for HPC environments. These limitations need to be explored to understand the full cost of using e-waste as a substitute for new computers in a HPC resource.

5.5 Advantages of using e-waste resources

There are many advantages to using e-waste in cluster computing. This section discusses the advantages of price and of resource-saving.

Many four-year-old personal computer systems can be obtained at no cost, as occurred with this research. The cluster used in this work was constructed entirely out of donated

equipment. All software used in the creation of this cluster was obtained at no cost, being free and open source software (FOSS). All connecting equipment, including network switch and cables, were also obtained at no cost. The total hardware and software price of this cluster was nil. The cost associated with the creation of the cluster was the time spent setting it up and sorting through the e-waste to identify suitable equipment. Setting up the operating environment and the cluster took one person three days, but it is anticipated that subsequent clusters would take considerably less time to set up. It is unlikely that setting up a similar-sized cluster with new hardware would have taken significantly less time. For this reason it is believed that clusters of e-waste can be cost effective as they provide some additional utility to an organisation at little cost. It should be noted that there are costs to an organisation of setting up such a facility that have not been accounted for, these costs include: the cost of floor space, air-conditioning, and electricity.

Many personal computers that would otherwise be recycled, dumped, or stored indefinitely could be given a second life as nodes in a cluster of reused commodity workstations. This is not an end-of-life solution for the computers; the savings would come not from the reuse of the computers but from not manufacturing new computers. If a cluster can be created from reused workstations and can meet the needs that would otherwise require a cluster or grid of new computers, then the creation of the cluster of reused computers would negate the need for the purchase and construction of the cluster of new computers. The savings would thus be the financial and resource savings of not manufacturing and purchasing a cluster of new computers. However, constructing a cluster of reused computers is not entirely without costs. As with all computing resources the computers would still need to be set up and would still consume electricity. Moreover, the nodes in the cluster would eventually need disposal.

A similar cluster constructed of new hardware may perform considerably faster, but at a greater expense. This chapter argues that a modest level of processing power can be obtained for little cost, and that it is therefore possible in some circumstances to replace a cluster of new computers with one of reused computers that would otherwise have been discarded. This project required dedicated hardware for an extended period of time to

perform spasmodic processing. For this purpose e-waste resources were more than adequate.

5.6 Limitations of using e-waste resources

The computers used in the cluster did not execute applications as fast as newer computers. If speed of processing was the objective then the cluster would require a greater number of nodes than a comparable resource made of newer hardware. The equipment obtained was nearing the end of its designed life-span, so hardware faults were to be expected, and thus these resources should not be considered for some applications. The added risk of hardware failure needs to be incorporated into decision making when assessing the viability of such a resource. The computers were old and therefore consumed considerable power relative to their computational performance. Although the individual computers did not consume any more power than a new computer the power consumption to processing performance ratio was higher.

The disused personal computers built into an e-waste cluster are typically three years old and entering their fourth year of service. These computers may still seem to perform reasonably fast, but their speed must be considered relative to that of a new computing resource. Although there is no way of knowing what future computational speeds will be, there are past trends that can be examined to give a indication of what future computational speeds might be. In 1965 Moore stated that

> "The complexity for minimum component costs has increased at a rate of roughly a factor of two per year. Certainly over the short term this rate can be expected to continue, if not to increase. Over the longer term, the rate of increase is a bit more uncertain, although there is no reason to believe it will not remain nearly constant for at least 10 years. That means by 1975, the number of components per integrated circuit for minimum cost will be 65,000. I believe that such a large circuit can be built on a single wafer." (Moore, 1965, p.2)

There is research that indicates that manufacturers have kept pace with Moore's law and are striving to continue that pace (Bai, 2009). An increase in the number of components on chips does not directly translate into an increase in benchmark performance or an increase in application performance. This is because the number of components per integrated circuit does not directly translate into usable computational performance, there are other limitations to computational performance. However, if the usable speed of processors were to double every year ceteris paribus, this would impose a limiting factor on the useful life of a computer. A cluster of e-waste computers n years old would need to have 2^n nodes to perform the same number of floating point computations per second as a new single computer.

If a cluster of 100 disused computers is constructed using computers that are four years old, how long will it be until that cluster is only as powerful as one new computer? If computational performance doubles each year, the answer is three years.

By the time the computers were acquired they were already in their fourth year. This means that the combined processing power (as measured in FLOPS) of 16 of these old computers would in theory match that of one new computer. Three years further on, it would take 128 seven-year-old computers to match the processing power of one new computer.

This does not mean that the conversion of e-waste into a distributed computing resource is a pointless exercise. What it does mean is that for the project to be of benefit to the end users, in many cases the computers need to be acquired in reasonably large numbers, and to be rotated out of service at reasonable intervals. Alternatively the resources could be used for purposes that do not use the resource for its computational ability, purposes such as teaching and some research projects. Although e-waste computers can be reused, they can only be reused for a time, as they will eventually offer little real benefit to the user.

The conclusion is not that e-waste computer systems should be used indefinitely in a cluster, but that such a cluster may usefully extend the life of the e-waste systems. These clusters may, in some instances, be used for there high performance but only for a relatively brief time.

5.7 Examination of the cluster

The cluster was examined in relation to its computational performance, reliability and power consumption and compared to a new computing resource.

5.7.1 Computational performance

To measure the computational performance of the cluster a common linear algebra algorithm was executed on a recently purchased computer and on the cluster of eight computing nodes. The time taken to execute the task was recorded. A synthetic benchmark, the High Performance Linpack (HPL) benchmark (Petitet et al., 2008), was used over the cluster and on the individual new computer. The HPL benchmark is a parallel benchmark that runs over MPI using the single process multiple data (SPMD) model of parallel computing (Wang et al., 2004). The HPL benchmark was chosen because it is the benchmark used in the top 500 supercomputer list (Top500.org, 2007).

The benchmark was configured and tuned in accordance with the accompanying documentation (Petitet et al., 2008). The HPL results stated that the cluster can perform at a maximum of 6.32 Gflops and the individual new computer at a maximum of 2.235 Gflops. It should be noted that the problem size for the cluster was significantly larger: the individual new computer would not be able to process as large a problem.

The HPL benchmark may not be suitable for an accurate measurement of cluster performance when heterogeneous hardware is used (Wang et al., 2004); it may underestimate the performance of a cluster with heterogeneous hardware, as the problem size is limited by the computer with the smallest amount of available RAM (Loreto et al., 2005). For this reason a second test was conducted to give an additional indication of performance.

The second test was designed to measure the time taken to execute the benchmark. The HPL benchmark was set to divide up the problem so that each node would have to process at least two sets of data. This test was designed not to gain an accurate picture of the

number of Gigaflops the cluster and node could perform, but rather to gain an idea of how quickly the cluster and the individual new computer would compute the same SPMD application.

The GNU/Linux `time` command was used to compare the performance of the two systems by measuring the time it took to execute the same application over them. The time-based test was executed ten times on both the cluster and the new computer; the cluster executed the task in an average time of 156 seconds, and the new computer in an average time of 189 seconds. Figure 5.1 displays a bar chart of the results.

The test of time taken was executed 10 times on the cluster and 10 times on the new computer. The cluster consistently outpaced the new computer. The results were not normally distributed according to the Shapiro-Wilk W test, which returned a p-value of < 0.0001 for the new node results. The Wilcoxon test was used to compare the execution times, and showed the difference to be significant ($p = 0.0002$ DF= 1 ChiSquare= 14.2857) at the 0.05 confidence interval[3]. The results indicate that a cluster of obsolete computers may be able to perform at least as well as a new computer.

5.7.2 Reliability

The cluster presented in this paper was created from a small subset of the computers received by this project. In total 56 computers were received by the time the cluster was created. Of these 56 computers, only 12 had some form of hardware fault, and most of these faults were in the hard disk drive or the floppy disk drive. As disk drives do tend to fail earlier than non-moving parts, it was decided to remove these drives and construct a cluster that did not require them. Reliability of the other components of the e-waste computers did not appear to be an issue.

[3] Raw statistical output is presented in Appendix B.0.1.

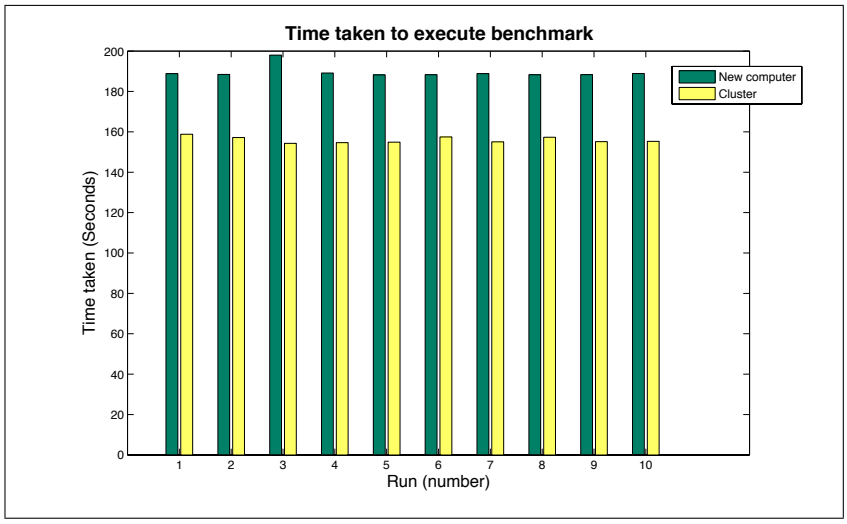

Figure 5.1: Performance test showing the time taken to execute each run on both the new computer and the cluster

5.7.3 Power Consumption

Power consumption was measured using an energy meter[4] that was accurate to ±10W (Power Tech Plus, 2008). Measurements were taken for each node of the cluster and for the new computer while the computers were idle, and again while they were executing a computationally intensive benchmark. All measurements were taken at room temperature (21°C) once the computers had been operating for at least 10 minutes. Comparative power consumption to performance was measured for all nodes in the cluster of eight nodes and for the new computer that was not part of the cluster. The cluster was measured as consuming 468W of power whilst idle, while the new computer used 80W of power whilst idle. When computing the performance benchmark both systems consumed slightly more power, 537W for the cluster and 92W for the new computer. The cluster used considerably more power than that of the new computer, as illustrated in Figure 5.2.

[4] The power tech plus Multifunction energy meter model MS-6115 (Power Tech Plus, 2008) was used to perform the power measurements used in this chapter. All cluster resources were measured once as a combined resource.

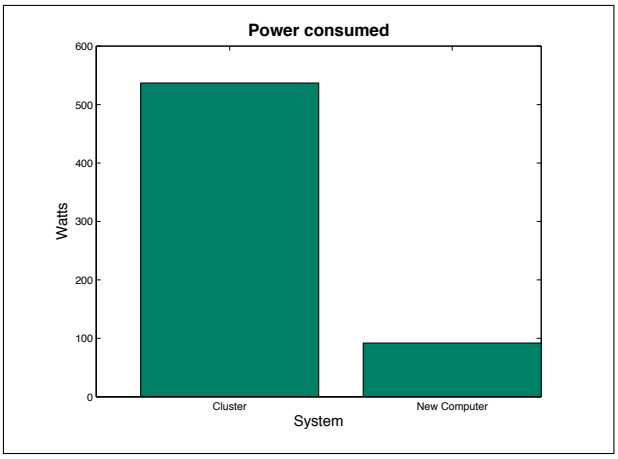

Figure 5.2: Power usage results for the cluster and the new computer while processing a computationally intensive task

The power efficiency (PE) of the cluster and of an individual new computer was measured by dividing the measured power consumption (MPC) by the measured performance (MP): $PE = MPC/MP$. The cluster was measured as consuming 537W of power while under processing load, giving a power efficiency of 537W / 6.32 Gflops = 84.96 W/Gflops. The new computer was measured as consuming 92W of power when under processing load, giving a power efficiency of 92W/2.235 Gflops = 41.16W/Gflops. The power requirements of the cluster need to be factored in when deciding whether or not an e-waste cluster is an acceptable use of resources.

5.8 Discussion

The results suggest that the life of an obsolete computer can be extended by reusing it in a cluster and that these clusters can computationally outperform a standard new computer. This finding is important not because it would be desirable to replace a standalone computer with a cluster for day to day use, but if a cluster were slower than a standalone computer there would be little advantage in using the cluster for many purposes. These results therefore suggest that a cluster of reused computers can offer, for processing purposes, a

performance advantage over what would otherwise be available to the researcher. At the very least, though, this research has shown that in circumstances that call for a cluster of computers, it is feasible to build that cluster from computers that were otherwise destined for the growing stream of e-waste. For teaching uses and some research projects, therefore, a cluster such as this should be more than adequate.

Disused computer systems should not be used indefinitely in a cluster, but a cluster may usefully extend the life of these systems for some purposes. The useful life time of the cluster is dependent on the number of nodes in it, the processing ability of those nodes and the reliability of the hardware.

Although the creation of a cluster from disused computers may negate the need for the creation of a new resource, the cluster will consume substantial quantities of energy which will go some way to offsetting these savings.

The results indicate that these clusters constructed from obsolete computers consume substantial amounts of energy. The cluster produced just under three times the processing ability of the individual new computer, but consumed just under six times the power. It could therefore be said that the cluster is not power or energy efficient.

5.9 Concluding remarks

The results suggest that in some instances the lives of obsolete computers can be prolonged by combining them in a cluster, where for computing-intensive operations they can outperform a standard individual new computer.

This chapter has presented the controlled reuse of disused computers as a method of e-waste management. This method offers the potential to save money and to reduce the flow of e-waste. With the creation of a small cluster, the chapter has shown that it is possible to create a usable high-performance resource out of e-waste. Moreover, it may be possible to create grid resources from such computers.

Distributed computing resources, including a cluster and an institutional grid, were created for the research conducted in subsequent chapters of this thesis from e-waste computers.

Although this chapter has demonstrated that it may be possible to prolong the use of older computers, it is apparent that the prolonged use of these resources is not energy efficient. The focus of the research presented in this thesis is on the conservation of energy in distributed computing environments. The methods of energy conservation that are explored enable the conservation of energy without additional hardware, and can be used on environments such as the aforementioned cluster created from obsolete computers.

5.9.1 Key findings

Key findings from the research presented in this chapter are:

- The lives of obsolete computers can be usefully prolonged by combining them in a cluster, where for computing-intensive operations they can outperform a standard new computer.
- Creating high-performance resources out of older hardware is not energy efficient.

Chapter 6

MODEL IMPLEMENTATION: AN EXAMINATION OF DIFFERENT ECONOMIC RESOURCE ALLOCATION MECHANISMS AND PERFORMANCE METRICS ON A CLUSTER AND GRID

6.1 Introduction

This chapter presents the results of tests conducted on the grid resource allocation mechanism allocating tasks over a real grid and cluster environment. Two simulation models, presented in Chapters 3 and 4, examined the effect that simple resource allocation can have on energy consumption and time taken to execute tasks. This chapter presents the details and results of the physical implementation that was created to test the results of the two simulated experiments. It examines the effect on energy consumption and time taken to execute tasks when either the performance metric or the resource allocation mechanism is altered. This chapter describes the resource allocation mechanism as implemented on the grid, the procedure used to test the mechanism, and the results and analysis of the conducted tests.

The research presented in this chapter was published in The Sixth Workshop on High-Performance, Power-Aware Computing (HPPAC) (Lynar, Simon, Herbert and Chivers, 2010).

The key contributions of the research presented in this chapter are:

- a study of the effectiveness of the different resource allocation strategies on a test cluster;

- a study of the effectiveness of the different resource allocation strategies on an institutional grid;

- an analysis of the energy usage of the different resource allocation strategies within the resource allocation mechanism;

- a comparative examination of a grid resource allocation mechanism on a heterogeneous cluster and in a grid environment.

6.2 Description of the resource allocation mechanism

The model used for resource allocation in this chapter is conceptually the same as the model used in Chapter 4. Some of the performance metrics presented in Chapter 3 are also included in the implemented mechanism and are examined in this chapter.

At the start of a run, a configuration file sets the resource allocator to allocate resources using one of three auctions, a Batch Auction (BA), a Continuous Random Allocation (CRA), or a Pre-Processed Batch Auction (PPBA). These auctions were discussed in Chapter 4. When users submit tasks to the grid for execution they do so through the command line of one of the nodes on the grid, indicating the number of processors required and the executable that is to be submitted to the grid. When it is ready to receive bids, the resource allocation mechanism reads a file of known hosts and sends out a request for bids to all nodes listed in the file. The agents (the computing nodes) respond to this request with bids, nodes with multiple processors sending multiple responses[1]. As with the simulations, the bids are single shot and sealed: that is, the agents bid only once for a given task and do not see each other's bids.

If an agent has reached a predefined CPU utilisation level it returns a bid of zero, indicating that it has no spare capacity with which to execute the task. Otherwise it bids the value

[1] All nodes used had only a single processor.

of its performance metric, which is based upon the known data of its energy consumption and its processing ability.

The resource allocator then deals with the nodes' responses according to the economic resource allocation mechanism that it has been set to use. Once allocation is determined, the task is allocated to a resource and executed.

The auctions used in this research have already been explained in Chapter 4, the implementation of these auctions differs slightly from the aforementioned theory. The difference is relating to the wait time, in the simulation the BA waited a set amount of time for bids to be returned, the implementation of the BA waits as long as it takes for all nodes to respond. The BA as implemented in this chapter is described in Algorithm 9, the CRA auction in Algorithm 10, and the PPBA in Algorithm 11.

Algorithm 9 Implementation of the BA

Ask each node for a bid

Sort the bids in order of value

Assign nodes

Start task execution

Algorithm 10 Implementation of the CRA

Shuffle nodes

Ask the required number of nodes to bid

Assign nodes

Start task execution

6.2.1 Performance metrics examined

In the simulation in Chapter 3 a number of performance metrics were used to rank nodes, leading to the conclusion that altering the metric alters the energy and time taken to

Algorithm 11 Implementation of the PPBA

Sort bids

Assign nodes

Start task execution

Ask each node for a bid

execute tasks. For the resource allocation mechanisms implemented in this chapter, three performance metrics have been included, as listed in Table 6.1. These metrics were selected because in the simulation they led to markedly different node rankings, energy consumption, and processing time.

Table 6.1: Performance metrics used in tests of the resource allocation application

Metric	Formula
Eflops	$\frac{Flops}{Max\ energy}$
Ettt	$\frac{1}{Test\ task\ time \times Max\ energy}$
Power	$\frac{1}{Max\ energy}$

6.2.2 Technical application description

The resource allocation application operates in two modes: as a server daemon, which waits in the background for requests, or as a client task submitter. When the application is executed as a daemon, it waits for an 'ask' or run command from the client application. When it receives an ask it checks its CPU utilisation and responds with either its predetermined bid if the utilisation is below a specified threshold, or a bid of zero if it is currently running to capacity. If the node receives a command to execute an external application, it does so. When running as a client, the resource allocation application notes a user's request, comprising the task to be executed and the number of nodes required. It then interrogates

a file of hosts, sends a request to each host, and stores all of the responses it receives. A high-level description of the application can be found in the activity diagram Figure 6.1.

In this research the application was set up as a daemon on all nodes. From one node the application was also used to submit tasks to the grid. The application was activated through workflow scripts that accurately automate the task submission process; the workflows that were used are described in Section 6.3.2.

6.2.3 Communication model

The resource allocation mechanism requires the nodes to communicate with each other. When a node has a request, consisting of a task executable and the number of processors required, it asks the other nodes for bids, and each node sends its bid to the requesting node. This pull model of communication was chosen because it conceptually fits with the auctions chosen, and in particular with the assumption that there is a delay in receiving new information from nodes once an ask is made to the system. The bids are stored by the requesting node, dealt with according to the resource allocation mechanism in use, and, depending on the chosen resource allocation mechanism, possibly used in subsequent allocations.

6.2.4 Communication protocol

To facilitate communication a protocol was produced. When a node makes a request, the request should indicate that it is a request for a bid, the time that the request was made, and the resource that made the request. The request needs to be properly formatted to distinguish between a genuine request and attempted use of the port by other application-s/individuals[2]. A valid response will include the request time so that old responses may be

[2] The environments used were isolated from the outside world. It was not possible to receive an invalid request in these experiments. However, the communication protocol has been defined with future use in an uncontrolled environment in mind.

