AN ENERGY-AWARE REAL-TIME VM-PROVISIONING FRAMEWORK FOR HETEROGENEOUS DATA CENTRES

by

Salam Ismaeel

B.S. Control and Computer Eng., University of Technology, Iraq, 1995

M.S. Control Eng., University of Technology, Iraq, 1998

PhD Computer Eng., Al-Nahrain University, Iraq, 2003

A Dissertation

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

in the program of

Computer Science

Toronto, Ontario, Canada, 2020

AUTHOR'S DECLARATION FOR ELECTRONIC SUBMISSION OF A DISSERTATION

I hereby declare that I am the sole author of this dissertation. This is a true copy of the dissertation, including any required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this dissertation to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this dissertation by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my dissertation may be made electronically available to the public.

Abstract

An Energy-Aware Real-Time VM-Provisioning Framework for Heterogeneous Data Centres

Salam Ismaeel, 2020

Doctor of Philosophy

Computer Science

Ryerson University

Increasing power efficiency is one of the most important operational factors for any data centre providers. In this context, one of the most useful approaches is to reduce the number of utilized Physical Machines (PMs) through optimal distribution and re-allocation of Virtual Machines (VMs) without affecting the Quality of Service (QoS). Dynamic VMs provisioning makes use of monitoring tools, historical data, prediction techniques, as well as placement algorithms to improve VMs allocation and migration. Consequently, the efficiency of the data centre energy consumption increases.

In this thesis, we propose an efficient real-time dynamic provisioning framework to reduce energy in heterogeneous data centres. This framework consists of an efficient workload pre-processing, systematic VMs clustering, a multivariate prediction, and an optimal Virtual Machine Placement (VMP) algorithm. Additionally, it takes into consideration VM and user behaviours along with the existing state of PMs. The proposed framework consists of a pipeline successive subsystems. These subsystems could be used separately or combined to improve accuracy, efficiency, and speed of workload clustering, prediction and provisioning purposes.

The pre-processing and clustering subsystems uses current state and historical workload data to create efficient VMs clusters. Efficient VMs clustering include less consumption re-

sources, faster computing and improved accuracy. A modified multivariate Extreme Learning Machine (ELM)-based predictor is used to forecast the number of VMs in each cluster for the subsequent period. The prediction subsystem takes users' behaviour into consideration to exclude unpredictable VMs requests.

The placement subsystem is a multi-objective placement algorithm based on a novel Machine Condition Index (MCI). MCI represents a group of weighted components that is inclusive of data centre network, PMs, storage, power system and facilities used in any data centre. In this study it will be used to measure the extent to which PM is deemed suitable for handling the new and/or consolidated VM in large scale heterogeneous data centres. It is an efficient tool for comparing server energy consumption used to augment the efficiency and manageability of data centre resources.

The proposed framework components separately are tested and evaluated with both synthetic and realistic data traces. Simulation results show that proposed subsystems can achieve efficient results as compared to existing algorithms.

Acknowledgments

I would like to express my heartfelt gratitude and appreciation to my supervisor Professor Ali Miri for his kindness and immense patience throughout the course of this academic journey. This work could not have been concluded without his ceaseless guidance, support, and time. It is his belief in me throughout the challenging times during my research and my graduation studies that continues to keep me motivated, even to this day.

I also owe a debt of gratitude to the thesis committee for generously offering their time, guidance, and goodwill throughout the entire process of my PhD defense.

I would like to take this opportunity to thank my friends along with all members of the lab mates family research group for their consistent encouragement.

I dedicated this thesis to my father who passed away while this work was being undertaken; my interest and knowledge is his valued heritage; to my mother, who is a model of generous disposition and kindness; to my wife, who is an embodiment of love, support and enormous patience; as well as my sons and daughters, who are models of a promising future.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ACO	Ant Colony Optimization
AHP	Analytic Hierarchy Process
AIC	Akaike Information Criterion
ANFIS	Adaptive Neuro-Fuzzy Inference System
ANN	Artificial Neural Network
ANP	Analytic Network Process
AR	Auto-regressive
ARIMA	Auto-regressive Integrated Moving Average
ART	Association Rules Technique
BF	Best Fit
BFD	Best Fit Decreasing
BIC	Bayesian Information Criterion
BIP	Binary Integer Programming
CFS	Constant Fixed Selection
CMS	Cloud Management System
CPU	Central Processing Unit
CV	Cross Validation
DCIM	Data Center Infrastructure Manager
DFQL	Dynamic Fuzzy Q-learning
DMA	Dynamic Management Algorithm
DVFS	Dynamic Voltage and Frequency Scaling
DWNN	Weighted Nearest Neighbors for Differenced data

ECU	Elastic Compute Unit
EC2	Amazon Elastic Computer Cloud
ELM	Extreme Learning Machine
EM	Expectation Maximization
EPA	Environmental Protection Agency
ESM	Exponential Smoothing Model
FCFS	First-Come First-Served
FCM	Fuzzy c-means
FEC	Fuzzy Expert System
FF	First Fit
FFD	First Fit Decreasing
FQL	Fuzzy Q-Learning
FS	Fuzzy System
FWP	Frequent Workload Pattern
GA	Genetic Algorithm
GFM	Gray Forecasting Model
GMM	Gaussian Mixture Model
HGA	Hybrid Genetic Algorithm
HMM	Hidden Markov Model
HTP	History Table Predictor
IaaS	Infrastructure as a Service
KNN	k-nearest Neighbor
LP	Linear programming
LR	Linear Regression

LTI	Linear Time-invariant
MA	Moving Average
MAD	Median Absolute Deviation
Mbps	Megabit per second
MB	Megabyte
MC	Maximum Correlation
MCI	Machine Condition Index
MCDM	Multi-Criteria Decision Making
MDS	Multidimensional Scaling
MIPS	Million Instructions per Second
ML	Machine Learning
MMT	Minimum Migration Time
NIC	Network Interface Card
NN	Neural Network
NP-hard	Non-deterministic polynomial
PaaS	Platform as a Service
PABFD	Power Aware Best Fit Decreasing
PALB	Power Aware Load Balancing
PCA	Principal Component Analysis
PM	Physical Machine
QoS	Quality of Service
RASA	Resource Aware Scheduling Algorithm
RBF	Radial Basis Function
RMSE	Root Meas Square Error

SaaS	Software as a Service
SLA	Service Level Agreement
SMM	Statistical Metric Model
SOM	Self-organizing Map
SSD	Sum of Squared Distances
SVM	Support Vector Machine
SWF	Stochastic Wiener Filter
SWM	Sliding Window Method
TPPC	Two Phases Power Convergence
VCPU	Virtual CPU
VL2	Virtual Layer 2
VM	Virtual Machine
VMM	VM Monitor
VMP	Virtual Machine Placement
VN	Virtual Network
WNN	Weighted Nearest Neighbors

Chapter 1

Introduction

Several energy minimization strategies can be used in data centres, but the most important of them is done by switching off unused Physical Machines (PMs). Unused PMs are obtained by an optimal distribution or after reallocating Virtual Machines (VMs) on the selected server.

In order to perform an optimal VM distribution/consolidation under Quality of Service (QoS), it is necessary for energy consumption constraints to undertake the implementation. In this thesis, we have built an energy-aware VM-provisioning framework that combines several subsystems. These subsystems including pre-processing, clustering, prediction, and placement, among others. The proposed framework also takes into considerations data provided by monitoring tools, historical data, user behaviour and server states in order to improve the proposed subsystems operations.

In this chapter, we introduce the importance of energy consumption in data centres. The chapter also include problem statement and contributions of the thesis.

1.1 Context and Motivation

In the realm of cloud computing, the IT and business resources, such as servers, storage, network, applications and processes can be dynamically provisioned to meet the users' needs and workloads. Cloud deployment models of public, private, hybrid or community describes how the cloud service are consumed. Public cloud where service provider makes resources available to the general public over the Internet, like Amazon EC2, Google Compute Engine, HP Cloud, Rackspace, and Windows Azure.

Service application delivered to end users is often referred to as Software as a Service (SaaS), Platform as a Service (PaaS), or Infrastructure as a Service (IaaS), depending on which level of the software stack is provided. In terms of cloud computing and especially Infrastructure-as-a-Service (IaaS), it is possible for users or companies to purchase services such as any utility. As a matter of fact, IaaS has become an effective solution for providing a flexible, on-demand and dynamically scalable computing infrastructure for several applications [1].

Much like distributed computing, in cloud computing applications run over multiple computers connected by a network. This network is often a data centre network. Each Physical Machines (PM) in the data centre can be used to hold one or many Virtual Machines (VMs), which are essentially virtualized environments on PMs with predetermined virtual resources. These resources may include CPU, memory, storage, bandwidth, VM configured, operating system, middle-ware, and one or more application programs [2]. The VMs can be categorized on the basis of their relationship with the PMs into: *System VM* and *Process VM*. The system VM is a software implementation of a VM that executes programs such as any PM. Such VM necessitates an operating system referred to as Guest Operation System. The process VM is an application, task or a user program that can be installed and run on

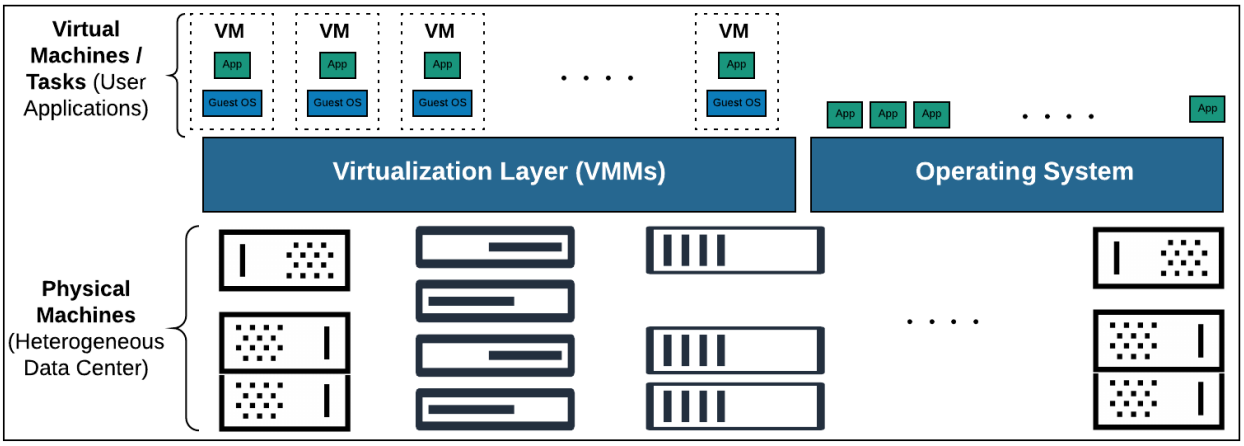

Figure 1.1: Simplify illustrative heterogeneous data centres

the host operating system. In both cases, an operating system called VM Monitor (VMM) is used to run on the virtualization platform. A VM controller enables VM to communicate with the hardware via VMM. It is notable that the VMM virtualizes the hardware for each VM [3], Figure 1.1.

Data centre power is among the large commodity expenditures in IT service for most organizations. A large-scale data centre may require multiple megawatts/hour. The electricity demand of global data centre, estimated around 194 TWH in 2014, has been doubling every five years [4], see Figure 1.2 [2]. Put differently, the energy costs of cloud data centres have recently emerged as a practical concern, as discussed in [5, 6].

Most of data centre consumed energy, that is, about 88%, has been geared towards powering and cooling the IT equipment. Amazon estimated that the cost of energy for data centres reached 42% of the total cost of operation [7]. In addition, according to the Environmental Protection Agency (EPA), every 1000 KWh of power consumption emits 0.72 tons of CO2. The reduction of energy has become a first-order objective in the design of modern computing systems [2].

Cloud users order a group of VMs based on Service Level Agreement (SLA). These

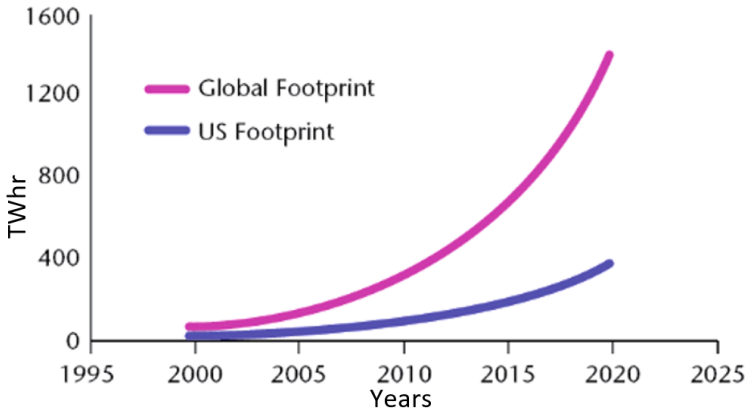

Figure 1.2: Projection of data centres electricity use

VMs are placed in different PMs and allow communications amongst each other. This enables cloud users and providers to scale up or scale down the resources depending on the requirement. In addition to ensuring better resource utilization, it reduces the cost and the energy by sharing physical resources among multiple VMs [8].

Efficient energy consumption is determined by: hardware efficiency, resource management systems deployed on the infrastructure, and the efficiency of applications running in the system [9]. VMs consolidation is undertaken to reduce the number of PMs by re-allocating or performing live migration of VMs among PMs without interrupting the services. It takes the performance based on a pre-defined SLA into consideration. This, in turn, will increase the power efficiency in the data centre by turning off unused servers.

There are two approaches of VM consolidation: *Static* and *Dynamic*. The static provisioning is useful when there are pre-defined resources of all types of VMs that will eventually be used. Energy reduction will be done mostly based on simple heuristics or historical VM demand patterns before commencing the auction. This could inflate the cost for application providers during the period of low demand resources. On the other hand, available resources may be insufficient during periods of high utilization [10]. Dynamic VM consolidation, which is the objective of this thesis, determines variable amounts of resources in a given time in

order to meet Quality of Service (QoS) expectations amidst variable workloads [11–13].

1.2 Thesis Objectives

The objective of this thesis is to propose a real-time dynamic provisioning framework to reduce energy consumption in a data centre through an efficient data pre-processing and VM clustering, novel PM indexing factor, developed multivariate prediction, as well as an improved placement algorithm. It takes into consideration user behaviour, the output of a data centre monitoring tool, and the state of PMs in order to improve these processes. This framework optimizes VMs consolidation under QoS constraints based on energy consumption in a data centre that contains heterogeneous physical resources. This thesis aims to answer the following questions:

- **How to predict the workload?**

  Predicting the future state of VMs in a data centre and new users' VMs request is a crucial process for efficient resource utilization. This will be necessary not only for energy saving purposes, but also for real-time control, resource allocation and capacity planning.

- **What is the current state of the data centre?**

  Monitoring tools and estimation algorithms is used to determine the data centre current state and behaviour. Monitoring tools helps provide online information on current physical and virtual resources state as well as about users usage. The state of data centre is necessary to manage data center's resources. Physical resources are not a single thing, and include servers, storage subsystems, networking switches, routers and firewalls, in addition to the cabling and physical racks that are used to organize and

interconnect the IT equipment. Monitoring tools such as Data Centre Infrastructure Manager (DCMI)[1] are able to provide this information in real-time. The virtual resources include the VM and user states. Algorithms are usually used to model and predict VM and user behaviours.

- **Which VMs to migrate?**

  To migrate a VM, it is necessary to determine which VMs should be migrated to reduce the number of PMs being used. The decision to migrate VMs must consider the following: the overhead caused by migration in QoS requirements, the status of PMs in terms of power efficiency, reliability, deviation in performance of hosted VMs, as well as the reason why migration is occurring to establish an order of migration.

- **Where is the best place for the new and migrated VMs?**

  Determining the best place (PM) and placement algorithm used for implementing the allocation process for both new and migrated VMs is another essential aspect that influences the quality of VM consolidation and energy consumption.

- **When and which PM to switch on/off?**

  In order to optimize energy consumption and to avoid violations of the QoS requirements, a unique metric must be put in place to compare PMs. Through this unique metric, we can determine when and which PM should be ON or OFF. Deactivation and reactivation will be useful either for saving energy, or for handling increases in the demand for resources.

In this work, we design an energy-aware VM consolidation framework consisting of several algorithms and techniques in real-time to reduce energy consumption through dynamic VM

[1] https://www.sunbirddcim.com/

consolidation. It is also necessary to use a distributed or a pipeline system in order to provide scalability and eliminate single points of failure. While traditionally global resources management algorithms are centralized, it is necessary to formulate a new approach for implementing the dynamic VM consolidation system in a distributed manner.

1.3 Thesis Contributions

Contributions of this thesis can be described as follows:

1. We have presented a background of the general framework. This framework includes multiple phases of complete proactive VM consolidation processes in a heterogeneous large-scale data centre, and will provide complete guidance on how these components work together in an interactive manner. This contribution has been published in part in the following:

    - "Proactive dynamic virtual-machine consolidation for energy conservation in cloud data centres," *Journal of Cloud Computing*, vol. 7, no. 1, pp. 1-28, 2018.
    - "Open source cloud management platforms: A review," *in IEEE* 2^{nd} *International Conference on Cyber Security and Cloud Computing*, pp. 470-475, November 2015.

2. We have designed and implemented a fast data centre workload pre-processing algorithm that is used to discover patterns relationships and extract data centre workload with correlated behaviour. This algorithm also take user behaviour into consideration [14]. This contribution has been published in part in the following:

    - "An efficient workload clustering framework for large-scale data centers," *in IEEE*

8^{th} *International Conference on Modeling, Simulation and Applied Optimization (ICMSAO)*, April 2019.

3. We have designed and implemented a systematic VMs clustering subsystem for large-scale data centres. This subsystem entails less consumption resources, faster computing and improved clustering accuracy. Furthermore, a simple filtering approach is used to effectively exclude unpredictable VM/task requests caused by users' actions. This contribution has been published (in part) in the following:

   - "A systematic cloud workload clustering technique in large scale data centers," *in IEEE World Congress on Services (SERVICES)*, vol. 2642-939X, pp. 362-363, July 2019.
   - "Energy-consumption clustering in cloud data centre," *in IEEE* 3^{rd} *MEC International Conference on Big Data and Smart City (ICBDSC)*, pp.1-6, March 2016.

4. We developed and implement a multivariate Extreme Learning Machine (ELM)-based predictor subsystem, which makes the prediction parameters tunable in real-time based on the actual request load for each iteration. This increases the prediction accuracy over time and obviates the need for frequent model training required by other machine learning approaches. By estimating the number of VMs in each interval, it is possible to determine the number of PMs required. This will narrow the target of the optimal placement algorithm. This contribution has been published (in part) in the following:

   - "Multivariate time series ELM for cloud data centre workload prediction," *in Human-Computer Interaction. Theory, Design, Development and Practice* (M. Kurosu, ed.), pp. 565-576, Springer International Publishing, 2016.

- "Using ELM techniques to predict data centre VM requests," *in IEEE* 2^{nd} *International Conference on Cyber Security and Cloud Computing*, pp. 80-86, November 2015.

5. We have proposed and implemented a novel index that is used to efficiently model hosts in any cloud data centre. The index is inclusive of quantifiable as well as non-quantifiable parameters. It is successfully used to select the PMs that are necessary to handle new or consolidated VMs. The proposed index is also used to convert the multi-objective placement algorithm into a single-objective. This contribution has been published (in part) in the following:

   - "A universal unit for measuring clouds," *in IEEE International Conference on Humanitarian Technology Conference (IHTC)*, pp. 1-4, May 2015.
   - "A novel host readiness factor for energy efficient VM consolidation in cloud data centers," *in IEEE* 8^{th} *International Conference on Modeling Simulation and Applied Optimization (ICMSAO)*, pp. 1-5, April 2019.

6. The proposed index enable us to extend and implement the placement algorithm for the purpose of handling requests with multiple resources such as CPU, memory, bandwidth, etc. This contribution has been published (in part) in the following:

   - "Energy efficient virtual machine placement algorithms in heterogeneous data centre using machine condition index (MCI)," *Under review* 2020.

7. We proposed an integrated pipeline resource provisioning framework that relies on consecutive algorithms and techniques to make suitable energy-aware resource management decisions. This contribution has been published (in part) in the following:

- "Real-time energy-conserving VM-provisioning framework for cloud-data centers," *in IEEE* 9^{th} *Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 0765-0771, Jan 2019.

In addition, we have also published the following as part of algorithms development in this thesis:

- "Energy-efficient resource allocation for cloud data centres using a multi-way data analysis technique," *in Human-Computer Interaction. Theory, Design, Development and Practice* (M. Kurosu, ed.), pp. 577-585, Springer International Publishing, 2016.
- "An extreme learning machine (ELM) predictor for electric arc furnaces' v-i characteristics," *in IEEE 2nd International Conference on Cyber Security and Cloud Computing*, pp. 329-334, November 2015.
- "Role-based multiple controllers for load balancing and security in SDN," *in IEEE International Conference on Humanitarian Technology Conference (IHTC)*, pp. 1-4, May 2015.

1.4 Thesis Organization

The remainder of the thesis is organized in the following manner:

In Chapter 2, a background and review of the framework techniques, as well as algorithms that are relevant to the scope of this thesis, is presented. This chapter also highlights various algorithms that are involved in implementing of our proposed energy-aware VM consolidation based on energy consumption and justifies their choices.

In Chapter 3, a real-time VM provisioning framework based energy consumption in large-scale heterogeneous data centres is presented. This chapter introduces the core components

of the proposed framework: pre-processing, clustering, prediction, and VM placement.

In Chapter 4, we present workload clustering applications in cloud data centres. More specifically, a novel systematic method to select the suitable VMs clustering method will be proposed. This selection process will be premised on clustering purpose, validation indices, and result comparison.

In Chapter 5, we develop a multivariate ELM-based predictor that will be used as an essential milestone in an efficient data centre workload prediction framework. Additionally, we will introduce the technique used to handle the problem of predicting window sizes in order to optimize PM utilization.

In Chapter 6, a novel machine index will be introduced. This machine index will meter all resources related to servers in cloud data centre and provides relevant information for design/implementation of resource management systems as well as resource allocation algorithms.

In Chapter 7, the contribution of this thesis is summarized. We also explicate some research problems and directions for the future.

Chapter 2

Background and Literature Review

In this chapter, we provide a background along with an in-depth survey of the most effective techniques and algorithms used in proactive dynamic VM consolidation focused on energy consumption. In addition, we present an overall review for the framework that uses multiple phases of a complete consolidation process. The review presented in this chapter has also appeared in [2].

2.1 Introduction

Data centre energy efficiency measures the reduction of energy used by hardware or software components for a given service or activity level. Hardware equipment includes both IT equipment (e.g. network and servers) and supporting equipment (e.g. power supply, cooling and data centre building itself). On the other hand, software components may include Cloud Management Systems (CMSs) that are used to manage the entire data centre or end-users' applications, hypervisors, operation systems and different applications [15, 16].

Given that a large component of power consumption of data centres is in their hardware equipment, this chapter focuses on the problem of reducing energy consumption through

efficient management of PMs and VMs in the data centre [9, 17]. Four different strategies are used to manage PMs and/or VMs to reduce energy:

- **VM resizing**: is the category of changing the amount of resources reserved for VMs through either adding or removing resource elements, or increasing or decreasing the capacity of each resource element in a VM. All these processes will be undertaken without executing a reboot, an application restart, reconfiguration or recreation of a VM [18]. Under this category, we will attempt to adjust the utilization of PMs to actual load, which typically leads to a reduction of power [19–22].

- **Optimal initial placement**: seeks to optimally assign VM or group of VMs to servers, as part of an initial deployment, such that the mapping minimizes the total inter-rack PMs used or traffic load within the network to reduce energy [23]. Section 2.5.1 will discuss these algorithms, such as those proposed in [24–26].

- **Overbooking of physical resources**: refers to the strategy of overlaying requested virtual resources onto physical resources at a higher ratio than 1:1 [27]. This strategy can facilitate an improved utilization of PM idle resources. However, special care must be taken to reduce risks associated with unmet QoS demand over peak utilization of PM resource [28, 29].

- **VM consolidation**: is the process of using minimum possible active PMs through migrating VMs overtime in an optimal fashion to reduce resource consumption [30–33].

As described in Section 1.1, there are two general types of VM consolidation: *static* and *dynamic*. Under static consolidation, sizing and placement of VMs on PMs are predetermined when a job arrives and the placement remains unchanged over a period of time. Thus, this type of VM consolidation is often suitable for short-running jobs for a couple of

hours, where PMs resources for different types of VMs are predefined [34]. Energy reduction will mostly be based on simple heuristics or historical VMs demand patterns. This may lead to an increase in the cost of application provider during the period of low resource demand, whereas during the available resources may be insufficient during high utilization periods [10].

Dynamic VM consolidation can lead to the utilization of fewer PMs by re-allocation or live migration of VMs among PMs without significant interruption of services. It takes into consideration the performance as it based on QoS, which is predefined via SLA between the tenant and the service provider. This, in turn, will increase the power efficiency in data centres by turning off unused servers to save power [11, 16]. Dynamic provisioning-based energy consumption can represent the most efficient methods to improve the utilization of the resources and reduce energy [11–13].

Dynamic provisioning falls under *reactive* or *proactive* categories. Reactive provisioning is set to change the initial placement after the system attains a specific undesired state. The change may be made due to performance, maintenance, power or load issues, or SLA violations. Under proactive monitoring, historical data and prediction algorithms are used for altering the VM's initial placement before the system attains a certain condition [35,36]. Proactive provisioning uses prediction-based approaches that help prepare advanced changes in the workload and system usage [37].

This chapter elucidates a comprehensive VM management framework to survey algorithms and techniques used under proactive dynamic VM-provisioning in data centres with an emphasis on energy consumption. We propose to reflect a large number of overlapped domains used in typical dynamic energy consumption based on VMs consolidation. These domains can be classified into the following four main interactive subsystems, as shown in Figure 2.1:

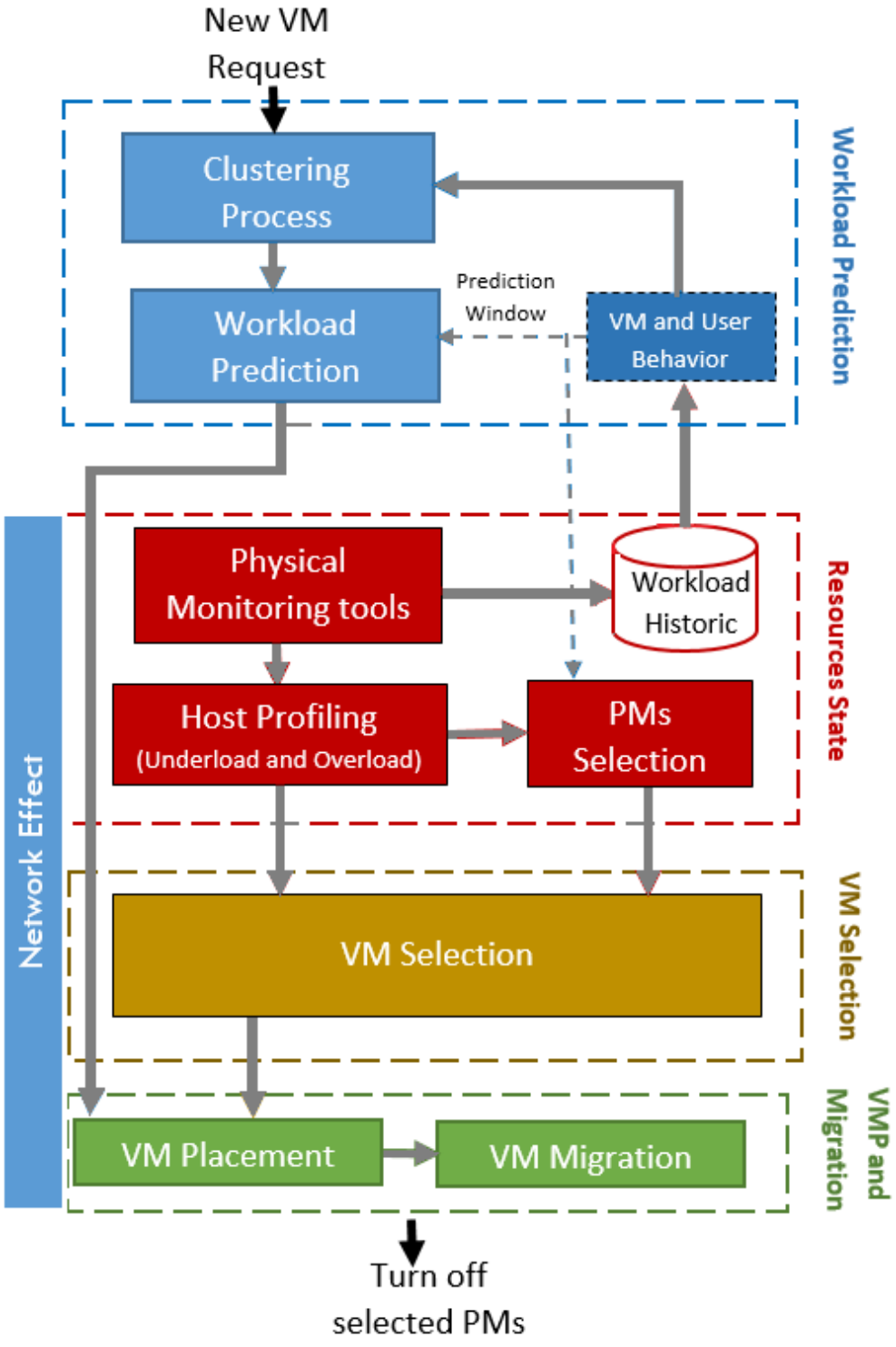

Figure 2.1: Proactive dynamic VM consolidation framework

1. **Workload Prediction Subsystems** focus on algorithms and techniques used in clustering process, VM and user behaviour estimation, prediction of window size, and forecasting process as part of the workload prediction subsystem.

2. **Resource State Subsystems** are used to identify the state of physical and virtual resources. This subsystem not only includes the monitoring and tracking tools, but also focuses on algorithms and techniques used in defining the PMs states.

3. **VM Selection Subsystems** focus on VM selection criteria, including VM state estimation.

4. **VM Placement and Migration Subsystems** deal with the question of how to migrate selected VMs.

Networking strategies play a pivotal role in this framework, given that network infrastructure topology and routing protocols can have a direct impact on the migration or consolidation with minimum network load [38]. Figure 2.2 illustrates the logical connection between these subsystems.

In this chapter, we will review and analyze available algorithms and techniques to answers the following questions:

- How to predict workloads? What are the expected future VM requests?
- What is the current state of the resources? How can the behaviour of physical and virtual resources be monitored and tracked?
- Which VMs are to be migrated and where?
- How to migrate selected VMs?

