School of Computing

Performance Evaluation of Virtual Machine Live Migration for Energy Efficient Large-Scale Computing

Osama Nasser Alrajeh

Submitted for the degree of Doctor of Philosophy (Integrated) in the School of Computing, Newcastle University

August 2019

Dedicated to my loving father, mother, wife and daughter...

who always inspire, encourage, and support
without whom none of this would be possible

Declaration

I declare that this thesis has not been previously submitted for a qualification or degree at Newcastle University or any other institution. I state that this thesis is my own work unless otherwise stated.

Osama Alrajeh

We confirm that, to the best of our knowledge, this thesis is from the student's own work and has been submitted with our approval.

Supervisors

Dr. Nigel Thomas

Dr. Matthew Forshaw

Acknowledgements

First and foremost, I would like to thank my supervisors Dr. Nigel Thomas and Dr. Matthew Forshaw. I am extremely grateful and indebted to them for their professional guidance, continued support, valuable advice and motivating encouragement throughout my IPhD study. They have supported me academically, morally, and spiritually from the beginning to the end of finishing this research.

Also, I would like to take this opportunity to thank my external examiner Prof William Knottenbet (Imperial College London) and my internal examiner Prof Paul Watson for their valuable discussion in the viva.

I would like to thank my friends and colleagues at the School of Computing at Newcastle University for their inspiration and motivation over the years as well as my friends in the UK, US, and back in Saudi Arabia.

I would also like to thank my home country Saudi Arabia, the Institute of Public Administration (IPA) in Riyadh and the Saudi Arabian Cultural Bureau in London for giving me the opportunity to undertake my IPhD study in Newcastle University.

Nobody has been more important to me in the pursuit of this thesis than the members of my family. I would like to thank my parents for all pray, support, love, and patience throughout my IPhD study. I wish to thank my brothers. Most importantly, I am very grateful to my supportive wife, Asseil, and my wonderful daughter, Aljawharah, who provide constant inspiration and motivation on the long journey.

Abstract

Large-scale computing systems must overcome a number of difficulties before they can be considered a long-term solution to information technology (IT) demands, including issues with power use and its green impact. Increasing the energy efficiency of large-scale computing systems has long posed a challenge to researchers. Innovations in efficient energy use are needed that can lower energy costs and reduce the CO_2 emissions associated with information and communications technology (ICT) equipment.

For the purpose of facilitating energy efficiency in large-scale computing systems, virtual machine (VM) consolidation is among the key strategic approaches that can be employed. Virtual machine (VM) live migration has become an established technology used to consolidate virtualised workload onto a smaller number of physical machines, as a mechanism to reduce overall energy consumption. Nevertheless, it is important to acknowledge that the costs associated with VM live migration are not taken into account in the context of certain VM consolidation techniques.

Organisations often exploit idle time on existing local computing infrastructure through High Throughput Computing (HTC) to perform the computation. More recently the same approach has been employed to make use of cloud resources in large-scale computation. To date, the impact of HTC scheduling policies within such environments has received limited attention in the literature as well as the trade-off between energy consumption and performance. Also, the benefits of using virtualisation and live migration are not commonly applied in High Throughput Computing (HTC) environments.

In this thesis, we illustrate through trace-driven simulation the trade-off between energy consumption and system performance for a number of HTC scheduling policies. Furthermore, the thesis demonstrates the way in which various workloads can affect the time of VM live migration. We use a real experiment to explore the relation between various workload characteristics and the time of VM live migration. In order to understand what factors influence live migration, we investigate three machine learning models to predict successful live migration using different training and evaluation

sets drawn from our experimental data.

Through this thesis, we explore how virtualisation and live migration can be employed in HTC environment and used as a fault-tolerance mechanism to reduce energy consumption and increase the utilisation of a single computer in a large computing infrastructure. We propose various migration policies and evaluate them through the use of our extensions to HTC-Sim simulation framework. Moreover, we compare the results between the policies as well as the system where migration is not considered. We demonstrate that our responsive migration could save approximately 75% of the system wasted energy due to job evictions by user interruptions where migration is not employed as a fault-tolerance mechanism.

Publications

During my IPhD program, I have contributed to the following publications:

1. Alrajeh O., and Thomas N. Energy consumption of scheduling policies for HTC jobs in the cloud. *In Proceedings of the 8th International Conference on Simulation Tools and Techniques.* (2015), SIMUTools '15, ACM, pp. 343-348 [31]

2. Alrajeh O., Forshaw M., and Thomas N. Performance of virtual machine live migration with various workloads. *In: 32nd UK Performance Engineering Workshop.* (2016), University of Bradford [32]

3. Alrajeh O., Forshaw M., and Thomas N. Machine learning models for predicting timely virtual machine live migration. *In European Workshop on Performance Engineering.* (2017), Springer, pp. 169-183 [30]

4. Alrajeh O., Forshaw M., A. Stephen McGough, and Thomas N. Simulation of virtual machine live migration in high throughput computing environments. *In 22nd IEEE International Symposium on Distributed Simulation and Real Time Applications.* (2018), IEEE [29]

5. Alrajeh O., Forshaw M., and Thomas N. Responsive live migration fault-tolerance approach in high throughput computing environments (submitted)

Contents

List of Figures

List of Tables

ACRONYMS

ALR Adaptive Link Rate

AM Analytical Modelling

Co_2 Carbon Dioxide

CPU Central Processing Unit

DVFS Dynamic Voltage and Frequency Scaling

HTC High Throughput Computing

HPC HPC High Performance Computing

ICT Information and Communications Technology

IT Information Technology

KVM Kernel-based Virtual Machines

ML Machine Learning

MIPs Million Instruction Per second

NAS Network-attached Storage

OS Operation System

SLA Service Level Agreement

PM Physical Machine

PUE Power Usage Effectiveness

QoS Quality of Service

RAM Random Access Memory

RF Random Forest

SGB Stochastic Gradient Boosting

SDN Software-Defined Networking

VM Virtual Machine

VMM Virtual Machine Monitor

1

Introduction

Contents

Introduction

Large-scale computing systems are widely used due to the ability of on-demand provisioning resources to deliver inexpensive, convenient and user-friendly environment to consumers. These systems are entirely dependent on a sustainable power supply in order to work. The energy consumption of large-scale computing systems has increased dramatically which also leads to significant carbon dioxide (CO_2) emissions.

Meanwhile, global energy demand has increased from 10 thousand terawatt-hours in 1990 to 20 thousand terawatt-hours today. By 2040, global demand is expected to approach 40 thousand terawatt-hours [116]. Power saving initiatives are motivated by a number of factors, including governmental pressure and the need to save money. For example, the British government aims to lower CO_2 emissions by 80% in the next 35 years [15]. In the US, data centres consumed approximately 70 billion kWh of electricity during 2014, which is almost equivalent to 1.8% of the US total energy consumption. By 2020, It is expected that data centres energy consumption in the US will increase by three billion kWh [148]. Also, data centres could contain idle servers which might consume up to 70% of their peak power [73].

Thus, it is very important that data centres improve the efficiency with which they use their power and a great deal of innovation has taken place to this end. For instance, cloud computing takes advantage of virtualisation to allocate and migrate VMs to increase the utilisation of the physical resources as well as reduce the number of running servers which change the amount of computing power allocated to users in line with their needs [35, 43, 95, 162]. Moreover, data centres employ VM migration as a fault-tolerance mechanism to avoid hardware failures which enhances the overall performance and the energy efficiency [72, 114, 127, 137].

In this thesis, we explore how virtualisation can improve performance and decrease energy consumption in large-scale computing systems. In particular, we apply virtualisation and live migration techniques into high throughput computing (HTC) system where the resources are allocated on multi-use clusters which public users share them with the system. Throughout this thesis, we demonstrate the trade-off between energy and performance of existing HTC jobs scheduling policies to cloud resources. Further,

we investigate the factors which influence the time of migration. Finally, we propose a fault-tolerance mechanism and policies for performance and energy efficiency in HTC environments.

1.1 Motivation

The issue of energy efficiency in large-scale computing motivated this research. As we illustrated above, the global energy demand grows each year which needs to be controlled in order to reduce the cost and the carbon dioxide (CO_2) emissions. At Newcastle University, Information and Communications Technology (ICT) use energy has increased by 2.5% each year since 1990 [92]; it is estimated that ICT energy cost is around £750k per year and the CO_2 emissions of ICT is over 5,500 tonnes per year [92]. The University has launched a Carbon Management Plan (CMP) to reduce the CO_2 emissions of campus and using virtualisation was one of the aims to reduce CO_2 emissions of ICT. The University applied virtualisation over 200 servers as a mechanism to save energy which reduced the carbon dioxide by an estimated 700t per year [92].

The HTCondor [110] system is provided for Newcastle University researchers. The researchers use the system to run complex tasks to obtain the results quickly. The University uses 35 spread clusters on campus as HTCondor pool and the pool contains 1400 desktop computers. However, these computers are shared with public users, and a user might arrive on a computer at any time. When an interactive user logs in to the computer while it is executing an HTCondor job, the job gets evicted and rejoins the HTCondor queue to be reallocated on another machine. As a result, the job restarts its execution which increases its energy consumption and the overhead of the system.

To avoid jobs from being evicted and rejoining the queue when interactive user logs into a machine which executing HTC job, we propose a mechanism which migrates the job into another physical machine for resuming its execution.

1.2 Research Problems

In order to adopt virtualisation and live migration techniques into high throughput computing (HTC) environments for enhancing the performance and reducing energy consumption, the following research problems are investigated:

What is the trade-off between energy and performance in the context of management policies in large-scale computing systems? A number of previous works proposed policies and algorithms for managing physical resources, queuing and scheduling tasks, and fault-tolerance in order to enhance the performance. However, the energy consumption of some of these techniques is not considered.

How to deploy virtualisation and live migration techniques in HTC systems? Based on the literature, there is no generalisable model for virtualisation and live migration techniques within HTC environments in order to enhance performance and reduce energy consumption.

What is the cost of live migration? The cost of live migration is not considered in most of VM live migration models. Some approaches used live migration to transfer the job into the most energy efficient physical machine within the system in order to reduce the overall energy consumption. However, the energy consumption of VM live migration can be higher compared to keeping the VM running in its current physical machine. It is important to know the parameters that influence the cost of migration before starting the migration of VMs to ensure the most significant decision on saving energy.

Is it possible to predict the time of live migration with high accuracy? Many approaches used an analytical modelling for predicting the time of migration which requires knowledge about the internal behaviour of the system. To the best of our knowledge, this thesis includes the first attempt to predict live migration time by using machine learning where the information about system behaviour is not required.

Is it always reasonable to terminate a running job when an interactive user arrives into a physical machine within the HTC system? Fault-tolerance mechanisms in HTC systems can recover jobs failure due to user interruptions. However,

reactive fault tolerance techniques could increase the overhead as well as have large latency which affects the overall performance of the system. As a result, the energy consumption of the system can significantly increase.

Which computer to select for migration? Determining which computer to migrate the job when the migration is needed can influence the performance and energy consumption of the system. It is essential to provide a selection policy mechanism which improves the performance and reduces the energy consumption of the system.

1.3 Research Hypothesis

Using virtual machine live migration in a high throughput environment will enhance the performance and reduce the energy consumption of the system.

1.4 Research Objectives

This research aims to increase performance and decrease the energy consumption of a multi-use HTC environment by using virtualisation and live migration. The following objectives need to be achieved in order to solve the above research questions:

- Explore the area of energy-efficiency in large-scale computing to gain knowledge about the challenges and approaches for reducing energy consumption.
- Investigate virtualisation and migration approaches in large-scale computing in order to employ them into HTC environments.
- Demonstrate and understand the trade-off between performance and energy of existing and proposed policies in HTC environment.
- Measure the time of VM migration with various workload characteristics, hardware capacities, and network speeds.
- Propose machine learning models to predict the VM migration time based on real experimental data.

- Propose live migration techniques and policies for multi-use HTC environment to reduce the energy and enhance the performance as well as avoiding user interruptions failure.

1.5 Research Methodology

Quantitative analysis is used to obtain the findings in this thesis. In particular, the thesis consists of three research methodologies as follows:

Experiment: In order to measure the migration time of VMs which is associated with different workload characteristics and capacities, we use real experiments which represent the VM live migration procedure as it takes place in real-world contexts.

Modelling: In order to predict successful migrations where VMs can be migrated within a specific time window, we employ three different classification machine learning models, namely, Stochastic Gradient Boosting (SGB), Random Forest (RF), and Bagging.

Simulation: In order to evaluate the energy and performance of the proposed methods and policies of this thesis, we use our extension of the HTC-Sim simulation framework [78] which is based on real-world workload traces from Newcastle University.

1.6 Research Contributions

The key contributions of this thesis are as follows:

- An overview of the energy-efficient large-scale computing challenges and approaches is presented as well as the state-of-the-art in virtualisation and live migration methods.
- Evaluation of the impact of existing policies for scheduling HTC jobs to cloud resources on energy consumption and performance.
- Proposal and evaluation of an experimental setup for measuring the time of VM live migration.

- An automated script for conducting VM live migration between physical machines and maintaining a log of the CPU utilisation, total system memory, free memory, memory used, buffer cache, I/O activities, queue size, and load average throughout the migration.
- Development and evaluation of three classification prediction models for predicting the successful migrations where VMs can be migrated within a specific time window. The effect of tuning the models with different values as well as training and evaluating the models among the various sub-datasets from the original dataset is provided.
- Designing, implementing and evaluating two novel virtualisation and live migration methods to reduce energy consumption within HTC systems by extending the HTC-Sim simulator.
- Proposal and evaluation of the impact of new six selection policies for determining the location of migrated jobs within HTC system on performance and energy.

1.7 Thesis Structure

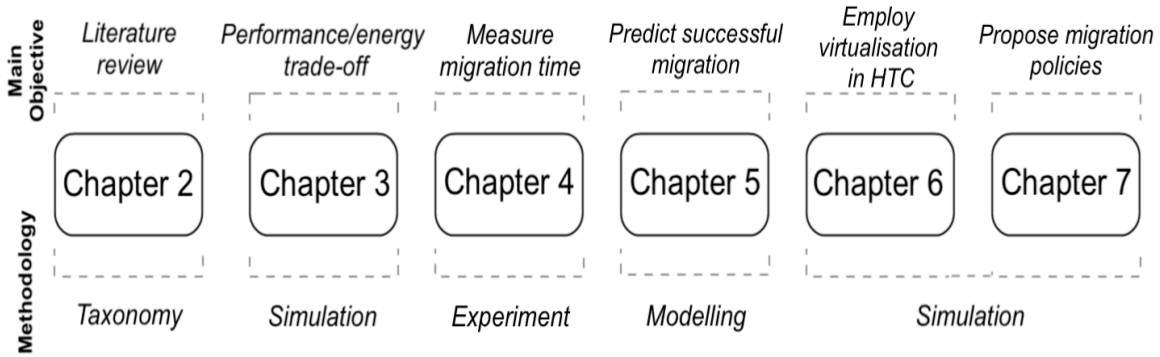

Figure 1.1: Thesis structure

Figure 1.1 presents an overview of the core chapters of this thesis. This thesis is structured as follows:

Chapter 1 describes the motivation behind this work and highlights the problems, main objectives and contributions of this thesis.

Chapter 2 provides background material and a summary of the most related work to the research scope which described in this thesis.

Chapter 3 investigates the trade-off between performance and energy of existing policies for scheduling HTC jobs to cloud resources by extending of HTC-Sim to incorporate energy measurement and evaluate the energy and performance. The results of this chapter are published in [31].

Chapter 4 presents a real experiment setup for measuring the time of VM live migration with various workloads characteristics. The chapter illustrates the link between the memory pages and the time of migration. The findings of this chapter are published in [32].

Chapter 5 exhibits the process of creating and testing three predictive models. The models are used to foretell how likely a VM is going to be migrated within the time frame of the running workload. Finally, the comparison between the models and others is provided. The contributions of this chapter are published in [30].

Chapter 6 discusses the implementation of virtualisation and live migration into HTC-Sim. Also, it presents two novel live migration methods which can be easily applied to HTC systems. The chapter discusses the evaluation of the proposed methods. The outcomes of the chapter are published in [29].

Chapter 7 evaluates the impact of a live migration fault-tolerance mechanism on performance and energy consumption within a multi-use HTC environment. The chapter proposes six selection mechanisms to determine the target computer when migration is needed. Throughout the chapter, we discuss some useful tips which would assist the administrator of the HTC system when they employ the live migration as a fault-tolerance mechanism in their system.

Chapter 8 summarises the conclusions of the work presented in this thesis and illustrates the limitations of work. Also, the chapter discusses directions for further work in the area.

2

Background and Related Work

Contents

Summary

This chapter provides an overview of the related background material motivating and underpinning the work presented in this thesis. The chapter presents an overview of the energy problem in large-scale computing systems. Also, it includes the energy efficient techniques which reduce the energy consumption of large-scale computing systems. Furthermore, the chapter investigates the related work of virtualisation which is commonly used in large-scale computing systems as energy-efficient and fault tolerance mechanisms. Through this chapter, we highlight the gaps of the relevant research feature and discuss how this thesis tackles these gaps.

2.1 Energy-efficiency

The World Energy Council defines efficient energy use as a lowering of the amount of power used by a facility or amenity [126]. Unfortunately, data centres are extremely complicated structures with a large number of components from different research areas such as management, computing, and networking. Thus, it can be very challenging to cover each energy aspect of each component within a system in detail due to their diversity. Based on the literature, some surveys such as [44, 164] describe energy models in terms of their stationary and moving energy use, which deals only with wasted power while the equipment is on but not in use. However, others [37] argue that there is a measurable difference between the power used by computing equipment and that used by supporting equipment in order to measure energy waste from supporting equipment. Nonetheless, some surveys such as [116] combined these two approaches in order to define energy efficiency more holistically.

Intended to create savings by a more efficient use of power while being as green as possible, energy-aware computing is a type of green computing which has begun to shift the high-level computing systems paradigm towards a balanced performance-energy system. As Couch *et al.* [103] wrote in 2008, "The goal of energy-aware computing is not just to make algorithms run as fast as possible, but also to minimise energy requirements for computation, by treating it as a constrained resource like memory or disk".

Green Computing: or "Green IT" is an approach to managing large data centres which aims to reduce the environmental damage caused by information technology. The aim of green IT is to maintain a balanced and environmentally-friendly IT service which has as slight an impact on the environment as possible. Also, the use of a green computing approach can have a number of advantages for organisations and the general public, such as lower power bills, better performance, and a more efficient use of space [164].

Mingay *et al.* [124] defines Green IT as "the optimal use of information and communication technology for managing the environmental sustainability of enterprise operations and the supply chain, as well as that of its products, services, and resources throughout their life cycles".

Data Centres Energy Consumption: the large-scale computing systems are housed in clusters and data centres. The energy consumption in data centres is not only based on the physical nodes. According to Bermejo *et al.* [45], the IT equipment uses around 50% of the data centre power consumption and 25% might be consumed by cooling systems.

Servers Energy Consumption: large-scale computing system such grids, clusters, and cloud can contain hundreds to thousands of servers in order to process and solve a large number of complex tasks. However, some of these servers are idle, and the others are not fully utilised which increase the energy waste as well as CO_2 emissions. According to Natural Resources Defense Council [67], approximately 30% of the servers within a data centre are idle. Moreover, over half of a server's maximum power consumption can be used when it is merely running idle [63]. Additionally, as Barroso *et al.* [39] argue typical data centres waste a massive amount of power by running servers idle or at less than half capacity. Also, unused computer parts such as fans can also waste over half the power they receive.

Network Energy Consumption: the pool of resources in large-scale computing need a network architecture in order to communicate with each other. The network hardware and traffic influence the energy consumption of the network within a data centre. As Brown *et al.* [52] observe, approximately 5% of the average data centre's total power use is absorbed by the network. This can go up to 12% when servers

are used at full load [23]. More recent research [150] explained that the network in a typical data centre consumes about 30% of the total energy where 15% of network energy used by access switches, 10% consumes aggregate switches, and 5% goes for core switches.

Applications and Operating Systems Energy Consumption: the computer has three separate subsystems which need to be assessed independently. Firstly, in order to link the physical equipment with its software, an operating system is required. Secondly, the software requires an appropriate runtime environment. Lastly, the end user needs to be able to use some piece of software that performs the equipment's primary role [116]. An operating system is required for running software, and may be standard software or a bespoke version created for a specific set of equipment. From the point of view of the equipment, any resources the operating system uses which do not contribute to the equipment's primary role are wasted. This is especially the case in situations such as repair or upkeep, where overheads are incurred completing tasks which are not related to the hardware [116].

2.1.1 Energy Efficiency Mechanisms

In this section, we provide a brief overview of different energy approaches introduced to achieve energy savings. In [21, 27, 44, 96, 116, 125, 165], the authors provide various surveys of energy-efficient computing.

Software: The hardware designers need to think about much more than simple battery life issues, and must consider the energy efficiency of every part of a computer in order to reduce their environmental impact and save money. Furthermore, the development of legislation and software to manage and exploit these hardware improvements has started to become more important, allowing the best use to be made of these advancements.

The weak software design could lead to both performance losses and high energy consumption. Assessing the energy usage of programs is difficult due to the instruction order which affects the generation and compilation of code as well as energy usage [44, 85, 116, 134].

To determine which applications or programs are keeping the CPU busy and wasting energy, programming techniques such as waiting loop and active polling [93] are introduced. Moreover, a program or application might keep the CPU busy without performing any useful work when it is waiting for input/output (I/O). As a result, the energy waste of the system increases. This issue is known as "wake-up the CPU", and the authors of [134] have presented various examples of programs and applications that unnecessarily wake-up the CPU.

Regarding the operating system (OS), several approaches are proposed in the literature such as Nemesis OS [130] and ECOsystem [166]. They mainly focus on enhancing the battery-powered devices to be more energy efficient. By using them, the system developers and operators can identify the energy as well as the quality of service (QoS) requirements for each application. Meisner *et al.* [122] examined how high-power and low-power operating states that have rapid transition times can be used for saving energy. Their approach could save up to 74% of server energy consumption.

Koller *et al.* [100] introduced WattApp which is a framework that helps in predicting application-aware energy in terms of data centres. The framework can be applied in a virtualised heterogeneous environment. Moreover, a study by Zhang *et al.* [169] showed that, based on performance variations of resources, the runtimes of similar applications would differ and result in varying energy consumption. It is thus essential to know that various workloads and the amount of energy they consume in order to allocate and run them more efficiently.

Server: Dynamic voltage and frequency scaling (DVFS) is a technique used widely to manage power by dynamically changing the processor's clock frequency so that power can be saved by allowing the supply voltage to reduce. Modern processors can function at varying frequencies which involve different energy consumption and processing speeds. Apart from processors, DVFS has also been applied to other devices including main memory, storage, and cache memory [57]. It is, however, important to note that as a decreased frequency can also diminish the circuit's performance, it is possible that DVFS can negatively impact the processor's performance, because of which DVFS must be applied smartly to ensure that the performance remains high [125]. The studies by Lai *et al.* [104] and Wa *et al.* [161] demonstrate this trade-

off.

Network: the key technology which allows components of large-scale computing systems to communicate with each other is the network. ASHRAE and Cisco provide best practices guidelines [18, 19] for networking design that could reduce the energy consumption of the network. Furthermore, a number of global plans have concentrated on using energy efficiently in all types of network. For example, a European Commission funded study [3], ECONET (low Energy COnsumption NETworks) aims to research wired network technologies which can prevent power being wasted while idling. The aim of the project is to lower the power requirements of network appliances by between fifty and eighty percent.

Nedevschi *et al.* [129] proposed putting the idle network appliances in a sleep mode which reduced their energy consumption. It also analyses how networks could benefit from the use of Dynamic Voltage Scaling (DVS). When there is little demand being placed on equipment, the study found that network could save up to half its energy costs.

A study by Giroire *et al.* [85] examined network energy consumption and how it can be reduced by considering this problem as an NP-hard bin-packing problem. To resolve this, an Integer Linear Programming (ILP) formulation is suggested with the objective function of reducing the number of network elements required for communication. ILP, however, may not be a suitable or quick method of achieving the goal in both global networks and large-scale data centre networks that include numerous resources, as explained by Ferreto *et al.* [76].

Adaptive Link Rate (ALR) is a widely-used approach to reduce the energy consumption of an Ethernet link by adapting link speed to traffic demand. Bilal *et al.* [47] did a brief taxonomy on ALR. However, ALR approach could affect the performance of the system as the packets need to be queued when the network equipment is in sleep mode.

Applying Software-Defined Networking (SDN) provides opportunities for networks which minimise energy consumption. The first study in this field was accomplished by Tu *et al.* [154] who evaluated how controlling the SDN network's traffic flow can

help in minimising energy consumption. In their work, a greedy algorithm and a 0-1 integer programming model are the two methods suggested that can result in reducing the energy cost by 30-40%.

Resource Management: resource management is used to schedule tasks, monitor and manage physical and virtual machines in order to improve performance and reduce energy consumption in large-scale computing systems. However, resource management might affect the performance of the system. This problem needs to be considered when the resource management deploys in a facility.

Scheduling: Liu *et al.* [111] studied the power efficiency of intensively managing data grids on distributed systems. They created a power-efficient management simulator called Distributed Energy-Efficient Scheduler (DEES) that provides energy savings by incorporating scheduling jobs with data assignment approaches. Information duplication and job transferals are lowered, reducing the power drain. In [58], the author proposed an energy efficient distributed scheduling and management algorithm for large-scale computing systems. The algorithm does not require prior information. The authors provided a mathematical analysis which showed the trade-off between energy consumption and performance of the system. Juarez *et al.* [94] proposed a real-time dynamic scheduling algorithm which reduced the energy consumption and makespan of tasks in a cloud environment. Garg *et al.* [82] introduced five scheduling policies to reduce the energy and CO_2 emissions of HPC applications in the cloud. They used a simulation tool to evaluate their policies, and their results showed 33% saving on energy. Beloglazov *et al.* [42] evaluate various energy-efficient resource allocation policies and scheduling algorithms in the cloud computing systems. They consider the evaluation of power consumption and Quality of Service (QoS) impacts, though they do not consider HTC workloads in their evaluation. Duy *et al.* [71] assess the performance of a neural networks predictor for enhancing server power utilisation in the Cloud. They utilise predictors of future load demand predicated on historical demand. The outcomes showed that the energy consumption can be reduced by 46.7%.

Virtualisation: De Alfonso *et al.* [64] deployed a virtual cluster in cloud environment and compared the cost of physical cluster and virtual cluster. The High Performance Computing (HPC) was deployed as virtual cluster on Amazon EC2. They construed

the energy consumption in their comparison by applying different energy policies into the physical cluster. They concluded that in some cases virtual cluster could be cheaper than physical cluster. Sharma *et al.* [147] analysed a variety of Virtual Machines (VM). They suggested the use of a new VM load balancing algorithm which could reduce costs and response times. Li *et al.* [107] suggested the use of a VM placement algorithm named EAGLE. This is able to lower power use by reducing the use of resources, lowering the amount of running PMs required. EAGLE has been found to lower power usage by up to 15%. In Section 2.2, we provide background about virtualisation and discuss VM migration as well as the VM migration energy efficiency techniques which are suggested in the literature.

2.1.2 Energy Efficiency Metrics

Developing recording procedures and successful enticements for the efficient use of power in computing are important aims of the EU Code of Conduct for Data Centres and the National Market Transformation Program. In addition, if we are to altering the behaviour of computer operations, it is vital that we are able to calculate the efficiency of energy transfers between equipment in order to obtain a grasp of the types of wastefulness present in modern data centres. In the literature, many energy-efficiency metrics are proposed in order to measure the consumed energy within a data centre or IT infrastructure. The authors of [22, 56, 97, 145, 146, 160, 165] provide reviews of various energy-efficiency metrics.

In order to calculate the amount of energy used by IT equipment such as network hardware within data centres, the Green Grid introduced the Power Usage Effectiveness (PUE) measurement [40]. This is shown below.

$$PUE = \frac{TotalFacilityPower}{ITEquipmentPower} \tag{2.1}$$

Where $TotalFacilityPower$ is the total amount of energy used by a facility (cooling, lighting, etc.) and $ITEquipmentPower$ is the total energy used by IT equipment (network, servers, etc.) within the facility. When facilities are associated with low PUE values, the facilities are considered to be energy-efficient facilities. However, the

PUE does not measure the energy efficiency of IT equipment; it measures how much energy is used by IT equipment compare to the energy consumed by other equipment within a facility.

The data centre considered to be very inefficient when the value of PUE is more than 3 [40]. Uddin *et al.* [155] did an experiment in a small data centre which included 150 racks to conduct PUE value. The PUE value of their data centre was 3.2 which represent a poor overall efficiency. The result of their experiment showed that 85% of the racks were underutilised where these racks did not execute work but consumed idle energy. Service providers can use these energy efficiency metrics as they determine the service performance as well as identify the resource that can aid in energy efficiency enhancement.

Furthermore, the Green Grid [40] introduced the Data Centre infrastructure Efficiency (DCiE) which is given by the following equation:

$$DCiE = \frac{1}{PUE} = \frac{ITEquipmentPower}{TotalFacilityPower} \tag{2.2}$$

Also, the Data Centre energy Productivity (DCeP) metric proposed by Green Grid [17] as shown in equation 2.3. The *UsefulWorkProduced* can be calculated as the number of jobs which are processed by the hardware during the assessment window. However, the accuracy of measuring the *UsefulWorkProduced* might be a drawback of this metric.

$$DCeP = \frac{UsefulWorkProduced}{TotalFacilityEnergyConsumption} \tag{2.3}$$

In order to measure the carbon of data centre operations, the Green Grid [38] proposed Carbon Usage Effectiveness (CUE) metric, and is defined as:

$$CUE = \frac{CO_2\ emitted(kg\ CO_2\ eq)}{unit\ of\ energy(kWh)} * \frac{TotalDataCentreEnergy}{ITEquipmentEnergy} \tag{2.4}$$

Further metrics have been proposed in the literature. The Space, Watts and Performance (SWaP) metric [12] measures the energy-efficiency of a data centre by three parameters: performance, space, and energy. The IT energy Productivity (ITeP) [145] measures useful IT work over IT consumed energy. The Rack Cooling Index (RCI) [87]

is introduced to measure the effectiveness of IT equipment cooling.

2.1.3 Benchmarking

The initial industry standard was introduced in 2007 by SPECpower_ssj2008 [11] and was used as a benchmark for all others in order to assess and stipulate a method of contrast between measured performance and measured power expenditure. This metric signifies the combination of the performance measured at individual stages or load levels (in ssj_ops) divided by the total of the average power (in watts) at each individual goals point including active-idle. As a benchmarking foundation standard SPECpower utilises the current benchmark to integrate energy measurements; the foundation for this is an enterprise Java workload. The benchmark itself instils progressive stages of load on a given machine, usually calculating the energy usage and performance of server hardware. The extremes of which are from active-idle (0%) and peak (100% load at 10% progressive load levels). Lately SPEC launched SPEC VIRT_SC 2013 [9], this merges various benchmark workloads (which incorporates web server, application server, mail server and CPU dominant workloads) in order to appraise the functions of the servers for virtualised conditions. Additionally the use of benchmarking comprises the measurement of power usage and power/performance associations.

The server efficiency rating tool (SERT) [8] was provided by the Standard Performance Evaluation Corporation (SPEC). The tool measures and evaluates the energy efficiency of servers. The tool provides a set of synthetic workloads which exercise the memory, processor, and I/O and tests the energy efficiency of a system at various loads levels. In [158], the authors used machine learning to predict the power consumption of servers. They used the SERT tool to obtain the dataset of the prediction models, and their prediction results associated with high accuracy.

2.2 Virtualisation

Virtualisation creates an abstraction layer between the hardware and operating system (OS). Virtual Machine Monitor (VMM) or Hypervisor is a software which creates and manages Virtual Machines (VMs) on physical machines. Each physical machine can

have a number of VMs depending on the demand. A virtual machine that is employed to the physical machine is capable of executing a desired operating system as well as being isolated from the rest of the VMs on the same physical machine. Thus, other VMs failure or malware infection that is being executed on the same physical machine do not impact each other.

The IBM Corporation developed virtualisation in the 1960s to divide a large mainframe computer into various logical instances. In 2000, this ability of dividing enabled several applications and processes to operate simultaneously which also increases the environment efficiency as well as reduces the maintenance cost [61]. Recently, there has been increasing interest in virtualisation technology not only because of the growth in computer performance but also because of the development of green computing and the limitation of space in data centres. Green initiatives, in particular, are frequently applied using virtual machine consolidation.

The following four virtualisation technology solutions are the most widely used: Kernel-based Virtual Machine (KVM), Xen hypervisor, VMware solutions, and Microsoft Hyper-V, are presented. Though these four solutions all support power management, they do not coordinate VM specific calls concerning power state changes. A crucial benefit of virtualisation is live virtual machine migration, which the majority of hypervisors provide. The live virtual machine migration feature will be focused on in this thesis.

2.2.1 Virtual machine migration

VM migration techniques substantially improve how data centres and clusters can be managed by transferring a complete state of the VM to the destination host from the source host. The virtualised environment provides two migration strategies: live migration and non-live migration. Hence, VM migration includes single as well as multiple migration. whereas single migration migrates only one VM at a time, multiple migration migrates numerous VMs. According to the literature [24, 59, 167], the benefits of VM migration can be listed as follows:

Power management: By switching the idle server to sleep or off mode according to the resource demands, significant energy can be saved as the idle server consumes 50%

of its peak power [63], as well as consolidating the running VMs to reduce the number of active hosts which also benefits in saving energy.

Load balancing: An overloaded server could diminish the quality of service (QoS) experienced by users. The servers that function in an underloaded state waste energy. Live VM migration helps all data centre servers function uniformly without decreasing QoS. Also, the load balancing ensures that a single point of failure can be avoided as it distributes the server workload across various physical hosts in a data centre.

Fault Tolerance: To reduce the impact of hardware failures on the execution of applications and on the performance of the system, hardware failures must be anticipated and then handled proactively by migrating the job to another host.

Hardware maintenance: This ensures that periodic maintenance helps in extending the system's life. VM migration can transfer the running task to another host so that the task can continue to be serviced during system maintenance.

Resource sharing: When a task needs more hardware resources such as cache, memory, and cores, the task could be migrated to a more powerful host.

2.2.2 Live migration

The virtual machine (VM) during the live migration process continues to run on the source physical machine as its memory pages get transferred to the target physical machine. At some point, the VM will stop executing on the source machine and resumes its execution on the target machine. To perform the live migration, the VM must be installed on network-attached storage (NAS) which is accessible by physical machines. Furthermore, the same virtual machine manager (VMM) should be installed in both source and target machines with both machines having appropriate ports open [139].

There are two live VM migration approaches which currently used in the data centres, namely, post-copy live migration and pre-copy live migration. This thesis refers to migration as the pre-copy live VM migration unless stated otherwise.

Post-copy live migration: The post-copy technique was proposed by Hines *et al.* [88, 89]. To initiate a post-copy VM migration process, the VM execution is suspended on

the source server while the necessary CPU states and memory pages to resume VM execution on the target server are transferred to the target server. The VM fetches memory pages from the source server when it is resumed at the target server. The remaining memory pages of the VM are then moved to the target server in on-demand basis. Hence, this approach can take long migration time which consumes the resources on the source as well as target servers for a longer time because of residual dependency. Also, the post-copy live migration has some downtime initially which make the VM's services eventually be unavailable as well as in case of a page fault.

Pre-copy live migration: Clark *et al.* [60] proposed the concept of a pre-copy live migration algorithm. This algorithm copies and transfers the memory pages of VM from the source server to the target server. The initial phase of the pre-copy live migration algorithm is the iterative phase where the pre-copying occurs in rounds. When the memory pages modified in a given round, the modified memory pages will be transferred again in the next round. Later, when there are relatively few uncopied pages, the VM get suspended on the source host, and the remaining pages get transferred to the target server as well as the CPU states termed as the stop-and-copy phase. In this way, the VM can be migrated from machine to another with minimal downtime.

Pre-copy live migration has been widely applied in data centres today for many management operations as it does not include residual dependency between the target and source servers. Moreover, the pre-copy approach is fault-tolerant as the source server always includes a consistent copy of migrating VM until the target server resumes the VM execution. As pre-copy migration does not have a residual dependency, the migration process needs less time to finish compared to post copy migration. Also, after migration, the source server is released when the memory pages of the VM is completely transferred to the target computer. As a consequence, data centres tend to use pre-copy live migration as an acceptable VM migration technique.

2.2.3 Live Migration Performance Metrics

VM live migration performance is characterised according to two time-related metrics: downtime and migration time. The migration time is the time window between the beginning of the migration process and the end of the migration process when the

VM resumes at the target server. Also, there is always some downtime during the migration process where the VM stops on the source computer and resumes in the target computer. The migration time, as well as the downtime, negatively impacts the performance of the application that is being executed on the migrating VM [49, 59, 159, 165].

Furthermore, based on the literature, there are three parameters which could influence the performance of live migration: bandwidth, the memory size of VM, and the memory change rate. For a particular bandwidth, a VM which has large memory needs more time to transfer the memory to the target server. Higher memory change rate, which indicates how much memory is modified in a second, suggest that more memory will be needed for re-sending in the iterative phase, thus increasing the migration time. The major factor impacting a live VM migration's performance is the network bandwidth. When the network speed is high, the live migration process will take a short time to complete. Thus, an inversely proportional relationship exists between the migration bandwidth and migration time and downtime.

In the following subsections, we present the studies that measured, evaluated and predicted the performance of migration.

2.2.3.1 Practical Test

A study by Akoush *et al.* [25] has demonstrated the effect of network bandwidth on the performance of the VM live migration. The VM memory transfer rate between source and target servers reduces in case of the migration occurring over a low-speed link, leading to longer migration time and downtime. A high-speed link, on the other hand, reduces the migration time and downtime. Therefore, a longer time is needed over a low-speed link for a larger VM which has higher memory change rate for completing its migration, causing a significant amount of migration downtime.

Strunk *et al.* [151] measured the duration of VM live migration and the energy overhead with various RAM sizes of the VM. Also, they used different network bandwidth capacities between two servers for VM migration. They used KVM as a hypervisor and developed their own tool to stress the memory. Their results showed that the time of migration increases when the memory size of VM increases and the time of migration

decreases when the network capacity increases. However, Strunk's workload generation method failed to represent genuine VM live migration. Through the employment of a benchmark which produces a range of workload features, we bypass this limiting factor.

In [62], the author shows the link between the memory size and bandwidth on the time and energy of live migration. When the VM executes a workload, the time of migration increase as well as the energy consumption. However, the benchmark which used in their experiment only stresses the CPU.

Rybina *et al.* [142] investigated the time of VM migration that runs with multiple running virtual machines and the impact of VM migration on the running virtual machines. They used the SPEC CPU2006 benchmark suite to generate CPU intensive workloads. They showed that starting to migrate VMs with intensive memory workload first is cheaper than migration VMs with intensive CPU workload first. However, their experiment is based on homogeneous physical hosts which we avoid in this thesis.

Another study by Salfner *et al.* [143] examined the impact on downtime and total migration time of used memory size, memory size, CPU load. They found that memory data migration performances are mainly affected by used memory size, configured memory size, and memory dirtying rate. They examined how these factors impact migration performances on VMware and Xen platforms. Their experiment, however, was on homogeneous servers and the configuration of all tested VMs had only one virtual CPU.

Hu *et al.* [90] assessed migration strategies by creating an automatic testing framework for comparing the migration performances regarding total migration time, total network traffic, and downtime. They focused on four hypervisors with varying parameters: Xen, VMware, KVM, and Hyper-V. Non-live migration, as well as storage data migration, are considered in their framework. However, they did not consider using different VM resource capacities in their experiment, and their evaluation results are based on one type of VMs.

Dhanoa *et al.* [69] employed an experiment to measure the time and energy of live migration. They applied KVM hypervisor to create and perform the live migration

between two identical physical machines. They used two types of VM where the memory size on the first VM is 1 GB and the second VM is 2 GB . The time and energy of migration were measured when the VMs were idle, and the VMs were migrated six times between the physical machines.

In [170], the authors evaluated the performance of non-live migrations by using varying configurations to assess single and multiple migrations. They found that the interruption time of the services which provided by the VM is longer compared to the total migration time due to the required time after migration for the VM to back to its original performance.

Li *et al.* [106] used KVM platform to evaluate how bandwidth limit impacts migration performance on total migration time as well as downtime and energy consumption. They created an analytical model for formulating the link between them based on the results of their experiments. They, however, included identical VMs with 2 GB RAM. Further, the VM has a running workload which is randomly specified memory size and the data which stresses it.

Bezerra *et al.* [46] evaluated the overheads which are caused by VM live migration from the clients' perspective on real as well as virtual environments. They determined that it is possible for a client to observe the performance degradation which can occur during the migration process.

In our experiments which are presented in Chapter 4 and 5, we use 20 different workloads. The benchmark that we used to generate the workloads has been not used previously in the literature for similar context. The benchmark can generate various workloads which stress the CPU, memory, and I/O. Also, we used nine distinct VM hardware capacities for migration. Our VMs vary in CPU and memory capacities which are not considered in most of the previous works. Furthermore, we used two different network speed to obtain the results of our experiments. Table 2.1 shows comparisons of VM live migration practical tests and illustrates that our experiments setup employ heterogeneous servers.

Paper	Hardware environment	Hypervisor type	VM type	Workload	Network Speed
Akoush *et al.* [25]	Homogeneous	Xen	Various	SPEC CPU, SPECweb SPECsfs	100 Mbps 1 and 10 Gbps
Strunk *et al.* [151]	Homogeneous	KVM	Various	Own tool (memory stress)	1 Gbps
Dargie *et al.* [62]	Homogeneous	KVM	Various	Own tool (CPU stress)	1 Gbps
Rybina *et al.* [142]	Homogeneous	KVM	Identical	SPEC CPU	1 Gbps
Salfner *et al.* [143]	Homogeneous	VMware Xen	Various	Own tool (CPU, memory stress)	?
Hu *et al.* [90]	Homogeneous	KVM, VMware Xen, Hyper-V	Identical	Own tool (CPU, I/O, memory stress)	1 Gbps
Dhanoa *et al.* [69]	Homogeneous	KVM	Various	–	?
Li *et al.* [106]	Homogeneous	KVM	Own tool (memory stress)	Identical	1 Gbps
Alrajeh *et al.* [30]	Heterogeneous	KVM	Various	SPECjvm2008	100 Mbps 1 Gbps

Table 2.1: Comparisons of VM live migration practical tests

2.2.3.2 Prediction

The modelling and prediction of virtual machine migration has formed the basis for a number of previous works. Also, the performance prediction of systems and applications involve two techniques: Analytical Modelling (AM) and Machine Learning (ML). These techniques are used widely for several purposes such as cost prediction, elastic scaling, faults detection, and energy consumption prediction.