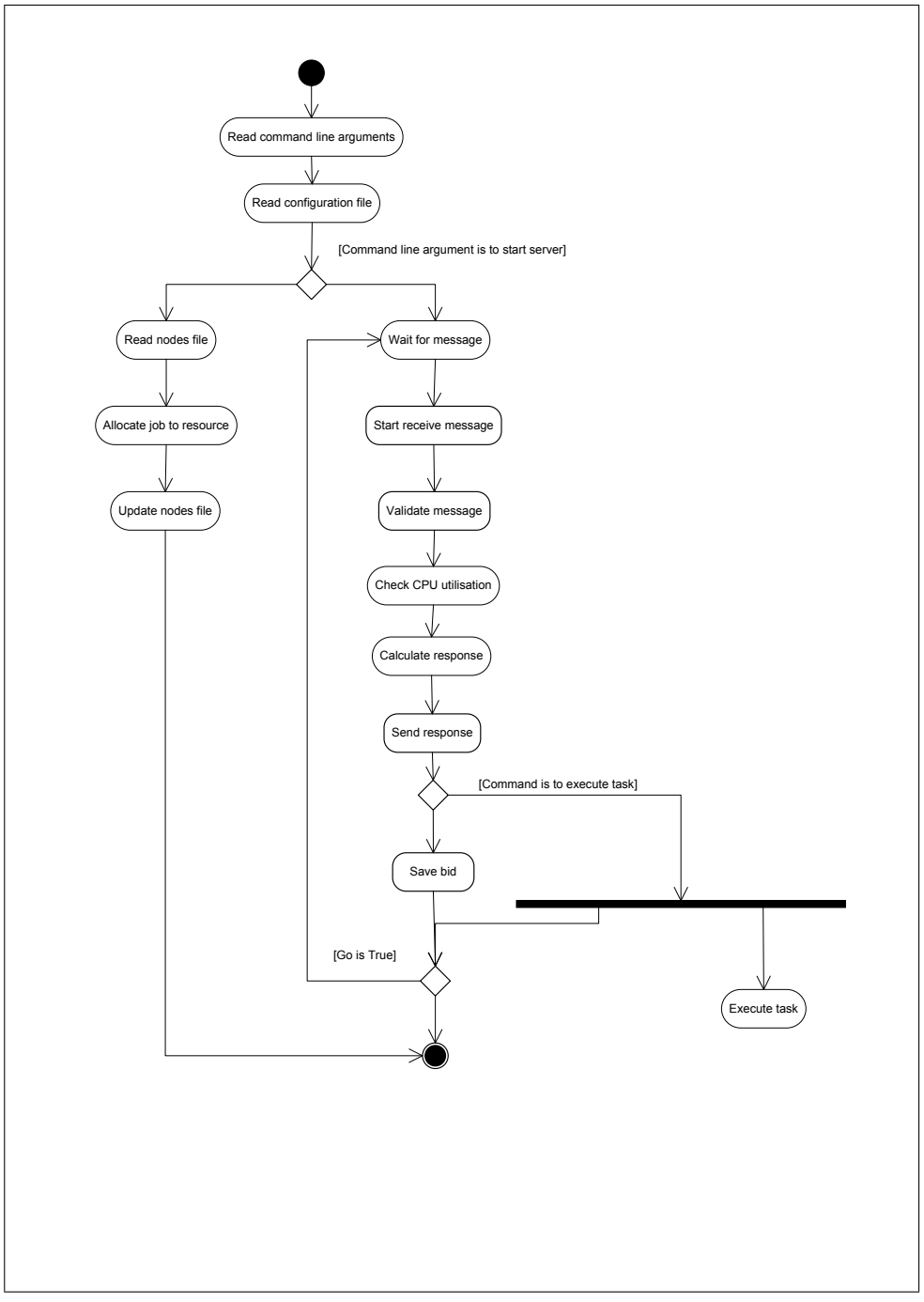

Figure 6.1: UML Activity diagram describing the resource allocation application

discarded and not used in the current round of bidding. Indicating the resource that made the request simplifies returning a response to that resource.

A properly formatted bid request contains the string "BIDREQ", then the time in the form of the number of milliseconds since the start of January 1, 1970 (UNIX time format), then the IP address of the requesting machine. The request is in a single string, without spaces, and with hyphens separating the sections, e.g. `BIDREQ-999697999-123.123.123.123`.

The response indicates that it is a response, the time of the request that the response is for, the bid of the responding node, and the IP address of that node. A properly formatted bid response contains the string "BIDRES", then the time stamp (UNIX time format) on the request that this response is for, followed by the bid and the IP address of the responding node. The response is in a single string, without spaces, and with hyphens separating the sections, e.g. `BIDRES-999697999-12.04-321.321.321.321`.

In the simulation model, it was possible to determine when a node was available simply by asking it. Matters are not so simple in the physical implementation, which uses a GNU/Linux command to determine what level of use the processor is in. The command used is

```
ps -e -o pcpu | awk '{ SUM += $1} END { print SUM }'
```

This command produces a number representing the processor load as a percentage. If the processor is idle then the resource allocation mechanism can assume that the processor is available. For this experiment the processing threshold for idleness has been set to 50%; that is, nodes that have less than 50% processor usage as indicated by the above command will be considered idle. If a node's processor usage is above the predefined threshold, it will return a bid of zero.

6.2.5 Limitations, assumptions, and differences of implementation

The implementation was based on the simulated model, except all nodes remained on during testing, the study of a working Variable-On Variable-Off (VOVO) mechanism has been left for future research. This study was limited to testing how effective the different performance metrics and the different auctions are at reducing grid energy consumption and the time taken to execute tasks.

A number of simplifying assumptions which were made for this implementation that may not apply to a non-research environment. These assumptions include:

- Although all nodes are capable of submitting jobs to the grid, only one node does so.
- All nodes are local; although it would be possible for nodes to be located anywhere in the world, it would not be practical to collect their results in a controlled manner.
- The computing nodes used for these experiments are dedicated to the tasks assigned to them during the experiments.
- It is assumed that no unknown tasks, such as background tasks, are executing on the nodes during testing.
- All of the tasks in each run have homogeneous processing requirements.
- All resources are capable of performing the sort of computation required by any task submitted to the system.
- Each task processes on only one resource, not over multiple resources at once, and only requires that one resource to process.
- There is no foreknowledge of how long it will take to execute an incoming task, or the incoming task's computational requirement.

- There is no facility to migrate processing tasks from one machine to another.

In addition to the simplifying assumptions and limitations there are a number of further differences between the implementation and the simulated models. In the simulated model presented in Chapter 4 the nodes know whether or not they are processing tasks, and nodes process only one task at a time. In the implementation a CPU threshold is used to determine whether or not a node will bid for a task, so it is possible for a node to be processing several tasks at the same time.

In the simulation model each run of the simulation lasts a set amount of time. In effect, the simulation assumes that if tasks from the workflow finish early the nodes will remain idle until the end of the simulation. In this chapter this assumption is not made: energy consumption readings are measured from the start of execution of the first task in the workflow to the end of execution of the last task in the workflow. In effect, the assumption is that after completion of a workflow the nodes that the distributed resource comprises of will be either used by another workflow or powered down.

6.3 Methodology

6.3.1 Hypotheses

The primary research question is: "does altering the resource allocation mechanism or the performance metric affect the allocation of resources in a way that alters the total energy used or the time taken in the execution of tasks?" This question will be asked through the following hypotheses:

H1: Altering the resource allocation mechanism will result in a significant difference in the amount of energy consumed.

H2: Altering the resource allocation mechanism will result in a significant difference in the time tasks take to execute.

H3: Altering the performance metric will result in a significant difference in the amount of energy consumed.

H4: Altering the performance metric will result in a significant difference in the time tasks take to execute.

The null hypothesis is therefore:

H0: Altering the performance metric or the resource allocation mechanism will not affect the allocation of resources in a way that alters the time to execute tasks or the total energy used in the execution of tasks.

A number of workflows were constructed consisting of homogeneous real computing tasks of known size. These workflows were executed first on a small pilot cluster (Table 6.4) and then on an institutional grid (Table 6.5).

6.3.2 Workflows

The workflows described in Table 6.2 were chosen to test the physical mechanism because they provide continuity with the simulation. In these workflows, the tasks are evenly distributed over a ten-minute period of time. To facilitate the variation in computational size, a task was created that has a dynamically changeable computational requirement.

Table 6.2: Basic computational workflows with homogeneous tasks – n denotes computational size

Workflow	Computation size	Number of tasks
1	Small (n=10)	100
2	Medium (n=100)	100
3	Large (n=200)	50

6.3.3 Calibration of tasks

In the simulation examined in the prior chapters it was possible to specify the exact number of time steps required to complete a task of given computational size on a specified node. Perfect calibration to that simulation is not possible with this experiment because the simulation was based on the results of benchmark tests. Benchmarks are not a perfect representation of a computer's real processing performance, and the computer cannot be relied upon to always take the same time to execute the same test task.

Three tasks were required for the workflows: a very small task that can be computed by the slowest resource in less than a second; a small task that can be computed by the fastest resource in less than a second; and a larger task that can be computed by the fastest resource in a number of seconds.

A prime number search script (Appendix A.3) was created for the primary task. This script was written to be more processor intensive than an ordinary prime number search script through the addition of redundant loops. The script is readily scalable from a very small task to a large task simply by increasing the search range. For instance, the script takes less than a second on most modern computers to find all the prime numbers between 0 and 100, but can take several minutes to find all the prime numbers between 0 and 10000. It should be noted that this task is executed on a single machine at a time.

6.3.4 Tests

Both the effects of modifying the performance metric and the effects of modifying the resource allocation mechanism were tested. The tests were executed using each of the three resource allocation mechanisms, and using three performance metrics. A total of eleven basic workflow tests were conducted, as described in Table 6.3.

Table 6.3: Tests conducted

Test	Workflow	Performance metric	Resource allocator
1	1 (small)	Eflops	BA
2	1 (small)	Eflops	CRA
3	1 (small)	Eflops	PPBA
4	2 (medium)	Eflops	BA
5	2 (medium)	Eflops	CRA
6	2 (medium)	Eflops	PPBA
7	3 (large)	Eflops	BA
8	3 (large)	Eflops	CRA
9	3 (large)	Eflops	PPBA
10	2 (medium)	Ettt	BA
11	2 (medium)	Power	BA

6.3.5 Testing equipment

Three primary pieces of testing equipment were employed in the experiment: an energy meter, the GNU/Linux `time` command[3], and the resource allocation application. To measure energy usage a Hioki 3197 power quality analyser was used. This data-logging energy meter has guaranteed accuracy[4] after a 30 minute warmup, provided that environmental factors of humidity and temperature are met, along with frequency and power specifications. It was used to measure the energy consumption while processing the tasks and while idle. Figure 6.2 shows the setup of the energy meter for the cluster environment. Figure 6.3 shows the setup of the energy meter for the grid environment.

[3] The source code for this executable can be obtained from `http://packages.debian.org/source/lenny/time`

[4] Guaranteed accuracy of Active power $\pm 0.3\%$ rdg $\pm 0.2\%$ f.s. (power factor = 1) Reactive power $\pm 1dgt$ Energy $\pm 1dgt$ applied to active and reactive power measurement accuracy. + clamp accuracy (10A) of $\pm 0.3\%$ rdg $\pm 0.02\%$ f.s. Measurement method 200 ms calculation. Real-time clock accuracy of $\pm 5ppm$ (within 13 s/mo @$25°C$). Guaranteed accuracy @ $23 \pm 5°C$, 80% RH or less (HIOKI E. E. Corporation, 2006). Measurements for energy were taken in watt hours. All measurements were taken in the same air-conditioned environment. See appendix A.2 for error calculations.

Node measures for power usage presented in this chapter were also performed using the Hioki 3197 power quality analyser. The values presented are mean values for nodes with identical hardware specifications. Error associated with the measurement is examined in Appendix A.2. The values obtained were input into the resource allocation mechanism and used to rank nodes[5].

The GNU/Linux `time` command was used to measure the time taken to execute each task. This command was initiated by the resource allocator that submitted the task for allocation, but was initiated when the task began execution, not at the time of task submission.

Figure 6.2: Setup of the cluster experiment, showing the energy meter (lower left) and some of the nodes used (stacked on the far right) in the tests

[5] The resource allocator treats the input values for node attributes as relative. Those value are only used to create ranks.

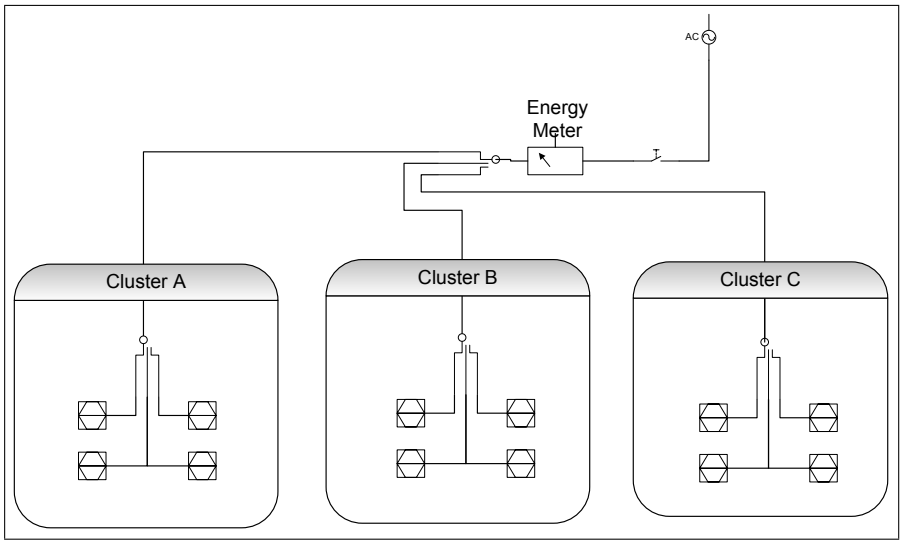

Figure 6.3: Setup of the grid experiment, showing the location of the energy meter relative to the computing resources

6.3.6 Testing procedure – cluster experiment

Testing the energy consumption of a multi-site grid requires substantial equipment. To ensure that the testing procedure was adequate, and that all testing tools and the resource allocation mechanism functioned as expected, a cluster was set up and tested. The testing procedure used is described in Algorithm 12.

Each test was conducted ten times, and the results recorded. The times taken to execute each task were recorded to a text file for each run. Energy readings were digitally logged while the tests were running.

6.3.7 Environments

The cluster tests were conducted on a cluster made up of a cross-section of the available nodes. The six nodes used in the cluster environment are described by their attributes for power usage and performance, as used by the resource allocation mechanism, in Table 6.4.

Algorithm 12 Data collection procedure

1: Connect computers
2: Boot computers off live distribution
3: Set configuration files
4: Initiate resource allocation mechanism server on all nodes
5: Ensure that the host file of known nodes is populated
6: Sync the clock on all nodes with that on the energy meter
7: Set energy meter to record energy usage
8: Ensure room temperature and humidity are within recording instrument's range of accuracy
9: Begin logging energy consumption
10: Begin execution of workflow
11: Record results
12: Save and backup results

The network setup of the nodes is described in Figure 6.4. Additional technical details on all nodes used in this research can be found in Table F.1 in Appendix F on page 221.

Table 6.4: Nodes used in the cluster experiment showing power drawn when idle W_{min}, power drawn when processing at capacity W_{max}, and millions of floating point operations per second $Mflops$

Node	W_{min}	W_{max}	$Mflops$
001	50	74	236
002	50	74	235
CN1	52	84	286
CN2	52	84	286
HPN1	50	94	318
HPN2	50	94	317

After completion of the cluster tests, the tests were scaled up to a full grid environment. The grid environment was comprised of three clusters, each comprised of homogeneous nodes that were different from those in the other clusters. In total there were 26 nodes in the grid's

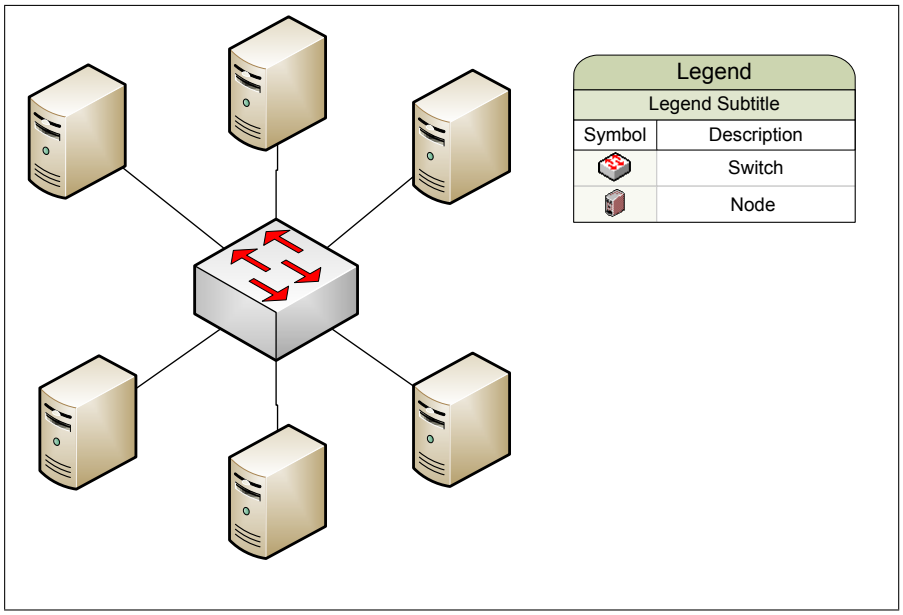

Figure 6.4: Network diagram of the cluster environment

three clusters[6]: eleven in cluster A, eleven in cluster B, and four in cluster C. The nodes are described in Table 6.5[7], and the network set-up of the grid is described in Figure 6.5. The nodes within each cluster were connected to one another through 100Mbit/s switches, and the clusters were connected to one another through the university's network. The grid was an institutional grid: during the testing period, the grid was completely isolated from the rest of the university's network, and used exclusively for this research.

[6] Cluster A consists of nodes 001 – 011, cluster B consists of nodes HPN1 – HPN11 and cluster C consists of nodes CN1 – CN4

[7] Values displayed are those used for input into the resource allocation mechanism.

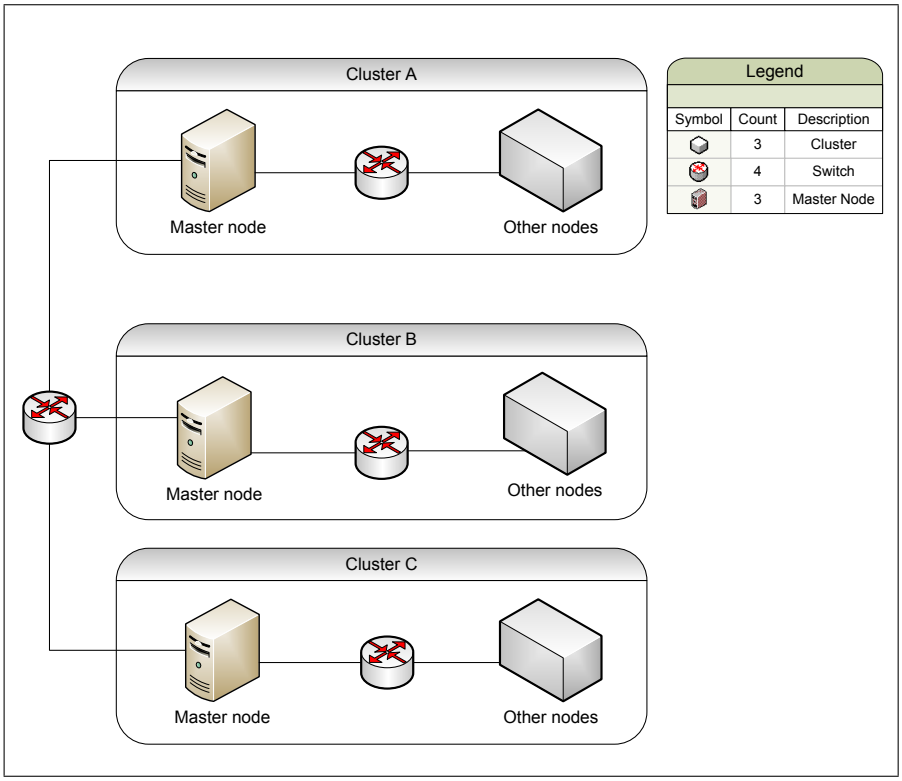

Figure 6.5: Network diagram of the grid environment

Table 6.5: Nodes of each cluster used in the grid showing power drawn when idle W_{min}, power drawn when processing at capacity W_{max}, and millions of floating point operations per second $Mflops$

Cluster	*Nodes*	W_{min}	W_{max}	$Mflops$
A	11	50	74	235
B	11	52	84	317
C	4	50	94	286

6.4 Results

In the cluster environment, ten samples were taken for each of the standard tests defined in Table 6.3. Table 6.6 presents the median time taken to execute and the median energy consumed in these tests[8].

Table 6.6: Summary of cluster results – median values for ten runs of each test

Test	Workflow	Performance metric	Mechanism	Median time taken (s)	Median energy usage (kJ)
1	1 (small)	Eflops	BA	624	190
2	1 (small)	Eflops	CRA	611	186
3	1 (small)	Eflops	PPBA	626	191
4	2 (medium)	Eflops	BA	831	364
5	2 (medium)	Eflops	CRA	1105	441
6	2 (medium)	Eflops	PPBA	853	376
7	3 (large)	Eflops	BA	1037	435
8	3 (large)	Eflops	CRA	1182	485
9	3 (large)	Eflops	PPBA	1007	428
10	2 (medium)	Ettt	BA	852	379
11	2 (medium)	Power	BA	867	386

For the grid environment, ten samples were taken for each of the first nine tests described in Table 6.2. Table 6.7 presents the median time taken to execute and the median energy consumed in these tests.