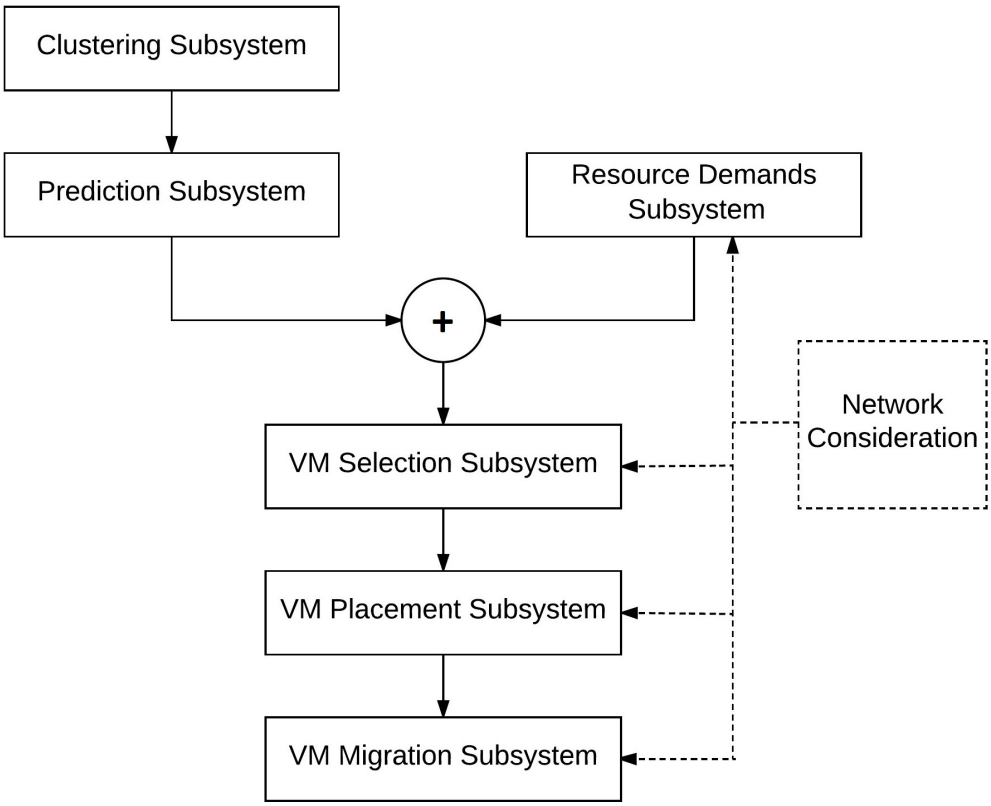

Figure 2.2: Logical connections between VM consolidation subsystems

The rest of the chapter is organized based on the subsystems described above. Section 2.2 reviews applications and techniques in workload prediction subsystems. Section 2.3 reviews algorithms related to data centre resource states, which are then used in VM selection subsystems in Section 2.4. VMs and hosts selections are described in Section 2.5. VMs placement and migration subsystem is covered in Section 2.6, while the network effect on all these subsystems is presented in Section 2.7. This is followed by a brief summary in Section 2.8.

2.2 Workload Prediction Subsystems

Resource estimation underlies various workload management strategies, including dynamic provisioning, workload scheduling, and admission control. All these approaches possess a common prediction module that provides estimations to determine whether or not more resources are to be added, the order of query execution is to be rearranged, and a new incoming query is to be admitted or rejected [39]. Prediction of future resource behaviour is a crucial process for efficient resource utilization in a dynamic cloud computing environment because workload forecasting for long or short periods will be necessary for real-time control, resource allocation, capacity planning and data centre energy savings in cloud computing [36].

For the purpose of energy conservation and precise proactive VM consolidation, the forecasting based on demand history must address some or all of the following challenges [40]:

- Finding a way to make predictions that take into consideration both user, virtual and physical resources variations,
- Overcoming the problem of time-varying demands,
- Estimating the required observation window size, and
- Detecting when the prediction is likely to be incorrect and how this problem can be overcome.

In a heterogeneous cloud environment, it is may difficult to predict the demand for each type of resource separately [39, 41]:

- Typically, VM requests consist of varying amounts and types of cloud resources (e.g., CPU, memory, bandwidth, etc.). The multi-resource nature of these VMs poses a unique challenge in terms of developing prediction techniques.

- Different cloud clients may request different amounts of VM resources that may be assigned on the same PM as opposed to separate machines. Therefore, it is both impractical and too difficult to predict the demand for each type of resource on a separate basis.

Therefore, it is logical to create different categories of VM clusters, before developing prediction techniques for each of these clusters. In addition to covering the most practical and recent published prediction algorithms, these subsystems will also include: clustering; prediction window size; and VM and user behaviours. Thus, the workload prediction subsystems are categorized into four functional areas. Each category is presented in a subsection:

- **Clustering Process**: this subsection reviews the recent literature in data centre workload clustering.
- **Prediction Process**: this subsection reviews algorithms and techniques used to forecast future resource demand in data centres.
- **Prediction and Observation Windows Size**: this subsection reviews prediction of window size and approaches of observation window size selection.
- **User and VM Behaviour**: this section reviews the current approaches in analyzing and supporting users along with resources behaviours. User and VM behaviours have a strong influence on the overall cloud workload.

2.2.1 Clustering Process

Clustering is an effective unsupervised learning technique that groups together items that are naturally similar to each other based on a certain metric [42]. This process is used to map each request received into one of a set of clusters with different types of VMs or tasks

during the predefined time period. It is notable that for fuzzy partitioning, a point can belong to more than one group [43]. The prediction algorithm can be used to predict the number of VM in each cluster rather than predicting each type of VM [44].

This subsection will present a brief summary of clustering techniques related to VM consolidation and scheduling in extant literature. In Chapter 4 we will propose a systematic clustering process used along with its implementation.

k-means has been used by Dabbagh *et al* [41] and Chowdhury *et al* [45] to create a set of clusters so as to group all types of VM requests. Each request represents a VM with CPU and memory for Google traces data [46]. k-denotes algorithm inputs, including Google traces and the number of clusters, while the output is the centres of these clusters. The selection of k should be balanced between two conflicting objectives: reducing errors and maintaining low overhead [39].

Khan *et al* [47] introduced a co-clustering algorithm to identify VM groups and the time periods in which certain workload patterns appear in a group. Subsequently, they used the Hidden Markov Model (HMM) to explore the temporal correlations in workload pattern changes. This helps predict individual VM's workload based on the groups identified in the clustering step.

A kernel fuzzy c-means (FCM) clustering algorithm was used to forecast future CPU loads by Xu *et al* [48]. After dividing historical long CPU load time series data into short equal sequences, they used kernel FCM to put the sub-sequences into different clusters.

Canali and Lancellotti [49] used Principal Component Analysis (PCA) as an automated methodology to cluster VMs by leveraging the similarity between VMs' behaviour. It is notable that they considered VMs as a member of classes running within the same software component. This methodology has been applied to two case studies: virtualization tested and a real enterprise data centre. This methodology can reduce the amount of collected

data, thus making effective contribution to addressing the scalability issues of the monitoring system. This technique is very useful for monitoring and reporting, but it is difficult to use it as an input to a prediction algorithm to forecast each type of VMs in the nearest future. This is attributed to the following reasons: (1) PCA relies on linear assumptions; and (2) PCA based on mean vector and covariance matrix, which means that some distributions may be characterized by this but not all.

Claudia and Lancellotti [50] combined the Bhattacharyya distance and ensemble techniques to evaluate the similarity between the probability distributions of multiple VM resource usage. The researchers considered both system and network-related data. Their proposal achieved high and stable performance in automatic VM clustering through their experiments on real-data gathered from an enterprise data centre. VM Clustering was used to lower the amount of data necessary in cloud monitoring.

The workload is always driven by the users; therefore, realistic workload models must include user behavioural patterns that are linked to tasks. The previously described approaches completely focused on tasks, thus neglecting the impact of user behaviour on the overall environment workload [51].

According to our observations, all the currently used various clustering techniques do not provide a structured model that can be used for conducting simulations. Most of previous work rely on comparing the execution time or cluster quality. Workload analysis needs to explore more than coarse-grain statistics and cluster centroids. In order to capture the patterns of clustered individuals, it is also necessary to conduct an analysis of the parameters and examine the trends of each cluster characteristic. This will underscore need for new methodologies, especially for real-time and online streaming data [51, 52].

In Chapter 4 of this thesis, we will propose a novel systematic framework to select the suitable VMs/tasks clustering method in large-scale data centres based on the following:

clustering purposes, validation indices, and results comparison. The proposed clustering framework also takes into consideration VM and user behaviour. In the proposed systematic clustering, our contributions will be as follows:

- To design a novel efficient VMs/tasks clustering framework for large-scale data centres, so as to achieve higher efficiency with the consumption of fewer computation resources, fast computing, and accurate results.
- To propose an efficient pre-processing algorithm for such applications, in order to convert the big data problem to a small 2D matrix include unique VMs/tasks with CPU and memory size only. By including VM and user characteristics in the clustering process.
- To combine VMs/tasks clustering techniques and validation indices to improve and select suitable clustering techniques.

2.2.2 User and VM Behaviour

Analyzing and supporting behaviours of users and tasks is a crucial process for both data centre providers and their prospective users. The behaviour analysis within a specific course of time enables decision-makers to plan ahead for incoming workloads into data centres and ensure the fulfilment of all behaviours. As an increasing amount of data is stored and processed in data centres, anticipating behaviour of users and VMs inevitably becomes an onerous process. The behaviour modeling and estimation can be used to anticipate requests as well as consumption patterns.

The workload analysis captures both user and VM behaviours. Subsequently, users and VMs are clustered based on characteristics are defined during workload modeling. Moreno *et*

al [51] showed that users with profiles U submitting tasks classified into profiles T. Each user profile U_i can be defined by the probability of each task profile T_i. The expectation $E(U_i)$ of a user profile is denoted by its probability $P(U_i)$, whereas the expectation $E(T_i)$ of a task profile is represented by its probability $P(T_i)$ conditioned to the probability of $P(U_j)$.

$$E(U_i) = U_i P(U_i) \tag{2.1}$$

$$E(T_i) = T_i P(T_i) | P(U_j) \tag{2.2}$$

Behaviour prediction models are used to predict application behaviours and VM behaviours in a cloud by tracing recently observed patterns. The frequent changes in workloads are used to calculate the required resources. This will be used to guide dynamic management decisions. Heuristic techniques, such as predefined thresholds and Auto-scaling, are used to perform scaling operations (adding or removing resources) without requiring human interactions [53].

In [54] monitoring changes in behaviours are saved in the History Table Predictor (HTP). In this table, each row presents pattern changes. After finding a new pattern, the model attempts to find a match in the table. This will be useful in the prediction phase or to store that new pattern in case no matches are found. Sarikaya *et al* [55] used the Statistical Metric Model (SMM) as an effective technique to outperform the HTP technique. They used another historical predictor for long-term global patterns modeling in application behaviour.

The SMM model can be applied in cloud environments using common resource components denoting the behaviour of the workload. In particular, these components include memory utilization, CPU utilization, and network utilization. The three components can be combined to use as a load volume and formulated as data centres.

Data mining techniques can be used to discover Frequent Workload Patterns (FWPs) in accordance with the previous history of resource usages, as observed in [56]. The resource allocations can be determined by using the Association Rules Technique (ART) based on the prediction of resource availability in a given time period. The idea behind using ART on the discovered data patterns is to identify the possibility that the same patterns will be repeated in the future. Put differently, ART can be used to represent the correlation between data patterns.

Techniques mentioned above work well in discovering patterns in workload data and prepare them for subsequent operations such as resource allocations and predictions. Other different techniques can be used to perform these operations. Since the data is already assumed to be trained, resource allocations and predictions will be improved.

Since user behaviours have a large impact on improving prediction results, this thesis will incorporate users as one of the key model variables in order to improve the clustering process and subsequently, prediction accuracy. Prediction accuracy, in turn, enables the proposed prediction framework to lower the number of required PMs, thus obtaining better energy conservation. Thus, we will do that in Section 4.3 and Section 5.3:

- Efficient pre-processing historical data to extract the number of effective users.
- Combining clustering for users as well as VMs workload in the prediction process, through the creation of a VM/user mapping matrix.
- Using user clustering to propose a filtering process to eliminate VMs with the lowest probability from the prediction process.

2.2.3 Prediction Process

As discussed in Section 2.1, a proactive dynamic VM consolidation is to estimate resource requests. This can be achieved by forecasting future resource demand values on the basis of demand history. Since workloads tend to trace resources patterns based on time, time series forecasting methods are expected to reliably predicted resource demand [40].

Prior works have focused on how to save energy, improve performance and increase profit and so on [57]. In this subsection, the most recent prediction techniques, especially ML techniques, applied in the field of VM consolidation based energy consumption, will be reviewed. Prior to that, a simple description of the basic principle of prediction problem and their techniques will be provided.

Basically, prediction problem requires the estimation of values of m outputs $Y(t) = [y_1(t), y_2(t), \cdots, y_m(t)]^T$ from sets of readily available n inputs $X(t) = [x_1(t), x_2(t), \cdots, x_n(t)]$, and can be formulated simply by:

$$\hat{Y}(t) = \hat{f}(X(t)) \tag{2.3}$$

Where $\hat{Y}(t)$ is the predicted values and $\hat{f}(X(t))$ denotes the estimated relationship between inputs and outputs of the system at a specific time t. This relation is either *linear* or *nonlinear*. In Linear Regression (LR) models, the relationship between one or more input variables and dependent output variable(s) is linear, such as Auto-regressive (AR), Moving Average (MA) and Gray Forecasting Model (GFM), and Wiener filter [40].

Linear models do not capture the real-world phenomena, which is why nonlinear models are necessary. In many situations, the underlying nature of the process being modeled is not clearly known or is difficult to model. In such cases, machine learning algorithms are effectively used to address many problems. These methods include basis functions such as

Radial Basis Functions (RBF), Artificial Neural Networks (ANN), and k-Nearest Neighbors (KNN). This section will summarize some of these algorithms that are used in workload prediction with energy consumption in cloud environments.

In Auto-regressive Moving Average (ARMA), the basic assumption made to implement this model is that the considered time series is linear, Auto-regressive (AR) and follows a particular statistical distribution, such as Normal distribution, Moving Average (MA). ARMA can be represented by:

$$\hat{Y}(t) = C + \sum_{i=1}^{O_p} \varphi(i) Y(t-i) + \sum_{j=1}^{O_m} \theta(j) \epsilon(t-j) + \epsilon(t) \tag{2.4}$$

where O_p and O_m denote integer constants representing the order of the models. $\varphi(i)$ and $\theta(j)$ are models parameter. $\epsilon(t)$ are random errors at time period t, whereas C is a constant.

If the original process $Y(t)$ is not stationary, then take the first-order difference process $\Delta Y(t+1) = Y(t) - Y(t-1)$ or the second-order differences $\Delta^2 Y(t+1) = Y(t) - 2Y(t-1) + Y(t-2)$ and so on. $Y(t+1)$ is said to be an Auto-regressive Integrated Moving Average process, while ARIMA(O_p, d, O_m), if $\Delta^d Y(t)$ is an ARMA(O_p, O_m) process.

Researches on workload prediction have been carried out based on statistical approaches, such as [58–60] who proposed an ARIMA algorithm. The basic assumption is that the considered time series is linear and follows a particular statistical distribution, such as normal distribution. AR, MA, ARMA and ARIMA techniques, all of which can be used to models several time series. A key tool in identifying a model is an estimate of the autocovariance function [40].

Gray Forecasting Model (GFM) can be used to forecast the behaviour of non-linear time series. This non-statistical forecasting method is particularly effective when the number of

observations is insufficient.

Grey forecasting model, precisely GM(1, 1) model, is one of the most widely used techniques in the Grey system [61,62]. Under this technique, the predicted value of $\hat{Y}(t)$ can be obtained by accumulated generation sequence of the original data sequence.

Jheng *et al* [62] proposed a GFM to forecast the workload of the PMs in a cloud data centre. The main characteristics of GFM are simple and have the ability in a time series prediction with the least amount of historical data. This is done by extracting actual models in a system that uses existing data [63], where the number of historical data must be more or equal to four. However, the main drawback is that it assumes new data grows exponentially and use time dependency rather than data dependency under a time series forecasting model.

Wiener filter is an optimal-linear discrete-time filter that can be used to produce an estimate of a desired or target random process by linear time-invariant multi-filtering of an observed noisy process, assuming the known stationary signal/noise spectra and additive noise [64].

Dabbagh *et al* [41,65] proposed a framework to forecast the number of VM requests, to be arriving in the near future, in addition to the amount of CPU and memory resources associated with each of these requests. The k-means that clustering was used to create a set of clusters containing all types of VM requests. Stochastic Wiener Filter (SWF) was used to estimate the workload of each cluster. Although, Wiener filter is unreliable for the dynamic behaviour of demand cloud resources because it is suitable to estimate the target random process by Linear Time-Invariant (LTI) for known stationary signal and noise spectra [66]. Dabbagh *et al* improved the original Wiener filter to support online learning, making it more adaptive to changes in workload characteristics.

An alternative approach to address the challenges associated with prediction is LR [67], which models the relationship between one or more input variables and a dependent output

variable by using a linear equation to observed data. Linear models on their own cannot be used to capture the real-world phenomena, which is why nonlinear models are necessary. In regression, all such models will have the same basic form, i.e. Equation 2.3. Typically, they turn to ML that are widely used and quite efficacious for many problems. These methods include basis function regression (including RBFs), ANNs, and KNNs.

Many researchers use a combination of pre-described techniques and other strategies to increase the accuracy of predictions. Cao *et al* [68] suggested an ensemble model for online CPU load prediction. Their model has multiple predictor sets, including AR model, Weighted Nearest Neighbors (WNN) model, Exponential Smoothing Model (ESM), most similar pattern model, and WNN model for differenced data (DWNN). Each predictor has a specific membership that is capable of adjusting dynamically. CPU workload has been estimated by these combined sets through the scoring algorithm. The main drawbacks in this predictors are: 1) it consists of two levels of prediction; all the predictors have specific weight and it is very difficult to find the optimal weight for each predictor; 2) relatively time-consuming in applying different algorithms simultaneously; and 3) most of the suggested set of predictors are based on statistical approaches.

In neural networks, a linear combination of shifted or smoothed step functions, linear ramps, and the bias term is made. This model represented by:

$$\hat{Y}(X(t)) = \sum_{j}^{O_b} w_j^{(1)} \mathbf{b}\left(w_j^{(2)} x(t) + bias_j^{(2)}\right) + bias^{(1)} \tag{2.5}$$

where $\mathbf{b}(X(t)) = [b_1(x(t)),\ b_2(x(t)),\ \ldots,\ b_{O_b}(x(t))]^T$ are the basis functions, O_b is the number of basis functions and $\mathbf{w} = [w_1,\ w_2,\ \ldots,\ w_{O_b}]^T$ are wights of basis function.$bias$ is the bias term. In ANN, a basis function could be a sigmoid function or RBF. The most common choice of sigmoid is given by:

$$\mathbf{b}(x) = \frac{1}{1 + e^{-x}} \tag{2.6}$$

These sigmoids are combined to create an ANN model used for regression with multi-dimensional inputs $X(t) \in \mathbb{R}_2^n$, and multi-dimensional outputs $Y(t) \in \mathbb{R}_1^2$, and for 1D case model, Equation 2.5. Usually, this objective function cannot be optimized in a closed-form, thus necessitating numerical optimization procedures.

ANN and LR are widely applied in previous works to forecast VMs workload in cloud environments [69]. The main problem with this approach, and in most of the LR applications, is as follows: (1) They considered the fact that future workload could be independent of their previous workload pattern [24]. (2) The workload has an obvious nonlinear feature [70]. (3) LR demands workloads that have a simpler behaviour than those of an ANN-based method [58].

Several studies use ANN as a prediction model [71–73]. Although ANN represents a universal approximation, it still suffers from the drawbacks of choosing a suitable algorithm, network structure, and initial condition. For improved performance, ANN may be combined with typical prediction methods such as Sliding Window Method (SWM) [74], AR model [70], and Fuzzy System (FS) [24, 70, 75].

Dynamic behaviour forecasting problems can be resolved with ANN [71–73], Adaptive Neuro-Fuzzy Inference System (ANFIS) [24, 70, 75], Support Vector Machine (SVM) [67], and latent feature learning-based models [68, 70, 76].

Bey *et al* [75] combined Adaptive Network-based Fuzzy Inference Systems (ANFIS) and clustering process to estimate the future value of CPU load. The model was carried out on real CPU load time series to determine the optimal number of clustering for one machine. According tot their findings, the CPU load prediction using ANFIS model for each category

performs better than using one ANFIS for the entire CPU time series without clustering.

Bey's work was improved by Chen *et al* [70], an ensemble model where a subtractive-fuzzy clustering-based fuzzy neural network were adopted. Fuzzy-neural network performance was optimized using a fuzzy-subtractive clustering algorithm that comprises of the FCM clustering algorithm and the subtractive clustering algorithm.

In [24], a neural network model was proposed to predict workload patterns in VMs, while Fuzzy Expert System (FES) was used to control near-future changes in workload patterns for every VM. This scheme has been used to determine the time during which the VMs will be overloaded and need to be migrated.

Combining fuzzy and ANN improves the modeling and prediction process; even ANFIS is reported to perform better than NN [77, 78], but both of them require training before being used.

Table 2.1 depicts that the prediction techniques can be divided into statistical, machine learning and hybrid approaches. In this table, Owin and Pwin denote the observation and prediction windows, respectively.

Table 2.1: Workload prediction techniques

	Techniques	References	Parameter		Clustering	User behaviour	Window size
			VM	PM			
Statistical	ARIMA	[58, 79]	✓				Fixed
		[59]		✓			Fixed
	GFM	[62]		✓			Fixed
	HMM	[47]	✓		co-clustering		Fixed
	Bays Model	[80]		✓		✓	Owin=1/2 Pwin
	Multi-Way Data Analysis	[81]	✓		FCM	✓	Fixed
Hybrid	AR Model, ESM, WNN, DWNN	[68]		✓			Owin=2 Pwin
	ECNN and LR	[74]		✓			Fixed
	Ensemble Model based FNN	[70]		✓	FCM/subtractive		Fixed
	Static and adaptive Winner Filter	[41]	✓		k-means		Fixed/overlapped
ML	SVM, NN, and LR	[67]		✓			Fixed
	GA to optimize Elman NN	[48]		✓	Kernal FCM		Fixed/overlapped
	NN and Fuzzy expert	[24]	✓				Fixed

In Chapter 5 of this thesis, we develop a multivariate ELM model, which will be a key part of an efficient workload prediction system in a data centre. This prediction framework

not only uses VM historical usage values, but also takes into account the behavior of VMs and users. The main features of the proposed predictor in comparison to the earlier work discussed are as follows:

- This predictor represents an online sequential process. The predictor helps eliminate the restrictions pertaining to observation window size and the number of clusters.
- ELM-based predictor overcomes conventional gradient-based learning methods requirements, such as ANN and ANFIS [70–73], including avoiding stopping criteria, selecting learning rates, size of learning epochs, and local minimums [39].
- Unlike in previous works [41, 48], they use a single ANN to predict all VM requests. This predictor allows each cluster to have its own predicting network.
- The proposed ELM-predictor resolves the problem of time-varying VM requests by depending on the actual service demand.

2.3 Resources State Subsystems

As a part of VM placement algorithm, a VM or group of VMs are assigned to server-racks. This mapping minimizes the total inter-rack PMs used or traffic load in the network to reduce energy, as an initial state [23]. Within a typical data centre, dynamic VM consolidation can be done though migrating VMs over time in an optimal fashion. The typical data centre is homogeneous, containing old and new PMs with different types. Put differently, in a typical data centre, PM's power consumption has varying parameters depending on the PM's load [82]. Thus, it is very important to identify the state of physical resources before and after an initial assignment of VMs for any efficient VM consolidation.

The next subsections review the practical useful monitoring tools in data centres and extrapolate the most recent algorithms and techniques used in extant literature to define PMs state. According to these algorithms, host(s) will be selected. This host represents the best placement of new or selected VM to migrate. Although host selection may depend on several factors such as workload dependencies, security, and network load, this section will encompass the selection process based on PM load. Subsections 2.3.1 and 2.3.2 discuss algorithms related to selecting the host that will be switched-off host underload, after which the host that will move some of the VMs due to overloading the host. However, prior to that, practical data centre monitoring tools will be discussed.

Data centre state monitoring represents all physical components considerations and monitoring by tracking the behaviour of these resources. Put succinctly, this is a process of continuously measuring and accessing infrastructure/application behaviours in terms of performance, reliability, and power usage without compromising on QoS. An effective data centre monitoring tools used in dynamic consolidation must able to:

- Provide power information as well as the state of PMs and VMs,
- Combine monitoring data arrived at a different sampling rate from unrelated monitoring systems,
- Analyze the measurement data, and select the most affected parameters to reduce the storage/computation load, and

Monitoring power consumption is required not only for understanding the manner in which power is consumed, but also for evaluating the impact of energy management policies [83]. It will help detect and track the variations or failure of resources and applications [84]. The myriad tools used in cloud monitoring include: *Collectd*, *Nagios*, *Ganglia*,

PRTG, *Nagios*, *SolarWinds*, *Zabbix* and *Ceilometer*, which provide the capability to monitor the computing, networking and storage resources utilization. Independent test results and case studies between these monitoring software can be found in [85]. In this thesis, we decided to consider the Data centre Infrastructure Manager (*DCIM*) as an effective tool that is capable of providing detailed information about server configuration, hardware, network connections, installed software, and so on. DCIM profiles the power consumed by every single part of the hardware in the data centre [44]. Cloud monitoring tools and platforms properties, issues, analyzing, and comparisons surveys can be found in [86–92].

2.3.1 Host Underload Detection

Host underload is regarded as a necessary component in the process of migration strategies. It refers to the state of a host wherein all VMs should be migrated. In extant literature, the two common techniques used for determining host underload state are as follows: least utilized host and static threshold [13]. If all VMs from the source host cannot be allocated, the host is kept active. Several algorithms are used to determine the underloaded PMs, the majority of which depend on the CPU load in the PM. Table 2.2 compares these algorithms, which can be summarized as follows [13, 93]:

Table 2.2: Host underload detection algorithms

Algorithm	**Policies**				**Characteristic**
	Available Capacity	Migration Delay	Number of VM	Host power	
Least utilized host [93]	✓				Based on a host with minimum resources Not cover the number of VMs on the host
Static Threshold [93]	✓				Depend on the mean of the last CPU used Difficult to find the optimal value of the threshold
Available capacity [13]	✓				Based on available host capacity compared to others Not necessary PMs has less power than the other
Migration delay [13]		✓			Based on the minimum time to complete VMs migration process Need a lot of prediction and estimation
Hybrid [13]	✓	✓	✓		based on MCDM More complicated and difficult for practical
Weighted CPU utilization [94]	✓		✓		Combines host utilization and a number of VMs Need less computation then hybrid

Least utilized: this technique uses CPU usage of the PM as a measure of determining

underloaded PMs. PM is considered as being underloaded when it uses minimal resources. This algorithm is cost-effective because any monitoring system for the CPU utilization will be sufficient enough to identify the underloaded PM . However, it does not take into consideration the number of VMs on that Host and the cost of moving such them to other PM.

Static Threshold: this technique depends on the mean of the latest CPU utilization measurements and compares it with a predefined threshold. If the mean CPU utilization is lower than the threshold, a host underload is detected. Put in Kashyap *et al* [95] use 0.2 for host CPU underload threshold. Constant values of the threshold will be useless, particularly in a heterogeneous environment. It is also difficult to find an optimal threshold value useful for all hosts.

Available capacity: this technique considers the available resource capacity instead of resource utilization as a measure of determining underloaded PMs. This is done by selecting a PM with an available capacity that has the least amount of PMs resources. However, the main drawback of this technique is that PMs with adequate resources do not necessarily consume less power than others. This technique also does not take into consideration the number of VMs on a specific host.

Migration delay: this technique selects PMs based on the minimum time needed to complete all VMs migration process to other PMs. This technique needs a pre-estimation of the migration delay for each VM as far as different PMs are concerned. This technique however, suffers from a poor prediction accuracy due to the underlying complexity of migration cost estimation.

Hybrid: this technique uses a multi-criteria decision-making method that takes into consideration the available capacity of the PM, the number of VMs on the PM, as well as the migration delays of VMs. Although this algorithm may yield more accurate results, it

is very complex to implement.

Weighted CPU utilization and VMs on Hosts: this technique combines the host CPU utilization CPU_{host} and the number of VMs on the host VM_{host} according to the following equation [96]:

$$U_{host_i} = \alpha \cdot CPU_{host_i} + \beta \cdot VM_{host_i} \tag{2.7}$$

Where U_{host_i} denotes the utilization of host i. α and β are weighted for CPU_{host_i} and VM_{host_i}, respectively. Such that, $\alpha + \beta = 1$, $0 \leq \alpha, \beta \leq 1$. Their values are optimized on the basis of workload type with the hill climbing method.

The same technique is used in [94], by combining the CPU utilization as well as the number of VMs according to reward function. While comparing this technique with Hybrid, it was noticed that: 1) it reduces the number of required migrated VMs; 2) Host with the least number of VMs has a better chance to be switched on to sleep mode in comparison with a host with more VMs; 3) It depends on both host utilization and the number of VMs on that host. Finally, it still requires more computation to identify the optimal values of α and β for each host.

2.3.2 Host Overload Detection

Host overload detection is the process of deciding whether a host is considered to be overloaded so that some VMs should be migrated from it to other active or reactivated hosts in order to avoid violating the QoS requirements. Static utilization thresholds, adaptive utilization, and regression-based are some useful techniques [9]. Table 2.3 classifies the host overload detection processes.

Static utilization threshold : is exactly the same as the static threshold in the host underload algorithm. The algorithm compares the latest CPU utilization measurements with

a predefined threshold [93]. As discussed in Section 2.3.1, this technique is unsuitable for dynamic and unpredictable workloads. e.g. Kashyap *et al* [95] use 0.8 for host CPU overload threshold.

Adaptive utilization threshold: is done by using an adaptive threshold on the basis of a statistical analysis of the VMs' historical data. Beloglazov [9, 97] proposed two adjustment criteria Median Absolute Deviation (MAD) and Interquartile Range. MAD depends on statistical dispersion, where a PM with large CPU utilization deviations is weighted more heavily as compared to others. After the threshold is calculated, the algorithm acts similarly to the static threshold algorithm by comparing the current CPU utilization with the calculated threshold. Interquartile Range follows the same principle of MAD, but the distance is calculated by computing the difference between the third and first quartiles in descriptive statistics [98].

Prediction-based algorithms: are based on the estimation of future CPU utilization. Wile they do offer better predictions of host overloading, they are also more complex. Prediction algorithms include:

- ***Local algorithms***: this is done by fitting simple models to localized observations of the CPU utilization in order to build a curve that approximates the CPU utilization.
- ***Robust algorithms***: the algorithm estimates the local parameter and uses them to forecast the future CPU utilization at the next time step, taking into consideration the VM migration time to be estimated [99].
- ***Markov overload detection***: in this algorithm, a constraint on the overload time fraction value will be added as a parameter of the algorithm, while maximizing the time between VM migrations, thus improving the quality of VM consolidation but increase the computation [100].

- ***k-nearest Neighbor***: Farahnakian *et al* [101, 102] proposed two regression methods to forecast the CPU utilization of a PM. These methods utilize the LR and the KNN regression algorithms, respectively, in order to approximate a function based on the data collected during the lifetimes of the VMs. Therefore, they use the function to predict an overloaded or an underloaded PM to lower the SLA violations and energy consumption.

Table 2.3: Host overload detection algorithms

Algorithms		**Characteristics**
Static utilization threshold [93, 95]		Depend on the mean of the last CPU used. Unsuitable for dynamic and unpredictable workload
Adaptive utilization threshold [9]	Median Absolute Deviation	Depend on statistical dispersion Similar to the static threshold
	Inter-quartile range [98]	Same as Median but compares third and first quartiles Provides poor prediction of host overloading
Regression-based algorithms [11, 100]	Local regression algorithms	By fitting simple models to CPU utilization
	regression robust	Uses regression to predict the future CPU utilization
	Markov overload detection	Add constraint in estimation increase the computation

Generally, adaptive utilization threshold algorithms are observed to be more robust than static CPU utilization threshold algorithms in dynamic environments. However, the accuracy of prediction of these algorithms is poor, and the vast majority of them depend on a single resource usage value, which can result in predicaments such as hasty decisions, unnecessary live migration overhead and stability issues [99].

Masoumzadeh and Hlavacs [103] proposed an intelligent and adaptive threshold-based algorithm to detect overloaded hosts by Dynamic Fuzzy Q-learning (DFQL). Its primary difference with regard to the previous technique lies in the fact that the algorithm benefits from the experiment gained by learning procedures to make better decisions about the numerical value of the CPU utilization threshold in the future.

