Assessment of Machine Learning as an Approach to Virtual Machine Selection

Dean Connell

School of Computer Science

National University of Ireland Galway

Supervisor(s)

Dr. Enda Howley

In partial fulfillment of the requirements for the degree of

M.Sc. in Computer Science (Data Analytics)

September 7, 2022

DECLARATION I, Dean Connell, hereby declare that this thesis, titled "Assessment of Machine Learning as an Approach to Virtual Machine Selection", and the work presented in it are entirely my own except where explicitly stated otherwise in the text, and that this work has not been previously submitted, in part or whole, to any university or institution for any degree, diploma, or other qualification.

Signature: *Dean Connell*

Abstract

Cloud computing is a computing paradigm which has seen monumental growth due to desirable qualities for hosting enterprise applications. As a result, data centres are being utilised on a larger scale to perform intense and demanding computational tasks, which results in a large strain on energy resources. Coupled with sub-optimal energy management techniques, the needless level of energy consumption has become alarming and is a cause for concern in context of environmental sustainability. This thesis aims to provide extensive research into the mishandling of energy in data centres and proposes a machine learning based approach to the selection and migration of virtual machines, based on a metric of lower overall energy consumption. The optimal selection of virtual machines may save major amounts of energy, by maintaining servers at ideal capacity for functionality and energy expenditure. The proposed method yields consistently lower energy consumption than the state-of-the-art LrMmt algorithm, averaging an improvement of 3.15%. However, this is achieved whilst also lowering SLA violations by 28.53%. Ultimately, this thesis will conclude that machine learning methods are a viable option for the selection of virtual machines for migration.

Keywords: Energy Optimisation, Cloud Computing, Dynamic Resource Allocation, Machine Learning.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

In this chapter, the motivation of exploring the topic of data centre energy optimisation will be highlighted, initial research questions will be defined and the structure of this thesis will be briefly outlined. This will aim to provide a comprehensive introduction to the goals of this research.

1.1 Motivation

In an age of great adversity due to climate change, consciousness about carbon footprint and power usage has been at the forefront of media and government attention. Particularly, data centres have been criticised in Irish media in recent years for the alarming level of power consumption, with the Irish Examiner reporting that data centres could account for 30% of all electricity consumption by 2030 [5]. This project will aim to alleviate this level of power consumption and slow this trend, by using advances in the field of machine learning to optimise the allocation of server resources to tasks in data centres.

1.2 Research Questions

The goal of this research project will be to supply coherent answers to the following research questions:

- RQ1: Which machine learning algorithm shows the most promise in this field of application? Can this algorithm integrate into the virtual machine selection process?
- RQ2: What are the optimal hyperparameter settings for this problem, given our choice of machine learning algorithm? What metric is most appropriate to measure the performance of our algorithm in this case?
- RQ3: Is the performance of our algorithm comparable to the state-of-the-art benchmarks employed by industry?

1.3 Structure of Thesis

For this project to become a reality, an intensive literature review must be conducted. This will encompass many research topics, such as the cloud computing paradigm; its evolution, characteristics, architecture and deployment models. This will be followed with a review of energy consumption pertaining to cloud computing centres. Finally, articles on machine learning will be reviewed, investigating different algorithms and how they can be, and have been, applied to this resource optimisation problem.

To gain an idea of topical issues in this problem domain, a detailed review of similar literature will also be conducted. Relevant information from these projects such as tools used, general approach and experimental conditions will be gathered and analysed. This is done such that this project is undertaken in a similar way, with an emphasis on comparison to previous benchmarks in this

domain. Limitations or possible future work suggested by these projects will be noted, to suggest research questions as a starting point for the purpose of this project.

Chapter 2

Background

This chapter aims to give a brief overview of the technical knowledge required to understand the problem domain of this project.

Modern computing paradigms will be explored, such as the differences between cluster, grid and cloud computing. Extensive research will be conducted on the different characteristics of cloud computing, many of which make it a desirable choice in industry. The typical architecture of a cloud computing network will be analysed level-by-level before looking at different kinds of cloud platforms in operation. Next, the concept of data centres will be visited, defining the concept of a data centre. Following on from this, data centre energy consumption will be debated, with a focus on energy consumption statistics and trends.

Technical knowledge pertaining to the proposed algorithmic solution to this problem will then be discussed. Beginning with foundational principles of machine learning such as supervised and unsupervised learning, reinforcement learning will then be explored along with deep learning. Finally, the CloudSim technology will be briefly explored, which will be used as a virtual environment to deploy our energy optimisation solution.

2.1 Computing Paradigms

Given the availability of hardware to perform automated tasks, this hardware can be arranged in a number of different ways to enhance the completion of each task, given the usecase. In this section, a number of different types of computing will be explored, detailing the context of the rise of cloud computing as a viable option.

2.1.1 Cluster Computing

Cluster computing is defined as the use of multiple separate computers which all work on computational tasks together. More specifically, this refers to the use of two or more computers which are totally separate from one another [6]. These computers work together as a single machine to execute these tasks, with each computer commonly referred to as a node in this local area network. This framework may be viewed as a hierarchy of nodes in some cases, with a parent and child nodes (also known as root and slave nodes), but all nodes contribute to the effective problem solving process.

With all of these nodes engaged and acting on a single task, cluster computing is a solution which increases performance times and scalability, without becoming a complex architecture.

2.1.2 Grid Computing

Akin to cluster computing in many ways, grid computing also consists of multiple node computers connected together and tackling computational tasks as a single unit. However, the subtle difference between the grid and cluster approach is that these computers are not viewed as independent bodies as such, and instead interact systematically on a data grid. These individual computers on the

network interact and share processing power enough to be termed as a virtual supercomputer [6]. It is also important to mention that a grid is not just based on a local area network - individual computers can be distributed geographically.

As is the case with cluster computing, this solution is suited to larger problems which may pose scalability issues to a single processing unit. The architecture is more complex than computing clusters in general, but promotes solving computational problems in a time efficient manner.

2.1.3 Cloud Computing

Cloud computing takes a different approach to computation than the previously discussed paradigms. Sharing some similarities to grid computing, computers are connected using wireless networks, which means that components of a cloud system can be distributed geographically [6].

However, the differences are more apparent when we consider the distribution of resources on grid and cloud architectures: cloud computing uses a centralised model, in which all of the computation is conducted on a single common server. In contrast, grid and cluster computing are decentralised, in which tasks can be split out on multiple computers.

2.2 Cloud Computing Characteristics

The success of cloud computing as a viable method to tackling large-scale problems is a direct effect of the distinct characteristics which cloud architecture possesses. In this section, these distinguishing characteristics of cloud computing will be outlined and discussed.

2.2.1 Affordability

Traditionally, in industrial settings, capital may be raised to finance the purchase of computing equipment. However, with the onset of cloud computing services, this upfront purchase of major technology to run business systems is not required. Instead, businesses may opt to use these services - which operate on a pay-per-use model. Rather than investing in equipment, which is an expensive commitment, instead businesses can utilise cloud processors and storage facilities on a pro-rata basis. In addition to this, the cloud provider also benefits from this arrangement, as providing these services on a large scale makes the investment more affordable for them [1].

2.2.2 Accessibility

Another major advantage of cloud computing technologies is the accessibility of cloud applications. Cloud hosted applications can be accessed using common network protocols such as HTTP or TCP/IP through any device with network capabilities. This remote hosting of applications is very useful, as these services can be accessed from any geographical location - a quality which was pertinent to the survival of the economy throughout the COVID-19 pandemic.

2.2.3 Availability of Resources

Another issue which may be identified from the hosting of an application on a static centralised server is the availability of resources on that server. It may be difficult to match or gauge the demand of the customer base and have the correct provisions in place for server traffic on a self hosted server. Instead, cloud services provide an on-demand availability of resources, which can be accessed according to the level of demand on the aforementioned application servers [7].

2.2.4 Agility

In our current economic climate, there is huge demand for the agile implementation of change in systems when change is required. Regular systematic updates can be performed on cloud architecture with great ease by using the extra resources available. Because of the processing speed and ability to deal with larger scale computational tasks, large systematic change procedures have a greatly reduced wait-time, leading to the more agile and smooth running of large computational processes [8].

2.3 Cloud Computing Architecture

Before exploring different types of cloud platforms that are available, it is important that the architecture of a typical cloud computing framework is well-defined and understood. Here, the four layers of cloud architecture are discussed - the application, platforms, virtualisation and hardware layers. These layers contribute to the ability for a cloud services company to provide different types of service, namely Software as a Service (SaaS), Platform as a Service (PaaS), Infrastructure as a Service (IaaS) and the less common Hardware as a Service (HaaS). These services will be explored in more detail below.

2.3.1 Application Layer

The application layer is the uppermost layer of the cloud architecture and can be accessed easily through the web. This layer is made up of typical cloud-based applications and web services such as Facebook, Youtube or Salesforce using the SaaS approach. These applications have access to unlimited resources as detailed above for a pay-as-you-go fee to the cloud service provider [9].

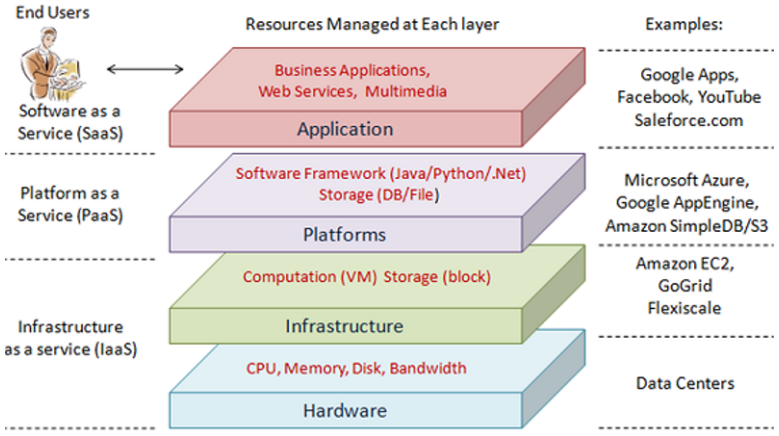

Figure 2.1: Schematic representation of cloud computing architecture [1]

2.3.2 Platforms Layer

This layer is associted with the PaaS approach and exists between the application and the virtualisation layers [10]. This layer is characterised by software frameworks and storage options, and provides software engineers the necessary tools to develop and launch cloud based applications. In this case, the service provider will control these tools such as frameworks, databases and packages required to run these applications.

2.3.3 Virtualisation Layer

Often seen as the layer that defines the cloud computing paradigm, the virtualisation layer (also referred to as the infrastructure layer) exists between the platforms and the hardware layers [11]. This layer is associated with the IaaS approach, and is characterised by the creation and existence of virtual machines and dynamic resource allocation. In this case, the service provider can hire out dynamic infrastructure with administrative access to virtual machines, and give the user complete access to their computational resources.

2.3.4 Hardware Layer

Finally, the hardware layer is the lowermost layer of the cloud computing architecture and is associated with the less common HaaS approach, in which the cloud computing hardware is hired for use by a company. This layer accounts for the physical machinery that is used to run a cloud computing establishment such as a data centre. This machinery includes processing units, internet routers, storage, power sources and cooling fans [11].

2.4 Types of Cloud

The cloud architecture previously discussed can be adapted for use in many different cases. Given the advantages and characteristics of cloud computing, many different variations of cloud computing have gained traction in recent years such as public, private, hybrid and community clouds. This section will give a brief overview of the deviations each cloud has from one another.

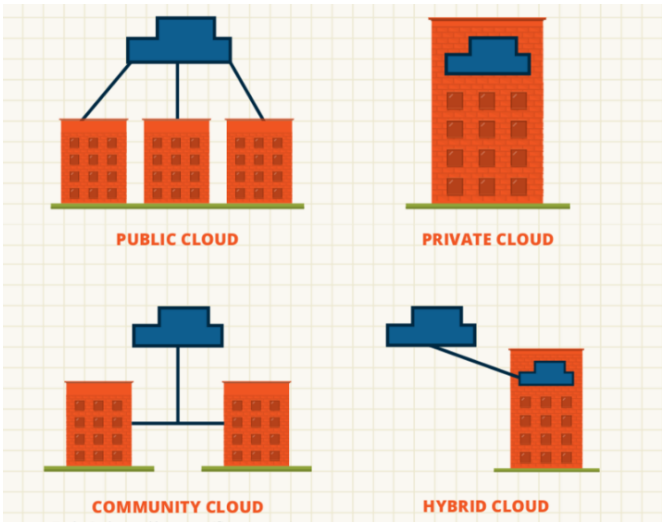

Figure 2.2: Simplistic diagram detailing different cloud types [2]

2.4.1 Public Clouds

Public clouds are commonly used in applications where services are supplied to ordinary members of the public [12]. These clouds offer many of the advantages of cloud architecture previously discussed, such as resource availability, price advantages and ease of accessibility. However, public clouds are typically the least secure and offer the least control over network processes.

2.4.2 Private Clouds

Private clouds, on the contrary, are not accessible by the general public. These are usually built up with existing private infrastructure and have some authentic users that can dynamically provision the resources [12]. While this approach has a lot of advantages such high class security and fidelity, private clouds can be very expensive to set-up and don't have the same scalability options you would expect from a public cloud.

2.4.3 Hybrid Clouds

Hybrid clouds are a combination of elements from the public and private models. More specifically, these are heterogenous distributed systems, which start with a private cloud model and incorporates elements of public cloud models such as services and resources [12]. This results in a compromise between public and private clouds, defining a more cost-effective and scalable solution to a private cloud, while improving on the control and security issues with public clouds. However, careful resource management is required to effectively split data between public and private elements.