The collected data has been cleansed by excluding erroneous results. No cleansing was required for the results obtained from the cluster environment, however some errors occurred during processing of the workflows using the CRA mechanism on the grid environment, and one transcription error was detected. The second run with the BA mechanism was recorded as taking 361 seconds, this is too low with the hardware available and the value was most likely the result of a transcription error. When using the CRA mechanisms to allocate

[8] Raw results for all tests performed in this chapter can be found in Appendix D Table D.1 on page 209.

Table 6.7: Summary of grid results – median values for ten runs of each test; tests 10 and 11 were not conducted on the grid

Test	Workflow	Performance metric	Mechanism	Median time taken (s)	Median energy usage (kJ)
1	1 (small)	Eflops	BA	601	734
2	1 (small)	Eflops	CRA	1442	1692
3	1 (small)	Eflops	PPBA	602	720
4	2 (medium)	Eflops	BA	661	900
5	2 (medium)	Eflops	CRA	1183	1588
6	2 (medium)	Eflops	PPBA	730	979
7	3 (large)	Eflops	BA	974	1300
8	3 (large)	Eflops	CRA	1739	2383
9	3 (large)	Eflops	PPBA	1040	1346

tasks occasionally the grid would appear to not complete processing for prolonged periods of time, despite the appearance of processor inactivity. It was later determined that there was a hardware fault with one of the least often used nodes. A total of four runs were effected by this error. Subsequently the eighth run of the CRA mechanism with workflow 1, and the third, fifth and sixth runs of the CRA mechanism with workflow 3 have been excluded.

The exclusion of the aforementioned ten results is conservative, all data excluded for being too high are from the CRA mechanism whereas the only data excluded for being too low is from the BA mechanism.

6.5 Discussion

The data collected were not normally distributed, as such an ANOVA was not appropriate so non-parametric tests were used. To analyse the results the non-parametric Kruskal-Wallis test was used to determine if there were significant differences in the time taken or energy used between the groups of results. To determine which groups are significantly different

the Wilcoxon Each Pair test was used. All tests were conducted at the 0.05 confidence interval. The raw statistical output is presented in Appendix B. Analysis was conducted on both the time taken to compute a workflow and the energy used to compute a workflow. Although an increase in the time taken to execute a workflow will typically result in an increase in energy usage, an increase in energy usage could occur without an increase in execution time if more use was being made of nodes with higher energy consumption. The analysis of the results obtained from the pilot cluster environment are first discussed, this is followed by a discussion of the results obtained from the grid environment.

6.5.1 Discussion – cluster results

The first hypothesis, H1, states that "Altering the resource allocation mechanism will result in a significant difference in the amount of energy consumed."

The data collected for energy used were not normally distributed according to the Shapiro-Wilk W test ($P < 0.0001$ $W=0.839$).

In workflow 1 when testing energy consumption there is a significant difference between the results when using different mechanisms. The Kruskal-Wallis chi-square approximation was used and produced a p-value of less than 0.0001 (DF=2 ChiSquare=22.0622). There was a significant difference between all pairs of mechanisms. The median energy consumption for the BA mechanism is 190kJ, for the CRA it is 186kJ and for the PPBA it is 191kJ. The Wilcoxon Each Pair test shows that there is a significant difference between energy consumed by the CRA and PPBA (P=0.0001 Z=3.81423), the CRA and BA (P=0.0001 Z=-3.84134), and also the BA and the PPBA (P=0.0465 Z=1.99038) mechanisms.

In workflow 2 the median energy used for the BA mechanism is 364kJ, for the CRA it is 441kJ, and for the PPBA it is 376kJ. The Kruskal-Wallis chi-square approximation was used and produced a p-value of < 0.0001 (DF=2 ChiSquare=22.7449). The Wilcoxon Each Pair test shows that there is a significant difference between all pairs of mechanisms (CRA-

BA P=0.0002 Z=3.75032, PPBA-BA P=0.0071 Z=2.69064, and PPBA-CRA P=0.0002 Z=-3.74466).

In workflow 3 there is also a significant difference in the energy used by the mechanisms (P<0.0001 DF=2 ChiSquare=18.7619). The median energy used for the BA mechanism is 435kJ, for the CRA it is 485kJ, and for the PPBA it is 428kJ. The Wilcoxon Each Pair test shows a significant energy difference between the CRA and BA mechanisms (P=0.0002 Z=3.74326), and also between the CRA and PPBA mechanisms (P=0.0003 Z=-3.59201), but not between the BA and PPBA mechanisms.

The significant differences in total energy consumption between mechanisms support the first hypothesis, H1.

The second hypothesis, H2, states that "Altering the resource allocation mechanism will result in a significant difference in the time tasks take to execute."

The data collected for time taken were not normally distributed according to the Shapiro-Wilk W test (P<0.0001 W=0.902).

In workflow 1 there is a significant difference between the times taken to execute workflow 1 when using different resource allocation mechanisms (P<0.0001 DF=2 ChiSquare=24.9304). The median time taken to execute workflow 1 is 624 seconds when using the BA mechanism, 611 seconds when using the CRA, and 626 seconds when using the PPBA. The Wilcoxon Each Pair test shows that there is a significant difference between all pairs of mechanisms (PPBA - CRA P=0.0002 Z=3.77319, PPBA - BA P=0.0006 Z=3.45134, and CRA - BA P=0.0002 Z=-3.76743).

Workflow 2 also shows a significant difference between the median workflow execution times when using different resource allocation mechanisms (P<0.0001 DF=2 ChiSquare=20.9271). The median time taken to execute all tasks in workflow 2 is 831 seconds when using the BA, 1105 seconds when using the CRA, and 853 seconds when using the PPBA. The Wilcoxon Each Pair test shows a significant difference between CRA and BA (P=0.0002 Z=3.74749),

and between CRA and PPBA (P=0.0002 Z=-3.74326), but not between the BA and PPBA mechanisms.

In workflow 3, as with the other two workflows, there was a significant difference in the time taken to execute workflow 3 when using different resource allocation mechanisms (P=0.0002 DF=2 ChiSquare=16.9954). The median time taken to execute workflow 3 was 1038 seconds with the BA, 1182 seconds with the CRA, and 1008 seconds with the PPBA. The Wilcoxon Each Pair test shows a significant difference between CRA and BA (P=0.0002 Z=3.66626), and between CRA and PPBA (P=0.0009 Z=-3.32734), but not between BA and PPBA.

The significant differences in total time taken to complete workflows between mechanisms support the second hypothesis, H2.

Table 6.8: Cluster results: mean values for ten runs of each test, shown as the percentage difference for each mechanism compared to the BA mechanism (tests 1–9) and the percentage difference for each metric compared to the Eflops metric (tests 10–11)

Test	Workflow	Metric	Mechanism	Time %	Energy %
1	1 (small)	Eflops	BA	0.0	0.0
2	1 (small)	Eflops	CRA	-2.1	-2.2
3	1 (small)	Eflops	PPBA	0.2	0.2
4	2 (medium)	Eflops	BA	0.0	0.0
5	2 (medium)	Eflops	CRA	34.2	21.8
6	2 (medium)	Eflops	PPBA	2.9	3.2
7	3 (large)	Eflops	BA	0.0	0.0
8	3 (large)	Eflops	CRA	22.2	18.4
9	3 (large)	Eflops	PPBA	1.3	1.9
10	2 (medium)	Ettt	BA	4.1	2.9
11	2 (medium)	Power	BA	6.0	4.3

The third hypothesis, H3, states that "Altering the performance metric will result in a significant difference in the amount of energy consumed."

Tests 4, 10, and 11, which all execute workflow 2 with the BA mechanism, examine the effects on execution time and energy consumption when the performance metric is altered.

The median energy consumed was 364kJ when using the Eflops performance metric to rank nodes, 379kJ when using Ettt, and 386kJ when ranking nodes according to their Power. Prior to conducting an ANOVA the distribution of the energy consumed in each run was analysed, and a Shapiro-Wilk W test was used to analyse the goodness of fit to a normal distribution. The p-value was < 0.0001, indicating that the data did not fit the normal distribution, so a Kruskal-Wallis test was applied to determine whether there is a significant difference in the energy used when the performance metric is altered. The Kruskal-Wallis ChiSquare approximation p-value was < 0.0001 (DF=2 ChiSquare=25.6136), indicating that the groups are significantly different. The Wilcoxon Each Pair test was then applied, and showed significant differences between all pairs of groups (Ettt-Eflops P=0.0002 Z=3.754573, Power-Eflops P=0.0002 Z=3.750317, and Power-Ettt P=0.0002 Z=3.675942). This suggests that altering the performance metric used to rank nodes can significantly alter the energy used. The third hypothesis was not rejected.

The fourth hypothesis, H4, states that "Altering the performance metric will result in a significant difference in the time tasks take to execute."

The median time taken to execute workflow 2 was 831 seconds when using Eflops as the performance metric, 852 seconds when using Ettt, and 867 seconds when using Power. The Shapiro-Wilk W test indicated that the data did not fit the normal distribution, so a Kruskal-Wallis test was applied to determine whether there is a significant difference in the execution time when the performance metric is altered. The Kruskal-Wallis ChiSquare approximation indicated that the groups are significantly different (P<0.0001 DF=2 ChiSquare=22.7297). The Wilcoxon Each Pair test was then used and showed that all groups are significantly different from one another (Ettt-Eflops P=0.0002 Z=3.746076 , Power-Eflops P=0.0002 Z=3.747488 , Power-Ettt P= 0.0002 Z=2.686579). This suggests that altering the performance metric can significantly alter the time taken to execute tasks, and thus supports hypothesis H4.

The results of the cluster tests show many interesting phenomena. The results differ significantly between mechanisms in workflows 1, 2 and 3 (tests 1–9), suggesting that altering the

resource allocation mechanism can make a significant difference to energy consumption and time taken to execute the workflow. The results do not differ significantly between all pairs of mechanisms for all workflows. When the tasks are trivial in computational size, such as those in test 1, different allocations of these tasks have little impact on energy consumption; however, the time spent actually allocating those tasks can have an impact on energy consumption and time taken to execute tasks. The performance of the CRA in workflow 1 suggests this to be true, as it essentially allocates resources randomly to nodes, yet it executed the tasks faster and with less energy consumption than the other mechanisms. Workflow 1 contains many tasks, equally spaced, requiring trivial amounts of computation. Compared to both the BA and the PPBA, the CRA shows more than 2% improvement in execution time and more than 2% improvement in energy consumption.

Workflow 2 contains the same number of tasks, but the tasks require substantially more computation. In the results of tests 4–6 it can be seen there is a significant difference in the energy consumption and time taken according to the choice of resource allocation mechanism. In this case, however, the CRA auction performed far worse than the other two mechanisms, taking on average 34% longer to execute and consuming 22% more energy than the BA. The PPBA performed better than the CRA but somewhat worse than the BA, consuming 3.2% more energy.

The third workflow contained fewer tasks than workflow 2, with each requiring more processing. When executing this workflow the differences in energy consumption and time taken to execute tasks were still prominent, although less prominent than those of workflow 2. Compared to the BA, the CRA auction took on average 22% longer to execute and consumed 18% more energy. The PPBA performed better than the CRA but still worse than the BA, taking 1.3% longer to execute and consuming 1.9% more energy. In workflow 3 the tasks were larger, but this did not result in larger differences between the results of the resource allocation mechanisms. It is possible that this is a result of reaching complete saturation at least once, with all the nodes processing at full capacity. When this occurs, a backlog of jobs forms, and each job is then distributed to the node that next becomes available. This negates the effect of resource allocation, as all nodes are actively processing.

Thus, although the third workflow has larger tasks, resource allocation is less effective at reducing energy consumption.

The BA has the advantage of being able to guarantee that it is allocating tasks to the most desirable available resources, however the disadvantage is that it takes time to ask for bids and receive a bid from each node. The results show that the time cost of having to wait for all nodes to respond prior to allocating tasks was outweighed by the benefit of allocating tasks to the most desirable resources. It should be noted that these tests were conducted on a cluster, and the response time between nodes was negligible. On a global grid the response times could be considerable, and may produce substantially different outcomes for the BA.

The CRA mechanism randomly allocates tasks to resources, and it does so quickly. This allocation results in shorter execution times and reduced energy usage when the tasks are of trivial size, but the CRA performs less favourably with bigger tasks. As the size of the tasks increases the opportunity cost of allocating those tasks to less desirable resources becomes substantial, and the opportunity to save time and energy through speedy allocation diminishes. These results indicate that the CRA produces the most energy efficient resource allocations when tasks are of a trivial size, but produces the least energy efficient allocations when tasks require substantial computation.

The PPBA performs less favourably than the BA, consistently resulting in higher energy costs and longer run times. The allocation decisions of the PPBA are based on historical data, which these results suggest can lead to substantially inefficient allocations. Allocations that were once efficient will not always remain so, because nodes may become unavailable or overloaded. There is a lag between the time that a task is assigned to a node and the time that the node's CPU usage increases. When the CPU usage exceeds a predefined limit the node responds to asks with a zero bid. In this scenario the BA would not assign another task to the node, but the PPBA might, as it would be using historical data.

The cluster results suggest that altering the resource allocation mechanism can make a substantial difference to the energy consumption and the time taken to execute a stream of

tasks, with the observed differences being most pronounced when there are large tasks and spare processing capacity.

In addition to the altering of resource allocation mechanisms, this research shows that altering the performance metric used to rank nodes for resource allocation can have a significant effect on both the time taken to execute tasks and the energy consumed during their execution.

6.5.2 Discussion – grid results

After completing the cluster study the experiment was scaled up to a grid environment. The grid consisted of three clusters, each made of homogeneous nodes. The workflows that were used in the cluster experiment were also used in the scaled-up grid experiment.

Analysis of the results will be in relation to the first two of the proposed hypotheses. Hypotheses H3 and H4 were not examined on the grid environment, only on the cluster environment, as tests 10 and 11 were not conducted in this phase of the experiment.

The first hypothesis, H1, states that "Altering the resource allocation mechanism will result in a significant difference in the amount of energy consumed."

The data collected for energy used were not normally distributed according to the Shapiro-Wilk W test ($P < 0.0001$ $W=0.754$).

In workflow 1 the median energy consumption was 734kJ when using the BA mechanism, 1692kJ when using the CRA, and 720kJ when using the PPBA. The non-parametric Kruskal-Wallis ChiSquare approximation test returned a p-value of 0.0002 (DF=2 ChiSquare =16.7279), indicating that there is a significant difference between groups. The Wilcoxon Each Pair test was then used to determine which groups were significantly different from one another, and it found no significant difference between the CRA and BA mechanisms. There were significant differences between the CRA and PPBA mechanisms (P=0.0012 Z=-3.23115), and between the PPBA and BA mechanisms (P=0.0002 Z=-3.72093).

In workflow 2 the median energy consumption was 900kJ when using the BA mechanism, 1588kJ when using the CRA, and 979kJ when using the PPBA. The Kruskal-Wallis ChiSquare approximation test was used, and indicated that there was a significant difference (P=0.0404 DF=2 ChiSquare=6.4155) between the energy used when using different resource allocation mechanisms. The Wilcoxon Each Pair test was then used to examine which groups were significantly different from each other; the only groups that were significantly different from one another were PPBA and BA (P=0.0163 Z=2.40204).

In workflow 3 the median energy consumption was 1300kJ when using the BA mechanism, 2383kJ when using the CRA, and 1346kJ when using the PPBA. There is no significant difference (P=0.1602 DF=2 ChiSquare=3.6632) between the results of the mechanisms.

Many workflows showed a significant difference to energy consumption based on the resource allocation mechanism used. Differences were found between the BA and PPBA, and the CRA and PPBA mechanisms in the full grid environment.

The second hypothesis, H2, states that "Altering the resource allocation mechanism will result in a significant difference in the time tasks take to execute."

The data collected for time taken were not normally distributed according to the Shapiro-Wilk W test (P< 0.0001 W=0.720).

For workflows 1, 2, and 3 the ChiSquare approximation test returned p-values of 0.7237, 0.509 and 0.1902 respectively, indicating that there is no significant difference between the groups.

There was no significant difference detected in the time taken to execute a workflow based on the resource allocation mechanism chosen.

The null hypothesis states that "Altering the performance metric or the resource allocation mechanism will not affect the allocation of resources in a way that alters the time to execute tasks or the total energy used in the execution of tasks." The results obtained through examining the grid and cluster environments has revealed that altering the performance

metric used to rank nodes or altering the resource allocation mechanism can significantly alter both the time taken and the energy used in the execution of tasks. The null hypothesis, H0, is therefore rejected.

Table 6.9: Grid results: mean values for ten runs of tests 1–9, shown as the percentage difference for each mechanism compared to the BA mechanism; tests 10 and 11 were not conducted on the grid

Test	Workflow	Mechanism	Time %	Energy %
1	1 (small)	BA	0.0	0.0
2	1 (small)	CRA	51.2	39.4
3	1 (small)	PPBA	4.2	2.5
4	2 (medium)	BA	0.0	0.0
5	2 (medium)	CRA	17.9	19.6
6	2 (medium)	PPBA	6.2	4.6
7	3 (large)	BA	0.0	0.0
8	3 (large)	CRA	61.1	59.7
9	3 (large)	PPBA	-13.1	-13.6

6.6 Comparison of cluster and grid results

The analysis of the grid results shows only small differences compared to the observed phenomena on the cluster. In the analysis of both environments the CRA mechanism generally results in higher energy consumption and longer execution times than either the BA or the PPBA mechanisms. In workflow 1 on the cluster, the CRA consumes significantly less energy and takes significantly less time to execute than either the PPBA or the BA mechanisms, but this phenomenon is not observed when the same workflow is executed on the grid environment.

In the cluster environment the PPBA and BA often produce significantly different results, although the greatest differences are only 3.2% for energy consumption and 2.9% for execution time (see Table 6.8), achieved in workflow 2. In the grid tests the mean differences

are larger (see Table 6.9), up to 13.1% in time taken and 13.6% in energy consumption for workflow 3, but these differences are not statistically significant.

6.7 Analysis of the relationship between execution time and energy usage

The results presented show a strong correlation[9] between the energy used and the time taken to execute workflows. In general, workflows that take a longer time to execute would also be expected to consume more energy when executing, but the analysis in this section examines the relationship between energy and time in more detail.

Table 6.10 shows the mean power that was being drawn during processing of the workflows in the cluster environment[10], and Table 6.11 shows the mean power that was being drawn during processing of the workflows in the grid environment. The tables show the mean, variance, minimum and maximum power drawn while executing each workflow with each resource allocation mechanism. The tables also indicate the results of a Tukey-Kramer HSD test, using the same letter to group sets of data that are not significantly different from one another.

One interesting phenomenon is that the power drawn is typically lower when using the CRA mechanism, particularly for the cluster results. In workflow 3 the CRA mechanism draws 412W whereas the BA draws 426W and the PPBA draws 428W. While the power draw may be lower when using the CRA mechanism, the time taken to execute the workflows is longer, resulting in higher total energy usage. Power draw has other implications, including heat generation. The effect of increased heat from using more energy-efficient nodes has been left for future research.

There were other differences observed between the two environments on energy usage and time taken, and these differences can be easily explained. The mean power draw is higher

[9] A spearman value of 0.9875 with a p-value of < 0.0001 for the cluster and a spearman value of 0.9224 and a p-value of < 0.0001 for the grid. Full statistical test output is presented in Appendix B.

[10] The values presented for power were derived from the measurement of time and energy.

on the grid in all tests; this is because the nodes in the grid far outnumber the nodes in the cluster. The idle energy consumption of grid nodes results in a consistently higher energy consumption on the grid than on the cluster when processing the same workflow. This result would support the suggestion that idle resources should be turned off to conserve energy.

The mean power draw *per node* is higher on the cluster in all tests, because the utilisation of the nodes in the cluster is invariably greater than the utilisation of the nodes in the grid environment.

A further examination of Table 6.6 and Table 6.7 shows that the mean time taken to execute a workflow is less on the grid than on the cluster except for tests 2 and 8. This shows that in general the tasks are being distributed on the grid to nodes that are able to process them more quickly than on the cluster. The exceptions, tests 2 and 8, use the CRA auction. There is a higher proportion of nodes with the oldest processor type in the grid (Cluster A in Table 6.5) than in the cluster (nodes 001 and 002 in Table 6.4), so the random allocation is more likely to select one of these nodes in the grid than it is in the cluster. Selecting one of these nodes for task execution slows down the execution of the workflow and subsequently results in higher energy consumption.