Host overload detection will become a more complex problem when a VM has multiple (e.g. CPU, memory, storage capacity, etc.). For example, authors [104] proposed to use

Multi-Criteria Decision Making (MCDM) algorithms as a promising to tackle the problem of VM selection that involves multiple computing resources. In this approach, these resources can represent the multiple criteria in the problem domain of VM selection. Using common MCDM algorithms such as Analytic Hierarchy Process (AHP) and Analytic Network Process (ANP), pair-wise comparisons can be performed so that the decision-maker (e.g. a cloud engineer or a data scientist) can determine the importance of each computing resource by assigning a weight (e.g. 1 to 10) to allow him/her to determine the influence of one criterion on the others. Not many efforts have been made to tackle the VM selection problem based on multiple resources.

In this thesis, we propose a new unified index as a scalar value to rank hosts in any cloud data centres. This index can incorporate both independent quantifiable and non-quantifiable parameters. This unique index can be used for identifying of underload and overload hosts.

2.4 VM Selection Subsystem

A dynamic energy-awar VM consolidation can partially achieved through migrating all VMs from underloaded host. This should be balanced with variability of workloads and keeping SLA. If a host overloaded some of the VMs moved, they must be moved to other hosts that may be required [96, 105].

VM selection denotes the process of selecting at least one VMs from the full set of deployed or planned VMs to be migrated to other server [12]. VM selection scheme must address the following: which VMs to migrate, and where migrate them to. The main function of the VMs selection subsystem is to determine the best subset of VMs to migrate in order to provide the most beneficial system reconfiguration in terms of energy consumption and many other parameters such as security and bandwidth.

In this thesis, we propose a VM selection technique to select overloaded PMs so as to migrate them to achieve better energy consumption. We will divide the VM selection techniques into conventional, non ML-based, and ML-based approaches. Details are provided in Subsection 2.4.1 and Subsection 2.4.2, respectively.

2.4.1 Conventional, non ML-based, Techniques

Conventional, non ML-based, can be categorized into:

- ***Random Choice***: this technique is based on a simple policy wherein the selection of VM is based on the uniform random process [9].
- ***Dynamic Management Algorithm (DMA)***: in this technique, the VM selection process is based on the CPU utilization of VMs. That is, the VMs with the lowest CPU utilization are selected [93].
- ***Minimum Migration Time (MMT)***: in this technique, the selected VM will be migrated based on the minimum time to complete the migration process relative to other VMs that are allocated on the same host. Beloglazov [9, 97] posited that the migration time refers to the amount of RAM utilized by the VM divided by the spare network bandwidth available for the host.
- ***Maximum Correlation (MC)***: under this technique, the VMs will be selected by calculating the probability correlation between resource usage by an application running on the oversubscribed server. If there is a higher correlation between the resource usages by applications running on an oversubscribed server, it will lead to a higher probability of server being overloaded. This implies that if the correlation of

the CPU utilization of VMs of a particular host is high, then the probability of this host being overloaded is also high [106].

- ***Constant Fixed Selection (CFS)***: this technique is similar to random choice. However, the VM removed from the fixed position in the VM list is found in the overloaded host [107].

- ***Multi-objective optimization model***: this technique is based on the analysis of the impact of CPU temperature, resource usage and power consumption in VM selection. The developed algorithm was evaluated by comprehensive experiments based on VM monitoring Xen. As per their findings, combining all these factors can achieve better VM selection with regard to resource usage, CPU temperature, and power consumption [108].

Although DMA, MMT and CFS are traditional techniques and perform well in static cloud environments, they are not suitable for decision-making in dynamic environments. Although MC needs more calculations, it produces the best selection because it matches the correlation between CPU usage of the current PM with VM behaviour.

2.4.2 Machine Learning Techniques

Most of ML-based techniques in the literature are based on fuzzy logic, given that the selection process is a decision-making problem.

- ***Fuzzy Q-Learning (FQL)***: this technique is an online decision-making strategy. The key component of this technique is to integrate multiple VM selection techniques and dynamically choose a suitable VM selection approach for the current state [11, 109].

- ***Fuzzy VM selection***: this technique has been proposed to select VM from an overloaded host and incorporates migration control in the fuzzy VM selection method. Simulation based on the CloudSim platform determined that the method provides better performance when considering all parameters [110].

However, the exact formulation of the VM selection process should take into consideration as many unrelated elements. Fuzzy logic is challenging to relate these elements in a systematic manner [111].

In this thesis, our VM selection process will be based on the status of PMs in the data centre, particularly for underloaded and overloaded PMs.

2.5 VM Placement (VMP) Subsystems

Virtual Machine Placement (VMP) is the process of mapping VMs to PMs in such a manner that the PMs can be utilized to their peak efficiency. This will help with shutting down unused PMs, depending on load conditions. Each VMP algorithm works well under certain specific conditions. Thus, it is important to choose a technique that suits the needs of cloud users and cloud providers. Additionally, the performance metrics are measured at the level of both the system and the application. These metrics for physical and virtual resources are characterized by their CPU (MIPS), RAM (MB), bandwidth (Mbps), etc. The goal of the VMP algorithm is to determine the minimum number of PMs required for given set of VMs.

Fixed mapping VMP during the lifetime of the VM is known as *Static VMP* and allows changes to initial placement due to unsatisfactory system performance, maintenance needs, power or load requirements. These changes can be *Reactive* or *Proactive* [35].

In the following subsections, we will review the recent algorithms used for dynamic VMP that are capable of improving PM power efficiency in a data centre. VMP techniques can be

classified into four categories: *Deterministic*, *Heuristic*, *Approximation* and *Meta-heuristic* algorithms. These subsections will provide an overview for each, followed by a summary table, Table 2.4.

2.5.1 Deterministic Algorithms

This type of algorithms are based on optimization techniques, where VM sizes and constraints are pre-defined.

Some of the algorithms listed in this category are as follows: Linear programming (LP) [112], Binary Integer Programming (BIP) [113], constraint programming [114, 115], convex optimization [116], pseudo-Boolean optimization [117].

One of the simplest algorithms used is the LP, where the performance goal is linearly related to the placement of VMs. For example, the optimal placement of new VMs on different PMs makes the assumption that the minimal number of PMs required and the resources in each server is subject to a linear function [118]. In BIP, each variable can only take on the value of 0 or 1, i.e. it denotes the selection or rejection of a placement. Constraints programming can be considered on extension LP by adding restrictions on the number of VMs in a single PM, or by limiting the number of VM migrations, etc. Convex optimization refers to a special class of mathematical optimization problems that is inclusive of both least-squares and LP problems [119].

The general common issues in these algorithms are as follows:

- They need a long time to generate the optimal solution, depending on the number of constraints.
- They can only be effective in static VM consolidation scenarios, since they require the exact size and constraints of the VM.

2.5.2 Heuristic Algorithms

Heuristic algorithms are used to find step-by-step solution by identifying a local best decision. The majority of these algorithms are based on Bin-Packing. The NP-hard bin-packing problem principle is premised on the local best decision to pack a series of VMs having specified sizes into the least possible number of PMs [45]. Most approaches used in extant literature are premised on classic packing algorithms such as the First Fit (FF), the Best Fit (BF), the First Fit Decreasing (FFD), the First-Come First-Served (FCFS), and the Best Fit Decreasing (BFD) algorithms [120].

These algorithms can be classified into *Online* and *Offline* algorithms. In online algorithms, such as FF, VMs are assigned to PMs as they arrive. There is no need for prior knowledge of the VMs to be deployed in future. Offline algorithms possess the knowledge about all the VMs to be assigned, which is why they are able to sort them beforehand. In offline algorithms, such as FFD, VMs are assumed to arrive sequentially and are placed on the first PM which can accommodate them, beginning from the first PM sorted according to a predefined metric, power efficiency [16].

Beloglazov and Buyya [93] proposed a simple Power-Aware Best Fit Decreasing (PABFD) algorithm that sorts the VMs based on their CPU utilization in decreasing order. For each VM, it checks all the PMs and identifies a suitable PM where the increase of power consumption is minimum.

Attempts have been made to improve bin-packing algorithms, such as those by CloudSim [9], Chowdhury *et al* [45] and Farahnakian *et al* [121]. However, most of their algorithms have the following common characteristics [15, 120]:

- They are fast and need fewer computation resources because it is done by comparing the VM's demand with the server's available capacity, without considering the balanced

utilization of multidimensional resources.

- They can't guarantee optimal placement, but can be considered for immediate goals or suboptimal solutions.
- The resulting minimum number of PMs used will not necessarily be the solution for less energy due to corresponding lower power consumption and hardware dependencies.

2.5.3 Approximation Algorithms

These algorithms deal with solution where exact cost of resource is unknown, but the probability distributions can be estimated such that network bandwidth of the VM is arrived at as in [120]. Approximation algorithms are different from deterministic algorithms in that they cannot be implemented using mean or maximum of the demand as its estimated value.

As an example, Farahnakian *et al* [122] formulate a VM consolidation as a bin-packing problem considering both the current and future utilization of resources. The future utilization of resources was predicted using a KNN regression-based model. Their experimental results reveal that this approach provides an improvement over other heuristic algorithms in reducing energy consumption, a number of VM migrations, and a number of SLA violations.

Authors in [123] suggested a heuristic algorithm to ensure multi-dimensional energy-efficient resource allocation. More specifically, they created multiple copies of VMs and then used dynamic programming as well as local search to place these copies on the PMs. Local search attempted to reduce the cost of energy by shutting down the underutilized servers, while dynamic programming initially identified the number of VM clones to be placed on PMs. They minimized the length of networks connecting all PMs so as to minimize the total connection costs and reduce energy.

Dalvandi *et al* [124], reduce power consumption by maximizing the benefit from the

overall traffic send by VMs to the root through the proposition of a time-aware VMP-routing algorithm. In their approach, each VM requires a given number of network resources and server resources for a time duration. They formulated this problem as a mixed-integer LP optimization on the basis of a power utilization model.

The common characteristics of these algorithms are as follows:

- They do not need to be predefined constrained, because they depend on the probability of the parameters.
- They need less computation time than their deterministic counterparts but more than their heuristic counterparts.
- They are useful for dynamic VM consolidation.

2.5.4 Meta-heuristic

Meta-heuristic, sometimes called biology-based optimization, is a way to solve the bin-packing problem with certain constraints. These approaches are premised on biology-driven optimization techniques such as Genetic algorithm (GA) [125], Ant Colony Optimization (ACO) method [126], and Hybrid Genetic Algorithm (HGA) [25]. These algorithms require more computation time and greater computing resources as compared to heuristic algorithms [11].

Genetic algorithms, nondominated sorting GA I and II were compared with common solution representation [125]. The simulation shows that the nondominated sorting GA II gives good and wind range of solutions as compared to the former algorithms.

Tang and Pan [25] used an HGA for the energy-efficient VM placement problem on PMs with communication network consideration in data centres. They developed a Java program that can randomly generate VM placement problems of different configurations, fixed and

variable number of PMs with 20 and 80 random VMs. As per the experimental results, the HGA is more accurate than the original GA.

Feller *et al* [126] developed a multidimensional bin-packing to place VMs into the least number of PMs necessary for the current workload based on ACO.

Lopez and Baran [127] proposed three objective functions to apply multi-objective memetic in solving VMP problems, where the critical application was considered for a specific SLA. They concluded that, the number of solutions and execution time to find these solutions decreased by increasing the percentage of VMs with high SLA.

The VMP categorization in [128] suggested three optimization approaches: 1) the *mono-objective* approaches, which consider the optimization of one or multiple objectives one at a time; 2) the *multi-objective* approaches, which consider multiple objective functions fused into one objective function; and 3) the pure multi-objective approaches. Five objective functions have been identified: energy consumption minimizations, network traffic minimizations, economic costs minimizations, performance maximizations and resource utilization maximizations.

In Chapter 6 of this thesis, we have used meta-heuristic VMP algorithms that is capable of treating the multi-objective as a single objective optimization problem. These objectives may include power, QoS, network traffic, etc. The proposed VMP produces an optimal solution that is capable of satisfying different predefined constraints. This algorithm taking into consideration all PM characteristics, not focused only on workload characteristics through efficient utilization of a machine index factor. The proposed machine index able to take into consideration the heterogeneity of PMs in terms of power efficiency and capacity. This index substitute the operation of host underload and host overload processes.

Table 2.4: Comparison of VMP algorithms

Solution Category	References	Considered Resources	Aspect	Evaluation	Performance Better Than
Deterministic					
Linear Programming	[112]	CUP, storage and network	PMs' resources subject to linear function	Simulation	Non
Integer Linear Programming	[113]	network	Tree and forest formulated on graph	Simulation	BF
Constraint Programming	[114]	CPU	Objective functions for optimality	Simulation	BF heuristic
Constraint Programming	[115]	CPU and bandwidth	maximum link utilization optimization	Simulation	BFD and Random algorithms
Convex function	[116]	CPU	PMs meeting all tasks' demands	Google Trace data	BF with Min, Max and random
Heuristic					
Bin-packing	[129]	CPU, memory and network	Volume to size ratio	Simulation	FF, BF, FFD
Bin-packing	[45]	CPU and memory	Redesign	CloudSim	
PABFD	[97]	CPU	on-line	CloudSim	FF, BF, FFD
Enhanced FFD	[130]	CPU	VM reuse strategy	CloudSim	FFD and Round-Robin
PABFD + minimum correlation coefficient	[8]	CPU	PABFD with minimum correlation VMs	CloudSim	PADFB
Approximation					
Utilization Aware BFD	[122]	CPU and memory	Use VM and PM prediction	CloudSim	Modified BFD, Modified FFD
Utilization Aware BFD	[123]	number of network and server resources	Network connection	Simulation	FFD
Utilization Aware BFD	[124]	Number of VMs	Use VM and PM prediction	Simulation	CPLEX Optimization Studio [131]
Meta-heuristic					
GA	[127]	CPU, memory, bandwidth and storage	multi-objective formulation of the VMP	Itaipu Technological Park DC	brute force exhaustive search algorithm
Hybrid genetic algorithm (HGA)	[25]	CPU and network	scalable with problem size	Simulation	GA
Non-dominated Sorting Genetic Algorithm (NSGA)	[125]	CPU, memory and bandwidth	Non-dominated Sorting GA	Simulation	GA
Ant colony optimization (ACO)	[126]	CPU, memory and bandwidth	Modeling Multidimensional bin-packing	Simulation	multi-dimensional bin-packing
Ant Colony System (ACS)	[105]	CPU	Find near optimal	CloudSim	SLA violation and migrations
multi-objective ACS	[132]	CPU, network and storage	Vector-algebra based resource utilization	Simulation	single-objective ACO, FFD

2.6 VM Migration

Performing VM live migration in data centres is not a straightforward task. Several challenges need to be addressed, such as maintaining a reasonable level of QoS requirements and optimum resource utilization for energy conservation [128]. The two key criteria for an efficient VM migration are as follows: the *downtime* during the migration and the *migration time* itself [133]. Downtime denotes the time when services are down due to the migration process. Migration time refers to the time required to transfer a VM from one node to another within a cluster [134]. In both criteria, the objective is to minimize their values so that the migration process does not interrupt the provisioning process.

Different techniques have been used to execute live migrations. Some of the well-known techniques used include the following:

- ***Pure stop and copy***: in this technique, VMs are stopped, copying its content to the destination, before the VM can be restarted. This process is simple, but the service downtime could be large and is proportional to the allocated memory to the migrated VM [133].

- ***Post copy***: as per this technique, only essential data structures are transferred to the destination that can be restarted. Other parts are migrated on-demand across the data centre. This technique reduces the VM downtime, but the migration time still takes a long time [135].

- ***Pre-copy***: this technique involves iteratively copying memory from the source VM to the destination server while keeping the migrated VM running. An iterative process is performed to consider any updates that may have occurred in the migrated VM to make sure that updates are available at the destination server [136]. The main shortcoming

of this technique is network overload, particularly with high memory usage VMs.

- ***Hybrid***: this technique combines the pre-copy and post-copy algorithms. Besides transferring the virtual CPU registers and devices states in post-copy, a small subset of memory is also transferred, which is frequently accessed by the VM. The advantages of both the pre-copy and post-copy can be exploited in the hybrid algorithm, which makes it more suitable for VMs migrations [137].

Live migration cost energy and any reconfiguration should aim to reduce energy consumption. Therefore a key decision in VM migration should be on the process of selecting those VMs whose replacements to save at least as much energy as their migrations cost. In order to achieve such energy-efficiency, we need an efficient migration cost model that quantifies the energy overhead of VM live migration in advance. Linear regression techniques are usually used to model the energy overhead of live migration [138–141].

Akiyama *et al* [142] proposed to integrate a performance model as well as an energy model of live migration to simulate dynamic VM placement. The proposed performance model estimates how long a live migration takes under a given environment. The input is the size of the target VM, network bandwidth available for migration, and workload running on the VM. This energy model simulates dynamic VM placements and estimates how much energy is lost by performing live migrations to process dynamic VM placements. The input is a number of memory pages transferred during live migration.

The advantage of their approach is the combination of energy consumption models of the placement and migration operations since both operations complement each other in the data centre environments. Moreover, it can simulate pre-copy live migration, as it works perfectly as a pre-copy live migration by reusing non-updated memory in the initial memory transfer. However, their model's performance is not compared with the hybrid migration

technique where both the pre-copy and post copy algorithms are used.

In Chapter 6 of this thesis, migration cost can be added to the proposed PM index. Migration cost can be either a static or dynamic factor, and are considered as part of the total cost of the objective function of the VMP algorithm.

2.7 Network Effect

With increasing numbers of servers and switches in data centres, the communication bandwidth has to scale exponentially to meet the growing requirements of data access, processing and storage. Yang *et al* [143] highlighted that thousands of MapReduce programs are implemented and run in different applications such as Yahoo, Facebook, Google's data centres on a daily basis. In addition, petabytes of daily data flow are transmitted among distributed jobs within a data centre. This, in turn, incurs a very high energy consumption.

One way of dealing with this scenario is to find efficient and cost-effective approaches to tackle the network effect. In data centres, there are two main approaches for *network setup*: switches-centric and servers-centric [144].

- ***Switches-centric***: this approach implements the hierarchical network topology constructed from off-shelf components. Under this approach, servers are positioned in such a manner that they are at the leaves of the hierarchy of the network. The advantage of such approach is that it enables better load balancing and is less prone to bottleneck [145, 146]. However, the weakness is the limitation in terms of scalability owing to the size of routing tables in switches [144].

- ***Servers-centric***: this approach implements the Cayley graph [147]. As opposed to the first approach of switches-based network, the data centre network resides within

servers. It provides programmable capabilities and intelligent routing. Thus, servers are not only able to process applications and data, but also act as routers to relay traffic. The main advantages of this approach are as follows: 1) The low cost of interconnections in data centres besides the ability to remove the bottleneck at the architectural level. 2) Due to high scalability, expanding the network does not require physical modification or enhancement of existing servers.

Liao *et al* [144] presented DPillar, a highly scalable network architecture for data centres. DPillar is built with dual-port servers and n-port switches where server columns and switch columns are alternately placed along a cycle. One disadvantage of DPillar is that it was not designed to produce a short path routing on account of its simplicity. Erickson *et al* [147] improved the DPillar algorithm efficiency by developing a single-path routing algorithm that always produces the shortest path. Wang *et al* [148]proposed survey on different network topologies used in data centres and divided their networks into two parts:

1. Tree-based topologies: classified into Basic-tree, Fat-tree, and Virtual-layer 2 (VL2). It is primarily based on switch routing architecture.

2. Recursive-based topologies: classified into DCell, BCube, FiConn, FlatNet, and SprintNet. It is primarily based on server routing architecture.

Although Recursive-based topologies do offer a significant remedy to the problems of the Tree-based topologies, they still have their own shortcomings. DCell is built based on low-level and hence, may cause a bottleneck. BCube is considered a topology with high wiring complexity. FiConn suffers from deficiencies in fault tolerance, network capacity, and long path traffic. FlatNet and SprintNet have low scalability as compared to other topologies. The improved version of DPillar topology [144, 149] remedies these problems by providing

highly scalable network topology, good fault tolerance, improved bottleneck throughput and latency, as well as the shortest path routing within the data centre network.

2.8 Chapter Summary

This chapter presents a background, comprehensive survey and analysis on VM consolidation with an emphasis on the energy consumption in data centres. In particular, the chapter focused on proactive dynamic VM consolidations in data centres with heterogeneous environments. A general framework with multiple phases that achieves a complete consolidation process was also presented.

Below are some of the key shortcomings observed in extant literature, which will be addressed in this thesis:

- Most algorithms only consider the CPU as their primary input. In order to improve their performance, these algorithms should be extended to consider other important resources, such as memory, storage, bandwidth, etc.

- Algorithms cannot always be compared to one another, as they may take into consideration different inputs, operation criteria or goals [17].

- There is no systematic clustering that makes efficient use of the historical VM request, user behaviour, and data centre monitoring tool instant output in the clustering process.

- Prediction window size can play an important role in the workload prediction calculations as well as its accuracy. Additional work should focus on estimating the forecast's window size. This is important because the models used in existing literature are mostly based on predefined period.

- Identifying the behaviours of cloud users' requests resources strongly influences the overall cloud workload. Uncovering the dependency relationships between users and VMs helps improve the accuracy of predictions and excludes unwanted data.

- The capabilities of PMs in the data centre play a key role in the consolidation process. Existing research has mostly focused on workload characteristics. Therefore, more focus is needed on taking other PM characteristics into consideration.

- Most VMP algorithms compare their proposed algorithms against trivial heuristics. A comparison against real data can yield more meaningful results, which in turn, can result in improved algorithms.

- Simplicity assumption made in many algorithms, such as the homogeneity of PMs or ignorance of PMs' power consumption characteristics, lead to the degradation of their performance under realistic settings. In particular, the heterogeneity of PMs in terms of capacity and power efficiency must be considered when designing a VMP algorithm.

- There are different VM selection criteria, all of which have their weaknesses and strengths for different applications and specifications, Section 2.4. It is useful to have a rule-based system such as fuzzy logic in order to improve the process of selection between these techniques, in accordance with environment states [45, 150].

Chapter 3

A Real-Time Energy-Conserving VM-Provisioning Framework for Cloud-data centres

In this chapter, we provide a general elucidation and operation for a proposed energy-aware VM-consolidation framework components. Additionally, we define the importance and usage of the framework subsystem. Towards the end of this chapter, we will also explicate the data used in the implementation phase for each proposed algorithm and technique that has been used in this framework.

3.1 Introduction

This thesis proposes a framework to represent a real-time proactive VM-provisioning so as to reduce energy consumption in heterogeneous large-scale data centres. This framework takes into consideration data provided by monitoring tools, user behaviour, and historical

data to forecast the number of future VMs and to improve the consolidation process with minimum energy consumption.

In a proactive VM consolidation, monitoring, historical data, and prediction algorithms are used to alter the VM's initial placement before the system reaches a certain condition. The proactive provisioning uses prediction-based approaches that help prepare changes in advance in the workload and system usage [2]. In this thesis, we propose a pipeline real-time proactive provisioning framework in order to reduce energy consumption in large scale heterogeneous data centres, Figure 3.1. The proposed framework consists of: 1) an efficient data pre-processing and VM clustering subsystem; 2) a prediction subsystem; and 3) a VM placement subsystem. Additionally, the framework takes into consideration: a) data centre current states such that PMs states, networking, power, etc. and b) historical data to estimate user behaviour, VMs execution time, etc. These considerations are used to improve pre-processing, clustering, prediction and placement subsystems. The proposed framework satisfies the following:

- An efficient pre-processing subsystem that improves filtering and normalization processes for the historical as well as the monitoring data by extracting VMs with correlated behaviour.

- A systematic VMs clustering subsystem that handles big monitoring data with less consumption resources, faster computing, and improved accuracy. Furthermore, clustering subsystem output for both users and VMs will be used for excluding unpredictable VM requests caused by users' actions.

- An ELM-based prediction subsystem that can predict the number of VMs in each cluster. This predictor is also able to improve the forecast's accuracy by taking into consideration VM and user behaviour.

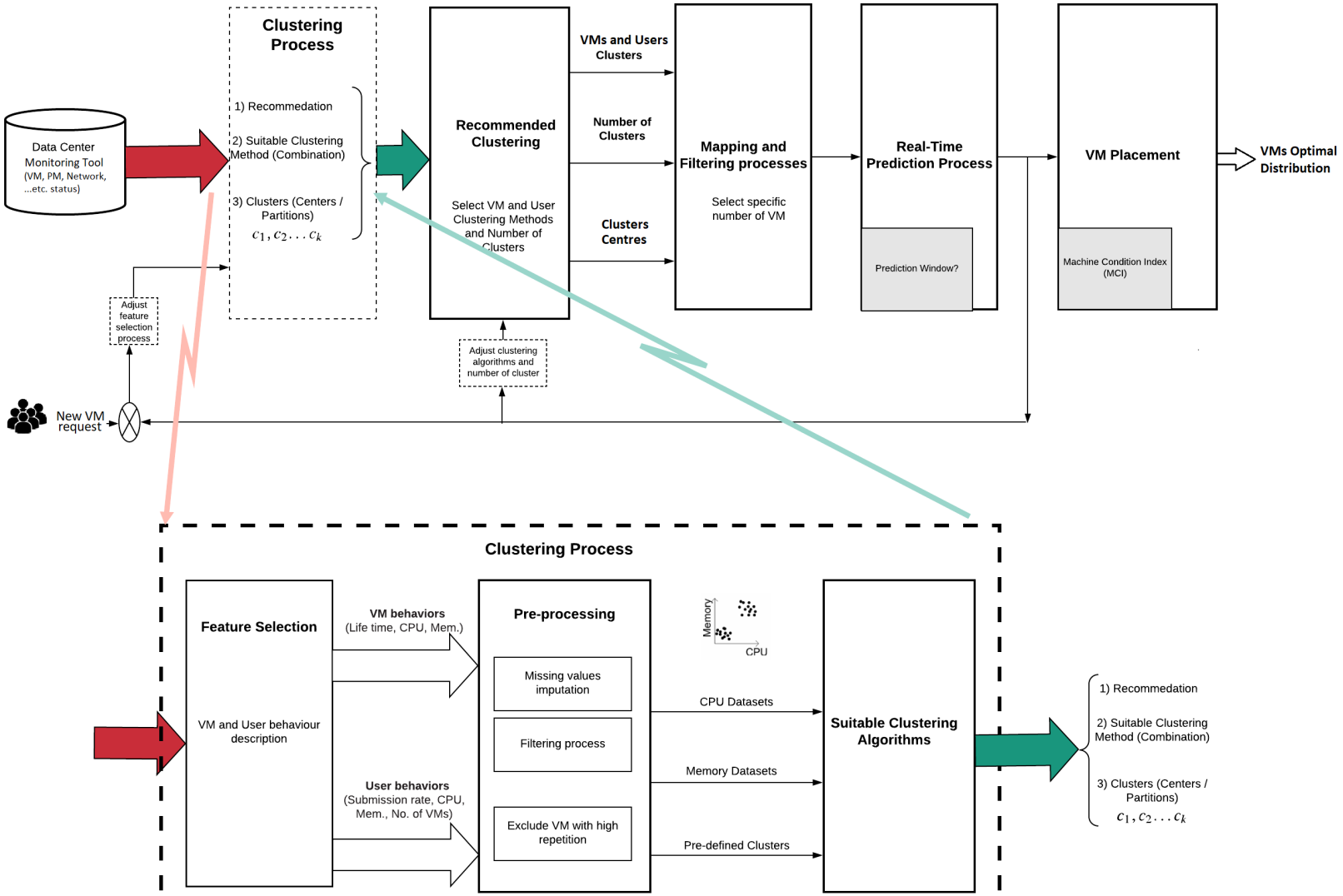

Figure 3.1: Proposed energy-aware real-time VMs provisioning framework

- A scalier unified index to model servers in heterogeneous data centres. The proposed index is a measure of the extent to which the PM is suitable to handle new or consolidated VMs.
- The proposed index converts the multi-objective function into a single objective function for real-time application with optimal displacement.

The proposed framework's operation can be implemented as a pipeline process that is summarized in the following steps.

1. First, historical and current state of data centre provided by a monitoring tool (e.g. DCIM) are used to identify cloud data centre states including PMs load, pre-defined group of VM features, user profiles.
2. Current state of the data centre, new requested and consolidated VMs delivered to the clustering subsystem. The clustering subsystem will:
    - Implement pre-processing, filtering and clustering processes based on updated features for VMs and users.
    - Clusters all VMs in accordance with the size and resource requirements.
    - Recommend clusters on the basis of clustering purpose, validation indices, and result comparison.
3. New and consolidated VMs, recommended VM and user clusters and these clusters' centres will be input to the prediction subsystem. The prediction subsystem will:
    - Map new and consolidated VMs based on new clusters' centres.
    - Use the findings of clustering processes for both VMs and users so as to filter low probability VMs with the nearest future.

- Rapidly forecast the number of VMs required for the next period.
- Estimate the number of PMs required on the basis of current and predicted VMs.

4. Using estimated number of VMs, PMs and the current state of PMs using MCI, the VM placement and MCI subsystem will:
   - Optimize the distribution of new and consolidated VM. The VMP is taking into consideration PM energy consumption, network, SLA, etc.
   - Pass the result to the scheduling unit to assign each VM request to the defined PM.
   - Each PM that does not accommodate VMs anymore power-cycles itself in order to save energy.
5. Adjust PMs usage by optimizing VMs execution time on each PMs (if possible).
6. Adjust the MCI for each PM, optimization factors, clustering indices factor. This will be done by providing the feedback between the framework blocks, Figure 3.1.

The feedback between blocks are used to interact with different parts of the framework. Put differently, the selection clustering index in the clustering process will impact the feature selection criteria.

In the remainder of this chapter, we will describe the basics of each block in the proposed framework in Section 3.2. The monitoring tools data as well as the simulated data used in thesis will be described in Section 3.3. In the last section, Section 3.4, a summary of the chapter will be provided.

3.2 Energy-Aware Framework Components

The target of the proposed energy-aware framework is to reduce the data centre energy usage through optimal and dynamic allocation as well as the reallocation of VMs by reducing the number of PMs used.

In this section, we will separately explain each component of the framework. Some of these blocks collectively denote a subsystem that can perform a complete specific task without needing others.

The last block in the framework represents power calculation and estimation. In this block, we expound on the manner in which we can achieve further power-saving improvements through: a) re-arranging the priority of VMs and b) reducing the number of migrated VMs with high power cost.

3.2.1 Feature Selection and Clustering Subsystem

The general clustering subsystem used in VM clustering along with user behaviours will be described in chapter 4. Clustering subsystem comprises of the following components, Figure 3.2:

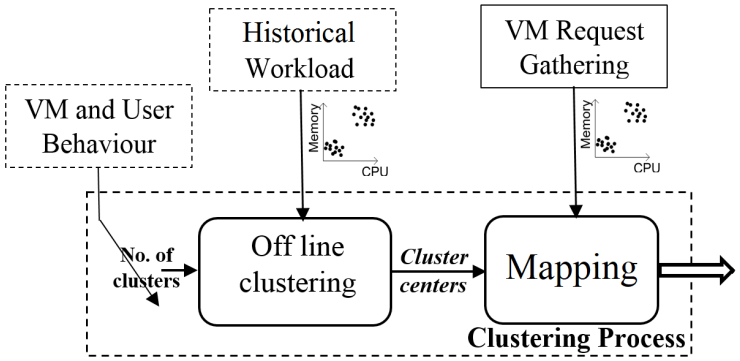

Figure 3.2: Clustering subsystem component

- **VM Request Gathering**: this includes cloud monitoring tool outputs. These out-

puts help detect the variations or failure of resources as well as applications during an observation.

- **Historical Workload**: this represents the historical data that is collected from a monitoring tool. This data is updated periodically and is used to forecast the next period VM request for each observation. It is also used to calculate centres of clusters on a period basis using long-term observations.

- **Off-line clustering**: this clustering is used to create a set of clusters for different types of VMs and users using historical data. These clusters are used to improve the systematic process to incoming requests. The number of clusters should be able to balance two conflicting objectives: (1) reducing errors and (2) maintaining low overhead.