2.4.4 Community Clouds

Community clouds are an amalgamation of different clouds to meet a particular common organizational need. In this case, a number of organizations may come together that share certain requirements such as similar privacy needs or security systems [12]. This cloud system has similar characteristics to hybrid clouds in terms of cost and security, and typically are more reliable and boast a better performance overall - due to catering to specific organisational needs. However, the scalability of community clouds is akin to private clouds, with limited room for expansion.

2.5 Data Centres

The rise of data centres has been characterised by the increased demand for computational power and cloud based services in recent years. With more and more applications deployed on web-based servers as we move forward, there is increased pressure on energy resources to meet this demand. This higher demand for cloud based services had a malignant effect on energy consumption, which will be explored in more detail in the following section.

2.5.1 Energy Consumption

According to [3], data centre energy consumption in Ireland has skyrocketed in recent years. The overall proportion of national metered electricity that is being consumed by operating data centres has increased from 5% in 2015 to 14% in 2021 according to figures from the Central Statistics Office. This accounts for more energy consumption than all rural Irish homes combined in 2021, which make up 12% of the share. Furthermore, these data centres have seen a growth of consumption of 265% between Q1 of 2015 and Q4 of 2021, and this trend

seems to be continuing with the addition of new data centres to the Irish power grid. A spokesperson from Eirgrid has predicted that data centres could account for 23-30% of metered electricity consumption by 2030. This characterises an alarming increase in national power consumption, which must be tackled from both an economic and sustainability standpoint.

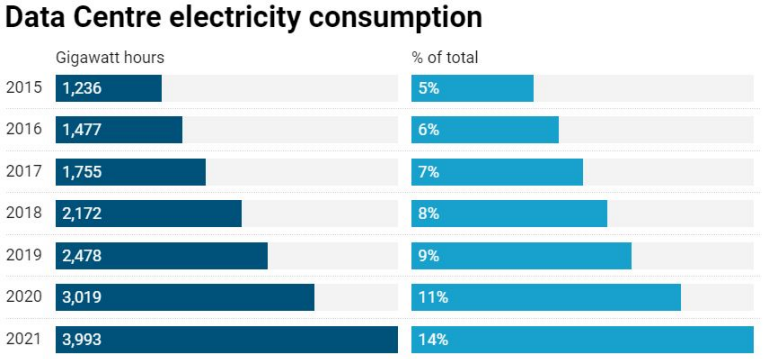

Figure 2.3: Irish data centre energy consumption trends [3]

2.5.2 Energy Management Solutions

Many ideas have been leveraged to tackle energy mismanagement in data centres. Approaches that have been taken to date include the use a simple thresholding model to migrate tasks to other servers. This is done once the current server is over-utilised and is at sub-optimal energy expenditure. Other approaches using Artificial Intelligence (AI) have been proposed, in which virtual machine migration and server selection are performed by means of machine learning techniques. In the following chapter, I will elaborate on some common practises used to reduce energy consumption.

2.6 Machine Learning Paradigms

One proposed solution to energy management problems that is beginning to come to the fore in research settings is the concept of machine learning. Machine learning is a discipline which leverages data to improve performance on some predefined set of tasks [13]. Usually, machine learning models train on some sample data, and tune themselves to make decisions given new data without being programmed explicitly to do so. Some common approaches to machine learning will be explored below - namely supervised, unsupervised, reinforcement learning and deep learning.

2.6.1 Supervised Learning

Supervised Learning is defined as the construction of a model given training data with input and output values [14]. Using this initial set of training data, an objective loss function (depending on the algorithm chosen) is minimised, which in turn allows for the optimisation of the hypothesis function. This hypothesis function can then be used to predict the output value, given a set of input values. The performance of this function is characterised by the ability to correctly predict output values of new data, given the corresponding set of input values.

2.6.2 Unsupervised Learning

Conversely, unsupervised learning works with data that comprises of only input values i.e. there is no defined output value. The goal of unsupervised learning is to find structure in the data by grouping it into sets depending on common attributes. In training an unsupervised learning algorithm, similarity criteria for grouping observations are extracted from the dataset, and these criteria are used to identify candidate subsets for unseen observations.

2.6.3 Reinforcement Learning

Reinforcement learning is unlike the aforementioned learning paradigms in how it operates. In this case, rather than using explicit input data and mathematical techniques to optimise a hypothesis function, reinforcement learning focuses on striking a balance between explorative and exploitative tendencies of intelligent agents, to maximise reward. In essence, this method involves these agents learning through trial and error how to interact with their environment [15]. This learning is facilitated by rewarding desirable behaviour and punishing undesirable behaviour.

2.6.4 Deep Learning

Deep learning is a subset of machine learning and is characterised by a composition of multiple processing layers which learn representations of data. Learning is facilitated by the use of a backpropogation algorithm to update internal parameters, known as weights. These weights are then used to compute data representations in the following computational layer using representations in the current layer. A deep learning architecture typically consists of a multilayer stack of simple modules, which all transform input to increase the invariance of data representations [16]. These multilayer systems can implement complex hypothesis functions, allowing for better performance. Deep learning frameworks exist in many different forms, with examples including deep neural networks, deep reinforcement learning, recurrent and convolutional neural networks.

2.7 CloudSim

The CloudSim platform will be used and referenced throughout this project to see the effects of proposed energy optimisation techniques. This is an open-

source framework that can be used to simulate cloud computing architecture and services. This platform will allow for the deployment of proposed solutions on a simulated data centre, and will provide the tools which will form part of the performance analysis of these methods.

Chapter 3

Literature Review

Because of the inefficient management of resources in cloud computing settings, there is evidence of quite a lot of research into solutions to combat the volume of energy consumption in these settings. The more prudent management of available resources and subsequently reduction of overhead costs, seem to be the goals of many of the existing literature on this subject. Both of these outcomes are desirable to the cloud service provider but also have a positive effect on the environment and consumer satisfaction.

Of course, modelling the efficient management of resources is a challenging problem due to the dynamic nature of the demand on cloud servers. There is no clear optimal solution in this case, however many solutions have been explored pertaining to energy efficiency and favourable allocation of resources. These approaches tend to be reducible into one of these three main common pillars:

- Threshold Based Approaches
- AI Based Approaches
- Reinforcement Learning Based Approaches

Existing literature corresponding to each of these approaches will be grouped and analysed in the following sections. Finally an overarching critical appraisal will be documented, giving personal insight into domain developments.

3.1 Threshold Based Approaches

The methods discussed in this section have used simple thresholding criterion to ameliorate energy management strategies and promote a more efficient use of available resources. These did not require AI - rather establishing basic heuristics determining when to migrate jobs to different servers based on throughput.

3.1.1 ECTC and MaxUtil

Extensive research was carried out into power consumption depending on server state in Lee et al. [17]. It was noted that idle servers still consume power, and an approach that takes this baseline consumption into account could lead to the more efficient management of power. In this case, two rule-based approaches were defined - namely ECTC and MaxUtil. Sharing a common goal of utility maximisation, these take into energy consumed on both active and inactive servers - assigning jobs to servers that require the least power to complete. MaxUtil uses averaging techniques and thresholds over these to identify where to place new jobs. On the other hand, ECTC ascertains a metric for energy consumed and manages to save 18% more energy than random allocation.

3.1.2 Consideration of Server Performance

Further development on effectively managing power consumption and under-utilised servers is researched in Srikantaiah et al. [18]. Research was also conducted into the impact of workload on system performance, with an algorithm

proposed that aimed to find the optimal allocation of jobs to servers considering energy consumption. This algorithm identified that optimal levels of energy consumption can be achieved at specific levels of performance and utilisation. However, the optimal level of performance found would not be deemed sufficient for service-level agreements, so a lower performance limit must be imposed for compliance purposes. Once this performance level is met on a particular server, jobs are deployed on a new server, and this process continues.

3.1.3 Virtual Machine Placement and Selection

Perhaps the most applicable threshold based approach to this project can be found in Beloglazov et al. [19]. Rather than focusing solely on assigning tasks to the correct server, this approach defines two tasks: Virtual Machine (VM) placement and VM selection.

VM placement defines the determination of the optimal server to host a virtual machine. This is done via a heuristic algorithm called power-aware best fit decreasing (PABFD), in which VMs are arrange in descending order based on their use of processing power and then subsequently allocated to servers which results in the lowest energy consumption.

VM selection defines the identification of optimal VMs to offload on the occasion that a server is overloaded. In this paper, three selection methods were outlined: Minimization of Migration (MM), Highest Potential Growth (HPG) and a random VM selection process. MM is a very basic policy, selecting the minimum number of VMs to offload such that server load is sub-threshold. HPG aims to move VMs that require the lowest processing power in proportion to the processing capacity of the server on which they operate. In the evaluation of these policies, service-level agreement violations were used as a performance metric, combined with energy usage and migration frequency. The MM policy

outperformed HPG and random selection on this metric.

3.1.4 VM Placement and Dynamic Thresholding

Following from the success of their research in VM selection and placement, Beloglazov et al. [20] conducted further study into the latter, identifying benefit in constantly considering which servers are optimal for each VM. Given the stochastic nature of cloud computing workloads, this research takes a dynamic approach to VM placement. New heuristics are defined for this process which aid in identifying server overuse. This new method uses local regression (Lr), which approximates data trends using parametric equations. These equations can be used as a prediction model to identify the probability of over-utilisation of servers in the following timesteps.

In addition to this, the development of a new VM selction policy is explored, namely Minimum Migration Time (Mmt). In this policy, the VM which would have the lowest movement duration is chosen to be migrated to a different, non-overloaded server. In combining the LR method to detect server overuse, the Mmt policy for VM selection and the PABFD algorithm for VM placement, there is a huge reduction in energy overheads due to server mismanagement.

3.2 AI Based Approaches

Rather than focusing on defining thresholds that cause a reaction once they are reached, this section will detail approaches that depend on AI to make decisions. Just as in the previous section, the main decisions pertain to where cloud based tasks should be placed and when they should be migrated.

3.2.1 Genetic Algorithms

In Portaluri et al. [21], a genetic algorithm is developed for the sole purpose of reducing energy overheads in data centres by prudently managing their resources. In basic terms, genetic algorithms aim to converge to optimal solutions by using concepts such as selection and evolution. In this case, the focus of the algorithm is to reduce both task completion time and energy overheads. Given the multi-objective nature of this problem, the Non-dominated Sorting Genetic Algorithm II is deployed, which discards any potential solution that is comparably unfavorable. As a result of running this algorithm, a number of favorable solutions will be obtained which place different levels of emphasis on the time-energy trade-off.

3.2.2 Particle Swarm Optimisation

By making slight modifications to the state-of-the-art Particle Swarm Optimisation (PSO), Dashti et al. [22] aims to optimise the VM placement process by dynamic means. In this research, VM placement is framed simply as a binpacking problem with bins as servers, items as VMs, bin size as processing power and cost as energy overheads of each server. Within this approach, the Mmt policy discussed above is used for the VM selection process and the VM placement process is determined by the modified PSO algorithm. This modification calculates a fitness value for potential placements by using a function of energy consumption increase for each. The resulting placement with the lowest energy increase is picked.

3.2.3 Supervised Machine Learning

A slightly different approach to what has been seen so far is explored in the research of Berral et al. [23] using machine learning methods. In this paper,

procedures such as powering servers on or off and consolidation algorithms are explored, in order to deal with stochastic processes. Supervised learning is utilised to form predictive models pertaining to the effects of different workloads on available servers. Energy overheads and server performance are used as a metric to determine these effects, and act accordingly. These actions could be to migrate workloads or to power down less busy servers.

3.2.4 Game Theory and Evolutionary Optimisation

Perhaps a more interesting approach, Wei et al. [24] frames this problem in the context of game theory. Firstly, integer programming is used to ascertain a local solution for each agent independently. Then an evolutionary method is used on an aggregated set of local solutions, uses this to approximate a global optima and uses this as a policy to allocate available resources.

3.3 Reinforcement Learning Based Approaches

Just as in the literature previously reviewed, reinforcement learning techniques can be applied to the tasks of VM selection and placement. Below, approaches that harness the power of reinforcement learning will be discussed, detailing the main focus of each application. Some of these cases involve the use of Multi-Agent Reinforcement Learning (MARL), to track the performance of multiple parties acting in the data centre environment.

3.3.1 Performance-Consumption Optimisation with MARL

One such case that deploys MARL to this energy optimisation problem is documented in research undertaken by Das et al. [25]. As in the supervised machine learning approach seen in the previous section, this approach aimed to switch off

hardware with low workloads to conserve energy. This method also incorporates load balancing across the data centre, allowing for the effective reallocation of tasks to facilitate the powering off of selected servers.

3.3.2 RL for VM Selection

In research undertaken by Duggan et al. [26], reinforcement learning is used for the VM selection process described previously. The goal of this research was to train an agent to identify the best VM relocation action to take, being rewarded when energy consumption is reduced. The Lr algorithm is incorporated to flag potential VMs for migration. This research is of particular note to the undertakings of this study because of their use of CloudSim to deploy the RL model for experimentational purposes. The reinforcement learning method proposed in this paper is compared to the state-of-the-art LrMmt algorithm under the usual metrics: service-level agreement violations, energy usage and migration frequency. The method proposed in this paper made significant improvements over these metrics, consolidating the viability of reinforcement learning in this domain.

3.3.3 RL for VM Placement

Contributing to this area even further, Shaw et al. [27] adopted the work of Duggan et al. and seeked to implement an improved system for VM placement. The proposed algorithm trains an agent to perform the VM placement process and uses this in conjunction with the existing Mmt algorithm for VM selection. Again, experimentation was conducted on CloudSim to monitor the performance of the RL agent. Once again, this proposed method performed better than LrMmt over the metrics of service-level agreement violations, energy usage and migration frequency. Yet again, confidence was established in the utility of reinforcement learning for the reduction of data centre energy consumption.