In [25, 26, 68, 106, 112, 113, 163, 168, 171], the authors introduced models for predicting the performance metrics of live migration such as migration time, downtime, and memory pages, but none of them used machine learning for the prediction.

Analytical Modelling: For several years, AM was used as the reference technique for performance evaluation as well as for predicting computing platforms in various application contexts. AM incorporates the expertise regarding the systems' or applications' internal dynamics and encodes this knowledge into a mathematical model that intends to capture the mapping of tunable parameters onto the performance. Generally, AM techniques require little to no training for conducting the predictions in the concerned scenario. However, some AMs tend to be dependent on assumptions regarding the behaviour of the modelled system and its workload. Thus, in situations where the assumptions do not match, their accuracy can be affected [70].

Nathan *et al.* [128] developed performance prediction models for page-skipping optimisation. Their model considers the pre-copy migration, and it is based on two factors.

The first involved the number of unique pages which need to be transferred in each iteration. The second involved the total of skipped pages for each iteration.

Aldhalaan *et al.* [26] developed multiple analytic models that can help in predicting migration performance, such as downtime, total network traffic, and network utilisation. For this, they considered three cases: pages being copied during the pre-copy phase, pages copied only during the VM's downtime, and the rate of dirty pages.

Salfner *et al.* [144] created models for predicting live VM migration's worst-case performances. These models focused on workload feature as well as host behaviour. The models determined that memory access pattern is the factor which influences the downtime and total migration time. They evaluated the models with real as well as artificial workloads on various virtualisation platforms, such as KVM, VMware, and Xen.

Akoush *et al.* [25] developed predictive models of live migration performance based on experimentation with a number of SPEC benchmarks, and observed that network link speed exists as the most dominant factor in migration performance.

Machine Learning: The Machine Learning (ML) modelling does not require any knowledge regarding the internal behaviour of the target system or application. In particular, ML approaches are based on observing the behaviour of the system in various settings for determining a statistical behavioural model without any understanding of the internal operations of the system. Though in the past few years, ML techniques have become increasingly widely used to predict the performance of complicated systems. However, a significant limitation of this technique is the accuracy which is primarily based on the representativeness of the dataset that is employed in the training phase. It should be noted that the prediction models which their features are not explored adequately during the training process tend to present low accuracy [70].

Machine learning approaches have been used operationally to inform various resource scheduling decisions within large-scale computing systems [118]. Uriarte *et al.* [156] apply a Random Forest method to service clustering in autonomic cloud environments. Further, machine learning was used for the first time in [81] for predicting the PUE of a data centre for which they used 200,000 training samples from Google data centres during two years. They used the neural network to build the prediction model by

using 19 features.

In [135], the authors used machine learning algorithms to anticipate how many memory pages get dirtied during ongoing iterations. In particular, they performed a time series prediction by applying a historical analysis of past data which concerns dirty memory pages.

Jo *et al.* [91] applied three machine learning techniques, namely, linear regression, support vector regression (SVR) with bootstrap aggregation, and SVR with non-linear kernels. They predicted six metrics of live migration: total amount of data transferred, total VM migration time, VM downtime, CPU and memory usage on the physical hosts, and performance degradation of the VM. It should be noted, however, that these features and the collected dataset may not be applicable for complex data centre environments which have a heterogeneous physical machine.

In thesis, in order to understand what factors influence live migration, we investigate three machine learning models to predict successful live migration using different training and evaluation sets drawn from our experimental data. In our work, we present the process of creating a Stochastic Gradient Boosted (SGB), Random Forest (RF) and Bagged Tree (BT) models from the results of the experiment in order to predict the time of migration, unlike the existing literature where the process of creating the prediction models are not shown. Also, we used various training and evaluation sets to create and test our models, unlike the work of the existing literature where they used one training set and evaluation set to create and test their models.

2.2.4 Virtual Machine Consolidation

In large-scale computing systems, the VMs are allocated across many physical machines, and some of these physical machines are not fully utilised. Virtual machine consolidation aims to lower the number of running physical machines by reallocating and merging VMs into a smaller number of physical machines. The idle physical machines will be switched to a sleep mode or switched off which reduces the energy consumption of the data centre. However, VM consolidation can impact the performance of the VMs when the physical machine is over-utilised. The trade-off between

performance and energy needs to be considered when the VM consolidation is applied. Also, the cost of live migration is not considered in most of VM live migration models. In some cases, the energy consumption of VM live migration can be higher compared to keeping the VM running in its current physical machine. It is important to know the parameters that influence the cost of migration before starting the migration of VMs to ensure the most significant discussion on saving energy.

As part of an attempt to lower energy usage and service-level agreement (SLA) violations, a number of papers in the literature have proposed dynamic VM consolidation approaches [35, 43, 95, 162]. Basically, the approaches migrate the VMs from under-utilised servers and switch them into a sleep mode.

Beloglazov *et al.* [43] formulated algorithms and policies linked to live migration and dynamic VM consolidation. The researchers drew on CloudSim instruments to facilitate the assessment of their approaches. Initially, to carry out the reallocation of the VMs, the physical machines were categorised into the following two groups: overload and underload machines. Following this, three distinct VM selection policies were employed to determine the VM which needs to be transferred from the present to the novel host. Although useful insights were gathered from their experiment, it should be noted that the researchers' method was not appropriate to ensure that the VM live migration costs were lower than the advantages gained.

Feller *et al.* [74] used the multi-dimensional bin-packing (MDBP) problem to solve the workload consolidation difficulty. Also, they designed a novel nature-inspired workload consolidation algorithm based on the Ant Colony Optimisation algorithm (ACO) for energy-efficient cloud computing. Their aim is to reduce the number of physical machines that are used to compute the current workload. The authors developed a simulator based on Java in order to evaluate their approach due to limitations of the CloudSim tool by comparing ACO with First-Fit Decreasing algorithm (FFD). The simulation results demonstrated that ACO saved much more energy than FFD.

In [131], the authors introduced a VM consolidation algorithm with multiple usage prediction (VMCUP-M). The algorithm aims to reduce the number of migrations, the number of running servers, and the servers' energy consumption. They used real and synthetic workloads to evaluate their approach by simulation, and their approach does

not violate the SLA.

Teng *et al.* [153], investigated the energy problem of virtual clusters on physical servers. They suggested two processes for batch-oriented consolidation and online placement. Their algorithm selects the most efficient frequency CPU. They evaluated their approach on Hadoop testbed.

2.3 Simulation Tools

The simulation of Grid and Cluster level has formed the basis for many previous works such as SimGrid [105], GridSim [53], OptorSim [41], SiCoGrid [123], and Chi-Sim [48]. These tools assist researchers in understanding the parallel and distributed systems and evaluate new policies of managing tasks in HTC. In addition, Cloud simulators such as CloudSim [55], GreenCloud [98], iCanCloud [132], and MDCSim [108] can determine the trade-off between performance and cost as well as energy. Though, unlike our extension to HTC-Sim [78], the tools as mentioned above might not be ready to evaluate VM migration techniques and policies for the purpose of reducing energy waste in HTC environment. Also, our simulation is unique in its capability to model live migration in multi-use clusters with interactive users besides using real-world workload traces.

MDCSim [108] is a data centre simulator, which can provide information about the energy consumption of data centres. The simulator has three configuration layers (a kernel layer, a user-level layer, and a communication layer) for modelling different characteristic of data centres.

Calheiros *et al.* [55] developed a CloudSim framework that can help to model and simulate cloud environments. The CloudSim framework provides many features and one of them is simulating and modelling the energy-aware datacentre. CloudSim is a generic tool that can be used to model and simulate varying energy policies and mechanisms. However, CloudSim does not support modelling of multi-use clusters with interactive user's workloads.

iCanCloud [132] simulates cloud computing systems based on the Amazon Elastic Compute Cloud (EC2), despite its creators stating that the application programming

interface (API) of iCanCloud can be extended to simulate other environments. The simulation is mainly used to predict the trade-off which exists between the cost and the performance of an application that runs in a particular hardware.

SimGrid [105] is a simulation framework to simulate scheduling algorithms in a distributed environment. It was initially developed to simulate grid computing. lately, it is extended to support various cloud computing use cases such as multi-purpose network representation as well as virtualisation.

Kliazovich *et al.* [98] explore the energy-aware cloud computing data centres through simulation, modelling the energy consumption of all data centre components including servers, switches, and links. The simulation results showed that 66% of server energy consumption was by idle servers, and networking accounted for 30% of total energy consumption.

Table 2.2 represents that the simulation framework HTC-Sim is novel to model multi-use cluster with the presence of interactive users. Furthermore, HTC-Sim is one of the only very few simulations which simulate live migration, real workloads and fault tolerance mechanisms.

	Energy Model	**Multi-use Interactive users**	**Fault tolerance Checkpointing**	**Language**	**Virtualisation**	**Live Migration**
SimGrid [105]	✓	–	–	C	✓	✓
GridSim [53]	–	–	–	Java	✓	–
OptorSim [41]	–	–	✓	Java	–	–
SiCoGrid [123]	–	–	–	Haskell	–	–
Chi-Sim [48]	–	–	–	C++	–	–
CloudSim [55]	✓	–	–	Java	✓	✓
GreenCloud [98]	✓	–	–	C++/OTcl	✓	?
iCanCloud [132]	✓	–	–	C++	✓	?
MDCSim [108]	✓	–	✓	C++/Java	–	–
HTC-Sim [29]	✓	✓	✓	Java	✓	✓

Table 2.2: Comparison of simulators

The authors of [54, 77, 115] provide reviews on simulation frameworks for cloud environments.

2.3.1 HTC-Sim

High Throughput Computing (HTC) is powerful for a large number of jobs with a long period of execution time, where jobs can be executed over a distributed set of

computers. HTC systems such as HTCondor [110], BOINC [34], and SGE [83] are popular choices for academic as well as industry researchers, to do complex computational tasks on existing, idle, shared facility (desktop grid) or dedicated resources.

HTCondor [110] is an open source high throughput computing software which provides workload management system technology for grid computing jobs. HTCondor has a job queuing mechanism, scheduling policy, priority scheme, resource monitoring, and resource management. HTCondor chooses where and when to start running the jobs that are submitted by users dependent upon its policy. CycleServer and CycleCloud [2] are software tools provided by Cycle Computing, and are used for managing and accessing cloud resources. CycleServer manages and monitors the jobs in HTCondor, while CycleCloud provides access to the resources though web services such as Amazon EC2. These tools have been used at large-scale to provide a huge number of instances from a public cloud provider, which can reduce the time of running and the cost.

Forshaw *et al.* [78] introduced a high-level, trace-driven, simulation to evaluate the energy consumption and performance of different policies of high throughput computing (HTC) systems. The simulation tool can be used to evaluate new policies and to find the impact of these policies on the infrastructure itself. The tool and the results of the paper are based on the Newcastle University's HTCondor logs. The simulation framework, however, is generalisable to other HTC systems.

In [79], the authors extended the simulation to add checkpointing as a fault-tolerance mechanism. They provided checkpoint policies for determining the interval between the checkpoint in order to reduce energy consumption and overhead of the system.

McGough *et al.* [120] presented some policies which can be applied to multi-use clusters in order to reduce the energy consumption in HTC environments. The policies are evaluated by HTC-Sim, and are shown to be capable of savings up to 55% of the energy consumption of the high throughput system.

McGough *et al.* [121] used HTC-Sim to run an entire cluster on the Cloud with trace logs from the Newcastle University HTCondor system. They proposed policies governing the scheduling of HTC jobs to Cloud instances, aiming to reduce cost and overheads. However, their approach did not consider the energy impact of these poli-

cies on the Cloud provider. In Chapter 3, we evaluated the energy and performance impact of these scheduling policies by extending the simulator to model energy consumption. We then compared the energy consumption of the same workload on servers whose performance and energy consumption characteristics differ.

In [119] the authors used two different machine learning approaches, namely Random Forest and MultiLayer Perceptron, to predict the idle time of computers within a HTC environment. The prediction helped them to develop a scheduler that allocates jobs to computers which associated with the longest idle time. They used the HTC-Sim simulator to show that their approach can save up to 51.4% of the HTC system energy consumption. However, their approach does not work as a mechanism to handle job failures due to user interruptions.

In Chapter 6, we extend the HTC-Sim simulation to incorporate virtualisation. Moreover, we perform the pre-copy live migration algorithm to provide a test environment for job live migration in HTC system. We explained and evaluated our interval and responsive migration methods. We used a simple random policy to show the outcomes of each migration technique. In Chapter 7, we use the responses migration technique as a fault-tolerance mechanism with various migration policies.

2.4 Fault-tolerance

Fault-tolerance approaches are widely used in large-scale computing to prevent or handle hardware failures and software faults. From the literature, the fault tolerance mechanisms can be categorised as follows:

Reactive fault-tolerance: this mechanism occurs after the event of failure to mitigate the influence of the failure in the system. In this technique, the system frequently saves its state and uses it as a recovery to handle the failures. For instance, checkpointing is a well-known technique of reactive fault-tolerance.

Proactive fault-tolerance: this mechanism occurs before the event of failure to prevent system fails. The system needs to predict the failures by continuously monitoring. For instance, migration is a well-known technique of proactive fault-tolerance.

Paper	Fault-tolerance Category	Technique Name	Environment	Objective
Forshawet *et al.* [79]	Reactive	Checkpointing	HTC	Energy, Overhead
Aupy *et al.* [36]	Reactive	Checkpointing	HTC	Energy
Amoon *et al.* [33]	Reactive	Checkpointing, Replication	Cloud	Cost, Overhead, Availability
Nagarajan *et al.* [127]	Proactive	Migration	HPC	Overhead
Liu *et al.* [114]	Proactive	Migration	Cloud	Overhead, Network, Makespan
Engelmann *et al.* [72]	Proactive	Migration	HPC	Utilisation
Polze *et al.* [137]	Proactive	Migration	HPC	Response time, Availability
Responsive Migration	Responsive	Migration	HTC	Energy, Overhead, Makespan, Utilisation

Table 2.3: Overview of fault-tolerance techniques in large-scale computing

Table 2.3 summarises some fault-tolerance from the literature. We classify them by fault-tolerance category, the name of the technique, the environment which technique is applied, and the optimisation objective.

Checkpointing is a well-known approach to achieve reactive fault tolerance by regularly storing snapshots of application state into storage. Aupy *et al.* [36] proposed energy-aware checkpointing strategies for divisible tasks. They showed how to decide the number of chunks, their size and execution speed.

Amoon *et al.* [33] presented an adaptive fault-tolerance framework based on checkpointing and replication for cloud systems to enhance the performance of overheads, cost, and availability. Their framework selects the optimal fault-tolerance technique based on the customer's requirements. When there is more than one VM to carry out the customer's request, the framework applies replication as a fault-tolerance technique. The number of replications for each VM is based on the failure probability and the gained profit. Also, the checkpointing is applied when there is a single VM to handle the consumer request. The checkpointing interval is based on the failure probability of the VM.

Forshaw *et al.* [79] used a high-level, trace-driven, simulation to evaluate the energy consumption and performance of various checkpointing strategies in high throughput computing (HTC) systems. Thier checkpointing strategies help to decide when to make energy-efficient checkpoints within HTC systems without affecting the performance. In this thesis, we avoid checkpointing strategies and use VM migration strategies to avoid job failures.

Proactive migration is a well-known approach that has been used in virtualised envi-

ronments to prevent failures from impacting the performance of the system [72, 114, 127, 137]. To achieve this, the system monitors its nodes and predicts the nodes that are about to fail to migrate the tasks from them to other stable nodes. Nagarajan *et al.* [127] used an experiment for performing VM live migration as a proactive fault tolerance to migrate the task from unhealthy node to a healthy node in high-performance computing (HPC) environments. They used Xen hypervisor to create and migrate the VMs. Also, an Intelligent Platform Management Interface (IPMI) is used to monitor and manage the hardware. Their approach monitors the health state and checks for threshold violations. When the threshold violation is detected, they migrate the VM to another healthy node. Instead of doing a live experiment, here, we use a simulation tool to demonstrate the benefit of live migration in the event of user interruption failure.

Liu *et al.* [114] introduced a proactive coordinated fault-tolerance (PCFT) mechanism to predict a deteriorating physical machine. They monitor the CPU temperature in order to predict the deteriorating physical machine. Their approach automatically finds an optimal target machine to migrate the VM from the deteriorating machine which improves the overall performance of the system.

In Chapter 6 and 7, we present a responsive fault-tolerance mechanism. The mechanism occurs at the time of interactive user interruption in HTC systems to prevent the job from been evicted and restarted. Unlike the reactive fault-tolerance, the responsive fault-tolerance is ideal for short running and real-time tasks. Also, our method does not require saving images of the job states which reduce the system overhead and the responsive time of handling the fault. Furthermore, proactive fault-tolerance is based on the accuracy of the prediction. The system could fail to handle some faults which decrease the overall performance of the system. Further, the system might make unnecessary decision to prevent the faults. Our methods could mitigate the disadvantages of reactive and proactive fault-tolerance mechanisms when they used to handle interactive user interruptions in HTC systems.

3

Energy consumption of scheduling policies for HTC jobs in the Cloud

Contents

Summary

This chapter presents the extension of an existing cloud simulation system to incorporate energy measurement and evaluate the energy and performance impact of existing policies for scheduling HTC jobs to cloud resources. We demonstrate, through trace-driven simulation, the trade-off between energy consumption and system performance for a number of HTC scheduling policies. Furthermore, we incorporate the energy impact of networking infrastructure into our model of total facility energy consumption.

3.1 Introduction

High Throughput Computing (HTC) systems are a popular choices in organisations, frequently employed to leverage existing, idle, infrastructure to perform computation in an *'cycle stealing'* fashion. Latterly, these same approaches are increasingly used to meet peak requirements for computation by leveraging the scale offered by Cloud Computing [2] which could not otherwise be satisfied using local infrastructure alone.

Thus, it is very important that data centres improve the efficiency with which they use their power and a great deal of innovation has taken place to this end. For instance, cloud computing takes advantage of allocation and migration policies in order to promote the use of physical resources, and can change the amount of computing power allocated to users in line with their needs.

Existing research has considered the use of the Cloud in this context to meet exceptional capacity which cannot be met on local infrastructures [65, 117, 157], and also for running full workloads [66]. However, this is often without consideration for the energy consumption incurred by the cloud provider.

In this chapter, we build on previous work to evaluate the energy and performance impact of scheduling policies for HTC workloads to cloud resources. We employ the HTC-Sim simulation provided in [121], extending it to model energy consumption such that we may evaluate the energy impact of policies governing the scheduling of HTC jobs to cloud instances. Then, we compare the energy consumption of the same workload on servers whose performance and energy consumption characteristics

differ. In addition, we evaluate policies using trace data obtained from a University HTCondor high throughput cluster.

The rest of the chapter is structured as follows. Section 3.2 presents the policies we evaluate through this work. Section 3.3 discusses our adaptations made to HTC-Sim simulation environment, and our approached of calculating the energy utilisation. We present and evaluate the results of our preliminary experimentation in Section 3.4, before concluding and discussing future work in Section 3.5.

3.2 Policy

McGough *et al* [121] have previously proposed policies governing the scheduling of HTC jobs to cloud instances, aiming to reduce cost and overheads. In this chapter, we seek to quantify the energy impact of these policies on the cloud provider. The policies we investigate in this chapter are detailed below.

P1: limiting the maximum number of cloud instances: this policy seeks to limit the potential cost on the cloud by imposing a maximum number of instances which can be active at a given time. If this limit has been reached, any arriving jobs will be queued until a resource becomes available. This policy seeks to reduce cloud cost and cloud idle time, at the expense of increased overheads for jobs submitted during busy periods.

P2: merging of different users' jobs: for reasons of security, an individual Cloud instance is adept at performing a single job at any given time. Limits can be placed on an instance to consent to take jobs from only a single individual user, which the user that the instance was created for, or conversely enable the instance to agree to take all jobs from any user. Enabling users to share the cloud instances can lower costs as a lower amount of instances will be needed thus lowering overheads as jobs would be prone to detect idle instances which can be exploited. In policy P2, they relax this assumption and allow HTC jobs from multiple users to execute on the same running instance. This should reduce overheads, cost and energy consumption.

P3: instance keep-alive: in [84], the authors showed that initialising an instance can take up to 3 minutes. This policy enables an idle instance at the end of the billing

stage to stay 'hired' for the next stage with probability *f(p)*, allowing it to serve the latest next in line jobs more efficiently. It is problematic to predict the arrival of subsequent jobs; consequently there are three alternative policies that can determine if an instance needs to be kept alive and define *f(p)* for each:

Fixed: instances will be kept alive with probability *f(p)=p*.

Idle: instances will be kept alive with a probability proportional to the number of currently idle instances: $f(p) = (\frac{1-i}{t})p$, where i is the number of idle cloud instances at decision time and t is the total number of cloud instance at this time.

Load: instances will be kept alive with a probability proportional to the current load on the system:

$$f(p) = \frac{\int_{t-T}^{t} u_i di}{\int_{t-T}^{t} a_i di} p$$

Where t is the decision time, T is the interval of load evaluation , u_i the quantity of active cloud instances at time i, a_i is the number of hired cloud instances at time i, and di is the differential of the variable i.

Such a policy could have additional ramifications on overheads, and as such a commencing job would have an increased chance of arriving to an idle instance. The number of instance hours consumed is expected to increase due to some instances running in the absence of jobs.

P4: delaying the start of instances: this policy, seeks to minimise idle time arising as a consequence of short running jobs using only partial hours on cloud instances. Under this policy, jobs which arrive to the system which cannot be allocated to an idle instance are queued. If the jobs do not receive an instance within t minutes of submission, a new instance is then created. The policy should incur overheads but reduce cloud cost and energy consumption.

3.3 Simulation Environment

In this chapter, we use the HTC-Sim simulation [78] to evaluate the performance of job management policies of HTC jobs to cloud instances. We extend the existing simulation environment to evaluate the energy consumption of these policies. Section 3.3.1 discusses changes to the cloud resource model and Section 3.3.2 discuss metrics considered in the remainder of the chapter. Section 3.3.3 describes the scenario we simulate in the remainder of this chapter.

3.3.1 Resource Model

We extend the resource model of the existing simulation framework to include energy consumption and performance figures, as described below.

3.3.1.1 Energy consumption:

We select readily available metrics from the SPECpower benchmark to obtain energy consumption values for servers in idle, booting and working modes. These may be then multiplied by the amount of time spent in each mode in order to calculate total energy consumption of the system under a given policy. Here, we assumed the load level for booting and working jobs is 100%.

3.3.1.2 Performance scaling:

Based on performance measures provided by the SPECpower benchmark for each server, we are able to scale the execution time of our workload to more closely model the performance observable if the workload were run on that hardware. In doing so we assume the SPECpower benchmark to be representative of the original HTC workload.

We scale the duration of jobs in our workload as follows:

$$D'^{s}_{j} = D_j \times \frac{P_b}{P_s} \tag{3.1}$$

where P_b is the baseline server performance, P_s is the performance of one of the other selected servers and D_j is the job duration which is the start time job minus end time job.

3.3.2 Metrics

Here, we outline a number of key metrics reported by the simulation, which are subsequently used in Section 3.4 to evaluate the policies outlined in Section 3.2.

3.3.2.1 Overhead

Our main performance metric is average overhead for jobs within the system, which is calculated as the difference between the execution time of the job D'^{s}_{j} and the amount of time the job took to run in the simulation. The job will first face an idle period before it is allocated to a cloud instance. The idle period could be due to time waiting for a cloud instance to become available or policy decisions within the system. Hence, we cannot consider using the ratio of the overhead as it may lead to divide by zero error condition.

3.3.2.2 Cloud hours

We report the number of *'instance hours'* arising from each policy under investigation. This chapter focuses primarily on provider-side so we do not provide financial cost to users; however, this may be trivially calculated by multiplying the number of instance hours by the hourly price.

3.3.2.3 Infrastructure Energy Consumption

We further extend the simulation to make use of the industry-standard Power Usage Effectiveness (PUE) metrics [40] to calculate the total facility power. The PUE metric provides a ratio of energy consumption attributed to computing equipment compared to the infrastructure required to support it, including cooling and air conditioning. Equation 2.1 which is previously discussed in Section 2.1.2 defines the PUE metric.

We go further to incorporate the energy impact of networking infrastructure into our model of total facility energy consumption. While other efforts [98] have performed detailed analysis of the networking available in data centre environments, we chose to leverage an observation by Brown *et al* [52] that approximately 5% of the average data centre's total power use is absorbed by the network. We included this supposition into our simulation to acquire an estimation of the total facility power.

3.3.3 Simulation Scenario

We evaluate the scheduling policies which discussed in Section 3.2 using historical logs for 409,479 completed jobs from the HTCondor cluster located at Newcastle University, to calculate the energy consumption by applying varying job management policies of HTC jobs to cloud instances.

No	Server Name	Cores	Peak Power (W)	Idle Power (W)	ssj_ops	Scaling ratio	Test Date
1	PRIMERGY RX2560 M1	36	267	40.1	3,256,040	3.66	Mar-15
2	Express5800/A1080a-E	64	1749	993	3,647,000	4.10	Dec-10
3	ProLiant DL385 G7	48	271	101	888,819	1.00	Mar-10
4	PRIMERGY TX150 S7	4	112	24.3	276,514	0.31	Jan-10
5	Proliant DL580 G5	16	387	271	359,523	0.40	Dec-07

Table 3.1: Selected server from SPECpower_ssj 2008 published [1]

Table 3.1 shows the server types which we consider in this chapter. We select Server 3 (*'ProLiant DL385 G7'*) to represent the baseline server against which task duration is scaled as in Equation 3.1. Server 3 was selected as the original HTCondor dataset originates from 2010, and the chosen server exhibits close to average peak power and performance characteristics from all published SPECpower results in 2010, as shown in Figure 3.1. The remaining servers were selected to offer a variety of performance and power characteristics, with this diversity of machines helping us indicate which policy is the best fit for the server based on its performance and energy consumption. In addition, the scaling ratio ($\frac{P_b}{P_s}$) is used to scale the duration of jobs in our workload as previously mentioned in Section 3.3.1.2, where P_b is the baseline server performance (ssj_ops) and P_s is the performance (ssj_ops) of one of the other selected servers.

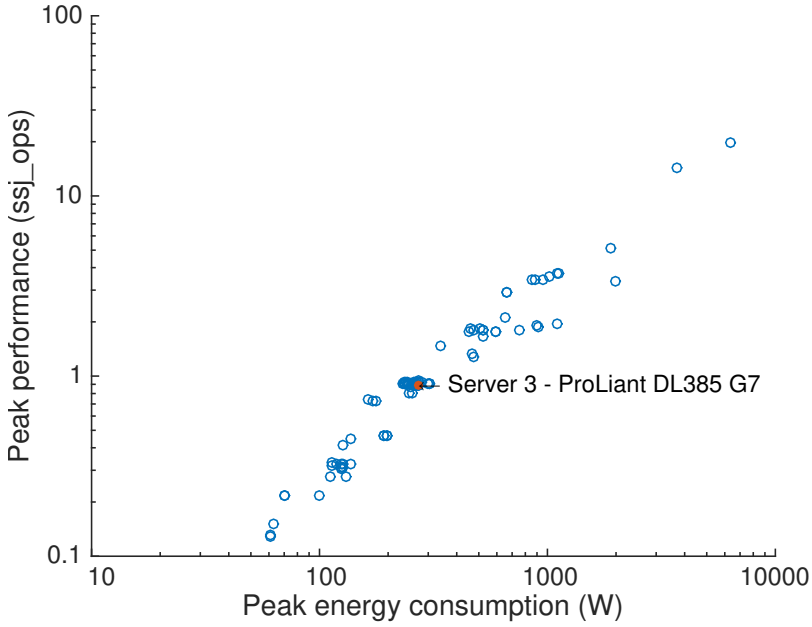

Figure 3.1: Performance to power ratio of SPECpower servers results in 2010

3.4 Results

In this section, we present the results of our simulation, and explore the performance and energy impacts of the policies outlined in Section 3.2.

3.4.1 Limiting the number of cloud instances

Figure 3.2 shows the impact of varying the maximum number of cloud instances. We see significant reductions in average overheads through increasing the instance count. The impact of the different performance levels of the servers is evident in these policies, with overheads much lower for servers such as 1 and 2 which exhibit better performance, and greater overheads observed for slower servers.

When considering the number of instance hours (hence cost incurred), we see for each of the computers that by 150 for higher performance servers, and by 300 instances for slower servers, the number of instance hours consumed plateaus and does not increase much further. Similar trends are evident when considering the energy consumption, with the exception of servers 4 and 5, which see an initially high energy consumption for lower number of instances, due to their substantially lower performance than the other benchmark servers.

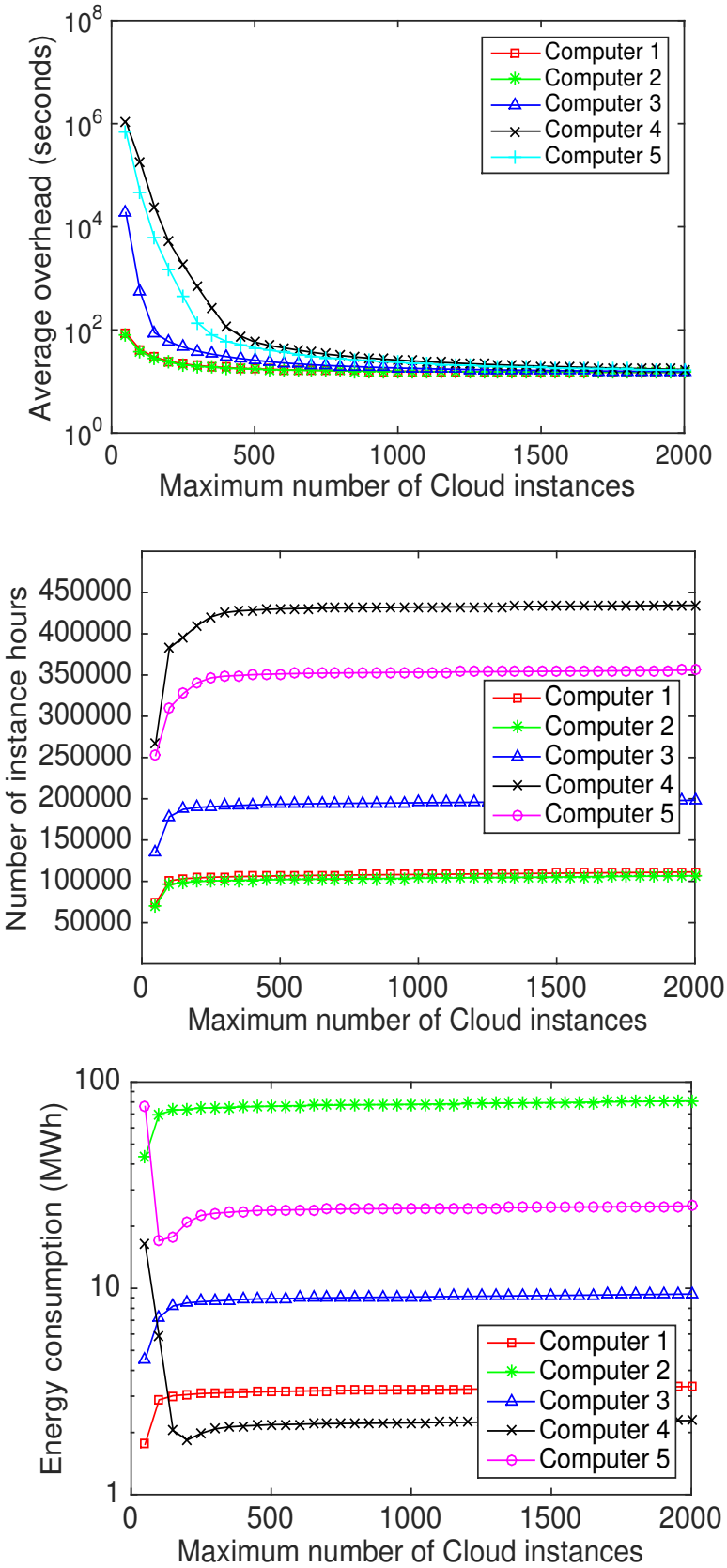

Figure 3.2: Impact of policy P1 on average overhead, number of instance hours, and energy consumption.

3.4.2 Merging of different users' jobs

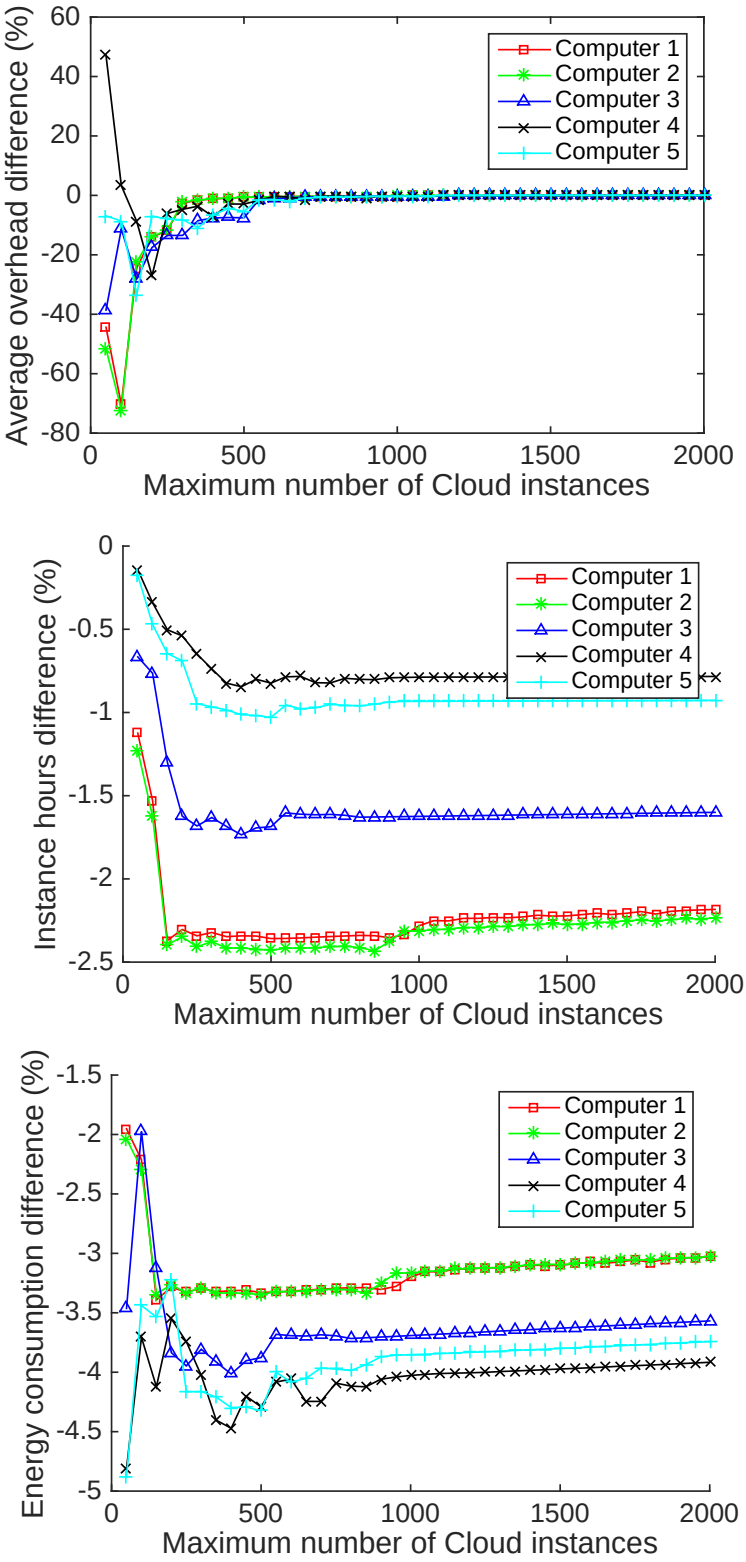

Figure 3.3: Impact of policy P2 on average overhead, number of instance hours, and energy consumption.

In Figure 3.3, we demonstrate the gains possible by merging workloads from multiple users, for policy P1 varying the maximum number of cloud instances. We see the merging of workloads leads to reduced average overheads, with jobs more likely to enter the system to find an available idle instance.

We acknowledge significant variability for this policy due to random effects in the simulation. In the simulation, the jobs and computers are selected randomly. For that reason, we can see high variation in some of this policy results when we compare them with the previous policy results.

As the maximum number of cloud instances increasing the average overheads tends towards zero. We see policy P2 is capable of modest reduction of the number of cloud hours required. Similarly, we observe general trends that policy P2 is capable of reducing energy consumption. Both the difference in number of instance hours and difference in energy efficiency will tend to zero as the number of cloud instances increases.

3.4.3 Instance keep-alive

In Figure 3.4, we explore the impact of policy P3, which governs whether instances are to be kept-alive and remain active during a subsequent billing period to serve jobs arriving jobs. Here, we show results for our baseline server (*ProLiant DL385 G7*) and a maximum instance count of 500. These results are representative of those for the other servers we consider.

Our results acknowledge this policy reduced average overheads significantly only under the assumption that cloud instances take 10 minutes to provision, with benefits minimal for smaller provisioning durations. Also, the results confirm the authors' observation of this policy in [121], and we further demonstrate the policy leads to increased energy consumption.

Figure 3.5 demonstrates the relative impact of the three variants of policy P3. The 'Fixed' policy clearly offers the most compelling overhead reductions but at the expense of the number of instance hours (and hence cost) consumed, and also has the most detrimental impact on the energy consumption of the system.

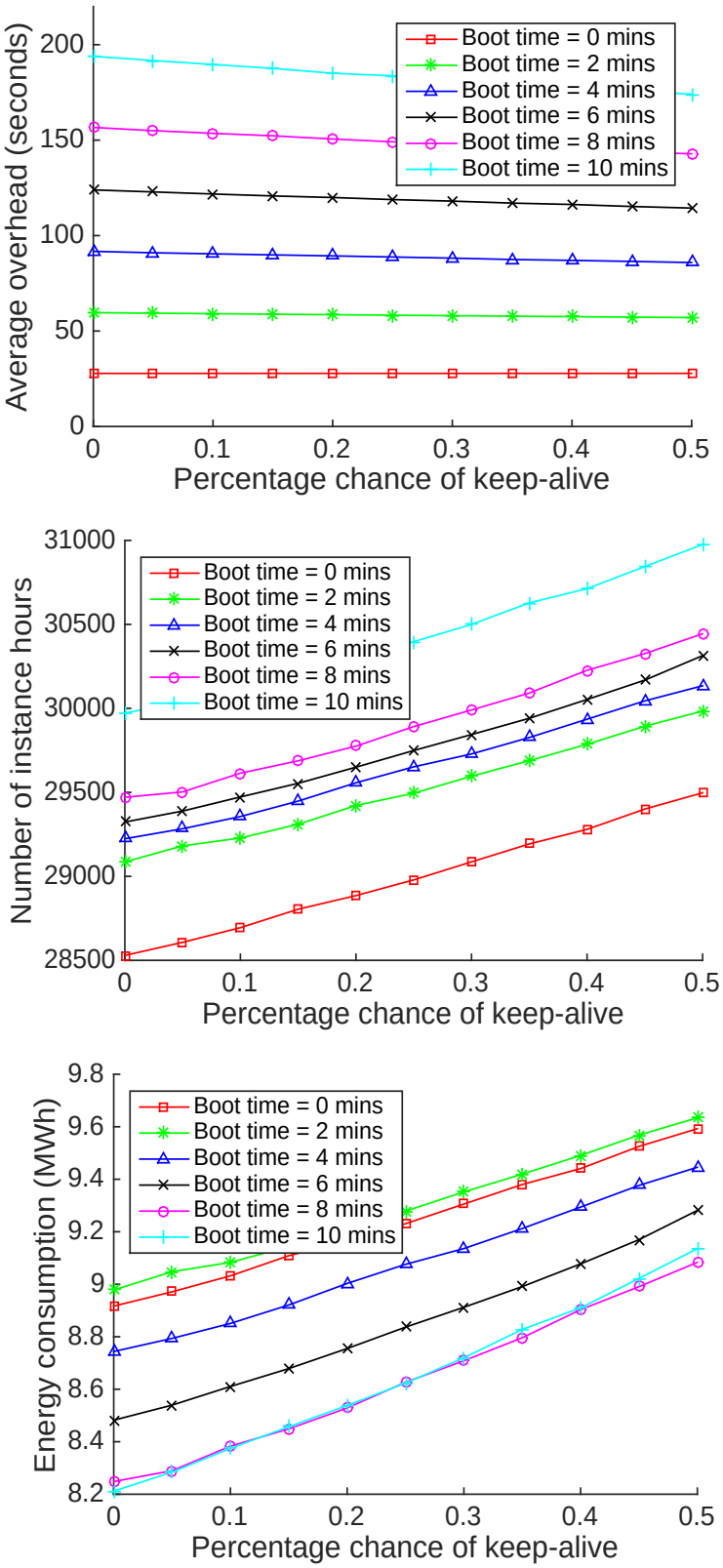

Figure 3.4: Impact of policy P3 on average overhead, number of instance hours, and energy consumption. Varying chance of keep-alive and boot time for Server 3.