- **User and VM behaviour**: this block is used to analyze both VM and user behaviours. The relationship between users and VMs is used to improve the accuracy of pre-processing, clustering and prediction.

- **Mapping**: this matching process is used to map each request received during a given observation time into one cluster in accordance with the calculated cluster.

3.2.2 Mapping and Filtering Process

The output of clustering subsystem is a group of VM clusters. The mapping and filtering process is to used to: a) map each VMs received during a pre-defined observation window into a specific cluster, for users as well as VMs; and b) to eliminate unexpected VMs from the workload estimation process by filtering low use VMs in the cluster of each user. More details about a proposed mapping algorithms will be presented in Section 5.3.

3.2.3 Prediction Process

The user-behaviour-driven filtered VMs clusters will be used to forecast the expected next period VMs requests. An elucidate description on the developed multivariate time-series ELM-based predictor will be mentioned in Section 5.6. The ELM-based predictor comprises of a single neural network for forecasting the number of VMs requested in each cluster. Under the the prediction subsystem, we use clustering technique not merely for VM requests, but also after receiving user requests. The filtering processes will used to filter unexpected VM requests caused by the unpredictable actions of users, Figure 3.3. The proposed subsystem of prediction will be used to resolve the following challenges:

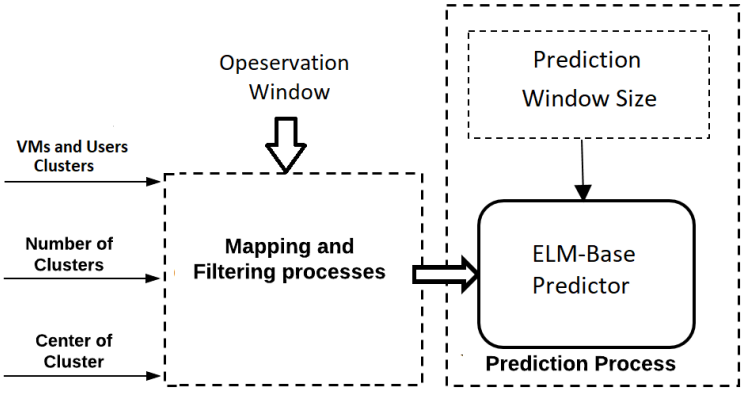

Figure 3.3: Recommended prediction subsystem

- In this subsystem, we include both user and VM variations in prediction process. Most related work focuses on VM variability only.
- We overcome the problem of time-varying VM requests.
- We eliminate the restrictions on observation window size and the number of VM clusters.

Any prediction process must take into consideration two key parameters: 1) the duration of the time period used, monitoring window, Section 5.4 and 2) the duration of the time

period in the future for which the workload needs to be predicted, forecast window, Section 5.5. We define both monitoring and prediction windows under the proposed prediction subsystem in order to make the forecast process more accurate and faster to calculate.

3.2.4 VM Placement Process

The process of mapping the VMs to the existing PMs is commonly known as the Virtual Machine Placement (VMP) problem.

The VM placement (VMP) denotes the process of mapping the VMs to existing PMs. In Section 2.5, we showed that heuristic or generally bin packing algorithms are most suitable for real-time application due to their pace and the fact that they require fewer computational resources. However, bin packing algorithms also provide a sub-optimal distribution of VM's. In our proposed real-time VM consolidation, a part of proposed index implementation, we use Best Fit Decreasing (BFD) algorithms for limited number of objectives (CPU usage only). Knowledge of the VM, which will be submitted in the future, will be used to pre-specify the number of PMs required. More details can be found in Section 6.5.2.

In this thesis, we will assign a pre-identified dynamic index for each server that will measure the extent to which the PM is suitable or handling new or consolidated VM(s). This new index, Machine Condition Index MCI in VM placement block of Figure 3.1, is a dynamic assessment rating various PM elements as well as its environments. This index constitutes the digital footprint of servers in data centres, details of which are provided in Section 6.2. This index will be used as follows:

- A condition index substitutes the operation of host underload and host overload process described in Subsection 2.3.1 and Subsection 2.3.2, respectively.

- An effective cloud measurement unit converts the multi-objective VMP to a single

objective VMP optimization problem for real-time purposes, Section 6.3

3.3 Data Monitoring and Thesis Simulated Data

The data centre monitoring tool is used to gather information about the running resources. It forms an essential part of any data centre. The collected data is then used to identify states of PMs, VMs, network, power usage, cooling system, users, etc. Under the proposed framework, data pertaining to VMs/tasks are collected not only to track volume of requests, but also to define VMs/tasks behaviour. Additionally, this work would use PMs and network information for optimizing the performance of VMP algorithm.

In the remainder of this thesis, we will use a real-world workload from the Google cluster traces [46] to simulate framework subsystems. We will utilize the data provided in the Task Event table, where each VM request is called a task and each VM submission/termination request is referred to as an event. This table has 144.6 million recorded observations, 672 thousand unique job IDs and 12583 unique machine IDs, Table 3.1 [151].

Table 3.1: Characteristics of Google traces

Trace Characteristic	**Value**
Time span of trace	29 days
Number of PMs	12,583
Number of observations	143M
Number of jobs	650K
Number of submitted tasks	25M
Number of users	925
Compressed size of data	39GB

The power information, which is not included in Google data, will be simulated and described in Subsection 6.5.3. Figure 3.4 (a) and (b) reveal the number of tasks request in each minute and hour, respectively, for one day (24 hours). The total number of tasks stood

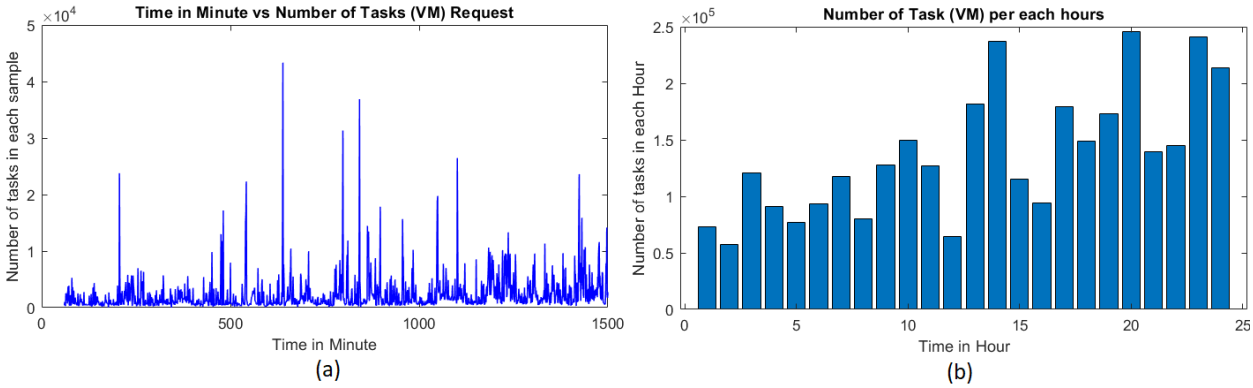

Figure 3.4: One day Google data with 3,295,896 tasks (a) task per min (b) tasks per hours

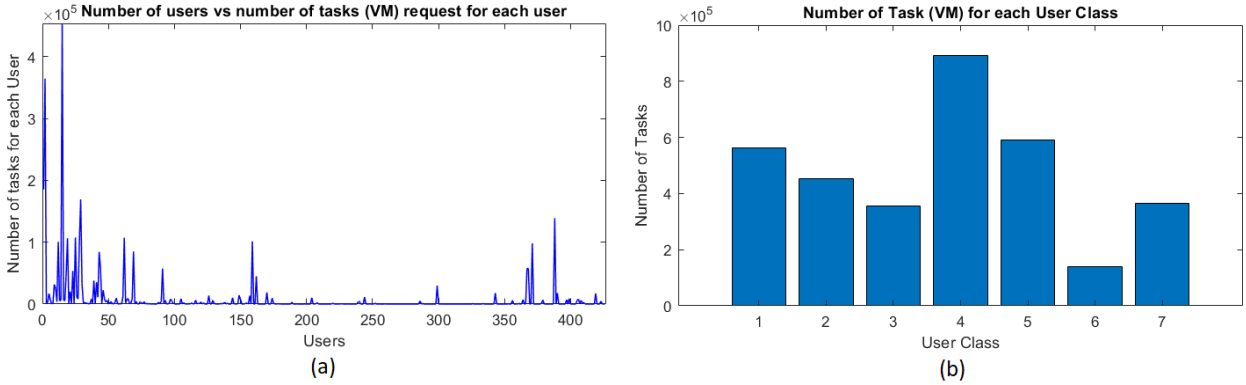

Figure 3.5: One day Google data with 426 users (a) tasks per user (b) task per user class

at 3,295,896, each of which has a value of CPU and memory. The number of users was 426. These users were then classified to seven types using k-means for simplicity, as illustrated in Figure 3.5 (a) and (b), respectively.

Furthermore, a randomly generated based Amazon EC2 instances types will be used for creating a heterogeneous environment and for simulating the proposed algorithms. These random Amazon EC2-based data will be used for partial clustering, Subsection 4.2.2, and VMP with multiple objectives, Subsection 6.5.3, where Google traced data does not encompass sufficient information.

For this thesis, all algorithms considered in the experiments have been implemented using

MATLAB on a Windows Operating System, with an Intel i5-6300U CPU at 2.4GHz and 8 GB of RAM.

3.4 Chapter Summary

In this chapter, we summarized the primary components of a real-time VM consolidation on the basis of the energy consumption framework. The most significant concept of this framework is to combine real-time processes including pre-processing, clustering, VMs and user behaviour analysis, filtering, prediction and VMP under a series of pipeline processes. This framework considers PMs index, characteristics model, VM execution time and networking as significant factors affecting scheduling and placement process. In the subsequent chapters, we will provide elaborate elucidations of the myriad approaches, algorithms and techniques used in the proposed real-time VM-provisioning framework.

Chapter 4

A Systematic Cloud Workload Clustering in Large-Scale Data Centres

In this chapter, we present cloud workload clustering categories as well as applications. In addition, we propose an efficient pre-processing and a systematic subsystem for selecting suitable VMs/tasks clustering algorithm in large-scale data centres. The systematic clustering is premised on clustering purpose, validation indices and result comparison.

4.1 Introduction

Cloud monitoring tools are usually used to provide information for the computing infrastructure and software resources. These tools generate a large volume of information about workloads in data centres, which, in turn, requires network bandwidth and hardware storage. In addition, the basic operations used for data monitoring, storing, analyzing processes and

retrieval operations can be both difficult and time-consuming [14, 152].

The workload clustering is usually used to investigate workload characterization in fields such as VMs/tasks scheduling, VMs allocation, and workload predictions. These characteristics represent resource usage, arrival process and workload pattern [153].

In this chapter, we propose a general cloud workload efficient pre-processing and systematic clustering subsystem for a large-scale data centre. This subsystem is useful in various cloud workload related applications such as monitoring, characteristics, modeling, prediction, etc. In the proposed subsystem:

- We develop an efficient cloud workload pre-processing algorithm for big monitoring data. The pre-processing efficiency consumes minimize computation resources, fast computing, and accurate results.
- We propose a systematic algorithm to select the suitable VMs/tasks clustering method along with the number of clusters in large-scale data centres.
- The systematic clustering algorithm recommends the best combination of VMs clustering techniques based on clustering purpose, validation indices, and result comparison.

In the next section, we will review clustering categories and reflect that on VMs/tasks clustering. In Section 4.3, we will present the proposed pre-processing and clustering subsystem for the large-scale data centre with practical recommendations. We will also objectively discuss the proposed validation techniques. Proposed systematic clustering will presented in Section 4.4. Finally, experiments and a summary of this chapter will be given in Section 4.5 and Section 4.6, respectively.

4.2 Cloud Workload Clustering

Clustering can be defined as an effective unsupervised learning technique. Clustering process groups naturally similar items with each other based on certain metrics. Clustering techniques combine objects into groups based on two criteria: (a) Homogeneity: the objects only contain data points that are member of a single class, and (b) Dissimilarity: there is a distance between objects belonging to two clusters under some criterion.

Various approaches have been utilized to infer cloud workload, more precisely VMs/tasks characterization and behaviour. A cloud data centre in VMs behaviours are traced by observing patterns that can be used to guide dynamic management decision, efficient monitoring, and prediction. The next subsections will provide a summary of clustering methods for which they get categorized into: *hierarchical*, *partitioning*, *grid-based*, d*ensity-based* and *model-based* methods.

4.2.1 Hierarchical Clustering

Hierarchical clustering refers to a set of nested clusters organized as a hierarchical tree (dendrogram). These techniques do not require a predefined number of clusters. Any desired number of clusters can be obtained by the tree at a proper level by merging similar clusters sequentially. This process is known as agglomerative hierarchical clustering.

Hierarchical clustering characteristics have been summarized in [154]: a) Select the level of clusters using dendrogram; b) Deterministic for linkage criteria, not optimal, greedy algorithm; c) Flexible with reference to linkage criteria; d) Slow and impractical, which is an N-cubed algorithm in most cases, and N-square for single linkage.

On the basis of these characteristics, hierarchical cluster algorithms are not recommended in cloud workload clustering. This is attributed to the fact that cloud workload is usually

dynamic/fast and has a copious amount of monitoring data. Balanced Iterative Reducing and Clustering Hierarchies (BIRCH), Clustering Using Representatives (CURE) and Chameleon are popular algorithms [155].

4.2.2 Partitional Clustering

Partitional clustering algorithms are used to simultaneously distribute a dataset into all clusters where the data are divided into non-overlapping clusters. Partitional clustering objectives are intended to determine the number of partition, k, in dataset. The VMs/tasks in these clusters are more similar to each other than to VMs/tasks in different clusters. For this reason, partitional clusters are the most adopted and useful clustering methods in terms of VM monitoring and VM behaviour prediction applications [2].

Under the partitioning methods, it is necessary to predefine the number of cluster k, which is often non-deterministic. This issue can partially succeed in addressing predefined instances. IaaS cloud providers offer various categories of predefined and custom VMs, with varying amounts of resources. For example, Amazon EC2[1] instances are categorized into five types. In EC2, there are more than 108 types of VMs along with different values in virtual CPUs (vCPU), memory capacity (GB), etc., as per a sample provided in Table 4.1. These predefined instances will enable service providers in the data centre to partially identify a number of required clusters.

In partitioning algorithms such as k-means, k-medoids, k-modes, PAM, CLARA, CLARANS and FCM [155, 156], no specific technique is used to select a number of clusters that can be generalized for cloud workload clustering applications. Different techniques and methods use prior knowledge of the application so as to choose initial values of k, or use predetermined indices [157, 158].

[1] http://aws.amazon.com/ec2

Table 4.1: A sample of typical general-purpose Amazon EC2 VMs

Instances	Model	vCPU	Mem (GB)	Storage	Net. Performance
T3	t3.nano	2	0.5	EBS-Only	
	t3.micro	2	1	EBS-Only	
	⋮			⋮	
	t3.2xlarge	8	32	EBS-Only	
T2	t2.nano	1	0.5	EBS-Only	
	t2.micro	1	1	EBS-Only	
	⋮			⋮	
M5	m5.large	2	8	EBS-only	Up to 3,500
	m5.xlarge	4	16	EBS-only	Up to 3,500
	⋮			⋮	
M4	m4.large	2	8	EBS-only	450
	m4.xlarge	4	16	EBS-only	750
	⋮			⋮	
	m4.16xlarge	64	256	EBS-only	10000

For example, in [154], they run hierarchical clustering on a subset of data in order to seek a good k. They used different values of k and evaluated the quality of results. through: (a) Selecting good initial centroids to ensure an appropriate distribution over the space; (b) Using multiple sets of randomly-chosen centroids, before selecting the best possible result.

In order to improve the initial selection of cluster numbers, a genetic algorithm can be used to estimate the initial selection of cluster centres, improved k-means++ [159], or even apply different distance metrics [160]. The number of clusters was determined using an evaluation graph that is depicted for k-means or FCM using a programmable logic array (PLA) [49], L-Method [161], or stability criteria described in [155].

4.2.3 Density-based Clustering

In density-based clustering algorithms, the VMs/tasks are separated on the basis of their regions of density, connectivity and boundary. Put differently, cluster can be defined as a

connected intensive component that grows in any direction that density leads to. These algorithms can be used to: a) Discover VMs/tasks clusters of arbitrary shapes; and b) Filter out or identify VMs requiring huge amounts of resources. OPTICS, DBCLASD, DENCLUE and DBSCAN are algorithms used in such clustering [162].

4.2.4 Model-based Clustering

The model-based clustering methods are used to optimize the fit between the given data and a predefined mathematical model. These methods are premised on the assumption that the data is generated by a mixture of implicit probability distributions. The number of clusters can be implicitly determined on the basis of standard statistics while taking noise into consideration.

Two commonly used model-based approaches include: *statistical* and *neural network* approaches. Expectation-maximization (EM) and MCLUST are almost best-known model-based statistical algorithms, while self-organizing maps (SOM) is a very well-known neural network (NN) approach [155].

4.2.5 Grid-based Clustering

In grid-based clustering, VMs/tasks monitoring datasets are divided into various grids. The computation is immediately done for fast processing. This is usually suitable for highly irregular data distributions to meet the required clustering quality and computation time. Wave-Cluster and STING are typical examples of this type of clustering [155]. Neural networks (NNs) have several properties that make them popular for such clustering because: a) NNs denote a parallel distributed processing architectures. b) Adjusting NNs interconnection weights for best data fitting makes it a best fit to normalize a prototype, where

patterns play the role of features. c) Extractors for various clusters. d) NNs process numerical vectors and require object patterns to be only represented by quantitative features. Many clusterings take care of only numerical data.

The pre-processing algorithm proposed in the next section will reduce dataset radically. For this reason, the grid-based clustering algorithms are recommended in the proposed systematic clustering algorithm.

Table 4.2 provides an overview of the clustering algorithm categories following the five classes of above-mentioned categorization.

Table 4.2: An overview of clustering categories

Clustering Categories	**Summary**	**Example**
Hierarchical	Permit clusters to have sub-clusters. Nested clusters organized as a tree	BIRCH, CURE and Chameleon
Partitional	Division of a dataset into non-overlapping clusters. Each data object is in exactly one subset	k-means, k-medoids, k-modes, PAM, CLARA, CLARANS and FCM
Density-based	Separation of dense datasets from each other by sparser areas	OPTICS, DBCLASD, DENCLUE and DBSCAN
Model-based	optimization of the fit between given dataset and a predefined mathematical model	EM, MCLUST and SOM
Grid-based	Division of the dataset space into grids	Wave-Cluster and STING

4.3 Proposed VMs/Tasks Clustering Subsystem

Cloud monitoring tools provide copious amounts of data with high density and variation [88]. The proposed VM clustering subsystem, Figure 4.1, divides this time-series big data problem into small processes of successive data clustering.

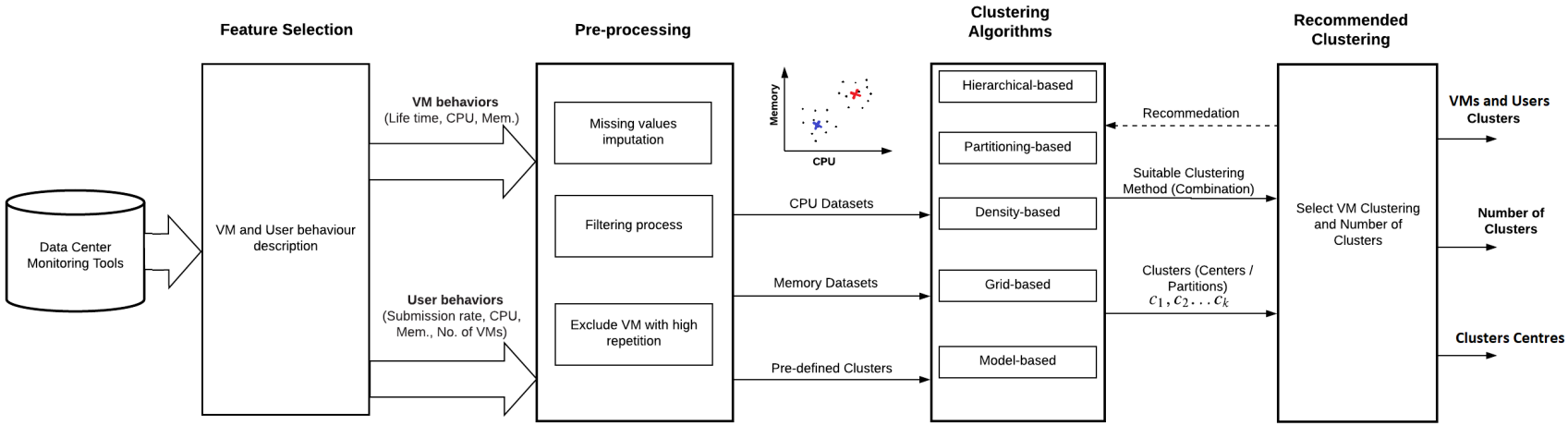

Figure 4.1: Proposed VMs/tasks clustering framework

In Figure 4.1, data centres monitoring tools represent data aggregate for new and consolidated (VMs/tasks) states. The collected information also pertains to running resources and users. In this figure, collected data represents VMs/tasks states. Algorithms and techniques are used to define VMs/tasks useful states and behaviour in this subsystem through the following stages:

- Feature selection stage,
- Pre-processing stage,
- Clustering stage, and
- Recommended clustering stage.

The next subsection will describe each stage in detail.

4.3.1 Feature Selection Stage

VMs and users behaviour must be properly selected to include as much information as possible [44]. The most important parameters that describe user behaviours in cloud workload include: CPU, memory requests, and VM/task submission rate. During VM/task lifetime, the average resource utilization for CPU and memory denotes essential parameters elucidating VM behaviour [51]. These values will be used to create the monitoring matrix.

4.3.2 Proposed Pre-processing Stage

The pre-processing stage is a cleansing process to remove VMs/tasks outliers, i.e. data objects with extreme values [156]. Proposed pre-processing that improve VMs clustering comprises of the following steps:

- **Missing Values Discarding**: the first step is to discard those instances that may contain a missing value. Most cloud monitoring systems can provide users and VM behaviour metric with high accuracy and availability. This will help prevent a bias in the clustering process.

- **Filtering Process**: the second step is the excluding process used to filter noisy VM related data. e.g. VMs with unexpected behaviour, size of VM greater than the host, VM with zero, and VM without CPU or memory values.

- **Exclude Correlated VMs**: the last step is to exclude VMs/tasks with high repetition for the same user or VMs with correlated behaviour. Correlated behaviour VMs/tasks, especially with the same user, denote a parallel processing request so that they can represent a separate cluster. These clusters are not required to include them in the clustering process; in other words they are clusters on their own.

These steps are represented in Algorithm 1. Inputs of this algorithm include the VMs/tasks represented by CPUs usage, memories usage, and user data represented by the number of VM/task per user. These values are obtained from any cloud monitoring tools. The output of the pre-processing stage is to have a 2D matrix with unique values of CPU and memory sizes, with each pair representing a VM/task.

Algorithm 1 Proposed pre-processing algorithm

Inputs: VMs/tasks and Users monitoring data
Outputs: 2D matrix (CPU and memory) each pair represents a unique VM/task

1: Normalize VMs/tasks data, and identify Users
2: Remove undefined users and terminated VMs
3: Remove VM/tasks without specified resources
4: Identified predefined instances (VMs/tasks) by a service provider
5: Exclude VMs with very high repetition for the same user or have correlated behaviour, separate clusters
6: Find the unique values of VMs (CPU and Memory)

4.3.3 Clustering Stage

As presented in Section 4.2, there are many existing clustering algorithms, but no single algorithm could handle all types of cluster shapes and structures. In addition, some challenges are associated with certain algorithms which are required to specify the number of clusters in advance, e.g. k-means, or need to set a threshold value, e.g. hierarchical. Most algorithms cannot automatically determine the number of clusters a dataset should be divided into.

The main advantage of the proposed clustering framework is to convert the problem of big monitoring data into a two-dimension matrix, including unique (i.e. not repeated) VMs/tasks data, and CPU/memory matrix. The two dimension matrix denotes the monitoring matrix for the prediction stage which helps us implement several expected clustering methods, or even combine multi-clustering methods with high efficiency, including computation, time and accuracy.

In this thesis, we selected the following algorithms as a candidate for cloud workload clustering:

- **Recursive k-means** and **k-means++**: the k-means++ is similar to k-means with better initialization procedure, where observations far from existing centroids have higher probabilities of being chosen as the next centroid. The initialization procedure is achieved using fitness proportionate selection.

- **Iterative Self-Organizing Data Analysis (ISODATA)**: this algorithm is essentially a splitting and merging operations. If the number of observations within one cluster is found to be less than one predefined threshold, two clusters are merged with minimum between-class distance. The splitting into two different sub-clusters when the variance of one cluster exceeds one predefined threshold.

- **Mean Shift**: this is a non-parametric model algorithm. Mean shift is used to find neighbors for each point in a dataset, after which mean vector is calculated until the point equals the mean. There is no need to pre-specify the number of clusters.

- **Density-Based Spatial Clustering of Application with Noise (DBSCAN)**: this algorithm is used to extend each cluster based on the connectivity between data points, taking noisy data into consideration. The main drawback of this algorithm is the selection of good parameters, which can difficult without prior knowledge.

- **Gaussian Mixture Model (GMM)**: this is an iteration process used to estimate parameters, mean and variance, as well as to generate a mixture of a pre-identified number of Gaussian distributions.

- **Self-Organizing Map (SOM)**: this algorithm is an unsupervised NN that uses a neighborhood function to preserve the topological properties of the input space.

- **Expectation-Maximization (EM)**: this is an iterative approximation used to create a statistical model with maximum likelihood parameter estimation.

4.3.4 Recommended Clustering Stage

Recommended clustering stage is the process of selecting the most suitable clustering algorithm to be used for a specific application. The recommended clustering algorithms are based on:

- **Clustering Purpose**: the clustering purpose denotes the target of the clustering algorithm used. The workload is used as an example to guide dynamic management decisions, efficient monitoring [49], prediction [39, 41, 153], etc. On the basis of the

required clustering purpose, we can decide, for example, if the clusters should be totally separated or an overlapping is acceptable.

- **Compare Results**: Comparing clustering algorithms can be done by implementing several clustering algorithms on the same data. These clustering algorithms may implemented separately or in combination. The selection is based on minimum error and clustering indices, as described in a separate subsection, Subsection 4.3.5.

- **Internal Clustering Validation**: the clustering validity indices usually operate as fitness functions to evaluate the clusters' quality. The internal index uses quantities and features inherited from the dataset to evaluate the goodness of a data partition [163]. Like Compactness (CP); Separation (SP); Davies-Bouldin Index (DB); Dunn Validity Index (DVI). More details about such indices can be found in [157,158,160]. For prediction proposes and due to the fact that a VM/task monitor data will be used, a selected internal index will be used for examining the clustering methods and identifying the best number of clusters.

4.3.5 Selected Validation Indices

The target of cluster algorithms in this subsystem is about identifying groups objects (VMs/tasks and users). In this subsection, we shall present the proposed candidate internal indices that are observed to be more suitable in cloud workload clustering. The variables and notations used in this subsection are defined in Table 4.3 for the sake of simplicity.

Calinski-Harabasz (CH)

This index evaluates the cluster validity on the basis of the average between and within-cluster sum of squares. The CH index is based on a ratio between cluster scatter matrix

Table 4.3: General Notation

Variable	Explanation
N	: Number of observations, $VM_1, VM_2, ...VM_N$
p	: Observation elements, each of which consists of (CPU, memory, etc.)
A	: Data matrix, observation matrix, size of A is $N \times p$, each value is represented as A_{ij}
C	: Set of clusters $C = \{C_1, C_2, \ldots, C_K\}$, data set within this cluster represented by $X_{C_k}(i,j)$
K	: Total number of clusters, $(1 \leq k \leq K)$
c	: Centres vector of the clusters, if any, $c = \{c_1, c_2, \ldots, c_K\}$,
NC_k	: Number of points in cluster C_k
μ_k	: Barycenter of observation in the cluster C_k
X_{C_k}	: Data points belonging to cluster C_k

(BCSM) and within-cluster scatter matrix (WCSM), as shown in Equation 4.1:

$$CH_k = \frac{BCSM}{K-1} \cdot \frac{N-K}{WCSM} \tag{4.1}$$

The $BCSM$ is based on the distance between clusters, Equation 4.2:

$$BCSM = \sum_{k=1}^{K} NC_k.d(c_k, \mu_k)^2 \tag{4.2}$$

where c_k denotes the centre of cluster C^k, NC_k represents the number of points in cluster C_k, μ_k centroid of all the data points and d signifies the Euclidean distance. The $WCSM$ is given in Equation 4.3:

$$WCSM = \sum_{k=1}^{K} \sum_{X_{C_k}(i,j)} d(X_{C_k}(i,j), c_k)^2 \tag{4.3}$$

where $X_{C_k}(i,j)$ denotes a data point that belongs to cluster C_k. In order to obtain well separated and compact clusters, $BCSM$ is maximized and $WCSM$ minimized. Therefore, the maximum value for CH indicates a suitable partition for the data set.

Davies-Bouldin (DB)

Davies-Bouldin (DB) uses the concepts of dispersion of a cluster and dissimilarity between clusters. The DB index is among the best factor of internal indices [164, 165]. It is capable of identifying cluster overlap by measuring the ratio of the sum of within-cluster scatters to between-cluster separations. The DB index is defined in accordance with Equation 4.4:

$$DB = \frac{1}{K} \sum_{i=1,j=1}^{K} \underset{j=1,\ldots,k,i\neq j}{max} \left[\begin{matrix} diam(C_i) + diam(C_j) \\ d(c_i, c_j) \end{matrix} \right] \tag{4.4}$$

Where *diam* denotes the diameter of a cluster given in Equation 4.5:

$$diam(C_k) = \sqrt{\frac{1}{NC_k} \sum_{X_{C_k} \in C_k} d(X_{C_k}(i,j), c_k)^2} \tag{4.5}$$

DB close to 0 indicates that the clusters are compact and far from each other.

Dunn Index (DI)

The Dunn index (DI) is defined as the quotient of smallest distances d_{min} between the closest points of two clusters (e.g. C_k and $C_{\acute{k}}$) and the largest distances d_{max} separating distinct points in a cluster (e.g. C_k), Equation 4.6.

$$DI = \frac{d_{min}}{d_{max}} \tag{4.6}$$

where:

$$d_{min} = \underset{k\neq\acute{k}}{min}\ d(X_{C_k}(i,j), X_{C_{\acute{k}}}(i,j)) \tag{4.7}$$

Where the largest distances separating distinct points in a cluster (e.g. C_k) are calculated as follows:

$$d_{max} = \max_{X_{C_k}(i,j) \in C_k, i \neq j} d(X_{C_k}(i,j), X_{C_k}(i,j)) \tag{4.8}$$

$X_{C_k}(i,j) \in C_k$ and $X_{C_{\acute{k}}}(i,j) \in C_{\acute{k}}$. The DI index used to determine the number of clusters by maximizing its value.

Silhouette Index (SI)

The Silhouette Index (SI) is used to validate the clustering performance based on the pairwise difference between and within-cluster distances. This is done by computing the width for each point used to identify membership of this point in any cluster. The Silhouette width is an average used for all observations, as illustrated in Equation 4.9:

$$SI = \frac{1}{K} \sum_{k=1}^{K} \sum_{r=1}^{NC_k} \frac{b_r - a_r}{max(a_r, b_r)} \tag{4.9}$$

where a_r : the average distance between point r and all other points in its own cluster:

$$a_r = \frac{1}{NC_k - 1} \sum_{r \neq j}^{NC_k} d(X_{C_r}, X_{C_j}) \tag{4.10}$$

b_r : the minimum of the average dissimilarities between r and points in other clusters.

$$b_r = \frac{1}{NC_k} \min_{r \neq j} \sum_{r \neq j}^{NC_k} d(X_{C_r}, X_{C_j}) \tag{4.11}$$

The optimal number of clusters is determined by maximizing the value of the SI index.