3.4 Critical Appraisal of Research

From examining related literature, it is clear to see the viability of machine learning and reinforcement learning in the domain of efficient energy optimisation. In the past, both ML and RL have been successfully implemented to alter elements of the state-of-the-art methods, and have been shown to contribute to the more efficient management of resources. Given the success of these projects, a prudent decision would be to build on the existing work of Duggan et al. [26], developing a novel approach that can manage the VM selection process in data centre settings.

While the progress that reinforcement learning is making in this domain is exciting, it is very possible that it may be a gross over-complication of the problem at hand. It is proven to be a viable option in this field, but when moving into an era where value is placed on explainable AI, this solution may be too complex. In addition to this, reinforcement learning may be too heavyweight and produce latency issues in an application domain where high fidelity is key to ensure the efficient running of cloud services.

As a result, a more basic VM selection heuristic will be sought after, using lighter machine learning methods to promote explainability and fidelity. The aim is that this may yield a comparable or even more optimal performance across service-level agreement violations, energy usage and migration frequency. If this is successful, it could revolutionise resource management in data centres, and improve the economic and sustainable viability of data centres for years to come.

Chapter 4

The CloudSim Environment

In the realisation of this project, it is important to ensure that the proposed method can be tested under varying conditions over time. In addition to this, it is desirable to compare the performance of the proposed method with established methods on the same workload. This may be achieved in many different ways, but for the purposes of this project, the CloudSim simulation environment will be used.

4.1 CloudSim Background

The CloudSim environment was created initially in the University of Melbourne, Australia in the Cloud Computing and Distributed Systems (CLOUDS) Laboratory [4]. Written entirely in Java, this modelling and simulation environment can be used to create a theoretical data centre parameterised entirely by changing input variables. Since its inception, the CloudSim environment has been a staple in cloud computing research, used widely in this area of resource optimisation. The ease at which solutions to optimisation problems can be implemented into the software makes CloudSim a desirable tool for use in this research domain. New

implementations can easily be integrated into the system through the wide use of abstract classes across the platform, which can easily be extended and interfaced. Furthermore, there are a wealth of libraries conducive to the implementation of energy aware solutions included in the build. Notably, implementations of the previously discussed local regression and threshold based VM selection policies and VM placement policies such as minimum migration time, amongst others, are integrated into the environment and available to run.

4.2 CloudSim Structure

In order to integrate a working solution into CloudSim, it is important to first understand the structure of the environment. In this section, I will briefly outline the main Java classes used in the implementation of a power aware VM selection solution. These classes have been described in detail in [4], and thus I am providing a simple outline of these classes for the purposes of this project.

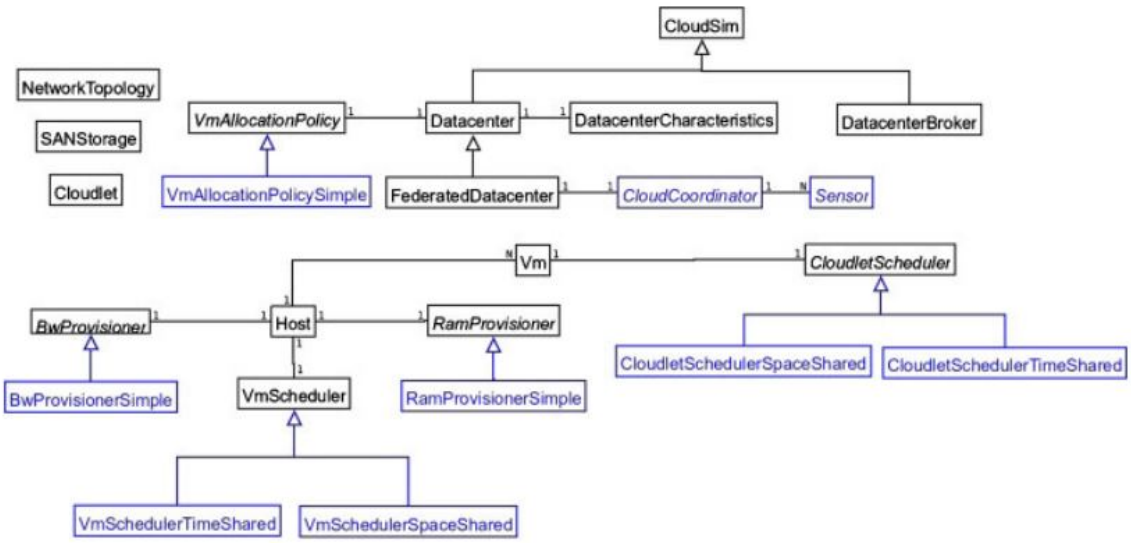

Figure 4.1: Structural dependencies present in CloudSim architecture [4]

- **BwProvisioner**: This class provides for the allocation of virtual machine bandwidth across the entire data centre. This abstract class can be extended to implement alternative bandwidth allocation methodologies.

- **CloudCoordinator**: This is in charge of monitoring the state of the abstract data centre and dynamic load-shredding processes. Once again, this is an abstract class which can be extended to integrate custom federated resource monitoring.

- **CloudletScheduler**: This class determines the share of CPU allocated to virtual machine cloudlets through class extension.

- **Datacenter**: This is responsible for modelling the overall hardware configuration of the data centre. In addition to this, each Datacenter instance corresponds to the configuration of an application provisoner, which is responsible for allocating bandwidth, memory and storage to virtual machines.

- **DatacentreBroker**: Essentially, this acts as a broker in terms of managing interactions between providers of software as a service and general cloud applications. This class may be extended to evaluate novel brokerage implementations.

- **DatacenterCharacteristics**: The purpose of this class is to store the values of global variables pertaining to the definition of resources in the abstract data centre.

- **Host**: The function of this class is to model each server in the theoretical datacentre. This encompasses specifications such as storage, memory and job scheduling methods.

- **Network Topology**: This defines the general network structure of the entire datacentre and models communication latency between servers.
- **RamProvisioner**: This abstract class is primarily responsible for the RAM allocation to virtual machines.
- **Vm**: This class models a virtual machine, which is managed and hosted by an instance of the Host.java class. These VM instances have access to typical virtual machine characteristics - memory, storage, bandwidth and a provisioning policy, extended from the CloudletScheduler class.
- **VmAllocationPolicy**: This is an abstract class which may be extended and overrided to implement a novel policy for the task of VM allocation. The functionality of this class is to select an available host that meets the storage and memory requirements for deployment.
- **VmSelectionPolicy**: Similarly to the VmAllocationPolicy class, this class is extended to implement a VM selection policy. The function of this class is to identify and reallocate virtual machines which are operating on over-utilised hosts.
- **VmScheduler**: This class, implemented by a host instance, simply models policies which allocate processors to virtual machines.

4.3 Project Integration in CloudSim

In order to be successful in defining a novel VM selection policy, it must be integrated seamlessly into the existing architecture. In addition to this, to maintain comparability with previous advancements in this field, the existing architecture may not be modified in such a way that changes the structure entirely. In this

section, any minor changes effected in existing CloudSim classes will be explained, to maintain the reproducibility of the results of this study.

4.3.1 Running Energy Aware Simulations

The implementation of energy aware policies is facilitated by the power subfolders in the CloudSim project. In this case, we are concerned with the path org.cloudbus.cloudsim.examples.power.random. As mentioned previously, there are a number of energy aware VM selection and VM allocation policies available to run. For example, implementations of LrMmt, LrrMmt, ThrMmt and ThrRs exist here and are defined with string objects representing each policy. These are subsequently passed to a runner object in order to run the selected implementation. The creation of a new energy aware policy works much the same as the above - a new java file is created in the same subfolder and the VM selection and allocation policies specified as String objects within. These are passed to a RandomRunner object, which simulates these methods on a randomly defined workload. This outputs relevant metrics pertaining to power consumption, SLA violations and VM migrations.

4.3.2 Implementing VM Migration Policies

In order to work with previously undefined VM selection or allocation policies, these must be defined in the CloudSim system. In the example above, when the String object representing the selection or allocation policy is passed to the runner object, this object must find the actual policy desired. Such policies are stored in the org.cloudbus.cloudsim.power path. A newly defined policy of this type must extend and override either the VmSelectionPolicy or VmAllocationPolicy class depending on the usecase. More specific to this project, when creating a new VM selection policy the getVmToMigrate method must be overridden and supplied

with a migration technique. Otherwise, no such migrations would occur.

4.3.3 Configuration of Hardware

CloudSim allows the user to choose the configuration of hardware and the overall characteristics for the abstract data centre. It is important that the configuration of this experimental data centre is accounted for and controlled for all tests.

In this case, a lightweight datacentre consisting of 200 dual-core servers was opted for. These servers each were provided with 1GB of disk space and a bandwidth of 1GB/s. Each server was also provided with 4GB of memory. Finally, two different host configurations were supplied as per the default CloudSim settings, both of which are HP ProLiant power models:

- ProLiant ML110 G4 Xeon 3040 (1860MHz)
- ProLiant ML110 G5 Xeon 3075 (2660MHz)

In terms of virtual machine instances, the default configuration of 4 virtual machine types was used:

- High-CPU Medium Instance: 2.5 EC2 Compute Units, 0.85 GB
- Extra Large Instance: 2 EC2 Compute Units, 3.75 GB
- Small Instance: 1 EC2 Compute Unit, 1.7 GB
- Micro Instance: 0.5 EC2 Compute Unit, 0.633 GB

By using a random workload, each instance is chosen at random from the types above modelling a realistic workload given the lower scale of the project. In this case, a stochastic workload of 300 virtual machines is generated on each run. For sections in which a direct comparison is made between VM selection/allocation models, provisions have been made to use the same workloads in experimentation.

Chapter 5

Evaluation Metrics

The aim of this chapter is to outline and furthermore identify the metrics on which the performance of each VM allocation and selection method may be compared. As seen in Chapter 3, many studies use a combination of metrics to determine the optimal method. These metrics will be defined in this chapter and explored in detail.

5.1 Energy Consumption

This metric encapsulates the total amount of power consumed by the theoretical data centre. Possibly the most important metric due to the focus of this study, our aim will be to minimise this metric as much as possible while still keeping the data centre at an acceptable working capacity. The units for this measurement is kilowatt hours (kW/h).

5.2 Virtual Machine Migrations

In addition to the goal of minimising power usage, it is important to look at the frequency of virtual machine migrations. Increases in power consumption can correspond with the movement of virtual machines, so we will aim to reduce the amount of migrations. Furthermore, frequent VM movement can contribute to increases in SLAVs, as time may be wasted by moving VMs unnecessarily. Thus, the development of a machine learning equipped VM selection algorithm is a prudent approach to this minimisation.

5.3 Performance Degradation due to Migrations

This metric, also known as PDM, aims to capture the decrease in performance efficiency as a result of moving virtual machines around the data centre. Ideally, this metric would yield a lower result so that VM migration would incur less of a performance overhead. This performance loss is calculated as the mean of the estimated individual PDM of each machine divided by total requested CPU capacity by each machine. This is formalised by the equation below:

$$PDM = \frac{1}{N}\sum_{i=1}^{N}\frac{D_i}{C_i}$$

where we have N virtual machines, D_i is the estimated performance degradation from moving machine i and C_i is the total CPU capacity requested by machine i over the course of the simulation.

5.4 Service Level Agreement Violation Time per Active Host

This metric, also known by the acronym SLATAH, simply describes the amount of time at which each active host is totally utilised. Again, the optimal solution would aim to keep this metric minimised, as totally utilised or over-utilised hosts will not be able to perform tasks as effectively. This would have a knock-on effect, preventing VMs from working efficiently. SLATAH is calculated by the mean of each machines proportion of time spent at maximum utilisation. The equation below formalises this notion:

$$SLATAH = \frac{1}{M} \sum_{i=1}^{M} \frac{T_i^m}{T_i^a}$$

where we have M hosts, T_i^m is the total time host i spends at maximum utilisation and T_i^a is the total time host i is active.

5.5 Service Level Agreement Violations

Service level agreement violations (SLAVs) are a very important consideration when assessing the overall performance of data centres. The goal of this project is to minimise the increase of these service violations while decreasing energy consumption. Perhaps the most critical measurement to be considered, it is important that a high level of service is maintained for clients using cloud services. This metric is a multiplicative combination of the PDM and SLATAH metrics discussed previously.

$$SLAV = PDM \times SLATAH$$

5.6 Combination of Metrics

In the objective to minimise multiple metrics, it may be prudent to consider a combination of important metrics. In this instance, it is discussed in [28] that energy consumption and SLAVs generally have an inverse relationship. This would mean that a decrease in one generally relates to an increase in the other. In order to minimise both energy consumption and SLAVs, a function of the two will be considered as a metric, simply called energy efficiency as seen in [28]. This function is outlined below:

$$EE = \frac{1}{SLAV \times EC}$$

Maximising this function theoretically yields the optimal solution for minimised SLAVs and energy consumption. More simply, higher values of EE may correspond to better solutions for the purposes of testing. This will be investigated further in the following chapter.

Chapter 6

Benchmarking

In this chapter, existing methodologies for virtual machine selection and migration will be explored and benchmarked as initial experimentation. This will form basis for the identification of a niche in this domain, which can be identified using data analysis techniques.

6.1 Techniques Benchmarked

Four existing implementations will be tested using the experimental conditions defined at the end of Chapter 4. This implementations are known as ThrRs, ThrMmt, LrMmt and LrrMmt.

6.1.1 ThrRs

The ThrRs method uses a threshold based VM allocation policy and a random VM selection policy. This is the most naive method that will be tested, as tasks are allocated to servers based on avoiding working above a certain capacity threshold (static CPU utilization thresholds). In addition to this, random machines are chosen to be migrated from over-utilised hosts to non-overloaded alternatives.