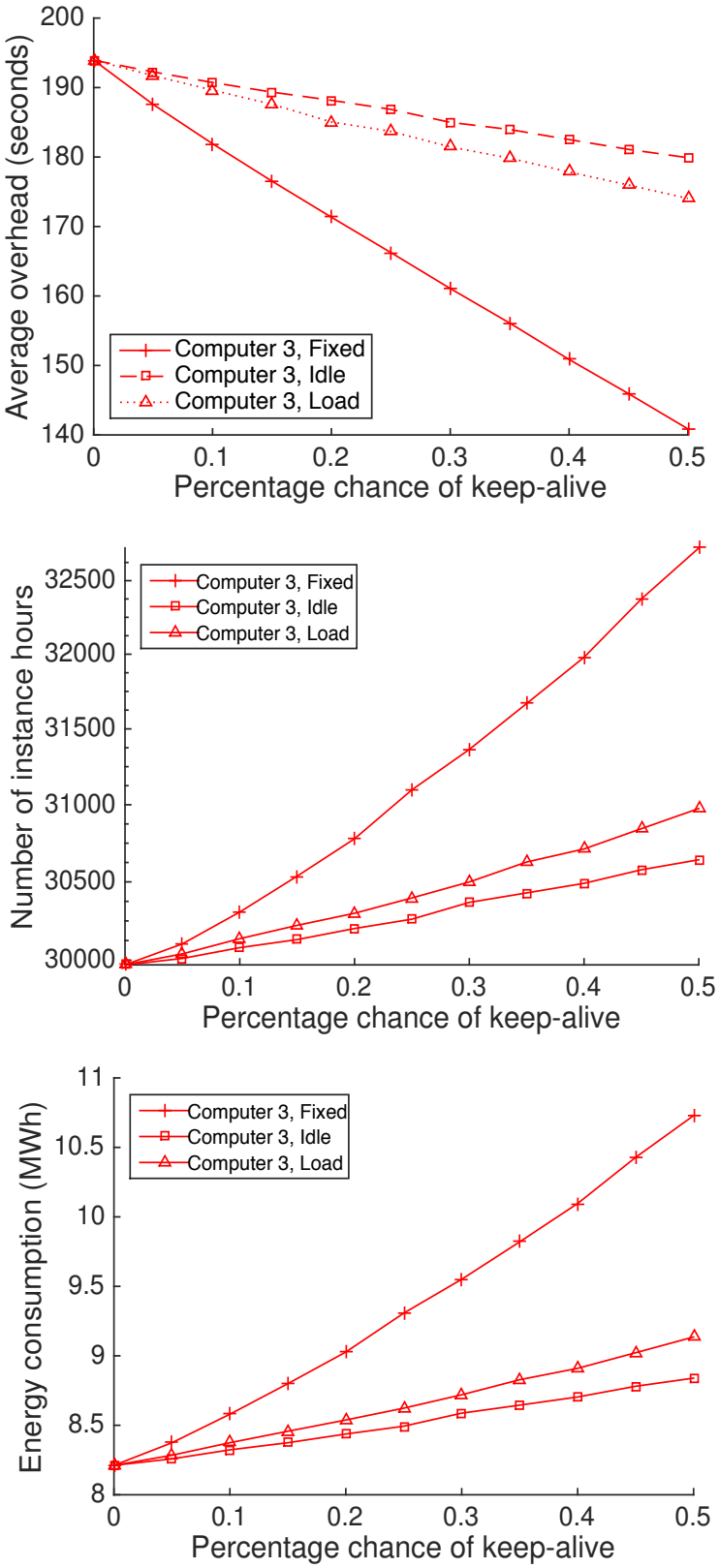

Figure 3.5: Impact of policy P3 on average overhead, number of instance hours, and energy consumption. Varying chance of keep-alive for a boot time of ten minutes.

The 'Idle' policy is the most promising policy, in that is is capable of reducing average overheads while incurring the least negative impact on the other metrics we consider. The 'Load' policy is shown to perform better in terms of overhead reduction, but at the expense of instance hours used and energy consumption.

3.4.4 Delaying the start of instances

Figure 3.6 demonstrates the impact of policy P4 on average overhead, number of cloud hours consumed and energy consumption. The policy seeks to reduce the negative impact of short-running jobs on the system, which would otherwise occupy only partial hours on a resource. By delaying the creation of a cloud instance, it is hoped that subsequent jobs will arrive and be able to make use of idle time on the hired cloud resource.

We demonstrate results here with a maximum cloud instance limit of 500, and vary the maximum job delay between zero and thirty minutes. We demonstrate that as expected, this policy results in a detrimental on average overheads, but leads to reductions in the number of instance hours consumed and hence energy consumption. There is clearly a trade-off between overheads incurred for the HTC workload and cost/energy consumption which must be reconciled.

Focusing on the impact on energy consumption, we see for each of the computers considered, imposing a maximum job delay of 30 minutes results in reductions of over 50%, but in exchange for an intolerable increase in task overheads. Imposing a delay of up to 5 minutes would reduce energy consumption by 20% in exchange for slightly less than a doubling in average overhead, which would appear more acceptable.

3.5 Conclusions

This chapter has evaluated the energy consumption of a number of policies governing the scheduling of a HTC jobs to cloud instances. These policies exhibit varying impacts on energy consumption and average overheads, and in all cases have demonstrated the criticality of the trade-off between energy consumption and performance/cost.

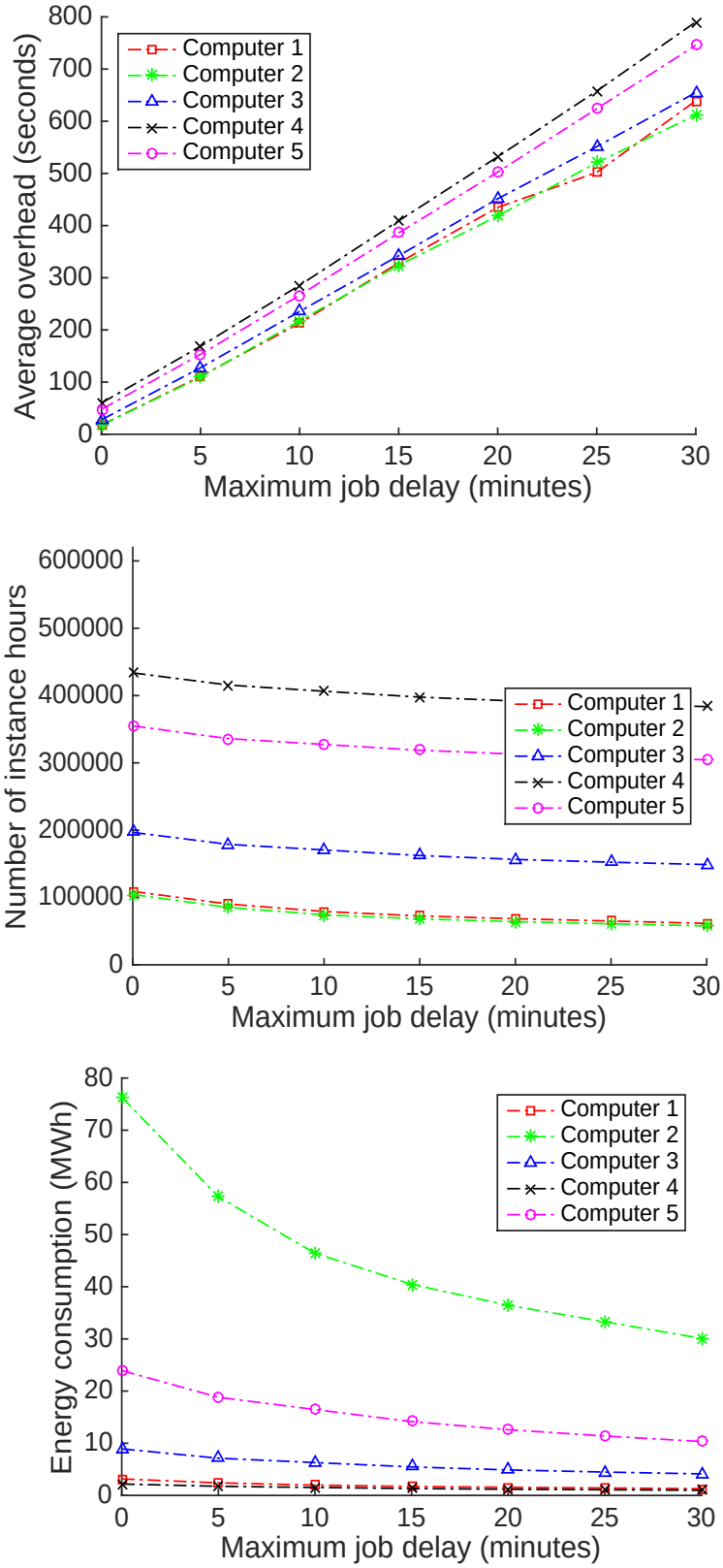

Figure 3.6: Impact of policy P4 on average overhead, number of instance hours, and energy consumption.

We evaluate policies under a number of assumptions made about the hardware used within a cloud data centre, and through this work have extended an existing simulation tool to allow practitioners to reason over resource allocation decisions in a cloud setting. The development of this tool will ultimately support the development and evaluation new algorithms for HTC workload scheduling in cloud environments.

In Chapter 6, we discuss our extension to HTC-Sim to support the modelling of virtualisation and live migration. This allows us to evaluate policies governing the consolidation of HTC workloads onto a virtualised environment. Also, we provide a foult-tolerance technique to job handle failures by using live migration. In Chapter 7, we evaluate classes of policy which leverage differences in clusters to make a selection on which to use. However, in order to implement virtualisation and live migration in HTC environment, we need to understand the factors that influence the time of live migration. As we explained in the previous chapter, many existing research works did not consider the cost of the live migration and assumed that live migration can always be achieved. In the next chapter, we investigate through an experiment the factors which influence the time of VM live migration.

4

Performance of Virtual Machine Live Migration with Various Workloads

Contents

Summary

This chapter introduces our approach for measuring the live migration time of a virtual machine (VM). We demonstrate through live experiment the link between various workload characteristics and the time of VM live migration. We present details of the live experiment as well as the benchmark which used to generate various workloads. We present and evaluate the experimental results of various VMs capacities. Finally, we provide a discussion regarding the experimental limitations.

4.1 Introduction

Cloud computing has been achieved mainly due to the ability of modern equipment and large-scale manufacturing processes that are able to deliver inexpensive, convenient and user-friendly products to consumers all over the world. Virtualisation is a core component of cloud computing which powers the cloud due to various benefits such as partitioning, isolation, easy manageability, cost efficiency and flexibility.

Virtual machine live migration is one of the features that is provided by the hypervisor. Live migration refers to the procedure of moving a running VM between physical hosts without powering down the VM. It widely used in data centres due to its ability of energy management, load balancing, and fault tolerance [60]. However, some of VM management approaches that take advantage of live migration such as VM consolidation do not consider the cost of VM live migration. In some cases, the energy consumption of VM live migration can be higher compared to keeping the VM running in its current physical machine. It is important to know the parameters that influence the cost of migration before starting the migration of VMs to ensure the most significant decision on saving energy.

In the previous chapter, we have evaluated the energy consumption of HTC scheduling policies applied to a cloud-based cluster. However, the virtualisation and migration are not considered nor included in the simulation. In order to consider virtualisation and migration in a HTC environment, we need to conduct an experiment to understand the aspects associated with the live migration process.

In this chapter, we perform a live experiment to measure the duration time of the VM live migration. We use two physical hosts to migrate VMs between them by using Kernel-based Virtual Machine (KVM) [4]. We choose SPECjvm2008 benchmark [10] to generate workloads on the VMs due to its ability to produce various workloads characteristics. Then, we compare the migration time of the same workloads on VMs whose hardware characteristics differ.

The rest of the chapter is structured as follows. In Section 4.2 we present the experiment environment. We present and evaluate the results of our preliminary experimentation in Section 4.3, before concluding and discussing future work in Section 4.4.

4.2 Experiment environment

In this section, we use a live experiment to measure the time of VM live migration with various workloads. We use the Kernel-based Virtual Machine (KVM) [4] as a hypervisor and SPECjvm2008 [10] benchmark to generate various workloads. Section 4.2.1 discusses set up of the experiment and Section 4.2.2 introduces the benchmark that generates VM's workloads, while Section 4.2.3 describes the scenario we experiment in the rest of this chapter.

4.2.1 Experiment set up

The principal obligation which has influenced the experimental setup is that it should reproduce the VM live migration procedure as it takes place in real-world contexts. For this reason, the Kernel-based Virtual Machine (KVM), a common hypervisor for data centres, has been employed. VM live migration necessitates the storage of VM images in a dedicated area that can be accessed by every physical host. Consequently, VM images are stored using network attached storage (NAS), and this has created a situation in which the procedure of VM live migration is restricted to copying the memory pages and the CPU state among physical hosts; this method is referred to as pre-copy live migration. For NAS, we used the free software licensed Openfiler [6]. The Openfiler is an operating system which delivers file-based network-attached storage (NAS). We installed it on a virtual machine by using VMware, and it employs 60 GB

disk space and 8 GB virtual RAM. In addition, the setup involves two servers and one client computer, linked together by a 100Mbps switch. The servers run CentOS 7 Linux [13] and KVM is installed on each. The hardware features of both servers as follows: server 1 employs 4 CPUs Core 2 Quad @ 2.66GHz and 4 GB DDR2 SDRAM. Server 2 has 8 CPUs Intel Core i7 @ 2.80GHz and 4 GB DDR3 SDRAM. The client computer operates Ubuntu 16.04 LTS [14] and employs 1 CPU Intel Core i7 @ 3.20GHz and 4 GB DDR3 SDRAM. The difference between the two servers is a factor that assists us to obtain various results. Finally, it should be noted that the client computer is employed to trigger VM live migration between the servers.

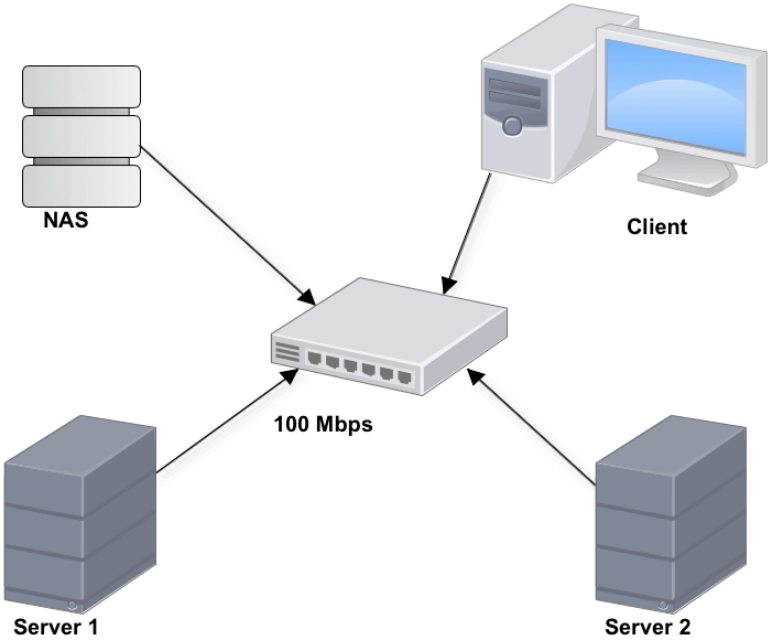

Figure 4.1: The setup of VM live migration experiment with 100Mbps switch

4.2.2 Benchmarks

As we mentioned in Section 2.2.3.1, some approaches considered using an identical workload which stresses the memory or CPU. We avoid this limitation by choosing the SPECjvm2008 benchmark. The SPECjvm2008 benchmark [10] includes 39 distinct workloads that test the performance of Java virtual machines (JVM) and hardware systems such as processor and memory. The default run time for each workload is four minutes. The benchmark contains two running modes: base and peak. The base has a fixed running time duration which is 120 seconds warm-up, continued by 240

seconds. In our experiment, we use 21 workloads of the SPECjvm2008 benchmark as illustrated in Table 4.1. Also, we use the base mode to run the benchmark without warm-up time duration. The remaining part of this section gives a brief description of each workload as mentioned in [149].

Table 4.1: SPECjvm2008 Benchmark workloads

Group name	**Workloads**
Compiler	compiler.compiler, compiler.sunflow
Compress	compress
Crypto	crypto.aes, crypto.rsa, crypto.signverify
Derby	derby
Mpegaudio	mpegaudio
Scimark Large	scimark.fft.large, scimark.lu.large, scimark.sor.large, scimark.sparse.large, scimark.monte_carlo
Scimark Small	scimark.fft.small, scimark.lu.small, scimark.sor.small, scimark.sparse.small, scimark.monte_carlo
Serial	serial
Sunflow	sunflow
Xml	xml.transform, xml.validation

Compiler: There are two workloads within the compiler group, namely compiler and sunflow. The compiler.sunflow workload determines the sunflow benchmark's compilation, while the OpenJDK compiler's compilation time is measured by the compiler.compiler workload. Input data is stored either in a file cache or in memory to reduce the impact of I/O as the aim of these two workloads is the evaluation of the compiler's performance.

Compress: The workload uses a modified Lempel-Ziv technique to compress and decompress data. The algorithm utilizes pseudo-random access based on the input data, which is extended to 34.36 MB from 90 KB. Data is buffered to

minimise the impact of I/O and the compression is done using internal tables of approximately 67 KB. As the JVM produces and operates on mixed length data accesses, the compress workload tests inlining, array access, just in time coupling and cache performance.

Crypto: There are three workloads within the crypto group: crypto.signverify, crypto.aes and crypto.rsa. Between them, they cover three important aspects of cryptography and test not only JVM execution but also different vendors protocol implementations.The crypto.rsa workload works on input data of 16 KB and 100 bytes and encrypts and decrypts using the RSA protocol. The crypto.aes workload encrypts and decrypts data. This is done according to the AES and DES protocols, using CBC/PKCS5P adding and CBC/NoP adding. The respective input data sizes are 100 bytes and 713 KB, respectively. The crypto.signverify, as its name suggests, signs and verifies protocols. In particular, it does this with SHA1 with DSA, SHA1 with RSA, SHA256 with RSA and MD5 with RSA, for input data sizes 1KB, 65 KB and 1 MB.

Derby: An open source, pure Java database called derby is used by this workload; several databases are instantiated each time the workload is started, with every four threads sharing a common database instance. Derby tests synchronization, database and BigDecimal operations. It took forward the telco benchmark of IBM such that it could synthesize business logic and test BigDecimal operations. The BigDecimal calculations in this workload are longer than 64-bit.

Mpegaudio: The mpegaudio workload is based around floating-point calculations and uses as an MPEG audio decoder the JLayer MP3 library. The input data files, whose sizes range from 20 KB to 3 MB, are six MP3 files.

Scimark: Scimark comprises a group of workloads that together evaluate data access patterns and floating-point operations in demanding mathematical calculations. It is based around the scimark workloads are arranged into two groups scimark.small and scimark.large, according to the dataset size. Each workload thread uses one dataset; the small group uses a 512 KB dataset to simulate the performance of in-cache access whiule the large group uses a 32 MB dataset in or-

der to reproduce the out of cache access performance. Each group comprises five workloads. These are monte_carlo, sparse, lu, fft and sor. scimark.monte_carlo runs once but is counted in both scimark.large and scimark.small; the workload does not operate on differently sized datasets.

Serial: This workload exercises the java.lang.reflect package and examines the serialization and deserialization of primitives and objects. The performance of these processes is evaluated using a dataset taken from a JBoss benchmark in memory byte arrays. Serial acts in a producer-consumer situation, in which the producer threads serialize the objects while, on the same system, the consumer threads deserialize them.

Sunflow: The sunflow workload is multi-threaded and runs a number of bundles of dependent threads. The workflow is reconfigurable. However, generally, there are four threads per bundle and as many bundles as there are hardware threads. Additionally, being floating-point intensive, the workload has a high object allocation rate, exercising the memory bandwidth. It is used as a benchmark simulating visualisation and graphics using ray tracing.

XML: Two workloads, xml.transform and xml.validate, constitute the XML group. Both have high rates of contended locks and object allocation and exercise string operations intensively. By performing XSLT transformations with SAX and DOM stream sources, xml.transform exercises the JAXP implementation. The workload utilizes ten use cases from real life and the XSLTC engine (this compiles xsl stylesheets into java classes). The xml.validation workload also exercises the JAXP implementation and uses just six use cases from real life.

4.2.3 Experiment scenario

We created a VM on server 1 by using KVM, and the image of the VM is saved in NAS. The VM runs Ubuntu 16.04 LTS as an operating system, and it has the SPECjvm2008 benchmark to generate various workloads. We used three different VMs' hardware capacities as shown in Table 4.2. We call them VM 1, VM 2, and VM 3.

Table 4.2: VMs specifications

VM Name	Number of Virtual CPUs	Memory Size (GB)
VM1	1	1
VM2	2	2
VM3	3	3

The experiment employs 21 SPECjvm2008 workloads, and each is operated independently. Once the first minute of workload operation has been completed, the VM live migration procedure is started to transfer the VM from Server 1 to Server 2. After the VM has been migrated to Server 2, the VM gets restarted and starts the previous steps again between Server 1 and Server 2. We repeat the procedure ten times for each workload within each VM type. The steps of the experiment are summarised in Figure 4.2.

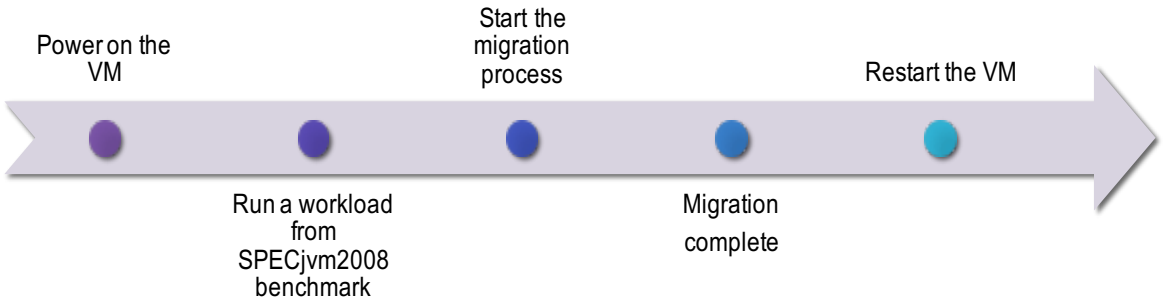

Figure 4.2: A flowchart of the experiment scenario

The presence of the client computer is important in facilitating the operation of the experiment in an automated way. We developed a script [28] that enables the client computer to run the benchmark on the VM and, following this, the initiation of the live migration between the two servers. The client accesses the VM and commences the operation of a single workload; following one minute of running the workload, the client facilitates the migration of the VM from the first to the second server; in turn, the commencement and completion times of the VM live migration are recorded in a log file. In addition to this, the script maintains a log of the memory usage of each workload over the course of its runtime from second to second, and this is measured by the `Memusg` script [5]. Furthermore, `Top` [16] is used to maintain a log of the CPU

utilisation and memory usage of the VM during the migration.

For future researchers who are interested in reproducing our test using their own hardware, a copy of the automated script can be found at [28]. It is possible to modify both the number of the test and the runtime of each workload.

4.3 Results

In this section, we present the results of our preliminary experimentation which show the different workload characteristics impacts of VM live migration time on various VM capacities.

Table 4.3 presents the results of VMs migration from Server 1 to Server 2. After one minute of executing the workload, the migration process starts. We migrated each VM with each workload ten times from server 1 to server 2 to obtain the results.

Table 4.3: Average migration time and memory usage of VM 1, VM 2, and VM 3

	VM 1		VM 2		VM 3	
Workloads	Migration Time (Sec)	Memory usage (KB)	Migration Time (Sec)	Memory usage (KB)	Migration Time (Sec)	Memory usage (KB)
compiler.compiler	231	219202	268	639368	317	909196
compiler.sunflow	222	190620	232	358673	264	470702
compress	210	115873	212	159170	228	226510
crypto.aes	209	81937	215	213467	237	330427
crypto.rsa	207	92021	208	108605	213	119630
crypto.signverify	208	78373	210	116095	221	141932
derby	442	345463	594	539221	630	626739
mpegaudio	207	100632	210	118431	217	132093
scimark.fft.large	212	169702	254	533904	269	639192
scimark.lu.large	223	227558	262	490671	293	738374
scimark.sor.large	211	118444	221	183212	233	217745
scimark.sparse.large	218	169940	219	236246	283	725774
scimark.fft.small	106	76095	209	94235	220	99275
scimark.lu.small	210	78432	210	99344	225	106371
scimark.sor.small	209	81134	218	137706	240	227848
scimark.sparse.small	210	80910	210	94887	219	97246
scimark.monte_carlo	208	78741	209	94415	221	102125
serial	209	158552	227	362733	246	503948
sunflow	207	89595	236	277815	235	403761
xml.transform	228	136392	255	325566	280	419212
xml.validation	210	126865	271	572309	272	745437

The table shows the average live migration time for each workload and VM type in seconds as well as the average peak memory usage in Kilobyte (KB). The results shows a clear variance in average migration time and memory usage between workloads within

each VM type due to the different workload characteristics. Also, the results show significant increases in average migration time when the size of memory is increased. The reason is that live migration process copies the memory pages from the source host (server 1) to the destination host (server 2). When the memory size of VM is increased, the time of live migration increases due to the increment on memory pages. In our experiment, VM 2 and VM 3 take more time to be migrated than VM 1 due to the memory size of VM 2 and VM 3 which are bigger than the memory size of VM 1.

We should mention that the number of operations in the SPECjvm2008 benchmark is based on the available resources. When there are more resources available in the server, the workloads generate more operations to test these resources. In our experiment, the workloads in VM 2 and VM 3 generates more memory pages than VM 1 due to the fact that they contain a bigger memory size compare to VM 1.

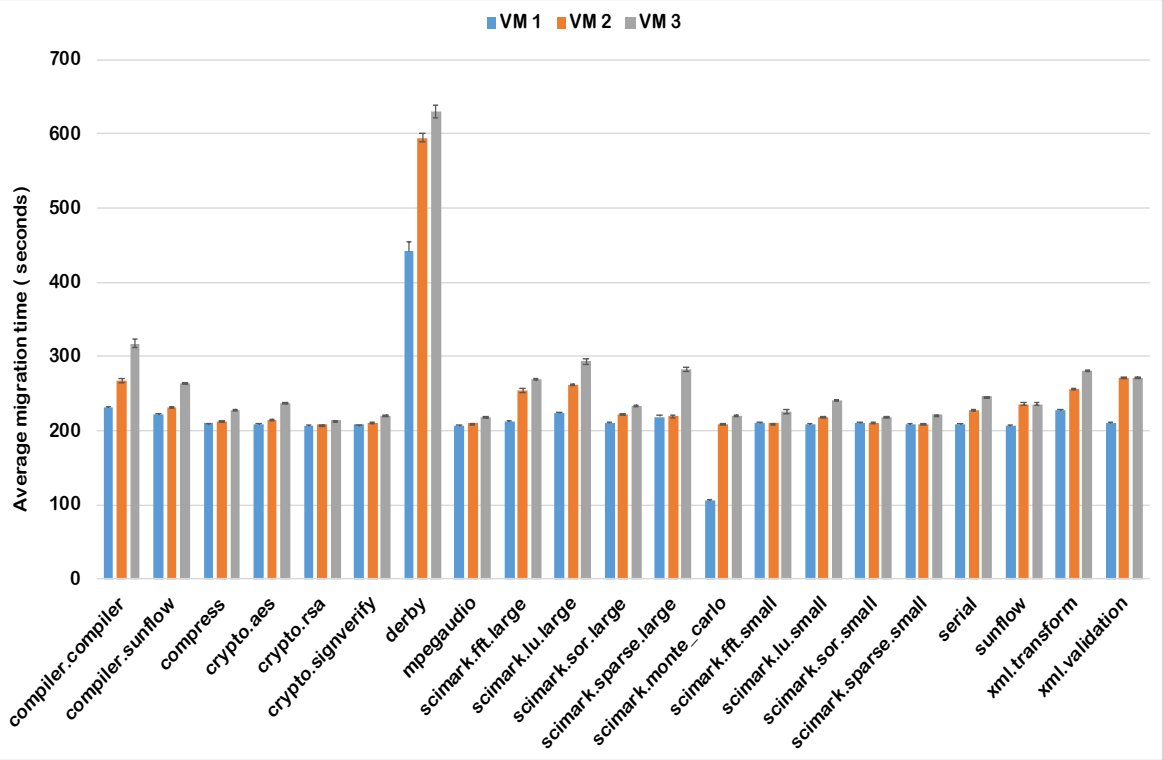

Figure 4.3: Average migration time of VM 1, VM 2, and VM 3 with various workloads

Figure 4.3 illustrates the VM average migration time of VM 1, VM 2, and VM 3 for each workload. The average migration time increases when the VMs resource capacity increases. Also, the figure shows that the migration process finishes after the execu-

tion of workloads except for `Derby` and `Scimark.monte_carlo`. The `Derby` workload migration time much exceeds the workload execution time. The system takes time after the `Derby` workload finishes backing into idle state. The `Scimark.monte_carlo` workload can be migrated during its execution time if the VM type is VM 1. The `Scimark.monte_carlo` workload does not generate many memory pages when the memory size is 1 GB. In addition, the standard deviation bars in the figure are shown for all workloads. The standard deviation is used to show how much the values in the data vary from the mean which is presented as black bars on the top of the histogram. Here, we do not observe high variation between the runs within each experiment setup.

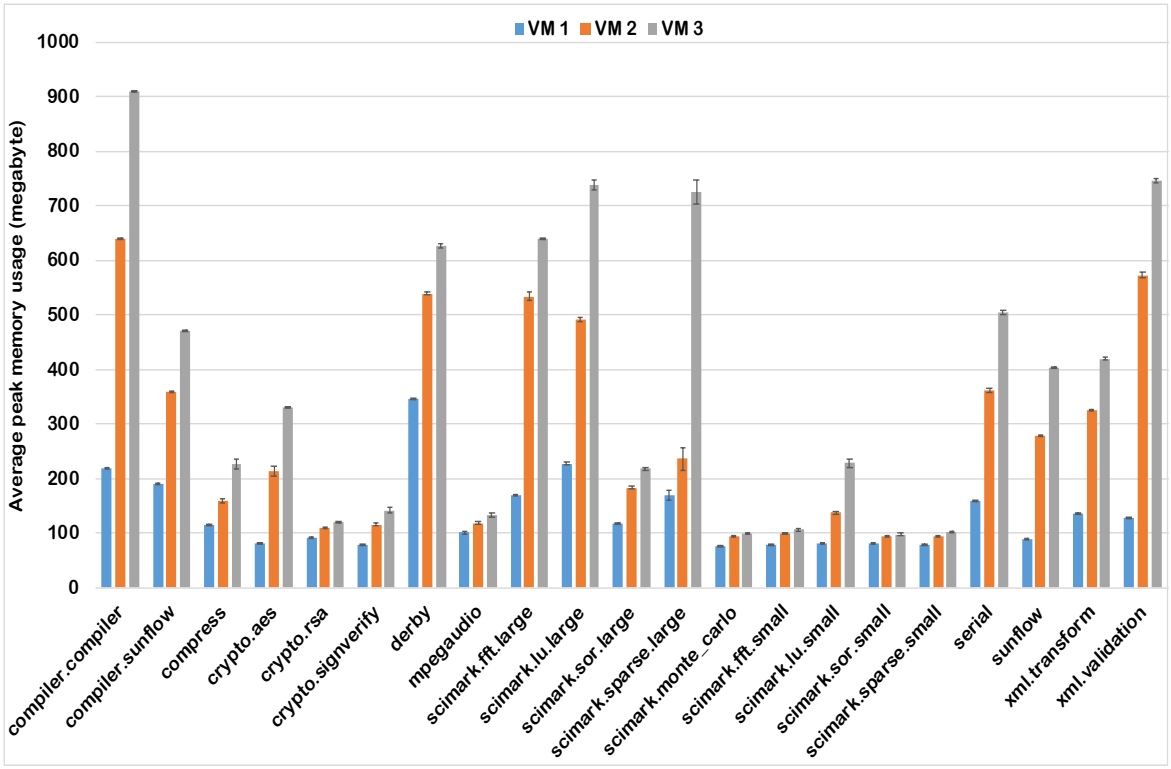

Figure 4.4: Average peak memory usage of each workload on various VMs

Figure 4.4 shows the average peak memory usage of each workload on each VM. The results do not show a link between the migration time of the VMs and the peak memory usage of the workloads. For example,the `Compiler.compiler` workload in VM 2 and VM 3 uses more memory than the `Derby` workload, but the VM with the `Derby` workload takes a longer time to be migrated than the VM with `Compiler.compiler`. Also, the results from one VM do not present a high variation in migration time

between the workloads event though the workloads memory usage are different. For example, the `Compress` and the `Crypto.aes` workloads on VM 1 have similar migration time, but the `Compress` workload uses more memory. Also, the standard deviation bars are shown for all results where we see that average peak memory usage exhibits low variance, across each benchmark and VM configuration.

This encouraged us to investigate more to find out the reason behind that. We modified our script to record the CPU utilisation and the memory usage of the VM during workload running time and migration procedure. Figure 4.5 shows the memory usage of the VM 1 with `Compiler.compiler`, `Crypto.aes`, `Derby`, `Scimark.lu.large`, `Scimark.sor.small`, and `Xml.transform` workloads. The VM migration starts after 60 seconds of running the workload and the workload runs for 240 seconds. As illustrated in Figure 4.5, the memory usage of the VM 1 starts to rise when the workload starts and drops when the run time of workload is finished except `Derby` workload.

During the live migration procedure, the VM remains running on a source host until it suspends and moves to a destination host. In our experiment, we observed that the VM continues running on Server 1 till the memory usage of the VM drops then it migrates to the Server 2. This is the reason for the low variation in the average migration time between the workloads within one VM as well as the high average migration time of VMs with `Derby` workload.

In this experiment, we use 100 Mbps switch to connect the servers and client computer as mentioned in Section 4.2.1. The available bandwidth in the network is 94.6 Mbits/sec. The VM live migration requires moving the memory pages from the resource host to the destination host. At the point when there is a high rate of changing the memory pages during the live migration with low network speed, the migration might never finish or might take a long time. We changed the default run of the workloads to be 1800 seconds instead of 240 seconds. The VM live migration is still finished after a few seconds of the workload execution where the memory usage drops.

In addition, we looked at the CPU utilisation of the VMs during the runtime of the workload and migration procedure. The CPU utilisation of workloads with one VM is about the same except `Derby` workload. Figure 4.6 exhibits the CPU usage of the VM 1 with `Compiler.compiler`, `Crypto.aes`, `Derby`, `Scimark.lu.large`,

`Scimark.sor.small`, and `Xml.transform` workloads.

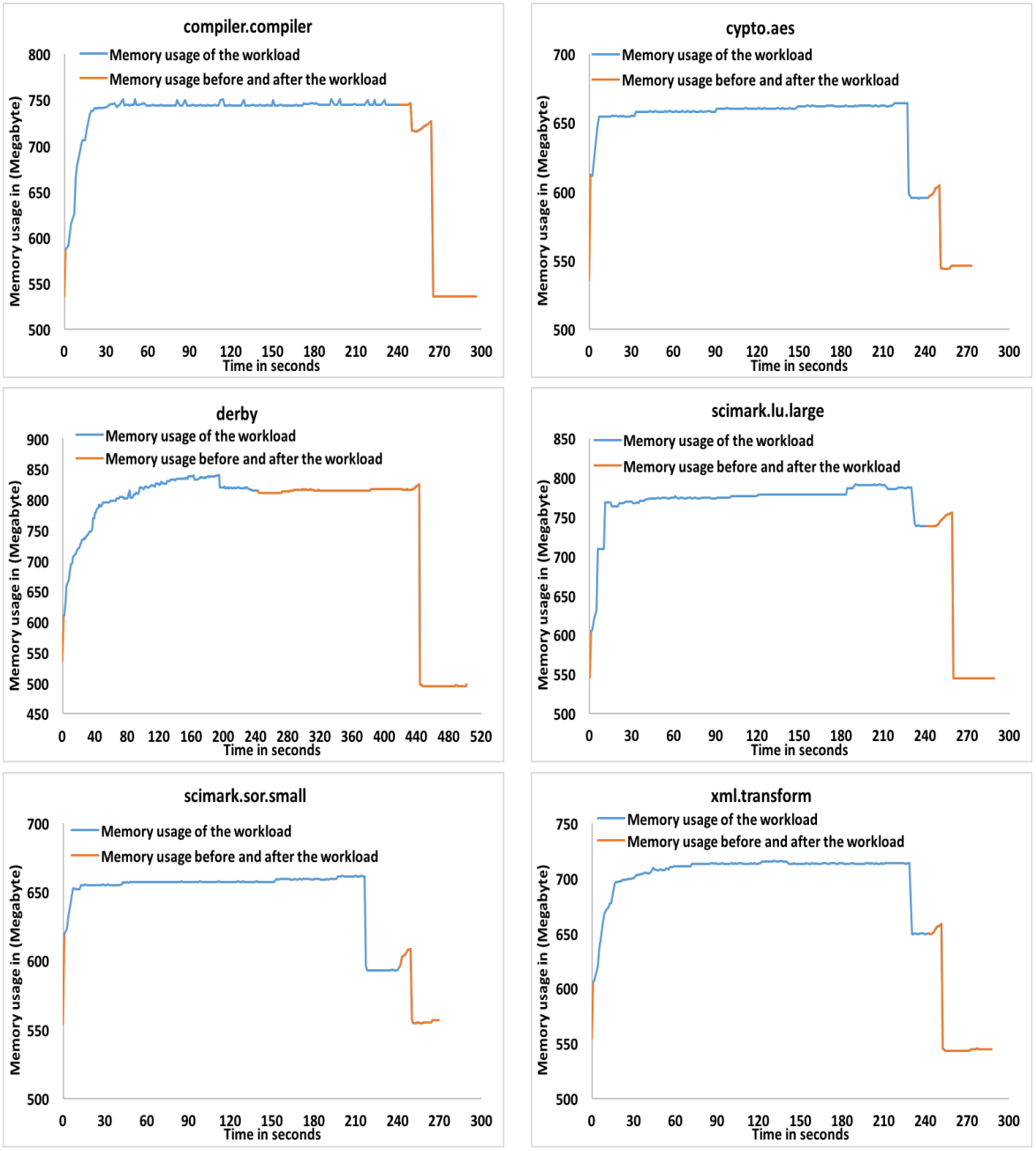

Figure 4.5: Memory usage of the VM 1 with Compiler.compiler, Crypto.aes, Derby, Scimark.lu.large, Scimark.sor.small, and Xml.transform workloads.

Figure 4.6 concludes that the CPU utilisation of the VMs with various workloads characteristics does not affect the migration time between server 1 and server 2. For instance, the Derby workload uses less CPU but takes more time to be migrated. Also, our result observation showed that some workloads utilise only 60% of the CPU.

In general, therefore, it seems that the network speed in this experiment does not cope

with the high change rate of memory pages during the migration of each VM. The VM live migrations need a high-speed network in order to perform efficiently which we did not address in this chapter. Thus, our results of the average migration time did not vary between each experiment setup. In the next chapter, we will address this limitation.

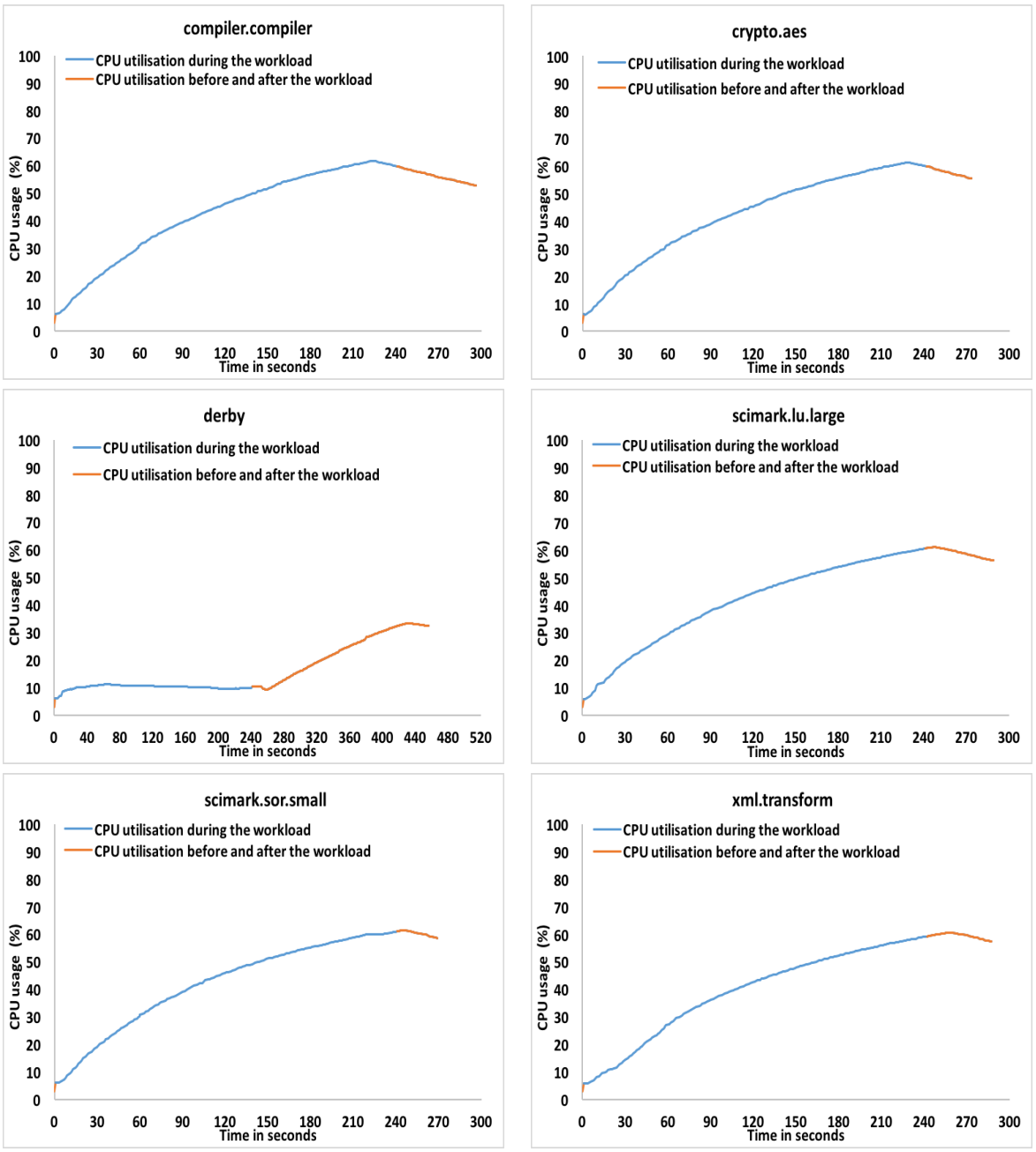

Figure 4.6: CPU utilisation of the VM 1 with with Compiler.compiler, Crypto.aes, Derby, Scimark.lu.large, Scimark.sor.small, and Xml.transform workloads

4.4 Conclusions

This chapter has measured the live migration time of different workload characteristics on various VMs capacities. We used KVM as hypervisor and SPECjvm2008 benchmark to generate the workloads. We provided a script which automated the run of the experiment. Our results showed an essential link between the time of VM live migration and the memory size of VM. When the memory size of VM increases, the time of VM live migration increases. Also, our results showed that the live migration time of a single VM on a host does not appear to be affected by the CPU utilisation of the VM. Furthermore, the results demonstrated a strong link between the time of live migration and network bandwidth. The live migration time is depending on the network speed. VM live migration requires a high-speed network to get the most from it.