Wemmert-Gancarski (WG)

The Wemmert-Gancarski (WG) index is used to find quotients of distances between the points that belong to a cluster and the barycenter of all the clusters, Equation 4.12:

$$WG = \frac{1}{N}\sum_{k=1}^{K} max\left\{0, NC_k - \sum_{k\neq\acute{k}} \frac{d(X_{C_k}, \mu_k)}{\underset{k\neq\acute{k}}{min}\ d(X_{C_k}, \mu_{\acute{k}})}\right\} \tag{4.12}$$

where μ_k denotes the Barycenter of observation in the cluster C_k and X_{C_k} represent data points belonging to cluster C_k. It is possible to obtain optimal number of clusters through the maximizing of the WG index.

Xie-Beni

The Xie-Beni index is the quotient between the mean quadratic error and the minimum of the minimal squared distances between the points on the clusters. Let the minimal squared distances $\delta(C_k, C_{\acute{k}})$ between the points in the clusters C_k and $C_{\acute{k}}$ be given as:

$$\delta(C_k, C_{\acute{k}}) = \underset{k\neq\acute{k}}{min}\ d\left(X_{C_k}(i,j), X_{C_{\acute{k}}}(i,j)\right) \tag{4.13}$$

where $\delta(C_k, C_{\acute{k}})$

The Xie-Beni index can be written as:

$$\textbf{Xie-Beni} = \frac{1}{N}\frac{WCSM}{\underset{k\neq\acute{k}}{min}\ \delta(C_k, C_{\acute{k}})^2} \tag{4.14}$$

The Xie-Beni index is an index of fuzzy clustering. However, it is also applicable on crisp clustering. The mean quadratic error, in the case of a crisp clustering, is simply the quantity 1. The optimal number of clusters can be obtained by minimizing the Xie-Beni index.

Table 4.4 depicts the proposed candidate indices that can be used for different clustering application. In this table, decision rules called *max* and *min* mean that one should select the greatest or the smallest index value, respectively. Some candidate clustering algorithms, such as DBSCAN, do not have centroids, which is why some internal indices are not applicable

to DBSCAN algorithms [166].

Table 4.4: Recommended Validation Indices

Name	**Rule**	**Description**
Calinski-Harabasz (CH)	*max*	is the average between and within-cluster sum of squares.
Davies-Bouldin (DB)	*min*	identifies clusters that are far from each other and compact.
Dunn Index (DI)	*max*	combines dissimilarity between clusters and their diameters.
Silhouette Index (SI)	*max*	computes a width for each point, depending on membership in any cluster.
Wemmert-Gancarski (WG)	*max*	represents distances between dataset points and the barycenter of all the clusters.
Xie-Beni	*min*	is the quotient between the mean quadratic error and the minimum of minimal squared distances between the points on the clusters.

4.4 Proposed Systematic VMs/tasks Clustering

The clustering quality is depend on several factors:

- ***Evaluation target***: the required target represents the characteristics used for selecting suitable indices. For example, the DB index used to evaluate intra-cluster similarity. In the event this is what the target considers, then the DB index will be considered to be good criteria. As another example, SI index may be used in the prediction process in this work because it measures the distance between one point and another in both same ad different clusters. The same holds true for all quality measures; indices should relevant to the required clustering goal.

- ***Normalization***: this process is to consider whether or not these quality measures are normalized. For instance, the SI is normalized and a value close to 1 is always

considered good for evaluated clusters. However, the DB for instance is not normalized and it is difficult to compare two values of DB from different data sets.

- ***Matching metric***: the matching metric is to make sure that the applied quality measure uses the same metric that the clustering algorithm results. Otherwise, this quality measure will be biased.

Algorithm 2 summarizes the experimental steps in a systematic way for selecting suitable VM/task clustering techniques and the number of clusters for VM/task prediction purposes.

Algorithm 2 Proposed systematic VMs/tasks clustering for prediction purposes

Inputs: 2D matrix from Algorithm 1 during a monitoring period
Outputs: Best clustering algorithm and the number of clusters for unclassified VMs/tasks

```
Apply DBSCAN with a threshold value
Extract VMs/tasks large resources (CPU,memory)
for Each clustering algorithm do
    i ⟵ 0
    for i ≤ Max. no. of clusters do
        Find clusters and centre of clusters
        i ⟵ i + 1
    end for
    for Each clustering indices do
        Calculate index value for the number of clusters
        Find the optimal number of cluster based on index rule
    end for
    Measure the similarity with predefined VMs/task classes (algorithm and index)
end for
Choose the best match pair using the pairwise distance
Select a clustering algorithm
Select the number of clusters
```

Before using the algorithm, an offline recursive algorithm is used to cluster the predefined classes and to compute the optimal number of classes for each candidate clustering algorithm. This will be done based on candidate indices, Subsection 4.3.5. The outputs are compared with the service provider VMs/tasks predefined classes and the comparison process is repeated we get a minimum error between real classes and processed classes. Put

differently, the offline algorithm will be used to create a matrix concerning the clustering technique versus the optimal number cluster based on validation indices.

Algorithm 2 shows the experimental procedures used for selecting suitable VMs/tasks clustering techniques as well as the number of clusters for new and migrated VMs/tasks. In this algorithm, we used a predefined VMs/tasks classes from a cloud service provider as a guide to cluster unknown VMs/tasks.

For each pre-processed unknown VMs/tasks dataset during a monitoring period, a DBSCAN algorithm will be applied first with a low threshold value. This algorithm is used to extract the VMs/tasks that may require a large PM. The second step is to find the optimal number of clusters of each candidate clustering algorithm, Subsection 4.3.3. This will be done by identifying the number of clusters based on the optimal number of indices. The last step is to find the similarity between the two matrices using pairwise distance. Both the candidate cluster algorithm and the number of clusters are premised on the best match between the two matrices.

This algorithm ensures the balance between conflict objectives in selecting number clusters: (1) reducing errors and (2) maintaining low overhead. When compared to other related to the same data, e.g. in Google workload trace, Xia *et al* [167] and Rasheduzzaman *et al* [168] chose $k = 6$ and 5, respectively, for the k-means clustering algorithm. Xia and Rasheduzzaman depend on the minimum value of k in order to reduce error. They did not factor in the effect of increasing k on the performance of the predictor. This problem was discussed by Dabbagh et al [41,65], who suggested choosing $k = 4$ for the same data sample. Meanwhile, the best selection of Moreno *et al* [51] was $k = 3$ because he included users behavioural patterns.

4.5 Experimental Evaluation and Comparison

The Google trace dataset was described in Section 3.3 for one day (24 hours) in our experiments. The first step after collecting VMs/tasks and users' information, is to apply the proposed pre-processing process, Algorithm 1. This includes normalizing the data, removing undefined users and terminated VM/task, identifying the predefined instances and excluding VM/task with high repetition for the same user, or VMs with correlated behaviour. The suggested that pre-processing algorithm lowers the dataset from 3.3M VMs/tasks, Figure 3.4 to 2000 unique and independent VMs/tasks only, Figure 4.2.

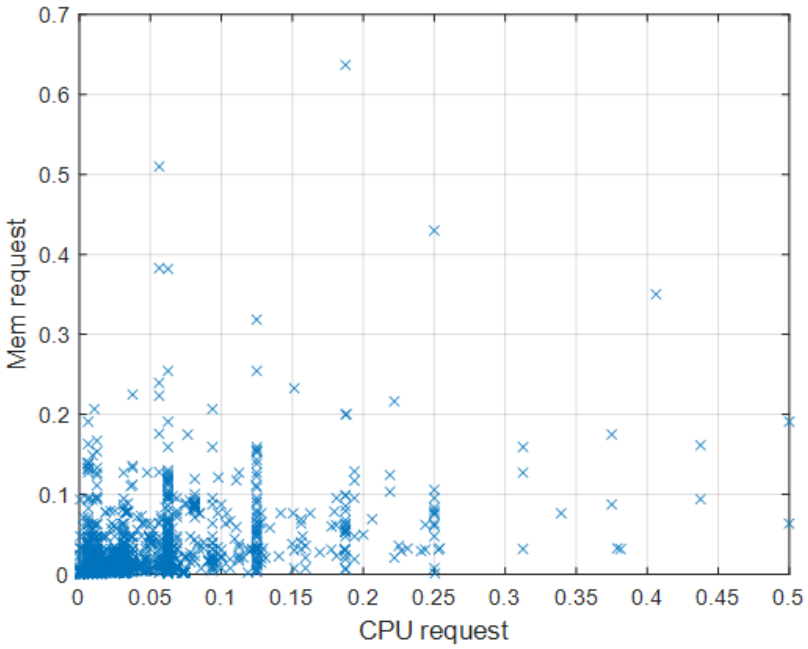

Figure 4.2: 2000 Unique and independents VMs for 24h

The proposed pre-processing converts the very large collected monitoring data into a simple 2D matrix, CPU versus memory usage for each VM/task. The small 2D matrix allows us to implement and combine multi-clustering techniques with high accuracy and minimum time. This is particularly significant for real-time application. Table 4.5 shows the runtime on the same machine, as described in Section 3.3 of 2 candidates clustering

algorithm, for example, k-means and FCM concerning a different number of clusters before and after the proposed pre-processing algorithm. According to our observations, the runtime for k-means and FCM clustering are reduced from 1 to 3 order of magnitude on average, respectively.

Table 4.5: Runtime sample of two candidate algorithms

Clustering Technique	***k-means***			**FCM**		
Number of Clusters	3	4	5	3	4	5
Without pre-processing (sec)	2.605	3.546	2.78	44.16	60.601	80.131
With pre-processing (sec)	0.209	0.212	0.219	0.053	0.056	0.062
Processing Speed Increase	12	17	13	833	1082	1292

Also, mapping of new monitoring data to each cluster becomes more separable. Separated clusters will make prediction more accurate in determining types and sizes of predicted VMs/tasks. Figure 4.3 (a) and (b) illustrate the results for 4 clusters using k-means, where each category is marked by a different color and centres of these clusters are marked by **"x"**, for the data with and without pre-processing, respectively. With pre-processing, the clusters become increasingly separable and the centres are spaced out.

With the reduced number of data, comparison, and analysis, the effect of the number of clusters now is convenient, within a short period of time. The Sum of Squared Distances (SSD) is plotted as a function of the number of clusters. SSD represents the error when each point in the data set is represented by its corresponding cluster center, as illustrated by the following equation [39]:

$$SSD = \sum_{i=1}^{\mathbf{k}} \sum_{r=C_i} d(r, c_i)^2 \tag{4.15}$$

where C_i denotes the cluster i, i.e., the set of all points belonging to the i^{th} cluster. $d(r, c_i)$ is the Euclidean distance between r and c_i.

Figure 4.4 illustrates the comparison between k-means and FCM for different numbers of clusters. Although the FCM algorithm needs long off-line time, we observed that it yields better results than the k-means based method. This leads to choosing a fewer number of clusters, which will

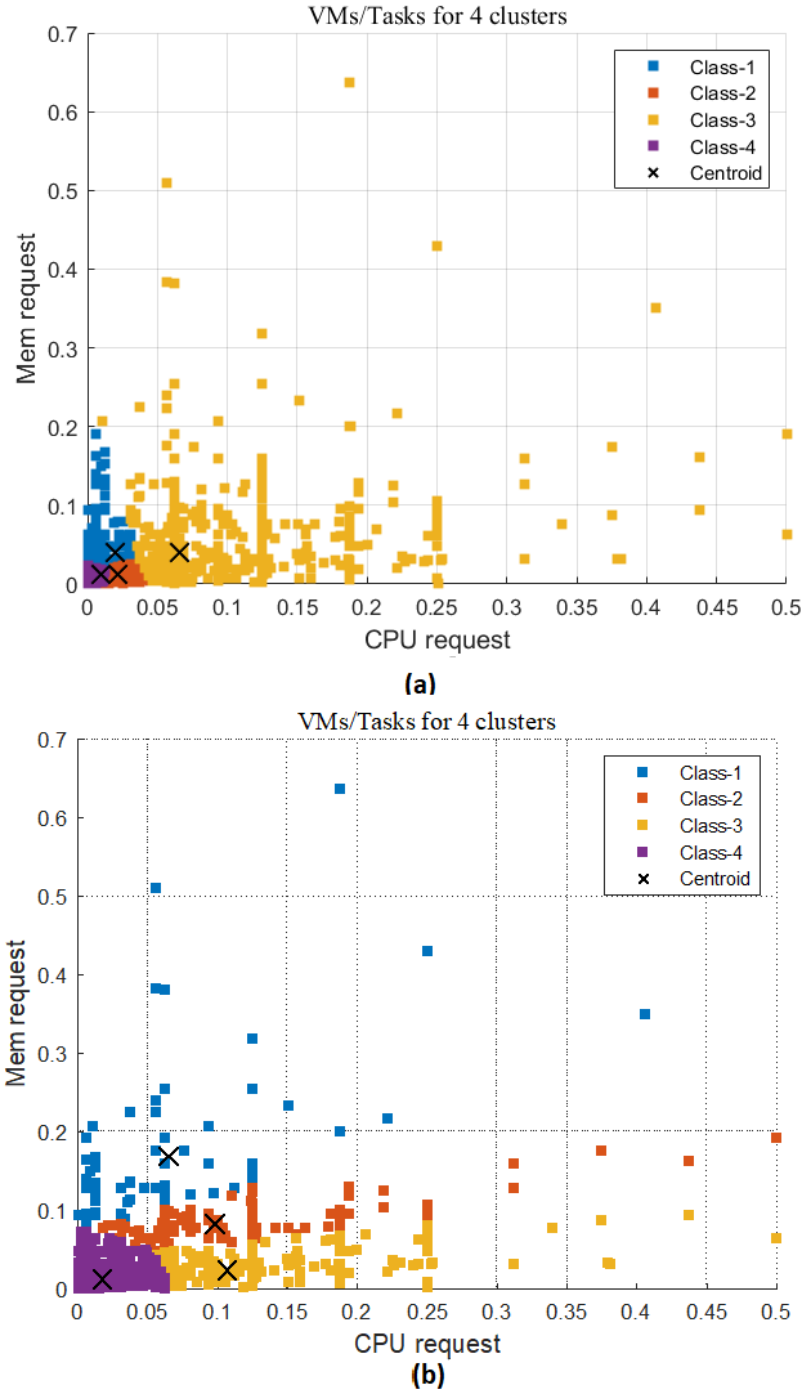

Figure 4.3: Resulting 4 VM/task clusters for 24 hours using k-means (a) Without (b) With pre-processing

affect the overall performance of the proposed framework for a cloud data centre based on energy consumption by finding a balance between reducing errors and maintaining low overhead.

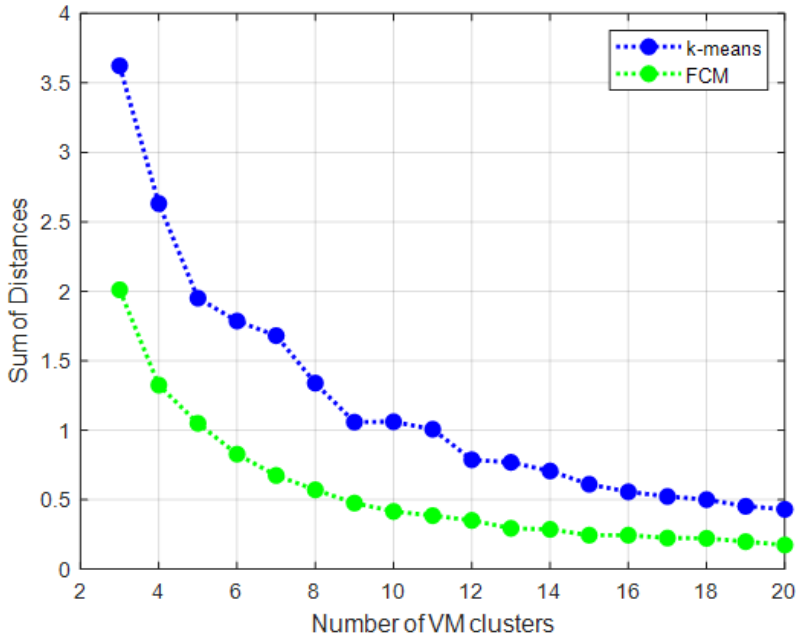

Figure 4.4: Number of VM clusters vs the sum of square error

Table 4.6 presents the results of the candidate clustering methods with reference to the candidate internal validity measures. This table has been prepared by implementing all candidate clustering algorithms for a different number of clusters (up to 15 clusters). For example, Figure 4.5 and Figure 4.6 represent CH index and WG index evaluations for a different number of clusters.

In Table 4.6, it can be observed that for algorithms that do not need to identify a number of clusters, such as DBSCAN, we changed the input to obtain a different number of cluster. For example, Figure 4.7 represents 2 VMs categories for the same data using DBSCAN clustering for epsilon = 0.03 and minimum points per cluster equal to 3. For VMs with large CPU and/or memory in DBSCAN is recommended especially in VM consolidation applications, as mentioned in Algorithm 2. It recommends keeping these large VMs in the same host due to the cost of moving the big VM.

In Table 4.6, the number of the cluster in each clustering algorithm has been identified based

Table 4.6: Candidate internal validity indices for selected clustering methods

Clustering Algorithms		Measures								
		BH	**BR**	**CH**	**DB**	**GD**	**BHG**	**SI**	**WG**	**Xie-Beni**
Kmeans	Value	0.0122	-18957.8	1421.5	0.8855	0.006	0.9332	0.4644	0.747	158.9
	Best	4	14	6	6	1	5	1	1	1
Kmeans++	Value	0.0102	-17075.3	1952.5	0.7236	0.0064	0.9731	0.4705	0.7512	43053.8
	Best	4	11	9	1	8	13	3	3	3
Hierarchical	Value	0.0014	NaN	91.2	0.0732	0.2901	0.9997	NaN	0.921	0.1299
	Best	2	1	1	1	1	1	NaN	1	1
ISODATA	Value	0.0062	-18912.7	805.2	1.215	0.0003	0.6518	0.2226	0.4279	84082.7
	Best	5	13	5	9	2	5	2	2	2
MeanShift	Value	0.0231	-11737.2	296.2	0.7384	0.0091	0.9583	0.5704	0.7505	112.2
	Best	6	12	7	10	12	10	11	10	4
DBSCAN	Value	0.0303	-10977.3	452.4	0.0732	0.2901	0.9997	0.6998	0.921	0.1299
	Best	2	5	2	6	6	6	5	6	6
GMM	Value	0.0231	-11737.2	296.2	0.7384	0.0091	0.9583	0.5704	0.7505	112.2
	Best	6	12	7	10	12	10	11	10	4
SOM	Value	0.0066	-20000.4	1717.7	0.6897	0.0005	0.9579	0.4663	0.7484	199.3
	Best	3	9	3	13	5	5	3	2	4
EM	Value	0.0052	-13841.5	1195	0.9899	0.0022	0.8149	0.3708	0.6322	1370.8

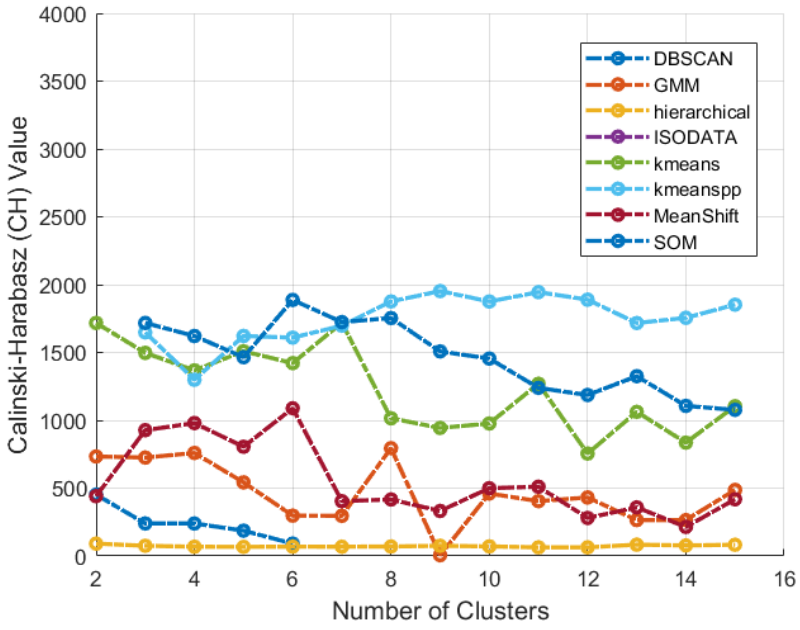

Figure 4.5: CH index evaluations for a different number of clusters

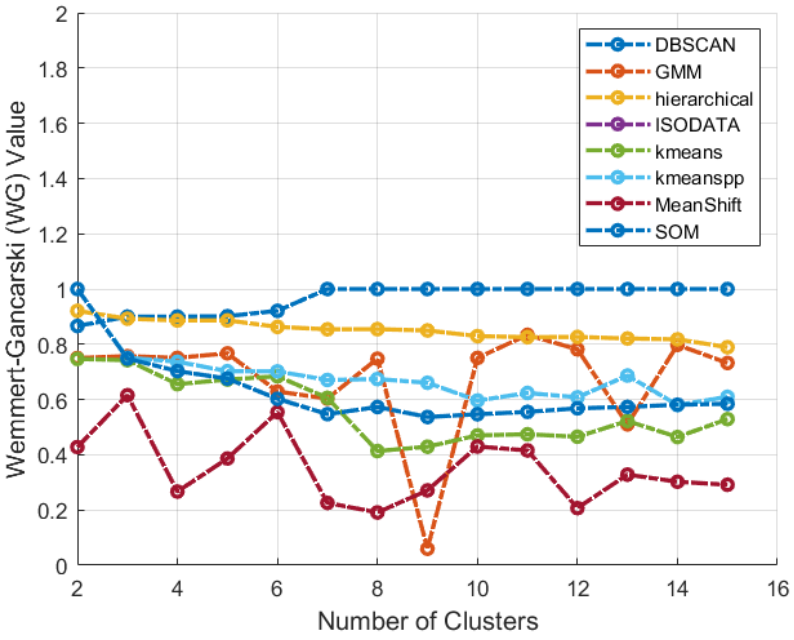

Figure 4.6: WG index evaluations for a different number of clusters

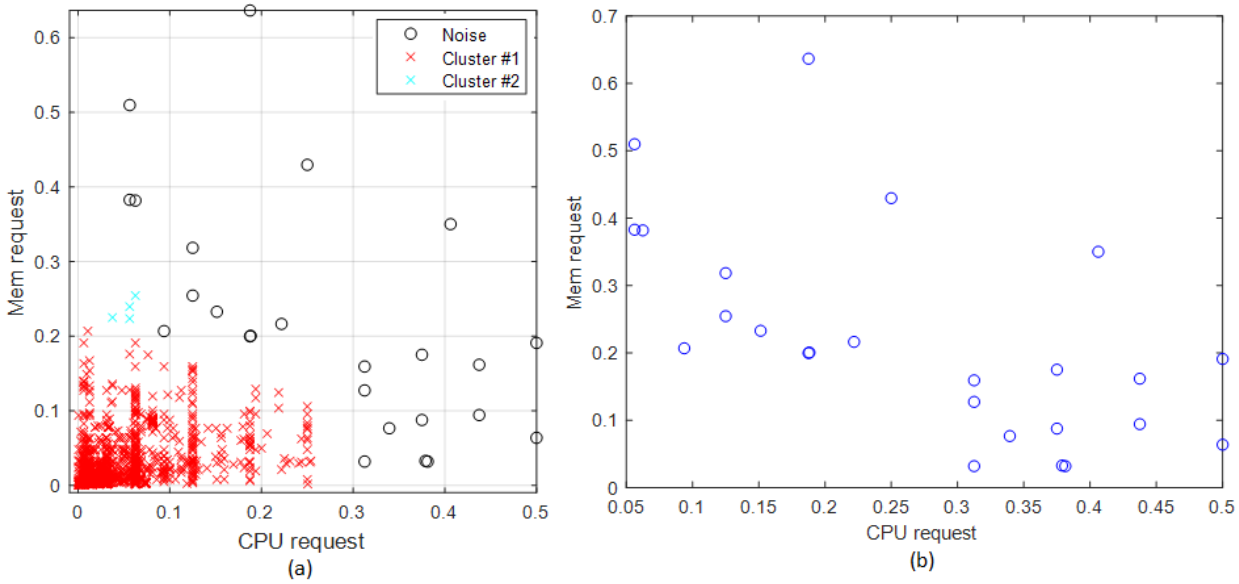

Figure 4.7: DBSCAN clustering (a) Resulting 2 categories (b) Resulting big VMs

on the rule of the indices. This table is essential in decision making for selecting the number and type of clustering method to be used in the specific application. There is no prior knowledge about the number of clusters required.

The last row in Table 4.6 represents efficient results of EM clustering with two clusters only, where its performance has been improved through statistical estimation of the initial condition for both the mean and variance of CPU and memory distribution using Kernel density estimation, described in [169], as shown in Figure 4.8.

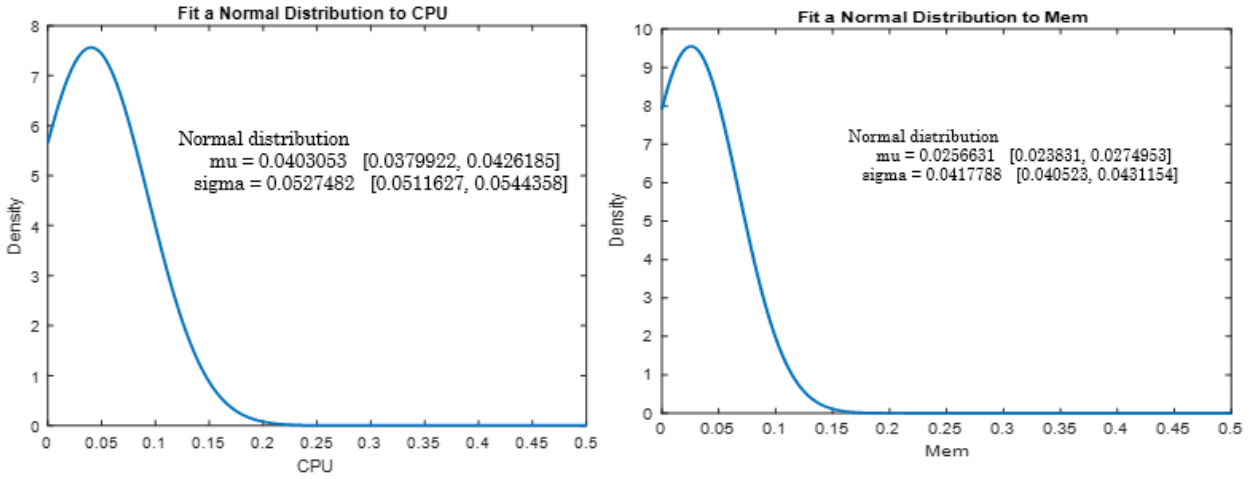

Figure 4.8: CPU and Memory probability distribution estimation

The last step entails finding out the similarity between this table and the predefined class table generated from an offline recursive algorithm for pre-classified VMs/tasks. Comparison and/or combine between several clustering methods will also facilitate the process of selecting the most appropriate combination. e.g. model-based clustering techniques using the SOM and GMM implemented on the same data. These methods provide stable results in identifying boundaries between clusters and subsequently use k-means or FCM to perform another clustering within these boundaries, as shown in Figure 4.9. The determination of boundaries can be implemented using a Support Vector Machine (SVM) algorithm [170].

The implementation of different combinations between clustering algorithms are represent the recommendation block, in Figure 4.1.

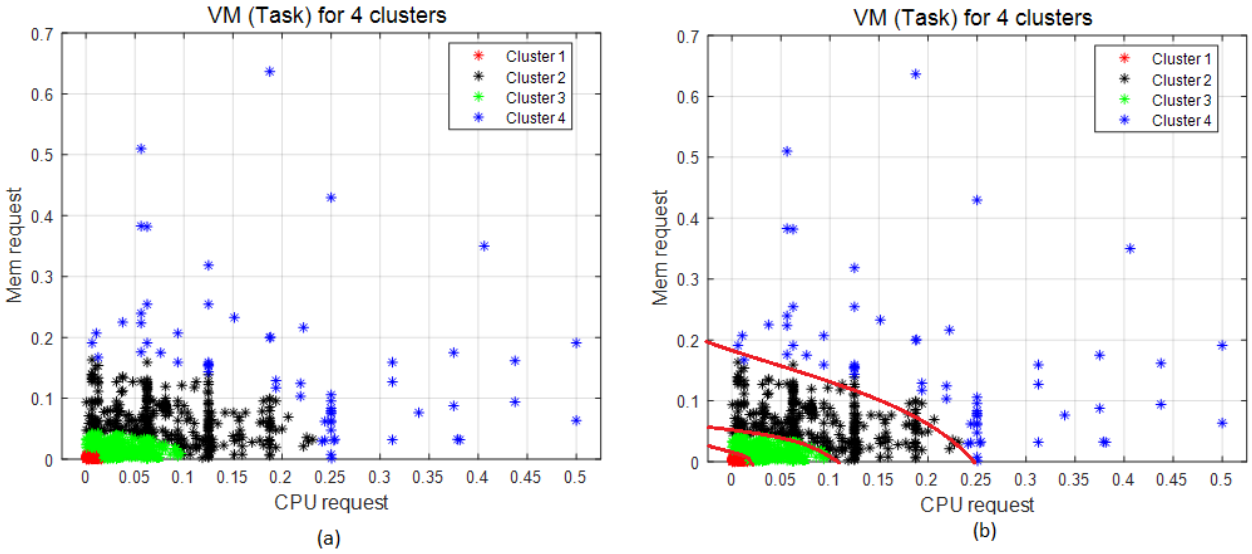

Figure 4.9: GMM clustering (a) Resulting 4 categories (b) Proposed boundaries

Appendix A presents a step-by-step simple implementation of the proposed systematic algorithm 2 for prediction purposes.

4.6 Chapter Summary

In this chapter, we discussed the clustering techniques that can be used for clustering big monitoring datasets, represented by VMs/tasks in large-scale data centres. More specifically, efficient pre-processing and systematic clustering were proposed to select the most appropriate clustering method based on clustering purpose, predefined validation indices, and clustering algorithms comparison. Additionally, we provided essential answers for the following questions: What is suitable VMs/tasks clustering technique for the large-scale data centre? What is the rationale behind this choice? In addition, we provide recommendations on clustering validation criteria.

Implementation of the proposed clustering subsystem on real big data shows a vast difference in accuracy, computation resources and execution time as compared to previous literature work, where the entire monitoring data is taken into consideration.

Chapter 5

User Behavior-Based Workload Prediction for Cloud-Data Centres

In this chapter, we propose an efficient workload prediction subsystem based on user behaviours. This prediction subsystem represents the key components of VM provisioning framework in real-time. The proposed prediction subsystem not only uses VM historical values, but also takes into consideration user behaviour and current states of the data centre. In this chapter, we also propose a technique for resolving the problem of predicting window sizes in order to optimize PM utilization.

5.1 Introduction

In our prediction subsystem, we use previous workload patterns to estimate future VM requests in data centres. Generally, the process of prediction comprises of two steps: VM clustering and VM prediction for each cluster. Prediction process is an important step for estimating the number of PMs required for the subsequent period.

Prediction of future resource required, e.g. VMs, is a crucial issue for efficient resource utilization in dynamic cloud-computing environments for the following reasons:

- Estimation of future performance or workload of each VM ensures service quality and minimizes costs [48].

- Prediction helps administrators take appropriate action to prevent the system from suffering traffic surges or the *Slashdot effect* attributed to high loads [68]. Prediction facilitates proactive job scheduling or host-load balancing decisions [80].