6.8 Concluding remarks

The research in this chapter examines two implementations of the resource allocation mechanisms and performance metrics that were examined through simulation in Chapter 3 and Chapter 4. The mechanisms were first implemented on a cluster and then on a grid. The results observed in the cluster environment differ from those observed in the grid environment. The increase in variance experienced in the results of the grid tests was expected, as there was a large increase in the number of nodes in the system. Overall the results from the tests performed on the grid closely match those from the tests performed on the cluster. Variance in the system can be explained by a number of factors, including minor

Table 6.10: Power in watts drawn by the cluster environment

Test	WF	Mech	Mean	Variance	Min	Max	Mean *per* node	Tukey group 0.05	Tukey group 0.10	Tukey group 0.20
1	1	BA	304	0.3	303	305	51	A	A	A
2	1	CRA	304	3.5	300	308	51	A	A	A
3	1	PPBA	304	0.5	303	305	51	A	A	A
4	2	BA	438	3.0	435	442	73	A	A	A
5	2	CRA	398	9.0	392	400	66	B	B	B
6	2	PPBA	440	32	433	449	73	A	A	A
7	3	BA	426	188	405	441	71	A	A	A
8	3	CRA	412	74	400	426	69	B	B	B
9	3	PPBA	428	129	406	444	71	A	A	A

Table 6.11: Power in watts drawn by the grid

Test	WF	Mech	Mean	Variance	Min	Max	Mean *per* node	Tukey group 0.05	Tukey group 0.10	Tukey group 0.20
1	1	BA	1213	29	1204	1222	47	A	A	A
2	1	CRA	1164	17453	790	1220	45	A	A	A
3	1	PPBA	1193	13	1186	1198	46	A	A	A
4	2	BA	1343	9277	1093	1481	52	A	A	A
5	2	CRA	1364	403	1335	1393	52	A	A	A
6	2	PPBA	1323	235	1292	1341	51	A	A	A
7	3	BA	1342	1246	1245	1370	52	A	A	A
8	3	CRA	1347	5592	1256	1490	52	A	A	A
9	3	PPBA	1339	1354	1291	1388	51	A	A	A

heterogeneity within the notionally homogeneous nodes of individual clusters, unpredicted network congestion caused by currently executing jobs, and the execution of operating system functions.

It is important to note that the results from studies of grid resource allocation mechanisms

are dependent on the structure and composition of the workflows and on the underlying resources. Different resource allocation mechanisms can only result in a difference to energy consumption and execution time when one mechanism allocates to a more energy-efficient node than another. The opportunity to do this is reduced if the workflow remains the same while the number of homogeneous nodes increases.

The cluster environment was not a grid environment; however, leaving aside the question of network latency issues, it can to some extent simulate a grid for the purpose of understanding the effects of different grid resource allocation mechanisms.

The increase in the number of nodes did result in one notable difference between the two environments. The CRA mechanism performed worse on workflows 1 in the grid environment than in the cluster environment. The slowest nodes were found in cluster A, which with eleven nodes was the equal largest cluster in the grid. Throughout the grid tests, nodes in this cluster were allocated tasks only when the CRA mechanism was in use. The CRA mechanism allocated randomly; given that a large proportion of the available nodes were in cluster A some of these nodes were selected when the CRA mechanism was in use. The extensive use of these nodes could have accounted for the significantly worse outcome for workflows 1 by negating the savings brought about by the lower overhead of the mechanism. The PPBA and BA also produced significantly different outcomes, but only under particular conditions.

The results indicate that economic resource allocation mechanisms that incorporate the power and performance data from nodes offer the ability to conserve significant amounts of energy whilst maintaining the speed of execution.

The results suggest that resource allocation can be used to conserve energy in distributed computing environments without any new hardware. Resource allocation can be used as an energy conservation mechanism where there is spare processing capacity and heterogeneous hardware.

The results obtained in these experiments were from a controlled environment with synthetic workflows. The results observed show that the use of the BA and PPBA mechanisms often

have the ability to conserve significant quantities of energy. However, the percentage of energy that can be conserved differs depending on the number of nodes and the workflows.

To extend the research on these economic resource allocation mechanisms a calibrated simulation is needed. This simulation is required to examine the effect of these resource allocation mechanisms when executing a real workflow. This is examined in Chapter 8.

The results ultimately show that different auctions can result in significantly different resource allocations, which perform more favourably in different scenarios.

6.8.1 Key findings

Key findings from the research presented in this chapter are:

- The use of different auctions to allocate resources can result in different resource allocations, which can perform more or less favourably in different scenarios.
- Incorporating information on the energy efficiency of different nodes into resource allocation decisions in distributed computing environments can result in substantial savings to energy.
- For the purpose of understanding the effects of different grid resource allocation mechanisms, a cluster environment can be used to simulate a grid with the exception of network latency and bandwidth issues.

Chapter 7

THE CALIBRATION OF A SIMULATION TO THE OBSERVATIONS FROM A CLUSTER AND GRID ENVIRONMENT

7.1 Introduction

A simulation of a resource allocation mechanism was described and examined in Chapter 4. The resource allocation mechanism was implemented on a physical cluster and an institutional grid in Chapter 6. This chapter analyses the results predicted by the simulation (Chapter 4) and the observations of the tests on the pilot cluster and grid (Chapter 6), and seeks to explain the differences between the two. This chapter details modifications made to the simulation to better fit the observed results, and its new predictions are examined both for the cluster and for the larger grid. Both the original simulation and the final calibrated simulation can be found on the attached media (Appendix A.1).

The key contribution of the research presented in this chapter is the calibration of a simulation to a real grid and cluster environment.

7.2 Comparison of the initial simulation and the pilot environment results

There are differences between the results produced by the initial simulation in Chapter 4 and the results from the pilot study in Chapter 6. Several factors contribute to the differences, including the use of different workflows, the use of different nodes, and accuracy in the original measurements of nodes. This section analyses the differences in these results. Initially the findings from the simulation are compared to the findings from the cluster

tests. Then the simulated nodes are adjusted (by adjusting their parameters), so that they behave more like the nodes used in the pilot and grid studies. The simulation is run again using the same workflows that were used in the cluster and grid studies and the tasks are calibrated to ensure that a given task takes the same amount of time to process on a simulated node as on the corresponding real node.

Key findings from the simulation model (Chapter 4) are listed beside those of the pilot environment (Chapter 6) in Table 7.1. The differences in the findings between the two could be due to a number of potential differences:

Parallel execution: In the simulation a node can process only one task at a time, while in the pilot tests multiple tasks can be executed in parallel on a single node.

Latency: There is a latency between submitting a task in the pilot environment and seeing its impact on CPU utilisation; neither this nor communication latency is incorporated into the model.

Nodes: The attributes and numbers of nodes are different.

Tasks: The tasks were not perfectly calibrated in the simulation. It does not take the same time to execute a task in the simulation as it does in the pilot study on the cluster.

Time: The granularity of time is in whole seconds in the simulation; all simulated tasks take a number of whole seconds to execute. In the pilot environment it was possible for tasks to execute in fractions of seconds.

Table 7.1: Findings from the initial simulation (Chapter 4) and the pilot study (Chapter 6)

Aspect	**Simulation findings**	**Pilot study findings (Cluster environment)**
Mechanisms	In some instances the CRA performs as well as the other mechanisms.	The CRA performs better than the other mechanisms when executing trivial tasks.
Energy	The PPBA performs best in most circumstances.	The PPBA performs better than the CRA but worse than the BA; the allocations of the PPBA were less efficient than those of the BA despite the faster allocation.
Saturation	The resource allocation mechanism can make a substantial difference until a point of saturation is met.	N/A (no finding)
Time	N/A (no finding)	The BA's time cost of having to wait for all nodes to respond prior to allocating tasks is outweighed by the benefit of allocating tasks to the most desirable resources.
Analysis of approach	The use of a resource allocation mechanism that incorporates energy data has the ability to conserve substantial energy.	The use of a resource allocation mechanism that incorporates energy data has the ability to conserve substantial energy.

7.3 Adjustments to the simulation

The results from the simulation in Chapter 4 appear to show some similarity to the observed results from the pilot environment. There were differences, however, in the characteristics of the simulated resources and tasks to the resources and tasks in the pilot environment.

A number of changes have been made to the simulation model to adjust for the identified differences and make it more representative of the real environment. To improve its accuracy in predicting real world phenomena, the simulation model was modified as follows:

Parallel execution: The simulation was modified to allow parallel execution of tasks on each node.

Latency: Communication latency was built into the model, along with a delay between task allocation and measurable increases in CPU utilisation.

Node calibration: More accurate measurements of the energy consumption and processing ability of the nodes were taken and written into the simulation.

Task calibration: The time taken to execute each task used in the workflows was measured for each node, and the simulation adjusted accordingly.

Workflows: The simulation was changed to use the workflows used in the pilot environment.

In the initial simulation model simulated parallel execution was not implemented: a node could process only one task at a time. In the modified model multiple tasks can be executed on the same hardware until node processor saturation occurs. The point of saturation differs for each node, but with the nodes used, it was observed that once a node begins processing a task it almost instantly registers a CPU usage reading of over 50%. The fastest nodes in the grid registered a CPU usage of 75% while processing, and the slower nodes were

typically closer to 100%. In the adjusted simulation, as in the pilot environment, nodes registering a CPU usage value of 50% or more are not allocated a task. Overloading was enabled in the model, but nodes are only assigned multiple tasks after saturation, and even then not if their CPU usage exceeds 50%. This behaviour is consistent with the behaviour that was discovered through observing the allocation of tasks on the cluster. The workflows used in the modified simulation are the same as those used in the pilot study and in the full grid study. The workflows, described in Table 7.2, contain homogeneous tasks that are equally spaced. The task used in the workflows is a prime number search script (Appendix A.3) with additional loops to increase its computational load. It is easily scalable, but the scaling is not linear. To address this in the simulation the time taken to execute the task with different task parameters on the different nodes was measured and incorporated into the model. Table 7.3 displays the idle and processing power and the mean time taken to execute each task for each node used in the pilot environment. Table 7.4 displays the idle and processing power and the number of nodes for each cluster in the grid environment. The power readings displayed were recorded using the same power quality analyser for both the pilot and grid tests (See Section 6.3.5 page 126 for details).

Table 7.2: Basic computational workflows with homogeneous tasks. n denotes computational size

Workflow	Computation size	Number of tasks
1	Small (n=10)	100
2	Medium (n=100)	100
3	Large (n=200)	50

In the initial simulation model, the time that a node would take to complete a task was an estimate based on that node's flops reading. For this recalibration it was decided to incorporate actual readings for the mean time taken to execute each task on each node. Table 7.3 displays the mean time taken by each node in the cluster to execute each task used. The readings were created by booting each node into a live CD environment, executing each task nine times, and recording the execution time with the GNU/Linux `time` command. Energy consumption for each node was also measured during execution and when idle.

Table 7.3: Nodes used in the pilot (cluster) environment, showing power in watts when idle (W_{min}) and when processing at capacity (W_{max}) and mean time taken (seconds) to complete each task (small, medium and large). The master node is indicated by *.

Node	W_{min}	W_{max}	Small Task ($n = 10$)	Medium task ($n = 100$)	Large Task ($n = 200$)
001	50	74	1.6	22.2	46.1
002	50	74	1.6	22.3	46.1
CN1	52	84	0.7	9.9	20.5
CN2	52	84	0.7	9.9	20.6
HPN1*	50	94	0.6	8.9	18.4
HPN2	50	94	0.6	8.9	18.6

Table 7.4: Nodes used in each cluster of the grid environment, showing power in watts when idle (W_{min}) and processing at capacity (W_{max})

Cluster	W_{min}	W_{max}	Nodes
A	50	74	11
B	52	84	11
C	50	94	4

The time taken to process a task as indicated in Table 7.3 cannot be used directly in the simulation. During the pilot and grid tests, nodes were PXE booted from a live CD, they shared a common filesystem, and they communicated with one another to bid for and assign tasks. Testing has shown that this set-up results in substantially different execution times to those when the node is running standalone with the same operating system. The new cluster execution times for each node, when connected to the cluster and using the resource allocation mechanism to allocate a single task, are presented in Table 7.5. With the exception of the master node, which enjoys a lower overhead than the other nodes, execution times in the cluster are three to four times those on the standalone nodes. The task execution times from Table 7.5 were incorporated into the model.

In the grid study there was a delay observed in the execution of tasks on the larger clusters

Table 7.5: Nodes used in the pilot (cluster) environment showing mean time taken (seconds) to complete test tasks as standalone processors and when in the cluster environment. The master node is indicated by *

Node	Small Task ($n = 10$)		Medium task ($n = 100$)		Large Task ($n = 200$)	
	Standalone	In cluster	Standalone	In cluster	Standalone	In cluster
001	1.6	5	22.2	75	46.1	142
002	1.6	5	22.3	75	46.1	142
CN1	0.7	3	9.9	39	20.5	76
CN2	0.7	3	9.9	39	20.6	76
HPN1*	0.6	3	8.9	20	18.4	36
HPN2	0.6	3	8.9	38	18.6	72

in the grid, increasing the execution times of tasks within those clusters. This was possibly due to file system overheads and/or network latency. Each cluster in the grid was using a shared filesystem, and clusters with more nodes showed a corresponding degradation in node response times. It should be noted that network latency was low when observed prior to execution of the tasks, but that it was not tested during workflow execution. The ability to simulate a latency between nodes was incorporated into the simulation to more accurately reflect reality.

7.4 Methodology

Statistical analysis was used to determine if the results produced by the simulation were significantly different from the observations obtained from the pilot and grid environments. A full factorial ANOVA was selected as the method of analysing the data. All analysis was performed using JMP (Sall et al., 2005), and all statistical tests were conducted with the 0.05 level of significance. Several data were excluded prior to statistical analysis, the excluded data are explained in Section 6.4. The data for energy and time was Log transformed prior to performing statistical analysis. The Log transformation of this data results in a better model with a higher Rsquare value. Two models were run, one for time taken and one

for energy. The models used analysed significant differences with the following variables as factors: mechanism, workflow, simulated and environment. Each factor had a number of levels, mechanism had three levels (BA, CRA, PPBA), workflow had three levels (1,2,3), simulated had two levels (yes, no), and environment had two levels (cluster, grid). The summary of fit indicated that the model for energy had an Rsquare value of 0.977 and an Rsquare adjusted value of 0.975, the model for time taken had an Rsquare value of 0.783 and an Rsquare adjusted value of 0.759.

7.5 Results

The results from the calibrated simulation are presented in Table 7.6 alongside the observations obtained from the real environments. Median results from ten runs for each mechanism processing each workflow in both environments are presented.

7.6 Discussion

The results from the execution of each workflow on each environment using each resource allocation mechanism are analysed, and it is examined whether or not the simulated results are significantly different from the observed results. The analysis is discussed first regarding the model for energy then the model for time taken.

7.6.1 Energy

The analysis of variance decomposition of the full factorial ANOVA for energy are displayed in Table 7.7.

According to the analysis of variance decomposition (Table 7.7) the only interaction that was not significantly different was Workflow*Mechanism*Simulated with a p-value of 0.840

Table 7.6: Median results from the pilot environment and the grid environment, both simulated and observed, showing time taken (seconds) and energy consumed for each mechanism processing each workflow (WF)

WF	Mechanism	Environment	$\tilde{x}$(Time) Observed	$\tilde{x}$(Time) Simulated	$\tilde{x}$(Joules) Observed	$\tilde{x}$(Joules) Simulated
1	BA	Cluster	624	614	190008	187825
1	BA	Grid	601	604	734400	754998
1	CRA	Cluster	611	614	186120	187582
1	CRA	Grid	1442	620	1692000	788713
1	PPBA	Cluster	626	614	190800	187828
1	PPBA	Grid	602	604	720000	755001
2	BA	Cluster	831	807	364320	355156
2	BA	Grid	661	641	900000	922668
2	CRA	Cluster	1105	726	440640	333456
2	CRA	Grid	1183	854	1587600	1288130
2	PPBA	Cluster	853	891	375840	380224
2	PPBA	Grid	730	641	979200	923007
3	BA	Cluster	1038	846	434520	362332
3	BA	Grid	974	678	1299600	962125
3	CRA	Cluster	1182	766	485280	340153
3	CRA	Grid	1739	1088	2383200	1573591
3	PPBA	Cluster	1008	836	428040	359483
3	PPBA	Grid	1040	673	1346400	956653

(DF=4). The analysis of variance decomposition indicated that for the interaction Mechanism*Simulated there was a significant difference between the simulated and observed results (p=0.015 DF=2). The Tukey Kramer HSD indicated where those differences were. This test indicated that there was not a significant difference between the simulated BA and observed BA (p=0.067 lower CL=-0.002 upper CL=0.106) or simulated PPBA and observed PPBA (p=0.639 lower CL=-0.025 upper CL=0.083). However, there was a significant difference between simulated CRA and observed CRA (p<0.0001 lower CL=0.05 upper CL=0.161). A Tukey Kramer HSD of the Workflow*Mechanism*Environment*Simulated interaction showed exactly which groups of observations were significantly different to one another. Table C.1 in Appendix C shows a representation of which groups are significantly

Table 7.7: Analysis of Variance Decomposition for energy

Source	DF	F Ratio	Prob > F
Workflow	2	1030.2193	<0.0001
Mechanism	2	55.4992	<0.0001
Environment	1	10991.8632	<0.0001
Simulated	1	32.3666	<0.0001
Workflow*Mechanism	4	7.2253	<0.0001
Workflow*Environment	2	148.9084	<0.0001
Mechanism*Environment	2	41.8882	<0.0001
Workflow*Simulated	2	23.6416	<0.0001
Mechanism*Simulated	2	4.2904	0.0145
Environment*Simulated	1	25.8821	<0.0001
Workflow*Mechanism*Environment	4	3.2334	0.0127
Workflow*Mechanism*Simulated	4	0.3551	0.8404
Workflow*Environment*Simulated	2	7.5482	0.0006
Mechanism*Environment*Simulated	2	11.2849	<0.0001
Workflow*Mechanism*Environment*Simulated	4	10.7535	<0.0001

different by giving groups that are significantly different to one another a different letter. The upper confidence intervals, lower confidence intervals and p-values for the comparison of each group to each other group is displayed in the ordered differences report (Appendix C).

For the cluster results there was no significant difference between the simulated results and the observed results for workflow 1. For workflows 2 and 3 the simulated CRA results were significantly different from the observed results (WF2: $p<0.0001$ lower CL=0.117 upper CL= 0.474, WF3: $p<0.0001$ lower CL=0.236 upper CL=0.593).

For the grid environment there were no significant differences for the results from workflow 1, there was a significant difference for the CRA results for workflow 2 (p=0.011 lower CL= 0.020 upper CL=0.376), and there was a significant difference for the BA results for workflow 3 (p=0.0018 lower CL=0.040 upper CL=0.396).

In regard to energy usage it can be seen that the simulation is not perfect. The simulated CRA mechanism consistently produced results that were significantly different to the observed results from the CRA mechanism for the cluster environment. However, it can be

Table 7.8: Analysis of Variance Decomposition for time taken

Source	DF	F Ratio	Prob > F
Workflow	2	292.1834	<0.0001
Mechanism	2	41.9743	<0.0001
Environment	1	72.3618	<0.0001
Simulated	1	83.5407	<0.0001
Workflow*Mechanism	4	5.9351	0.0001
Workflow*Environment	2	29.6495	<0.0001
Mechanism*Environment	2	28.3494	<0.0001
Environment*Simulated	1	10.2189	0.0015
Workflow*Simulated	2	22.6550	<0.0001
Mechanism*Simulated	2	10.6141	<0.0001
Workflow*Mechanism*Simulated	4	0.6402	0.6342
Workflow*Mechanism*Environment	4	3.1809	0.0139
Workflow*Environment*Simulated	2	6.6306	0.0015
Mechanism*Environment*Simulated	2	16.4958	<0.0001
Workflow*Mechanism*Environment*Simulated	4	13.6754	<0.0001

seen that in most situations most simulated mechanisms produced results that were not significantly different from the observed results.

7.6.2 Time taken

The time test results of the full factorial ANOVA for energy are displayed in Table 7.8.

With regards to time taken the only term that was not significantly different according to the analysis of variance decomposition (Table 7.8) was Workflow*Mechanism*Simulated with a p-value of 0.6342 (DF=4).

A Tukey Kramer HSD of Workflow*Mechanism*Environment*Simulated showed exactly where each group of observations were significantly different to the simulated results. Table C.2 in Appendix C shows a representation of which groups were significantly different by giving groups that were significantly different to one another a different letter. The

upper confidence intervals, lower confidence intervals and p-values for the comparison of each group to each other group is displayed in the ordered differences report (Appendix C).

For the cluster results there was no significant difference for workflow 1. For workflow 2 and workflow 3 the simulated results for the CRA mechanisms were significantly different to the cluster observations (WF2: $p<0.0001$ lower CL=0.254 upper CL=0.636, WF3: $p<0.0001$ lower CL=0.316 upper CL=0.698).

For the grid results there was no significant difference between the simulated results and the observed results for any mechanisms when processing workflow 1 or 2. For workflow 3 the simulated BA results were significantly different from the BA observations ($p<0.0001$ lower CL=0.083 upper CL=0.465).

In regards to time taken, the simulation often produced results that were not significantly different to the observed results. The CRA mechanism produced significantly different results for 2 out of 3 workflows for the cluster environment. For the grid environment only the simulated BA mechanism produced significantly different results to the observed results and only for workflow 3.