In addition, we are going to use our achievement of this chapter to extend our previous work which was mentioned in Chapter 3. We are aiming to add the virtualisation on the simulation tool. Then, we will evaluate the energy consumption of the VM in HTC environment. It is essential to know the cost of live migration before we consider the virtualisation and live migration in HTC systems in order to avoid unnecessary migrations.

In the next chapter, we use the same setup of this experiment to measure the live migration time with higher network speed. We show how changing the network switch impacts the migration time of a VM. Also, we use different machine learning techniques to predict the time of migration.

5

Machine learning models for predicting VM live migration

Contents

Summary

This chapter presents an extension of the previous experiment where we introduce different VMs' capacities as well as upgrade the network speed. The preliminary findings illustrate that it is impossible to migrate certain VMs in the timeframe of the operating workload, and, furthermore, that the migration of certain VMs can be achieved within the time of the operating workload. In combination with this, we provide a comparison between machine learning models to predict how likely a VM is going to be migrated within the time frame of the running workload.

5.1 Introduction

Virtual Machine (VM) consolidation in large-scale computing is a core strategic approach carried out to achieve energy efficient data centres. As we mentioned earlier, VM Live Migration has become an established technology used to consolidate virtualised workload onto a smaller number of physical machines, as a mechanism to reduce overall energy consumption.

It is important to acknowledge that the expenses associated with live migration are not taken into account for the majority of the existing VM live migration models, and, in certain situations, VM live migration can be characterised by greater energy usage when considered in relation to maintaining VM operation on the existing physical machine. To guarantee energy efficiency, it is worthwhile to derive an understanding of the limits which impact migration expenses prior to initiating VM migration.

This chapter addresses the above need by presenting predictive models for VM live migration which can be employed to select the VMs that can be readily migrated based on the characteristics of their workload. In order to build the models, we extend the live experiment which discussed in Chapter 4 to measure the VM live migration duration time, CPU utilisation, memory usage, buffer cache size, number of threads, and I/O activities during the migration process. Also, we increased the network bandwidth and introduced 9 VMs characterised by various hardware constraints.

Furthermore, the chapter exhibits the process of creating three machine learning mod-

els from the results of the experiment. Also, the comparison between the proposed models are presented as well as the difference in accuracy between the proposed models and other machine learning models is provided.

The remainder of this chapter is organised as follows. Section 5.2 introduces the experiment environment. Section 5.3 presents the results of our preliminary experimentation. We explain and build the predictive models in Section 5.4. In Section 5.5, we evaluate the model with various datasets, before concluding and motivating future work in Section 5.6.

5.2 Experiment Environment

A live experiment constitutes the basis of this chapter, and this is carried out to measure the durations of VM live migration and resource utilisation of the VMs during the migration in the context of different workloads. In order to produce the different workloads, the SPECjvm2008 benchmark [10] is employed, and KVM [4] is used as a hypervisor. Section 5.2.1 describes the experimental setup; following this, Section 5.2.2 introduces the benchmark for VM workload generation and Section 5.2.3 details the experimental scenario used for the rest of the chapter.

5.2.1 Experiment set up

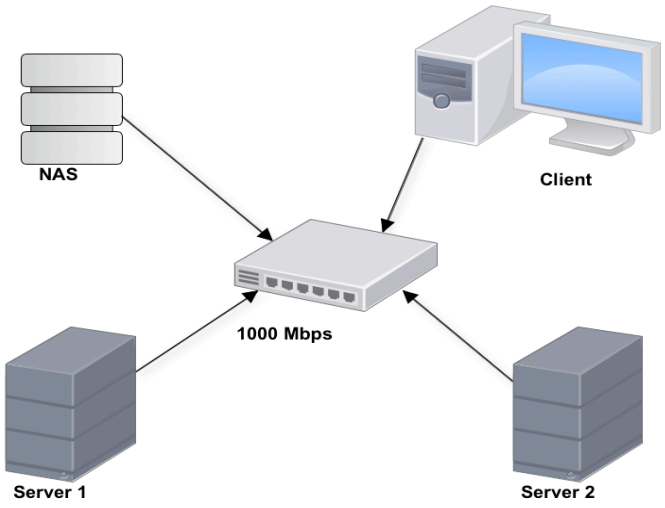

Figure 5.1: The setup of VM live migration experiment with 1000Mb switch

The setup of the experiment is similar to the setup which was presented previously in Section 4.2.1 of Chapter 4 except the switch. Here, we change the network switch to support a 1000 Mbps Ethernet speed for connecting the servers and the client computer which illustrated in Figure 5.1. The difference between the two setups is a factor that assists us to obtain various results.

5.2.2 Benchmarks

In order to perform the experiment with various workload characteristics, we used SPECjvm2008 benchmark which discussed in Section 4.2.2 of Chapter 4. Here, we use 20 workloads from the SPECjvm2008 benchmark as shown in Table 5.1. We excluded the `Derby` workload because the VM that runs it is corrupted during migration.

Table 5.1: SPECjvm2008 Benchmark workloads

Group name	**Workloads**
Compiler	compiler.compiler, compiler.sunflow
Compress	compress
Crypto	crypto.aes, crypto.rsa, crypto.signverify
Mpegaudio	mpegaudio
Scimark Large	scimark.fft.large, scimark.lu.large, scimark.sor.large, scimark.sparse.large, scimark.monte_carlo
Scimark Small	scimark.fft.small, scimark.lu.small, scimark.sor.small, scimark.sparse.small, scimark.monte_carlo
Serial	serial
Sunflow	sunflow
Xml	xml.transform, xml.validation

5.2.3 Experiment scenario

We first formulated a VM in Server 1 by using KVM. In turn, the image was stored on the NAS. The VM runs Ubuntu 16.04 LTS, and it incorporates the SPECjvm2008 to generate a range of workloads. Several distinct VM hardware capacities have been employed. Each VM is denoted as VM_{ij} where $i,j=\{1,2,3\}$, i is the number of CPUs, and j is the RAM capacity in Gigabytes (GB).

The experiment employs 20 SPECjvm workloads, and each is operated independently. Once the first minute of workload operation has been completed, the VM live migration

procedure is started to transfer the VM from Server 1 to Server 2. After the VM has been migrated to Server 2, the VM gets restarted and starts the previous steps again between Server 2 and Server 1.

The presence of the client computer is important in facilitating the operation of the experiment in an automated way. We developed a script [28] that enables the client computer to run the benchmark on the VM and, following this, the initiation of the live migration between the two servers. The client accesses the VM and commences the operation of a single workload; following one minute of running the workload, the client facilitates the migration of the VM from the first to the second server; in turn, the commencement and completion times of the VM live migration are recorded in a log file. In addition to this, the script maintains a log of the memory usage of each workload over the course of its runtime from second to second, and this is measured by the `Memusg` script [5]. Furthermore, `top` [16] and `sar` [7] are used to maintain a log of the CPU utilisation, total system memory, free memory,memory used, buffer cache, I/O activities, queue size, and load average over the course of the migration.

5.3 Experimental Results

In this section, we discuss the results of our preliminary experimentation. We demonstrate the impact of different workload characteristics, and their impact on VM live migration time, on various VMs capacities.

Figure 5.2 presents the results of VM migration from Server 1 to Server 2. The VMs migration starts after one minute of running the workload. We migrated each VM with each workload ten times from Server 1 to Server 2 and from Server 2 to Server 1. The figure illustrates the average duration of the VM live migration of each workload from Server 1 to Server 2. The results of the migration time between Server 2 to Server 1 has drawn the same conclusion in general, but have some variation compared to Server 1 because of the difference between the hardware of the servers. The results of the migration time between Server 2 to Server 1 is shown in Figure 5.3.

The results represent a clear discrepancy in average migration time between workloads due to the various workload characteristics. Some workloads such as `Com-`

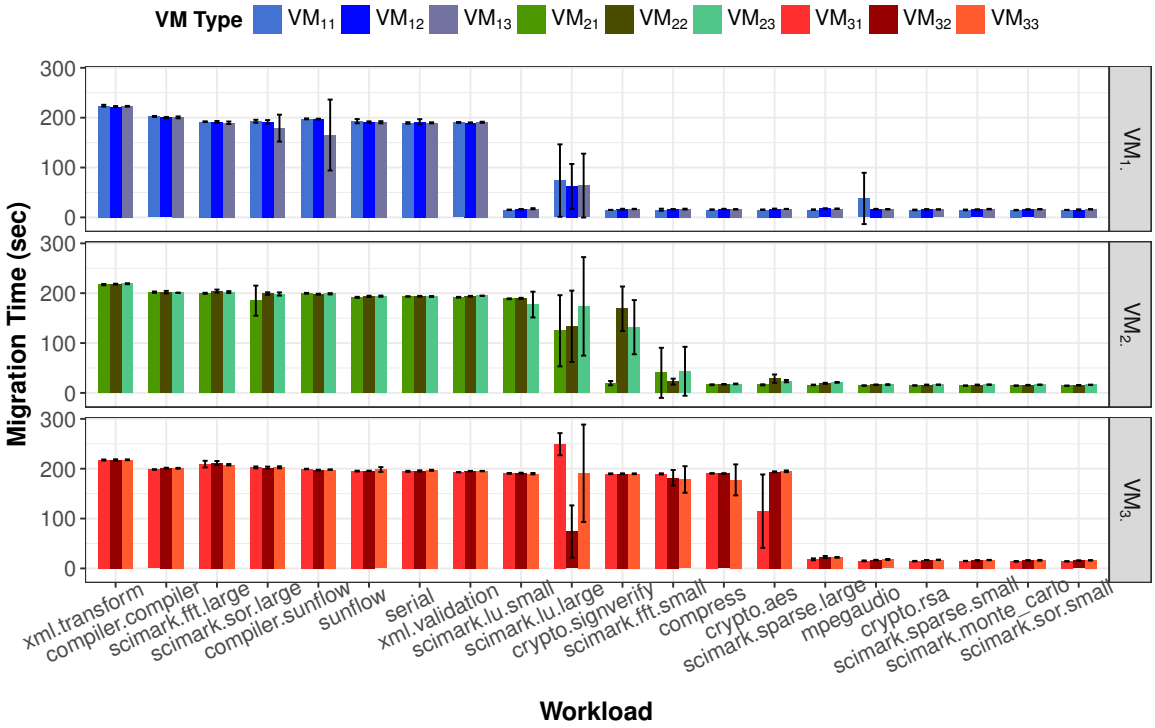

Figure 5.2: Average migration time of VM_{ij} with various workloads from Server 1 to Server 2

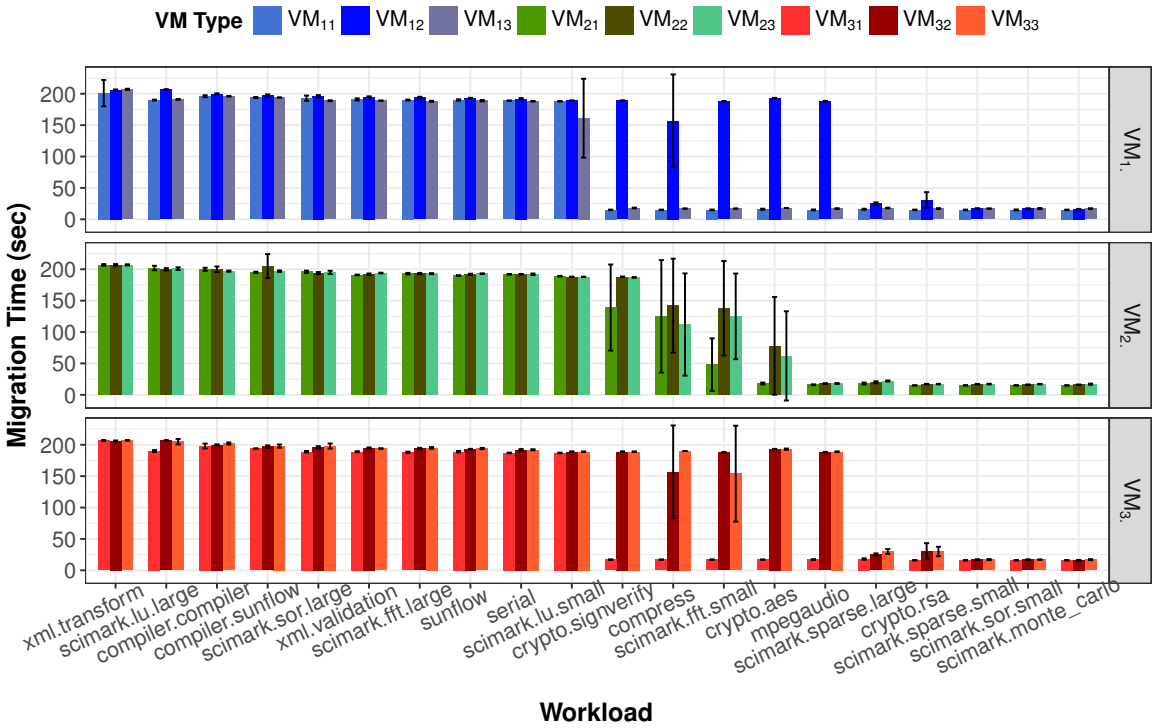

Figure 5.3: Average migration time of VM_{ij} with various workloads from Server 2 to Server 1

`piler.compiler`, `xml.transform`, and `xml.validation` consume much memory than others which increases the time of VM live migration between servers. The reason is that VM live migration process requires copying the VM's memory pages from the source host to the destination host. At the point when there is a high rate of changing memory pages during the live migration with an insufficient network speed, the migration might never complete or might take a long time. Also, the number of threads that produced by the workloads is another reason of delaying the VM live migration or making it impossible. For example, the threads that generated by `Serial` and `Sunflow` workloads put pressure on the memory which defines them as non-migratable workloads in our experiment.

The number of operations in the SPECjvm2008 benchmark is based on the available resources. When there are more resources available in the VM, the workloads generate more operations to test these resources. In our experiment, some workloads in $VM_{2.}$, and $VM_{3.}$ have longer migration time than the workloads in $VM_{1.}$ due to the fact that they contain more memory pages compared to $VM_{1.}$.

In addition, the standard deviation bars in the figures are shown for all workloads. Here, we observe in some workloads that average migration time demonstrate a variance, across each workload and VM configuration. Due to the uncertainty and inconstancy of the average migration time on some workloads over various VMs and servers, we selected 10 stable workloads to create our VM live migration predictive models as discussed in the next section.

5.4 Virtual Machine Live Migration Modelling

In this chapter, we used supervised learning in the form of classification to build our predictive models. We used classification and regression training (`caret`) package [102] by using R language [138]. Section 5.4.1 briefly introduces Stochastic Gradient Boosted (SGB), Random Forest (RF), and Bagged Tree (BT) Models. Section 5.4.2 discusses the dataset that is used for building the models, while Section 5.4.3 presents the comparisons between the models.

5.4.1 Stochastic Gradient Boosted, Random Forest, and Bagged Tree

Stochastic Gradient Boosting (SGB) [80] is an advanced model of the Gradient Boosting model. The basic idea of the Gradient boosting method is to fit a classifier, typically a decision tree, at each iteration, so that the next classifier is trained to improve the existing trained ensemble. In the SGB, the subset of the training data is selected randomly in each iteration. Then, the random subset of the training set is used to fit the base learner and tune the model for the current iteration. The SGB model has many parameters [141] which can be defined by the user. The number of trees which determines what number of trees are required to be built in the model. The interaction depth parameter can control the maximum size of each tree. The impact of each consecutive tree on the final predictions controlled by shrinkage also known as learning rate. The low learning rate with a few number of trees can give poor results, while a large number of trees, may improve results.

Random Forest (RF) [51] is a collection of decision trees that are different in structure. RF selects the best splits of each node between a random subset of the features. Also, the bootstrap or subsampling methods used by the training set to grow each tree. The RF has two parameters which can be specified by the user; the total number of trees of the model, and the number of splits controls the number of features in each node split. Building RF models with a large number of trees does not lead to overfitting due to the "Strong Law of Large Numbers" [75]. According to Breiman [50], the Strong Law of Large Numbers ensures that the generalisation error always converges when increasing the number of trees.

Bagging [50] is a method that takes different samples datasets, creates a set of high-variance base learners (usually decision tree), and then averages the prediction results. In Bagging, the subset of the training data is drawn with replacement from the training set which makes it different from SGB. Also, all features are considered for splitting a node, unlike RF where a subset of the features is randomly selected, and the best features of the subset are used to split the node.

5.4.2 Dataset

In order to create the predictive model for VM live migration decision, we need to select data from the experiment results which address the following question "what VMs are migratable or non-migratable within the time frame of the running workload?". The answer to the question assists choosing the VMs with a low migration time to be migrated. That will reduce the associated cost with the migration as well as the network traffic.

It is critical that we feed the model with the right data that solves the above question. Consequently, the dataset for training and testing the models is selected from 10 stable workloads' results with 10 features and marked with two class labels as illustrated in Table 5.2. When the workload can be migrated during its run time, we mark it as migratable workload. The migratable workloads take around 15 seconds to be fully migrated between hosts.

Workload	**Class Label**	**Features**
crypto.rsa		Total System Memory
scimark.monte_carlo		Memory Used
scimark.sor.small	Migratable	Free Memory
scimark.sparse.small		Buffer Cache
scimark.sparse.large		Number of CPUs
compiler.compiler		CPU Usage
serial		Load Average
sunflow	Non-migratable	Queue Size
xml.transform		Blocked Tasks
xml.validation		Transactions (I/O)

Table 5.2: Dataset structure

The dataset is determined to 20 seconds which is from 50 to 70 seconds where the migrations starting point occurred during this period. Each row of the dataset represents one second which is taken from the recorded logs of the experiment, and it has the values of the features as well as the class labels. The dataset contains 18000 rows and is divided into two sets: a training set (75%) and testing set (25%). The training set is used to build the model, and the testing set is used to evaluate the performance of the model. We took advantage of the built-in function *createDataPartition* in `caret` package [102] to have stratified random splits within each class label of the dataset.

Table 3 presents a sample of the dataset.

Table 5.3: Dataset sample

Total System Memory	Memory Used	Buffer Cache	Free Memory	CPUs	CPU Usage	Load Average	Queue Size	Blocked Tasks	I/O	Class Label
992.56	744.44	229.88	18.24	1	100	1.17	1	0	0	Non-migratable
2000.30	691.87	235.32	1073.12	3	99.7	2.2	3	0	0	Migratable
3008.44	1017.48	240.52	1750.43	2	99.5	1.76	5	0	2	Non-migratable
992.56	656.73	235.38	100.45	1	89.1	2.26	6	0	1.98	Non-migratable
3008.31	1035.91	236.05	1736.36	3	82.1	2.07	5	0	0	Migratable
992.56	657.16	232.44	102.96	1	98	0.85	1	1	4.95	Migratable
2000.43	871.60	238.69	890.14	2	93.5	2.03	4	2	0	Migratable
3008.57	730.71	321.20	1956.66	3	95	2.09	2	1	352	Non-migratable

5.4.3 Tuning the models

We used `caret` package [102] to create our predictive models. The package depends on 27 packages, and *train* is the main function that can be used to build and evaluate the models. The first step in creating a predictive model is to choose the model. In this chapter, we selected the following models: Stochastic Gradient Boosted (SGB), Random Forest (RF), and Bagged Tree (BT). We used `gbm` package [141] for SGB, `randomForest` package for RF, and `ipred` package for BT.

Next, we need to set the tuning parameters of the models. There are no tuning parameters for BT model. However, for the SGB model, we tuned combinations of values of the number of trees, *n.tree = seq(100, 1000, by=50)*, the depth of each tree, *interaction.depth = seq(1, 7, by=2)*, and the learning rate (or shrinkage), *shrinkage =c(0.01 , 0.1)* which is in total generates 152 combinations. For RF model, the values of the number of trees to grow, *ntree= seq(100, 1000, by=50)*, and the number of variables at each node splits, *mtry = seq(1, 7, by=2)*.

Then, we need to specify the measures of performance for the models and pass it to the metric argument of the train function. We chose the area under the Receiver Operating Characteristic (ROC) curve, or simply AUC, to assess the performance of the models as advised in [86] and [109]. Finally, we need to specify the method of resampling such as cross-validation or bootstrap by using the *trainControl* function. We selected repeated K-fold cross-validation as the resampling method where $K = 10$ which recommended in [99] and repeated 5 times. After resampling, the train function determines the best values of the tuning parameters and fit them to the final model.

5.4.4 *Performance evaluation of the models*

We evaluate and compare the performance of the models to determine which model performs best for our dataset. We explore the impact of model parameters on the performance, training time, and prediction time. We then identify the most important features of each model, before comparing the accuracy of the SGB, RF, BT models with seven alternative models.

Figures 5.4 and 5.5 exhibit the correlation between different tuning parameters of SGB and RF models and the resampled estimate of the AUC. For SGB model, the AUC value increases when the number of trees grows, and the value max tree depth increases. Also, the SGB performs better with 0.1 shrinkage value. The best parameters value of the SGB model for our dataset when the number of trees is 1,000, the depth is 7, and the shrinkage is 0.1. However, the best AUC value can be achieved with fewer trees and number of splits in RF model as illustrated in Figure 5.5. Also, it should be noticed that the increasing of the number of the splits of each node could reduce the value of AUC which appeared in Figure 5.5 when the number of splits is 5 or 7.

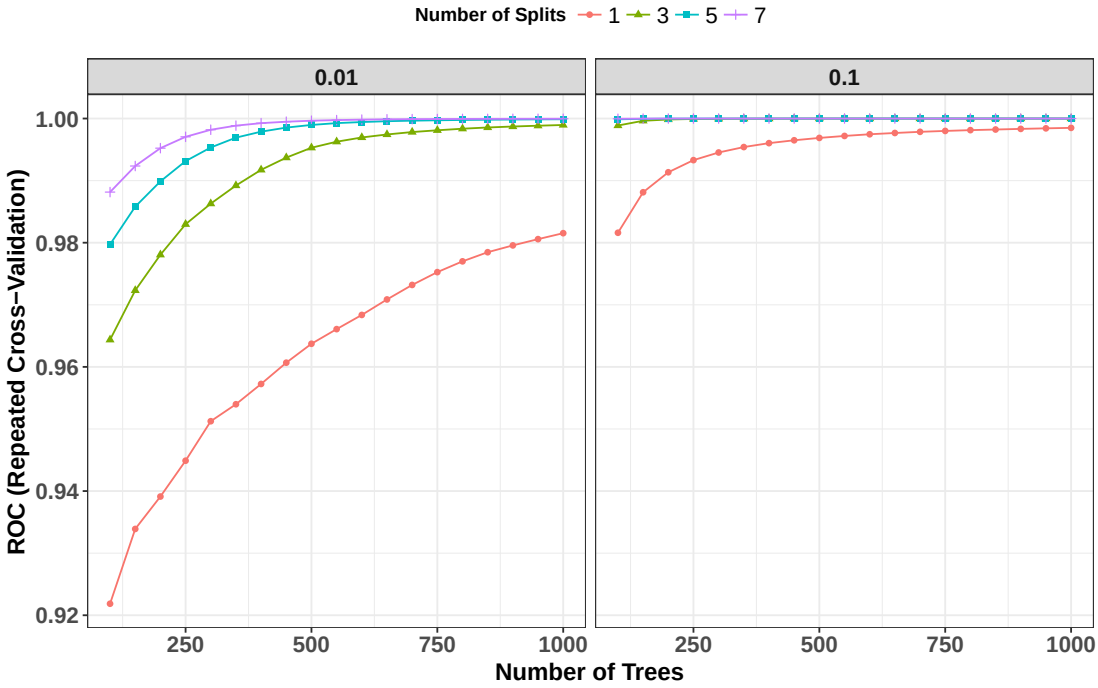

Figure 5.4: The value of AUC with various SGB parameters' values

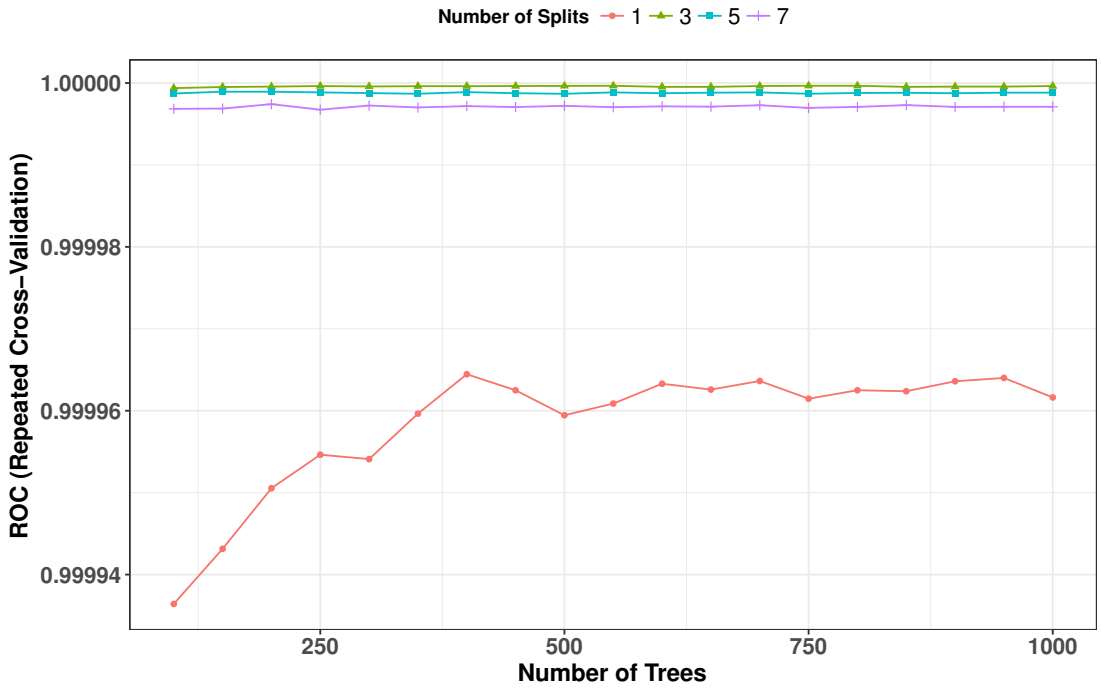

Figure 5.5: The value of AUC with various RF parameters' values

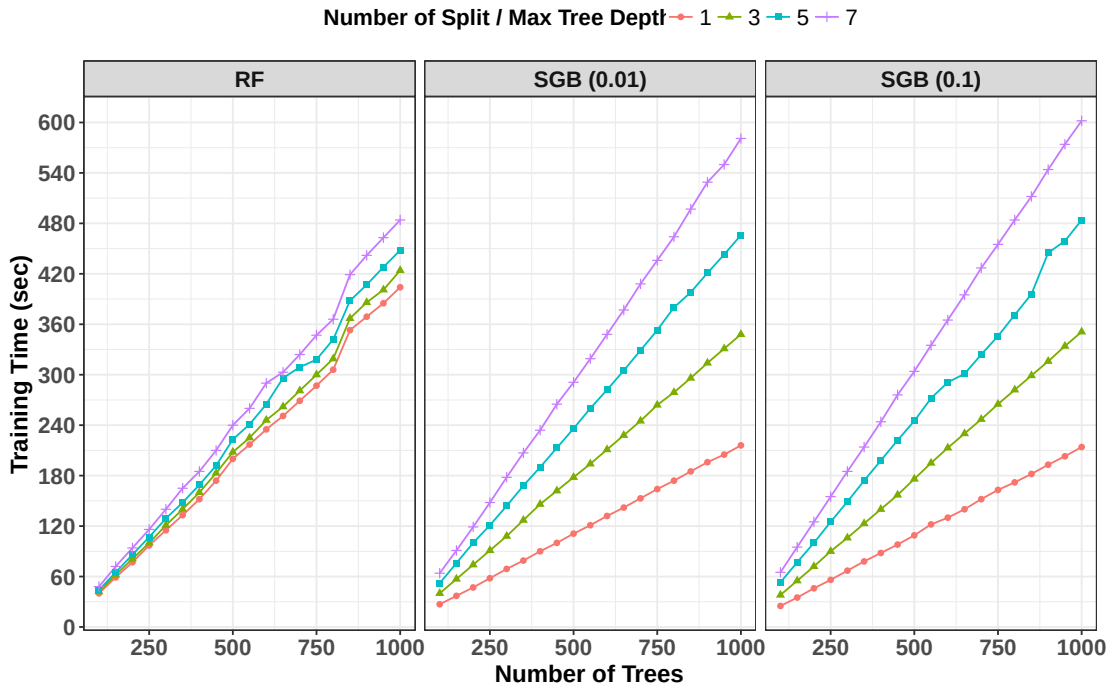

Figure 5.6: Training time of SGB and RF

We compare the training time of each model with various tuning parameters values as exposed in Figure 5.6. The training time of BT model is 62 seconds which is not in Figure 5.6 because the model has no tuning parameters to compare between their

values. Figure 5.6 shows a link between the training time and the number of trees and the number of splits or depth. When the number of trees, the number of splits, or the tree depth grows, the training time increases. Also, when the learning rate value of the SGB increases, the training time increases. Furthermore, the results show that the training time of SGB model can be faster than RF model when its tree depth is 1 or 3 and longer when its tree depth is 5 or 7. Moreover, the training time of different SGB models shows a wide variation while the training time of the RF models presents less variation. In addition, the prediction time of the models is very fast, between 1 to 15 milliseconds.

However, we used the *varImp* function to characterise the importance of predictors on the models. Each model has its method for estimating the link of each feature to the model described in [101]. Figure 5.7 presents the impact of each feature on the models and the order. The importance is scaled to have a maximum value of 100. The feature importance values and their order are distinct in each model due to the different algorithms and methods that are used to build the classifiers and determine the most important features in the models.

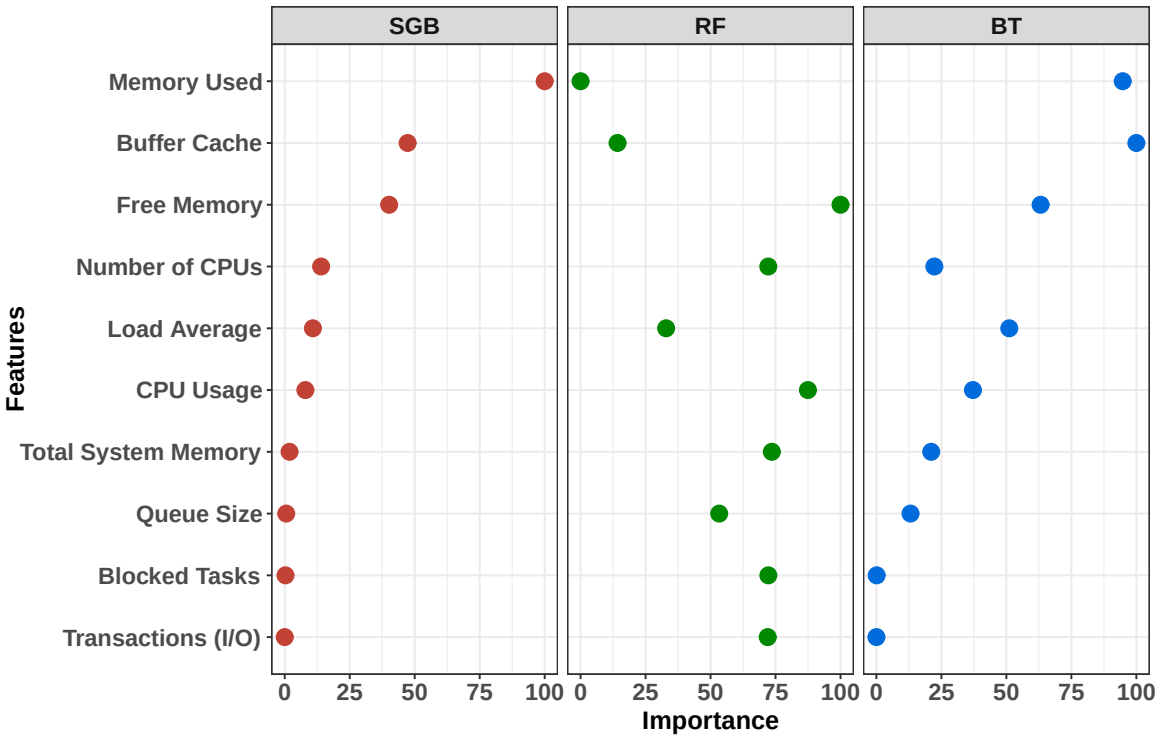

Figure 5.7: Feature importance of SGB, RF, BT

We went further to evaluate the accuracy of SGB, RF, and BT models with another seven models as follows: Support Vector Machines with Radial Basis Function Kernel (SVMRadial), K-Nearest Neighbors (KNN), Support Vector Machines with Linear Kernel (SVMLinear), Naive Bayes (NB), NeuralNetwork (NNet), Linear Discriminant Analysis (LDA), and Learning Vector Quantization (LVQ). We assessed the models in the same way with the same dataset to gain a fair comparison. To evaluate the models' performance on the test set, we used the `confusionMatrix` function to obtain a summary of the prediction results on our models. The function can provide information such as the accuracy rate, the confidence interval, the number of correct and incorrect predictions, the sensitivity, and the specificity. Figure 5.8 illustrates the estimated accuracy of each model as well as the 95% confidence interval. From Figure 5.8, we can conclude that there is little difference between the SGB, RF, and BT in estimated accuracy and they have the highest accuracy compared to other models.

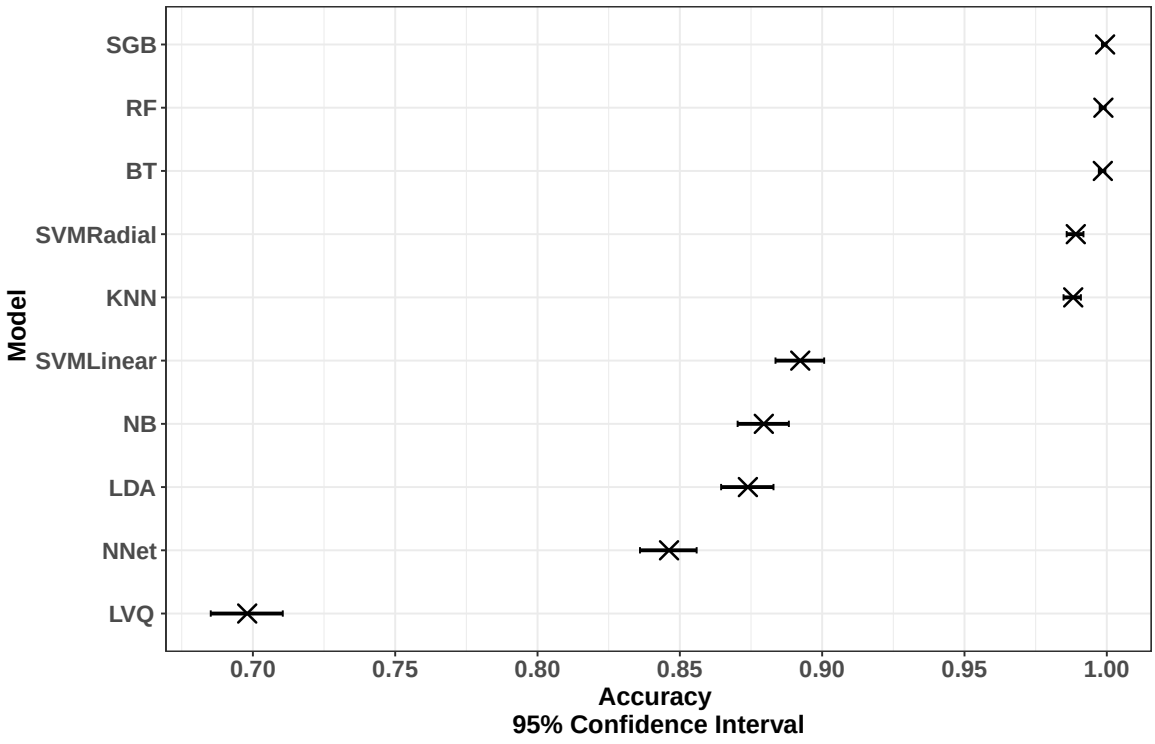

Figure 5.8: Compare different models

5.5 Predicting migration outcome

This section introduces the performance evaluation of SB, RF, BT models when created from various subsets of the original dataset. The aim is to understand the effect of one type of workload or VMs on the estimated accuracy. First, we train the models with 9 workloads (dataset) and evaluate the classifiers with the 10th workload (test set). For example, we train the model with `Scimark.monte_carlo`, `Scimark.sor.small`, `Scimark.sparse.small`, `Scimark.sparse.largel`, `Compiler.co-mpiler`, `Serial`, `Sunflow`, and `Xml.transform` workloads then we test it with `xml.va-lidation`. In total, there are 10 distinct training sets to create the models and 10 different 10 test sets to evaluate the models.

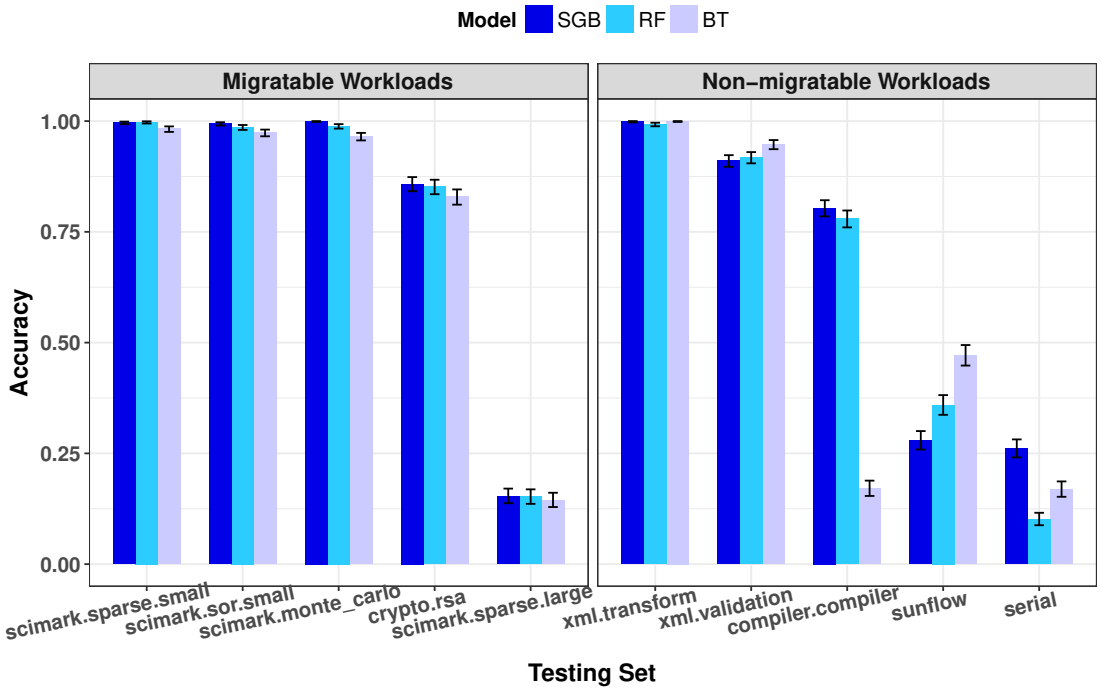

Figure 5.9: Compare SGB, RF, and BT with various datasets

Figure 5.9 shows the results of the estimated accuracy and the confidence 95% interval for each classification problem. In general, the SGB models perform better than other models in most of the cases. The models showed a poor performance in some of the cases than others due to the variation in the workloads' characteristics. When the model has been trained with a similar type of dataset and tested with a new kind of dataset, the model will fail to give a high accuracy on the prediction of the new test

set which is the case in `scimark.sparse.large`, `sunflow`, and `serial`. For example, the `sunflow` workload produces a greater number of threads than others which make its features different than other workloads. However, the SGB is not the best prediction to the `sunflow` workloads. The reason is that the SGB feature importance as previously discussed in Section 5.4 gives the load average a small importance rate than other models which affect the estimate accuracy in this case.

In the migratable workloads, `Scimark.sparse.large` has a poor performance over the models. The workload has similar characteristics of the non-migratable workloads which make the model predict it as a non-migratable workload. We can conclude that other workloads' features are not accurate to predict the `scimark.sparse.large`.

We went further to evaluate the estimated accuracy of the various type of VMs' datasets. Our aim is to know which characteristics of the VMs present the most impact on the estimated accuracy. Regardless the memory size of the VMs in our dataset, Figure 5.10 shows the results of training the models with VMs that are different in the number of the CPUs while Figure 5.11 present the estimated accuracy of VMs which are different in memory sizes.