6.1.2 ThrMmt

The ThrMmt method once again uses a threshold based VM allocation policy but instead uses a minimum migration time VM selection policy. In this case, machines are chosen to be migrated from over-utilised hosts using a metric of lowest projected migration time. The idea is that jobs that take less time to move are prioritised for movement.

6.1.3 LrMmt

This method is prolific in literature for being a lightweight but powerful method for energy conservation. This uses local regression for VM allocation, which aims to predict server overuse using parametric methods. In this method, this is used alongside the minimum migration time VM selection policy.

6.1.4 LrrMmt

Finally, the LrrMmt method uses a robust version of the previously discussed local regression for VM allocation. This aims to provide a more robust host overload detection method than local regression. This methodology is defined by giving less weight to deviant influential observations in regression parameter calculations. Once again, minimum migration time is used as the accompanying VM selection function.

6.2 Experiment Design

Data is collected on all four methods by creating a new Java file in the org.cloudbus.cloudsim.examp
directory. All methods are placed into a for loop, in which the same random workload is shared by each method for every iteration. Minor changes were made to

the output method such that after each run has been executed, all of the statistics for that method are written onto the next line of a CSV file. These files are then imported into an R script for exploratory data analysis and visualisation.

6.3 Parameter Testing for Allocation Policies

In order to compare the performance for these methods fairly, it is important to identify the optimal parameter for each method. This is done using the design of the experiment above and varying parameter values over 1000 iterations of each method. Parameters for the Thr and Lr VM allocation methods were randomly chosen on each run from a discrete list: thresholds in the range of 0.4-0.9 for the Thr method and safety parameters in the range of 0.7-1.7 for the Lr methods. These ranges were chosen from initial tests on predefined PlanetLab workloads and are discretised by increments of 0.1. Finally, the optimal parameter for each method will be chosen as a result of low energy consumption, number of VM migrations and service level agreement violations. These operations are facilitated by the ParameterTesting.java file located in the org.cloudbus.cloudsim.examples.power.random folder.

6.3.1 Energy Consumption

The mean energy consumption of ThrRs and ThrMmt were found at each threshold value. The results of this analysis are documented in a grouped barplot, generated using the ggplot2 library.

The lowest power consumption can be seen at a threshold of 0.9 in both of the threshold based approaches. Generally, higher values of the threshold parameter yield lower levels of energy consumption. As expected, the ThrMmt implementation consumed less energy on average than the ThrRs method, due

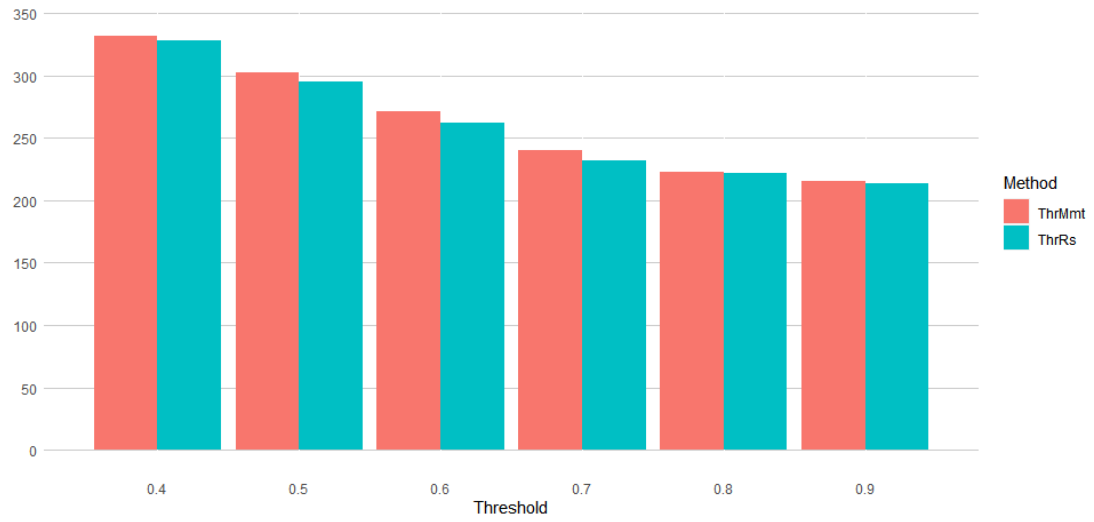

Figure 6.1: Mean energy consumption of threshold based methods grouped by parameter value

to the use of the minimum movement time metric over a naive selection.

Similarly, the mean energy consumption of LrMmt and LrrMmt at each safety parameter value has been found. As in the threshold based example, a visual aid is also provided to convey how the magnitude of energy usage changes depending on parameter value.

Both the LrMmt and LrrMmt methods share the very similar power consumption metrics at the each value of the safety parameter. It was expected that the LrrMmt method would perform slightly better, however the experimental conditions being of such a small scale may influence this result. From this analysis, it can be seen that lower values of the safety parameter generally associate with minimised mean energy consumption. Additionally, local regression methods generally have lower levels of power consumption than threshold based methods.

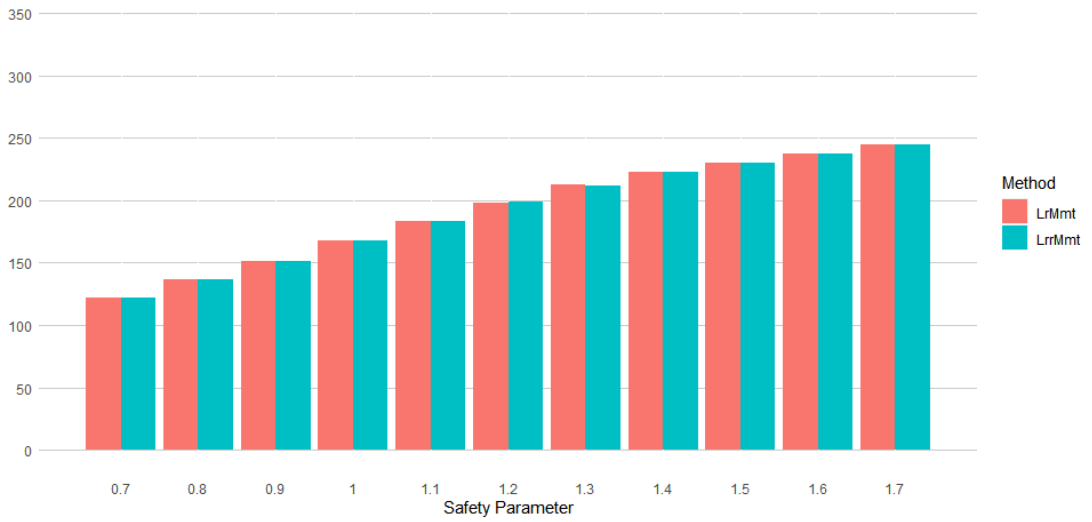

Figure 6.2: Mean energy consumption of local regression based methods grouped by parameter value

6.3.2 Virtual Machine Migrations

In this section, the mean number of virtual machine migrations will be explored for each parameter value. Considering the ThrRs and ThrMmt methods, we investigate the mean number of migrations seen at each threshold value.

As detailed by Figure 6.3, more extreme threshold parameters yield fewer VM migrations on average. The highest levels of VM migration are seen at threshold values of 0.6 and 0.7. These high values are encouraging, suggesting more effort to optimise resources. However, this may also suggest sub-optimal performance, as unwise VM migrations may be responsible for unnecessary increases in energy consumption. Overall, the ThrRs method makes fewer migrations than the ThrMmt counterpart.

Similar results have been plotted for the local regression based methods studied:

Again, extreme values of the safety parameter generally prove to have lower

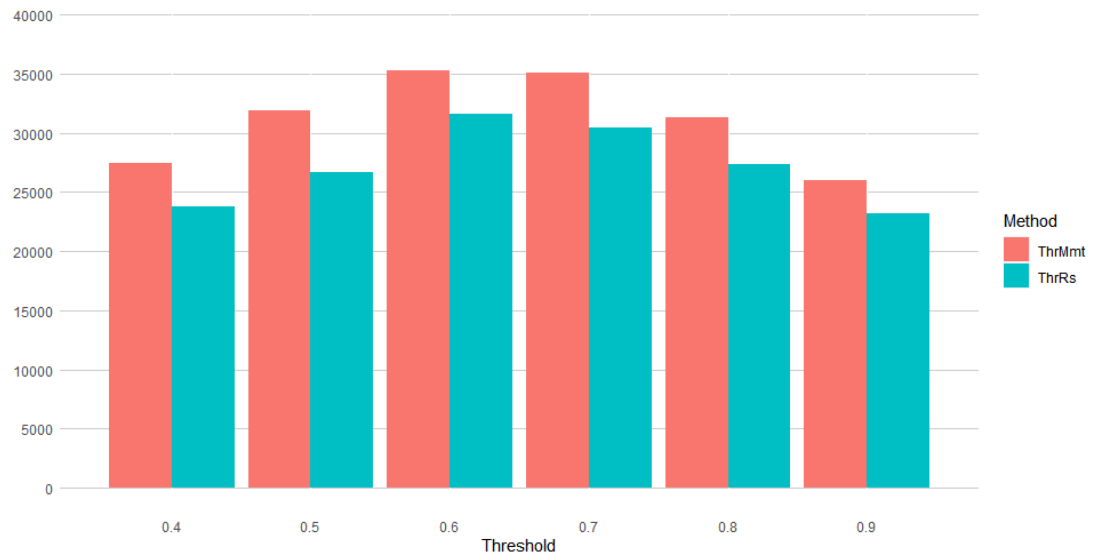

Figure 6.3: Mean number of VM migrations of threshold based methods grouped by parameter value

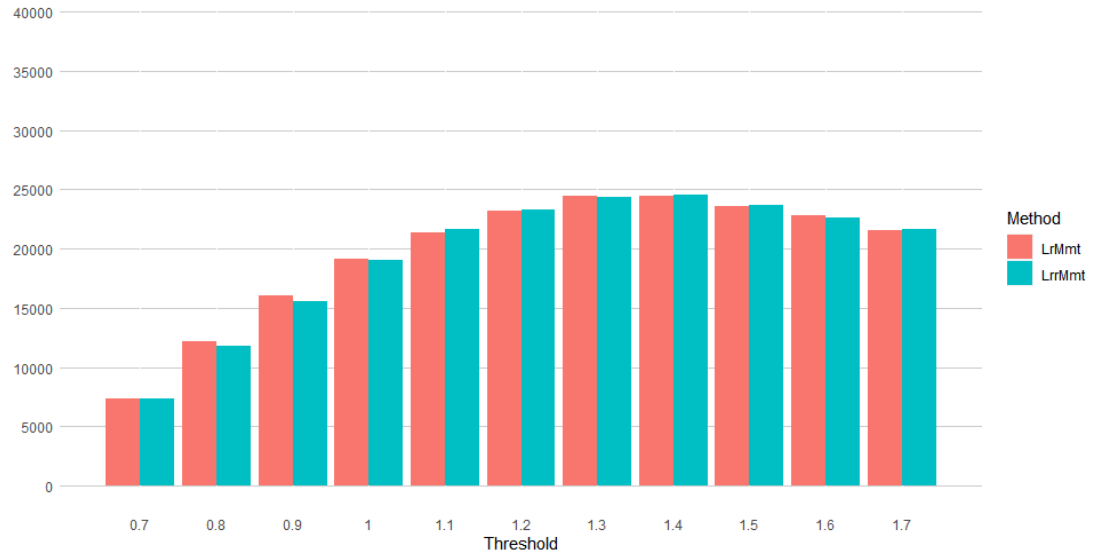

Figure 6.4: Mean number of VM migrations of local regression based methods grouped by parameter value

rates of migration. As was the case with the power consumption results, the data for the robust local regression method are quite similar to that of the simpler

local regression method. Generally, the local regression methods have far fewer migrations than threshold based methods. When coupled with results from the power consumption parameter testing, the studied local regression models tend to perform better than threshold based methods.

Because of the ambiguity of the efficacy of these migrations on an individual basis, this metric should hold the least weight on the decision process of the optimal parameters.

6.3.3 SLA Violations

In this section, the SLAV metric is considered. This combines the following metrics: performance degradation due to migration (PDM) and service level agreement violation time per active host (SLATAH). The parameter testing results can be seen for threshold based approaches in Figure 6.5.

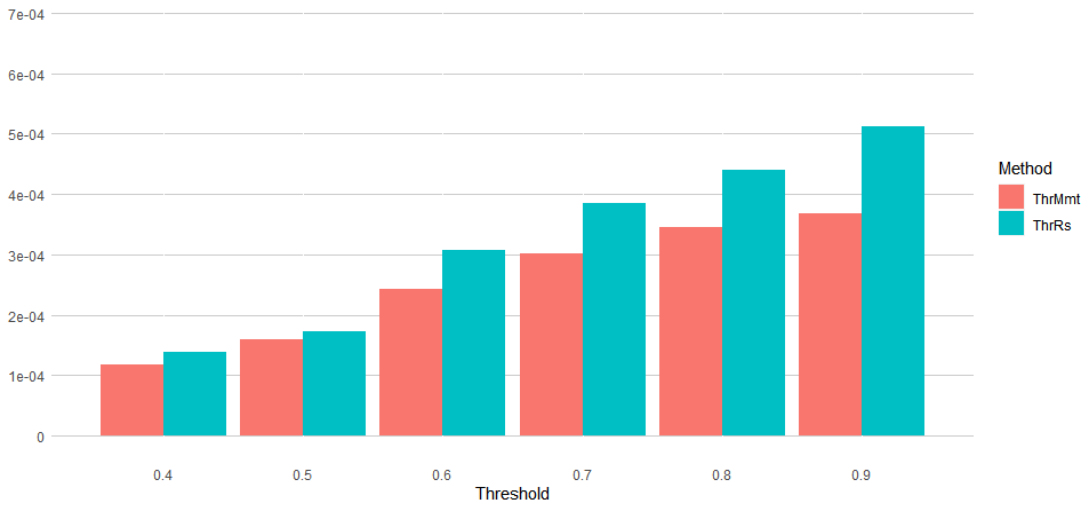

Figure 6.5: Mean SLAVs of threshold based methods grouped by parameter value

In both threshold based cases, the lowest assessed threshold parameter achieved the lowest proportion of SLAVs. This can quite simply be attributed to the fact

that rather than moving jobs for energy optimisation, each job is left to be completed to avoid violating any service agreements. The lower threshold would facilitate this.