No simulation can be a perfect representation of an underlying system, because simulations incorporate simplifying assumptions. "The simplicity of models, compared with reality, lies in the fact that only the relevant properties of reality are represented" (Ackoff et al., 1962, p.108). The simulation model used in this research often produced results that showed the same phenomena and were not significantly different to the observed results. It is for these reasons that the simulation was believed to be a close analogue of the simulated system, and was consequently used in the remainder of this research.

7.7 Concluding remarks

This chapter has described the methods used to calibrate the simulation to real distributed resources. Results from the calibrated simulation show that it produced the same phenomena as those observed in the real distributed resources in most situations. A calibrated

simulation allows for the examination of new and old resource allocation strategies by examining them processing a variety of workflows that would have taken too much time and resources to be practical with physical hardware.

Chapter 8

THE EXTENSION AND ANALYSIS OF THE CALIBRATED SIMULATION

8.1 Introduction

The simulation model described in Chapter 4 was calibrated to a cluster and grid environment in Chapter 7. In this chapter the model is extended to incorporate an additional performance metric and an additional auction, and this extended model is used to simulate the execution of two workflows derived from two well known grid traces.

The key contributions of the research presented in this chapter are: a calibrated simulation processing a number of grid workflows and an analysis of the simulated execution of a real grid workflow.

8.2 Extension and analysis of the calibrated simulation

One of the benefits of a simulation is the ability to ask the "What if?" questions. This ability was used to examine additional resource allocation mechanisms and large workflows. Chapter 7 has discussed and detailed the calibration of the simulation to the real grid and cluster. Five resource allocation mechanisms have been used in the examination of the calibrated simulation, three of the mechanisms, the BA, PPBA, and CRA, were described in Chapter 4 and have been widely used in the research reported to this point. Two additional methods of resource allocation have been added, the Marginal Increase Mechanism (MIM) and the Continuous Double Auction (CDA). MIM (Algorithm 13) is actually not a new

resource allocation mechanism as defined in Chapter 4. Rather, it is a standard batch auction (BA) that uses a different performance metric in the sense of Chapter 3: instead of allocating the task to the node with the greatest ratio of performance to power, MIM allocates the task to the node that has the greatest ratio of performance to *marginal* power. The value is calculated as $compPower/(Watts_{max} - Watts_{min})$, and is designed to reward nodes that consume less additional energy when moving from an idle state to a processing state. MIM is most useful when there is no ability to switch off idle nodes or to send idle nodes into a low energy mode. Although MIM is an existing resource allocation mechanism with a different performance efficiency metric, it is convenient for the purposes of this chapter to regard it as a distinct mechanism and to compare it with the other mechanisms.

Algorithm 13 Marginal Increase Mechanism (MIM) simplified

Ask each node for a bid using marginal performance efficiency metric

Sort bids in order of value

Assign task to highest bidding node

Start task execution

The other new mechanism is the Continuous Double Auction (CDA), described in Algorithm 14. Like CRA, CDA is a random allocation; but unlike CRA, rather than allocating a task to the first node to bid, it waits until two nodes have bid and allocates the task to the one with the higher bid. In this simulation of the CDA the resources are shuffled between asks, to simulate a random return of bids.

Algorithm 14 Continuous Double Auction (CDA) simplified

Shuffle nodes

Ask each node for a bid

Wait until two bids have been received

Assign task to highest bidding node

Start task execution

8.3 Methodology

The simulation has been calibrated to both the cluster and the grid. It is now used to analyse a number of synthetic and non-synthetic workflows. The primary research question is: "does altering the resource allocation mechanism or the performance metric affect the allocation of resources in a way that alters the total energy used or the time taken in the execution of tasks?" This question will be asked through the following hypotheses:

H1: Altering the resource allocation mechanism will result in a significant difference in the amount of energy consumed.

H2: Altering the resource allocation mechanism will result in a significant difference in the amount of time taken to execute the workflow.

H0: Altering the resource allocation mechanism will not affect the allocation of resources in a way that alters the time to execute tasks or the total energy used in the execution of tasks.

8.3.1 Synthetic workflows

The three synthetic workflows described in Table 7.2 were used to analyse the comparative performance of the five resource allocation mechanisms. Table 8.1 displays the results for the BA, CRA, CDA, MIM, and PPBA resource allocation mechanisms, expressed as percentages of the results for the BA mechanism. The CRA and CDA results are obtained from static runs of those mechanisms; that is, their random number generators were seeded so that they would generate the same sequence of nodes. For comparison, Table 8.2 displays average results from unseeded runs of the CDA and CRA mechanisms as a percentage of the results produced by the BA mechanism.

Table 8.1: Results from the calibrated simulation for five mechanisms when processing three synthetic workflows; all results are shown as percentage differences from the BA mechanism. All mechanisms are non-stochastic or seeded for these runs.

	Time as a percentage of BA			Energy as a percentage of BA		
Node	Workflow 1	Workflow 2	Workflow 3	Workflow 1	Workflow 2	Workflow 3
BA	0.0	0.0	0.0	0.0	0.0	0.0
CRA	2.7	28.6	56.9	4.5	34.5	60.8
CDA	-0.2	25.7	42.8	0.8	29.4	45.0
MIM	-0.3	0.0	7.5	-1.1	-1.3	5.4
PPBA	0.0	0.0	-0.7	0.0	0.0	-0.6

Table 8.2: Results from the calibrated simulation for multiple runs of CDA and CRA as a percentage difference of BA results, when processing the synthetic workflows

Mechanism	Workflow	Number of Runs	Mean power (% BA)	Mean time (% BA)	Mean energy (% BA)
CRA	1	10	1.38	1.60	1.44
CRA	2	10	4.99	6.26	27.24
CRA	3	10	2.11	4.63	52.86
CDA	1	10	0.61	0.78	0.60
CDA	2	10	1.60	2.55	30.23
CDA	3	10	0.40	3.53	42.06

8.3.2 Analysis of resource allocation mechanisms

The simulation has been calibrated to the grid. In this section two well known grid traces have been examined and results are presented on the execution of these grid traces over the now calibrated simulated grid using each of the five auctions.

To further examine the effect on grid energy consumption and workflow execution time of using different auctions to allocate resources, two grid traces from the Grid Workloads archive (Anoep et al., 2009) have been converted to workloads and their execution simulated

on the calibrated grid.

The first trace selected was the DAS-2 Grid trace[1]. This trace, which consisted of 1,124,772 tasks over 659 days, has been described in detail in Section 4.3.2 of this thesis. Figure 4.2 displays the number of tasks submitted per day, and Figure 4.3 showed the simulated computation required per day to complete these tasks.

The second trace used in this research was the LCG trace[2] retrieved from The Grid Workloads Archive. This trace consisted of 188,041 tasks submitted over 11 days. The number of tasks submitted per day is shown in Figure 8.1. The simulated computation for the LCG trace is the same value as the number of tasks.

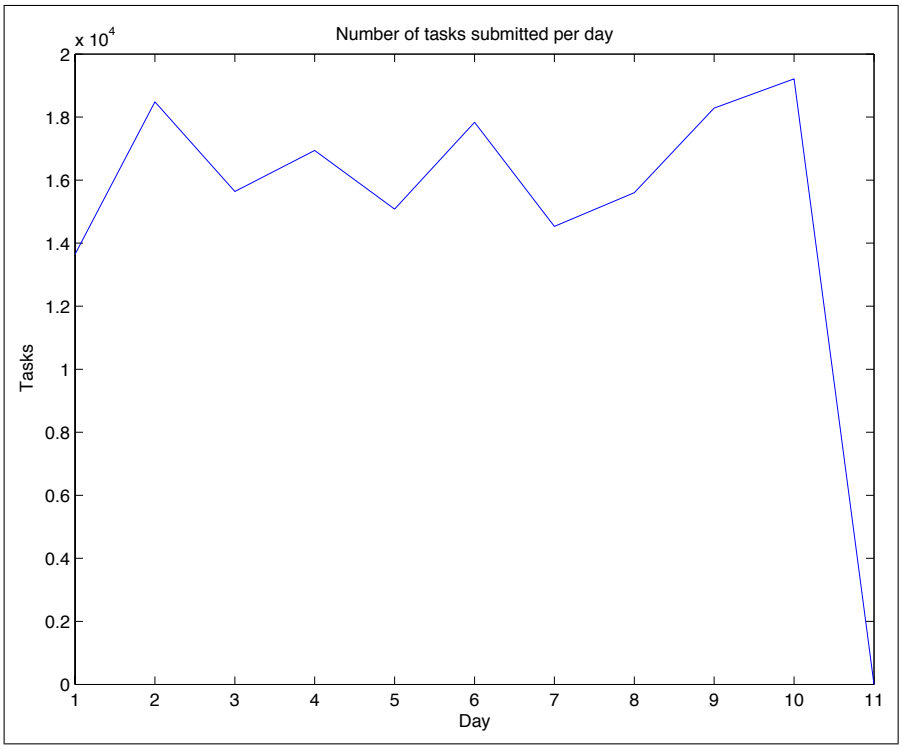

Figure 8.1: Number of tasks submitted per day in LCG grid trace

[1] provided by the Advanced School for Computing and Imaging, Delft University of Technology, Netherlands, `http://www.asci.tudelft.nl/` retrieved from The Grid Workloads Archive

[2] provided by the e-Science Group of HEP at Imperial College London

The traces have been used to provide a more realistic flow of load for the proposed resource allocation mechanisms. Each trace was presented in the form of a list of tasks submitted to the grid, containing the submit time, the average processor time, and the number of processors used. Information on computation that was not available is represented by the value -1. It cannot be determined from the traces how much load was applied to any of the nodes in the grids.

For use in the simulation the traces have been used as a guide for the flow of tasks, transformed into grid workflows, and executed over the simulated grid. Where the trace shows a value of -1 for number of processors or processing time, this has been replaced with 1.

The nodes used in the simulation for the execution of the grid trace were modelled on the same nodes that were used in the grid environment (Chapter 6).

8.4 Results

The results of the grid trace experiments are presented in Table 8.3. The results show power usage (Watts), total energy consumption, total end of execution (EOE)[3] energy consumption, and time to execute. All values displayed in the table are as a percentage of the results of the BA mechanism running the same workflow. Total energy consumption was measured from the start of the simulation to the end of the simulation, total EOE energy consumption was measured from the start of the simulation until the last task finished in the simulation.

8.5 Discussion

The collected data were analysed to see if there was a significant difference between sets. The BA, PPBA, and MIM mechanisms were non-stochastic in the simulation, but the CRA

[3] End of execution energy consumption measures energy consumption from the start of execution of the first task until the end of execution of the last task in a workflow.

Table 8.3: Results of grid trace tests as a percentage of the results for the BA mechanism

	Energy consumption as a percentage of BA		Tasks completed as a percentage of BA	
Mechanism	DAS-2 trace	LCG trace	DAS-2 trace	LCG trace
BA	100.0	100.0	100	100
CDA mean	100.5	104.2	77	100
CRA mean	101.5	107.1	77	100
MIM	99.6	99.0	100	100
PPBA	98.7	100.0	99	100
Individual runs contributing to CDA mean				
CDA	100.49	104.22	77	100
CDA	100.50	104.20	77	100
CDA	100.51	104.19	77	100
CDA	100.54	104.19	77	100
CDA	100.53	104.18	77	100
Individual runs contributing to CRA mean				
CRA	101.45	107.09	77	100
CRA	101.48	107.06	77	100
CRA	101.46	107.06	77	100
CRA	101.45	107.08	77	100
CRA	101.45	107.06	77	100

and CDA mechanism were run five times each, as both rely on random allocation of tasks. Kruskal-Wallis ChiSquare approximations and the Wilcoxon Method for each pair were used to determine if there was a significant difference between the energy consumption and execution times for the resource allocation mechanisms. Raw statistical output of these tests is presented in Appendix B.0.2. The kruskal-Wallis ChiSquared approximation showed that the results for energy and time were all significantly different for both the DAS2 workflow (Energy: $p<0.001$ DF=4 ChiSquare=23.6, Time: $p=0.0032$ DF=4 ChiSquare=15.9) and the LCG workflow (Energy: $p<0.001$ DF=4 ChiSquare=23.6, Time: $p=0.0002$ DF=4 ChiSquare=22.6). For energy all mechanisms produced results that were significantly different from one another and all at the 0.05 confidence interval for both the the DAS2 and LCG workflows. For time, all stochastic mechanisms produced results that were signifi-

cantly different from one another and all deterministic mechanisms with the exception of the CDA and CRA with the LCG workflow. The results indicate that the time taken to execute a workflow and the energy used in workflow execution is affected based on the resource allocation mechanism chosen. The null hypothesis (H0) is therefore rejected.

The BA, CDA, CRA, MIM, and PPBA mechanisms were used to assign tasks when running a simulation that used the DAS-2 workload for its stream of tasks. The simulated energy consumption are presented in Table 8.3. The results have been presented relative to the results produced by the BA mechanism. For energy consumption; total energy, power usage (Watts) and total EOE were recorded. For the three aforementioned values the BA mechanism performed worse than both the PPBA and the MIM mechanisms. The PPBA consumed 1.27% less energy and the MIM consumed 0.42% less energy than the BA. However, the CRA mechanism consumed significantly more energy, on average the CRA mechanism consumed 1.46% more energy than the BA mechanism, whilst the CDA consumed 0.51% more energy than the BA mechanism. With regards to the time taken to process tasks there was no difference between mechanisms, when simulating processing the DAS-2 workflow. There were a number of tasks that were not completed during the simulation indicating that the simulated resources became saturated, and could not finish processing the tasks. The PPBA, BA, and MIM mechanisms completed close to 100% of tasks. However the CRA and CDA mechanisms were only able to complete 76.8% of tasks within the simulated time. The number of jobs remaining when utilising the CRA and CDA mechanisms indicated that many tasks were assigned to slower resources, and could not be completed within the allotted time window. The time window is the maximum amount of time that the simulation can simulate for before the workflow is ended. Overall it can be seen that the random allocation produced by the CRA and CDA mechanisms consumed significantly more energy than the other mechanisms when processing the simulated DAS-2 workflow in this simulated environment.

The results of the LCG workflow were similar to those of the DAS-2 workflow. The BA, CDA, CRA, PPBA, and MIM mechanisms were used to allocate resources when simulating using the LCG workflow. For energy consumption the allocations produced by the BA

and PPBA mechanisms resulted in identical energy consumption figures, however those produced by the CRA and CDA resulted in energy consumption that was significantly higher than the BA and the allocation produced by the MIM produced an allocation that utilised significantly less energy than that of the BA. The allocations produced by the CRA on average used 7% more energy than the allocations produced by the BA mechanism. The allocations produced by the CDA on average used 4.2% more energy than the allocations produced by the BA mechanism. The allocation produced by MIM utilised 0.9% less energy than the BA or PPBA mechanism, which represents a significant energy saving.

The results for time were less than 1% different across any mechanism when simulating the LCG workflow. Unlike the simulation for DAS-2, nearly all tasks were completed with each mechanism. There was less than 0.01% difference in the number of tasks completed. Overall the simulation showed that the BA and PPBA resulted in near identical outcomes, the BA and PPBA used significantly less energy than the CRA mechanism and MIM used significantly less energy than the BA. There were no significant differences observed in the time taken to execute the LCG workflow.

8.6 Concluding remarks

This chapter has described the results of executing grid workflows constructed from two well known grid traces. The results presented indicate that the use of simple resource allocation mechanisms that incorporate energy and performance attributes of individual nodes can reduce the energy consumption of a distributed computing environment such as a grid or cluster by up to 7% (the LCG trace processed with the CRA mechanism on average consumed 107.06% of the energy consumed by the BA mechanism).

A calibrated simulation allows for the examination of new and old resource allocation strategies by examining them processing a variety of workflows that would have taken too much time and resources to be practical with physical hardware.

The composition of the simulated resources has an impact on the effectiveness of resource

allocation in conserving energy. Both heterogeneous resources and spare processing capacity are needed for resource allocation to be an effective method of energy conservation. Varying the diversity of the available resources or the number or resources results in changes to the amount of energy that can be conserved.

This chapter has presented a calibration study that has examined a number of economic resource allocation mechanisms and adds weight to evidence that suggests that the use of these mechanisms can result in reductions to energy usage in distributed computing environments.

8.6.1 Key finding

The key finding from the research presented in this chapter is:

- The use of simple resource allocation mechanisms that incorporate energy and performance attributes of individual nodes can reduce the energy consumption of a distributed computing environment such as a grid or cluster by up to 7% when utilising realistic workflows.

Chapter 9

CONCLUDING REMARKS AND FURTHER WORK

9.1 Summary of research

This thesis examines the question: Can economic resource allocation mechanisms be used in distributed computing environments to reduce energy consumption whilst maintaining execution speed? This thesis shows that in some circumstances economic resource allocation mechanisms can reduce the total energy consumption of the distributed resource whilst maintaining execution speed.

Chapter 3 examined the use of different performance metrics for the ranking of nodes in distributed computing environments. The examination occurred through simulation, which showed that the use of different metrics to rank nodes can make a difference to the relative rank of nodes and thus to total energy consumption. The argument was made that a performance metric used to rank nodes should account for a node's energy consumption and computational performance. Ranking is an important aspect of resource allocation, although it is not the only aspect of resource allocation. The use of a metric that accounts for both the node's power consumption and computational ability will result in the most energy efficient allocations.

In Chapter 4 a simulation study of three economic resource allocation mechanisms is presented, simulating the execution of eighteen synthetic workflows. The simulation indicated that the choice of resource allocation mechanism can make a difference to the energy used and the time taken to execute tasks. The simulation showed that the PPBA and the BA mechanisms resulted in the quickest execution times and the least energy usage. The CRA

resulted in considerably higher energy consumption when processing most workflows. It was shown through simulation that economic resource allocation mechanisms can be implemented to reduce the energy consumption of a distributed computational environment.

Chapter 5 described the reasoning behind the use of e-waste resources for the construction of the cluster and grid environment. These environments were used to examine the effectiveness of economic resource allocation mechanisms in reducing energy consumption whilst maintaining execution speed. This chapter described the creation of a cluster similar to the cluster that is used in subsequent experiments. It examined the merits of the use of e-waste resources as high-performance distributed computing resources. It was found that it is possible to use them as such, but that the ratio of energy consumption to computational performance of such resources was low. This chapter highlights the amount of resources that go into the creation of computers and posits that it is possible to re-use these resources under certain circumstances. It is possible and indeed, in some situations desirable to use e-waste resources as computational grids or clusters.

Chapter 6 detailed the implementation of the simulated resource allocation mechanisms onto real environments. The effectiveness of the resource allocation mechanisms at reducing energy consumption and maintaining performance are explored on both a cluster and a grid environment. The resource allocation mechanisms described were created and initially used to allocate tasks submitted to a pilot cluster; this environment was then scaled up to an institutional grid environment. On both environments an experiment was performed to determine the effect, if any, altering the resource allocation mechanism would have on the time taken to execute tasks and the energy used in task execution. In these lab conditions using resource allocation alone can save significant energy without sacrificing speed. The tests show a significant difference, however the tests are in a lab environment using synthetic workflows.

The simulation was initially of a cluster environment. Chapter 7 examines the calibration of the simulation to both the cluster and the grid used. The results from this calibrated simulation are compared to the results observed from the real environments. This chapter

describes the calibration of the simulation described in Chapter 4 to the cluster and grid environments in Chapter 6. This chapter shows that the now calibrated model accurately produces phenomena that are the same as those produced by the real environments, and produces results that are not significantly different from the real environments. The initial simulation model is now calibrated to the two environments used in the experiments. This simulation uses the same synthetic workflows and contains the same nodes for each environment. The analysis of this simulation suggests that although it does not perfectly show the same numbers, the simulation produces the same phenomena.

Chapter 8 details and examines a number of extensions to the model to aid in the understanding of the use of resource allocation to conserve energy whilst maintaining performance. The simulation of two well known grid workflows are presented and additional resource allocation mechanisms are explored. The results from the simulation suggest that with a realistic workflow, savings of up to 7% could be obtained through the use of resource allocation alone.

9.2 Findings

A number of findings were made throughout the course of this research and will be summarised on a topic by topic basis.

Node ranking methods

Resource allocation mechanisms require some method of ranking nodes; in this research different performance metrics were examined for use in ranking of nodes. Some metrics accounted for only computational performance, some for only power usage, and others both.

Chapter 3 examined what effect the use of differing performance metrics for node evaluation by a resource allocator has on overall energy and speed of execution. Initially a simulation

was used to investigate the effect that selection of performance metrics can have on total energy used and mean task execution time. The computational resources were modelled on real resources. The simulation showed that in most cases the choice of performance metric had an impact on the total energy consumed and on the mean time taken to execute a task. The Ettt performance metric generally resulted in the least total energy consumption, but generally at a slight expense in execution time.

The observations from experimentation on a cluster environment examined in Chapter 6 confirmed that the performance metric chosen to rank nodes can make a significant difference when it comes to the time tasks take to execute and the energy used. It was shown that selection of the optimal performance metric could save up to 6% of time and 4% of energy usage.