In each case, the models created from 3 various training sets and validated with two testing sets. In Figure 5.10, the experiment results of $VM_{1.}$, $VM_{2.}$, and $VM_{3.}$ are the datasets to build and evaluate the models. We used one of the datasets to create the models, and the other two datasets are used to assess the models. For example, the experiment results of the VMs with one CPU used to build the models, and the results of the VMs with two and three CPUs used to evaluate the model. The $VM_{1.}$ models are better in predicting the VMs with 2 CPUs, and the $VM_{2.}$ models perform better in predicting the VMs with 3 CPUs while $VM_{3.}$ models are much beneficial in predicting the VMs with 2 CPUs.

We used the experiment results of $VM_{.1}$, $VM_{.2}$, and $VM_{.3}$ to train and test the models as shown in Figure 5.11. For example, the experiment results of the VMs with 1 GB RAM used to build the models, and the results of the VMs with 2 and 3 GB RAM used to evaluate the model. The $VM_{.1}$ models are not the best in predicting the VMs with 2 and 3 GB RAM compared to the other models.The $VM_{.2}$ models perform better in predicting the VMs with 3 GB RAM while $VM_{.3}$ models are much useful in predicting

the VMs with 2 GB RAM.

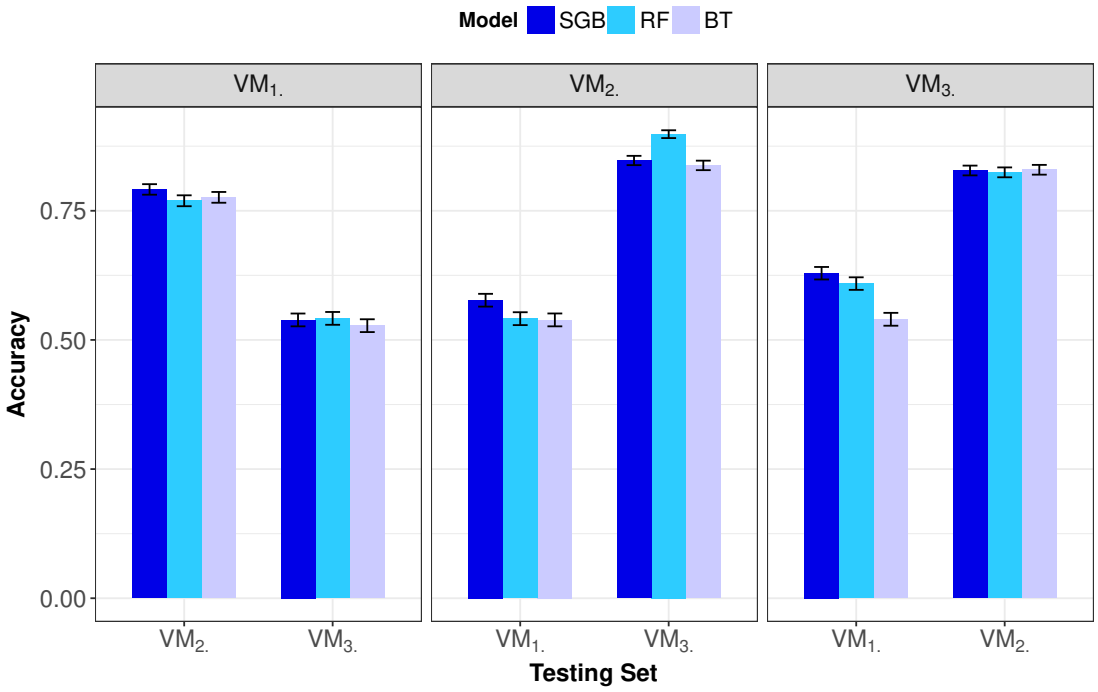

Figure 5.10: Compare SGB, RF, and BT with various datasets of $VM_{i.}$ types

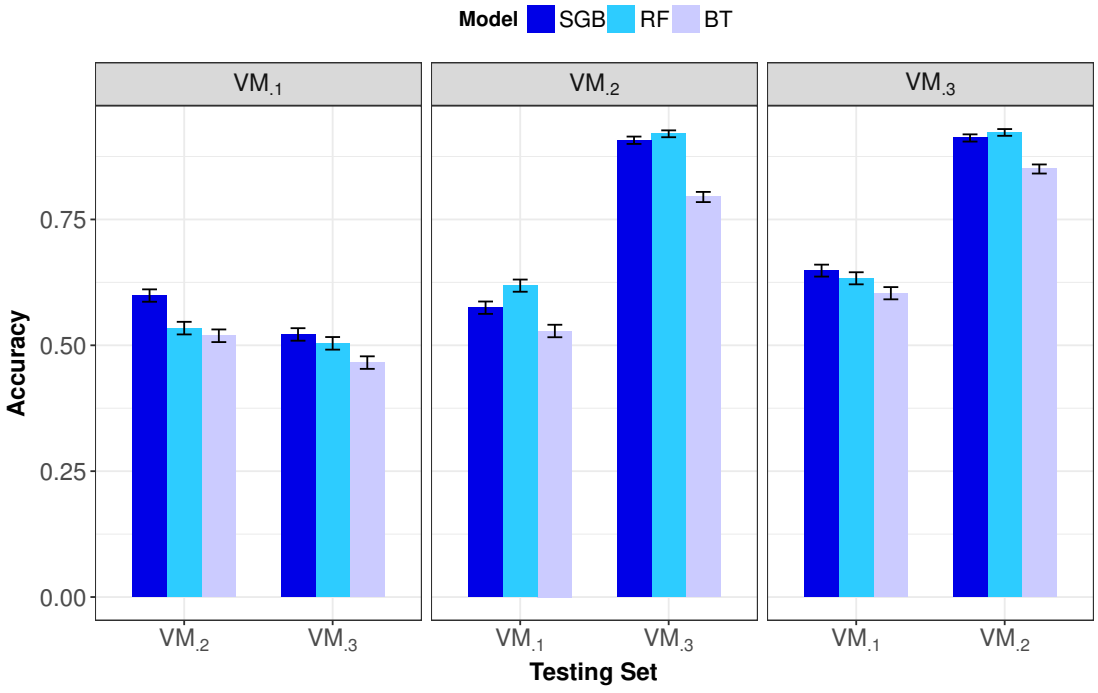

Figure 5.11: Compare SGB, RF, and BT with various datasets of $VM_{.j}$ types

The prediction results of the models are influenced by a number of factors. First, the differences in the sub-datasets that are used to train and test the models can affect

the estimated accuracy of the prediction. In order to create a predictive model, the sub-datasets of the original dataset should reflect the dataset. Second, the influence of predictors on the each model (importance of features) can affect the prediction results of the models. Finally, the process of building the models are different in each model. Each model has its algorithm and parameters to create the model. Any change on one of the parameters leads to amending the predictions results.

5.6 Conclusions

This chapter has measured the live migration time for different workload characteristics on various VMs capacities. We used KVM as hypervisor and SPECjvm2008 benchmark to generate the workloads. The results show that some VMs can be migrated within a short time while others take a long time to migrate and some cannot be migrated during the workload execution. The chapter presents the process of creating Stochastic Gradient Boosted (SGB), Random Forest (RF) and Bagged Tree (BT) models from the results of the experiment. We showed the effect of tuning the models with different values as well as training and evaluating the models among the various sub-datasets from the original dataset. It is clear that there is no easy choice of the best model to employ and in practice a combination of the models presented could be used to gain a better prediction of which VMs to migrate.

In the next chapter, we will use our finding of this chapter to introduce two methods for deploying virtualisation and live migration in HTC systems. We will extend a trace-driven simulation environment in order to evaluate our methods under different workload and migration time assumptions.

6

Simulation of VM Live Migration in HTC Environments

Contents

Summary

This chapter introduces an extension of an existing trace-driven simulation to incorporate virtualisation. Furthermore, we implement the pre-copy live migration algorithm to provide a test environment for live job migration in a high throughput computing system (HTC). We present details of our extension into HTC-Sim as well as two approaches to perform the live migration in HTC environments. In this chapter, we use a simple random policy for picking the jobs in order to migrate them and selecting the target hosts to allocate the migrated jobs. Finally, we provide a discussion of the simulation outcomes for each migration method.

6.1 Introduction

Computer simulation is a powerful evaluate complex systems and understand the reaction of the systems in the real world. As VM live migration becomes a mechanism for reducing overall energy consumption in large-scale computing by consolidating VMs onto fewer physical machines, simulation assists to enhance the policies of VM live migration decisions to gain more saving in energy and better performance.

High through computing (HTC) systems are popular choices for academic and industry researchers because of the economy of scale. Researchers can share many resources to do their jobs, and the resources are available for use 24/7. Since the resources in HTC environment is shared usage, the HTC jobs might be affected by other users. For instance, universities use students' computer clusters as a resource pool for their HTC system where HTC jobs can be executed on computers that are not being used by students. There is a high probability that an HTC job can be interrupted when a user starts using the computer. Such an interruption cause job eviction or suspension which increase the power consumption and job makespan.

It is important to run these long-running jobs in the most energy-efficient machines to decrease the energy consumption. However, the most energy-efficient machines might be not available at the beginning of the HTC jobs initiation. Furthermore, it is necessary to prevent the HTC jobs from being evicted from the computer or suspended

for a long time to minimise energy waste and makespan of the jobs.

Virtualisation has previously been applied to HTC systems such as HTCondor where VMs considered as jobs to be executed [152]. The power of VM live migration could be used to gain more saving in energy by migrating the jobs into energy efficient machines when they become available. Moreover, it can prevent the jobs from being evicted or suspended for a long time due to user login by migrating them to other idle machines. However, VM live migration in HTC system has received a little attention in the literature. Existing research has considered the use of the Checkpointing [36, 79, 133, 140] in this context to minimise the effect of jobs interruption on overall performance and energy consumption of HTC systems.

In this chapter, we propose a different approach in which we use the ability of VM live migration to mitigate the energy waste and the makespan of jobs in HTC environment. In order to achieve this, we extend a trace-driven simulation HTC-Sim [78] to support virtualisation, in particular, VM live migration. We implemented the pre-copy migration algorithm [60] in the same manner as in the real world. The simulation tracks the number of successful and failed VM migrations, the time of VM migrations, and the energy consumption of VM migrations. Additionally, the expansion of this simulation will ultimately assist the development and evaluation of algorithms for VM consolidation in large-scale computing.

The rest of this chapter is organised as follows. We introduce the simulation environment in Section 6.2. In Section 6.3, we present two scenarios to perform the live migration in our simulation. We explain the outcomes of the simulation in Section 6.4 before concluding in Section 6.5.

6.2 Simulation Environment

We implement virtualisation and the pre-copy live migration algorithm into HTC-Sim [78] simulator. HTC-Sim is a Java-based trace-driven simulation for simulating multi-use resources shared with interactive users as well as dedicated resources. Moreover, the tool is based on a real dataset collected from Newcastle University HTCondor system in 2010.

6.2.1 Datasets

The Newcastle University employs HTCondor to provide a high throughput computing environment to its researchers. The HTCondor pool contains 1400 desktop computers spread across 35 clusters on campus. When a student log into the computer, the job on that computer gets evicted and rejoins the queue of the HTCondor to be restarted on another machine. The log file of the interactive user contain login timestamps, logout timestamps, computer's name, and a hash of the user's identity, while the logs file of the jobs includes the job id, job submission time, job duration, the state of the ran job (either 'success' or 'terminated'), and the hash id of job owner. Furthermore, the cluster specification is stored in cluster file which has the cluster name, opening and closing times, and list of the computer resources and their specification associated with an energy profile.

Type	**Cores**	**Power Consumption (W)**		
		Active	**Idle**	**Sleep**
Normal	2	57	40	2
High End	4	114	67	3
Legacy	2	100-180	50-80	4

Table 6.1: Computer Type

Table 6.1 shows the energy consumption of different computer types based on Newcastle University HTCondor resource pool. The computers are allocated within clusters and each cluster contains equivalent computer resources. Our data do not contain the CPU utilisation that utilised by the HTC jobs. As a result, the performance of computers considered to be homogeneous with a different energy profile. Hence, we identify three states where a machine may belong. The states are based on the Advanced Configuration and Power Interface (ACPI) specification [20]. The states are as follows:

Active: the machine in this state is used by a HTC job or an interactive user. Here, we assumed the CPU load level is 100%. This state equates to ACPI state S0.

Idle: the machine here is not processing a work for an interactive user nor HTC job. The CPU load level in this state is approximately 5-10%. Also, this state is S0 according to ACPI.

Sleep: the components of the machine in this state are powered off expect RAM which allows the operating system to resume without the need for restarting it. This state is S3 as stated by ACPI.

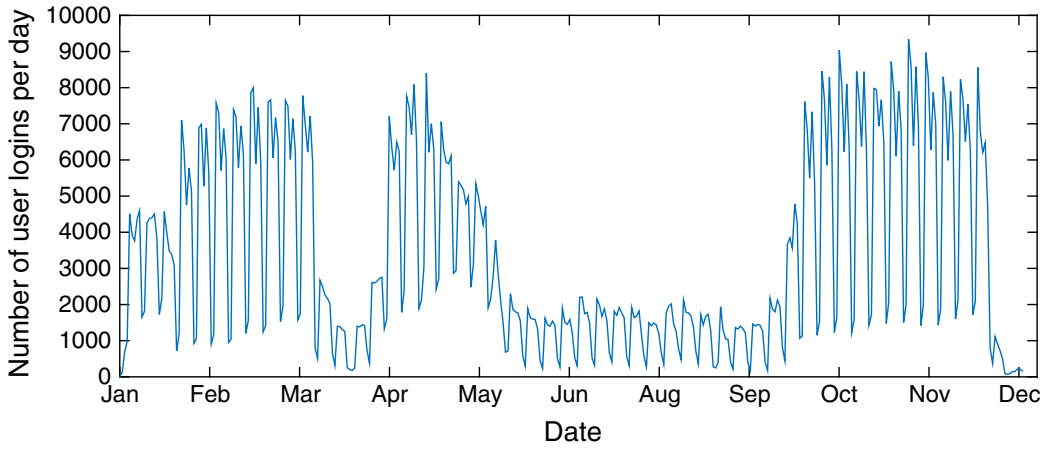

Figure 6.1: Interactive user logins per day in 2010 [78].

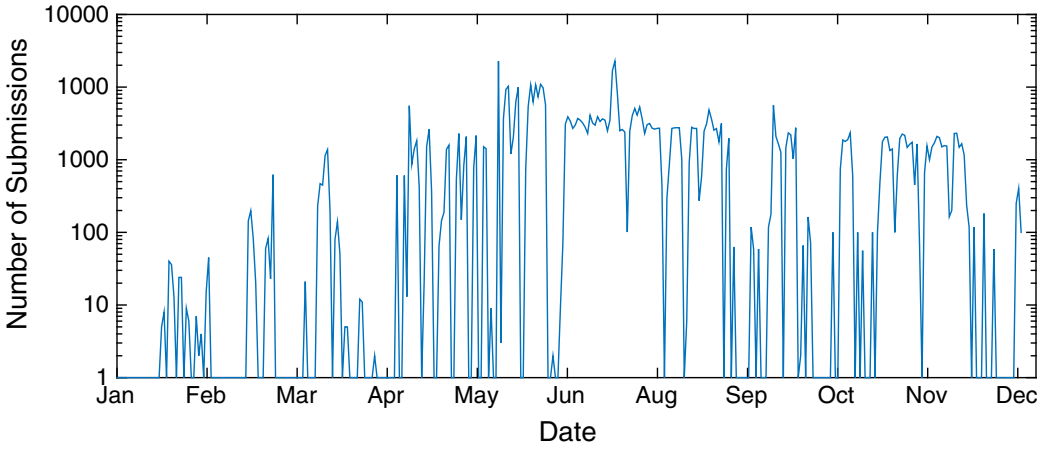

Figure 6.2: HTCondor jobs submission in 2010 [78].

Figure 6.1 shows the number of user logins per day during 2010 while Figure 6.2 illustrates the number of HTCondor jobs in the same period. A total of 1,229,820 user logins were occurred over 2010, and the total number of submitted HTCondor jobs was 561,851. From the figures, we can see a high probability for a job to be interrupted by interactive users during its execution.. We can stop jobs from being interrupted by migrating them to resume their execution on other machines. To achieve this, the HTCondor jobs that run in a closed cluster can be moved to other idle computers just before its opening time for students. Also, when a user logs in while HTCondor task is running, the task can be migrated to another machine.

6.2.2 Conceptual Simulation Architecture

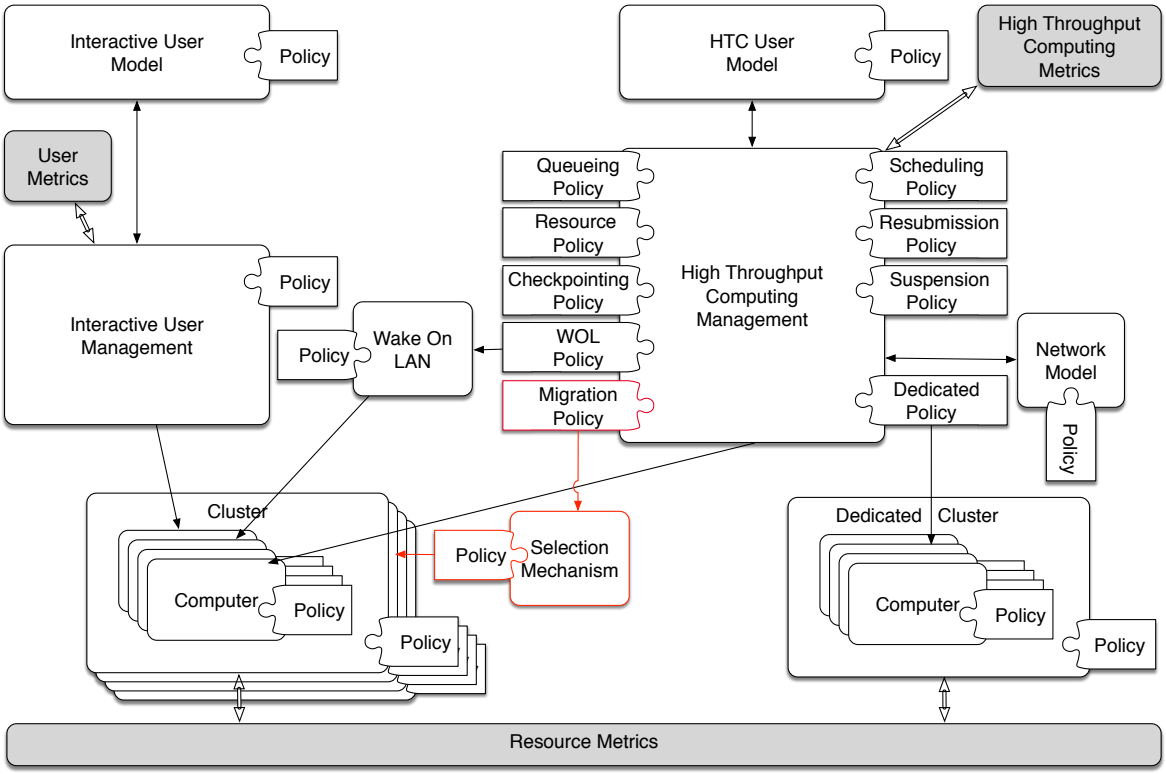

Figure 6.3: Conceptual architecture for HTC-Sim

The original conceptual architecture for HTC-Sim exhibited in [78]. The conceptual architecture has been extended to perform job live migration as illustrated in Figure 6.3. The unshaded boxes within the figure show a distinct service within the simulation and the metric collection points are represented by the shaded boxes. Hollow arrows represent the information flow to the metrics collectors while the information flow between services is represented by solid arrows. Also, the services within the architecture are implemented to provide the common actions of the service which can be specified to a particular realisation by a "pluggable" extension interface – the plugin "Policy" element.

In Figure 6.3, we add the pluggable Migration policy to the original HTC-Sim architecture. The pluggable Migration policy is a Java class that specifies when and which job to migrate. The Selection Mechanism class handles the computer reservations when the migration is needed, and the pluggable policy specifies which computer to select from the HTCondor pool.

6.2.3 Migration Model

The simulator requires four files in order to perform. The first file contains the policies configurations which needs to be evaluated by the simulation. The trace log of HTCondor workloads is included in the second file. The workload records specify the submission time of the jobs, their duration, and their memory usage at the time of completion. The third file has the trace log of user login to computers and logout from computers, in addition, the fourth file contains the specifications of computers.

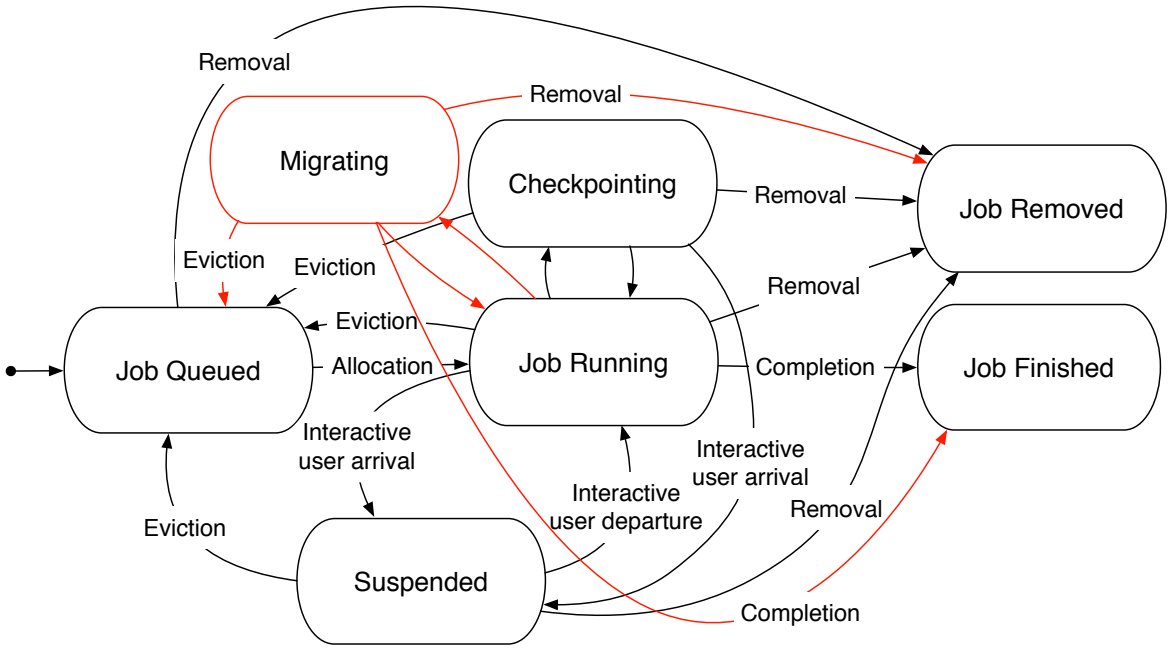

Figure 6.4: Job state transition diagram with migration state.

The original state transition diagram for a single job in HTC-Sim has been presented in [78]. The HTCondor job enters the queue when it is submitted by a user and then waits to be allocated on a machine. During its runtime, the job might be interrupted by interactive users. As a result, the job gets suspended for a while or evicted immediately. The task can also regularly checkpoint during its execution time. Furthermore, a system administrator or the owner of the job may manually remove the job at any state.

In Figure 6.4, we add the Migration state to the initial workload state diagram. The migration of the workload can occur at anytime of its execution time. Additionally, migration policies manage the migration process by determining which job, when,

and where to migrate. Also, live migration allows jobs to continue executing while migration is taking place. Moreover, we assume the completion of the job might happen during the migration along with job eviction.

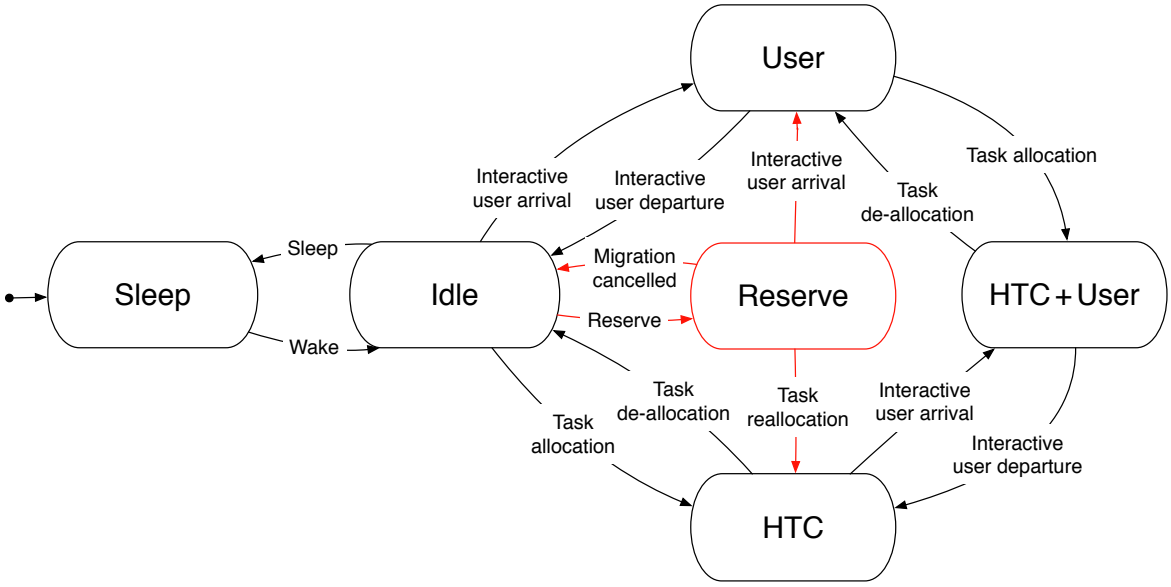

Figure 6.5: Computer state transition diagram with reserve state.

Figure 6.5 shows the computer state transition diagram which previously introduced in [78]. We include the Reserve state in it where computer enters when it is selected as a target host for a migrated job. When the job starts transferring from the source host to the target host, the computer enters the HTC state. Furthermore, the Reserve state changes to the User state if an interactive user starts using it. Also, the reserved computer begins idle when the migration is cancelled. When a machine is in a Sleep state, it is powered-down except the RAM. In this way, the machine consumes very low energy and can be powered-up very quickly without restarting the operating system. Unlike the machine when it is idle or reserved, it consumes more energy than the Sleep state, but much lower than HTC, User, and HTC+User states.

6.3 Simulation scenario

In this section, we present two different techniques to perform the live migration in Newcastle University's HTC system. The techniques are based on our observation of the system in order to mitigate jobs energy waste and makespan. To achieve this, the

system needs to prevent jobs from rejoining the queue and restarting the execution due to jobs' evictions.

6.3.1 Interval Migration

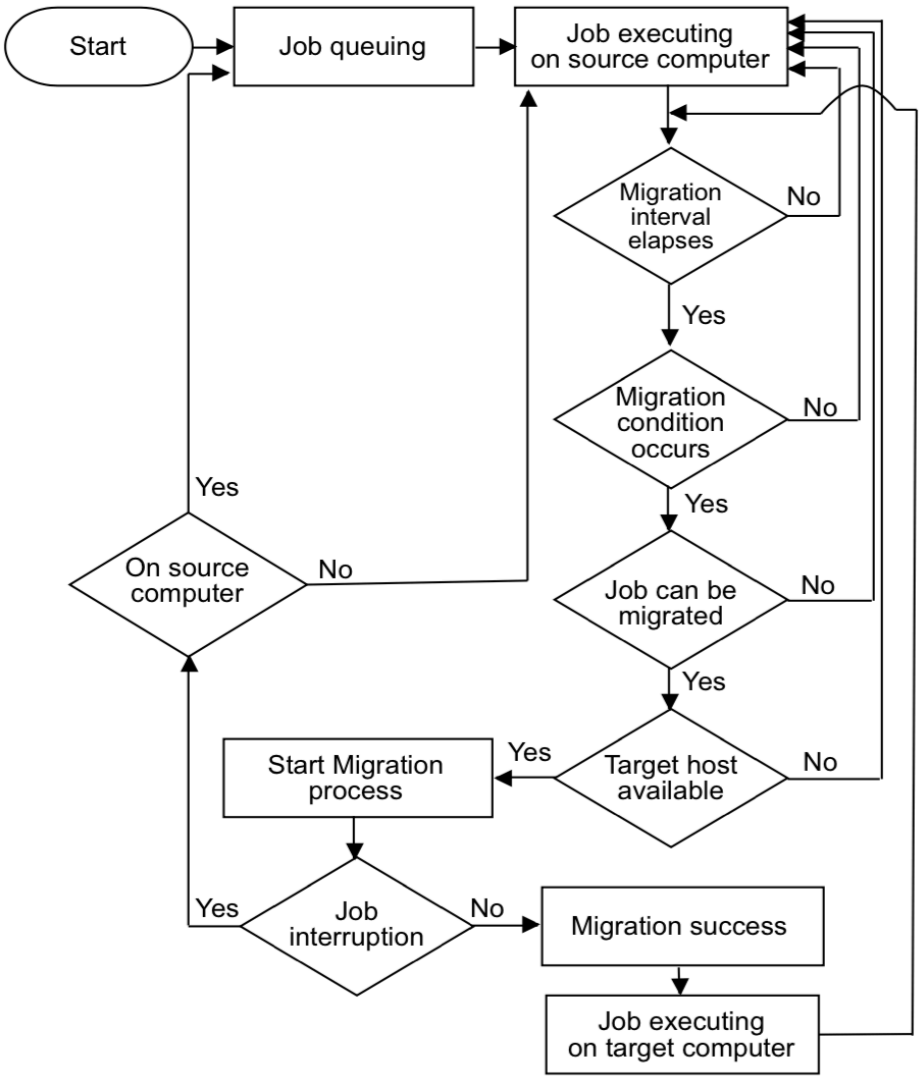

Figure 6.6: Flowchart of interval live migration.

In this technique, we introduce a fixed period of time during job execution to check if the condition of migration occurs. The condition can be specified by the administrator of the system in order to gain a better performance or save energy.

In Chapter 5, we measured the live migration time for various workload characteristics on different VMs capacities. Also, we introduced three predictive models to predict the time of live migration. Our results showed that some workload might take a long

time to migrate and some cannot be migrated during the workload execution. We have implemented these facts into our simulation tool.

When there is a need for job migration, the system indicates whether or not the job can be migrated. If the job cannot be migrated within a short time, the job keeps executing on the source computer, and the system waits for the next migration interval to elapse for rechecking. If the job takes only a short time to be migrated, the system selects a target computer from HTC system computer pool and the process of live migration starts on the source and target computers. However, the job might be interrupted by an interactive user during the migration process which leads to job eviction. if a user logins into the target computer during the migration, the migration process stops, and the job resumes on the source computer. Nevertheless, if the job is interrupted by a user on the source computer, the job gets evicted and rejoins the queue.

The flowchart in Figure 6.6 illustrates the method of interval migration. Where a set of k jobs J=$\{J_1$,J_2,J_3,..., $J_k\}$ presents in the queue. Each J_i; $i \in \{1,...,k\}$ has duration time $DT(J_i)$. The starts time of the job $ST(J_i)$ is given when the job is allocated on a host for executing. Then, the finish time $FT(J_i)$ of the job is calculated as

$$FT(J_i) = ST(J_i) + DT(J_i). \tag{6.1}$$

When there is an interruption, the job J_i rejoins the queue and gets a new starting $ST(J_i)$ as well as new finishing time $FT(J_i)$ when it gets allocated on a host for executing. Moreover, the queuing time can be represented as $QT(J_i)$ for a job J_i which represents the waiting time for a job J_i in the queue. After interval time N, when the migration condition occurs, the job J_i starts migrating from the source computer C_s to target computer C_t. The job migration process is expressed as $J_i : (C_s, \rightarrow C_t)$ and the time of migration is represented as $MT(C_s \rightarrow C_t)$. Since we are using live migration, the job J_i is still running on a source machine C_s during the migration. Then, the job J_i stops running on the source computer C_s and resumes executing on the target computer C_t. The starting time of the job J_i on target machine C_t is given as

$$ST_t(J_i) = ST_s(J_i) + ET_s(J_i) + MT(J_i : (C_s \rightarrow C_t)). \tag{6.2}$$

where $ST_s(J_i)$ is the start time for a job J_i on the source machine C_s and $ET_s(J_i)$ is the execution time for a job J_i on the source machine C_s before starting the migration process.

The remaining execution time of job J_i on a target machine C_t is calculated as

$$ET_t(J_i) = DT(J_i) - ET_s(J_i) - MT(J_i : (C_s \rightarrow C_t)). \tag{6.3}$$

Here, the migration time $MT(C_s \rightarrow C_t)$ has been considered because the job is still executing on the source machine C_s during the live migration process.

6.3.2 Responsive Migration

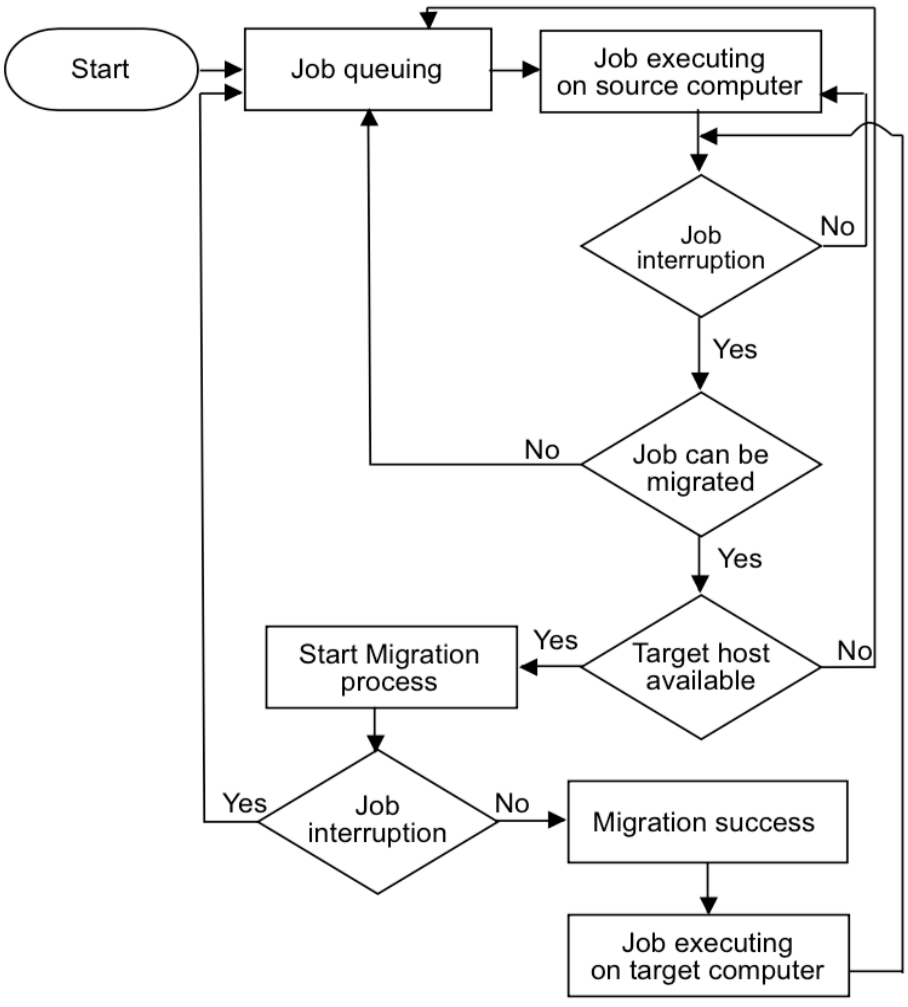

Figure 6.7: Flowchart of responsive live migration.

In this method, we migrate the job when an interactive user arrives while an HTC job is running which causes job interruption as illustrated in Figure 6.7. The HTC job and the interactive user share the same machine until the job is migrated to another machine. However, this can affect the performance of the running applications by the interactive user on the source machine. The interactive user will have full CPU and memory capacities of the source machine when the job is migrated.

In this technique, when the job cannot be migrated within a short time, the job is removed from the source machine and restart running on another machine. Also, the job rejoins the queue when there is no target machine available in the HTC system pool as well as when an interactive use begins working on the target computer during the migration process. Then, the finish time of the job will be calculated according to Equation 6.1. Furthermore, Equations 6.2 and 6.3 are used to determine the starting time of the job on the target computer and the remaining execution time.

6.4 Simulation outcome

Our extension of the HTC-Sim simulation tool can produce outcomes that associated with the migration process in HTC environment. The simulation provides the total of successful migrations and failed migrations, the overall time of successful and failed migrations, and the energy consumption of the migration process. Also, the simulation gives the reasons for migration failures during the migration process such as user interruption on the source or the target computers and the killed jobs by the administrator or job owner.

The presented results in this section aim to give a clear understanding of our simulation outcomes. Here, we demonstrate the migration interval and the migration responsive techniques. The total submitted jobs in the simulation are 532,467, and we assumed that not all jobs are migratable within the specified migration times. We have used migration proportion value to determine the percentage of workload which can be migrated. Also, we have used the random policy as a baseline to validate our extension to HTS-Sim. The policy selects the jobs randomly in order to migrate them as well as the selection of target computers. Furthermore, we have not introduced any policy

or algorithm to manage the migration process such as when to migrate, which job to migrate, and where to migrate. However, our tool can easily adapt policies and algorithms to reduce system's energy consumption and jobs' makespans. Section 6.4.1 represents the migration interval technique outcomes while Section 6.4.2 expresses the migration responsive results.

6.4.1 Interval Migration

Based on our observations of the VM live experiment presented in Chapter 4 and Chapter 5, we assumed that all the jobs cannot be migratable within the time frame of the running workload (termed as non-migratable jobs). As a result, we have added a migration proportion parameter into our simulation to limit the number of jobs that can be migrated.

Figure 6.8 shows the results of successful migration with different migration durations and intervals where the migration proportion value is 50%. The proportion value limits the number of jobs that can be migrated to 266,233. The figure presents the total number of successful job migrations, overall jobs migrations time, and the energy consumption that jobs consumed during the migration process. We see significant reductions in the results when the interval value increases. If the interval time occurs during the job execution time, the system checks its migration policies and starts the migration process if needed. Consequently, a short-running job might have fewer migration attempts than a long-running job which leads to decreases in migration energy consumption and time. Thus, it is crucial to determine when to migrate as well as the reason for migration to avoid inefficient migration.

We acknowledge a significant relation between the migration time and the energy consumption of the migration process. When a job takes a long time to be migrated, it consumes more energy. Furthermore, we see this as an existing problem for many VM consolidation techniques where the energy consumption of migration process is not considered. To get over this problem, a policy could be developed, leveraging our observations in chapter 5, to selectively migrate jobs based on estimated runtime and migration time. In doing so, the policy could curtail the costs of migrating jobs, and achieve an overall benefit for the system.

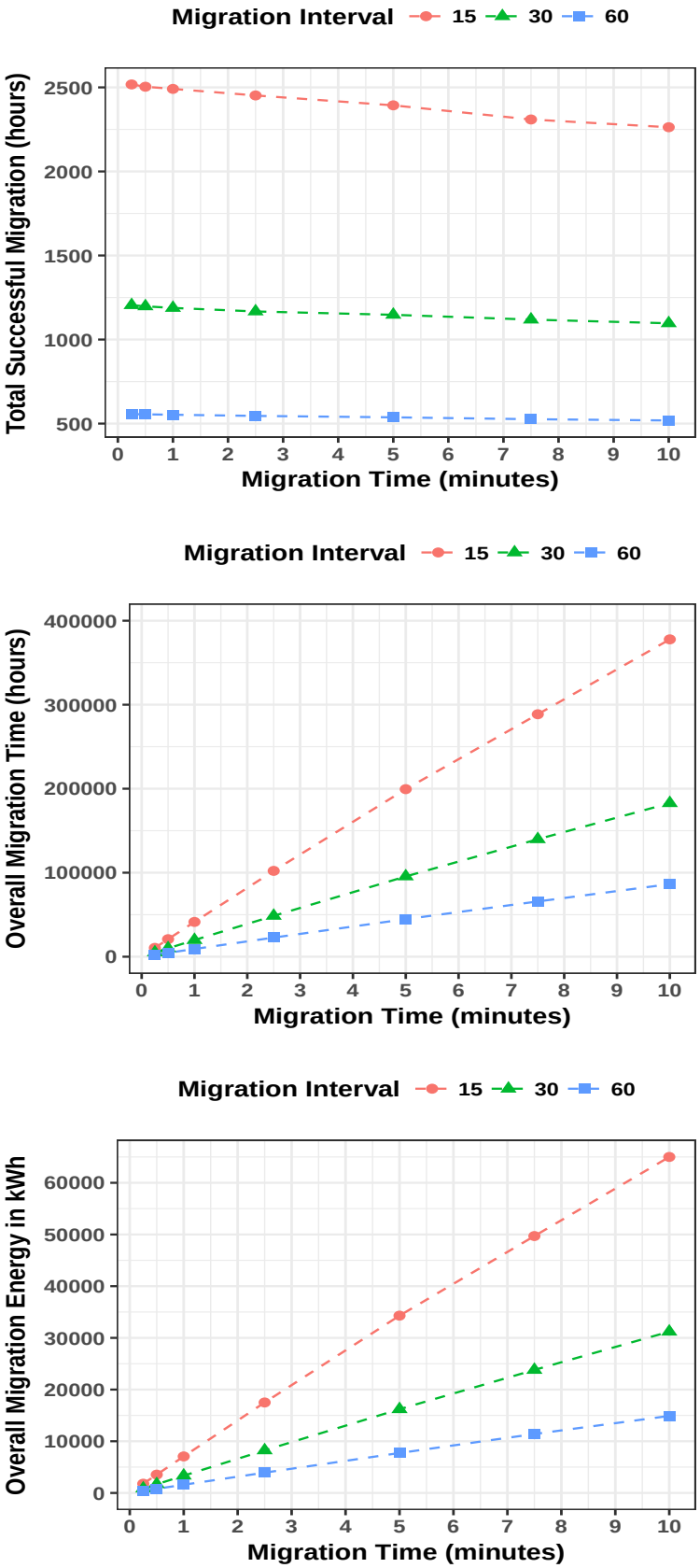

Figure 6.8: Total migrations, overall migrations time, and overall energy consumption of successful migrations with different migration durations and intervals.