These same metrics were recorded for the local regression based approaches. As with the previous examples, both methods yielded similar results across all parameters. These results are highlighted in Figure 6.6.

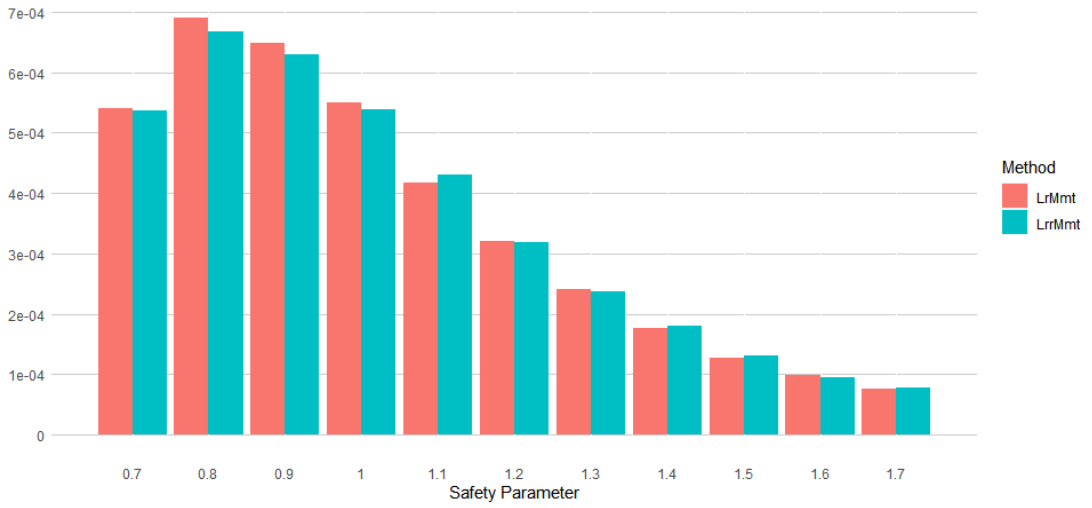

Figure 6.6: Mean SLAVs of local regression based methods grouped by parameter value

In both local regression based approaches, higher values of the safety threshold seemed to minimise the SLAVs in general. The highest parameter value tested optimised this metric. Thus, we can conclude that a safety parameter of 1.7 is optimal from the point of view of SLAV reduction.

6.3.4 Results

Considering the combination of the previous metrics, the optimal parameters in each case are not trivial. Depending on the priority of the application, we cannot

easily determine an option that minimises energy consumption and SLAVs.

As a result, a combination of these results must be considered to solve this multi-objective issue. One such method is to use a metric such as the one used in [28]. This simply uses the inverse of a product of energy consumption and SLAV to generate an overall performance metric. This metric will be used to choose the optimal parameter for the purposes of this study. The results of these calculations have been shown in Table 6.1 for threshold based methods and Table 6.2 for local regression methods.

Threshold	EE ThrRs	EE ThrMmt
0.4	22.0	25.6
0.5	19.6	20.8
0.6	12.4	15.2
0.7	11.2	13.8
0.8	10.3	13.0
0.9	9.14	12.6

Table 6.1: Energy efficiency calculations for threshold based approaches

Using the energy efficiency calculation as the primary performance metric, a threshold of 0.4 is chosen to be the optimal parameter for both the ThrRs and ThrMmt methods.

In the case of the local regression methods, the safety parameter of 1.7 was identified as the optimum from the set studied. In the following section, these algorithms, equipped with the parameters found, will be compared on the same workloads.

6.4 Comparative Analysis of Benchmarks

Now that the optimal parameters have been found, these will be fixed and used to compare overall performance over the course of 50 workloads. This will be done in

Safety Parameter	EE LrMmt	EE LrrMmt
0.7	15.1	15.3
0.8	10.6	11.0
0.9	10.2	10.5
1	10.9	11.1
1.1	13.1	12.7
1.2	15.7	15.8
1.3	19.5	19.8
1.4	25.4	24.8
1.5	34.3	33.2
1.6	43.1	44.3
1.7	53.1	52.1

Table 6.2: Energy efficiency calculations for local regression based approaches

order to determine the overall efficacy of each method in this application domain, given stochastic workloads. These operations are facilitated by the Comparison-Testing.java file in the org.cloudbus.cloudsim.examples.power.random folder.

6.4.1 Energy Consumption

Energy consumption metrics for all methods over the fifty workloads are shown in Figure 6.7.

Both local regression based methods have a significant advantage over their threshold based counterparts when it comes to energy efficiency. ThrRs consumes slightly less energy than the ThrMmt method, which may be attributed to the lack of heuristic used when choosing VMs for migration. However, LrMmt and LrrMmt have very similar performance throughout testing. Both methods are optimal for energy consumption at times, depending on workload. These methods show little to no deviation throughout testing, being consistent with energy consumption.

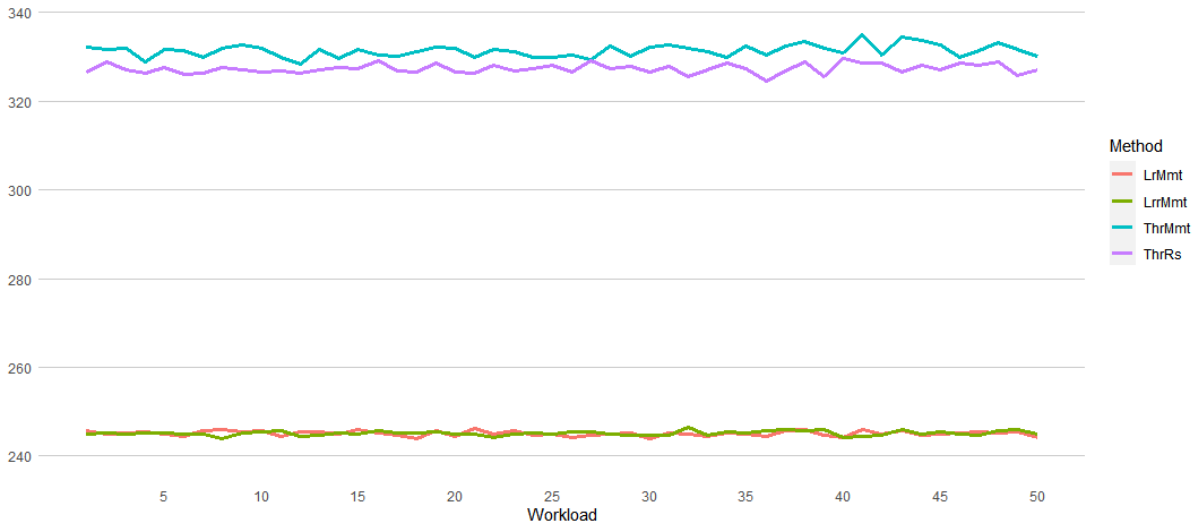

Figure 6.7: Energy consumption of all methods over fifty random workloads

6.4.2 Virtual Machine Migrations

Virtual machine migrations have also been considered for all methods over the test workloads. The results are shown in Figure 6.8.

Again, local regression based methods tend to outperform threshold based methods, yielding fewer migrations on every run. ThrMmt makes a lot more migrations than all other methods in general, which highlights issues in sub-optimal VM selection for migration. As in the case of the energy consumption metric, LrMmt and LrrMmt tend to be equally effective, with outperformance depending entirely on the workload considered.

6.4.3 SLA Violations

Finally, SLA violations have been compared for each method depending on workload. The results of this study can be seen in Figure 6.9.

In this case, ThrRs has the worst performance - regularly violating service

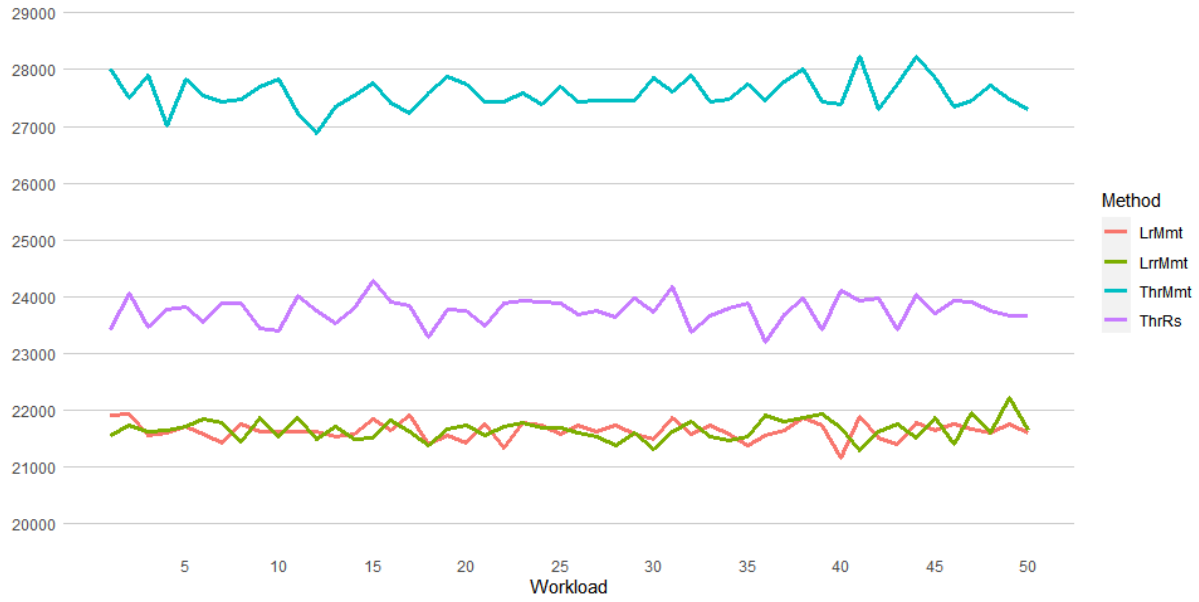

Figure 6.8: Virtual machine migration frequency of all methods over fifty random workloads

Figure 6.9: Service level agreement violations of all methods over fifty random workloads

level agreements in comparison to other methods. This can be attributed to the random selection element of this method, which does not account for jobs which may take longer to move in the VM selection process. Local regression based methods are superior in this case once again, with both methods yielding the lowest proportion of SLAVs depending on workload assigned.

6.4.4 Results

Considering the combination of the previous metrics, it is clear to see that local regression based methods regularly outperform threshold based methods. As expected, the use of heuristics for both VM selection and allocation contribute to the optimal performance of each method. Given that minimum movement time is an extremely simple heuristic which seems to make quite large improvements to performance overall, adding the dimension of simplistic machine learning algorithms should boost this performance further. In this case, the local regression method for VM allocation seems to be performing reasonably well and has made a massive improvement on threshold based approaches already. Therefore, improvements to this will not be the focus of this project.

Instead, the goal will be to provide a comparable method to LrMmt and LrrMmt, cutting down on power consumption, reducing the frequency at which virtual machines are migrated while avoiding the breach of service level agreements and providing an acceptable quality of service. This should be achievable by replacing the VM selection method with a smarter algorithm. The implementation of such a VM selection method will be detailed in the next chapter.

Chapter 7

Machine Learning Algorithm

In this chapter, the implementation of the selected machine learning algorithm will be discussed. Furthermore, the integration of the algorithm into the CloudSim architecture will be described. This integration process involved creating multiple classes to facilitate the learning mechanism, and the rework of the problem at hand into one which can harness the power of classification algorithms.

7.1 Algorithm Specifications

The first research question that will be explored is the identification of a machine algorithm which shows the most promise in the application domain. To answer this, it is important to determine the dataset we will use to train the algorithm.

7.1.1 Training Data

Training data for this problem has been derived from actions taken by two other VM selection policies - minimum migration time (Mmt) and minimum utilization (Mu). Both of these methods are heuristic based and metrics are extracted from each possible VM for migration for training. These metrics are the VM RAM,

CPU utilisation, Host utilisation and SLA violations. The decision taken whether to migrate the VM is used as the response variable in this case, and is encoded for binary classification. This training data consists of over 300000 cases, with balanced classes to optimise the training process.

7.1.2 Algorithm Choice

Given the nature of the data generated, a supervised learning approach will need to be deployed in order to learn effectively from the labelled dataset. The origin of the data is from two different VM selection policies, which use completely different heuristics to determine which VMs to migrate. As a result, it would not be wise to assume that the dataset is linearly seperable. This would count out any algorithms which can only approximate linear hypothesis functions. Flexibility in hypothesis function complexity is key in the choice of algorithm.

Considering the above carefully, the algorithm chosen to be implemented in this case is the multilayer perceptron. This is a simple but effective classifier for non-linearly separable data, and given the nature of the data available, this method seems to be fit for purpose. The implementation I will use is from the Weka machine learning library [29].

7.2 Multilayer Perceptron Algorithm

A multilayer perceptron (MLP) is, in essence, a fully connected feed-forward neural network. Composed of multiple layers of perceptrons, these networks consist of three or more layers - the input, output and hidden layers.

7.2.1 Structural Description

The input layer is made up of attributes which are considered for classification. The variable number of hidden layers consist of nodes, with values calculated by taking a function of some linear combination of node values from the previous layer. The function applied to this linear combination is known as the activation function, and maps the resulting value from the linear combination into a desired range. Finally, the output layer in this case produces a classification of the attributes. This is done using the same method as the hidden layers, combined with thresholding to determine the classification of given attributes.