Resource allocation mechanisms

The thesis primarily examined the use of different economic resource allocation mechanisms in distributed computing environments, it assessed those mechanisms on their ability to reduce energy consumption whilst maintaining execution speed. In Chapter 4 a simulation of a distributed computational environment was constructed. The simulation analysed the effect that different resource allocation mechanisms could have on the time taken to process tasks and on energy consumption. Both the time taken to execute tasks and the energy utilised in execution have been shown to alter with choice of resource allocation mechanism. These differences are accentuated when there are many tasks that require substantial computation, and are much smaller as the size and the number of tasks diminishes.

The results of this simulation study suggest that in some circumstances, resource allocation that incorporates the energy efficiency of the nodes can on its own make a difference to the total energy consumption of the distributed environment. This was shown in the simulation results where both the BA and PPBA mechanisms consistently resulted in lower consumption of energy when processing the same workflow as the CRA mechanism.

The zero intelligence approach used by the CRA mechanism to allocate resources, may be appropriate in some circumstances. It was shown to perform well relative to the other mechanisms when tasks required little computation; it was posited that this was due to the ability of the CRA to very quickly allocate resources.

It was shown that different mechanisms are suited to different workflows. With regard to energy consumption it can be seen that the PPBA performed best in most circumstances, but there were workflows for which other mechanisms performed as well as or better than the PPBA.

The overall impact on energy consumption of changing the resource allocation mechanism was shown to be small. The largest energy saving, realised in the simulation, was 3.4%.

It was discovered that the use of VOVO in combination with the use of different resource allocation mechanisms results in greater energy savings than resource allocation alone. When processing every workflow the total energy consumption was lower when running VOVO than it was when executing the same workflow, running the same mechanism, without the use of VOVO. A marked example is the processing of a workflow using the BA which consumed 1437 watt hours with VOVO turned off, compared with only 224 watt hours with VOVO turned on, a saving of nearly 85%.

E-waste resources

The research required exclusive use of dedicated distributed computing resources. Chapter 5 presented the controlled re-use of previously disused computers as a method of e-waste management. The use of e-waste resources offers the potential to save money and to reduce the flow of e-waste. The creation of a small cluster of e-waste resources was examined. It was demonstrated that although it may be possible and desirable to prolong the use of older computers, it is apparent that the use of these resources is not energy efficient. Work produced as part of this research was conducted using distributed computational environments created from e-waste resources.

Experimentation and simulation

Chapter 6 describes tests that were conducted on a real cluster and grid environment. The tests examined both the use of different resource allocation mechanisms and the use of different performance metrics to rank nodes. It was found that economic resource allocation mechanisms that incorporate the power and performance data from individual nodes offer the ability to conserve significant amounts of energy whilst maintaining the speed of execution. Results from synthetic workflows in the cluster environment showed energy savings of up to 34%. Results from synthetic workflows in the grid environment also showed significant savings, although less pronounced. A calibrated simulation study was presented in Chapter 8 which showed that with a workflow derived from a real grid trace, a saving of up to 7% of energy could be achieved. Furthermore it was discovered that for the purpose of understanding the effects of different grid resource allocation mechanisms, a cluster environment could be used, to some extent, as an analogue of a grid with the exception of network latency and bandwidth issues.

Resource allocation can be used to conserve energy in distributed computing environments without any new hardware. It is most effective when there is spare processing capacity and heterogeneous computing resources.

The results obtained in the experiments conducted were from a controlled environment with synthetic workflows. The results observed show that selection of an appropriate mechanism can often conserve significant quantities of energy, however, the percentage of energy that can be conserved differs depending on the number of nodes and the characteristics of the workflow used.

9.3 Future research directions

In the course of this research several potential extensions came to light. These extensions include additional analysis of VOVO, the analysis of the effectiveness of a second price

auction, the examination of cooling systems, the study of an in-use grid environment, examination of first-come first-serve, and the use of virtualisation.

In Chapter 4 there was a brief analysis of the effect of VOVO. Although a basic implementation of VOVO was examined, this research could be extended to further investigate combining the resource allocation mechanisms examined in conjunction with the use of a mechanism such as VOVO to dynamic alter the power states of nodes.

The research presented on resource allocation in Chapters 4, 6, and 8 examined the execution of an entire workflow using a single resource allocation mechanism at a time. It was discovered that different mechanisms were better suited to different workflows. A useful extension would be to create a self-adjusting resource allocator that dynamically alters the resource allocation mechanism to best allocate the predicted incoming stream of tasks.

A number of auctions were examined, but one notable type of auction that was not examined is a second-price auction such as the Vickrey auction. The implementation of a Vickrey auction may result in the resource allocator distributing tasks to the second most energy efficient node available at any given point in time. Although this may not result in the most energy efficient allocations, it could be an interesting extension of this research in comparison to the other auctions studied.

The focus of this research was predominantly on energy consumption. However, the energy consumption readings did not include the energy required to cool server rooms. The energy required to cool server rooms would obviously fluctuate based on ambient temperature and also the heat produced by the computational resources. Incorporating cooling energy into the calculations of the resource allocation mechanisms could be a worthwhile extension to this research.

The research in this thesis was performed through simulation and in controlled cluster and institutional grid environments. A study on the implementation of the same resource allocation mechanisms in an active grid environment would be difficult due to the dispersed nature of such resources, however it would be a useful extension to this research.

This research used the CRA, which is a random allocation, to compare the effectiveness of the other mechanisms. An alternative resource allocation mechanism to compare to would have been first-come first-serve, where the next task is submitted to the first available node. This is a conceptually logical way of allocating tasks, in that it is the same method that is used at the supermarket to allocate people to registers, or at the bank to allocate people to bank tellers. The energy efficiency of first-come first-serve could depend heavily on the energy efficiency of the fastest node. It would be of interest to examine first-come first-serve in relation to the other mechanisms examined. It may be appropriate to use it as a baseline in future research.

Virtualisation is increasingly being used in industry. Virtualisation technology allows many virtual servers to reside on a single physical computing resource. This has brought with it new opportunities and challenges. Virtualisation can save energy by reducing the amount of physical hardware that is in use. A worthwhile extension of this research would be to modify the resource allocator to examine the effect of an independent virtualisation un-balancer on the effectiveness of the studied resource allocation strategies.

This research examined a number of real and synthetic workflows. The real workflows used were derived from high performance distributed computing resources. It would be of interest to extend this research to examine the economic resource allocation mechanisms used in this research in other domains, such as web-hosting environments.

9.4 Final remarks

This thesis examined the use of economic resource allocation mechanisms in distributed heterogeneous computing environments, to discover if the use of economic resource allocation could minimise energy consumption whilst maintaining execution speed. Economic resource allocation mechanisms were examined through a combination of simulation and experimentation.

The results suggest that the economic resource allocation mechanisms examined can be

used to conserve energy in distributed computing environments without any new hardware. Resource allocation can be used as an energy conservation mechanism where there is spare processing capacity and heterogeneous computing resources. The results indicate that economic resource allocation mechanisms that incorporate the power and performance data from nodes offer the ability to conserve significant amounts of energy whilst maintaining the speed of execution.

To conduct the experiments a cluster and grid were created using e-waste resources. It was discovered that in some circumstances such resources can be used to produce usable environments whilst minimising the flow of e-waste. The main original contribution of this thesis was the creation of a novel resource allocation mechanism that utilised auctions to conserve energy and maintained the speed of task execution in distributed computing environments such as grids.

Many future directions have been identified as a result of the research undertaken. It is hoped that these directions will be explored with the goal of decreasing the environmental footprint of distributed computing.

BIBLIOGRAPHY

ABS (2006), Australia's environment: Issues and trends 2006, Technical Report cat no. 4613.0, The Australian Bureau of Statistics (ABS), Canberra.

Ackoff, R., Gupta, S. K. and Minas, J. S. (1962), *Scientific Method: optimizing applied research decisions*, John Wiley & Sons Inc, New York, NY, USA.

Adams, J. C. and Brom, T. H. (2008), Microwulf: a Beowulf cluster for every desk, *in* 'SIGCSE '08: Proceedings of the 39th SIGCSE technical symposium on Computer Science Education', ACM, New York, NY, USA, pp. 121–125.

Ahluwalia, P. K. and Nema, A. K. (2007), 'A life cycle based multi-objective optimization model for the management of computer waste', *Resources, Conservation and Recycling* **51**(4), 792–826.

Anderson, D. P. (2004), BOINC: A system for public-resource computing and storage, *in* 'Proceedings. Fifth IEEE/ACM International Workshop on Grid Computing', pp. 4–10.

Anderson, D. P. (2007), Local scheduling for volunteer computing, *in* 'IEEE International Parallel and Distributed Processing Symposium IPDPS 2007.', pp. 1–8.

Anoep, S., Dumitrescu, C., Epema, D., Iosup, A., Jan, M., Li, H. and Wolters, L. (2009), 'The grid workloads archive', Retrieved March 3, 2009, from `http://gwa.ewi.tudelft.nl`.

Arnold, K., Gosling, J. and Holmes, D. (2000), *The Java Programming Language*, 3rd edn, Addison-Wesley.

AuYoung, A., Chun, B. N., Snoeren, A. C. and Vahdat, A. (2004), Resource allocation in federated distributed computing infrastructures, *in* 'Proceedings of the 1st Workshop on Operating System and Architectural Support for the On-demand IT InfraStructure', Boston, MA, USA.

Aziz, A. and El-Rewini, H. (2009), Power aware scheduling in computational grids, *in* 'The 2009 International Conference on Parallel and Distributed Processing Techniques and Applications', CSREA Press, Las Vegas, NV, USA.

Bai, P. (2009), Advancing Moore's law: Challenges and opportunities, *in* 'ASICON '09. IEEE 8th International Conference on ASIC, 2009.', pp. 7–8.

Banks, J. (1998), *Handbook of simulation*, John Wiley & Sons, New York, chapter Principles of Simulation, pp. 3–30.

Basel Convention on the Control of Transboundary Movements of Hazardous Wastes and Their Disposal (2008), Retrieved May 4, 2008, from `http://www.basel.int/text/con-e-rev.pdf`. Adopted 22 March 1989, 28 ILM 657 (in force 2 May 1992), Geneva, CH; United Nations Environment Programme (UNEP).

Baumann, D. (2008), 'Debian live, live Debian systems', Retrieved July 19, 2008, from `http://debian-live.alioth.debian.org/`.

Beloglazov, A. and Buyya, R. (2010), Energy efficient resource management in virtualized cloud data centers, *in* 'The 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid)', pp. 826–831.

Bubendorfer, K. and Thomson, W. (2006), Resource management using untrusted auctioneers in a grid economy, *in* 'Second IEEE International Conference on e-Science and Grid Computing. e-Science '06.', pp. 74–74.

Buyya, R., Abramson, D., and Stockinger, J. G. H. (2002), 'Economic models for management of resources in grid computing', *The Journal of Concurrency and Computation: Practice and Experience* **14**(13), 1507–1542.

Buyya, R., Abramson, D. and Venugopal, S. (2005), 'The grid economy', *Proceedings of the IEEE* **93**(3), 698–714.

Buyya, R. and Murshed, M. (2002), 'Gridsim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing', *Concurrency and Computation: Practice and Experience* **14**(13–15), 1175–1220.

Byde, A. (2003), Applying evolutionary game theory to auction mechanism design, *in* 'EC '03: Proceedings of the 4th ACM conference on electronic commerce', ACM Press, New York, NY, USA, pp. 192–193.

Cairns, C. N. (2005), E-waste and the consumer: improving options to reduce, reuse and recycle, *in* 'Proceedings of the 2005 IEEE International Symposium on Electronics and the Environment', pp. 237–242.

Cao, J., Spooner, D. P., Jarvis, S. A., Saini, S. and Nudd, G. R. (2003), Agent-based grid load balancing using performance-driven task scheduling, *in* 'Proceedings. International Parallel and Distributed Processing Symposium'.

Carlson, S. (2003), 'Old computers never die–they just cost colleges money in new ways', *Chronicle of Higher Education* **49**(23), A33–A35.

Cartier, M. (2004), 'An agent-based model of innovation emergence in organizations: Renault and ford through the lens of evolutionism', *Computational and Mathematical Organization Theory* **10**, 147.

Chapman, C., Musolesi, M., Emmerich, W. and Mascolo, C. (2007), Predictive resource scheduling in computational grids, *in* 'Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International', pp. 1–10.

Chase, J. and Doyle, R. (2001), Energy management for server clusters, *in* 'HOTOS '01: Proceedings of the Eighth Workshop on Hot Topics in Operating Systems', IEEE Computer Society, Washington, DC, USA, p. 165.

Chivers, W. J. and Herbert, R. D. (2003), 'The effects of varying parameter values and heterogeneity in an individual-based model of predator-prey interaction', *Advances in Complex Systems* **6**(3), 441–456.

Choi, B.-C., Shin, H.-S., Lee, S.-Y. and Hur, T. (2006), 'Life cycle assessment of a personal computer and its effective recycling rate', *The International Journal of Life Cycle Assessment* **11**(2), 122–128.

Chun, B. N. and Culler, D. E. (2000), Market-based proportional resource sharing for clusters, Report UCB/CSD 00/1092, Berkeley : Computer Science Division, University of California.

Costa, G. D., Gelas, J.-P., Georgiou, Y., Lefevre, L., Orgerie, A.-C., Pierson, J.-M., Richard, O. and Sharma, K. (2009), The green-net framework: Energy efficiency in large scale distributed systems, *in* 'Parallel and Distributed Processing Symposium, International', pp. 1–8.

Curnow, H. J. and Wichmann, B. A. (1976), 'A synthetic benchmark', *Computer Journal* **19**(1), 43–49.

Eadline, D. (2008), 'Green HPC: The new secret to going fast', Retrieved October 2, 2008, from `http://www.linux-mag.com/id/7028`.

Elnozahy, E., Kistler, M. and Rajamony, R. (2003), *Energy-Efficient Server Clusters*, Lecture Notes in Computer Science, Springer Berlin / Heidelberg, chapter Power-Aware Computer Systems, pp. 179–197.

Foster, I., Fidler, M., Roy, A., Sander, V. and Winkler, L. (2004), 'End-to-end quality of service for high-end applications', *Computer Communications* **27**(14), 1375–1388.

Foster, I. and Kesselman, C. (2001), Computational grids, *in* J. Palma, J. Dongarra and V. Hernández, eds, 'Vector and Parallel Processing — VECPAR 2000', Vol. 1981 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 3–37.

Foster, I., Kesselman, C., Nick, J. and Tuecke, S. (2002), 'Grid services for distributed system integration', *Computer* **35**(6), 37–46.

Foster, I., Kesselman, C. and Tuecke, S. (2001), 'The anatomy of the grid: Enabling scalable virtual organizations', *The International Journal of High Performance Computing Applications* **15**(3), 200–222.

Frachtenberg, E. and Feitelson, D. G. (2005), Pitfalls in parallel job scheduling evaluation, *in* '11th Workshop on Job Scheduling Strategies for Parallel processing', Cambridge, MA, pp. 257–282.

Freeh, V. W., Pan, F., Kappiah, N., Lowenthal, D. K. and Springer, R. (2005), Exploring the energy-time tradeoff in MPI programs on a power-scalable cluster, *in* 'IPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)', IEEE Computer Society, Washington, DC, USA, p. 4a.

Fu, Y., Chase, J., Chun, B., Schwab, S. and Vahdat, A. (2003), Sharp: an architecture for secure resource peering, *in* 'SOSP '03: Proceedings of the nineteenth ACM symposium on Operating systems principles', ACM Press, New York, NY, USA, pp. 133–148.

Ge, R., Fend, X., chun Feng, W. and Cameron, K. W. (2007), Cpu miser: A performance-directed, run-time system for power-aware cluster, *in* 'International Conference on Parallel Processing (ICPP07)', p. 18.

Germain-Renaud, C., Loomis, C., Mościcki, J. T. and Texier, R. (2008), 'Scheduling for responsive grids', *Journal of Grid Computing* **6**(1), 15–27.

Gilbert, N. and Terna, P. (2000), 'How to build and use agent-based models in social science', *Mind & Society* **1**(1), 57–72.

Gode, D. K. and Sunder, S. (1993), 'Allocative efficiency of markets with zero-intelligence traders: Market as a partial substitute for individual rationality', *The Journal of Political Economy* **101**(1), 119–138.

Goldstein, H. (2007), 'Cure for the multicore blues', *IEEE Spectrum* **44**, 37–39.

Gropp, W., Lusk, E. and Sterling, T., eds (2003), *Beowulf Cluster Computing with Linux*, Scientific and Engineering Computation Series, second edn, MIT Press, London, UK.

Grosu, D. and Das, A. (2004), Auction-based resource allocation protocols in grids, *in* 'Proceedings of the 16th International Conference of Parallel and Distributed Computing and Systems', pp. 20–27.

Hajiaghayi, M. T. (2005), Online auctions with re-usable goods, *in* 'EC '05: Proceedings of the 6th ACM conference on electronic commerce', ACM Press, New York, NY, USA, pp. 165–174.

Harada, F., Ushio, T. and Nakamoto, Y. (2006), Power-aware resource allocation with fair QoS guarantee, *in* 'RTCSA '06: Proceedings of the 12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications', IEEE Computer Society, Washington, DC, USA, pp. 287–293.

Hargrove, W. W., Hoffman, F. M., Sterling, T. and Kay, C. (2001), 'The do-it-yourself supercomputer', *Scientific American* **285**(2), 72–80.

Heath, T., Diniz, B., Carrera, E. V., Meira Jr., W. and Bianchini, R. (2005), Energy conservation in heterogeneous server clusters, *in* 'PPoPP '05: Proceedings of the tenth ACM SIGPLAN symposium on principles and practice of parallel programming', ACM, New York, NY, USA, pp. 186–195.

Herbert, R. D. and Turton, P. (2007), Simple strategies of agents in an evolving auction model, *in* L. Oxley and D. Kulasiri, eds, 'MODSIM 2007 International Congress on

Modelling and Simulation', Modelling and Simulation Society of Australia and New Zealand, pp. 1103–1109.

HIOKI E. E. Corporation (2006), *3197 power quality analyzer Instruction Manual*, Ueda, Nagano, Japan.

Hsu, C. and Feng, W. (2005*a*), A Feasibility Analysis of Power Awareness in Commodity-Based High-Performance Clusters, *in* '7th IEEE International Conference on Cluster Computing (CLUSTER'05)', Boston, Massachusetts.

Hsu, C. and Feng, W. (2005*b*), A power-aware run-time system for high-performance computing, *in* 'ACM/IEEE SC2005: The International Conference on High-Performance Computing, Networking, and Storage', Seattle, Washington.

Hsu, C., Feng, W. and Archuleta, J. S. (2005), Towards efficient supercomputing: a quest for the right metric, *in* '1st IEEE Workshop on High-Performance, Power-Aware Computing (in conjunction with the 19th International Parallel & Distributed Processing Symposium)', Denver, Colorado.

Huang, J., Jin, H., Xie, X. and Zhang, Q. (2005), An approach to grid scheduling optimization based on fuzzy association rule mining, *in* 'e-Science and Grid Computing, 2005. First International Conference on'.

Inc, A. (2009), 'Xserve environmental report', Retrieved March 16, 2011, from `http://images.apple.com/xserve/pdf/Xserve-Environmental-Report.pdf`.

Inc, A. (2010), 'Xserve environmental report', Retrieved March 16, 2011, from `http://images.apple.com/environment/reports/docs/21.5inch_iMac_Product_Environmental_Report_20100727.pdf`.

Khan, S. U. (2009), A game theoretical energy efficient resource allocation technique for large distributed computing systems, *in* 'The 2009 International Conference on Parallel

and Distributed Processing Techniques and Applications', CSREA Press, Las Vegas, NV, USA.

Khargharia, B., Hariri, S. and Yousif, M. S. (2008), 'Autonomic power and performance management for computing systems', *Cluster Computing* **11**(2), 167–181.

Krawczyk, S. and Bubendorfer, K. (2008), Grid resource allocation : Allocation mechanisms and utilisation patterns, *in* W. Kelly and P. Roe, eds, 'Sixth Australasian Symposium on Grid Computing and e-Research (AusGrid 2008)', Vol. 82 of *CRPIT*, ACS, Wollongong, NSW, Australia, pp. 73–81.

Krikke, J. (2008), 'Recycling e-waste: The sky is the limit', *IT Professional* **10**(1), 50–55.

Kwaśnicki, W. (1999), *Computational Techniques for Modelling Learning in Economics*, Vol. 11 of *Advances in Computational Economics*, Kluwer Academic Publishers, Boston, MA, USA, chapter Evolutionary Economics and Simulation, pp. 3–45.

Lai, K., Rasmusson, L., Adar, E., Sorkin, S., Zhang, L. and Huberman, B. A. (2005), 'Tycoon: an implementation of a distributed, market-based resource allocation system', *Multiagent and Grid Systems* **1**(3), 169–182.