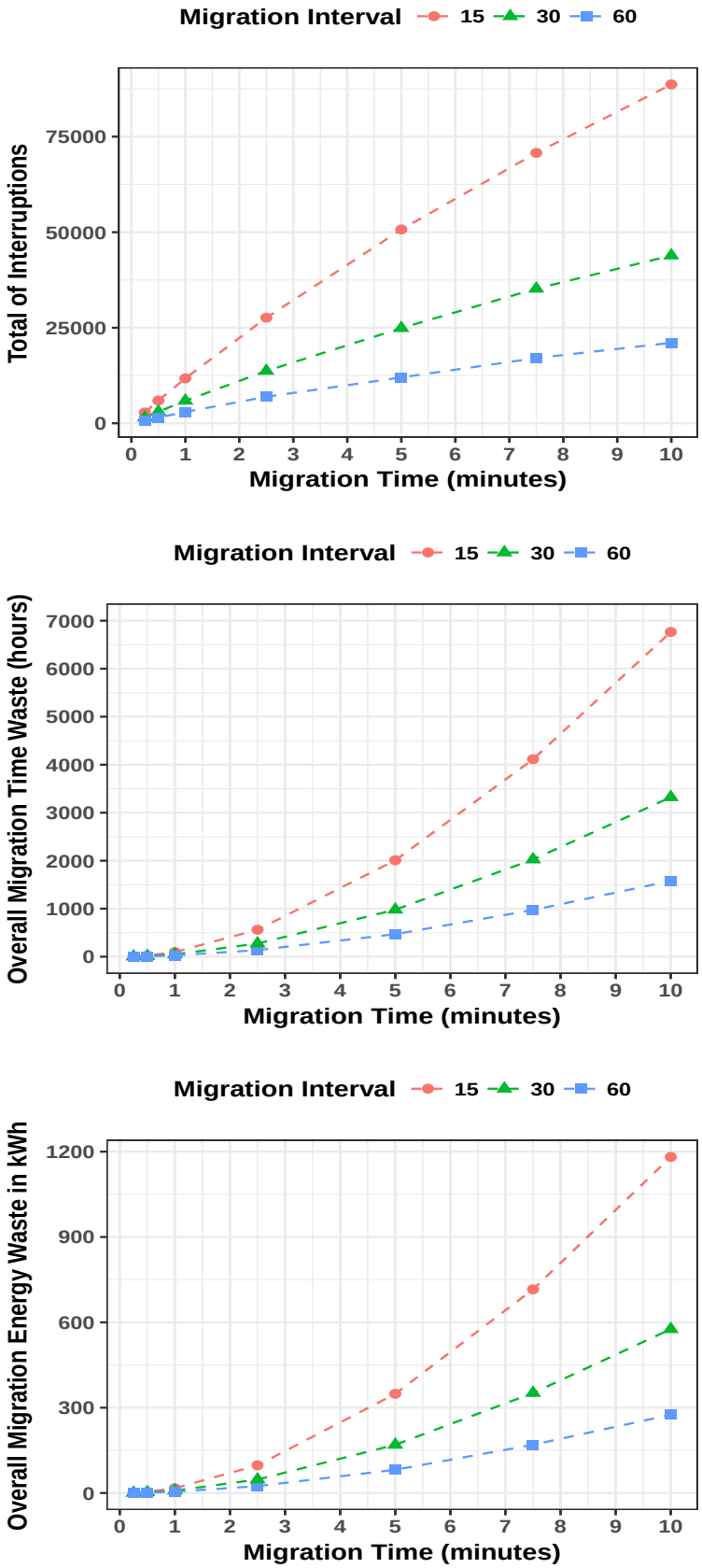

Figure 6.9: Total number of interruptions during migration, overall migrations waste time, and overall waste energy of failed migrations with different migration durations and intervals

Figure 6.9 demonstrates the relative impact of jobs interruptions during the migration process where a user logs in to the source or target computer during the migration. We observe increasing in the number of jobs' interruptions when the migration time of the job increases. As a result, the energy waste of jobs with a long migration time is much higher than jobs associated with short migration time.

Similarly, we observe an increase in energy waste of the removed jobs during the migration when the migration time is long. Thus, it worth to check the opening and closing times of student clusters and design policies according to that to decrease the number of users interruption.

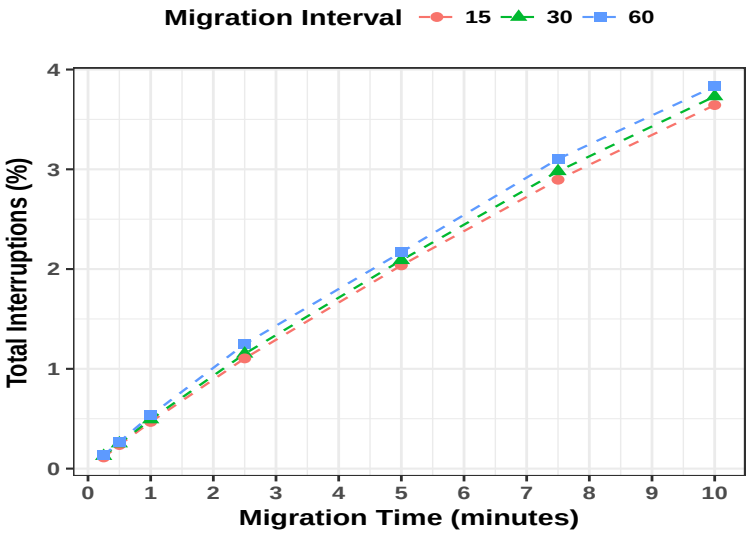

Figure 6.10: Proportion of migrations interruption.

Furthermore, the interactive user can login into the source computer or target computer at any time during the migration process regardless of the migration interval value. Since we are using random migrations and random target computers to host the migrated job, we cannot predict when the users' login into the system will occur. Consequently, the chance of interruptions compares to the number of migrations almost the same between the different migration interval values as illustrated in Figure 6.10. The figure shows the proportion of migrations interrupted in the system. Clearly, the number of interactive users in the system makes an impact on migration success, but there is no immediate link between the interval values and the users' interruptions. As a result, it is crucial to determine when and where to migrate to avoid unsuccessful

migrations.

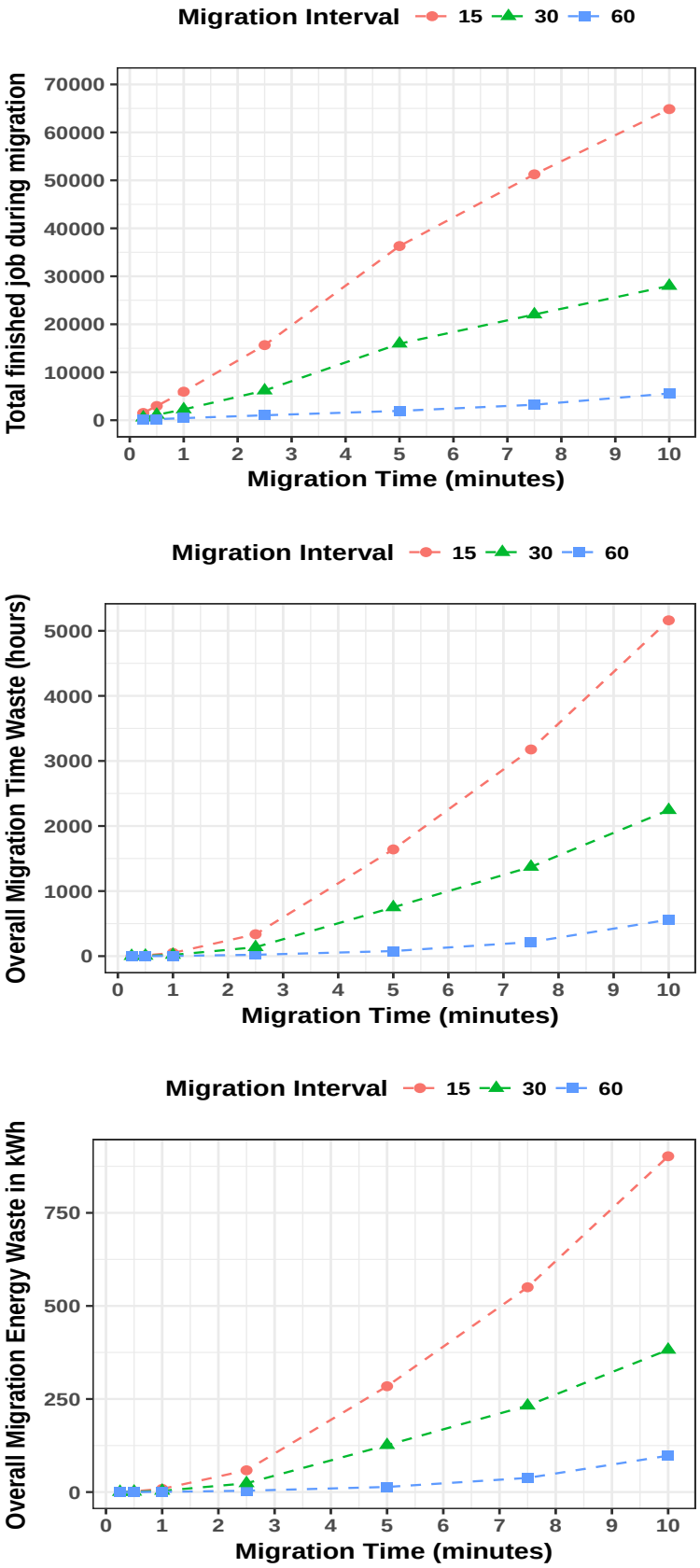

Figure 6.11: Total number of jobs finished during migration, overall migrations waste time, and overall waste energy of failed migrations with different migration durations and intervals

In Figure 6.11, we explore the impact of the finishing jobs during the migration process. In this situation, the jobs migrations are considered to be worthless migrations. It increases the overall energy consumption of the system. Also, it reduces the number of available machines in the system pool. As a consequence, the jobs might be queued for a long time waiting for an idle computer to become available.

Furthermore, number of migration attempts failed due to an unavailable machine for job migration. Our simulation can count these failed attempts during its runtime which gives us the opportunity to understand how many resources are usually available for migration when it is needed.

6.4.2 Responsive Migration

Here, we show the response migration method outcomes with different migration proportion values. We do not consider any interval migration value since the migration occurs when the user logs into a computer that is executing a Condor job. However, user interruption might occur if the user logs into the target computer during the migration process. As a result, the migration fails, and the job gets evicted from the source computer and rejoins the queue.

Figure 6.12 presents the results of successful responsive migrations with various migration durations and migration proportions. The figure shows the total number of successful job migrations, overall job migration time, and the energy consumption that jobs utilised while migrating. The results illustrate a notable reduction in the total number of successful migrations when the migration proportion is associated with a small value. Hence, the total attempted migrations are low when the value of the proportion migration is small.

In addition, the time of migration has a significant impact on the total of successful responsive migrations. It is most likely for a job to be interrupted by use in the target computer when a job takes more time to be migrated. During our analysis, we discovered that up to 15.7% of responsive migration attempts could be interrupted by users when the time of migration is 10 minutes. Thus, it is essential to determine where to migrate jobs in order to prevent an inefficient migration.

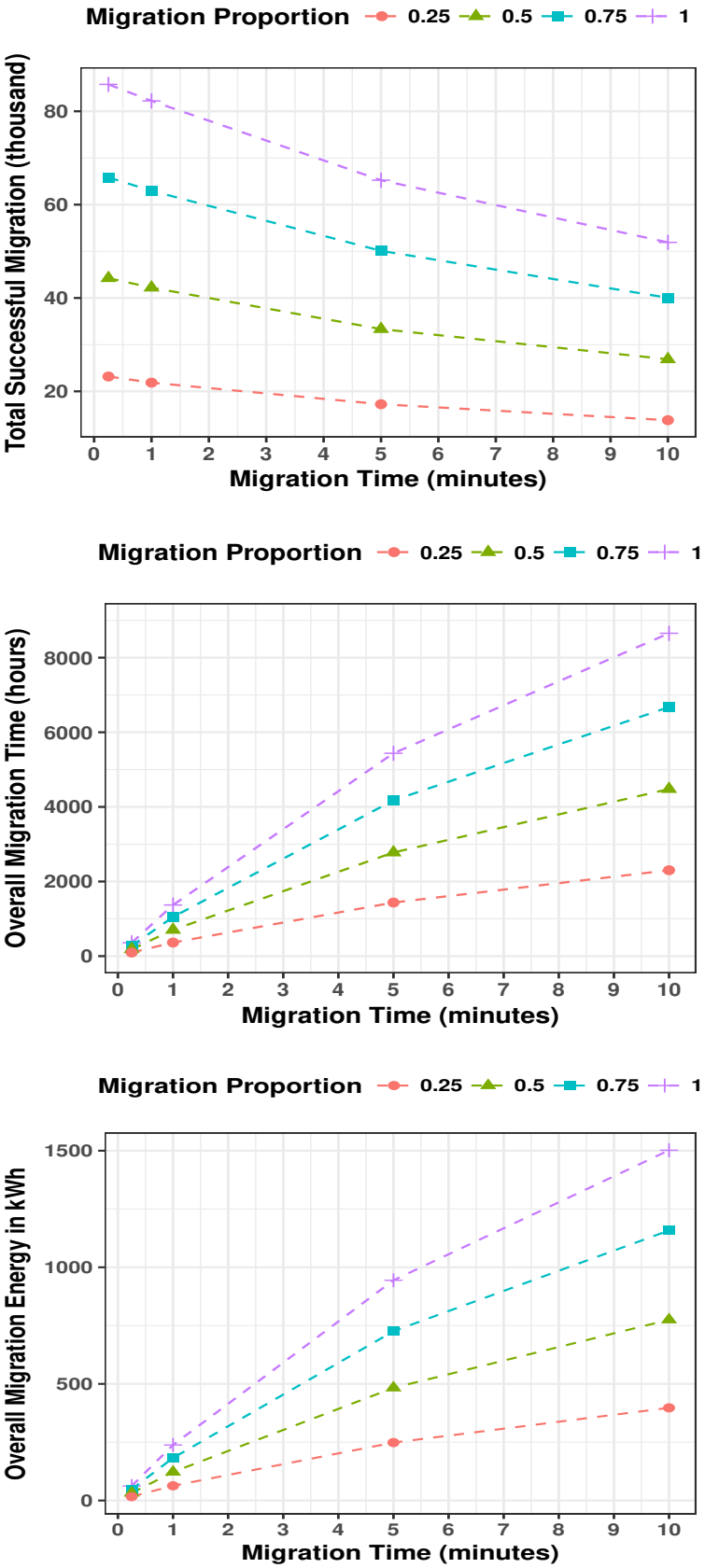

Figure 6.12: Total migrations, overall migrations time, and overall energy consumption of successful responsive migrations with different migration durations and proportions.

Furthermore, the energy consumption of the migration process increases when the time of migration increases. Consequently, the trade-off between the need for migration and the time of migration needs to be considered to achieve a beneficial migration.

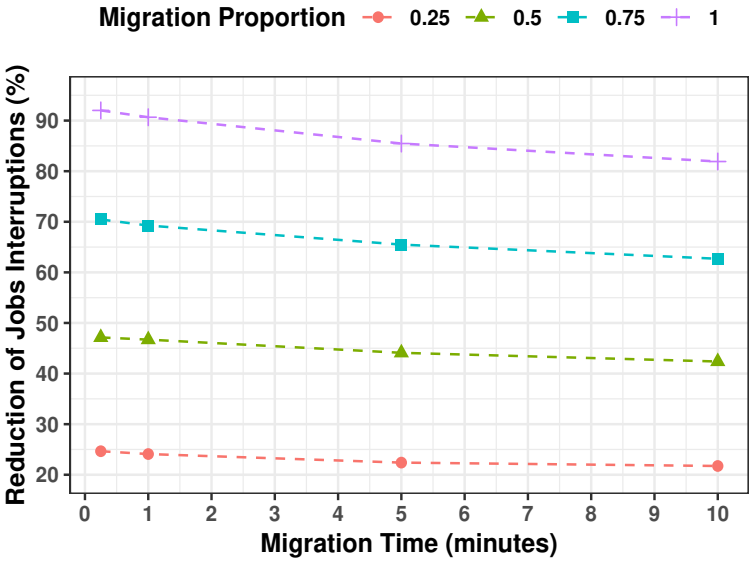

Figure 6.13: Reduction of total interruptions by responsive migration.

In Figure 6.13, we compare the number of interruptions of the system with responsive migrations against a baseline where we do not migrate. The figure exhibits a significant percentage decrease in evicted jobs by user interruptions when all jobs are migratable. However, the simulation shows that the system could decrease about 25% of its evicted job if the only number of migratable jobs is 133,117. Moreover, when the time of migration increases, the reduction of job interruptions decreases.

In Figure 6.14, we compare the energy interruptions waste of the system with responsive migrations against a baseline where we do not migrate. The figure shows the energy interruptions drop of each migration duration for each migration proportion. If all jobs can be migrated when the user interruption occurs, the system could save up to 75% of its energy waste on the baseline where there is no migration occurs. However, the energy saving that gains by migrating the jobs when a user interrupts could be significantly decreased if a 25% of the submitted jobs can be migrated. Moreover, the responsive migration could increase the job execution energy consumption by 2.9% when the proportion migration value is 1 and the time of migration is 10 minutes.

When the job rejoins the queue due to an interruption, the job spends time on the

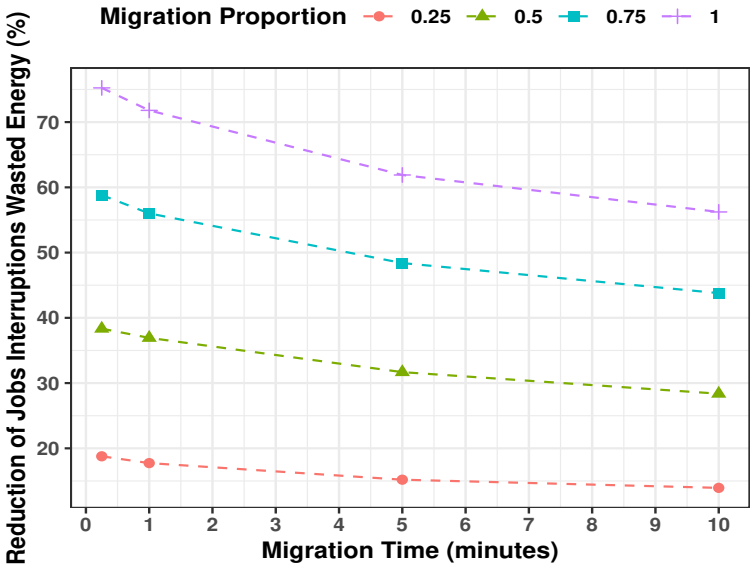

Figure 6.14: Overall interruptions energy saved by responsive migration.

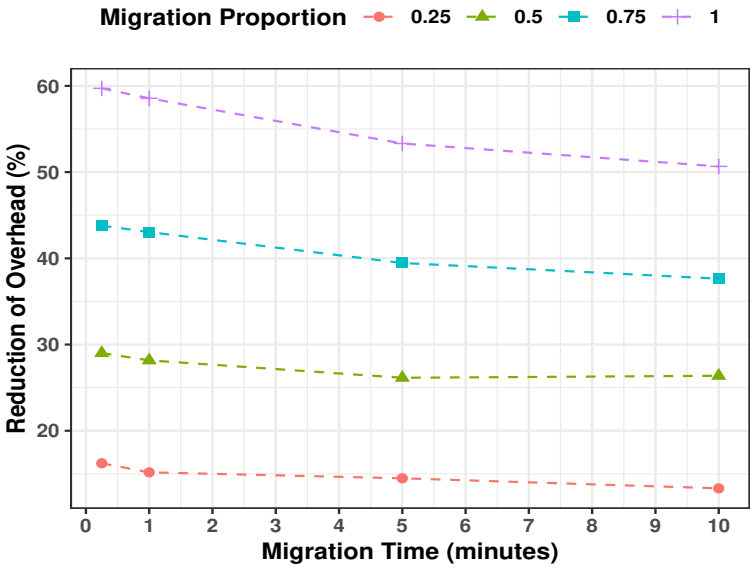

Figure 6.15: Reduction of overhead due to responsive migration.

queue waiting to be reallocated to a new machine. Also, the job loses its progress and starts re-executing when it is allocated to the new machine. As a result, the jobs makespan on the system increases which rise the energy consumption and overhead of the system. In Figure 6.15, we expose the impact of responsive live migration on the overhead. The responsive live migration could be reduced by approximately 60% when the time of migration is 15 seconds and all jobs considered to be migratable. Consequently, there will be more available resources to allocate new jobs as well as

the jobs that cannot be migrated which lead to increase the overall performance of the system.

6.5 Conclusions

This chapter has presented the virtualisation and live migration in HTC systems by extending an existing simulation tool. This allows researchers to evaluate policies and algorithms in order to reduce the energy consumption and jobs makespan by using live migration techniques in HTC environment. A simple random policy has validated our outcomes of the tool. The tool can calculate the overall migration time, the overall energy consumption and waste, and the number of failed migration due to interactive users interruptions or removed jobs. Also, the simulation can count the number of migration attempts when there is not host for migration. Moreover, our results showed that the energy consumption significantly increases when the migration process takes a long time to be finished. We demonstrate that our responsive migration could reduce the wasted energy by 75%.

Through this chapter, we have suggested some approaches to mitigate the energy waste of live migration in a large-scale computing environment. In the next chapter, we will use the responsive migration as a fault-tolerance mechanism. Also, we will introduce policies to determine the selection of target computers when the migration is needed. The policies show the trade-off between energy and performance and aim to reduce energy consumption and the makespan of jobs by avoiding job interruptions.

7

Responsive Live Migration in HTC Environments

Contents

Summary

In this chapter, we use the responsive live migration as a fault-tolerant mechanism to prevent the job from been evicted and restarted in the event of interactive user interruption. We demonstrate the performance and energy impact of our responsive live migration approach in high throughput computing (HTC) systems with various migration policies. The proposed policies focus on which computer to select for migration from HTC resource pool. We extend our HTC-Sim simulation framework previously introduced in Chapter 6 to support six different migration policies. Moreover, we compare the results between the policies as well as the system where migration is not considered.

7.1 Introduction

High Throughput Computing (HTC) systems are commonly used to execute a large number of tasks associated with long running time. HTC is a fundamental approach to fully use the available idle resources within an institution or a company. As a consequence, faults might occur at any point in the system. Especially, when the resources of the system are considered to be multi-user resources included of faculty and student machines, where tasks could be interrupted during their execution by users login to start using the machines. As a result, the tasks get evicted from the resource and re-execute on another machine. Such interruptions lead to increase the energy consumption of the system as well as the makespan of the task.

The reactive fault-tolerance techniques such as Checkpointing are commonly used in HTC environments to handle the event of a failure. However, checkpointing could significantly increase the overhead of the HTC system. The Los Alamos National Lab (LANL) study [136] showed that the overhead of the checkpointing in petaflop systems could extend a 100-hour job without failure to 151 hours. Also, it is difficult to avoid unnecessary snapshots of the task which increase the latency and the energy computation of the task.

From Figure 6.1 and Figure 6.2, we can observe a high probability for a job to be

interrupted by an interactive user. However, it is hard to predict when a job could be submitted or an interactive could login into a computer. For that reason, we propose a mechanism in the event of user job interruption to prevent jobs from being evicted and rejoin the queue. To achieve this, we migrate the interrupted jobs to other computers for resuming their execution. In this way, we can avoid the large latency which may cause by the traditional reactive fault-tolerance techniques as well as the wrong prediction that might occur by proactive fault-tolerance techniques.

In this chapter, we use the responsive migration method mentioned on Chapter 6 as a responsive fault-tolerance mechanism to handle failures that caused by user interruption. We propose migration policies to reduce the number of jobs failure by user interruption. Accordingly, the system resources will be utilised efficiently, and their wasted energy will be much less.

The policies focus on which resource to select in order to migrate the interrupted job. We investigate the impact of each policy on the performance and energy by using HTC-Sim simulation. Also, we compare the migration policies with the system where the migration is not considered.

The rest of this chapter is structured as follows. Section 7.2 introduce the proposed policies. We illustrate the results of the simulation in Section 7.3. We provide a discussion to the HTC administrator to take them into account when they use migration as a fault-tolerance mechanism in Section 7.4 before concluding in Section 7.5.

7.2 Policies

In this section, we introduce and discuss six live migration policies which can determine the decision of selecting a target computer for job migration. For each policy, we explain how it works and how we would expect the performance and energy of the system to be affected.

When the job is interrupted by an interactive user, the policies first try to find an idle computer from the HTC resource pool. If an idle computer is found, the computer will be reserved, and the job migration process begins. If there is no idle computer in

the HTC resource pool, the policies will wake up a sleeping computer and reserve it for the migration process.

In addition, the HTC resource pool has a limited number of resources. At some point, the resources can be fully utilised by interactive users and HTC jobs. As a result, there will be no machine to reserve when job migration is needed. In such a situation, the job will be removed from the source computer and will rejoin the queue.

Random: is a simple policy which chooses the target computer randomly from the HTC resource pool. We use this policy as a baseline against which the competitiveness of our other proposed policies might be assessed.

Prefer closed clusters: this policy targets closed cluster for interactive users. Each HTC resource is allocated within a cluster, and each cluster has a pre-defined opening and closing times. When there is a need for migration, the system sorts closed clusters in descending order based on their opening time. Then, the system selects a computer from a cluster which opens last for users. Furthermore, the migration might occur while all clusters are opened to users. In this situation, the system sorts the opened clusters in ascending order based on their closing time. Then, a computer will be selected from a cluster that is associated with the earliest closing time.

Migrating interrupted jobs into a closed cluster or cluster that is most likely to be closed soon will reduce the chances from getting the job interrupted by an interactive user during or after migration.

Prefer closed clusters quiet: some clusters within the university have the same opening and closing times. When there is no computer available in a closed cluster for migration, the system will select a computer from one of the opened clusters as mentioned in the previous policy. The opened clusters which have identical opening and closing times might vary in the number of available machines. As a result, the system calculates the percentage of the available computers within each cluster. Then, a computer will be selected from the cluster that has the highest availability percentage value.

Choosing a computer from the quietest opened cluster when there is no closed cluster might enhance the performance of the previous policy.

Most quiet cluster: this policy does not take into consideration the opening and closing time of the clusters. The system calculates the percentage of the available machines within each cluster and selects a computer from the cluster that is associated with the highest resource availability percentage value. In this way, the system will be allowed choosing a computer from the unbusy opened clusters event if they have the latest closing time.

In the university, some clusters are open most of the day for public users. However, these clusters could be not used by the interactive users especially at late night or during holidays. As a result, this policy targets these clusters, and the job might have a high possibility of not been interpreted during or after migration.

Most quiet cluster eco: some clusters within the university cloud have the proportion of available machine at the time of migration, but their energy consumption characteristics differ. Accordingly, we have optimised the previous policy to select a computer from the most energy efficient cluster when the clusters have an identical availability percentage value.

Enhancing the previous policy to be more energy efficient could reduce the energy consumption of the system. Also, it will not impact the improvement on performance which could be gained by the previous.

Most energy efficient: this policy targets the most energy-efficient computers of HTC resource pool regardless of the opening and closing times for the clusters as well as the resources availability percentage of the clusters.

Selection of the most energy-efficient computers for migration could save energy, but cloud impacts the overall performance of the system, especially if the energy-efficient clusters are the busiest clusters when migration is needed.

7.3 Results

In this section, we evaluate the previously specified policies. We compare the performance and energy impacts of each policy. Also, we explore the improvement of the responsive migration policies on the performance and energy against a baseline where

the responsive migration is not considered. Section 7.3.1 exhibits the migration process results of each policy. Section 7.3.2 shows the improvement of the system on the performance and energy when responsive migration is employed.

7.3.1 Policy performance

Here, we discuss the results of the migration process for each policy. We show how the number of migrations, the overall time of migrations, and overall energy consumption of migrations are varied between each policy. Also, we do not show the impacts of policies on the overall performance and energy of the system. In the next section, we present how the migration policies improve the overall performance and reduces the energy consumption of the system.

7.3.1.1 Number of migrations

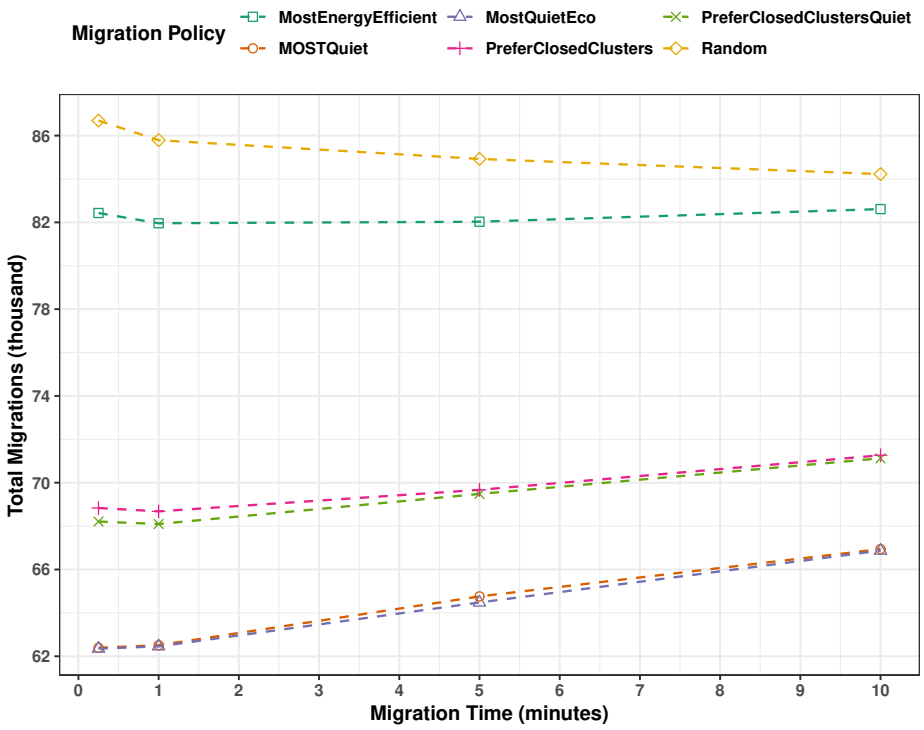

Figure 7.1: Total migrations with various migration time

As we explained in Section 6.3.2, migration occurs when an interactive user arrives into a machine while it is executing an HTC job. The proposed policies vary on determining the target computer for migration, which leads to a significant difference

in the number of migrations as illustrated in Figure 7.1. The figure shows the number of total migrations of each policy with different migration times.

The Random policy has the highest number of migrations where the MostQuiet and MostQuietEco have the lowest migration numbers. The Random policy might select a target computer in a busy cluster for migration where the job is most likely to be interrupted again during or after the migration process. As the result, the number of migrations per job increase which influence the overall total migrations of the policy. However, migrating the interrupted job to a closed or a quiet cluster reduce the possibility of the job being interrupted by an interactive user.

Moreover, as the migration time for a job increases, the number of migrations increases as well. That is because of the high probability for an interactive user to login into the target computer during the migration process, which leads to job eviction. Interestingly, the number of migrations in the Random policy reduces when the time of migration increases. The reason behind this is the system resources at some point are fully utilised when new job interruptions occur, which leads to job evictions rather than job migrations.

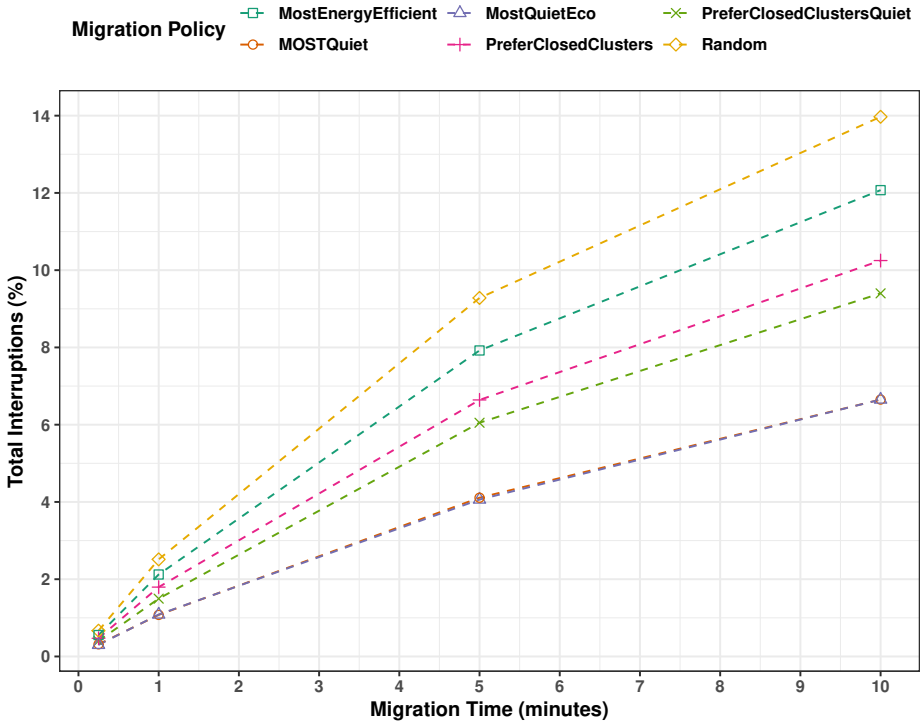

Figure 7.2: Proportion of interrupted migrations

During the migration process, an interactive user can arrive into the target computer. As a consequence, the migration fails and the job rejoins the queue for re-execution on another computer. Figure 7.2 presents the proportion of interrupted migrations by interactive users. We acknowledge a significant relation between the migration time and the number of interruptions as it is most likely for an interactive user to interrupt the migration process when the migration takes a long time to finish. Also, there is a clear decrease in the interrupted jobs during the migration process, when the selected resources are within quiet clusters.

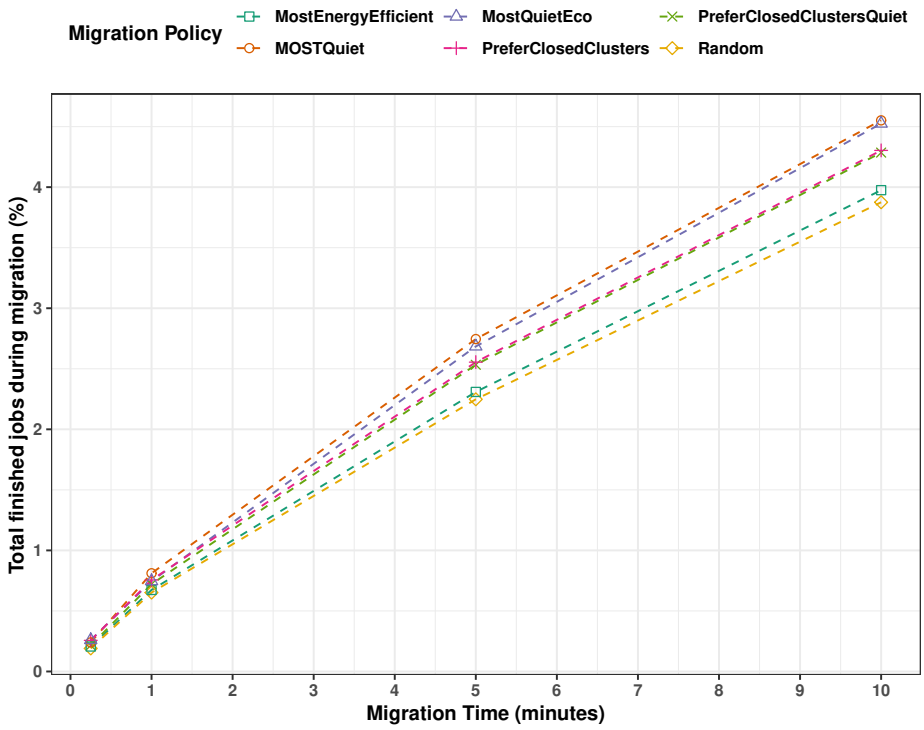

Figure 7.3: Proportion of finished jobs during migration

In our system, the job duration is not predictable. The job might finish at any point and provide the outcomes to its owner. As a result, the job might finish during the migration. Figure 7.3 shows the proportion of finished jobs during migration for each policy. There are no significant differences between the number of the finished jobs during the migration when the migration time is short. The results, as shown in Figure 7.3, indicate that the MostQuiet and MostQuietEco have a higher finished job during migration compared to other policies. Because, the number of evictions per job of these policies is less than the other policies.

Similarly, the interactive user behaviour is unpredictable. The interactive user might leave the interrupted computer during the migration. In this situation, the system cancels the migration, and the job resumes its execution on the source computer. It is apparent from our results that very few cancelled migrations occur when the time of migration is 15 seconds. However, when the time of migration increases, the cancelled migrations become significantly noticeable. The proportion of cancelled migrations for the policies is between 19% to 22% when the migration process takes 10 minutes to complete.

It is unnecessary to start the migration process when the migration time is higher than the execution time of a job. Also, the user might login into a computer and use it for a short time which leads to useless migration. To avoid these kinds of unnecessary migrations, the system needs a proactive mechanism to predict the jobs duration and users behaviour which we will consider it in the future work.

7.3.1.2 Overall migration time

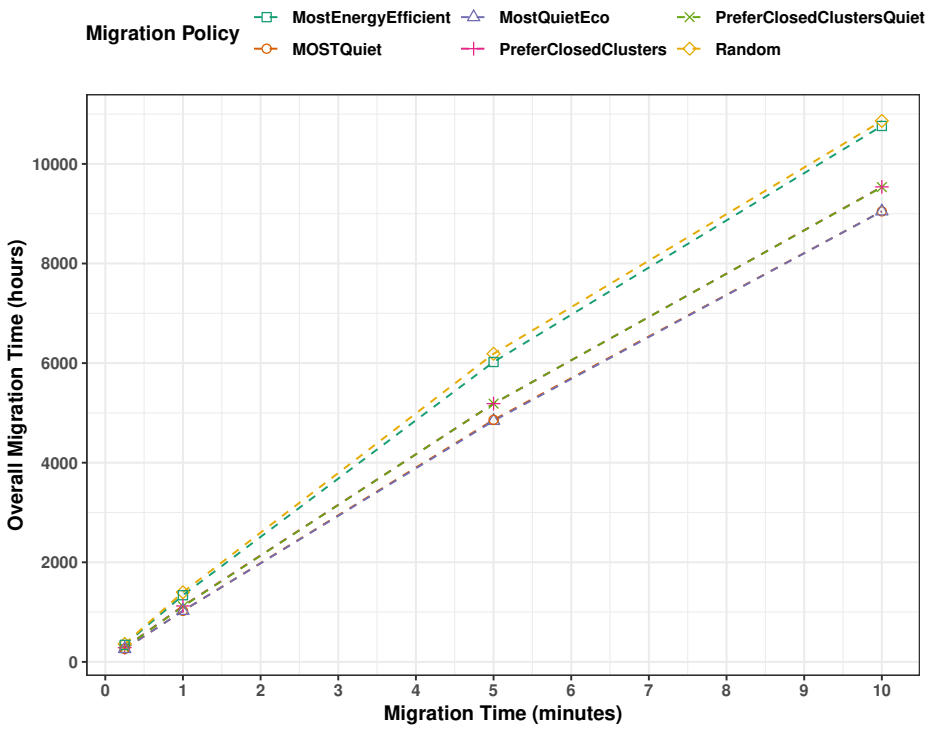

Figure 7.4: Overall migration time

The overall migration time of each policy in the system is based on the total migrations

as well as the duration of the migration process. If a policy is associated with large number of migrations, the policy will have a high overall migration time. In Figure 7.4, we present the overall migration time for each policy. As it is expected the overall migration time for the Random policy is higher than the other policy. Also, the figure strongly presents the impact of the long migration duration on the overall migration time of the system.

The overall migration time of each policy comprises the time of interrupted migrations by the interactive users as well which we call it as migration wasted time due to user interruptions. The migration wasted time is notable when the migration process takes a long time to finish. When a task has been migrating for a long time and an interactive user login only a few seconds before the task finishes migrating, the migration of the task fails and the wasted migration time will increase crucially.

Based on our observation, the migration wasted time is significant when the migration process takes 10 minutes to finish. The migration wasted time is the highest in the Random policy, and it is about 8% of overall migration time where it is the lowest in MostQuiet and MostQuietEco, and it is about 4%.

7.3.1.3 Overall migration energy

The total migration energy of each policy differs due to the number of migrations, migration time, and the energy characteristics of the physical machines which are shown in Table 6.1. The migration energy is the energy which consumed by the source and the target computers during the migration process.

In Figure 7.5, we show the overall migration energy for each policy and migration time. Also, the figure illustrates the link between the migration time and the energy consumption as well as the impact of the total migrations on the migration energy. When the job migration takes a long time, the energy consumption of job migration increases.

Interestingly, there is no direct link between the number of migrations and the overall migration energy consumption. As we can see in the figure, the MostEnergyEfficient policy is associated with a high number of migrations as compared to the MostQuiet

and MostQuietEco policies. However, their migration energy consumption is similar and even slightly better than the MostEnergyEfficient policy when migration time is 10 minutes.