- Prediction is important for lowering the cost of the energy consumed by switching servers in order to reduce power states when they are not expected to be used [171].

- Workload estimation is not only used to decide whether to add or remove resources, but also to rearrange the order of query execution, and admit or reject new incoming resource requests [172].

- Accurate prediction host loads is also a significant factor in satisfying SLAs.

The forecasting load for a period of time (e.g. minutes) is necessary for real-time control, resource allocation, capacity planning and data centre energy saving in cloud computing. However, such accurate predictions are extremely challenging, owing to the possible instantaneous fluctuations in the load [80].

In this chapter, we propose a prediction subsystem that combines the systematic clustering presented in Chapter 4 and user behaviour in order to forecast future VM requests. The next section elucidates the components of the prediction subsystem. In Section 5.3, we describe user clustering components and propose the filtering process of VMs filtering. Section 5.4 and Section 5.5 discuss the observation and prediction windows sizes. Improved multivariate time series ELM prediction algorithm is given in Sections 5.6. Experimental implementation of the proposed subsystem will be discussed in Section 5.7. The summary of the chapter will be presented in Section 5.8.

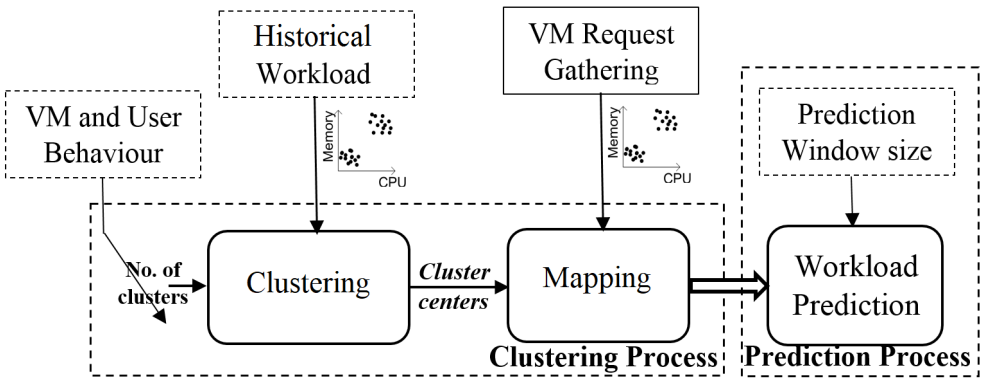

Figure 5.1: Proposed Prediction Subsystem

5.2 Prediction Subsystem

The rapid variations in VM requests and user behaviours make it difficult to use conventional machine learning algorithms with off-line learning. However, this work proposes a modified ELM algorithm to obtain an accurate prediction of VM requests. A multivariate time series ELM algorithm with the systematic clustering is presented in chapter 4, after which user behaviour will be combined in a prediction framework so as to forecast VM requests. A real-time VM workload prediction subsystem consists of the following main components, Figure 5.1:

Clustering component represents the clustering process, which is used to create a set of clusters for different types of VMs and users. The centres of VMs and users clusters are used for classifying collected data. The collected data denote the new request and/or consolidated VMs, during a specific time frame referred to as observation window.

Mapping component is used to map or distribute each request received during a given observation window into one cluster on the basis of calculated cluster centres.

User and VM behaviours component influences the overall cloud workload. Comprehensive workload models must consider both VMs and users, and even individual VM behaviour in order to reflect realistic conditions by excluding unwanted VMs or users from the workload estimation process.

Historical workload: the historical workload is updated periodically, and used to forecast

the next period VM request for each observation window.

VM Request gathering tool components represent the monitoring tools, Section 3.3. These monitoring tools are also used for detecting and tracking the variations or failure of resources as well as applications during an observation window.

Workload prediction component is used to estimate future VM requests. In this component, current and previous usage patterns are used to estimate future VM requests in a data centre.

Prediction window size is used to decide whether or not PMs need to be switched to sleep mode. Its value depends on the configuration of the data centre, especially the server hardware. The monitoring frame is determined based on the prediction window, the clustering, and prediction algorithm used.

The main characteristics of the proposed prediction subsystem are as follows:

- Proposed prediction subsystem integrates clustering not only on VM requests, but also on user requests. Consequently, a proper filtering process excludes unexpected VM requests caused by unpredictable users' actions.

- This subsystem overcomes the problem of time-varying VM requests. Time-varying means VM not only shifts the resource required in time, but also due to changes caused by the behaviour of other data centre resources. The proposed prediction subsystem depends on the actual service demand.

- This subsystem is an ELM-base online sequential framework which eliminates restrictions about observation window size and the number of VM clusters.

A time delay or time advance of input not only shifts the output signal in time, but also changes other parameters and behavior. It is notable that time variant systems respond differently to the same input at different times.

5.3 User Behavior-Based Filtering Process

The data centres workloads are driven not only by VMs' characteristics, but also by user behavioural patterns. User behaviours are significantly more diverse than VMs behaviours [51]. In the proposed prediction subsystem, user behaviour helps identify the dependency and/or the relationship between users and VMs/tasks so as to improve the accuracy of the prediction. This will be done through excluding VMs with unstable user behaviour from prediction process. Generally, users can be classified into three event types:

- Users with high VM number can be considered as a significant event, which implies that the user has a high number of useful resources on a frequent basis.
- Users with moderate VM number can be considered as medium events, which implies that the user is using the resources in a medium frequency
- Users with low VM numbers or no events can be considered as low events.

Users behaviour in a cloud data centre is difficult to predict. User behaviour modeling depends on the problem setting and its domain. Given that this research depends on monitoring tools wherein users' and their creating patterns that define each user VM requests are provided, it is important to clarify and highlight definitions of user activity, user event and user behaviour in the context of VMP. We have used the following definitions:

- **Activity** is the current action being carried out by the user.
- **Event** is grouping similar user activities that take place by a certain user at a certain time.
- **Behaviour** is a collection of performed events to be stored as history about a specific user.

In cloud data centres, user identity is undefined for security and privacy proposes. For example, user names are hashed and provided as opaque base64-encoded strings in data described in Section 3.3. Our proposed algorithm will: a) Identify users for current monitoring period; b)

Define a specific numeric code for each user; c) Cluster users on the basis of their behaviour (see Algorithm 3).

Algorithm 3 Clusters users based on their behaviour

Inputs: Names of all users and all VMs/tasks
Outputs: Names of users without repetition and number of VMs/tasks for each user

1: Extract all names in a specific period
2: Delete repeated names by matching strings
3: Coding user to numbers
4: Use a suitable user clustering technique

In this chapter, the clustering algorithm will be used as a means of grouping users based on their behavior. Usually, users' activities are represented using vectors of their behaviour and time. The clustering algorithm clusters them in different forms based on event types.

In large scale data centres, the number of users is usually large and is continuously changing. Therefore users should be clustered based on used characteristics. Our proposed clustering will be based on the number of submitted VMs for each monitoring sample. This means that we totally depend on the user's instant profiles and the number of VM requests during a single monitoring period. In this process, it is important to leverage users' behaviour in filtering VMs clusters for prediction algorithm.

With the user clustering process, a mapping process is used to distribute VMs received during an observation to a specific user cluster. The output of this stage will be in a 2D matrix form combined VMs and users clusters represented by a matrix, in which each row represents a time period, as shown below:

$$
\begin{array}{cccccccccc}
V_1U_1 & V_1U_2 & \cdots & V_1U_u & V_2U_1 & \cdots & V_2U_u & \cdots & V_mU_u & \text{Time(min)} \\
\begin{bmatrix}
V_1^0U_1^0 & V_1^0U_2^0 & \cdots & V_1^0U_u^0 & V_2^0U_1^0 & \cdots & V_2^0U_u^0 & \cdots & V_m^0U_u^0 \\
V_1^1U_1^1 & V_1^1U_2^1 & \cdots & V_1^1U_u^1 & V_2^1U_1^1 & \cdots & V_2^1U_u^1 & \cdots & V_m^1U_u^1 \\
\vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\
V_1^iU_1^i & V_1^iU_2^i & \cdots & V_1^iU_u^i & V_2^iU_1^i & \cdots & V_2^iU_u^i & \cdots & V_m^iU_u^i
\end{bmatrix} &
\begin{matrix} t_0 \\ t_1 \\ \vdots \\ t_i \end{matrix}
\end{array}
\tag{5.1}
$$

where V_i denotes VMs in cluster i; U_i represents users in cluster j; m is number of the VMs' clusters; and u signifies the number of users' clusters. e.g. $V_i^kU_j^k$ represents the number of VMs in cluster i and user in cluster j at time period k.

The filtering process will be used to remove users with a low number of VMs. This will increase the accuracy of the prediction algorithm by removing VMs with the lowest probability. Algorithm 4 summarizes the steps where the output of this process is filtered VMs clusters to be used in the next prediction stage.

Algorithm 4 Filtering process based user behaviour clustering

Inputs: VMs clusters and users monitoring data

Outputs: Filtered VMs clusters each cluster with CPU and memory size only

1: Calculate the number of users from historical data, Algorithm 3
2: Using historical data, long-period users clustering based on the number of VMs/tasks
3: Find the number of VMs for each user during the sampling period
4: Calculate the number of VMs for each user cluster
5: Find a unique user/VM clustering matrix
6: Filter user/VM matrix by deleting users with a small and very-low number of VMs

In this algorithm, historical data are used to cluster users based on the number of VMs for each user during a specific period that represents the observation windows to be used for forecasting the next period of time. A unique user/VM clusters matrix will be filtered by deleting the column with the lowest expectation of VMs, i.e. unexpected user clusters

with a specific type of VM. As an implication, the filtering process eliminates VMs with the lowest probability from the prediction process. The filtering process can be done by using the following:

- ***Predefined threshold***: This filter is used to delete the column with the lowest expectation of VMs, i.e. unexpected user clusters with a specific type of VMs. Put differently, the filtering process eliminates VMs with the lowest probability from the prediction process.

- ***Feature selection***: This filter is used to exclude a specific or predefined types of user or unwanted users under a process that is suitable for data with predefined classes. However, this filter does not match with the proposed framework where there is no information about the data centre users.

- ***Dimensionality reduction***: This filter is based on table minimization technique. The dimensions of the input space are used to facilitate the exploration of structures in high-dimensional data. Dimensions reduction techniques are not recommended in our proposed prediction subsystem due to the nature of the problem in mixing between two independent variables users and VMs behaviour.

5.4 Observation Window Size

The determination of observation windows size is a process of observing and monitoring workload variations during past time periods. Observation window will be used to specify the time frame for gathered data to feed the prediction process. A long time frame will increase the calculation, while a short time frame will reduce the prediction's accuracy.

For example, in Google trace data described in Section 3.3, Di *et al* [80] estimated the

observation window to half of the prediction window length. Dabbagh *et al* [41, 171] used experiments to estimate the duration of the observation window in each cluster for the same data. i.e. different observation window sizes for each cluster. They increased the size of the observation window gradually until a point is reached beyond which the prediction error can no longer be reduced and even the window size increases.

In the proposed subsystem, we estimate the size of the observation window after classifying VMs into clusters. A unique observation window size for all clusters will be determined. This is estimated based on clusters behaviour during previous time periods. Clusters' behaviour will be represented by the maximum degree of the equation that can fit into cluster centres. This degree is the number of previous periods required to feed the predictor.

5.5 Prediction Window Size

The prediction window is the time period for the workload that is making the forecast. The prediction window size represents the algorithms and optimization techniques used to determine the minimum number of times that the prediction calculations must be performed. For example, Prevost *et al* [7], presented a dynamic prediction quantization method for determining the number of prediction calculation intervals to be performed within the required future load SLAs. They didn't take into consideration the PMs status, which is necessary for power consumption framework.

The prediction window size completely depends on the configuration of the data centre, especially the server hardware. We adopted Dabbagh *et al* [41, 171] proposal in estimating prediction window size. It is based on the difference between the energy cost for keeping the PM idle or PM Off/On power cost, as mentioned in the following equations:

$$E_{sleep} = E_0 + P_{sleep} \cdot (T_p - T_0) \tag{5.2}$$

where T_p denotes the length of the prediction window, P_{sleep} is the consumed power when in the sleep mode, E_0 signifies the energy needed to switch the PM to the sleep mode as well as the energy needed to wake up it later, and T_0 represents the transitional switching time. The estimated time required to keep the PM ON and idle (T_b) consumes an amount of energy that is equal to the energy consumed due to mode transition plus that which is consumed while the PM is in the sleep mode during that same period:

$$P_{idle} \cdot T_b = E_0 + P_{sleep} \cdot (T_b - T_0) \tag{5.3}$$

where T_b denotes the beak-event time. This means energy can be saved by switching PM to sleep mode if and only if the PM stays idle for a time period longer than T_b. That is, $T_p \geq T_b$ must hold in order for power switching decisions to be energy efficient.

According to the above, and for heterogeneous data centre environments, T_p value can be easily estimated based on individual PM profiles provided by a monitoring tool. In our implementation, we are going to use the PM energy measurement study of PMs conducted in [173] in order to estimate the break-even time, T_b.

5.6 Improved ELM Predictor

Generally, machine learning, and especially neural networks, are powerful tools for solving problems associated with complex systems. For this reason, machine learning is the best approach for modeling systems with a high complexity, where there is good information or where the information is almost correct owing to available monitoring tools. The main

objective of this prediction approach is to find a suitable model so as to ensure the best fit between the predicted and actual results request.

In this next section, we propose an improved ELM-based predictor and combine it with the proposed clustering technique in Chapter 4 to forecast the number and type of VM in a data centre. The main features of the proposed predictor are compared with related work as discussed in Subsection 2.2.3 are:

- ELM is single-layer neural network, with one iteration. The ELM is perfect for real-time application
- Overcome conventional gradient-based learning methods requirements, such as avoiding stopping criteria, size of learning epochs, and local minimums.
- An online sequential process, which is able to eliminate the restrictions on observation window size and the number of clusters.
- Allows each cluster to have its own predicting network, so as to reduce errors.
- Overcomes the problem of time-varying VM requests, by depending on the actual service demand.

The ELM is a single hidden layer feed-forward neural network, with one input layer, one hidden layer, and one output layer. The predictor is used to estimate the number of VM requests for each VM cluster during the observation window. This will be done by using the previous states of each cluster $C(k) = [c_1(k)\ c_2(k)\ \ldots\ c_K(k)]'$ to forecast the number of VM for next period for each cluster $C(k+1) = [c_1(k+1)\ c_2(k+1)\ \ldots c_K(k+1)]'$. Where $c_i(k)$ is current number VMs in cluster i and $c_i(k+1)$ is the predicted number VMs in cluster i.

The number of clusters (input/output) of predictor that may change in each monitoring period depends on the recommended number of cluster from the clustering subsystem. Put

differently, the ELM predictor has a variable structure by changing number of input/output nodes. The number of hidden nodes is given by $\boldsymbol{l}$ in Figure 5.2. w_{ji} denotes the weight between the i^{th} neuron in the input layer and the j^{th} neuron in the hidden layer. β_{jr} represents the weight between the j^{th} neuron in the hidden layer and the r^{th} neuron in the output layer. Finally, b_j signifies the threshold in the j^{th} hidden layer.

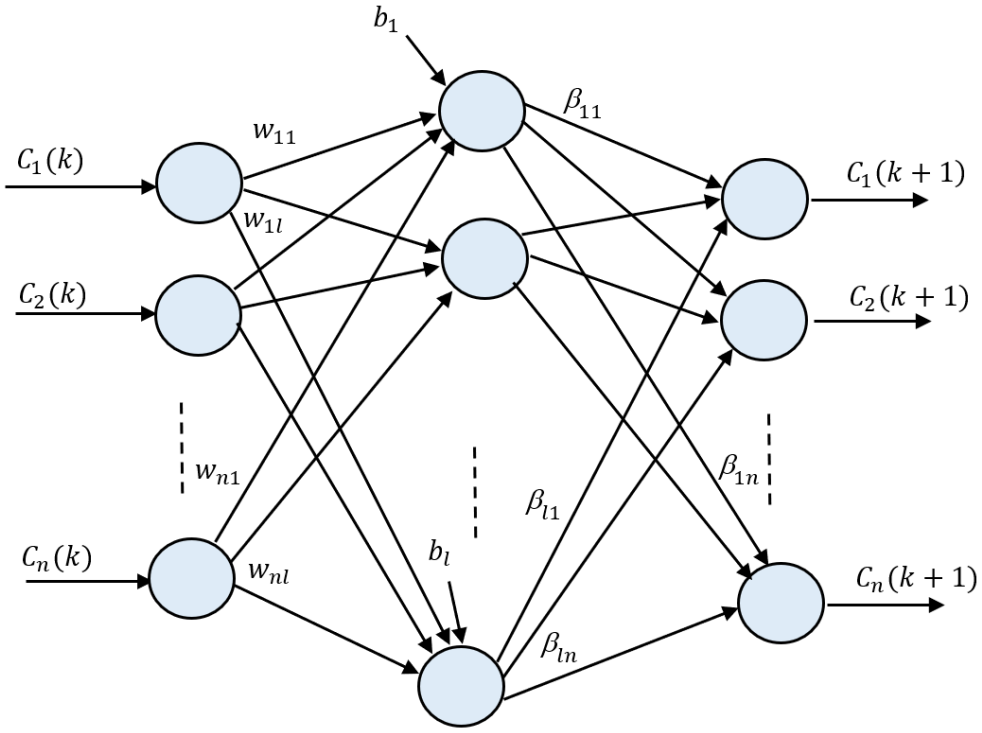

Figure 5.2: ELM predictor

For a general setting, let $X(k)$ and $Y(k)$ be the input and the output vectors at sample k, respectively. Let $\mathbf{W}$ be the input weight matrix ($\boldsymbol{l} \times n$). Let $\mathbf{B}$ be the output weight matrix ($\boldsymbol{l} \times n$), and $\mathbf{b}$ be the bias vector. Then, we get:

$$X(k) = [C_1(k)\ C_2(k)\ \ldots\ C_n(k)]^T \tag{5.4}$$

$$Y(k) = [C_1(k+1)\ C_2(k+1)\ C_n(k+1)]^T \tag{5.5}$$

$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & & \vdots \\ w_{l1} & w_{l2} & \cdots & w_{ln} \end{bmatrix}_{l \times n} \tag{5.6}$$

$$\mathbf{B} = \begin{bmatrix} \beta_{11} & \beta_{12} & \cdots & \beta_{1n} \\ \beta_{21} & \beta_{22} & \cdots & \beta_{2n} \\ \vdots & \vdots & & \vdots \\ \beta_{l1} & \beta_{l2} & \cdots & \beta_{ln} \end{bmatrix}_{l \times n} \tag{5.7}$$

$$\mathbf{b} = \begin{bmatrix} b_1 & b_2 & \dots & b_l \end{bmatrix}^T_{l \times 1} \tag{5.8}$$

If there are Q samples of data, the input matrix $\mathbf{X}$ and the output matrix $\mathbf{Y}$ are given by:

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1Q} \\ x_{21} & x_{22} & \cdots & x_{2Q} \\ \vdots & \vdots & & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nQ} \end{bmatrix}_{n \times Q} \tag{5.9}$$

$$\mathbf{Y} = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1Q} \\ y_{21} & y_{22} & \cdots & y_{2Q} \\ \vdots & \vdots & & \vdots \\ y_{n1} & y_{n2} & \cdots & y_{nQ} \end{bmatrix}_{n \times Q} \tag{5.10}$$

The ELM with l hidden nodes with an activation function $g(x)$ and the output $Y(t)$ is given by:

$$Y(t) = \sum_{i=1}^{l} \beta_i g_i(x_j) = \sum_{i=1}^{l} \beta_i g_i(w_i.x_j + b_i), \quad j = 1, \ldots, Q \tag{5.11}$$

If the target vector $\mathbf{T}$ is $[t_1, t_2, \ldots, t_Q]^T_{n \times Q}$, and $t_j = [t_{1j}, ..., t_{nj}]_{n \times 1}$, using Equation 5.11, the target matrix $\mathbf{T}$ can be written as follows:

$$\mathbf{T} = \begin{bmatrix} \sum_{i=1}^{l} \beta_{i1} g_i(\mathbf{w}_i \mathbf{x}_j + bi) \\ \sum_{i=1}^{l} \beta_{i2} g_i(\mathbf{w}_i \mathbf{x}_j + bi) \\ \vdots \\ \sum_{i=1}^{l} \beta_{in} g_i(\mathbf{w}_i \mathbf{x}_j + bi) \end{bmatrix} \tag{5.12}$$

By separating β and therefore $\boldsymbol{g}$ in Equation 5.12, we get:

$$\mathbf{H}\beta = \mathbf{T} \tag{5.13}$$

where $\beta = [\beta_1, \beta_2, ..., \beta_l]^T$ and $\mathbf{H}$ is the activation matrix provided by the following substitutions:

$\mathbf{H}(\mathbf{w}_1, ..., \mathbf{w}_l, b_1, \ldots, b_l, \mathbf{x}_1, \ldots, \mathbf{x}_Q) =$

$$\begin{bmatrix} g(\mathbf{w}_1 \mathbf{x}_1 + b_1) & g(\mathbf{w}_2 \mathbf{x}_1 + b_2) & \cdots & g(\mathbf{w}_l \mathbf{x}_1 + b_l) \\ g(\mathbf{w}_1 \mathbf{x}_2 + b_1) & g(\mathbf{w}_2 \mathbf{x}_2 + b_2) & \cdots & g(\mathbf{w}_l \mathbf{x}_2 + b_l) \\ \vdots & \vdots & \vdots & \vdots \\ g(\mathbf{w}_1 \mathbf{x}_Q + b_1) & g(\mathbf{w}_2 \mathbf{x}_Q + b_2) & \cdots & g(\mathbf{w}_l \mathbf{x}_Q + b_l) \end{bmatrix} \tag{5.14}$$

$\mathbf{H}$ is called the hidden-layer output matrix of the network. The i^{th} column of $\mathbf{H}$ is the i^{th} hidden node's output vector with respect to the inputs $x_1, x_2, \cdots, x_n$ whereas the j^{th} row of $\mathbf{H}$ is the output vector of the hidden layer concerning the input x_j [174].

Huang *et al* [175] demonstrated that the input weights w_j and biases b_j for $j = 1, 2, \cdots, \boldsymbol{l}$ associated with the hidden layer can be assigned random values, and hence do not require any training. The output weights β_i, $i = 1, 2, \cdots, m$ are determined through learning from the training instances by solving the following objective function:

$$\underbrace{min}_{\beta} \|\mathbf{H}\beta - \mathbf{T}\| \tag{5.15}$$

and its solution is:

$$\hat{\beta} = \left(\mu I + H^T H\right)^{-1} H^T \mathbf{T} \tag{5.16}$$

for $\boldsymbol{l} < N$, where I is the identity matrix; μ is a regularization parameter.

Equation 5.16 represents the optimal weight current sample, which can be updated for each previous time series samples to forecast the one step ahead sample according to:

$$\hat{\beta} = \hat{\beta} + \lambda \Delta \hat{\beta} \tag{5.17}$$

As is the case with any delta function, this proposed improvement is added to the updated weight in regular feed-forward neural network with back propagation training algorithm where the new weight depends on previous weight with delta error between two previous estimated weights.

$$\Delta \hat{\beta} = \left(\mu I + H^T H\right)^{-1} H^T \mathbf{e} \tag{5.18}$$

where λ is a control parameter $(0, 1)$ and $\mathbf{e} = [e_1, e_2, \ldots, e_m]^T$ is the error vector.

The predicted output $\hat{Y}(k+1)$ will be calculated for the last estimated $\hat{\beta}$ for a new data set through:

$$\hat{Y}(k+1) = \mathbf{h}\hat{\beta} \tag{5.19}$$

where **h** denotes the **H** matrix for the new input.

Algorithm 5 describes the steps of the Multivariate Time Series ELM predictor.

Algorithm 5 Multivariate time series ELM predictor

Initialization: Choose μ, l; randomly generate w_{ij} and b_i; initial error E_{old}
Inputs: Normalize *X(k)* and *T(k)* to scales [0.1- 0.9]
Output: Optimal$\hat{Y}(k+1)$

while $E_{new} < E_{old}$ **do** ▷ repeat for each estimation
  for $k = 1 : P$ **do** ▷ P is the number of observed samples
    Calculate $H(X(k), w, \beta)$ using Equation 5.14
    Calculate $\mathbf{h}(z, w, \beta)$ where $z = X(k+1)$
    Calculate $\hat{Y}(k+1)$ using Equation 5.19
    Calculate new error $e(k) = T(k) - Y(k)$
    $E_{new} = E_{old} + \frac{1}{2}\sum e^2(k)$
  end for
  Calculate $\Delta\hat{\beta}$ using Equation 5.18
  Update weights using Equation 5.17
  if $E_{new} >= E_{old}$ **then**
    $\mu = \rho\mu$
  else
    $\mu = \rho/\mu$;
  end if
end while

In this algorithm, ρ is an update parameter and P denotes the chunk data number. When $P > 1$, the improved ELM algorithm can update the output weights when every P sample is observed.

5.7 Experimental Results

In this section, the efficiency of the proposed prediction subsystem will be evaluated using the Google trace data that was explained in Section 3.3. Figure 5.3 shows k-means and FCM

clustering for different numbers of user clusters for the same period of Google data. The number of users over 10 hours was 299. FCM achieved better results for a small number of clustering, which will be very useful in reducing the number of inputs in a prediction system. In turn, this will improve the overall proposed framework energy consumption optimization. If we choose six classes of users, the number of tasks given to each class is illustrated in Figure 5.4.

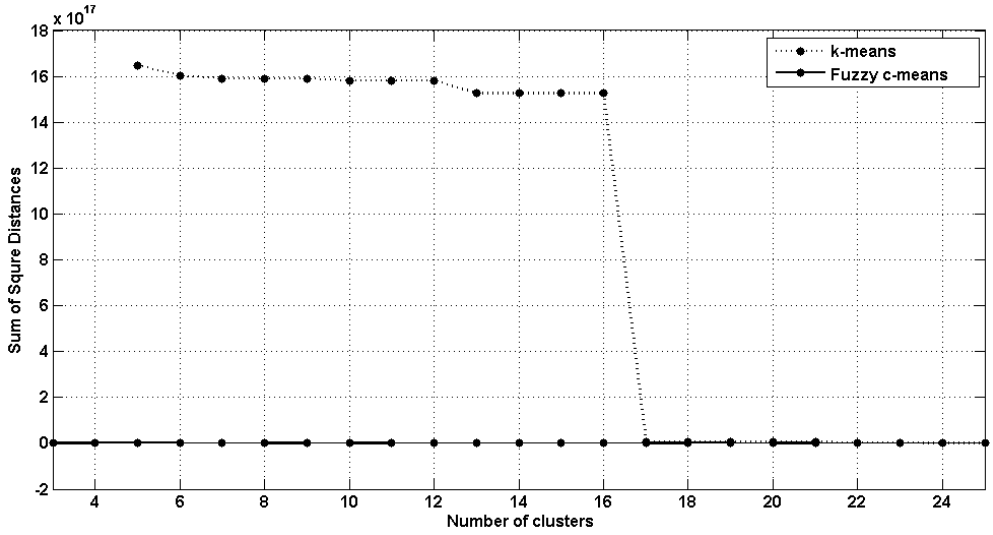

Figure 5.3: Number of user clusters vs sum of square error

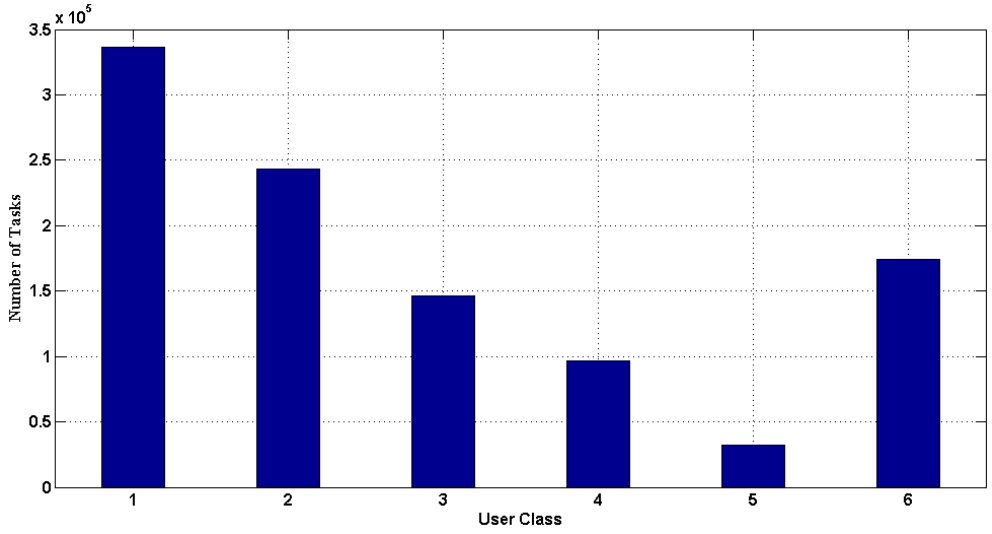

Figure 5.4: Number of tasks for each user's cluster

Table 5.1 depicts a sample of the VM/user mapping matrix described in Algorithm 3, where the first column represents the time (in minutes). It commences from minute 60 because we use the first day of Google Trace data, where the monitoring starts after the first hour [46]. This means that the mapping matrix in this example consists of 1440 rows (representing the number of minutes in 24 hours) and 28 columns (denoting 4 VM times 7 user clusters).

Table 5.1: Sample of VM/user clustering matrix for 4/7 VM/user clusters

Time(Min)	**VM1U1**	**VM1U2**	**VM1U3**	**VM1U4**	**VM1U5**	**VM1U6**	**VM1U7**	**VM2U1**	$\cdots$	**VM4U7**
60	1	0	0	0	0	0	35	3	$\cdots$	15
61	0	0	0	0	0	0	731	9	$\cdots$	6
62	0	0	0	0	1	0	260	6	$\cdots$	3
63	0	0	0	0	0	0	68	6	$\cdots$	6
64	0	0	0	0	3	0	2181	9	$\cdots$	11
65	34	0	0	0	2	0	2109	0	$\cdots$	1
66	57	0	0	104	0	0	774	15	$\cdots$	16
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
1500	74	0	0	1	2	0	632	1176	$\cdots$	111

The filtering process is used to delete the column with the lowest expectation of VM, i.e. unexpected user cluster with a specific type of VM. This implies that the filtering process eliminates VMs with the lowest probability from the prediction process.

Experiments show a greater variance of user cluster behaviours SSD error, Equation 4.15, when compared to its VM cluster behaviours. In addition, user cluster results reveal that in some cases, users have a very small number of specific VM clusters. Hence, long term data during the off-line clustering can be used to identify and remove these types of user clusters from the prediction computation. Such filtering will then improve the accuracy of the prediction by removing the outliers from the dataset used.

In order to evaluate the proposed ELM predictor, we compare it with a primarily ELM predictor with 2 state feedback [39]. In other word, Equation 5.4 and Equation 5.5 written as:

$$X(k) = [C_1(k)\ C_1(k-1)\ C_1(k-2) \dots\ C_3(k-1)\ C_3(k-2)]^T \tag{5.20}$$

$$Y(k) = [C_1(k+1)\ C_2(k+1)\ C_3(k+1)]^T \tag{5.21}$$

Primarily predictors like the improved ELM predictor is used to estimate the number of VM requests for each VM cluster during the observation window. ELM mainly uses the three previous states of each cluster, based on the clustering subsystem where each cluster is represented by a second-order equation during the sampling period (monitoring window). This means that if we have three types of VM clusters, primarily predictor will use 9 inputs and 3 output. In other words, the resulting primarily ELM predictor has a structure of 9-input nodes, $\boldsymbol{l}$ hidden nodes, and 3-output nodes [39].