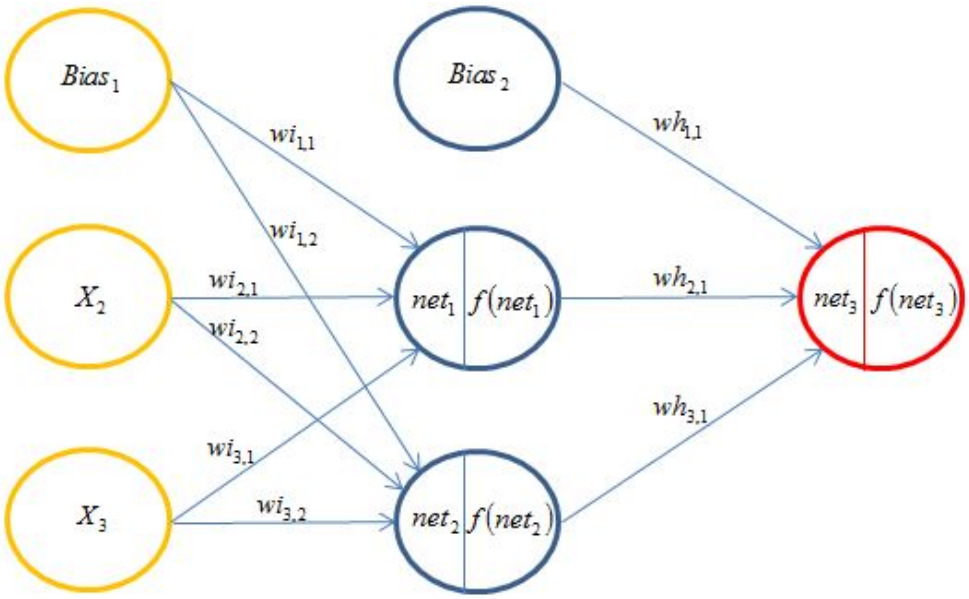

Figure 7.1: Schematic representation of multi-layer perceptron

In Figure 7.1, this structure can be soon more clearly. Here X_n are attributes and a bias term is used at each pre-output layer to shift the activation function by a constant each time. The net_j relate to linear combinations of the previous layer, for example:

$$net_1 = Bias_1 * wi_{1,1} + X_2 * wi_{2,1} + X_3 * wi_{3,1}$$

Finally, f is the activation function in this case, which calculates the node value for the next layer.

7.2.2 Training Phase

The training phase of the MLP involves tuning the multiplicative constants in the described linear combinations, known as weights. These are calculated on a rolling basis by using backpropogation. This method updates weights by adjusting them when a prediction is incorrect. This is done according to an update rule which is described in detail in Algorithm 1. It is important to note the following notation in this algorithm, which has been taken from [30].

- $W^l_{i,j}$: the weight associated with node j in layer $l-1$ into node i in layer l
- b^l_i: the bias into node i in layer l
- z^l_i: the linear combination of inputs to node i in the lth layer
- a^l_i: activation of ith node in the lth layer i.e. $f^l(z^l_i)$
- a^0_i: the attribute x_i and is used for generalisation of equations.
- $\hat{y}$: the prediction of the value of label y. $\hat{y} = a^L_1$, where L denotes the output layer.

7.3 Policy Registration

As discussed in Chapter 4, policies are stored in the org.cloudbus.cloudsim.power path. A newly defined policy of this type must extend and override the VmSelectionPolicy class. This policy must be registered so that it can be called by the RandomRunner function successfully, given string input. This registration is

Algorithm 1 Multi-layer perceptron training algorithm

Initialise hyperparameter values (learning rate etc.)
Initialize weights W and biases b with small arbitrary values
for maximum iterations or until convergence **do**
  Select one training case at random
  Calculate output for training case using the following formulae:
  $z_i^l = \sum_{j=1}^{n} W_{ij}^l a_j^{l-1} + b_i^l$
  $a_i^l = f^l(z_i^l)$
  Calculate cost function, where $\hat{y}$ is predicted label and y is actual:
  $J_{W,b}(y, \hat{y}) = -(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y}))$
  Propogate output errors back through the network.
  Output layer:
  $\Delta z_1^L = a_1^L - y_1$
  $\Delta W_{1,i}^L = \Delta z_1^L a_i^{L-1}$
  $\Delta b_1^L = \Delta z_1^L$
  Hidden layers: Loop backwards from layer $L - 1$ to 1.
  $\Delta z_i^l = f'(z_i^l) \sum_j (z_j^{l+1} W_{j,i}^{l+1})$
  $\Delta W_{j,i}^l = \Delta z_j^l a_i^{l-1}$
  $\Delta b_j^l = \Delta z_j^l$
  Update weights and biases with numerical derivatives calculated:
  $W_{j,i}^l - = \alpha \Delta W_{j,i}^l$
  $b_j^l - = \alpha \Delta b_j^l$
end for

achieved by adding it to the getVmSelectionPolicy method (located in the RunnerAbstract.java class), which takes the string representation of the policy and generates the desired VM selection object - representing the newly implemented selection policy.

7.4 New Classes

The following section aims to describe the purpose of all classes added in implementing this novel VM selection method. These classes were added to the CloudSim architecture to enable the use of the MLP in selecting virtual machines

to migrate.

7.4.1 PowerVmSelectionPolicyClassification

This class is responsible for both the training of the MLP and classification of new observations into migration and non-migration groups. As discussed previously, this method extends the PowerVmSelectionPolicy class and overrides the getVmToMigrate() method. This method is called every time a host is over-utilised, which makes this class pertinent to the success of this project. This is located in the org.cloudbus.cloudsim.power folder.

When this method is called, a set of migratable VMs are obtained and two new functions are called to get the host utilisation and SLA metrics for all migratable VMs. If no model has been trained, the training phase is entered. This imports a .arff file (used by Weka) of training data, trains an MLP model using user specified hyperparameters and saves the model to a .model file.

However, if a model has been trained, it is imported for use. The list of migratable VMs is looped through and a new Weka instance is set up for each VM. This instance contains values for all metrics considered in the training phase. The model is then called to classify whether the instance should be migrated or not. One such migratable VM is returned in each case, and if none are deemed to be migrated, this function returns null.

7.4.2 LrCl

This class uses the local regression policy for VM allocation, and the classification policy defined above for VM selection. As is the case with the LrMmt class, this file is located in the org.cloudbus.cloudsim.examples.power.random directory. This class is identical to the LrMmt class, in which the main method creates a RandomRunner object. The only difference is that the selection policy variable

is set to "Cl", which corresponds to the classification selection policy registered in the getVmSelectionPolicy function. This will then run the simulation on a random workload given the Lr allocation and Cl selection policies.

7.5 New Methods

Alongside these new classes, new methods have been added to existing classes in order to integrate the new VM selection process. These are briefly described below.

7.5.1 getHostUtil

This method is concerned with getting the host utilisation metrics for use in training the MLP. This calculates the total host utilisation by iterating through all VMs on the host and summing individual utilisation proportions. We then consider all migratable VMs separately and find the proportion of the host that each migratable VM is using. This method returns a list of these proportions, corresponding to each migratable VM. This method is defined in the PowerVmSelectionPolicyClassification file, however it is also used in the corresponding Mmt and Mu selection policy files for extraction of training data.

7.5.2 getSla

This method is used to get SLA metrics for use in training the multilayer perceptron. A list is initialised to store the SLA violation metric for each virtual machine. All migratable VMs are iterated through, with the historical totals for allocated mips and requested mips calculated. Finally, the following formula is used to calculate the SLA violation for each VM:

$$SLAV = \frac{R_{MIPS} - A_{MIPS}}{R_{MIPS}}$$

where R_{MIPS} is the total requested MIPS and A_{MIPS} is the total allocated MIPS.

A list of these SLA violation metrics are returned, corresponding to each migratable VM. This method is defined in the Runner file and is used in all PowerVMSelectionPolicy files to obtain the SLA atrribute values for classification.

Chapter 8

Parameter Testing

Now that an MLP has been integrated into a VM selection process, the second research question must be considered. This chapter will be concerned with finding the optimal hyperparameter settings for the selected algorithm. In order to determine which settings are optimal, the energy consumption, VM migration frequency and SLAV metrics discussed in Chapter 5 will be used.

8.1 Hyperparameters Considered

The Weka implementation of the MLP allows for the selection of the following hyperparameters [31]. Although there are more tunable parameters, the focus will be on the parameters described. Note that the Weka implementation does not permit the usage of weight optimisers such as Adam and RMSProp, instead defaulting to the gradient descent method. It also fixes the activation function as the sigmoid function. This will further be discussed in Chapter 10.

8.1.1 Learning Rate

This is set using the MultilayerPerceptron.setLearningRate() method. This sets the learning rate for the backpropogation algorithm. This learning rate is a multiplicative factor in weight update calculations and controls the influence which the current update will have on the parameter.

8.1.2 Momentum

This is set using the MultilayerPerceptron.setMomentum() method. This sets the momentum rate for the backpropogation algorithm. This is used in the gradient descent algorithm to drive gradients in the correct directions, contributing to faster convergence. This is a multiplicative factor applied to the previous weight to determine the influence the previous weight value has on the current update.

8.1.3 Training Time

Training time is set using the MultilayerPerceptron.setTrainingTime() method. This specifies the maximum number of training epochs to perform.

8.1.4 Hidden Layers

The number and configuration of hidden layers in the multilayer perceptron can be specified by using the MultilayerPerceptron.setHiddenLayers() method. This configuration is specified by a String object of integers seperated by commas. For example setHiddenLayers("4, 3") will make two hidden layers, with 4 and 3 nodes respectively.

8.2 Hyperparameter Tuning

In this section, the optimal configuration of hyperparameters will be found using combinations of the values below. Rather than trying all available combinations, the hidden layer configuration will be found first fixing learning rate, momentum and training time. All tests will be performed on the same workload which can be done by setting the workload generation random seed in RandomConstants.java. It is also important to note that the Lr VM allocation method will be used in each case, with the previously determined optimal parameter of 1.7 (see Chapter 6).

- Hidden Layer Configuration: {"2", "4", "8", "2,2", "4,2", "8,2", "2,2,2", "4,3,2"}
- Learning Rate: {0.01, 0.05, 0.1}
- Momentum: {0.99, 0.95, 0.9}
- Training Time: {500, 1000, 2000, 3000, 5000}

8.2.1 Hidden Layer Configuration

For this testing, we will fix the value of learning rate to be 0.01, momentum to be 0.99 and training time to be 2000. The results of this testing are detailed in Table 8.1. Here we can see that a single hidden layer of 8 nodes promotes the best energy efficiency, and has the lowest proportion of service level agreement violations with just 0.00663%. This configuration will be fixed for the remainder of the parameter tests.

Configuration	Energy	Migrations	SLAVs	Energy Eff.
2	241.35	20220	0.0000667	62.11934
4	240.7	19997	0.0000665	62.47442
8	240.57	20053	0.0000663	62.69675
2,2	241.53	20053	0.0000670	61.79511
4,2	235.33	19092	0.0000708	60.01910
8,4	242.51	20600	0.0000754	54.68888
2,2,2	242.24	20214	0.0000720	57.33524
4,3,2	231.33	18674	0.0000755	57.25601

Table 8.1: Parameter testing results for each hidden layer configuration considered

8.2.2 Learning Rate and Momentum

All combinations of learning rate and momentum listed above will be tested in this section. The training time will remain fixed at 2000 for the purposes of these tests. The results for each combination are shown in Table 8.2. Two results stand out from this test:

The case with a learning rate of 0.01 and momentum of 0.95 produces the lowest SLA value at 0.00599%. When combined with energy consumption, this metric combination produces the highest energy efficiency value of all combinations. The other case of interest is one with a learning rate of 0.05 and momentum of 0.99. This produces the lowest energy consumption value from all combinations tested of 229.13 kw/h. As expected, this has a number of migrations on the lower end and, in addition to this, the SLAV metric is also relatively low. Although this doesn't yield the best energy efficiency, it is important to consider that this is an arbitrary value calculated from more concrete metrics.

In this case, an exception will be made to considering just energy efficiency, as the other result seems to be generally better across the metrics considered. Therefore, a learning rate of 0.05 and momentum value of 0.99 will be fixed for the remainder of testing.

Learning Rate	Momentum	Energy	Migrations	SLAVs	Energy Eff.
0.01	0.99	240.57	20053	0.0000663	62.69675
0.01	0.95	238.65	19875	0.0000599	69.95387
0.01	0.9	235.80	19531	0.0000635	66.78554
0.05	0.99	229.13	18613	0.0000632	69.05593
0.05	0.95	239.15	19795	0.0000657	63.64499
0.05	0.9	241.53	19990	0.0000697	59.40133
0.1	0.99	231.04	18575	0.0000865	50.03763
0.1	0.95	241.09	20221	0.0000705	58.83445
0.1	0.9	239.67	19796	0.0000653	63.89592

Table 8.2: Parameter testing results for each combination of learning rate and momentum considered

8.2.3 Training Time

The final parameter that will be tuned is the training time. Fixing all of the other parameters with their optimal values, all metrics have been detailed for each training time in Table 8.3. The optimal training time has been found to be 3000 epochs.

Training Time	Energy	Migrations	SLAVs	Energy Eff.
500	224.14	17315	0.0000558	79.95515
1000	228.86	18656	0.0000692	63.14283
2000	229.13	18613	0.0000632	69.05593
3000	236.84	19391	0.0000523	80.73154
5000	242.45	19712	0.0000896	46.03306

Table 8.3: Parameter testing results for each training time considered

8.3 Chosen Hyperparameter Values

Finally, from the above hyperparameter tests, it can be concluded that the following set of hyperparameters can be used to successfully tune the MLP.