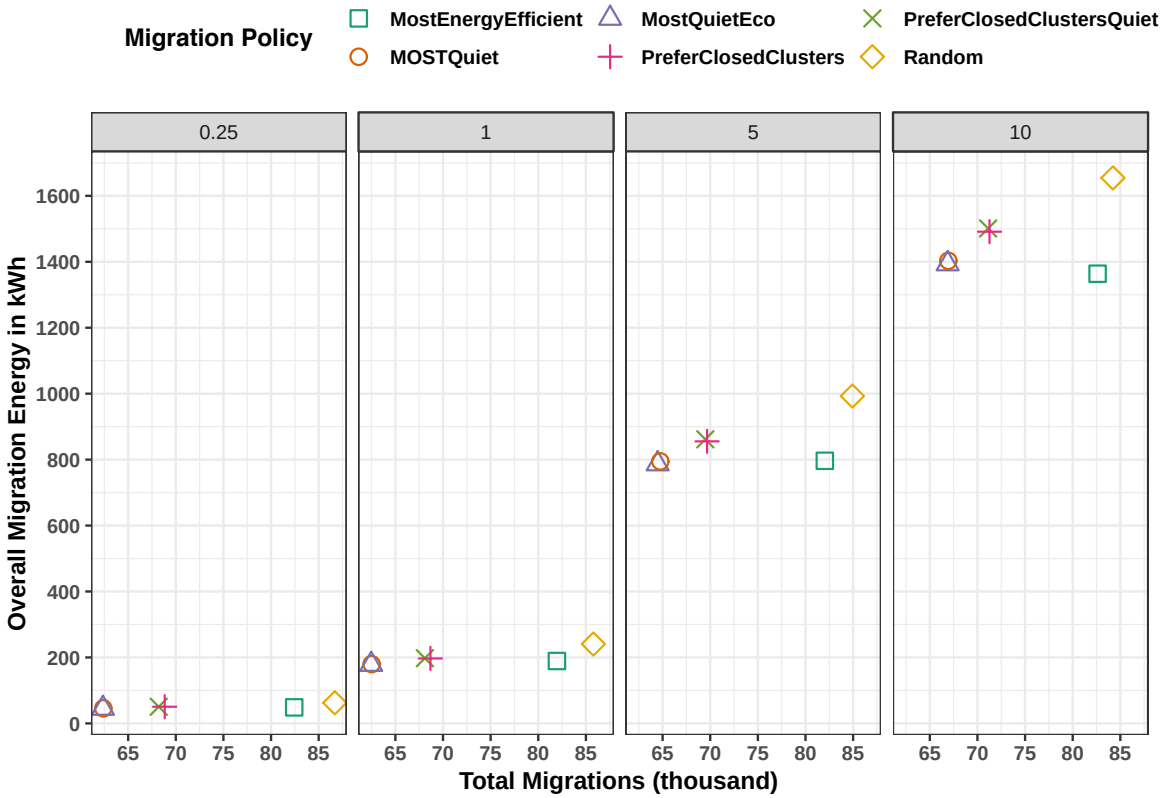

Figure 7.5: Overall migration energy of each policy compared with total migrations and various migration times (0.25, 1, 5 ,10) in minutes as shown in the panels

The reason behind that is the variation of the energy characteristics between the physical machines. For example, the MostEnergyEfficient policy selects the most energy-efficient computer when there is a need for migration. As a result, the overall energy consumption of the MostEnergyEfficient migration policy is less than other policies regardless of the number of migrations. However, the performance of the MostEnergyEfficient is not the best compared to other policies as shown in Figure 7.5. The MostQuiet and MostQuietEco policies are expected to select the High End and Legacy computers as defined in Table 6.1. This is due to the fact that these kinds of computer are located within the quiet clusters which are not frequently used by students.

The trade-off between the energy saving and performance improvement can be clearly observed based on our discussion of Figure 7.4 and Figure 7.5. It is critical to find a

balance between the performance and energy of the system. As a result, we apply the MostQuietEco policy to address this need.

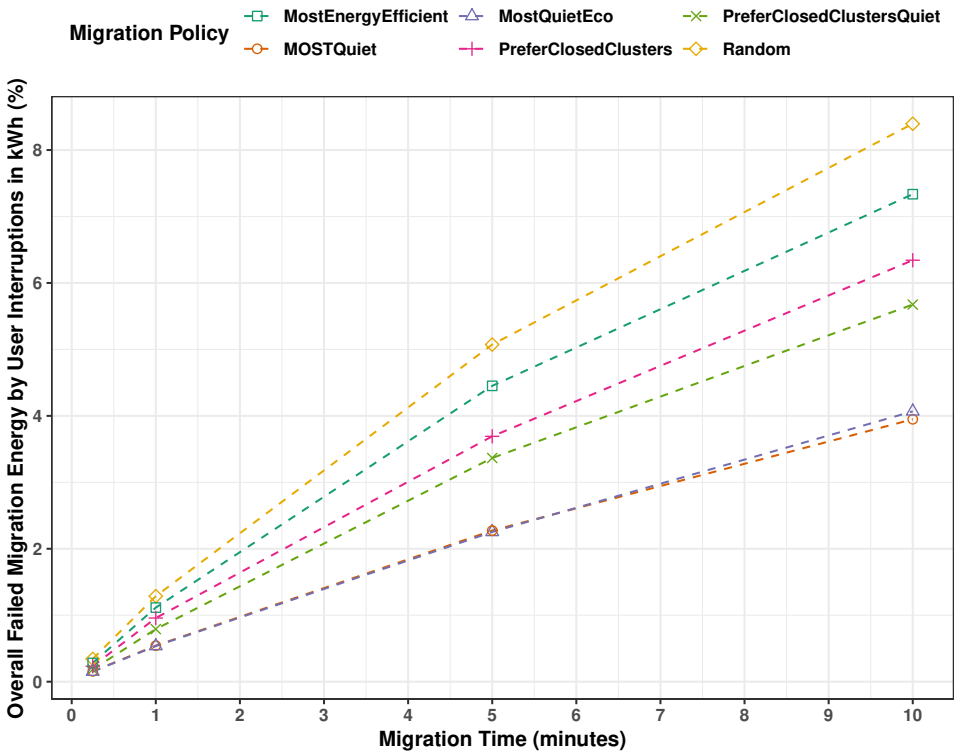

Figure 7.6: Migration energy wasted by users interruption

We have plotted the percentage of the migration energy waste due to the user interruptions on the target computer during migration as shown in Figure 7.6. The migration energy waste is essential when the time of migration is long. Also, migrating the job into closed or quiet cluster increases the percentage of successful migration and the system performs effectively and efficiently.

7.3.1.4 Number of migrations per job

Here, we present the impact of each policy on the number of migrations per job. Figure 7.7 shows the empirical cumulative distribution function (ECDF) of the total migrations per job for each policy with different migration time. It is clear that the number of migrations per job is less when the MOSTQuiet and MOSTQuietEco policies are used. However, for all policies, when the migration process takes a long time, the number of migrations per job increase as well. Also, the figure exhibits that few jobs within each policy are associated with a large number of migrations.

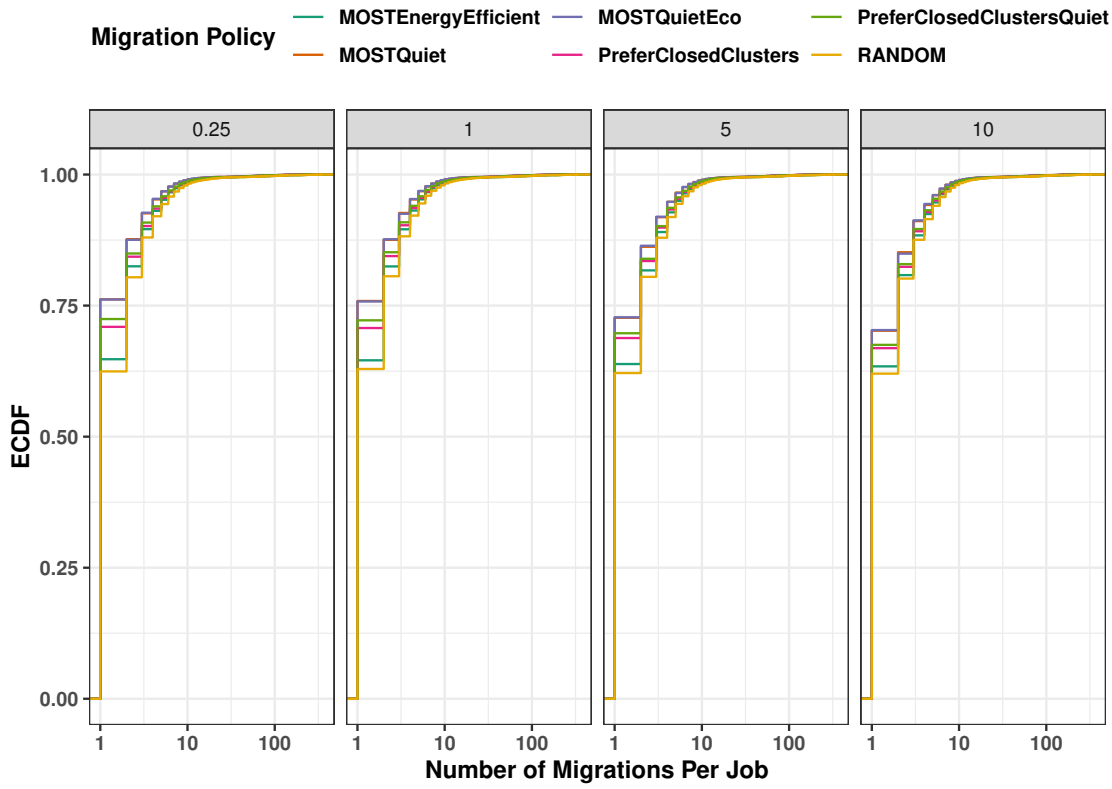

Figure 7.7: Empirical cumulative distribution function of migration numbers per job of each policy with various migration times (0.25, 1, 5 ,10) in minutes as shown in the panels

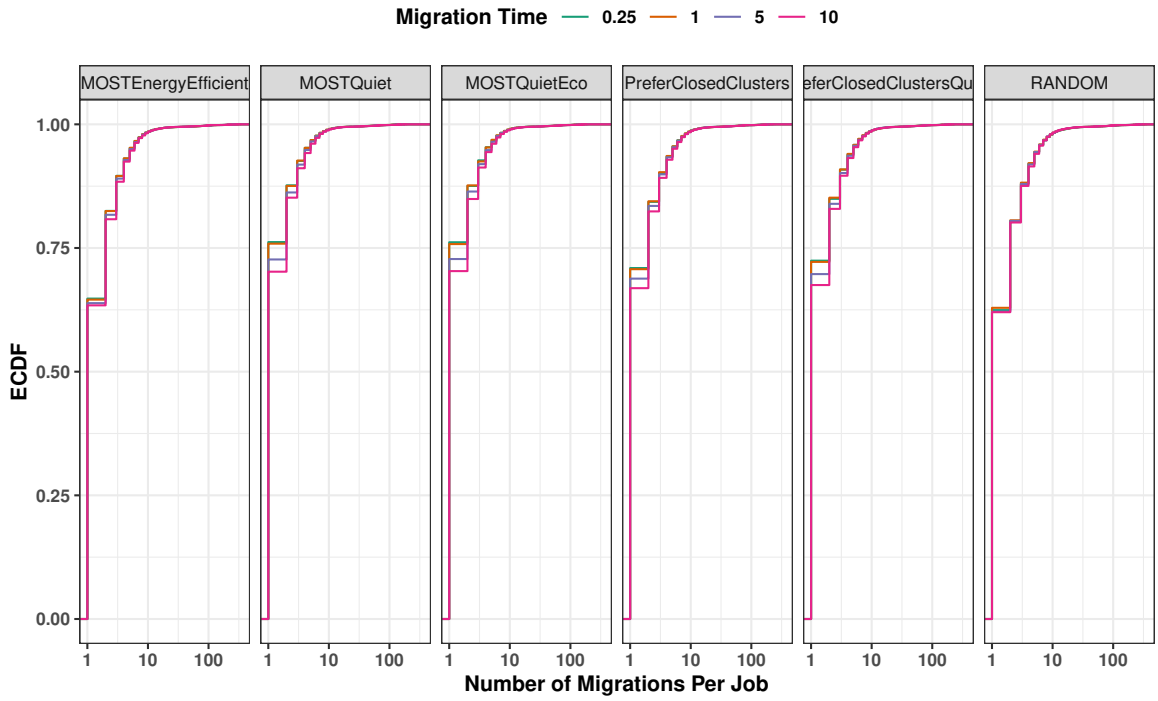

Figure 7.8: Empirical cumulative distribution function of migration numbers per job of each migration time with various policies shown in the panels

In addition, we plot another ECDF where each facet is the policy, and the different lines are the migration times to understand those trends better as illustrated in Figure 7.8. The time of migration within some policies such as Random and MostEnergyEffient has no impact on the number of migrations per job. However, on the other policies, the number of migration per job decreases when the migration process takes a short time to complete.

7.3.2 System improvement

Here, we compare the system with responsive migrations against a baseline where we do not migrate. We show the impact of each responsive migration policy on the overall performance and energy.

7.3.2.1 Number of evictions

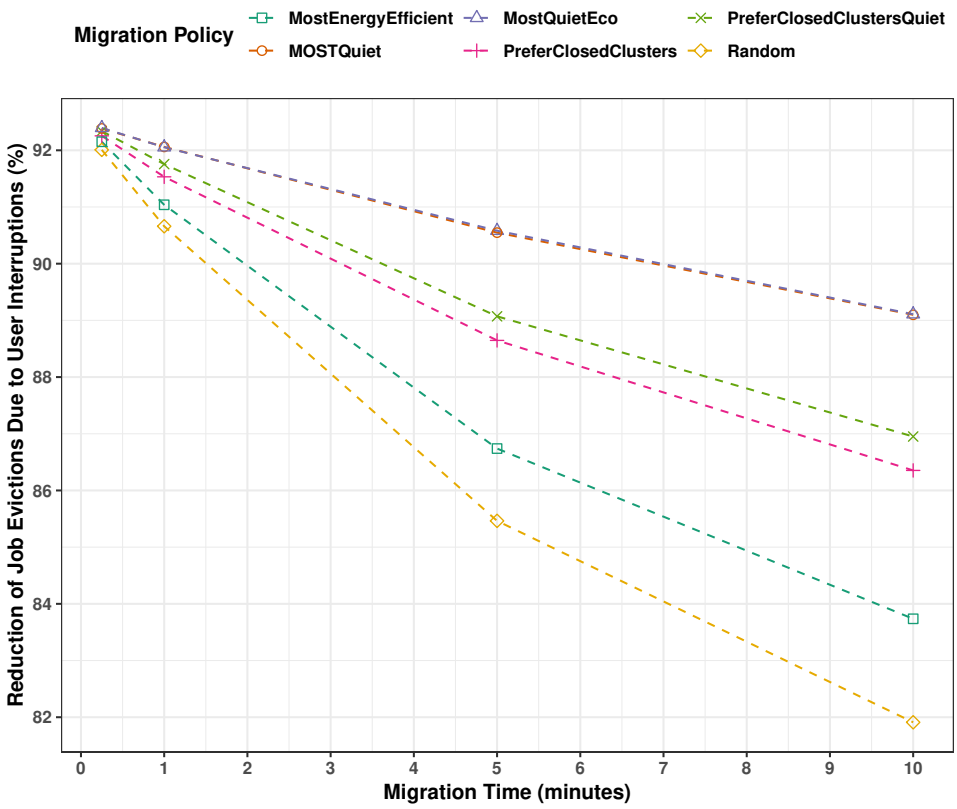

Figure 7.9: Reduction of total interrupted job evictions

The primary aim of this chapter is to reduce the number of evicted jobs in the HTC systems where the job rejoins the queue for re-execution. The total evicted jobs due to

users interruption is 102,165 evictions where the responsive migration is not employed as a fault-tolerance mechanism.

As Figure 7.9 shows, the responsive migration can approximately reduce the number of job evictions by 92% when the time of migration is 15 seconds. Also, no significant differences were found in the number of job evictions between the policies when the migration duration is 15 seconds.

Moreover, the figure exposes a lower reduction of job evictions when the time of migration increases due to the user interruptions on the target computer during migration. As the figure shows, the MostEnergyEfficient policy has more job evictions than other policies except for Random policy. Because the most energy efficient machines in the Newcastle University are allocated within busy clusters such as the library which makes the job most likely to be interrupted during migration.

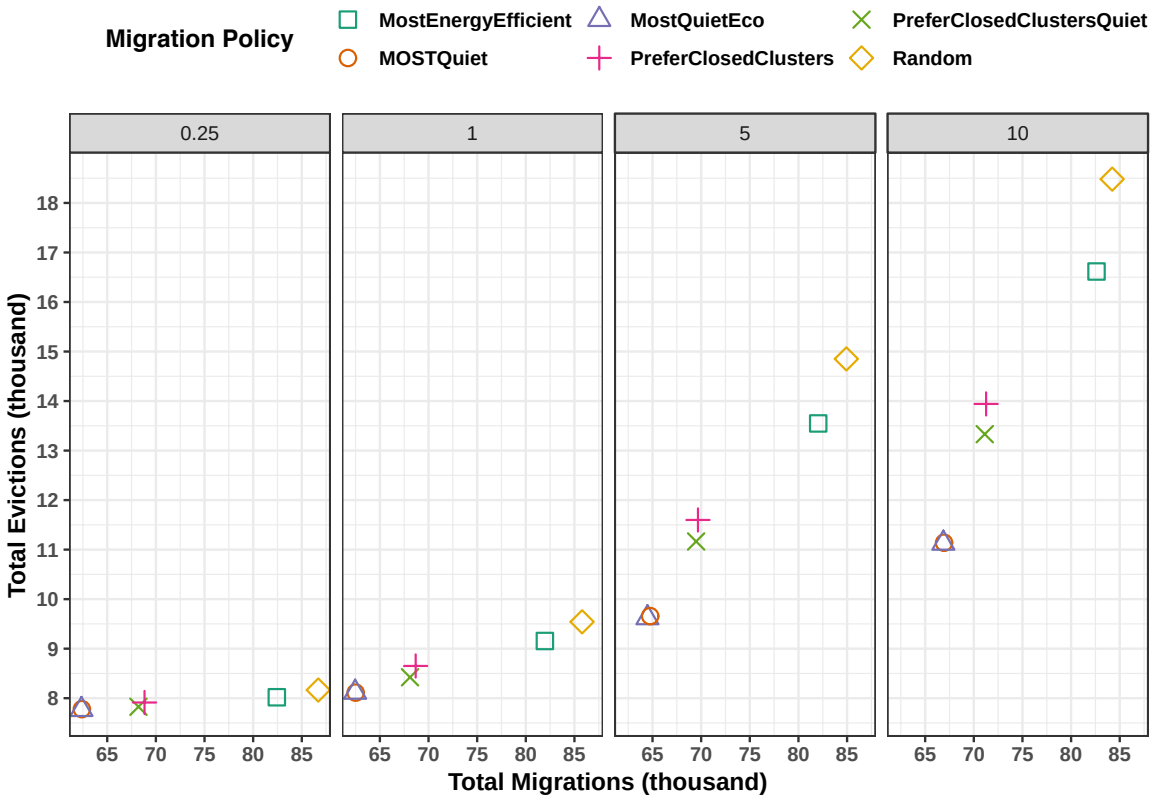

Figure 7.10: Total evictions of each policy compared with total migrations and various migration times (0.25, 1, 5 ,10) in minutes as shown in the panels

Interestingly, for each policy, there is a significant difference between the total migrations compare to the number of evictions. Figure 7.10 present the link between them.

When the time of migration is 15 seconds, the variation on the number of job evictions is small. However, the total migrations is crucially varied between the policies. The number of migrations per job increases when the job is migrated into a busy cluster, which raises the possibility of getting it interrupted again by users.

7.3.2.2 Overhead

We calculate the overhead for jobs within the system as given by the difference between the execution time and the amount of time the job spend to finish in the system. In the event of job eviction, the job loses its progress and starts re-executing when it is allocated to a new machine. The job might spend a long time in the queue while it is waiting for an idle computer to become available. As a result, the job's makespan on the system rises which increases the energy consumption and overall overhead in the system. Figure 7.11 exhibits the impact of each responsive migration policy on the overhead.

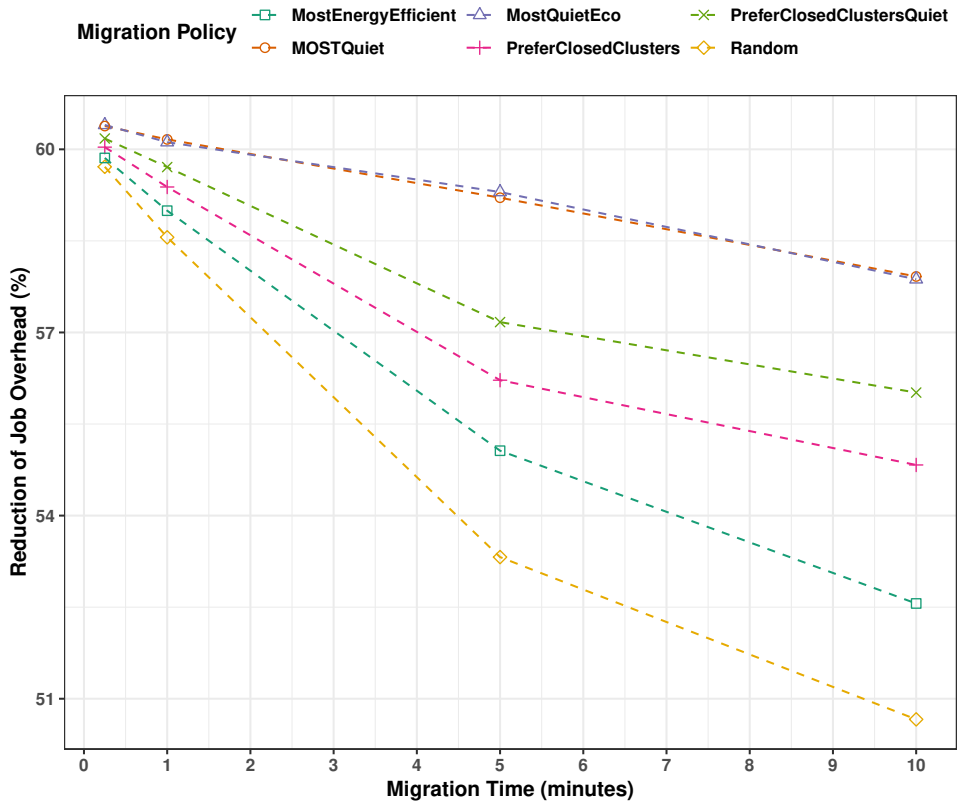

Figure 7.11: Reduction of overhead

The figure shows no high differences between the policies on the overhead when the

time of migration is 15 seconds. However, when the time of migration increases, the overhead increases as well. Comparing the overhead of each policy when the time of migration increases, it can be seen that the overhead of the MostQuiet and MostQuietEco is less compared with other policies.

In addition, it is notable that the reduction on the overhead does not vary significantly between the policies when the migration process takes a short time. However, if we compare the number of migrations across the policies, we will find a notable variation between the policies as shown in Figure 7.12.

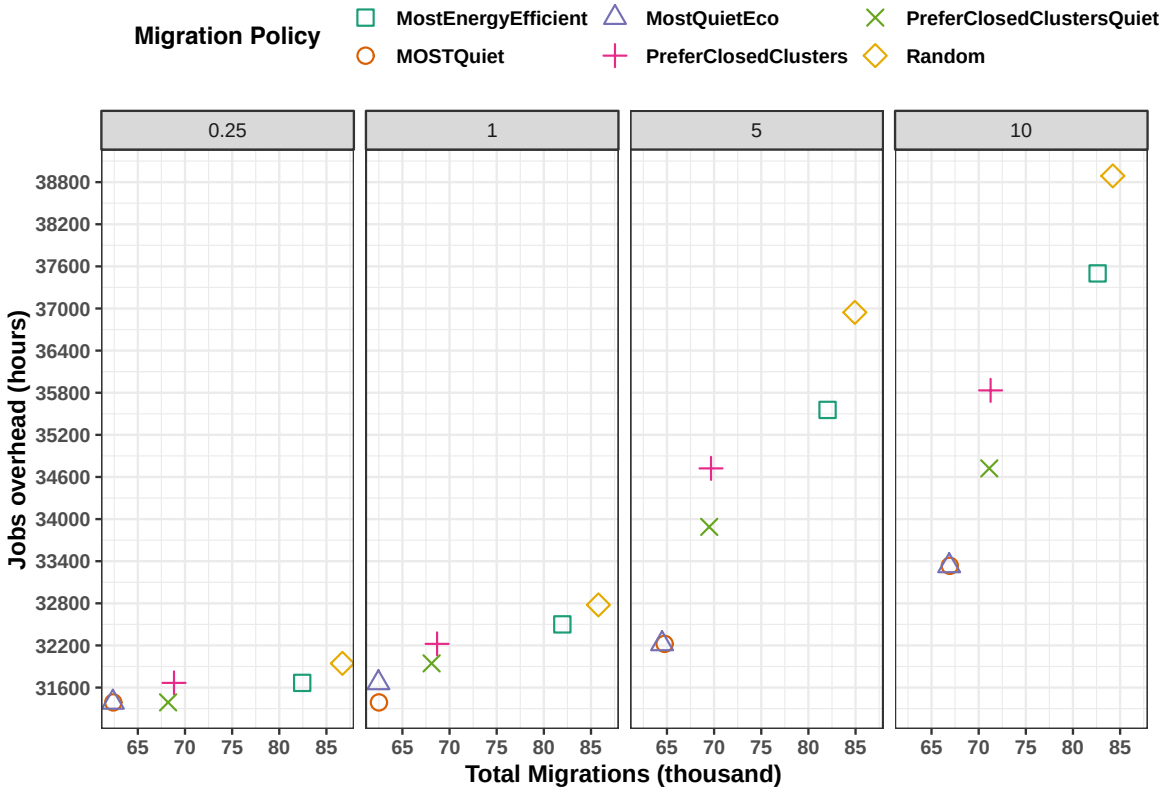

Figure 7.12: Jobs overhead of each policy compared with total migrations and various migration times (0.25, 1, 5, 10) in minutes as shown in the panels

7.3.2.3 Energy

When the responsive migration is not used as a fault-tolerance mechanism, the energy waste of job evictions due to user interruptions is 14,442 kWh. The responsive migration policies could approximately save 75% of the evicted jobs wasted energy due to user interruptions if all jobs are migratable and the time of migration is 15

seconds as shown in Figure 7.13. Also, there were no significant differences between the MostQuiet and MostQuietEco on the saved of evicted jobs wasted energy due to user interruptions.

Furthermore, our simulation results showed that the policies might save approximately 38% of the evicted jobs wasted energy due to user interruptions if 50% of jobs are migratable and the time of migration is 15 seconds.

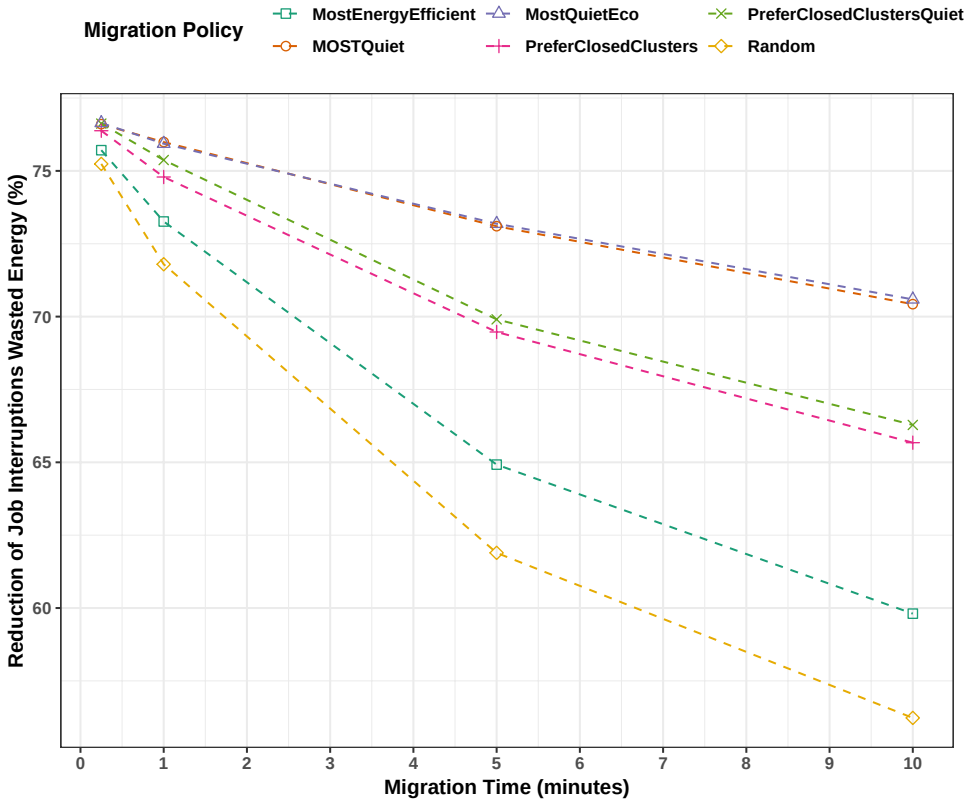

Figure 7.13: Overall interruptions energy saved by responsive migration

In addition, we have explored the impact of each policy on the jobs energy consumption as illustrated in Figure 7.14. To obtain the energy consumption values of the computer we used the information mentioned on Table 6.1. These then multiplied by the amount of time which a job spent executing in the system to calculate the total jobs energy consumption under a given policy. Here we assumed that the load level is 100% for jobs execution.

Figure 7.14 presents the jobs total energy migrations of each policy and each migration time. Interestingly, for those policies aimed to select quiet clusters, their jobs energy

consumption are higher than PreferClosedClusters and MostEnergyEfficient policies. The reason behind that is the quiet clusters within the university do not contain energy efficient machines. The busiest clusters within the university such as libraries have the most energy efficient machine. For that reason, the MostEnergyEfficient policy associated with a high number of migrations. Also, the figure shows that the PreferClosedClusters policy consumes slightly less energy than MostEnergyEfficient policy due to the number of migration which is much less in the PreferClosedClusters policy.

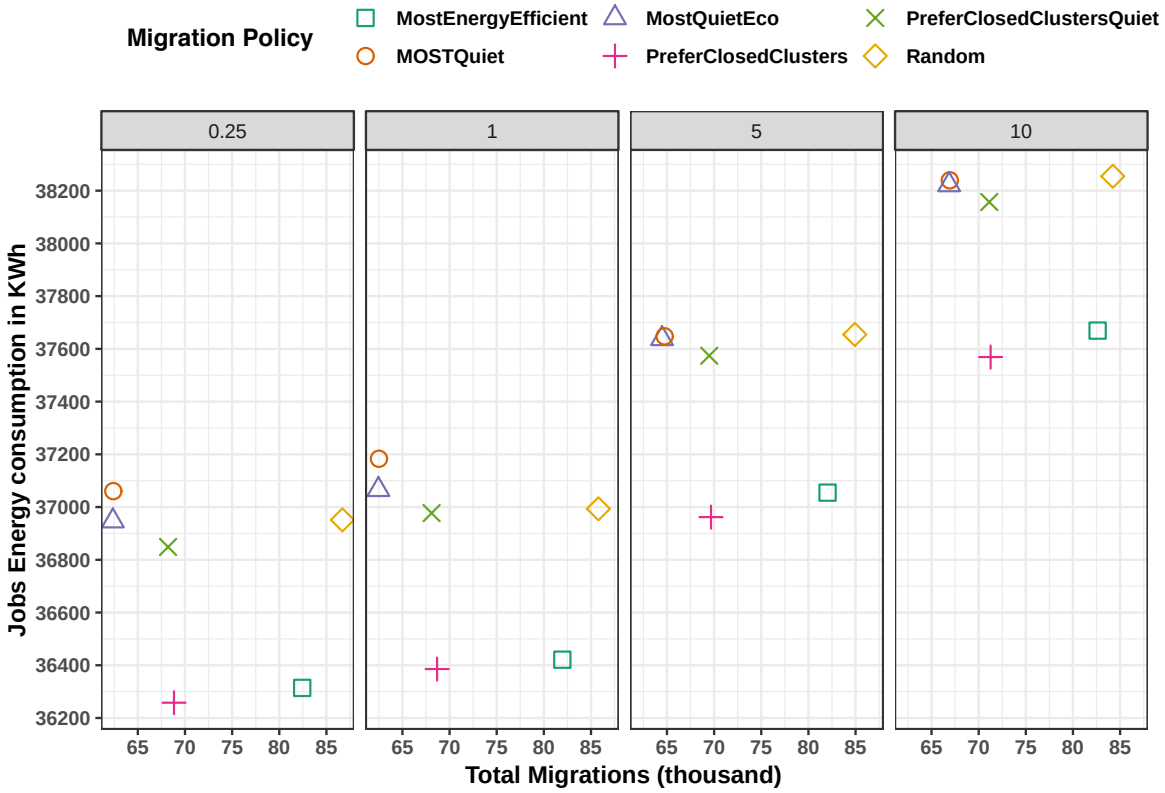

Figure 7.14: Jobs energy consumption of each policy compared with total migrations and various migration times (0.25, 1, 5, 10) in minutes as shown in the panels

7.3.2.4 Killed jobs

In the system, the job can be killed by its owner or the administrator of the system. The user could decide to stop the job from running at any time during its execution. Also, some of the assigned jobs have an error which needs to be removed by the administrator. The killed jobs might run for a long time in the system before they

get killed and removed. As a result, these jobs increase the utilisation and the energy consumption of the system. Also, our responsive migration could migrate them many times before they get killed.

In general, when we migrate the jobs in the event of user interruption, the jobs are most unlikely to rejoin the queue for re-submission. As a consequence, the system resources will be utilised efficiently. However, this would be a problem when the killed jobs by users keep running for a long time. The user interruptions will occur, and our responsive migration mechanism will keep targeting these jobs for migration. That will increase the resource utilisation as well as energy consumption.

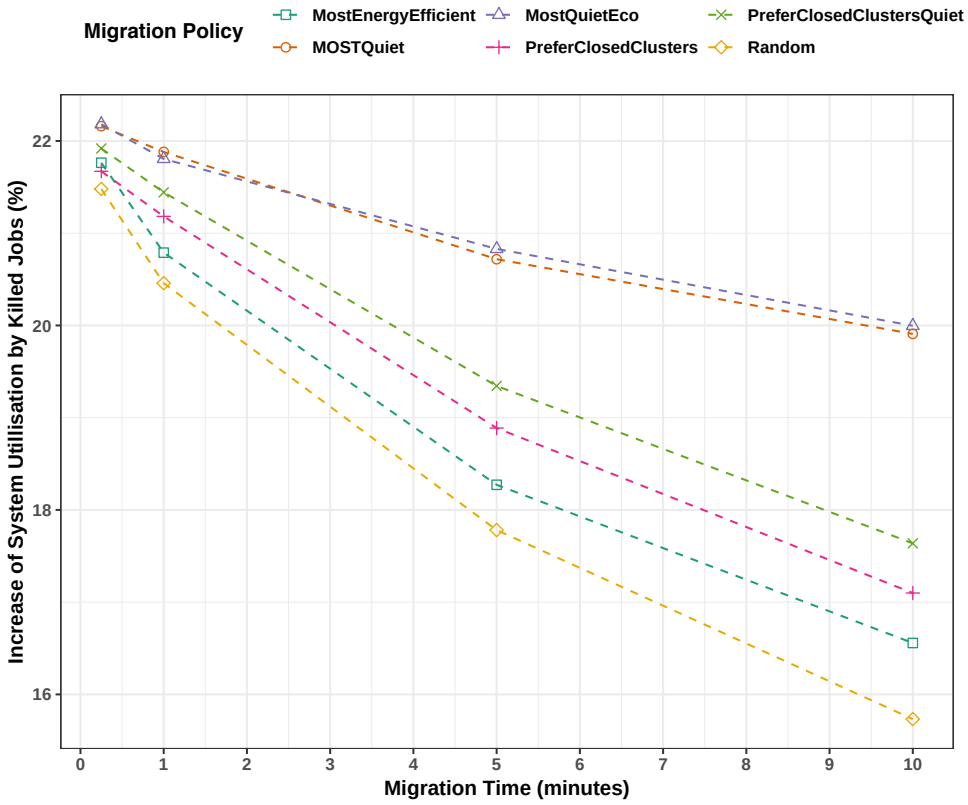

Figure 7.15: Killed jobs resource utilisation

In Figure 7.15, we compare the utilised resources by killed jobs between responsive migration policies and the baseline. When a policy is associated with a lower number of evictions due to the user interruption, the resources utilisation of the system increases. The reason behind that is the killed jobs with the policies spent less time in the queue compared to the baseline. The system would be more effective if we can predict when the job most likely to be killed. We will address this in the future by employing a

proactive migration mechanism.

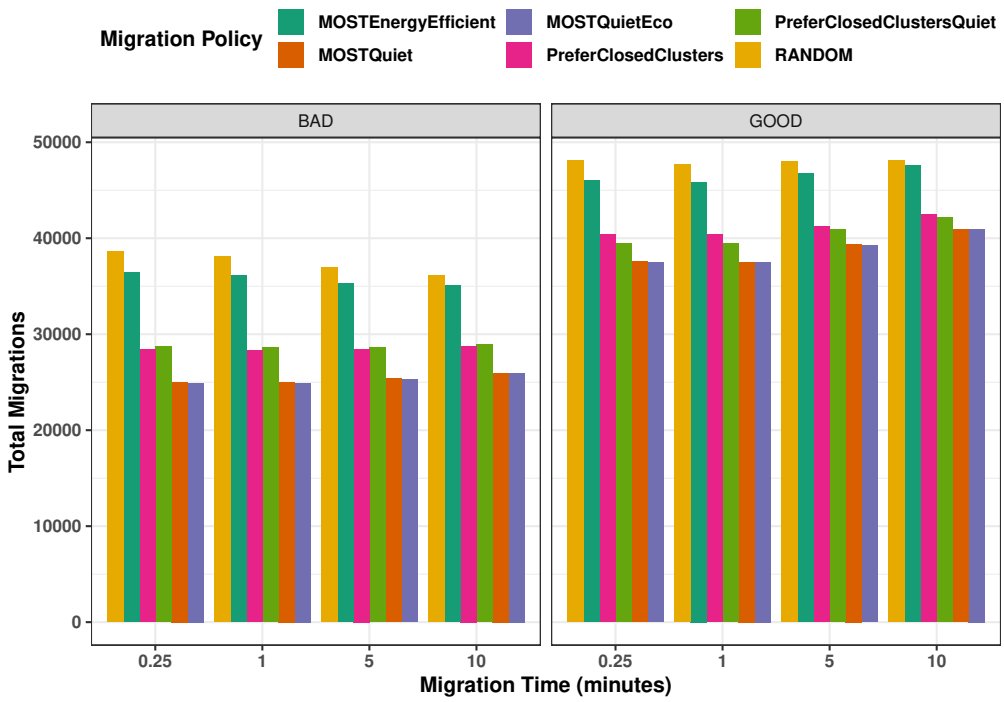

Figure 7.16: Number of bad and good migrations

Figure 7.16 exposes the number of bad migrations compares to the number of good migrations where the job is not killed after the submission. The killed jobs are associated with a large number of migrations. When the job migrations increase in the system, the energy consumption increases as well. To avoid this problem, we need to predict which kind of jobs that are most likely to be killed after submission.

7.4 Discussion

In this section, we highlight some points to the administrator of an HTC system to take them into account when they employ the responsive migration in their system as a fault-tolerance mechanism.

7.4.1 Placement

Here, the jobs are placed on a computer randomly. It would be more efficient and effective to place the job in a place that most unlikely to be interrupted by users.

Also, the administrators should consider the trade-off between energy and performance when they place a job into a computer.

In addition, we have implemented the migration policies as placement policies. The placement policies determine which computer to choose in order to allocate the job that is waiting in the queue. The initial allocation of the job is critical to gain better performance and energy efficient system.

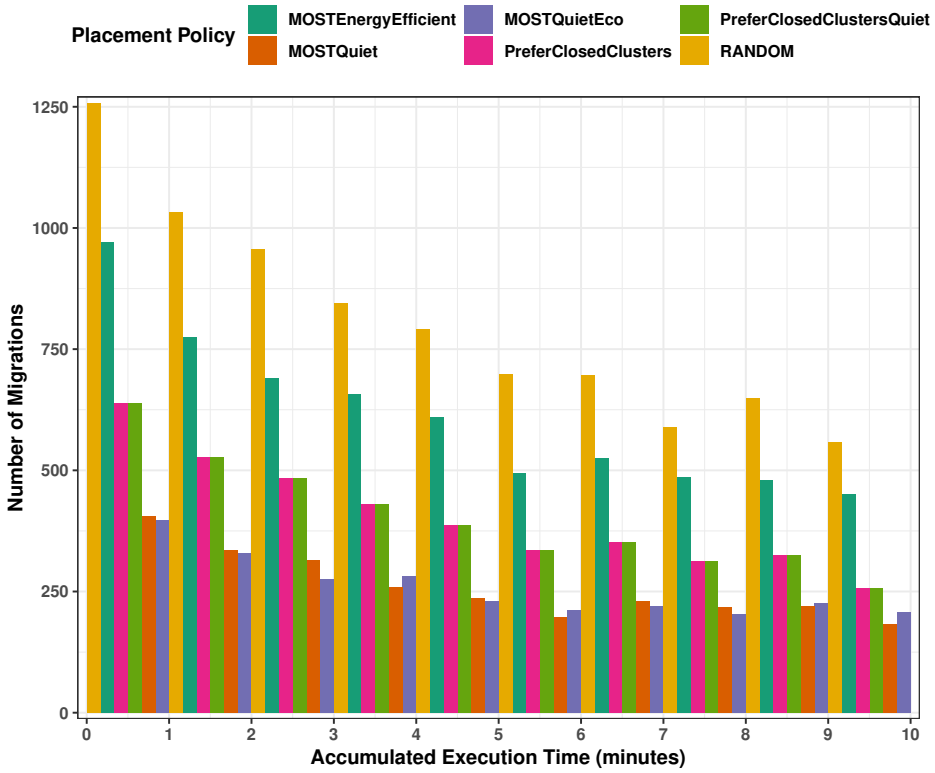

Figure 7.17: Number of migrations for jobs within 10 minutes of their accumulated execution time

Figure 7.17 shows the first 10 minutes of the accumulated execution time for all jobs within the simulation, and the first migration occurs to the jobs due to user interruptions. The number of migrations increases when the job is allocated randomly. As we explained earlier, the most energy efficient computers are allocated within busy clusters such as the library where the job is most likely to be interrupted by a user. We are still implementing and evaluating the placement policies to come up with a better saving on energy.