Leon, Y. I., Takadama, K., Nawa, N. E., Shimohara, K. and Katai, O. (2003), 'Analyzing the agent-based model and its implications', *Advances In Complex Systems* **6**(3), 331–347.

Li, C. and Li, L. (2004), 'Competitive proportional resource allocation policy for computational grid', *Future Generation Computer Systems* **20**(6), 1041–1054.

Li, J., Tian, B., Liu, T., Liu, H., Wen, X. and Honda, S. (2006), 'Status quo of e-waste management in mainland China', *Journal of Material Cycles and Waste Management* **8**(1), 13–20.

Loreto, D., Nordlander, E. and Oliner, A. (2005), 'Benchmarking a large-scale heterogeneous cluster', Retrieved June 20, 2008, from `http://beowulf.lcs.mit.edu/18.337-2005/projects/top500writeup.pdf`.

Lu, L.-T., Wernick, I. K., Hsiao, T.-Y., Yu, Y.-H., Yang, Y.-M. and Ma, H.-W. (2006), 'Balancing the life cycle impacts of notebook computers: Taiwan's experience', *Resources, Conservation and Recycling* **48**(1), 13–25.

Lubin, B., Kephart, J. O., Das, R. and Parkes, D. C. (2009), Expressive power-based resource allocation for data centers, *in* 'Proceedings of the 21st international joint conference on Artificial intelligence (IJCAI-09)', Morgan Kaufmann Publishers Inc., pp. 1451–1456.

Lynar, T., Herbert, R. D., Simon and Chivers, W. J. (2009), Impact of node ranking on outcomes of grid resource allocation, *in* H. R. Arabnia and G. A. Gravvanis, eds, 'The 2009 International Conference on Grid Computing and Applications (GCA'09)', Monte Carlo Resort, Las Vegas, NV, USA, pp. 17–22.

Lynar, T. M. and Herbert, R. D. (2009), Allocating grid resources for speed and energy conservation, *in* 'The 6th International Conference on Information Technology and Applications (ICITA09)'.

Lynar, T. M., Herbert, R. D. and Chivers, W. J. (2007), Implementing an agent based auction model on a cluster of inexpensive heterogenous workstations, *in* L. Oxley and D. Kulasiri, eds, 'MODSIM 2007 International Congress on Modelling and Simulation', Modelling and Simulation Society of Australia and New Zealand, pp. 1947–1953.

Lynar, T. M., Herbert, R. D., Simon and Chivers, W. J. (2010), 'Clustering obsolete computers to reduce e-waste', *International Journal of Information Systems and Social Change* **1**(1), 1–10.

Lynar, T. M., Herbert, R. D., Simon and Chivers, W. J. (2011), 'Resource allocation to conserve energy in distributed computing', *the International Journal of Grid and Utility Computing (IJGUC)* **2**(1), 1–10.

Lynar, T., Simon, Herbert, R. D. and Chivers, W. J. (2010), Reducing grid energy consumption through choice of resource allocation method, *in* 'The sixth Workshop on High-Performance, Power-Aware Computing (HPPAC)'.

McClave, J. T. and Dietrich, II, F. H. (1991), *Statistics*, 5th edn, Maxwell Macmillan International, San Francisco, CA, USA.

McFadzean, D., Stewart, D. and Tesfatsion, L. (2001), 'A computational laboratory for evolutionary trade networks', *IEEE Transactions on Evolutionary Computation* **5**(5), 546–560.

McFadzean, D. and Tesfatsion, L. (1997), *Evolutionary Programming VI*, Vol. 1213-1997 of *Lecture Notes in Computer Science*, Springer Berlin and Heidelberg, chapter An Agent-Based Computational Model for the Evolution of Trade Networks, pp. 73–83.

McFadzean, D. and Tesfatsion, L. (1999), 'A C++ platform for the evolution of trade networks', *Computational Economics* **14**, 109–134.

Mizuno, M. and Nishiyama, N. (2003), 'Interacting TV viewers: A case of empirical agent-based modeling and simulation for business application', *Advances in Complex Systems* **6**(3), 361–373.

Moore, G. E. (1965), 'Cramming more components onto integrated circuits', *Electronics* **38**(8).

Negri, A., Poggi, A. and Tomaiuolo, M. (2005), Intelligent task composition and allocation through agents, *in* 'Enabling Technologies: Infrastructure for Collaborative Enterprise, 2005. 14th IEEE International Workshops on', pp. 255–260.

Patel, C., Sharma, R., Bash, C. and Graupner, S. (2002), Energy aware grid: Global workload placement based on energy efficiency, Technical Report HPL-2002-329, Hewlett Packard, HP Laboratories Palo Alto.

Petitet, A., Whaley, R. C., Dongarra, J. and Cleary, A. (2008), 'HPL - A portable implementation of the high-performance linpack benchmark for distributed-memory computers', Retrieved October 2, 2008, from `http://www.netlib.org/benchmark/hpl/`.

Pinheiro, E., Bianchini, R., Carrera, E. and Heath, T. (2001), Load Balancing and Unbalancing for Power and Performance in Cluster-Based Systems, *in* 'Proceedings of the Workshop on Compilers and Operating Systems for Low Power (COLP'01)'.

Plaszczak, P. and Wellner Jr., R. (2006), *Grid computing: the savvy manager's guide*, Elsevier, London, UK.

Porter, R. (2004), Mechanism design for online real-time scheduling, *in* 'EC '04: Proceedings of the 5th ACM conference on electronic commerce', ACM Press, New York, NY, USA, pp. 61–70.

Power Tech Plus (2008), 'Multifunction energy meter instructions'.

Rusu, C., Ferreira, A., Scordino, C. and Watson, A. (2006), Energy-efficient real-time heterogeneous server clusters, *in* 'RTAS '06: Proceedings of the 12th IEEE Real-Time and Embedded Technology and Applications Symposium', IEEE Computer Society, Washington, DC, USA, pp. 418–428.

Sall, J., Creighton, L. and Lehman, A. (2005), *JMP Start Statistics: A Guide to Statistics and Data Analysis Using JMP and JMP IN Software*, 3rd edn, Thomsom, Belmont, CA, USA.

Schmidt, C. W. (2006), 'Unfair trade e-waste in Africa', *Environmental Health Perspectives* **114**(4), A232–A235.

Schneck, P. B. (1990), 'Supercomputers', *Annual Reviews Computer Science* **4**, 13–36.

Scriven, I., Lewis, A., Smith, M. and Friese, T. (2008), Resource evaluation and node monitoring in service oriented ad-hoc grids, *in* W. Kelly and P. Roe, eds, 'Sixth Australasian Symposium on Grid Computing and e-Research (AusGrid 2008)', Vol. 82 of *CRPIT*, ACS, Wollongong, NSW, Australia, pp. 65–71.

Shah, A. J. and Krishnan, N. (2008), 'Optimization of global data center thermal management workload for minimal environmental and economic burden', *IEEE Transactions on Components and Packaging Technologies* **31**(1), 39–45.

Software in the Public Interest Inc (2007), 'Debian – the universal operating system', Retrieved May 28, 2007, from `http://www.debian.org/`.

Spooner, D., Jarvis, S., Cao, J., Saini, S. and Nudd, G. (2003), 'Local grid scheduling techniques using performance prediction', *IEE Proceedings Computers and Digital Techniques* **150**(2), 87–96.

Springer, R., Lowenthal, D. K., Rountree, B. and Freeh, V. W. (2006), Minimizing execution time in MPI programs on an energy-constrained, power-scalable cluster, *in* 'PPoPP '06: Proceedings of the eleventh ACM SIGPLAN symposium on principles and practice of parallel programming', ACM, New York, NY, USA, pp. 230–238.

Sterling, T., Salmon, J., Becker, D. and Savarese, D. F. (1999), *How to Build a Beowulf*, MIT Press, London, UK.

Streicher-Porte, M., Widmer, R., Jain, A., Bader, H.-P., Scheidegger, R. and Kytzia, S. (2005), 'Key drivers of the e-waste recycling system: Assessing and modelling e-waste processing in the informal sector in Delhi', *Environmental Impact Assessment Review* **25**(5), 472–491.

Stuer, G., Vanmechelen, K. and Broeckhove, J. (2007), 'A commodity market algorithm for pricing substitutable grid resources', *Future Generation Computer Systems* **23**(5), 688–701.

Subrata, R., Zomaya, A. Y. and Landfeldt, B. (2010), 'Cooperative power-aware scheduling in grid computing environments', *Journal of Parallel and Distributed Computing* **70**(2), 84–91.

Tan, Z. and Gurd, J. (2007), Market-based grid resource allocation using a stable continuous double auction, *in* 'Grid Computing, 2007 8th IEEE/ACM International Conference on', pp. 283–290.

Tarricone, L. and Esposito, A. (2004), *Grid computing for electromagnetics*, Artech House, Boston, MA, USA.

Terazono, A., Murakami, S., Abe, N., Inanc, B., Moriguchi, Y., ichi Sakai, S., Kojima, M., Yoshida, A., Li, J., Yang, J., Wong, M. H., Jain, A., Kim, I.-S., Peralta, G. L., Lin, C.-C., Mungcharoen, T. and Williams, E. (2006), 'Current status and research on e-waste issues in Asia', *Journal of Material Cycles and Waste Management* **8**(1), 1–12.

Tesfatsion, L. (2001), 'Introduction', *Computational Economics* **18**(1), 1–8.

Tesfatsion, L. (2002), 'Agent-based computational economics: Modelling economies as complex adaptive systems', *Information Sciences* **149**(4), 262–268.

Tong, X. (2004), Global mandate, national policies, and local responses: scale conflicts in China's management of imported e-waste, *in* '2004 IEEE International Symposium on Electronics and the Environment, 2004. Conference Record.', pp. 204–207.

Top500.org (2007), 'Top500 supercomputing sites', Retrieved June 20, 2008, from `http://www.top500.org`.

Vahdat, A., Lebeck, A. and Ellis, C. S. (2000), Every joule is precious: The case for revisiting operating system design for energy efficiency, *in* 'EW 9: Proceedings of the 9th workshop on ACM SIGOPS European workshop', ACM, New York, NY, USA, pp. 31–36.

Velte, T. J., Velte, A. and Elsenpeter, R. (2008), *Green IT: Reduce your information system's environmental impact while adding to the bottom line*, McGraw-Hill, New York, NY, USA.

Verma, A., Ahuja, P. and Neogi, A. (2008), Power-aware dynamic placement of HPC applications, *in* 'ICS '08: Proceedings of the 22nd annual international conference on Supercomputing', ACM, New York, NY, USA, pp. 175–184.

Wang, P., Turner, G., Lauer, D., Allen, M., Simms, S., Hart, D., Papakhian, M. and Stewart, C. (2004), LINPACK performance on a geographically distributed Linux cluster, *in* 'Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International', pp. 245–250.

Weicker, R. P. (1990), 'An overview of common benchmarks', *Computer* **23**(12), 65–75.

Wells, A. J. (2008), *Grid application systems design*, Taylor & Francis Group, Boca Raton, FL, USA.

Williams, E., Kahhat, R., Allenby, B., Kavazanjian, E., Kim, J. and Xu, M. (2008), 'Environmental, social, and economic implications of global reuse and recycling of personal computers', *Environmental Science and Technology* **42**(17), 64466454.

Williams, E., Kahhat, R., Allenby, B., Kavazanjian, E., Xu, M. and Kim, J. (2008), Sustainability review of the international reverse chain for reuse and recycling of computers, *in* 'IEEE International Symposium onElectronics and the Environment, 2008. ISEE 2008', pp. 1–6.

Wooldridge, M. (2002), *An Introduction to MultiAgent Systems*, John Wiley & Sons Ltd, West Sussex, England.

Xiao, L., Zhu, Y., Ni, L. M. and Xu, Z. (2005), Gridis: An incentive-based grid scheduling, *in* 'Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International'.

Yamamoto, H., Kawahara, K., Takine, T. and Oie, Y. (2006), 'Performance comparison of task allocation schemes depending upon resource availability in a grid computing environment', *IEICE Transactions on Information and Systems* **E89-D**(2), 459–468.

Zhao, J., Chen, D., Tian, Z. and Zhai, Z. (2006), A novel auction-based grid resource allocation algorithm, *in* 'Pervasive Computing and Applications, 2006 1st International Symposium on', pp. 269–272.

Appendix A

ITEMS ON THE ATTACHED MEDIA

The attached media contains a number of related materials that can also be obtained by contacting the author.

A.1 Simulation models

The source code of the simulation models, and the accompanying scripts, used in this research is included on the attached media.

A.2 Error measurements

Error calculation for the measurements taken by the Hioki power quality analyser 3197 can be found on the attached media in the file `errorCalcs.xls`.

A.3 Prime number script

The original script was created by Mike Golvach and can be obtained from `http://linuxshellaccount.blogspot.com/2008/06/shell-script-to-produce-prime-numbers.html`. The script used in this research is modified from the original and is available on the attached media and printed below.

```
#!/bin/bash
#
# primes.sh - find all prime numbers up to a certain number
# 2008 - Mike Golvach - eggi@comcast.net
#
# Creative Commons Attribution-Noncommercial-Share Alike 3.0 United States License
#
echo Start `date +%H:%M:%S` >> /home/user/time.out
factorial () {
 local factorial_count=$1
 if [ "$factorial_count" -eq 0 ]
 then
  factorial_count=1
 fi
 for (( factor=$((factorial_count -1)); $factor >= 1; --factor ))
 do
  factorial_count=$(($factorial_count * $factor))
 done
 echo $factorial_count
}

prime_number () {
 local prime=$1
 p_minus_1=$(($prime - 1))
 fact_p_minus_1=`factorial "$p_minus_1"`
 fact_plus_1=$(($fact_p_minus_1 + 1))
 echo $fact_plus_1
}

highest_number=$1

if [ -z $highest_number ]
then
 echo
 echo "Usage: $0 highestNumber"
 echo
 exit 1
fi

if [ $highest_number -eq 0 ]
then
 echo
 echo "Sorry.  0 is not a prime number"
 echo
 exit 0
elif [ $highest_number -eq 1 ]
then
 echo
 echo "Sorry.  0 and 1 are not prime numbers"
 echo
 exit 0
fi

echo "Generating Prime Numbers Up To $highest_number"
if [ $highest_number -eq 2 ]
then
 echo
```

```
 echo -n "2"
else
 echo
 echo -n "2 3 "
fi

count=4
while [ $count -le $highest_number ]
do
 prime_return=`prime_number "$count"`
 prime_test=$(($prime_return % count))
 if [ $prime_test -eq 0 ]
 then
  echo -n "$count "
 fi
 count=$(($count + 1))
for wa in {1..800}
        do q=$wa*$wa-1
        for ii in $q
                do zz=ii*$q-1
        done
done
done

echo
echo
echo "All Set!"
echo
echo doneIt `date +%H:%M:%S` >> /home/user/time.out
exit 0
```

A.4 Extended resource allocation model results

The resulting output from the experiments presented in Chapter 4 are included on the attached media in the file `APXPOPOresults.csv'`.

Appendix B

RAW STATISTICAL OUTPUT

All raw statistical output is available on the attached media.

B.0.1 Statistical results from the comparison of the initial cluster to the new node

The file `E-waste-RAW-ANOVA.eps` contains the raw statistical output of tests performed in Chapter 5.

B.0.2 Statistical results from the Model extension and execution chapter

The file `CH8-WilcoxonEachPair.eps` contains raw statistical output for the tests performed in Chapter 8.

Appendix C

EXTENDED STATISTICAL OUTPUT FOR THE MODEL CALIBRATION

This appendix contains extended statistical output for Chapter 7. Table C.1 contains the Tukey HSD connecting letters report for energy and Table C.2 contains the Tukey HSD connecting letters report for time taken. The letter reports groups not connected by the same letter are significantly different from one another.

The ordered differences report for energy is included on the attached media with the file name `EnergyFullCLandPvalues.csv`.

The ordered differences report for time taken is included on the attached media with the file name `TimeFullCLandPvalues.csv`.

Table C.1: Tukey HSD connecting letters report for energy, groups not connected by the same letter are significantly different

WF	Mech	Environment	Simulated														
1	BA	Cluster	No														N
1	BA	Cluster	Yes														N
1	CRA	Cluster	Yes														N
1	CRA	Cluster	No														N
1	PPBA	Cluster	No														N
1	PPBA	Cluster	Yes														N
2	BA	Cluster	No											K	L	M	
2	BA	Cluster	Yes												L	M	
2	CRA	Cluster	No									I	J				
2	CRA	Cluster	Yes													M	
2	PPBA	Cluster	Yes										J	K	L	M	
2	PPBA	Cluster	No										J	K	L	M	
3	BA	Cluster	No									I	J	K	L		
3	BA	Cluster	Yes											K	L	M	
3	CRA	Cluster	No									I					
3	CRA	Cluster	Yes													M	
3	PPBA	Cluster	No									I	J	K			
3	PPBA	Cluster	Yes											K	L	M	
1	BA	Grid	Yes						F	G	H						
1	BA	Grid	No							G	H						
1	CRA	Grid	No				D	E	F								
1	CRA	Grid	Yes					E	F	G	H						
1	PPBA	Grid	Yes						F	G	H						
1	PPBA	Grid	No								H						
2	BA	Grid	Yes				D	E									
2	BA	Grid	No				D	E	F	G							
2	CRA	Grid	Yes	A	B												
2	CRA	Grid	No			C	D										
2	PPBA	Grid	Yes				D	E									
2	PPBA	Grid	No				D	E									
3	BA	Grid	No		B	C											
3	BA	Grid	Yes				D										
3	CRA	Grid	No	A													
3	CRA	Grid	Yes	A													
3	PPBA	Grid	No			C	D										
3	PPBA	Grid	Yes				D										

Table C.2: Tukey HSD connecting letters report for time, groups not connected by the same letter are significantly different

WF	Mech	Environment	Simulated													
1	BA	Cluster	No											K	L	M
1	BA	Cluster	Yes												L	M
1	CRA	Cluster	Yes												L	M
1	CRA	Cluster	No												L	M
1	PPBA	Cluster	No										J	K	L	M
1	PPBA	Cluster	Yes												L	M
2	BA	Cluster	No				D	E	F	G	H					
2	BA	Cluster	Yes					E	F	G	H	I				
2	CRA	Cluster	No	A	B											
2	CRA	Cluster	Yes							G	H	I	J	K	L	M
2	PPBA	Cluster	Yes			C	D	E	F							
2	PPBA	Cluster	No				D	E	F	G						
3	BA	Cluster	No		B	C	D									
3	BA	Cluster	Yes				D	E	F	G						
3	CRA	Cluster	No	A												
3	CRA	Cluster	Yes							G	H	I	J	K	L	M
3	PPBA	Cluster	No	A	B	C	D									
3	PPBA	Cluster	Yes				D	E	F	G	H					
1	BA	Grid	Yes												L	M
1	BA	Grid	No												L	M
1	CRA	Grid	No						F	G	H	I	J	K	L	
1	CRA	Grid	Yes												L	M
1	PPBA	Grid	Yes												L	M
1	PPBA	Grid	No													M
2	BA	Grid	No										J	K	L	M
2	BA	Grid	Yes										J	K	L	M
2	CRA	Grid	Yes				D	E	F	G	H					
2	CRA	Grid	No					E	F	G	H	I	J	K		
2	PPBA	Grid	No								H	I	J	K	L	M
2	PPBA	Grid	Yes										J	K	L	M
3	BA	Grid	No			C	D	E								
3	BA	Grid	Yes									I	J	K	L	M
3	CRA	Grid	No	A	B	C										
3	CRA	Grid	Yes	A	B	C	D									
3	PPBA	Grid	No					E	F	G	H	I	J			
3	PPBA	Grid	Yes									I	J	K	L	M

Appendix D

RAW CLUSTER AND GRID TEST RESULTS

This section of the appendix presents the raw results from the cluster and grid tests conducted in Chapter 6. All data is presented in Table D.1.