It is now important to test this configuration of parameters on a number of

Hyperparameter	Value
Configuration	"8"
Learning Rate	0.05
Momentum	0.99
Training Time	3000

Table 8.4: Final parameter values for multilayer perceptron algorithm

different workloads. This will be done in the following chapter and compared to the most successful benchmark.

Chapter 9

Comparison to LrMmt

This chapter will aim to answer the single remaining research question - whether the performance of our proposed VM selection algorithm is comparable to an industry benchmark. In Chapter 6, LrMmt was identified as a current industry standard which excels at the task of power aware virtual machine migration. The optimal novel classification model from the previous chapter has been saved and will be imported for these comparisons. In the following tests, the LrMmt and LrCl methods will both be run on 50 generated stochastic workloads. These will be analysed and compared across the usual metrics discussed so far in this thesis - energy consumption, virtual machine migrations and SLA violations.

9.1 Energy Consumption

Energy consumption metrics for both methods over the 50 randomly defined workloads are detailed in Figure 9.1. Clearly, the classification VM selection method outperforms the LrMmt method in terms of energy consumption across all 50 randomly generated workloads. To put this improvement into perspective, the LrCl method yields a total energy saving of 385.894 kw/h over the LrMmt

methods in the space of all workloads. This is an average saving of 7.71788 kw/h over the industry benchmark method per workload. A Welch two sample t-test confirms this improvement, giving a 95% confidence interval for improvement of LrCl over LrMmt of (7.522585, 7.913175).

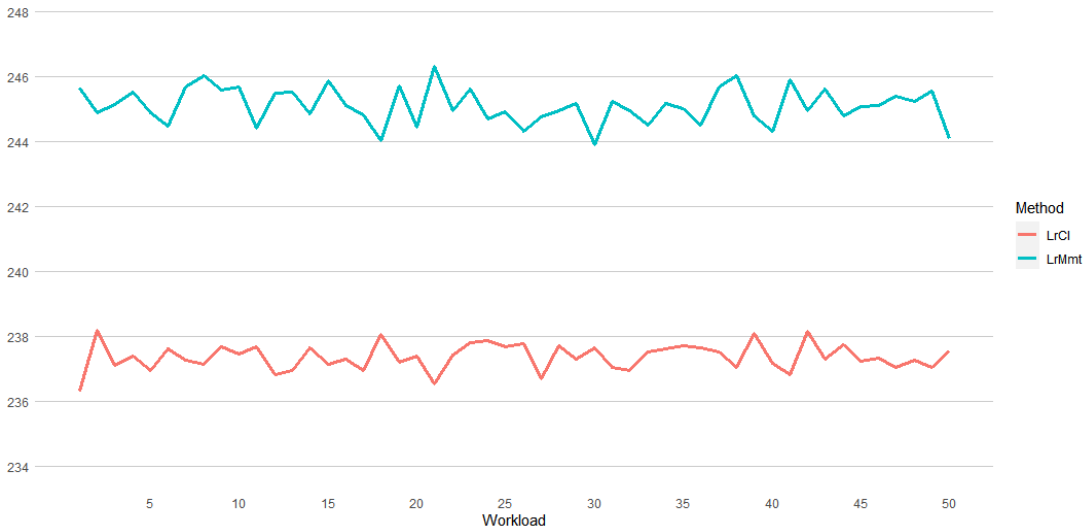

Figure 9.1: Energy consumption of both methods over fifty random workloads

9.2 Virtual Machine Migrations

Virtual machine migration frequency for both methods over the 50 randomly defined workloads are detailed in Figure 9.2. The classification VM selection method makes less migrations than the LrMmt method across all workloads considered. Given the lower energy consumption explored in the previous section, this suggests that the classification method implemented makes wiser migration choices overall. On average, LrCl makes 2023.16 fewer migrations per workload, with the Welch two sample t-test yielding a confidence interval of 1959.541 to 2086.779 less migrations on average.

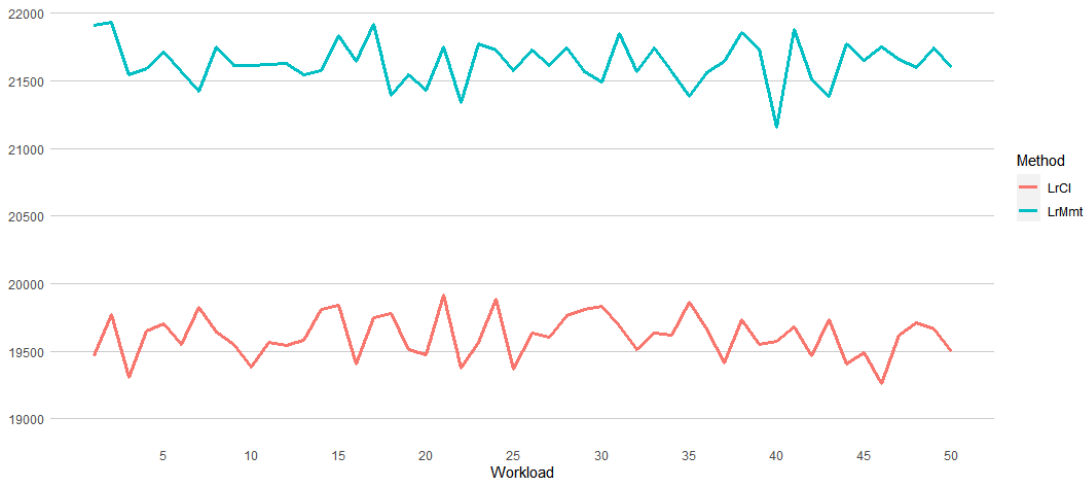

Figure 9.2: Virtual machine migrations of both methods over fifty random workloads

9.3 SLA Violations

When optimising energy consumption, it is very important that a low level of SLA violations are incurred. This is perhaps the most important metric to maintain in terms of operating to cloud service standards for clients. SLA violation metrics over the 50 workloads for each method can be seen in Figure 9.3. Again, the classification VM selection method outperforms the LrMmt method across all generated workloads. The LrCl method is responsible for a 0.111% total reduction in SLAVs, averaging a reduction of 0.00222% per run. These improvements seem numerically small, but the magnitude of these improvements can be visualised below. Once again, the Welch two sample t-test proves that there is a significant statistical improvement made by the proposed algorithm.

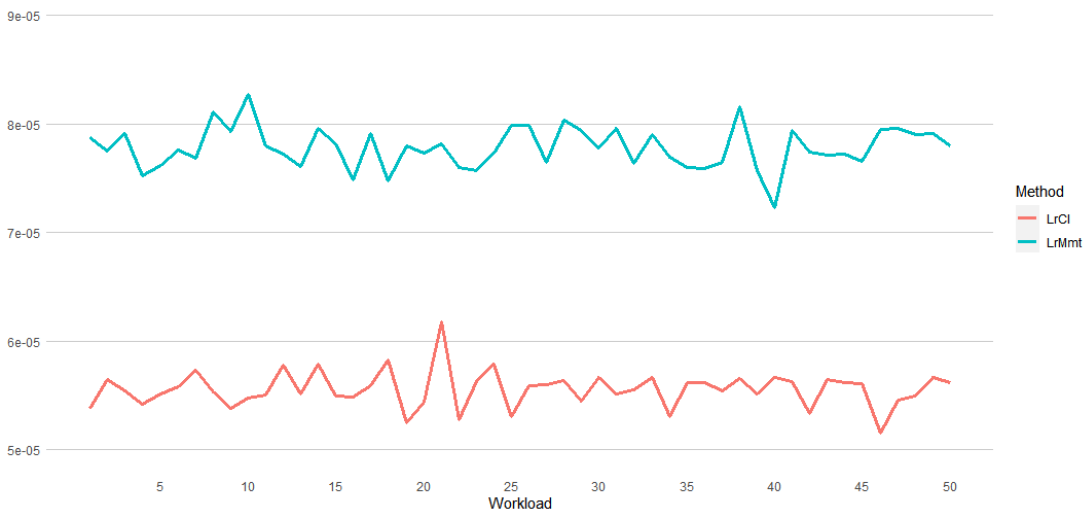

Figure 9.3: Service level agreement violations of both methods over fifty random workloads

9.4 Combination of Metrics

Considering the improvements made by LrCl over LrMmt in the above metrics, improvement in the energy efficiency metric is expected. This can be seen over each workload in Figure 9.4. As suggested by the graph, LrCl outperforms the LrMmt algorithm in every case considered. Since the interpretability of this metric is not as concrete as those already considered, the only analysis that will be provided in this case is the result of the two sample t-test. This returns a 95% confidence interval of (22.64933, 24.14659) for the mean improvement of LrCl over LrMmt.

9.5 Results

This chapter has consolidated the efficacy of using an MLP classifier in selecting VMs for migration. The success of this implementation has been shown against

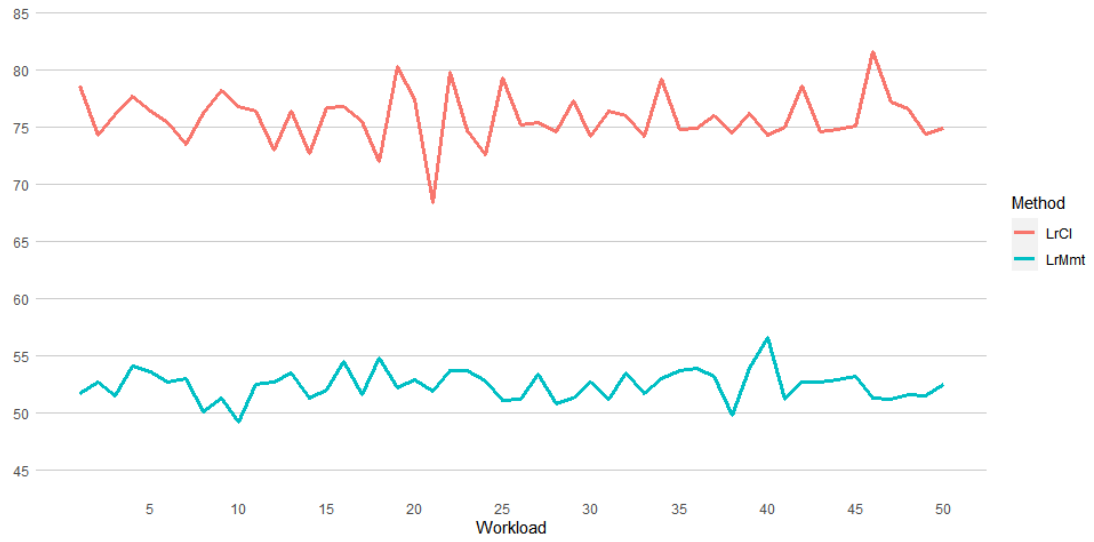

Figure 9.4: Energy efficiency of both methods over fifty random workloads

the state-of-the-art LrMmt method, making significant improvements across all metrics considered on all workloads generated. Figure 9.5 details the percentage improvement of this method over LrMmt across the energy consumption, SLA violations and energy efficiency metrics. This provides a clear answer to our final research question - the novel VM selection method implemented can not only perform comparably to the state-of-the-art LrMmt method, but it garners significant improvements. This method steadily has a lower energy consumption across all workloads - decreasing energy consumption by an average of 3.14821%. Simultaneously, it also reduces SLA violations by an average of 28.53163% over the LrMmt method. This suggests a much more effective migration process which is service-centric. This improvement is significant in terms of effective data centre management. When coupled with energy savings, this seems like a very viable option to use for a more sustainable, service-optimised data centre.

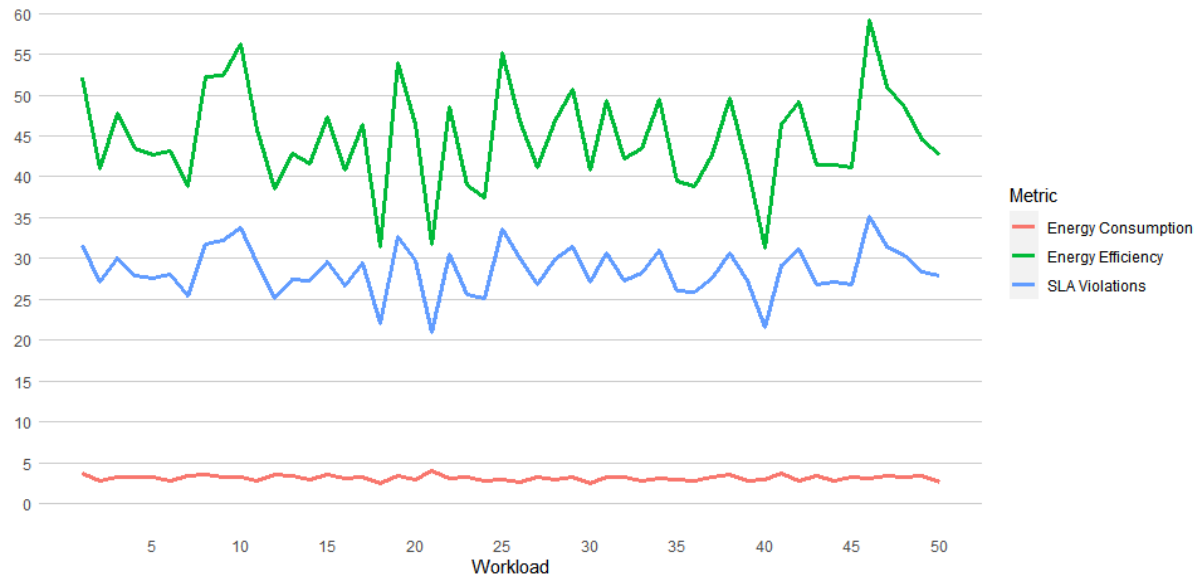

Figure 9.5: Percentage improvement of LrCl over LrMmt in each performance metric

Chapter 10

Conclusion

The sharp rise in demand for cloud based services has brought with it a major necessity for more cloud computing infrastructure. This has gained negative media attention in recent years due to the global climate change crisis. Data centres have been flagged as major power consumers on the total Irish power grid, consuming 14% of the entirety of the national metered electricity [3]. Forecasts predict that by 2030, given that current trends are followed, 30% of all Irish electricity consumption could be attributed to data centres [5]. Therefore, reducing this level of energy consumption while still maintaining high working standards in cloud computing centres is a very lucrative and worthy endeavour.