Both predictors will be evaluated by calculating and comparing the Root Mean Square Error (RMSE) defined as follows:

$$RMSE = \sum_{i=1}^{3} \sqrt{\frac{1}{N} \sum_{k=1}^{N} \left(\hat{Y}_i(k) - Y_i(k)\right)^2} \tag{5.22}$$

where i denotes one of the 3 categories used, and N represents the total number of samples.

For the primarily ELM predictor, Algorithm 5 changed to Algorithm 6:

Algorithm 6 Preliminary ELM predictor

Input : Regular parameter R instead of μ; number of hidden neurons l; Random $\mathbf{W}$ and $\mathbf{b}$
Input : Normalized $[X(k),\ Y(k),\ Y(k-1),\ Y(k-2)]$
Output : Normalized $Y(k+1)$

1: Calculate the hidden layer output matrix $H(X(k), w, \beta)$ using Equation 5.14.
2: Calculate the inverse matrix term $G = (\frac{I}{R} + H'H)^{-1}$
3: Calculate the output of the hidden layer using the new unknown input $\mathbf{h}(X(k+1), w, \beta)$
4: Calculate the predicted output, of the ELM $\hat{Y}(k+1)$ using $\hat{Y}(k+1) = [C_1(k+1)\ C_2(k+1)\ C_3(k+1)]^T = \mathbf{h}\hat{\beta}$

In both predictors, we first select the number of hidden nodes $\boldsymbol{l}$ and the regulation

parameter R or μ. Figure 5.5 and Figure 5.6 illustrate the relationship between the number of hidden neurons $\boldsymbol{l}$ and the RMSE for the preliminary and multivariate predictors, respectively. Evidently, RMSE is reduced by increasing the number of hidden neurons $\boldsymbol{l}$, like any other conventional gradient descent algorithm. The main difference is that the error is less in the multivariate algorithm for a law number of the hidden neuron. Importantly, number of hidden neurons in the second predictor is about 1/3 of the preliminary predictor.

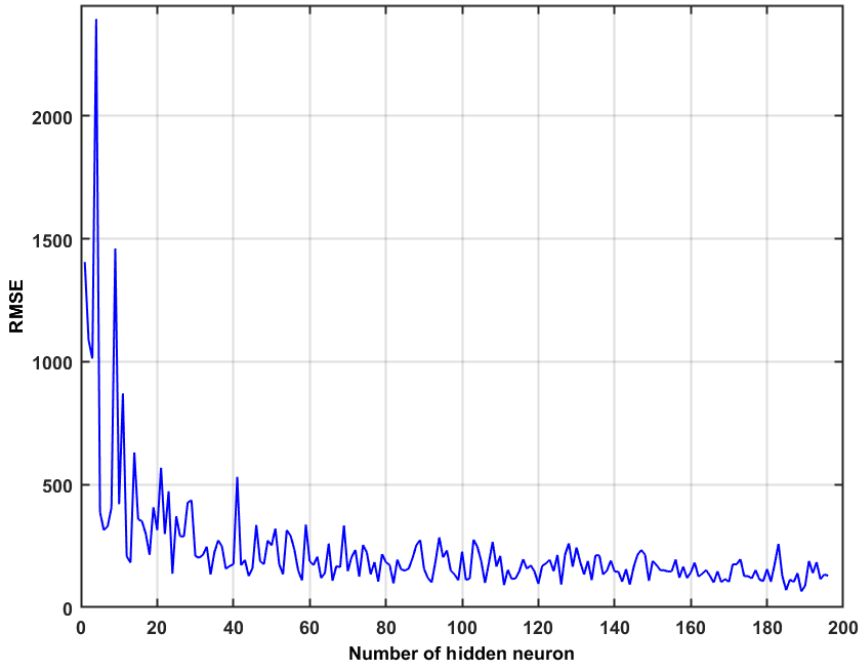

Figure 5.5: RMSE vs number of hidden neuron $\boldsymbol{l}$ in preliminary predictor

The value of the regulation parameters R and μ does not have a major effect on the value of RMSE in both predictors, as illustrated in Figure 5.7 and Figure 5.8 which reveal the relationship between the regulation parameters and the RMSE. In subsequent experiments, we used $\boldsymbol{l} = 100$ and $\mu = 1/5000$ for the proposed forecaster.

Figure 5.9 shows an arbitrary 10-hour data with 5 min sampling rate of the actual number of requests received as compared to both preliminary and multivariate predictors outputs. The two predictors almost gave the same behaviour as the actual number of VM requests.

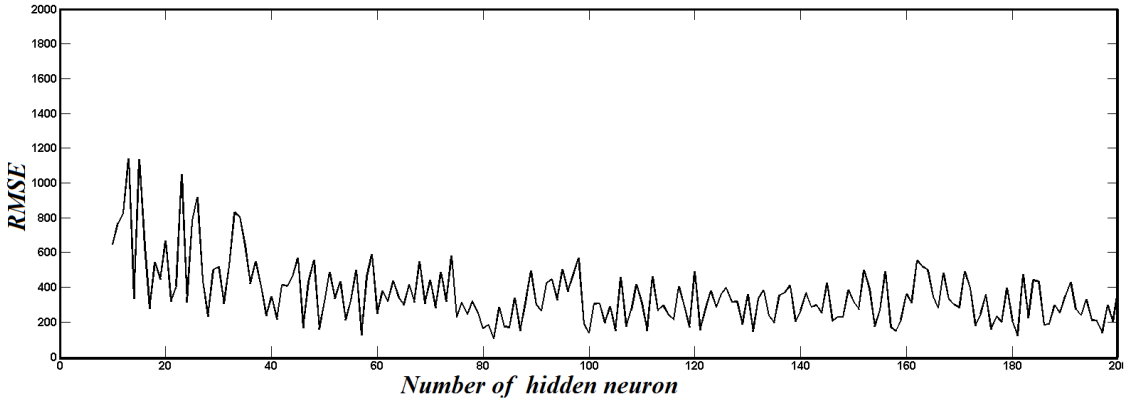

Figure 5.6: RMSE vs number of hidden neuron $\boldsymbol{l}$ in the multivariate predictor

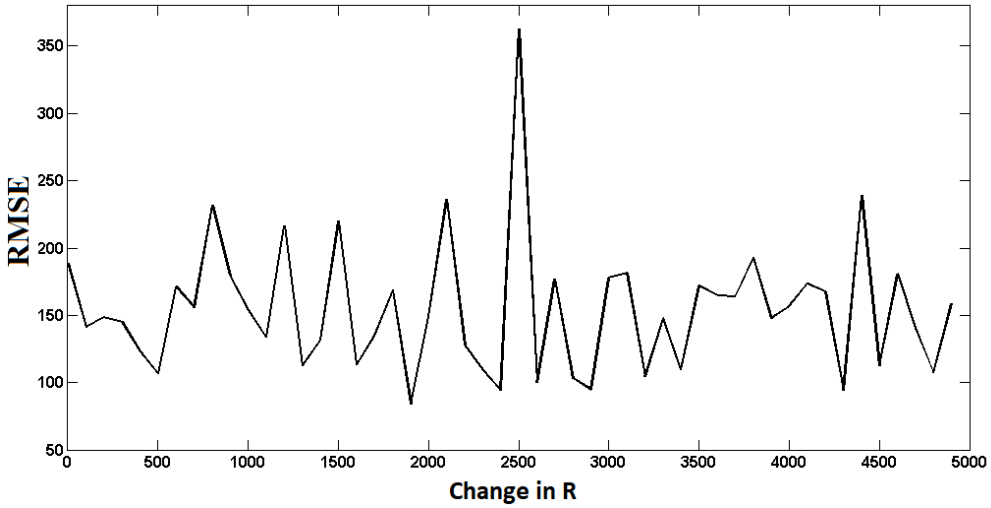

Figure 5.7: RMSE vs regulation parameter R for the preliminary predictor

Execution time (in seconds) for both preliminary and multivariate predictors were 1.9661 sec. and 0.7106 sec., respectively. The proposed multivariate forecaster is faster than the preliminary counterpart by almost 60-70%.

In order to illustrate the efficacy of our approach, we have compared the proposed predictor accuracy with the followings: Preliminary (Simple) ELM, Last minute predictors, Min predictors, Max predictors, Average predictor, Exponential Weighted Moving Average (EWMA) predictors, Linear Regression (LR) predictors, and Wiener filters [41, 65, 176]. These algorithms are used for estimating the number of request in each of the four clusters

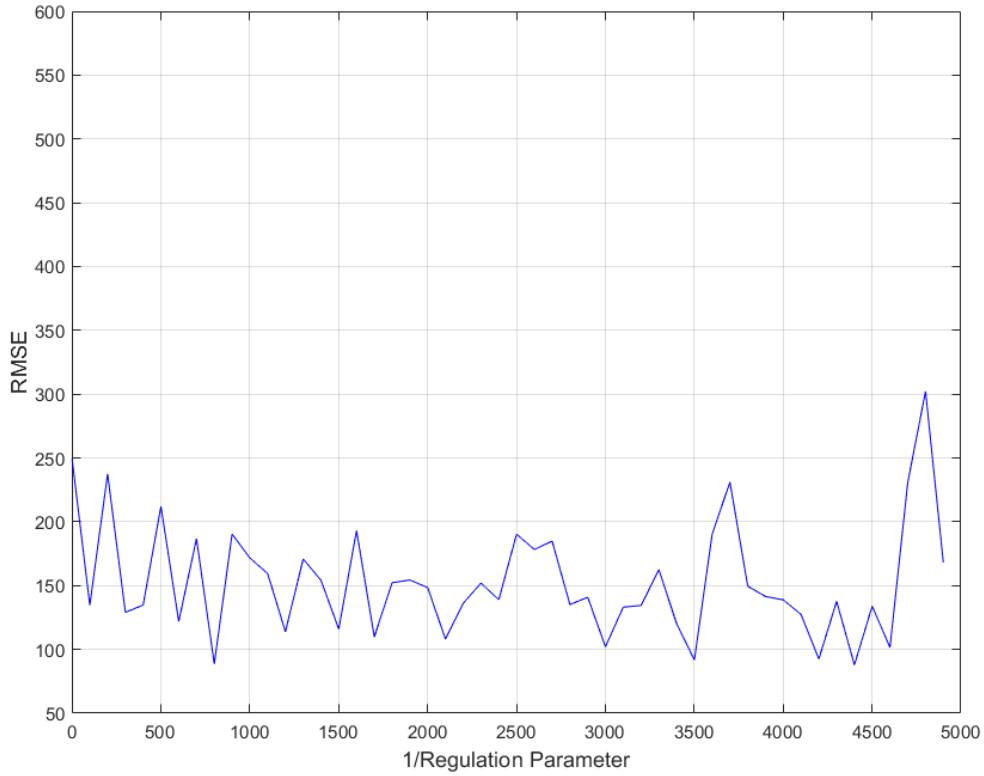

Figure 5.8: RMSE vs regulation parameter μ for the proposed predictor

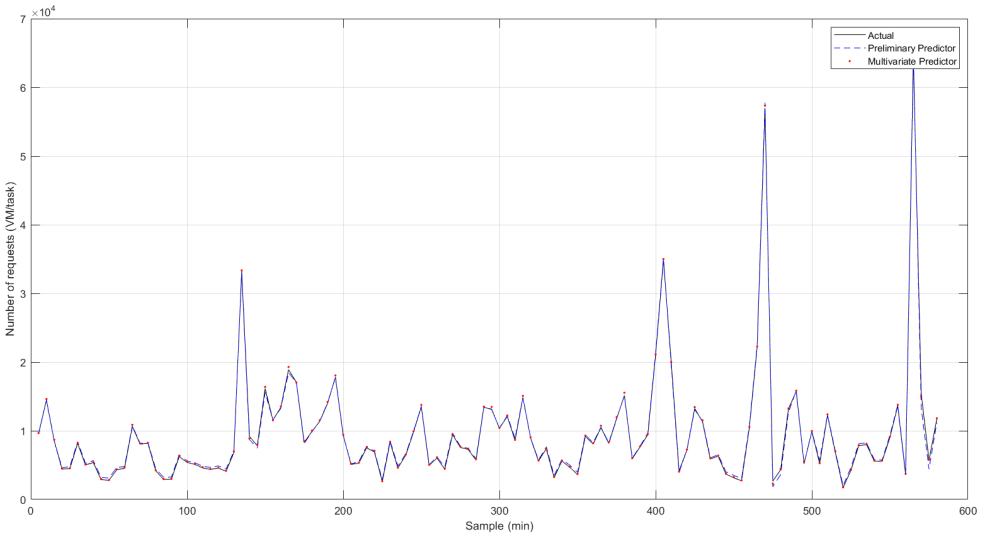

Figure 5.9: Sample of actual vs predicted number of requests

of VMs.

Figure 5.10 shows the RMSE of each of these prediction approaches, for 5 hours test

sample with 5 minutes prediction window. Our proposed predictor is found to yield the lowest RMSE.

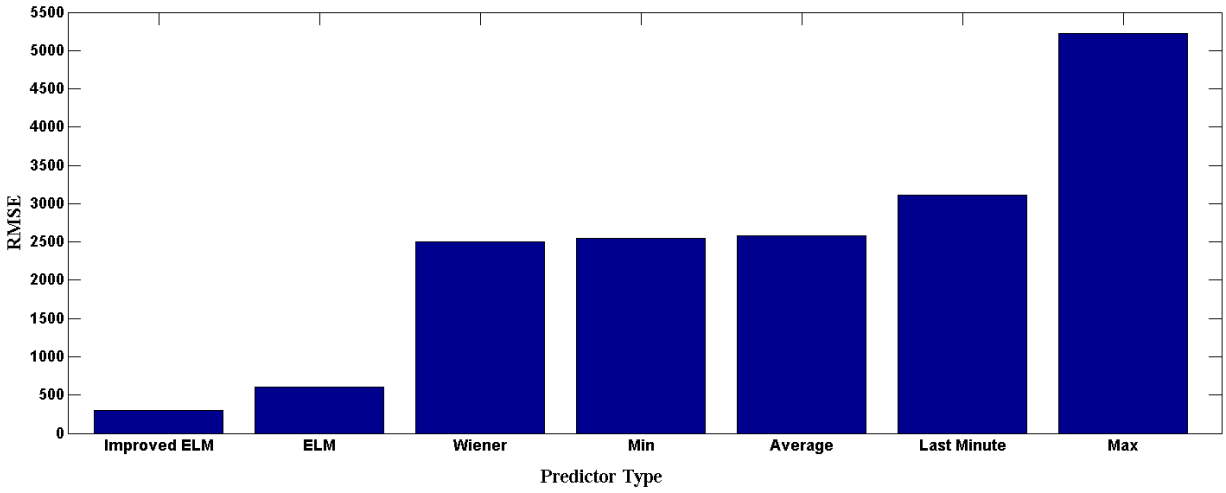

Figure 5.10: RMSE comparisons of different predictive approaches

Several scenarios in the types and sizes of predictor inputs are used to investigate the effect of inserting user behaviour (user clusters) on VM request forecasts. All scenarios are implemented for six user clusters and four VM clusters. These scenarios are (1) ELM with three states of VM clusters; (2) ELM with three states of VM and one state of user clusters; (3) ELM with three states of VM and user clusters; (4) ELM with one state of VM and user clusters; and (5) improved ELM with one state of VM clusters.

Figure 5.11 and Figure 5.12 shows a comparison of several scenarios in the types and sizes of predictor inputs, while Figure 5.11 depicts these comparisons for each VM cluster. It evident that the proposed multivariate ELM gives a faster and minimum error for both in totality and for individual clusters.

Experiments confirm that the proposed filtering process can be used to remove VM requests that have less probability, on the basis of historical data. We also noticed that adding only user clusters as the input to the ELM prediction algorithm will not be efficacious. Such addition will increase the rate of error, because the principle of ELM is to estimate the

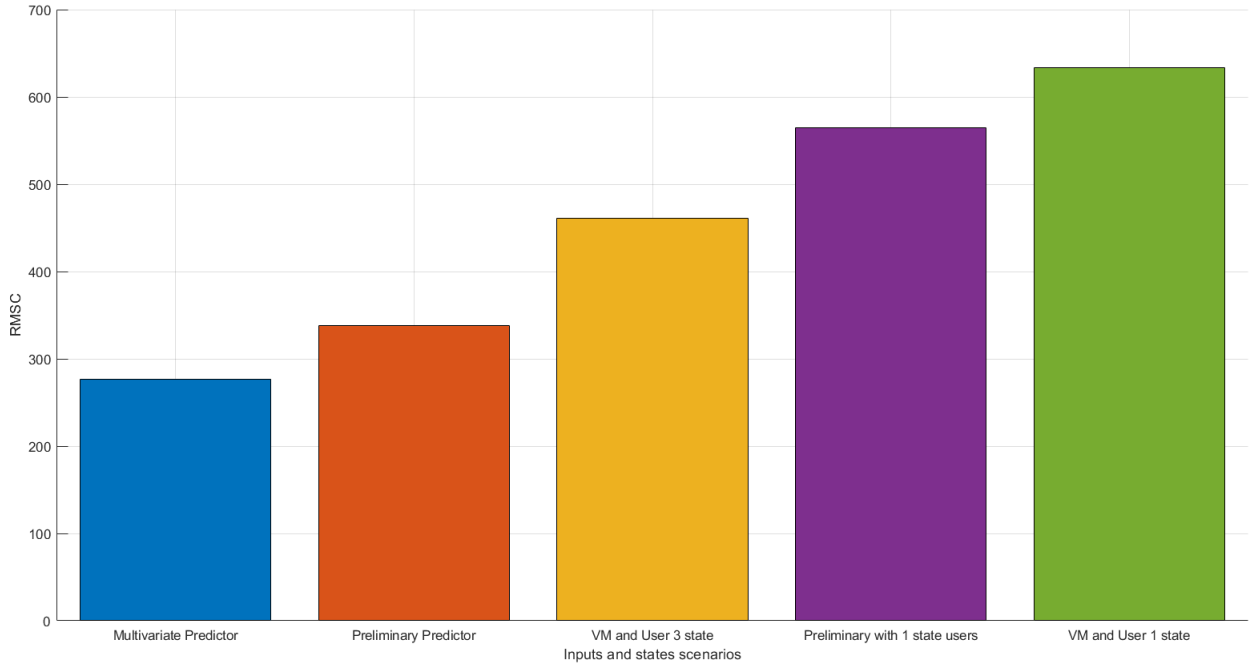

Figure 5.11: RMSE comparisons of different ELM inputs and states scenarios

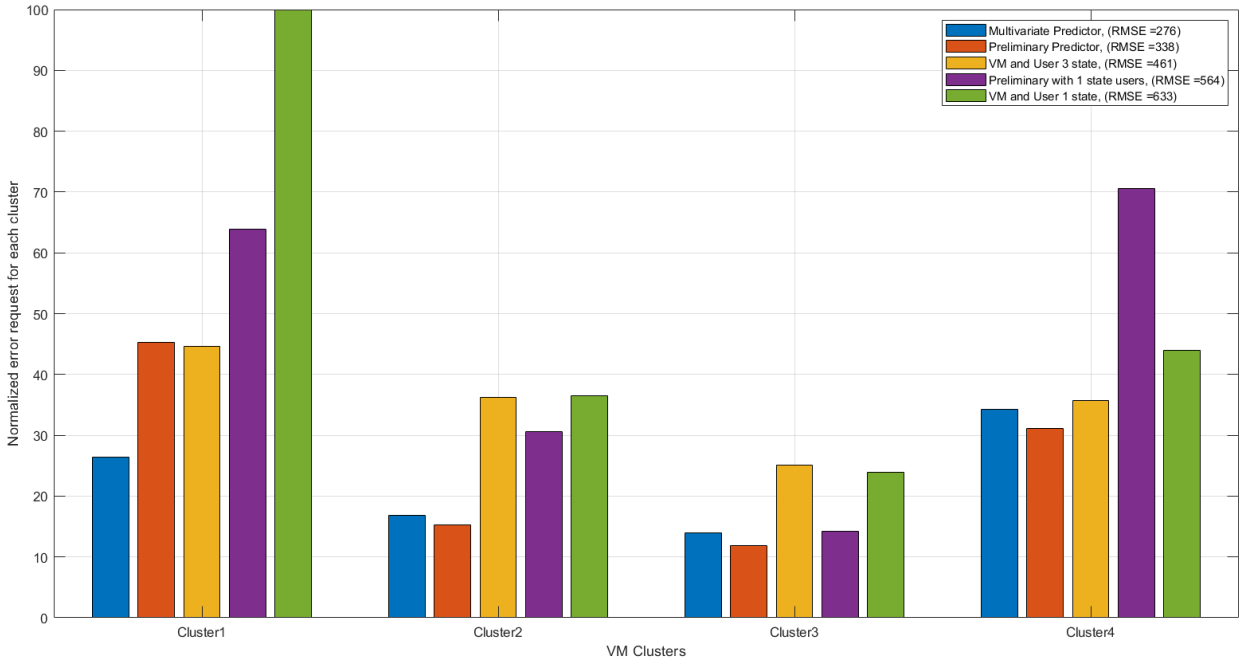

Figure 5.12: ELM request error for each cluster without user behaviour for 5 hours data

nonlinear function, which will have more errors if we add more variables. On the other hand, our approach of using both user clusters and VM request together with the proposed ELM can provide as much as 20% improvement in accuracy. Further improvement in accuracy may be derived on the basis of using a long term offline filter, given a set of data. Value selection for update parameter ρ and control parameter λ can also impact the accuracy of results. In our experimentation, optimal values were found using a trial-and-error approach. Part of future work can focus on finding a systematic way of finding off-line optimal values of ρ, λ and even the type and number of hidden neuron l.

The on-line multivariate time series ELM eliminates any restrictions on observation window size and the number of VM clusters (inputs) for the predictor. In our online computations, we simply input the current state and used an iteration of previous states to cover all possible observation window states, instead of using all the previous state for observation windows as inputs. This approach also reduces the computation that is required to find the inverse of the Hessian matrix in Equation 5.18.

5.8 Chapter Summary

In this chapter, we proposed a real-time workload prediction subsystem that can be used for a better energy conservation strategy. A key component of the proposed subsystem is the modified ELM predictor with a multivariate time series algorithm for VM request forecasting. The improved ELM can deal with the problem of time-varying VM requests, eliminating any restrictions on observation window sizes as well as the number of VM clusters for the ELM predictor. The subsystem is premised on the efficient use of historical VM requests, user cluster algorithms, the current state of the data centre, and an effective prediction window

size.

The long term historical workload time series data from cloud computing platforms can be used for finding the centre of VM size cluster and for filtering the VM class-based on the probability of user behaviour class. These centres and filtered historical data with predefined window size, and the propose ELM-predictor, will improve the accuracy of the estimated number of requests in various VM categories.

We evaluated the predictor performance using Google trace data for different sampling rates and duration of time. The proposed ELM algorithm showed an improvement in accuracy of up to 20%. For a given set of data, it may be possible to derive further improvements in accuracy by using a long term offline filter.

Chapter 6

VM Placement and Machine Condition Index

This chapter introduces a concept called the Machine Condition Index (MCI) of a server. MCI is a single unit that meters all resources relating to servers in a cloud data centre.

For a cloud service provider, MCI is a tool used to compare services, increase efficiency, and ensure manageability of resource usage in the enterprise. MCI is a measure of the extent to which the PM is suitable enough or handling the new or consolidated VM(s) in large scale heterogeneous data centres. MCI will be used to convert the multi-objective optimization so as to solve the VMP problem into a single-objective problem. This is very useful in real-time VMP applications.

In this chapter, we will identify the MCI components and the manner in which we can use it as a cloud resource unit. Then a proposed VMP algorithm uses MCI as a main objective function that will be developed and implemented.

6.1 Introduction

In this thesis, the proposed real-time framework uses VMP to map VMs to PMs in order to minimize the number of PMs required by the set of VMs. This will save energy usage by switching-off unused servers. In other words, the objective of the VMP is to determine the minimum number of PMs required by the set of VMs. Details in Section 2.5.

For proactive dynamic VM consideration, a deterministic approach, optimization-based, technique should be considered to achieve an exact VMP in a cloud data centre. The scope of this chapter is to find an optimal VMP for a heterogeneous data centre for real-time applications. The main part of any VMP algorithm is the PM.

The majority of prior work has focused on CPU utilization of the PM as the main factor for specifying the utilization of a PM. In this chapter, a novel factor named Machine Condition Index (MCI) will be introduced as a new concept in describing the PM that scores PMs in the data centre through pre-defined direct (measurable) and indirect (estimated) values.

MCI will be used to effectively formulate the problem of VMP from that of multi-objective optimization into that of a single-objective. By using MCI as an objective function to optimize the VMs placement, the proposed VMP are imbued with the following characteristics:

- VMP algorithm does not have to consider CPU as their only input. The algorithm is extended to consider other important resources as well, such as memory, storage, bandwidth, cooling, etc. The proposed VMP algorithm takes all PM characteristics into consideration, as opposed to merely workload characteristics.

- MCI takes into consideration the heterogeneity of PMs in terms of power efficiency and capacity.

- MCI can also be used to rank available PMs, which, in turn, can assist with VMP and resource scheduling.

MCI is implemented as a cloud unit for metering all resources related to the services delivered by a cloud service provider. MCI is adopted as an effective tool for comparing the services available by those providers, and can lead to increased efficiency and manageability of resource usage within an enterprise. MCI can be considered as a standard unit for measuring cloud resources within a data centre IT infrastructure. The main advantages of using MCI as a cloud unit are as follows:

- MCI simplifies service offerings into a single unit with a known metric.
- MCI enables organization and cloud users to better predict costs.
- MCI monitor and distribute users' cloud resources effectively and efficiently.

The remainder of this chapter is organized as follows. Description, formulation and features for the proposed MCI will be introduced in Section 6.2. Similarly, multi-objective VMP based MCI will be discussed in Section 6.3. Experimental implementation and analysis of results examining the validity of the MCI as a cloud pricing unit and VM placement algorithms will be described in Section 6.5. Finally, a summary of the chapter will be presented in Section 6.6.

6.2 Machine Condition Index (MCI)

MCI provides a dynamic structure in which all server components can be reduced to a quantifiable variable. It is capable of capturing PM operations environment (power and

cooling), infrastructure (CPU, memory, storage, and network), security (firewall etc.), in addition to data centre service tiers (platinum, gold, silver) and availability attributes.

MCI is a nonlinear index that contains many variables operating separately and being measured independently, or in concert with one another. Meanwhile, variables considered can be both quantitative and qualitative. This implies that the estimation of MCI variable coefficients is premised on measurements and expert knowledge. This will be done by identifying the parameter that will have a more pronounced effect, taking energy-conservation as an example. MCI consists of direct and indirect measurements of different digital components within the data centre. These values are based on:

- Infrastructure components: power and cooling
- Server hardware: CPU, memory, bandwidth, I/O, and storage
- Server software: hypervisor type, OS, and applications
- Service attributes: deployment and component model

MCI is a dynamic assessment rating of various PM elements, through direct and indirect measurements of various imperative digital components in a data centre. The fundamental configuration of the proposed unit is denoted by:

$$\text{MCI} = K \sum_{i=1}^{n} w_i x_i = KW'X \tag{6.1}$$

where $X = [x_1, x_2, \ldots, x_n]'$ measurable infrastructure components (CPU, memory, network, I/O, cooling, etc.); n denotes the number of measurable components; $W = [w_1, w_2, \ldots, w_n]'$ are coefficients calculated based on the component unit power effect concerning the entire system; K signifies the host attribute factor, including security, availability, performance, and priority.

The MCI value can be derived via paramedic estimated based specific cost function estimation. In order to find the n parameters of W in Equation 6.1 we can use:

$$\text{MCI} = f(KW'X) + e \tag{6.2}$$

Where e is an error between estimated value and real values of MCI. Based on the least squares that minimizes the Euclidean distance between the estimated values and the real values, the W' parameters can be calculated by:

$$W = (X'X)^{-1}X'Y \tag{6.3}$$

Given an arbitrary X we can then estimate the cost function for the MCI. The above elements represent the measurements at the macro level. For an operator with more specific resource utilization (e.g. network can be split into firewall rules, IP addresses, storage can be split into primary and secondary storage, etc.), these elements can be measured as allocated, consumed or a combination of both. Figure 6.1 illustrates such an implementation in this data centre.

In all cases, measurements are taken at specific parameter levels and analyzed in order to provide a consumed value. The principle is to apply the weighted cost of the "split out" component so as to derive its coefficient, and the unit value is the multiple of the component scalar value and its coefficient.

MCI is not limited to a specific components, and can be extended, depending on the nature of the server components and available measurements. For example, high-performance disks, installed software, etc. can be added and used.

All data centre resources can be reduced to quantities of the basic unit, via the component "drag coefficients". Drag coefficients are derived via the multidimensional vector scaling

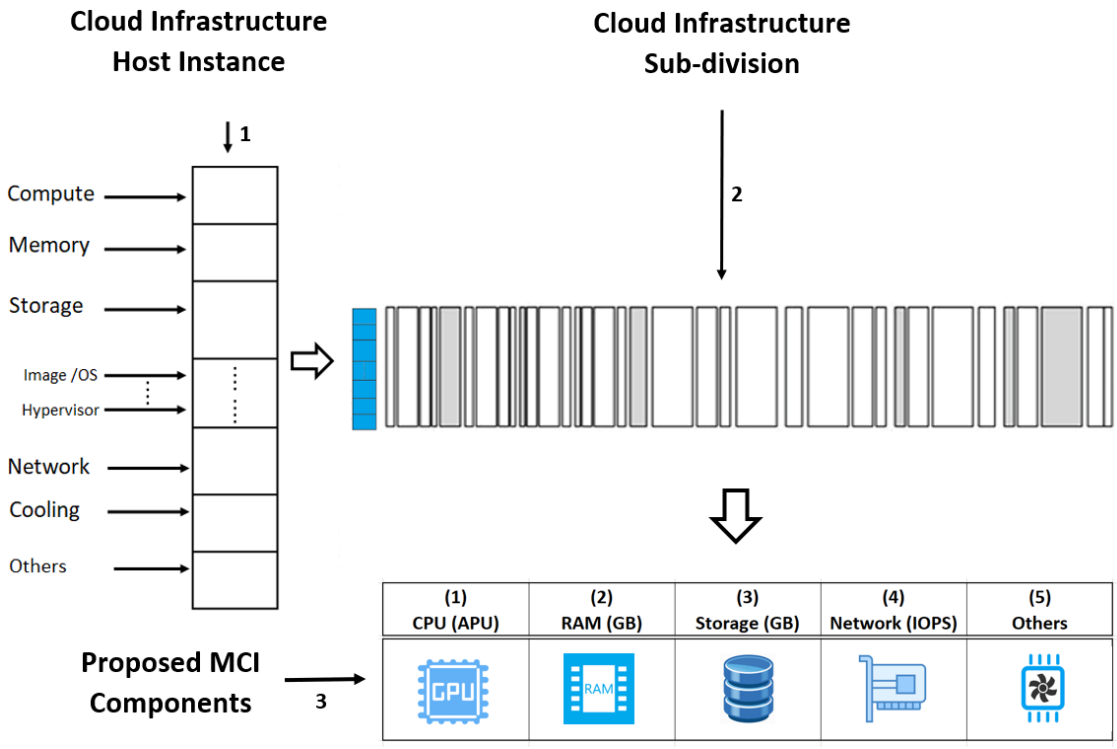

Figure 6.1: Resource allocation based on MCI

method (MDS) using large sample sets, Algorithm 7. Reduced dimensions matrix will be generated using the square root of the dot product of the eigenvectors matrix and the diagonal matrix of the largest eigenvalues.

Algorithm 7 Classical MDS

Input: Data centre resources instant power and usage (from a monitoring tool)
Outputs: Reduced MCI coefficient vector (eigenvalues)

1: Given a power measurement vs percentage usage for each MCI element
2: Create a distance matrix between elements
3: Create a square form for the matrix
4: Find the eigenvalues using Equation 6.3

The development of such a unit assumes great significance for IT resource management and control, since it aggregates and quantifies IT resources as a unified block with an associated unit count. Portions of this resource block can then be assigned to user accounts as budget. Controls can be implemented in a resource metering application on the basis of

unit value, which prevents user account (or cloud admin account) from going beyond the allocated unit amount.