Table D.1: Raw cluster and grid test results

Workflow	Mechanism	Energy used (kWh)	Time Taken	Environment	Performance Metric
1	Batch	0.201	601	Grid	Eflops
1	Batch	0.121	361	Grid	Eflops
1	Batch	0.203	601	Grid	Eflops
1	Batch	0.203	601	Grid	Eflops
1	Batch	0.204	601	Grid	Eflops
1	Batch	0.202	601	Grid	Eflops
1	Batch	0.203	601	Grid	Eflops
1	Batch	0.202	601	Grid	Eflops
1	Batch	0.202	601	Grid	Eflops
1	Batch	0.203	601	Grid	Eflops
2	Batch	0.25	659	Grid	Eflops
2	Batch	0.2	659	Grid	Eflops
2	Batch	0.244	653	Grid	Eflops
2	Batch	0.248	660	Grid	Eflops
2	Batch	0.249	660	Grid	Eflops
2	Batch	0.249	660	Grid	Eflops
2	Batch	0.248	603	Grid	Eflops
2	Batch	0.249	661	Grid	Eflops
2	Batch	0.249	659	Grid	Eflops
2	Batch	0.249	659	Grid	Eflops
3	Batch	0.341	911	Grid	Eflops
3	Batch	0.328	880	Grid	Eflops

Table D.1: Raw cluster and grid test results continued

Workflow	Mechanism	Energy used (kWh)	Time Taken	Environment	Performance Metric
3	Batch	0.361	974	Grid	Eflops
3	Batch	0.315	911	Grid	Eflops
3	Batch	0.285	749	Grid	Eflops
3	Batch	0.342	910	Grid	Eflops
3	Batch	0.343	912	Grid	Eflops
3	Batch	0.339	899	Grid	Eflops
3	Batch	0.336	893	Grid	Eflops
3	Batch	0.339	899	Grid	Eflops
1	CDA	0.201	599	Grid	Eflops
1	CDA	0.204	607	Grid	Eflops
1	CDA	0.202	600	Grid	Eflops
1	CDA	0.203	601	Grid	Eflops
1	CDA	0.356	1067	Grid	Eflops
1	CDA	0.47	1442	Grid	Eflops
1	CDA	0.199	597	Grid	Eflops
1	CDA	0.387	1764	Grid	Eflops
1	CDA	0.202	601	Grid	Eflops
1	CDA	0.286	844	Grid	Eflops
2	CDA	0.242	641	Grid	Eflops
2	CDA	0.363	975	Grid	Eflops
2	CDA	0.441	1183	Grid	Eflops
2	CDA	0.21	556	Grid	Eflops
2	CDA	0.269	695	Grid	Eflops
2	CDA	0.255	663	Grid	Eflops
2	CDA	0.312	814	Grid	Eflops
2	CDA	0.294	769	Grid	Eflops
2	CDA	0.244	658	Grid	Eflops
2	CDA	0.283	746	Grid	Eflops
3	CDA	0.353	993	Grid	Eflops
3	CDA	0.662	1705	Grid	Eflops
3	CDA	0.739	2118	Grid	Eflops
3	CDA	0.441	1206	Grid	Eflops
3	CDA	0.697	1845	Grid	Eflops
3	CDA	0.764	2095	Grid	Eflops
3	CDA	0.607	1638	Grid	Eflops

Table D.1: Raw cluster and grid test results continued

Workflow	Mechanism	Energy used (kWh)	Time Taken	Environment	Performance Metric
3	CDA	0.622	1739	Grid	Eflops
3	CDA	0.216	522	Grid	Eflops
3	CDA	0.214	535	Grid	Eflops
1	PPBA	0.200	602	Grid	Eflops
1	PPBA	0.199	601	Grid	Eflops
1	PPBA	0.198	601	Grid	Eflops
1	PPBA	0.199	602	Grid	Eflops
1	PPBA	0.199	601	Grid	Eflops
1	PPBA	0.199	601	Grid	Eflops
1	PPBA	0.200	601	Grid	Eflops
1	PPBA	0.199	601	Grid	Eflops
1	PPBA	0.200	601	Grid	Eflops
1	PPBA	0.199	601	Grid	Eflops
2	PPBA	0.231	620	Grid	Eflops
2	PPBA	0.258	705	Grid	Eflops
2	PPBA	0.258	719	Grid	Eflops
2	PPBA	0.266	730	Grid	Eflops
2	PPBA	0.263	721	Grid	Eflops
2	PPBA	0.253	686	Grid	Eflops
2	PPBA	0.251	683	Grid	Eflops
2	PPBA	0.231	622	Grid	Eflops
2	PPBA	0.272	730	Grid	Eflops
2	PPBA	0.264	719	Grid	Eflops
3	PPBA	0.374	1005	Grid	Eflops
3	PPBA	0.212	550	Grid	Eflops
3	PPBA	0.221	576	Grid	Eflops
3	PPBA	0.221	575	Grid	Eflops
3	PPBA	0.262	700	Grid	Eflops
3	PPBA	0.257	688	Grid	Eflops
3	PPBA	0.345	952	Grid	Eflops
3	PPBA	0.373	1040	Grid	Eflops
3	PPBA	0.319	882	Grid	Eflops
3	PPBA	0.292	805	Grid	Eflops
1	BATCH	0.053	624	Cluster	Eflops
1	BATCH	0.053	624	Cluster	Eflops

Table D.1: Raw cluster and grid test results continued

Workflow	Mechanism	Energy used (kWh)	Time Taken	Environment	Performance Metric
1	BATCH	0.053	625	Cluster	Eflops
1	BATCH	0.053	624	Cluster	Eflops
1	BATCH	0.053	624	Cluster	Eflops
1	BATCH	0.053	624	Cluster	Eflops
1	BATCH	0.053	624	Cluster	Eflops
1	BATCH	0.053	624	Cluster	Eflops
1	BATCH	0.053	624	Cluster	Eflops
1	BATCH	0.053	625	Cluster	Eflops
2	BATCH	0.101	832	Cluster	Eflops
2	BATCH	0.101	829	Cluster	Eflops
2	BATCH	0.102	833	Cluster	Eflops
2	BATCH	0.101	831	Cluster	Eflops
2	BATCH	0.101	831	Cluster	Eflops
2	BATCH	0.101	834	Cluster	Eflops
2	BATCH	0.102	831	Cluster	Eflops
2	BATCH	0.1	817	Cluster	Eflops
2	BATCH	0.101	831	Cluster	Eflops
2	BATCH	0.101	832	Cluster	Eflops
1	CDA	0.052	612	Cluster	Eflops
1	CDA	0.051	611	Cluster	Eflops
1	CDA	0.052	611	Cluster	Eflops
1	CDA	0.052	611	Cluster	Eflops
1	CDA	0.051	609	Cluster	Eflops
1	CDA	0.051	611	Cluster	Eflops
1	CDA	0.052	611	Cluster	Eflops
1	CDA	0.052	614	Cluster	Eflops
1	CDA	0.052	613	Cluster	Eflops
1	CDA	0.051	610	Cluster	Eflops
2	CDA	0.126	1152	Cluster	Eflops
2	CDA	0.121	1097	Cluster	Eflops
2	CDA	0.127	1162	Cluster	Eflops
2	CDA	0.121	1091	Cluster	Eflops
2	CDA	0.123	1103	Cluster	Eflops
2	CDA	0.121	1091	Cluster	Eflops
2	CDA	0.127	1143	Cluster	Eflops

Table D.1: Raw cluster and grid test results continued

Workflow	Mechanism	Energy used (kWh)	Time Taken	Environment	Performance Metric
2	CDA	0.119	1072	Cluster	Eflops
2	CDA	0.124	1124	Cluster	Eflops
2	CDA	0.122	1106	Cluster	Eflops
1	PPBA	0.053	625	Cluster	Eflops
1	PPBA	0.053	626	Cluster	Eflops
1	PPBA	0.053	625	Cluster	Eflops
1	PPBA	0.053	626	Cluster	Eflops
1	PPBA	0.053	626	Cluster	Eflops
1	PPBA	0.053	626	Cluster	Eflops
1	PPBA	0.053	626	Cluster	Eflops
1	PPBA	0.053	626	Cluster	Eflops
1	PPBA	0.053	626	Cluster	Eflops
1	PPBA	0.053	625	Cluster	Eflops
2	PPBA	0.106	884	Cluster	Eflops
2	PPBA	0.101	819	Cluster	Eflops
2	PPBA	0.104	851	Cluster	Eflops
2	PPBA	0.106	883	Cluster	Eflops
2	PPBA	0.105	854	Cluster	Eflops
2	PPBA	0.107	887	Cluster	Eflops
2	PPBA	0.1	805	Cluster	Eflops
2	PPBA	0.103	831	Cluster	Eflops
2	PPBA	0.107	890	Cluster	Eflops
2	PPBA	0.102	834	Cluster	Eflops
3	BATCH	0.107	877	Cluster	Eflops
3	BATCH	0.109	892	Cluster	Eflops
3	BATCH	0.12	1051	Cluster	Eflops
3	BATCH	0.122	1057	Cluster	Eflops
3	BATCH	0.107	879	Cluster	Eflops
3	BATCH	0.121	1050	Cluster	Eflops
3	BATCH	0.11	897	Cluster	Eflops
3	BATCH	0.121	1025	Cluster	Eflops
3	BATCH	0.124	1068	Cluster	Eflops
3	BATCH	0.125	1107	Cluster	Eflops
3	CDA	0.161	1435	Cluster	Eflops
3	CDA	0.135	1196	Cluster	Eflops

Table D.1: Raw cluster and grid test results continued

Workflow	Mechanism	Energy used (kWh)	Time Taken	Environment	Performance Metric
3	CDA	0.128	1081	Cluster	Eflops
3	CDA	0.145	1308	Cluster	Eflops
3	CDA	0.13	1120	Cluster	Eflops
3	CDA	0.132	1139	Cluster	Eflops
3	CDA	0.146	1310	Cluster	Eflops
3	CDA	0.138	1200	Cluster	Eflops
3	CDA	0.134	1168	Cluster	Eflops
3	CDA	0.133	1148	Cluster	Eflops
3	PPBA	0.112	906	Cluster	Eflops
3	PPBA	0.119	994	Cluster	Eflops
3	PPBA	0.123	1039	Cluster	Eflops
3	PPBA	0.127	1084	Cluster	Eflops
3	PPBA	0.12	1021	Cluster	Eflops
3	PPBA	0.11	899	Cluster	Eflops
3	PPBA	0.114	950	Cluster	Eflops
3	PPBA	0.114	952	Cluster	Eflops
3	PPBA	0.132	1168	Cluster	Eflops
3	PPBA	0.119	1025	Cluster	Eflops
2	BATCH	0.105	852	Cluster	ETTT
2	BATCH	0.104	846	Cluster	ETTT
2	BATCH	0.104	850	Cluster	ETTT
2	BATCH	0.104	844	Cluster	ETTT
2	BATCH	0.106	864	Cluster	ETTT
2	BATCH	0.107	870	Cluster	ETTT
2	BATCH	0.105	848	Cluster	ETTT
2	BATCH	0.106	859	Cluster	ETTT
2	BATCH	0.106	851	Cluster	ETTT
2	BATCH	0.106	858	Cluster	ETTT
2	BATCH	0.107	870	Cluster	Power
2	BATCH	0.109	872	Cluster	Power
2	BATCH	0.107	867	Cluster	Power
2	BATCH	0.108	869	Cluster	Power
2	BATCH	0.108	867	Cluster	Power
2	BATCH	0.106	852	Cluster	Power
2	BATCH	0.107	863	Cluster	Power

Table D.1: Raw cluster and grid test results continued

Workflow	Mechanism	Energy used (kWh)	Time Taken	Environment	Performance Metric
2	BATCH	0.107	861	Cluster	Power
2	BATCH	0.107	863	Cluster	Power
2	BATCH	0.108	873	Cluster	Power

Appendix E

EXTENDED DESCRIPTION OF THE PERFORMANCE EFFICIENCY METRIC SIMULATION

The simulation model used in Chapter 3 is described in the class diagram (Figure E.1) and in pseudo-code showing the main loop (Algorithm 15), clearing of the auction (Algorithm 16), and simulated processing of tasks (Algorithm 17). The source code of this model is included on the attached media.

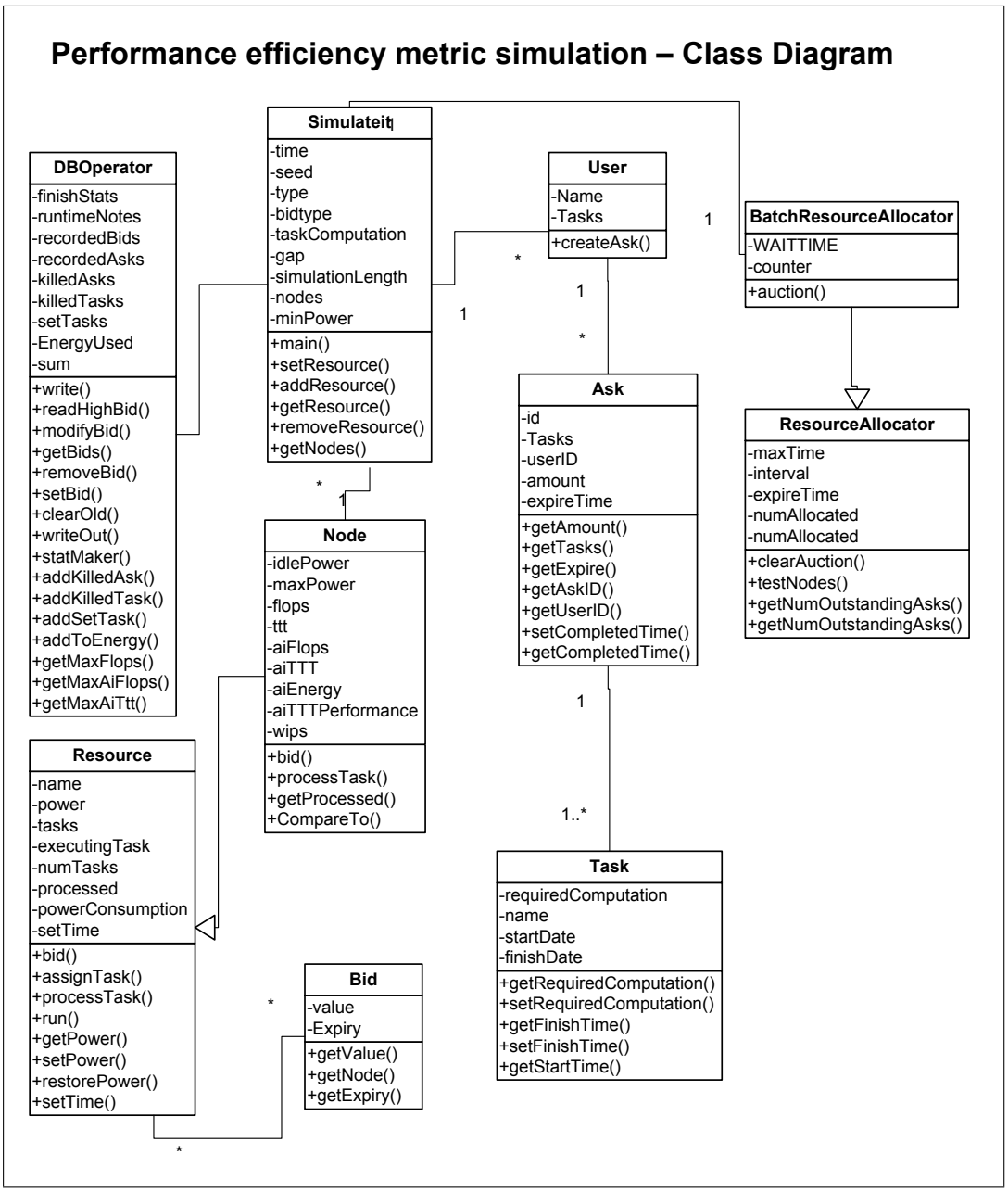

Figure E.1: Class diagram of the performance efficiency metric simulation

Algorithm 15 Analyse resources

```
Create resource allocator
Define nodes
Create user
while Time ≤ simulation length do
  if Time |gap| == 1 then
    Create an Ask
  end if
  Increment Time
end while
Allocate resources
Record statistics
for each node do
  if the node has no tasks then
    Record idle energy
  else
    Process its tasks
  end if
end for
```

Algorithm 16 Clear Auction

```
if counter == WAITTIME then
  Sort the Asks
  Sort the Bids
  if The size of asks ≠0 and The size of Bids ≠0 then
    for each Ask do
      for each Bid do
        if The value of the Bid ≥ The value of the Ask then
          Clear the matching pair
          Break the loop
        end if
      end for
    end for
  end if
end if
```

Algorithm 17 Process Task

```
if tasks > 0 then
  Record max energy
  for each task do
    required = The tasks required computation
    required = required - This nodes processing power
    The tasks required computation = required
    if required ≤ 0 then
      Remove the task
    else
      Save the task
    end if
  end for
end if
```

Appendix F

HARDWARE DETAILS

Table F.1: Node details

Node ID	CPU	Clock speed (GHZ)	Memory (MB)	L2 Cache	BIOS DATE	sSpec Number
001	P3 Coppermine	0.93	512	256k	2001	SL4C9
002	P3 Coppermine	0.93	512	256k	2001	SL4C9
003	P3 Coppermine	0.93	512	256k	2001	SL4C9
004	P3 Coppermine	0.93	512	256k	2001	SL4C9
005	P3 Coppermine	0.93	512	256k	2001	SL4C9
006	P3 Coppermine	0.93	512	256k	2001	SL4C9
007	P3 Coppermine	0.93	512	256k	2001	SL4C9
008	P3 Coppermine	0.93	512	256k	2001	SL4C9
009	P3 Coppermine	0.93	512	256k	2001	SL4C9
010	P3 Coppermine	0.93	512	256k	2001	SL4C9
011	P3 Coppermine	0.93	512	256k	2001	SL4C9
012	P3 Coppermine	0.93	512	256k	2001	SL4C9
013	P3 Coppermine	0.93	512	256k	2001	SL4C9
014	P3 Coppermine	0.93	512	256k	2001	SL4C9
015	P3 Coppermine	0.93	512	256k	2001	SL4C9
016	P3 Coppermine	0.93	512	256k	2001	SL4C9
017	P3 Coppermine	0.93	512	256k	2001	SL4C9
018	P3 Coppermine	0.93	512	256k	2001	SL4C9
019	P3 Coppermine	0.93	512	256k	2001	SL4C9
020	P3 Coppermine	0.93	512	256k	2001	SL4C9
021	P3 Coppermine	0.93	512	256k	2001	SL4C9
022	P3 Coppermine	0.93	512	256k	2001	SL4C9
023	P3 Coppermine	0.93	512	256k	2001	SL4C9
024	P3 Coppermine	0.93	512	256k	2001	SL4C9
025	P3 Coppermine	0.93	512	256k	2001	SL4C9
026	P3 Coppermine	0.93	512	256k	2001	SL4C9

Table F.1: Node details continued

Node ID	CPU	Clock speed (GHZ)	Memory (MB)	L2 Cache	BIOS DATE	sSpec Number
027	P3 Coppermine	0.93	512	256k	2001	SL4C9
028	P3 Coppermine	0.93	512	256k	2001	SL4C9
029	P3 Coppermine	0.93	512	256k	2001	SL4C9
030	P3 Coppermine	0.93	512	256k	2001	SL4C9
031	P3 Coppermine	0.93	512	256k	2001	SL4C9
032	P3 Coppermine	0.93	512	256k	2001	SL4C9
033	P3 Coppermine	0.93	512	256k	2001	SL4C9
034	P3 Coppermine	0.93	512	256k	2001	SL4C9
035	P3 Coppermine	0.93	512	256k	2001	SL4C9
036	P3 Coppermine	0.93	512	256k	2001	SL4C9
037	P3 Coppermine	0.93	512	256k	2001	SL4C9
038	P3 Coppermine	0.93	512	256k	2001	SL4C9
039	P3 Coppermine	0.93	512	256k	2001	SL4C9
040	P3 Coppermine	0.93	512	256k	2001	SL4C9
041	P3 Coppermine	0.93	512	256k	2001	SL4C9
CN1	P4 Northwood	2.4	256	512k	2002	SL4C9
CN2	P4 Northwood	2.4	256	512k	2002	SL6EF
CN3	P4 Northwood	2.4	256	512k	2002	SL6EF
CN4	P4 Northwood	2.4	256	512k	2002	SL6EF
CN5	P4 Northwood	2.4	256	512k	2002	SL6EF
CN6	P4 Northwood	2.4	256	512k	2002	SL6EF
CN7	P4 Northwood	2.4	256	512k	2002	SL6EF
CN8	P4 Northwood	2.4	256	512k	2002	SL6EF
CN9	P4 Northwood	2.4	256	512k	2002	SL6EF
CN10	P4 Northwood	2.4	256	512k	2002	SL6EF
NN1	Core 2 Duo E6550	2.33	2048	4096K	2007	–
NN7	Core 2 Duo E6550	2.33	2048	4096K	2007	–
HPN1	P4 Northwood	2.66	512	512k	2003	SL6PE
HPN2	P4 Northwood	2.66	512	512k	2003	SL6PE
HPN3	P4 Northwood	2.66	512	512k	2003	SL6PE
HPN4	P4 Northwood	2.66	512	512k	2003	SL6PE
HPN5	P4 Northwood	2.66	512	512k	2003	SL6PE
HPN6	P4 Northwood	2.66	512	512k	2003	SL6PE
HPN7	P4 Northwood	2.66	512	512k	2003	SL6PE
HPN8	P4 Northwood	2.66	512	512k	2003	SL6PE

Table F.1: Node details continued

Node ID	CPU	Clock speed (GHZ)	Memory (MB)	L2 Cache	BIOS DATE	sSpec Number
HPN9	P4 Northwood	2.66	512	512k	2003	SL6PE
HPN10	P4 Northwood	2.66	512	512k	2003	SL6PE
HPN11	P4 Northwood	2.66	512	512k	2003	SL6PE
NN2	Athlon	2.80	1024	–	–	–