This thesis set out to develop a novel algorithm which primarily aims to reduce energy consumption in data centres. Harnessing machine learning in order to allocate resources more wisely, the goal of this research was to produce a VM selection policy which rivals industry standards. Taking inspiration from the state-of-the-art LrMmt algorithm discussed in [20], particular attention was paid to optimising which machine is chosen to migrate, identifying that using just minimum migration time may be limiting the potential of selecting an optimal machine for migration. While the work undertaken in [26] achieves a sizeable

improvement on this method by using reinforcement learning, this project aims to simplify this procedure by using a more explainable and lightweight model. This has been done successfully and produces results that are still viable when compared to the state-of-the-art.

10.1 Research Questions

In undertaking this research, three research questions were defined which were paramount in the development of this solution:

- RQ1: Which machine learning algorithm shows the most promise in this field of application? Can this algorithm integrate into the virtual machine selection process?
- RQ2: What are the optimal hyperparameter settings for this problem, given our choice of machine learning algorithm? What metric is most appropriate to measure the performance of our algorithm in this case?
- RQ3: Is the performance of our algorithm comparable to the state of the art benchmarks employed by industry?

10.1.1 RQ1: The Algorithm

This question was considered in Chapter 7 of this thesis. In the realisation of a solution for this project, virtual machines selected for migration by two heuristic-based methods formed basis as the training data. In addition to this, more metrics were extracted (host utilisation and a running total of SLA violations) to provide more features to aid in classification. It became apparent that this data may not be linearly separable due to the nature of this formulated training set, so it was important to select an algorithm that would be robust in learning more complex

hypothesis functions. As a result, a multilayer perceptron neural network was selected for this task.

This algorithm was quite simple to integrate into the CloudSim environment through the use of abstract classes and available methods to override. The entirety of the algorithm is executable in the PowerVmSelectionPolicyClassification.java. The Weka implementation of an MLP is used in this project seamlessly. The training phase is entered using a boolean flag and the model configuration is saved for later use. An existing model is imported and used to classify whether a VM should be migrated given a set of extracted attributes. One such machine is chosen to be migrated, which is returned by the getVmToMigrate() method and is integrated into the entire allocation-selection process.

10.1.2 RQ2: Hyperparameters and Performance Metrics

Chapter 8 was concerned with finding a solution to the hyperparameter optimisation problem. An intensive hyperparameter testing phase was performed as part of this project to tune the algorithm to perform the given task optimally. As part of this, 8 different shallow neural network configurations were tested, of which a network with one eight-node hidden layer was selected. A grid of 9 combinations of learning rate and momentum were considered which yielded an optimal learning rate of 0.05 with momentum of 0.99. Finally, training time was considered and the model achieved optimal performance at 3000 epochs. The optimal hyperparameter settings were chosen by using the metrics described below.

Performance metrics were defined and selected in Chapter 5 of this thesis. It was noted from the extensive literature review carried out that performance in this domain is characterised by three main metrics. Energy consumption is considered as its minimisation is the main focus of the project. However we must adhere to service agreements so SLA violations were also used as a measurement of

success. Finally, migration frequency is usually considered in tandem with energy consumption, as less migrations associated with low energy consumption would suggest intelligent migrations are executed. Finally a combination of the above metrics used in [28] is used to quantify energy efficiency and acts as a single overall metric over which policies are compared. Generally speaking, a combination of these metrics is most appropriate to measure algorithm performance here.

10.1.3 RQ3: Comparison to State-Of-The-Art

Comparisons were drawn between this method and the state-of-the-art LrMmt method in Chapter 9. Both algorithms were compared over fifty randomly generated workloads and compared across the selected metrics. Significant improvements could immediately be seen with all metrics through the accompanying visualisations. In summary, the LrCl method reduces energy consumption by 3.15% on average while also managing to reduce SLA violations by a massive 28.53%. In addition to these metrics, virtual machine migrations were reduced across the board, which suggests a more prudent VM migration policy. This favourable comparison to the widely regarded state-of-the-art LrMmt algorithm clearly demonstrates the potential that machine learning has in this domain.

10.2 Limitations

Although this project has been successful in harnessing machine learning to promote energy consumption reduction in data centres, there are some limitations of the software being used, particularly with the Weka package and CloudSim calculations.

10.2.1 Weka Limitations

The MultiLayerPerceptron package in Weka provides the opportunity to tune many hyperparameters. However, it does not allow the user to specify the activation function to be used in the implementation. Furthermore, it does not support the use of optimisers such as Adam and RMSProp, instead relying entirely on an implementation of gradient descent for convergence. This Weka package does not currently support the initialisation of small, random network weights, instead using specific seeded weights from the offset. This results in a more deterministic training process, which is not ideal for retraining to ascertain the best model. This may have limited the performance of the MLP in this instance, so there may be future work in implementing an MLP from a different package for this use.

10.2.2 CloudSim Calculations

One major flaw with the CloudSim system in this case is that it does not take runtime of VM selection/allocation policies into account. No term has been added for latency in time based calculations, instead assuming that virtual machines are selected to migrate and migrated instantly every 300 seconds. Because this latency is not accounted for, SLA violation and energy consumption calculations may not be totally indicative of real-life results.

10.3 Future Work

Considering the limitations in the implementation in this project, there may be some workarounds available. For example, using an alternative machine learning package such as DeepLearning4j may prove to yield better results. Another possible workaround would involve developing activation functions and/or opti-

misation functions which interface with the existing Weka library. In addition to these options, it may be worth considering machine learning for the task of VM placement. Reinforcement learning has been shown to be effective for this task in Shaw et al. [27], and therefore it may be a worthwhile consideration to reframe the problem as a machine learning problem. This can be motivated to combat issues of latency due to computation time and explainability. Furthermore, it may be possible to integrate the VM placement process into the existing machine learning algorithm - creating a hybrid VM selection and placement policy. There is great scope for a hybrid method of this nature to yield even better results than the current isolated methods. For example, feedback loops may be used for the co-optimisation of policies.

A final alternative suggestion is to fine tune the current algorithm depending on the results desired. For example, the optimisation of hyperparameters in this case was facilitated mostly through the energy efficiency metric defined in [28]. Throughout this study, this metric skewed our optimisation technique to minimise SLA violations moreso than energy consumption, due to the imbalance of units used in the calculation. It may be worth investigating a different measure that balances the importance of these two metrics. This may be used to tune a neural network that finds an optimisation balance between energy consumption and SLA violations.

References

[1] K. Feng, W. Lu, S. Chen, S. Wang, B. Yang, C. Sun, and Y. Wang, "Embedding ensemble learning into simulation-based optimisation: A learning-based optimisation approach for construction planning," *Engineering Construction Architectural Management*, vol. ahead-of-print, 07 2021. viii, 7, 9

[2] I. Spiceworks, "Public cloud vs private cloud (and hybrid cloud too)." [Online]. Available: https://community.spiceworks.com/cloud/articles/2501-public-cloud-vs-private-cloud-and-hybrid-cloud-too viii, 10

[3] S. Carswell, "Data centres now consuming more electricity than rural homes - CSO," *The Irish Times*, May 2022. viii, 12, 13, 68

[4] 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*, vol. 41, no. 1, pp. 23–50, 2011. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.995 viii, 25, 26

[5] P. Hoare, "Warning that data centres could account for 30% of all electricity consumption by 2030," *Irish Examiner*, Jan 2022. 1, 68

[6] N. Sadashiv and S. M. D. Kumar, "Cluster, grid and cloud computing: A

detailed comparison," in *2011 6th International Conference on Computer Science Education (ICCSE)*, 2011, pp. 477–482. 5, 6

[7] R. Buyya, C. S. Yeo, and S. Venugopal, "Market-oriented cloud computing: Vision, hype, and reality for delivering IT services as computing utilities," in *2008 10th IEEE International Conference on High Performance Computing and Communications*. IEEE, sep 2008. [Online]. Available: https://doi.org/10.1109%2Fhpcc.2008.172 7

[8] S. Marston, Z. Li, S. Bandyopadhyay, J. Zhang, and A. Ghalsasi, "Cloud computing — the business perspective," *Decision Support Systems*, vol. 51, pp. 176–189, 04 2011. 8

[9] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A View of Cloud Computing," *Commun. ACM*, vol. 53, pp. 50–58, 04 2010. 8

[10] L. Youseff, M. Butrico, and D. Da Silva, "Toward a unified ontology of cloud computing," in *2008 Grid Computing Environments Workshop*, 2008, pp. 1–10. 9

[11] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud Computing: State-of-the-art and Research Challenges," *Journal of Internet Services and Applications*, vol. 1, pp. 7–18, 05 2010. 9, 10

[12] A. Nag, "Cloud Computing: A Paradigm Shift in IT Infrastructure," *CSI Communication*, vol. 38, p. 8, 01 2015. 11, 12

[13] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997. 14

[14] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach (4th*

Edition). Pearson, 2020. [Online]. Available: http://aima.cs.berkeley.edu/ 14

[15] A. Geron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition.* 15

[16] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015. 15

[17] Y.-C. Lee and A. Zomaya, "Energy efficient utilization of resources in cloud computing systems," *The Journal of Supercomputing*, vol. 60, pp. 268–280, 05 2010. 18

[18] S. Srikantaiah, A. Kansal, and F. Zhao, "Energy-Aware Consolidation for Cloud Computing," *Cluster Computing - CLUSTER*, vol. 12, pp. 10–10, 11 2008. 18

[19] 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*, vol. 24, no. 13, pp. 1397–1420, 2012. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.1867 19

[20] 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*, vol. 28, no. 5, pp. 755–768, 2012, special Section: Energy efficiency in large-scale distributed systems. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X11000689 20, 68

[21] G. Portaluri, S. Giordano, D. Kliazovich, and B. Dorronsoro, "A power efficient genetic algorithm for resource allocation in cloud computing data centers," in *2014 IEEE 3rd International Conference on Cloud Networking (CloudNet)*, 2014, pp. 58–63. 21

[22] S. E. Dashti and A. M. Rahmani, "Dynamic VMs placement for energy efficiency by PSO in cloud computing," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 28, no. 1-2, pp. 97–112, 2016. [Online]. Available: https://doi.org/10.1080/0952813X.2015.1020519 21

[23] J. L. Berral, I. n. Goiri, R. Nou, F. Julià, J. Guitart, R. Gavaldà, and J. Torres, "Towards energy-aware scheduling in data centers using machine learning," in *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*, ser. e-Energy '10. New York, NY, USA: Association for Computing Machinery, 2010, p. 215–224. [Online]. Available: https://doi.org/10.1145/1791314.1791349 21

[24] G. Wei, A. V. Vasilakos, Y. Zheng, and N. Xiong, "A game-theoretic method of fair resource allocation for cloud computing services," *J. Supercomput.*, vol. 54, no. 2, p. 252–269, nov 2010. [Online]. Available: https://doi.org/10.1007/s11227-009-0318-1 22

[25] R. Das, J. Kephart, C. Lefurgy, G. Tesauro, D. Levine, and H. Chan, "Autonomic multi-agent management of power and performance in data centers," vol. 3, 01 2008, pp. 107–114. 22

[26] M. Duggan, K. Flesk, J. Duggan, E. Howley, and E. Barrett, "A reinforcement learning approach for dynamic selection of virtual machines in cloud data centres," in *2016 Sixth International Conference on Innovative Computing Technology (INTECH)*, 2016, pp. 92–97. 23, 24, 68

[27] R. Shaw, E. Howley, and E. Barrett, "An advanced reinforcement learning approach for energy-aware virtual machine consolidation in cloud data centers," in *2017 12th International Conference for Internet Technology and Secured Transactions (ICITST)*, 2017, pp. 61–66. 23, 73

[28] Z. Zhou, J. Abawajy, M. Chowdhury, Z. Hu, K. Li, H. Cheng, A. A. Alelaiwi, and F. Li, "Minimizing SLA Violation and Power Consumption in Cloud Data Centers Using Adaptive Energy-Aware Algorithms," *Future Gener. Comput. Syst.*, vol. 86, no. C, p. 836–850, sep 2018. [Online]. Available: https://doi.org/10.1016/j.future.2017.07.048 34, 43, 71, 73

[29] I. W. Eibe, I. H. Witten, E. Frank, L. Trigg, M. Hall, G. Holmes, and S. J. Cunningham, "Weka: Practical Machine Learning Tools and Techniques with Java Implementations," in *Proc ICONIP/ ANZIIS/ANNES99 Future Directions for Intelligent Systems and Information Sciences*. Morgan Kaufmann, 1999, pp. 192–196. 49

[30] M. Madden, "Lecture notes in CT5133 - Deep Learning," February 2022. 51

[31] "Weka.Classifiers.Functions.MultiLayerPerceptron," Jan 2022. [Online]. Available: https://weka.sourceforge.io/doc.dev/weka/classifiers/functions/MultilayerPerceptron.html 56

Appendix A

Project Code

The code for this project is available at the below link. This includes all Java files developed which are described in the project, and all R scripts used for data analysis.

https://github.com/deanconnell1999/masters-project