7.4.2 When to migrate

In this chapter, we migrate all the interrupted jobs. However, migration is most useful when a job has been executed for a long time. The job could be interrupted just a few seconds after the placement which makes the cost of job migration is higher than the cost of job eviction. Also, the administrator could take advantage of the presented approach to do a proactive fault-tolerance where jobs can be migrated before the opening time of clusters.

7.4.3 Which job to migrate

Some of the jobs can be killed at any point after the submission by its owner or the system administrator as we discussed previously. The system should avoid migrating these kinds of jobs in order to gain better performance and to save more energy.

7.4.4 Where to migrate

This is the key point that we addressed in this chapter. The selecting mechanism of the target computer for migration could impact the energy and performance of the system. It is critical to adopt the policy which fit the most under into particular HTC system. As we showed before, the Most Energy Efficient policy could save energy more than other policies, but the performance is not the best compared to other policies.

7.5 Conclusions

In this chapter, we have presented the use of live migration as a fault-tolerance mechanism in HTC systems. We have discussed the methodology of our responsive migration model. Also, we have introduced six responsive migration policies to determine the selection of the target computer when migration is needed.

We have used a trace-driven simulation to explore the impact of the policies on the performance and energy. All of these policies have the potential to enhance the system overall performance and decrease the jobs overhead and energy consumption. We demonstrated that our responsive migration could save approximately 76% of the

system wasted energy due to job evictions by user interruptions where migration is not employed as a fault-tolerance mechanism.

Moreover, our results exposed the number of job evictions due to user interruption during migration. When the time of migration increases, the number of user interruptions during migration also increases. The interrupted jobs during migration rejoin the queue for re-execution which increase the energy consumption and overhead of the system. We demonstrated that our responsive migration could reduce the number of job evictions approximately by 92% when it is used as a fault-tolerance mechanism.

Also, the chapter discussed the impact of responsive migration on the killed jobs. We showed how the system could be inefficiently utilised when we migrate jobs which are killed at later time after migration.

Finally, we provided a discussion which would assist the administrator of the HTC system when they employ the responsive migration in their system as a fault-tolerance mechanism.

8

Conclusion

Contents

Summary

This chapter provides a summary of the work which presented in this thesis. Throughout this chapter, we discuss the limitation of this work and highlight future research directions.

8.1 Thesis Summary

Chapter 2 gave the fundamental terminology and their discussion that we used in this research. Also, some related work in the literature introduced and discussed in this chapter. For instance, we mentioned the difference between our research and other research in term of migration experiment, migration performance prediction, simulation tools, and using migration as a fault-tolerance mechanism.

Chapter 3 discussed the trade-off between energy and performance of existing HTC job scheduling policies to cloud instances. We extended HTC-Sim framework in order to measure the energy consumption of each policy. Our evaluation of the policies was under a number of assumptions which made about the hardware used within a cloud datacentre. The hardware energy and performance specifications were taken from the published results in [1] which obtained by using the SPECPower2008 benchmark. All policies showed different impacts on energy consumption and average overheads. Moreover, the simulation tool which used in this chapter is extended further in Chapter 6 and Chapter 7 to evaluate live migration methods and policies.

Chapter 4 presented a real experiment to measure the time of VM live migration. We developed an automated script to trigger the live migration process between two physical machines. In order to create the VMs, we used KVM as a hypervisor. Also, we installed the SPECjvm2008 benchmark in each physical machine to generate the workloads. The results of the experiment showed an important link between the VM memory size and the migration time. The time of the VM migration increases as the memory size of the VM increase. In this chapter, we considered a three type of VMs, and the speed of the network was 100Mb. In Chapter 5, we extend this experiment to support more type of VMs as well as higher network speed.

Chapter 5 exhibited the extension of the live migration experiment which mentioned in Chapter 4 where we used a 1000Mb switch and nine different type of VMs. Furthermore, the chapter demonstrated the process of training and testing three machine learning models based on the results from the live migration experiment. The models used to predict the successful VM migrations where the VMs takes a short time to be migrated between two physical machines. Our models showed a high accuracy when we compared with another seven models. We adopted the findings of this chapter to support the assumption that made to extend the HTC-Sim framework as discussed in Chapter 6.

Chapter 6 described our extension to HTC-Sim which includes the virtualisation and live migration mechanisms. In this chapter, we proposed two live migration methods which can be used in HTC environment in order to improve the performance and save energy, namely, responsive migration and interval migration. The simulation is based on real data which collected from the Newcastle University HTCondor system during 2010. Moreover, our simulation can provide the overall migration time, the overall energy consumption and waste, and the number of failed migrations due to interactive users interruptions or removed jobs. Also, the simulation can count the number of migration attempts when there is no host for migration. Our results showed that the responsive migration method could save up to 75% of the system wasted energy. In the next chapter, we introduced six migration policies which could optimise the performance of the responsive migration.

Chapter 7 demonstrated how selecting the target computer for migration can influence the performance and energy of the system. We have proposed six responsive migration policies to determine the selection of the target computer when migration is needed. Also, this chapter focused on using live migration as a fault-tolerance mechanism where the jobs get migrated to another physical machine in the event of user login. We extended the HTC-Sim to implement and evaluate the policies. The results showed that our responsive migration method could reduce the number of job evictions approximately by 92% when it is used as a fault-tolerance mechanism. Furthermore, we provided some advice to the administrator of HTC systems which would assist them when they use the live migration as a fault-tolerance mechanism.

8.2 Limitations

In this thesis, we presented real experiments to measure the migration time of various VM types between two different physical machines as well as two different network speed. However, the VMs were created and migrated by using one hypervisor solution. The experimental results might vary if the hypervisor solution is replaced with another one in the experiment.

Also, the prediction models of this thesis were built from diverse datasets which involved 20 type of workloads. However, the workloads belong to one benchmark, and the dataset which used to train and test the models are considered to be small.

Furthermore, we used a trace-driven simulation where the trace logs are obtained from a real environment. However, the trace logs do not include all the features which we used to create the predictive models. As a result, the models are not implemented in the simulation to predict the time of live migration. Instead, the time of migration in the simulation is assumed based on our observations of the VM live migration experiments and models. Moreover, the generalisability of the simulation results is another limitation of this thesis which needs to be addressed in the future by applying the migration methods and policies to another dataset.

The comparison between the real world and the simulation as well as the models are missing in this thesis. Through this thesis, we used real data to build the models and the simulation, but the gained results have been not compared with the real world outcomes which we consider it as a limitation of this work.

8.3 Future Research Directions

In this section, we discuss some directions for future research, arising from lessons learnt during the PhD.

8.3.1 VM live migration experiment

In Chapter 4 and 5, we presented a real experiment to measure the time of live migration between two different physical machines. The experiment could be extended

to measure the downtime of the VM as well as the energy consumption of the migration process in the source and target computers. Also, the experiment involves one hypervisor solution which is KVM. In the futures, the experiment can be extended to understand the influence of other hypervisor solutions such as Xen, VMware, and Hyper-V on the migration process. Also, we used two bandwidth capacities which are widely used in HTC environment. In the future, the experiment can involve a much higher bandwidth capacity that is used in most large-scale datacentres.

Furthermore, our results of the experiment are based on one benchmark which provides various workload types. The experiment can be extended to support different benchmarks which will provide a better understanding of the workload impacts on the migration process.

Finally, the traffic of the network might influence the migration time of a VM. To understand the impact of network traffic on the time of migration, the experiment should be performed under different network traffic status.

8.3.2 Prediction Models

In this thesis, we used classification prediction models to predict successful VM live migrations where the VMs take a short time to be migrated. The models could be changed to be regression models where the time of migration can be estimated rather than categorisation into groups.

Furthermore, when the migration experiment is extended according to the mentioned suggestions above, the number of prediction model features increase which could enhance the generalisation and interpretability of the models. Also, the prediction could be extended to include migration cost, migration energy consumption, and VM downtime.

8.3.3 Simulation

The simulation is used to evaluate HTC jobs policies, and it is based on real data which collected from Newcastle University. In order to make the simulation more generic, the trace logs need to be collected from other HTC systems such as HTCondor at the

University of Wisconsin-Madison. Also, the trace logs should include all information that needs to be used by the prediction models.

Also, the simulation can be easily extended to support other environments such as cloud. In the future, we will provide a copy of the simulation that supports a cloud environment. In order to do that, the simulation should be able to assign multiple jobs on one source machine. We could achieve this by implanting dynamic VM placement and migration into HTC-Sim. Also, we need to apply VM consolidation strategies which enhance the performance and save energy.

8.3.4 Fault-tolerance methods

In this work, we provide a method and policies for fault-tolerance. The method avoids the job failures due to the user interruptions. Our method is a reactive fault-tolerance which takes place at the event of failure. In the future, the proactive fault-tolerance mechanism could be designed and implemented in order to enhance the performance and prevent unnecessary migrations.

Furthermore, the policies could be designed and implemented to support the proactive fault-tolerance mechanism. For example, when the jobs are migrated from clusters before the opening times to closed clusters, the jobs will have a low chance to be interrupted during their execution.

Bibliography

[1] All published SPECpower_ssj2008 results. https://www.spec.org/power_ssj2008/results/power_ssj2008.html. Accessed: 2018-12-06.

[2] Cycle computing (homepage). http://cyclecomputing.com. Accessed: 2017-05-05.

[3] ECONET. https://www.econet-project.eu Acessed: 2018-12-10.

[4] KVM. http://www.linux-kvm.org. Accessed: 2018-12-06.

[5] Memusg. https://gist.github.com/netj/526585. Accessed: 2017-05-05.

[6] Openfiler. http://www.openfiler.com. Accessed: 2017-05-05.

[7] Sar. http://www.linuxcommand.org/man_pages/sar1.html. Accessed: 2017-05-05.

[8] SERT Suite. https://www.spec.org/sert. Accessed: 2018-12-10.

[9] SPEC VIRT_SC 2013. https://www.spec.org/virt_sc2013. Accessed: 2018-12-06.

[10] SPECjvm2008. http://www.spec.org/jvm2008. Accessed: 2017-10-05.

[11] SPECpower_ssj2008. http://www.spec.org/power_ssj2008. Accessed: 2018-12-06.

[12] SWaP (Space, Watts and Performance) Metric. http://www.sun.com/servers/coolthreads/swap.

[13] The CentOS Project. http://www.centos.org. Accessed: 2017-05-05.

[14] Ubuntu for desktops. http://www.ubuntu.com. Accessed: 2017-05-05.

[15] UK Research Council End Use Energy Demand (EUED) Centres. http://www.eued.ac.uk. Accessed: 2017-05-05.

[16] Unix top. http://www.unixtop.org. Accessed: 2017-05-05.

[17] A framework for data center energy productivity, 2008. https://www.greenbiz.com/sites/default/files/document/GreenGrid-Framework-Data-Center-Energy-Productivity.pdf. Accessed: 2018-12-06.

[18] Data center networking equipment - issues and best practices. Technical report, ASHRAE Technical Committee, 2012. https://goo.gl/RJfvtS.

[19] Guidelines and best practices for the installation and maintenance of data networking equipment. Technical report, Cisco Systems, Inc., 2013. https://www.cisco.com/c/dam/en_us/training-events/downloads/guidelines_and_best_practices_for_the_Installation_and_maintenance_of_data_networking_equipment.pdf. Accessed: 2018-12-06.

[20] Advanced configuration and power interface specification. Unified EFI Forum, Inc., 2017. http://www.uefi.org/sites/default/files/resources/ACPI_6_2.pdf. Accessed: 2018-12-06.

[21] Energy efficient computing, clusters, grids and clouds: A taxonomy and survey. *Sustainable Computing: Informatics and Systems*, 14:13 – 33, 2017.

[22] F. Abaunza, A.-P. Hameri, and T. Niemi. Eeui: a new measure to monitor and manage energy efficiency in data centers. *International Journal of Productivity and Performance Management*, 67(1):111–127, 2018.

[23] D. Abts, M. R. Marty, P. M. Wells, P. Klausler, and H. Liu. Energy proportional datacenter networks. *SIGARCH Computer Architecture News*, 38(3):338–347, 2010.

[24] R. W. Ahmad, A. Gani, S. H. Ab. Hamid, M. Shiraz, F. Xia, and S. A. Madani. Virtual machine migration in cloud data centers: A review, taxonomy, and open research issues. *Journal of Supercomputer*, 71(7):2473–2515, 2015.

[25] S. Akoush, R. Sohan, A. Rice, A. W. Moore, and A. Hopper. Predicting the performance of virtual machine migration. In *IEEE MASCOTS*, pages 37–46, 2010.

[26] A. Aldhalaan and D. A. Menascé. Analytic performance modeling and optimization of live vm migration. In M. S. Balsamo, W. J. Knottenbelt, and A. Marin, editors, *Computer Performance Engineering*, pages 28–42, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[27] S. A. Ali, M. Affan, and M. Alam. A study of efficient energy management techniques for cloud computing environment. *arXiv preprint arXiv:1810.07458*, 2018.

[28] O. Alrajeh. VM live migration script. http://www.github.com/oalrajeh/VM_Live_Migration. Accessed: 2018-12-06.

[29] O. Alrajeh, M. Forshaw, A. S. McGough, and N. Thomas. Simulation of virtual machine live migration in high throughput computing environments. IEEE, 2018.

[30] O. Alrajeh, M. Forshaw, and N. Thomas. Machine learning models for predicting timely virtual machine live migration. In P. Reinecke and A. Di Marco, editors, *Computer Performance Engineering*, pages 169–183, Cham, 2017. Springer International Publishing.

[31] O. N. Alrajeh and N. Thomas. Energy consumption of scheduling policies for htc jobs in the cloud. In *Proceedings of the 8th International Conference on Simulation Tools and Techniques*, SIMUTools '15, pages 343–348. ACM, 2015.

[32] F. M. Alrajeh O. and T. N. Performance of virtual machine live migration with various workloads. In *32nd UK Performance Engineering Workshop*. University of Bradford, 2016.

[33] M. Amoon. Adaptive framework for reliable cloud computing environment. *IEEE Access*, 4:9469–9478, 2016.

[34] D. P. Anderson. BOINC: a system for public-resource computing and storage. In *Fifth IEEE/ACM International Workshop on Grid Computing*, pages 4–10, 2004.

[35] A. Ashraf and I. Porres. Multi-objective dynamic virtual machine consolidation in the cloud using ant colony system. *International Journal of Parallel, Emergent and Distributed Systems*, 33(1):103–120, 2018.

[36] G. Aupy, A. Benoit, R. Melhem, P. Renaud-Goud, and Y. Robert. Energy-aware checkpointing of divisible tasks with soft or hard deadlines. In *2013 International Green Computing Conference Proceedings*, pages 1–8, 2013.

[37] R. Ayoub, R. Nath, and T. Rosing. Jetc: Joint energy thermal and cooling management for memory and CPU subsystems in servers. In *IEEE International Symposium on High-Performance Comp Architecture*.

[38] D. Azevedo, M. Patterson, J. Pouchet, and R. Tipley. Carbon usage effectiveness (CUE): a green grid data center sustainability metric, 2010. https://goo.gl/jaoagW.

[39] L. A. Barroso, J. Clidaras, and U. Hölzle. The datacenter as a computer: An introduction to the design of warehouse-scale machines. *Synthesis lectures on computer architecture*, 8(3):1–154, 2013.

[40] C. Belady, A. Rawson, J. Pfleuger, and T. Cader. Green grid data center power efficiency metrics: PUE and DCIE. Technical report, Green Grid, 2008.

[41] W. H. Bell, D. G. Cameron, A. P. Millar, L. Capozza, K. Stockinger, and F. Zini. Optorsim: A grid simulator for studying dynamic data replication strategies. *The International Journal of High Performance Computing Applications*, 17(4):403–416, 2003.

[42] A. Beloglazov, J. Abawajy, and R. Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems*, 28(5):755–768, 2012.

[43] A. Beloglazov and R. Buyya. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers. *Concurrency and Computation: Practice and Experience*, 24(13):1397–1420, 2012.

[44] A. Beloglazov, R. Buyya, Y. C. Lee, and A. Y. Zomaya. A taxonomy and survey of energy-efficient data centers and cloud computing systems. *CoRR*, abs/1007.0066, 2010.

[45] B. Bermejo, C. Juiz, and C. Guerrero. Virtualization and consolidation: a systematic review of the past 10 years of research on energy and performance. *Journal of Supercomputing*, pages 1–29, 2018.

[46] P. Bezerra, G. Martins, R. Gomes, F. Cavalcante, and A. Costa. Evaluating live virtual machine migration overhead on client's application perspective. In *2017 International Conference on Information Networking (ICOIN)*, pages 503–508, 2017.

[47] K. Bilal, S. U. Khan, S. A. Madani, K. Hayat, M. I. Khan, N. Min-Allah, J. Kolodziej, L. Wang, S. Zeadally, and D. Chen. A survey on green communications using adaptive link rate. *Cluster Computing*, 16(3):575–589, 2013.

[48] G. Bisson and F. Hussain. Chi-sim: A new similarity measure for the co-clustering task. In *2008 Seventh International Conference on Machine Learning and Applications*, pages 211–217, 2008.

[49] T. Bloch, R. Sridaran, and C. Prashanth. Understanding live migration techniques intended for resource interference minimization in virtualized cloud environment. In V. B. Aggarwal, V. Bhatnagar, and D. K. Mishra, editors, *Big Data Analytics*, pages 487–497, Singapore, 2018. Springer Singapore.

[50] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, Aug 1996.

[51] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001.

[52] R. E. Brown, E. R. Masanet, B. Nordman, W. F. Tschudi, A. Shehabi, J. Stanley, J. G. Koomey, D. A. Sartor, and P. T. Chan. Report to congress on server and data center energy efficiency: Public law 109-431. 2008. https://escholarship.org/uc/item/74g2r0vg.

[53] R. Buyya and M. Murshed. 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, 2002.

[54] J. Byrne, S. Svorobej, K. M. Giannoutakis, D. Tzovaras, P. J. Byrne, P.-O. Östberg, A. Gourinovitch, and T. Lynn. A review of cloud computing simulation platforms and related environments. 2017.

[55] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya. Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41(1):23–50, 2011.

[56] A. Capozzoli, M. Chinnici, M. Perino, and G. Serale. Review on performance metrics for energy efficiency in data center: The role of thermal management. In S. Klingert, M. Chinnici, and M. Rey Porto, editors, *Energy Efficient Data Centers*, pages 135–151, Cham, 2015. Springer International Publishing.

[57] P. H. Castro, V. L. Barreto, S. L. Correa, L. Z. Granville, and K. V. Cardoso. A joint CPU-RAM energy efficient and SLA-compliant approach for cloud data centers. *Computer Networks*, 94:1–13, 2016.

[58] Y. Chen, C. Lin, J. Huang, X. Xiang, and X. Shen. Energy efficient scheduling and management for large-scale services computing systems. *IEEE Transactions on Services Computing*, 10(2):217–230, 2017.

[59] A. Choudhary, M. C. Govil, G. Singh, L. K. Awasthi, E. S. Pilli, and D. Kapil. A critical survey of live virtual machine migration techniques. *Journal of Cloud Computing*, 6(1):23, 2017.

[60] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *Proceedings of the 2Nd Conference on Symposium on Networked Systems Design & Implementation - Volume 2*, NSDI'05, pages 273–286, Berkeley, CA, USA, 2005. USENIX Association.

[61] J. Daniels. Server virtualization architecture and implementation. *Crossroads*, 16(1):8–12, 2009. ACM.

[62] W. Dargie. Estimation of the cost of vm migration. In *23rd International Conference on Computer Communication and Networks (ICCCN)*, pages 1–8. IEEE, 2014.

[63] S. Dawson-Haggerty, A. Krioukov, and D. E. Culler. Power optimization-a reality check. *EECS Department, University of California, Berkeley, Technical Report UCB/EECS-2009-140*, 2009.

[64] C. De Alfonso, M. Caballer, F. Alvarruiz, and G. Moltó. An economic and energy-aware analysis of the viability of outsourcing cluster computing to a cloud. *Future Generation Computing Systems*, 29(3):704–712, 2013.

[65] M. D. de Assuncao, A. di Costanzo, and R. Buyya. Evaluating the cost-benefit of using cloud computing to extend the capacity of clusters. In *Proceedings of the 18th ACM International Symposium on High Performance Distributed Computing*, HPDC '09, pages 141–150, New York, NY, USA, 2009. ACM.

[66] E. Deelman, G. Singh, M. Livny, B. Berriman, and J. Good. The cost of doing science on the cloud: The montage example. In *Proceedings of the 2008 ACM/IEEE Conference on Supercomputing*, SC '08, pages 50:1–50:12, Piscataway, NJ, USA, 2008. IEEE Press.

[67] P. Delforge. America's data centers are wasting huge amounts of energy. *Natural Resources Defense Council (NRDC)*, pages 1–5, 2014.

[68] L. Deng, H. Jin, H. Chen, and S. Wu. Migration cost aware mitigating hot nodes in the cloud. In *2013 International Conference on Cloud Computing and Big Data*, pages 197–204, 2013.

[69] I. S. Dhanoa and S. S. Khurmi. Analyzing energy consumption during vm live migration. In *International Conference on Computing, Communication Automation*, pages 584–588, 2015.

[70] D. Didona, F. Quaglia, P. Romano, and E. Torre. Enhancing performance prediction robustness by combining analytical modeling and machine learning. In *Proceedings of the 6th ACM/SPEC International Conference on Performance Engineering*, ICPE '15, pages 145–156, New York, NY, USA, 2015. ACM.

[71] T. V. T. Duy, Y. Sato, and Y. Inoguchi. Performance evaluation of a green scheduling algorithm for energy savings in cloud computing. In *2010 IEEE international symposium on parallel & distributed processing, workshops and Phd forum (IPDPSW)*, pages 1–8. IEEE, 2010.

[72] C. Engelmann, G. R. Vallee, T. Naughton, and S. L. Scott. Proactive fault tolerance using preemptive migration. In *17th Euromicro International Conference on Parallel, Distributed and Network-based Processing*, pages 252–257, 2009.

[73] X. Fan, W.-D. Weber, and L. A. Barroso. Power provisioning for a warehouse-sized computer. In *ACM SIGARCH computer architecture news*, volume 35, pages 13–23. ACM, 2007.

[74] E. Feller, L. Rilling, and C. Morin. Energy-aware ant colony based workload placement in clouds. In *Proceedings of the 2011 IEEE/ACM 12th International Conference on Grid Computing*, pages 26–33. IEEE Computer Society, 2011.

[75] W. Feller. *An introduction to probability theory and its applications*, volume 2. Wiley, New York, 1971.

[76] T. C. Ferreto, M. A. Netto, R. N. Calheiros, and C. A. D. Rose. Server consolidation with migration control for virtualized data centers. *Future Generation Computer Systems*, 27(8):1027 – 1034, 2011.

[77] C. K. Filelis-Papadopoulos, K. M. Giannoutakis, G. A. Gravvanis, C. S. Kouzinopoulos, A. T. Makaratzis, and D. Tzovaras. *Simulating Heterogeneous Clouds at Scale*, pages 119–150. Springer International Publishing, Cham, 2018.

[78] M. Forshaw, A. McGough, and N. Thomas. HTC-Sim: A trace-driven simulation framework for energy consumption in high-throughput computing systems. *Concurrency and Computation: Practice and Experience*, 28(12):3260–3290, 2016.

[79] M. Forshaw, A. S. McGough, and N. Thomas. Energy-efficient checkpointing in high-throughput cycle-stealing distributed systems. *Electronic Notes in Theoretical Computer Science*, 310(C):65–90, 2015.

[80] J. H. Friedman. Stochastic gradient boosting. *Computational Statistics Data Analysis*, 38(4):367–378, 2002.

[81] J. Gao. Machine learning applications for data center optimization, 2014. https://ai.google/research/pubs/pub42542.

[82] S. K. Garg, C. S. Yeo, A. Anandasivam, and R. Buyya. Environment-conscious scheduling of HPC applications on distributed cloud-oriented data centers. *Journal of Parallel and Distributed Computing*, 71(6):732–749, 2011.

[83] W. Gentzsch. Sun grid engine: towards creating a compute power grid. In *Proceedings First IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 35–36, 2001.

[84] L. Gillam, B. Li, J. O'Loughlin, and A. P. S. Tomar. Fair benchmarking for cloud computing systems. *Journal of Cloud Computing: Advances, Systems and Applications*, 2(1):6, 2013.

[85] F. Giroire, D. Mazauric, J. Moulierac, and B. Onfroy. Minimizing routing energy consumption: From theoretical to practical results. In *2010 IEEE/ACM Int'l Conference on Green Computing and Communications Int'l Conference on Cyber, Physical and Social Computing*, pages 252–259, 2010.

[86] J. A. Hanley and B. J. McNeil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143(1):29–36, 1982.

[87] M. K. Herrlin. Rack cooling effectiveness in data centers and telecom central offices: The rack cooling index (RCI). *Transactions-American Society of Heating Refrigerating and Air conditioning Engineers*, 111(2):725, 2005.

[88] M. R. Hines, U. Deshpande, and K. Gopalan. Post-copy live migration of virtual machines. *SIGOPS Oper. Syst. Rev.*, 43(3):14–26, 2009.

[89] M. R. Hines and K. Gopalan. Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning. In *Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, VEE '09, pages 51–60, New York, NY, USA, 2009. ACM.

[90] W. Hu, A. Hicks, L. Zhang, E. M. Dow, V. Soni, H. Jiang, R. Bull, and J. N. Matthews. A quantitative study of virtual machine live migration. In *Proceedings of the 2013 ACM cloud and autonomic computing conference*, page 11. ACM, 2013.

[91] C. Jo, Y. Cho, and B. Egger. A machine learning approach to live migration modeling. In *Proceedings of the 2017 Symposium on Cloud Computing*, SoCC '17, pages 351–364, New York, NY, USA, 2017. ACM.

[92] S. Jobling. Carbon management plan, 2016. https://www.ncl.ac.uk/sustainable-campus/assets/documents/NewcastleUniversityCMP_2016_V1.pdf.

[93] T. P. Joe Olivas, Mike Chynoweth. Benefitting power and performance sleep loops. https://software.intel.com/en-us/articles/benefitting-power-and-performance-sleep-loops, 2015.

[94] F. Juarez, J. Ejarque, and R. M. Badia. Dynamic energy-aware scheduling for parallel task-based application in cloud computing. *Future Generation Computer Systems*, 78:257 – 271, 2018.

[95] N. J. Kansal and I. Chana. Energy-aware virtual machine migration for cloud computing - a firefly optimization approach. *Journal of Grid Computing*, 14(2):327–345, 2016.

[96] A. Khosravi and R. Buyya. Energy and carbon footprint-aware management of geo-distributed cloud data centers: A taxonomy, state of the art, and future directions. In *Sustainable Development: Concepts, Methodologies, Tools, and Applications*, pages 1456–1475. IGI Global, 2018.

[97] A. Kipp, T. Jiang, M. Fugini, and I. Salomie. Layered green performance indicators. *Future Generation Computer Systems*, 28(2):478 – 489, 2012.

[98] D. Kliazovich, P. Bouvry, and S. U. Khan. GreenCloud: a packet-level simulator of energy-aware cloud computing data centers. *The Journal of Supercomputing*, 62(3):1263–1283, 2012.

[99] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, pages 1137–1143, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[100] R. Koller, A. Verma, and A. Neogi. WattApp: An application aware power meter for shared data centers. In *Proceedings of the 7th International Conference on Autonomic Computing*, ICAC '10, pages 31–40, New York, NY, USA, 2010. ACM.

[101] M. Kuhn. Building predictive models in R using the caret package. *Journal of Statistical Software*, 28(1):1–26, 2008.

[102] M. Kuhn. Caret package. *Journal of Statistical Software*, 28(5):1–26, 2008.

[103] K. Kumar. Workshop on power aware computing and systems (HotPower'08). 2008. https://www.usenix.org/legacy/publications/login/2009-04/openpdfs/hotpower08.pdf.

[104] Z. Lai, K. T. Lam, C.-L. Wang, and J. Su. Latency-aware DVFS for efficient power state transitions on many-core architectures. *The Journal of Supercomputing*, 71(7):2720–2747, 2015.

[105] A. Legrand, L. Marchal, and H. Casanova. Scheduling distributed applications: the simgrid simulation framework. In *CCGrid 2003. 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid, 2003. Proceedings.*, pages 138–145, May 2003.

[106] J. Li, J. Zhao, Y. Li, L. Cui, B. Li, L. Liu, and J. Panneerselvam. iMIG: Toward an adaptive live migration method for KVM virtual machines. *The Computer Journal*, 58(6):1227–1242, 2015.

[107] X. Li, Z. Qian, S. Lu, and J. Wu. Energy efficient virtual machine placement algorithm with balanced and improved resource utilization in a data center. *Mathematical and Computer Modelling*, 58(5):1222 – 1235, 2013.

[108] S.-H. Lim, B. Sharma, G. Nam, E. K. Kim, and C. R. Das. MDCSim: A multi-tier data center simulation, platform. In *2009 IEEE International Conference on Cluster Computing and Workshops*, pages 1–9, Aug 2009.

[109] C. X. Ling, J. Huang, and H. Zhang. *AUC: A Better Measure than Accuracy in Comparing Learning Algorithms*, pages 329–341. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[110] M. J. Litzkow, M. Livny, and M. W. Mutka. Condor-a hunter of idle workstations. In *Distributed Computing Systems, 1988., 8th International Conference on*, pages 104–111. IEEE, 1988.

[111] C. Liu, X. Qin, S. Kulkarni, C. Wang, S. Li, A. Manzanares, and S. Baskiyar. Distributed energy-efficient scheduling for data-intensive applications with deadline constraints on data grids. In *2008 IEEE International Performance, Computing and Communications Conference*, pages 26–33, 2008.

[112] H. Liu and B. He. Vmbuddies: Coordinating live migration of multi-tier applications in cloud environments. *IEEE Transactions on Parallel and Distributed Systems*, 26(4):1192–1205, 2015.

[113] H. Liu, C.-Z. Xu, H. Jin, J. Gong, and X. Liao. Performance and energy modeling for live migration of virtual machines. In *Proceedings of the 20th International Symposium on High Performance Distributed Computing*, HPDC '11, pages 171–182, New York, NY, USA, 2011. ACM.

[114] J. Liu, S. Wang, A. Zhou, S. Kumar, F. Yang, and R. Buyya. Using proactive fault-tolerance approach to enhance cloud service reliability. *IEEE Transactions on Cloud Computing*, 2016.

[115] A. T. Makaratzis, K. M. Giannoutakis, and D. Tzovaras. Energy modeling in cloud simulation frameworks. *Future Generation Computer Systems*, 79:715 – 725, 2018.

[116] T. Mastelic, A. Oleksiak, H. Claussen, I. Brandic, J.-M. Pierson, and A. V. Vasilakos. Cloud computing: Survey on energy efficiency. *ACM Computing Surveys*, 47(2):33:1–33:36, 2014.

[117] M. Mattess, C. Vecchiola, and R. Buyya. Managing peak loads by leasing cloud infrastructure services from a spot market. In *Proceedings of the 2010 IEEE 12th International Conference on High Performance Computing and Communications*, HPCC '10, pages 180–188, Washington, DC, USA, 2010. IEEE Computer Society.

[118] A. S. McGough, N. Al Moubayed, and M. Forshaw. Using machine learning in trace-driven energy-aware simulations of high-throughput computing systems. In *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion*, pages 55–60. ACM, 2017.

[119] A. S. McGough, M. Forshaw, J. Brennan, N. A. Moubayed, and S. Bonner. Using machine learning to reduce the energy wasted in volunteer computing environments. *arXiv preprint arXiv:1810.08675*, 2018.

[120] A. S. McGough, M. Forshaw, C. Gerrard, P. Robinson, and S. Wheater. Analysis of power-saving techniques over a large multi-use cluster with variable workload. *Concurrency and Computation: Practice and Experience*, 25(18):2501–2522, 2013.

[121] A. S. McGough, M. Forshaw, C. Gerrard, S. Wheater, B. Allen, and P. Robinson. Comparison of a cost-effective virtual cloud cluster with an existing campus cluster. *Future Generation Computer Systems*, 41:65–78, 2014.

[122] D. Meisner, B. T. Gold, and T. F. Wenisch. Powernap: Eliminating server idle power. *SIGARCH Comput. Archit. News*, 37(1):205–216, 2009.

[123] V. Méndez and F. García. SiCoGrid: A complete grid simulator for scheduling and algorithmical research, with emergent artificial intelligence data algorithms. Technical Report RR-06-11. DIIS. UNIZAR. 2005.

[124] S. Mingay. Green IT: the new industry shock wave. *Gartner RAS Research Note G*, 153703(7), 2007.

[125] S. Mittal. Power management techniques for data centers: A survey. *arXiv preprint arXiv:1404.6681*, 2014.

[126] F. Moisan and D. Bosseboeuf. Energy efficiency: A recipe for success. *World Energy Council, London, UK, Tech. Rep*, 2010.

[127] A. B. Nagarajan, F. Mueller, C. Engelmann, and S. L. Scott. Proactive fault tolerance for HPC with Xen virtualization. In *Proceedings of the 21st annual international conference on Supercomputing*, pages 23–32. ACM, 2007.

[128] S. Nathan, U. Bellur, and P. Kulkarni. Towards a comprehensive performance model of virtual machine live migration. In *Proceedings of the Sixth ACM Symposium on Cloud Computing*, SoCC '15, pages 288–301, New York, NY, USA, 2015. ACM.

[129] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy, and D. Wetherall. Reducing network energy consumption via sleeping and rate-adaptation. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, NSDI'08, pages 323–336, Berkeley, CA, USA, 2008. USENIX Association.

[130] R. Neugebauer and D. Mcauley. Energy is just another resource: energy accounting and energy pricing in the nemesis os. In *Proceedings Eighth Workshop on Hot Topics in Operating Systems*, pages 67–72. IEEE, 2001.

[131] T. H. Nguyen, M. Di Francesco, and A. Yla-Jaaski. Virtual machine consolidation with multiple usage prediction for energy-efficient cloud data centers. *IEEE Transactions on Services Computing*, 2017.

[132] A. Núñez, J. L. Vázquez-Poletti, A. C. Caminero, G. G. Castañé, J. Carretero, and I. M. Llorente. iCanCloud: A flexible and scalable cloud infrastructure simulator. *Journal of Grid Computing*, 10(1):185–209, 2012.

[133] D. Nurmi, J. Brevik, and R. Wolski. Minimizing the network overhead of checkpointing in cycle-harvesting cluster environments. In *2005 IEEE International Conference on Cluster Computing*, pages 1–10, 2005.

[134] A.-C. Orgerie, M. D. d. Assuncao, and L. Lefevre. A survey on techniques for improving the energy efficiency of large-scale distributed systems. *ACM Computing Surveys*, 46(4):47:1–47:31, 2014.

[135] M. Patel, S. Chaudhary, and S. Garg. Machine learning based statistical prediction model for improving performance of live virtual machine migration. *Journal of Engineering*, 2016, 2016.

[136] I. R. Philp. Software Failures and the Road to a Petaflop Machine. In *Proceedings of the 1st Workshop on High Performance Computing Reliability Issues (HPCRI) 2005, in conjunction with the 11th International Symposium on High Performance Computer Architecture (HPCA) 2005*, San Francisco, CA, USA, 2005. IEEE Computer Society.

[137] A. Polze, P. Troger, and F. Salfner. Timely virtual machine migration for pro-active fault tolerance. In *2011 14th IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops*, pages 234–243, 2011.

[138] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2016.

[139] S. Radvan, D. Parker, C. Curran, and J. Mark. Red Hat Enterprise Linux 5 Virtualization Guide. https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/5/pdf/virtualization/Red_Hat_Enterprise_Linux-5-Virtualization-en-US.pdf, 2014.

[140] X. Ren, R. Eigenmann, and S. Bagchi. Failure-aware checkpointing in fine-grained cycle sharing systems. In *Proceedings of the 16th International Symposium on High Performance Distributed Computing*, HPDC '07, pages 33–42, New York, NY, USA, 2007. ACM.

[141] G. Ridgeway. Generalized boosted models: A guide to the gbm package. *Update*, 1(1):2007, 2007.

[142] K. Rybina, A. Patni, and A. Schill. Analysing the migration time of live migration of multiple virtual machines. In *Proceedings of the 4th International Conference on Cloud Computing and Services Science*, CLOSER 2014, pages 590–597, Portugal, 2014. SCITEPRESS - Science and Technology Publications, Lda.

[143] F. Salfner, P. Tröger, and A. Polze. Downtime analysis of virtual machine live migration. In *The Fourth International Conference on Dependability (DEPEND 2011). IARIA*, pages 100–105, 2011.

[144] F. Salfner, P. Tröger, and M. Richly. Dependable estimation of downtime for virtual machine live migration. *International Journal On Advances in Systems and Measurements*, 5(1), 2012.

[145] B. Schödwell, K. Erek, and R. Zarnekow. Data center green performance measurement: State of the art and open research challenges. 2013.

[146] B. Schödwell, M. Wilkens, K. Erek, and R. Zarnekow. Towards a holistic multi-level green performance indicator framework (GPIF) to improve the energy efficiency of data center operation - a resource usage-based approach. In *Electronics Goes Green 2012+*, pages 1–6, 2012.

[147] M. Sharma and P. Sharma. Performance evaluation of adaptive virtual machine load balancing algorithm. *IJACSA) International Journal of Advanced Computer Science and Applications*, 3(2), 2012.

[148] A. Shehabi, S. Smith, D. Sartor, R. Brown, M. Herrlin, J. Koomey, E. Masanet, N. Horner, I. Azevedo, and W. Lintner. United states data center energy usage report. 2016.

[149] K. Shiv, K. Chow, Y. Wang, and D. Petrochenko. *SPECjvm2008 Performance Characterization*, pages 17–35. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[150] J. Shuja, S. A. Madani, K. Bilal, K. Hayat, S. U. Khan, and S. Sarwar. Energy-efficient data centers. *Computing*, 94(12):973–994, 2012.

[151] A. Strunk and W. Dargie. Does live migration of virtual machines cost energy? In *27th International Conference on Advanced Information Networking and Applications (AINA)*, pages 514–521. IEEE, 2013.

[152] H. Team. HTCondor version 8.7.7 manual. Technical report, Center for High Throughput Computing, University of Wisconsin-Madison, 2018.

[153] F. Teng, L. Yu, T. Li, D. Deng, and F. Magoulès. Energy efficiency of VM consolidation in IaaS clouds. *The Journal of Supercomputing*, 73(2):782–809, 2017.

[154] R. Tu, X. Wang, and Y. Yang. Energy-saving model for SDN data centers. *The Journal of Supercomputing*, 70(3):1477–1495, 2014.

[155] M. Uddin, A. Shah, R. Alsaqour, and J. Memon. Measuring efficiency of tier level data centers to implement green energy efficient data centers. *Middle-East Journal of Scientific Research*, 15(2):200–207, 2013.

[156] R. B. Uriarte, F. Tiezzi, and S. A. Tsaftaris. Supporting autonomic management of clouds: Service clustering with random forest. *IEEE Transactions on Network and Service Management*, 13(3):595–607, 2016.

[157] R. Van den Bossche, K. Vanmechelen, and J. Broeckhove. Cost-optimal scheduling in hybrid iaas clouds for deadline constrained workloads. In *2010 IEEE 3rd International Conference on Cloud Computing*, pages 228–235, 2010.

[158] J. von Kistowski, J. Grohmann, N. Schmitt, and S. Kounev. Predicting server power consumption from standard rating results. In *Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering*, ICPE '19. ACM, 2019.

[159] W. Voorsluys, J. Broberg, S. Venugopal, and R. Buyya. Cost of virtual machine live migration in clouds: A performance evaluation. In M. G. Jaatun, G. Zhao, and C. Rong, editors, *Cloud Computing*, pages 254–265, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[160] L. Wang and S. U. Khan. Review of performance metrics for green data centers: a taxonomy study. *The Journal of Supercomputing*, 63(3):639–656, 2013.

[161] C.-M. Wu, R.-S. Chang, and H.-Y. Chan. A green energy-efficient scheduling algorithm using the DVFS technique for cloud datacenters. *Future Generation Computer Systems*, 37:141–147, 2014.

[162] Q. Wu, F. Ishikawa, Q. Zhu, and Y. Xia. Energy and migration cost-aware dynamic virtual machine consolidation in heterogeneous cloud datacenters. *IEEE Transactions on Services Computing*, pages 1–1, 2018.

[163] Y. Wu and M. Zhao. Performance modeling of virtual machine live migration. In *4th International Conference on Cloud Computing*, pages 492–499. IEEE, 2011.

[164] J. Xu and I. S. Moreno. Energy-efficiency in cloud computing environments: Towards energy savings without performance degradation. *International Journal of Cloud Computing*, 1(1):17–33, 2011.

[165] M. Zakarya. Energy, performance and cost efficient datacenters: A survey. *Renewable and Sustainable Energy Reviews*, 94:363 – 385, 2018.

[166] H. Zeng, C. S. Ellis, A. R. Lebeck, and A. Vahdat. Ecosystem: Managing energy as a first class operating system resource. *SIGPLAN Not.*, 37(10):123–132, 2002.

[167] F. Zhang, G. Liu, X. Fu, and R. Yahyapour. A survey on virtual machine migration: Challenges, techniques, and open issues. *IEEE Communications Surveys Tutorials*, 20(2):1206–1243, 2018.

[168] J. Zhang, F. Ren, and C. Lin. Delay guaranteed live migration of virtual machines. In *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, pages 574–582, 2014.

[169] Z. Zhang, L. Cherkasova, and B. T. Loo. Exploiting cloud heterogeneity to optimize performance and cost of mapreduce processing. *SIGMETRICS Perform. Eval. Rev.*, 42(4):38–50, 2015.

[170] M. Zhao and R. J. Figueiredo. Experimental study of virtual machine migration in support of reservation of cluster resources. In *Proceedings of the 2Nd International Workshop on Virtualization Technology in Distributed Computing*, VTDC '07, pages 5:1–5:8, New York, NY, USA, 2007. ACM.

[171] J. Zheng, T. S. E. Ng, K. Sripanidkulchai, and Z. Liu. Pacer: A progress management system for live virtual machine migration in cloud computing. *IEEE Transactions on Network and Service Management*, 10(4):369–382, 2013.