Similar to financial budgeting, this cloud unit can be transferred from one user to another to ensure optimum resource management. A cloud unit provides the best means for controlling IT resources within a data centre. In addition, it enables resource allocation "given" to specific users or departments while maintaining control of overall system resources at an admin level.

For VMP application-based power consumption, a server's MCI is a measure of the extent to which the PM is suitable for handling new or consolidated VM(s) in large scale heterogeneous data centre. It can be used to indicate which PM should be migrated to. MCI combines the operation of two processes: host overload and host underload described in Subsection 2.3.1.

6.3 Multi-Objective VMP based MCI

The VMP represents a multi-objective problem, including a set of p decision variables, q objective functions, and r constraints. Using Equation 6.1, x denotes the decision vector, while y signifies the objective vector. The decisions, objectives, and constraints denoted by X, Y, and G receptively, can be expressed as:

$$Y = F(X) = [f_1(x), f_2(x), ...f_q(x)] \tag{6.4}$$

subjected to:

$$G(x) = [g_1(x), g_2(x), ...g_r(x)] \geq 0 \tag{6.5}$$

Where $x = [x_1, x_2, \ldots, x_p]'$ is the decision vector; $g_i(x)$ is constrain $i$that is related to

decision variables x.

Typically, not all objective functions in Equation 6.4 can be optimized simultaneously. For this reason, an optimal solution will require the concept of Pareto dominance. Pareto optimal solution represents a trade-off approach that is used to identify the lower values for all objectives as compared to any other solution. More applied details can be found in [177, 178].

The number of iteration used to find Pareto solution can be reduced using lower (L_x) and upper (U_x) bounds associated with each objective function, Equation 6.6:

Lower and upper bounds:

$$L_x \leq f(X) \leq U_x \tag{6.6}$$

This section discusses VMP as an optimization problem including power consumption and resource utilization, i.e. optimization with a multi-objective problem. VMP in a large scale data centre comprises of a set of requested VMs and PMs at each discrete time t. VMP is required to incrementally seek placement of requested VMs into available PMs for the next time instant $t + 1$, thus satisfying the constraints and optimizing the considered objectives.

For real-time application, placement decisions are performed at each period t, if there is sufficient information on upcoming VMs and available PMs. Using current PMs states provided by real-time monitoring and the VM predictor proposed in Chapter 5, the VMP algorithm will have the expected type and size of requested VMs along with the maximum number of PMs required.

6.3.1 Objective Functions

Objective functions can represent all MCI components, which have more than one to optimize. Optimization may include power, resource usage (minimum CPU, memory, storage),

time, security, etc. Equation 6.1 can be rewritten as a multi-objective function to include power, QoS, migration cost, security and others.

Power Consumption Minimization

Total power of a PM (P_{server}) can be computed on the basis of servers related to power consumption, such as CPU, memory and Network Interface Card (NIC) utilization:

$$f_1 = \sum_{k=1}^{n} P_{server}\alpha_k = \sum_{k=1}^{n}\sum_{i=1}^{r} w_{ki}x_{ki}\alpha_k \tag{6.7}$$

where f_1 denotes the power objective function, n is the number of PMs required to handle the requested and consolidated VMs for the next time period, α_k signifies a binary variable to show if the PM_i is ON (1 value) or OFF (0 value), and r denotes the number of resources considered. To illustrate, let us consider CPU, memory, storage and network resources. That is $r = 4$ and Equation 6.7 can be rewritten as:

$$f_1(X) = \sum_{k=1}^{n} \left(w_{k1}U_{CPU} + w_{k2}U_{mem} + w_{k3}U_{disk} + w_{k4}U_{NIC} + E_{base_k}\right)\alpha_k \tag{6.8}$$

U_{CPU}, U_{mem}, U_{desk} and U_{NIC} denote CPU, memory, disk and network interface card utilization, respectively. E_{base} represents the idle state of the server.

Offline training can be used to compute weights w_1 to w_4 (for more details, see Algorithm 7). For example, Economou *et al.* [179] used collected component utilization metrics by the operating system to estimate the weights of two different server systems (highly integrated blade server and an Itanium server) for which the power models are depicted as:

$$P_{blade} = 0.236U_{CPU} + 4.47e^{-8}U_{mem} + 0.00281U_{disk} + 3.1e^{-8}U_{NIC} + 14.45 \tag{6.9}$$

$$P_{Itanium} = 0.1108U_{CPU} + 4.05e^{-7}U_{mem} + 0.00405U_{disk} + 635.62 \tag{6.10}$$

where the constant value in the end of each two equations above represents the PM idle power consumption.

Quality of Service

QoS can be used to give more weight to VMs with the highest level of priority associated to the SLA. This objective is proposed to be:

$$f_2 = \sum_{j=1}^{m} (Max_{power} SLA_j \beta_j) \tag{6.11}$$

where Max_{power} denotes the maximum estimated power by all predicted PMs required. It constant value proposed to be large enough to prioritize VMs with large values of SLA, and to vary based on the number of PMs required. On the other hand, β_j is a binary variable that shows if V_j located on PM (1 value) or not (0 value).

Network Traffic

Network traffic can be the estimated by the sum of average network traffic generated by each VM with other VMs located and run on different PMs [180].

$$f_3 = \sum_{j=1}^{m} \sum_{k=1}^{m} (Vnet_{jk} \delta_{jk}) \tag{6.12}$$

where $Vnet_{jk}$ denotes the average network traffic between VMs (V_j and V_k) on different PMs. Put differently, $Vnet_{kk} = 0$. δ_{jk} is a binary variable that shows if V_j and V_k are located in different PMs (1 value) or not (0 value).

All these objectives, f_1, f_2, and f_3, and others, can be represented in a single MCI with

different wights.

6.3.2 Input Data

The input data consists of two parts:

Destination represents the estimated number of PMs required and their specifications including CPU, memory, etc.

$$PM_k = [PM_{CPU_k}, PM_{mem_k}, PM_{desk_k}, PM_{NIC_k}],\ \forall k \in 1, \ldots, n \tag{6.13}$$

where $PM_{CPU_k}, PM_{mem_k}, PM_{desk_k}$, and PM_{NIC_k} represent processing, memory, storage and network resources of PM PM_k and n is the number of PMs.

Source represents the estimated number of VMs in each cluster. As any PM, each VM requires processing, memory, storage resources, it may also include the SLA to indicate its priority level, security level requirements, etc., as shown below:

$$VM_j = \left[VM_{CPU_j}, VM_{mem_j}, VM_{desk_j}, SLA_j\right],\ \forall j \in 1, \ldots, m \tag{6.14}$$

where VM_{CPU_j}, VM_{mem_j}, and VM_{desk_j}, denote processing, memory, and storage resources requested for VM_j respectively. Similarly, SLA_j represents the service level agreement of VM_j, where $SLA_j \in 1, \ldots, s$ and s signifies the highest priority level. m is number of VMs.

The network traffic can be part of source data and will be useful in the placement algorithm. This traffic represents the network communication between different VMs that are run on different PMs, as illustrated in the previous section.

$$VMnet_j = [VMnet_{j1}, VMnet_{j2}, \ldots VMnet_{jm}]\ \ \forall j \in 1, \ldots, m \tag{6.15}$$

6.3.3 Output Data

The solution of objective functions, Equations 6.10 to 6.12 using multi-objective and Pareto principle, will produce a complete placement of all VMs into a minimum number of PMs in the time period $t+1$. Notably, it will be a binary two dimensions matrix ($\mathbf{X}_{m\times n}$) indicates whether each VM is located on candidate PMs or not.

6.3.4 Constraints

Constraints 1-Sufficient Physical Resources: Each PM may have one or more VMs. It should be:

$$\sum_{j=1}^{m} VM_{CPU_j} x_{ji}(t+1) \leq \ PM_{CPU_j}, \ \forall_i \in 1,\ldots,n \tag{6.16}$$

$$\sum_{j=1}^{m} VM_{mem_j} x_{ji}(t+1) \leq \ PM_{mem_j}, \ \forall_i \in 1,\ldots,n \tag{6.17}$$

$$\sum_{j=1}^{m} VM_{desk_j} x_{ji}(t+1) \leq \ PM_{desk_j}, \ \forall_i \in 1,\ldots,n \tag{6.18}$$

which states that the aggregate allocated resources for all the VMs hosted on a PM must not exceed its capacity.

Constraints 2- Unique Placement: All requested and consolidated VMs should be located to run on a single PM.

$$\sum_{i=1}^{n} x_{ji} \leq 1, \ \forall_j \in 1,\ldots,m \tag{6.19}$$

Where x_{ji} refers to a binary variable to indicate whether a VM_j is located on PM_i or

not.

Constraint 3-Assure SLA Provisioning: It is mandatory for VM with the highest level of SLA to be located to run on a PM.

$$\sum_{i=1}^{n} x_{ji} = 1, \ \forall_j : SLA_j = s \tag{6.20}$$

6.4 VMP based Multi-Objective Genetic Algorithm

This section includes a description of the proposed multi-objective algorithms for the VMP problem. This algorithm produces a set of non-dominated optimal solutions that satisfy constraints Equations 6.16-6.20, whereas objectives power, QoS errors, and network traffic are minimized. Non-dominated sorting genetic algorithm (meta-heuristic) are mixed with the main characteristics of a genetic algorithm and here, the concept of Pareto dominance will be used.

Firstly, the algorithm creates a random population for combined objectives $[f_1, f_2, f_3]$. This is because multi objective solutions are ranked into several classes or fronts in accordance to its non-domination level.

All non-dominated solutions are included in front level 1. This front represents the best efficient set and is temporarily disregarded from the population. Iteratively, non-dominated solutions are determined and assigned to front level 2. This new front denotes the second-best efficient set and the process is repeated until the population is empty. This procedure is called a fast non-dominated sort.

In order to maintain population diversity, a second value called crowding distance is calculated for solutions that belong to the same non-dominated front. This measure estimates population density around a solution within the objective space. The extreme points of

Algorithm 8 Multi-objective VMP based on MCI

Inputs: Predicted PMList M_k, VMList V_j, Objective funs [f_1, f_2, f_3], and Constraints Equations 6.15-6.20
Initialization: Population size, number of iterations, upper and lower bounds
Outputs: VMs allocated with min power

1: Random initial population POP_0
2: Calculate fitness functions, Equations 6.7, 6.11, and 6.12.
3: Use the fast non-dominated sort to assign a rank to each solution
4: Calculate the crowding distance for each solution
5: **while** stop criteria is not reached **do**
6: Select parents
7: Apply crossover and mutation
8: Calculate the objective functions
9: Use the fast non-dominated sort to assign a rank to each solution in $POP_0 \cup POP_1$
10: Calculate the crowding distance of each solution in $POP_0 \cup POP_1$
11: Replace solution in POP_0 with the best solution in $POP_0 \cup POP_1$
12: **end while**

each front are assigned with an infinite distance in order to preserve them and have them introduce more dispersion in the population. As a consequence, every chromosome will have two attributes, the non-domination rank and a crowding distance.

Next, a binary tournament is applied. Two solutions are picked randomly from the population, and the winner is the lowest-ranked individual; if the rank is the same for both, the winner will be the one with the highest crowding distance. This strategy is applied to select pairs of parents, after which the crossover and mutation operators are applied to obtain a new population POP_1. Finally, the fast non-dominated sort and the crowding distance are applied to all solutions in $POP_0 \cup POP_1$, whereas the N best-ranked solutions are retained to the next population. This process is repeated in a predefined number of generations, see Algorithm 8.

In order to apply genetic operators, a proper encoding scheme (chromosome) consisting of m genes presents the number of VM. The value of gene is an integer between 1 and n, which denotes the PM where the VM is allocated. Figure 6.2 shows an example of VMP

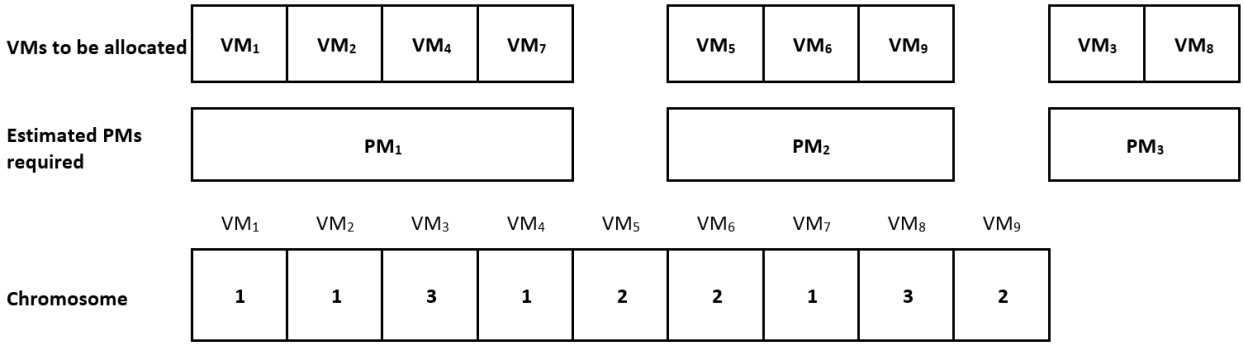

Figure 6.2: VMP resources and corresponding chromosome: An example

and its corresponding chromosome.

6.5 Experimental Results

Simulation results obtained by proposed MCI and VMP-based multi-objective algorithms in carefully designed experiments validate its effectiveness, taking into considerations the challenges associated with the resolution of the VM consolidation problem introduced in this thesis.

In next Subsection 6.5.1, we use MCI as a pricing unit for a private cloud. In our second experiment, we perform the evaluation of using the MCI on the real-time VM consolidation framework. Subsequently, we compare the quality of solutions obtained by a simple bin packing algorithm against an optimal solution for the same data used in previous chapters, as explicated in Section 3.3. Our last experiment will evaluate VMP using MCI as a single objective function to minimize data centre power for different scenarios of PMs, VMs sizes, and heterogeneity.

6.5.1 MCI as a Cloud Pricing Unit

MCI can be used as a pricing unit that can be implemented in a data centre used for delivering cloud services. Information pertaining to these resources was collected from a private data centre IT infrastructure that is capable of hosting thousands of applications. This centre has the following specifications:

1. In computing, an Intel x86 platforms or Oracle/Sun SPARC with 4 processors are used. A memory pool was used for high-speed data caching.

2. Computer network had at least one network router, firewall, load balance, network switch (core, distribution and access), network card on the server and internet bandwidth.

3. A storage system comprises of two defining elements: storage size and its performance. The cloud unit describes the amount of disk under RAID 0; RAID 1 is used where 50% of the total amount of disk space gets lost due to redundancy, after which a factor of 2 is applied to correct for this loss. The same principle is applicable to all disk redundant formats.

4. Conditioned redundant power and cooling of the IT infrastructure consists of at least one power-conditioned uninterruptible power source. Cloud instance power represents the measure of the amount of power that is consumed by the cloud instance measured in Watts.

In order to represent all data centre components, the proposed MCI unit includes one CPU with 1.3 GHz, 1 GB RAM, 10 GB storage, 10 GB internet traffic, 1 instance of Ubuntu OS, and instance of Apache. A Standard Cloud instance consuming one unit is assumed to be equivalent to 8 Watts.

The index values or weighted coefficients are determined from the proportionate cost of the components in the unit divided by their scalar value: i.e. the CPU constitutes 61.75% of the unit cost or 0.6175 of 1 and has a scalar value of 1; the coefficient is 0.61754/1; similarly, the network cost is $0.28 and has 10 GB. The network constitutes 9.866% of the unit cost or 0.09867 of 1 with a scalar value of 10. Table 6.1 depicts the cloud component costs in the data centre along with the weighted coefficients.

Table 6.1: An example of component cost and weighted coefficients

Parameter	**Coefficient**	**Value**	**Monthly Unit Cost**
CPU (1.3 GHz)	1	0.6175	$17.83
RAM (GB)	1	0.1883	$5.44
Storage (GB)	10	0.0485	$1.40
IOPS	10	0.0470	$1.36
Network (GB)	10	0.0987	$2.85
Total:		1	$28.87

It is notable that this factorization can be used for any dimensional scaling. For example, cloud infrastructure with double the resources (i.e. 2 compute nodes, 2 GB memory, 20 GB storage, 20 IOPS, 20 GB network) will be equivalent to 2 cloud units. This decomposition principle can also be used to compute the size of any cloud instance in singularity with simplicity.

Notably, these weighted coefficients and costs vary from one cloud computing provider to another. In this implementation, all the cloud resources were inserted in one place. Put differently, a specific data centre was used to implement this unit, which enabled the cloud provider to make cost estimations for each component used by the user and VM as well. At the same time, the service provider could monitor, measure and estimate the values of all used resources.

Table 6.2: Some of the MCI components to estimate the energy used

Parameter Value	**Unit**
P sleep	107 Watt
P idle	300.81 Watt
P peak	600 Watt
E on to sleep	5510 Joule
E sleep to on	4260 Joule

6.5.2 Real-Time VM Consolidation Framework

As a case study, the output of the prediction algorithm used in Section 5.7 will be applied in this experiment. In particular, the Task Event Table from Google dataset with 5 minutes prediction windows is used.

Heuristic or bin packing based algorithms are the most suitable for real-time applications because they require less computational resource and they are very fast, as shown in section 2.5.2. However, bin packing algorithms also provide a sub-optimal distribution of VMs. For real-time VM consolidation, we use best fit decreasing (BFD) algorithms to assign VMs to specific hosts. In addition, we use prediction algorithm to estimate the type and size of requested VMs, in order to pre-specify the number of PMs required.

To use MCI in VMP, we developed the simple power aware best fit decreasing (PABFD) algorithm proposed in [93], Algorithm 9. In our modified PABFD method, MCI was not only used for PMs, but also for each VM. MCI is used to classify VMs, by arranging them in decreasing order. Subsequently, for each VM, it checks all PMs to identify a suitable solution where power consumption is minimum.

Figure 6.3, compares the total consumed energy during the entire test period for 125 hosts (PMs) only. The energy measurements used MCI with special values are shown in Table 6.2.

The results of this experiment demonstrate that MCI can represent used as a VM and a

Algorithm 9 PABFD

Inputs : arranged decreasing list MCI, VMList
Outputs : VMs allocated with min power

```
for each VM in arranged decreasing list do
    PM with min. MCI ← VM with max MIC requirements (e.g. CPU)
    allocated PM ← null
    for each PM in the PM List do
        if PM has enough resources for this VM then
            Calculate new MCI on the PM
            if MCI < min MCI then
                allocated PM ← VM
                Update min MCI ← new MCI
            end if
        end if
        if allocated PM ≠ null then
            allocate the VM in specified PM
        end if
    end for
end for
```

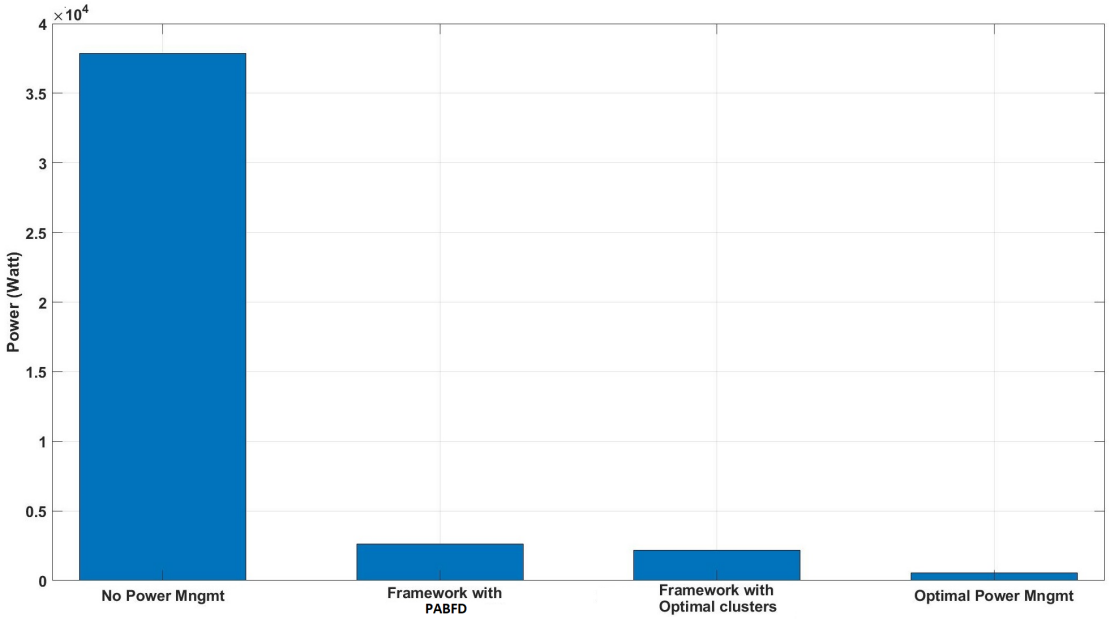

Figure 6.3: Total consumed energy during the entire test period (for 125 hosts only)

server condition index. This leads to most of the power estimation that were previously used in the works concerning VM provisioning based energy consumption, which can be treated as a special case of the MCI.

6.5.3 Power Consumption Experiment Results

In this experiment, we use a simulated data centre, including two types of PMs and different types of VMs to create a heterogeneous environment. Table 6.3 depicts the number of VMs and PMs used in each experiment. Table 6.4 gives a sample of PMs considered in first experiment. Table 6.5 gives the configuration of VMs used in first experiments. These VMs are randomly generated on the basis of Amazon EC2 instances types. The values of VMs and PMs for these experiments can be found in[1]

Table 6.3: No. of PMs and VMs in three Experiments

	No. of VMs	**No. of PM**
Exp1	8	4
Exp2	50	12
Exp3	1000	100

Table 6.4: PMs considered in the Exp1

PM Type	**CPU (ECU)**	**Mem (GB)**	**Storage (GB)**
PM1	180	512	10000
PM2	350	1024	10000
PM3	180	512	10000
PM4	350	1024	10000

Table 6.5: VMss considered in the Exp1

VM Type	**CPU (ECU)**	**Mem (GB)**	**Storage (GB)**
VM1	4	15	80
VM2	8	30	160
VM3	2	4	32
VM4	4	8	80
VM5	8	15	160
VM6	16	30	320
VM7	32	60	640
VM8	2	15	32

[1]https://github.com/flopezpires/iMaVMP/tree/master/inputs

Amazon EC2 instances (VMs) are categorized into five types of categories with more than 108 types of VMs along with different values in Virtual CPUs (vCPU) and main memory (GB) capacity[2], see Table 4.1.

As a case study, the power model assumed for each to be represented by two PM models is described in Equation 6.9 and only the power objective function is given by Equation 6.8. The idea is to evaluate the proposed MCI model to solve the VMP problem with homogeneous (blade or Itanium PMs), non-homogeneous or hybrid (40% blade and 60% Itanium) PMs. All other factors such as data centre infrastructure, number of instantly requested VMs, the size and instantly available PMs, etc are assumed to be available by the data centre monitoring tool.

MCI depends on the information provided by any cloud monitoring tools such as Data Center Infrastructure Manager (DCIM) which provides elaborate information about a server configuration, hardware, network connections, installed software, and so on. DCIM profiles the power consumed by each part of the hardware in a data centre [44].

Figure 6.4 represents the optimization progress versus generation for Experiments 1, 2 and 3 for a homogeneous blade data centre in Equations 6.9 and population size of 100. From these figures, oscillated convergences in all experiments are directly proportional to increase the size of the data centre. Put differently, a growing number of PMs and requested VMs will increase the oscillation. This oscillation is attributed to the use of random Gaussian distribution for the mutation selection option to each entry of the parent vector. Typically, the amount of mutation, which is proportional to the standard deviation of the distribution, lowers at each new generation.

Figure 6.5 compares the normalized total energy with the optimal energy calculated in the proposed algorithm for 4 PMs in Experiment 1. The proposed optimal algorithm saves

[2]http://aws.amazon.com/ec2

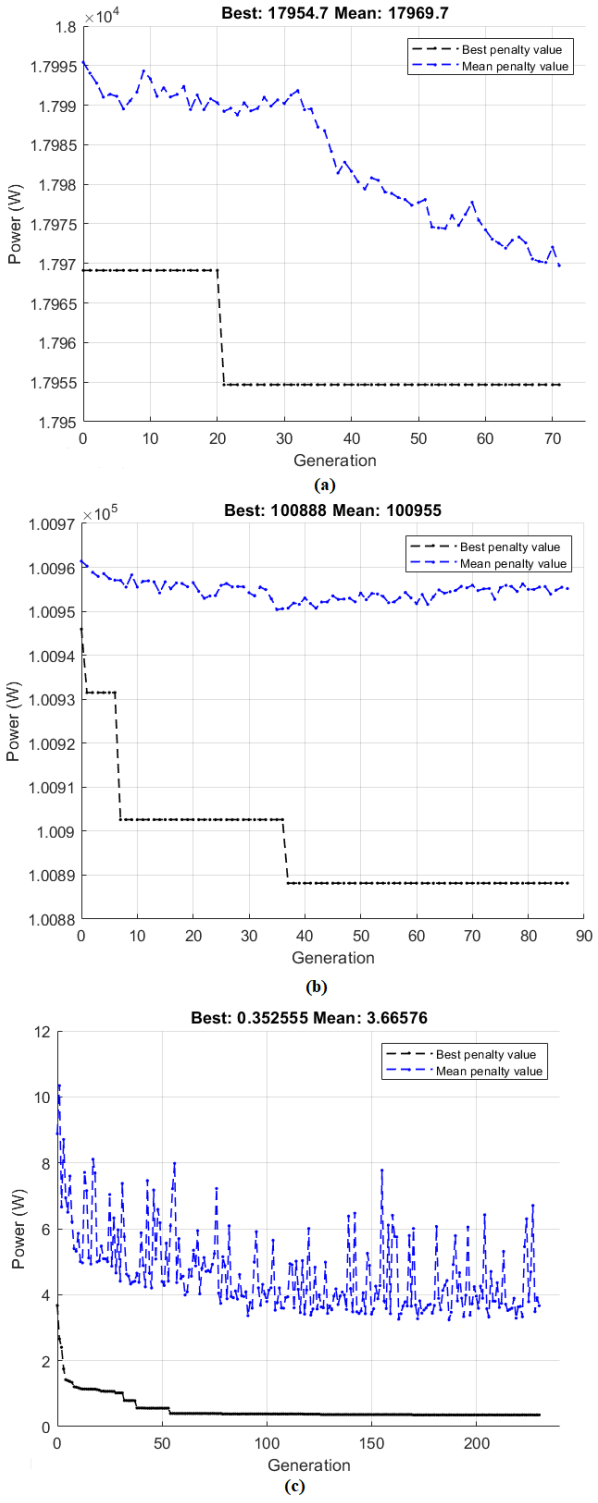

Figure 6.4: Energy objective optimization progress versus generations for the homogeneous blade data centre in (a) Exp 1 (b) Exp 2 and (c) Exp 3

more than 60% of the regular consumption of the data centre. Results show only one PM that is sufficient enough to accommodate all 8 VMs, as illustrated in Figure 6.6

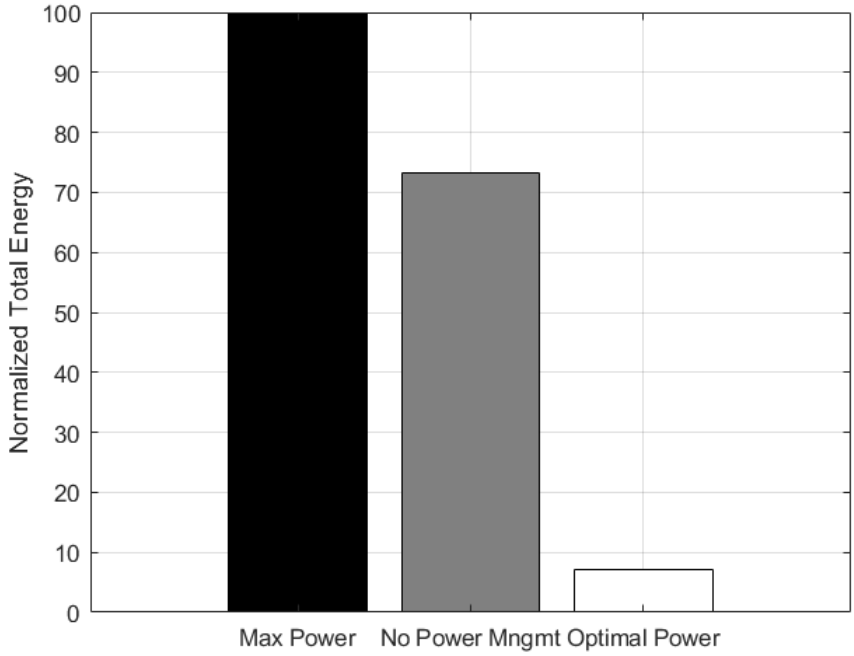

Figure 6.5: Total consumed energy vs optimal VM distribution in Experiment 1 with Blade machine

Figure 6.7 three types of PMs used for the data centre in Experiment 1. Blade, Itanium and hybrid with 40% blade and 60% Itanium machines. The data centre with Itanium machines is shown to have the lowest power usage. This implies that the proposed MCI based optimization process is completely predicated on the MCI values.

Table 6.6 compares the results of the three experiments with different types of PMs in a data centre (Blade, Itanium and Hybrid).

The average executing time for all experiments is relatively small as compared to the monitoring period, see section5.4 for the suitable monitoring period. Executing time in Table 6.6 represents the average five times running for each experiment. As any regular machine learning algorithm, GA convergence time increases by augmenting the population size, i.e. increase the number of PMs and VMs/tasks.

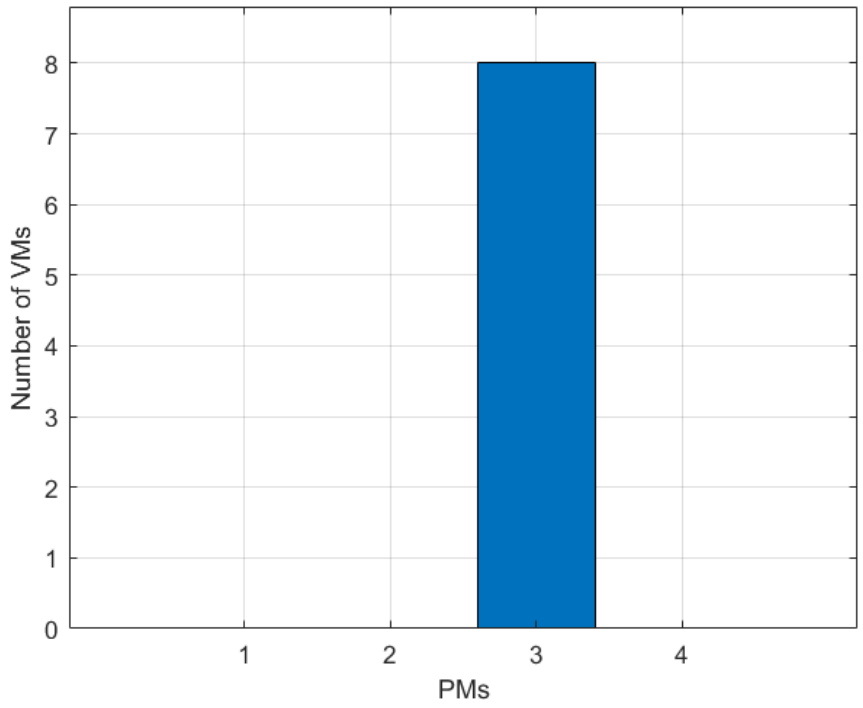

Figure 6.6: Distribution of VMs on PMs in Experiment 1 with Blade machine

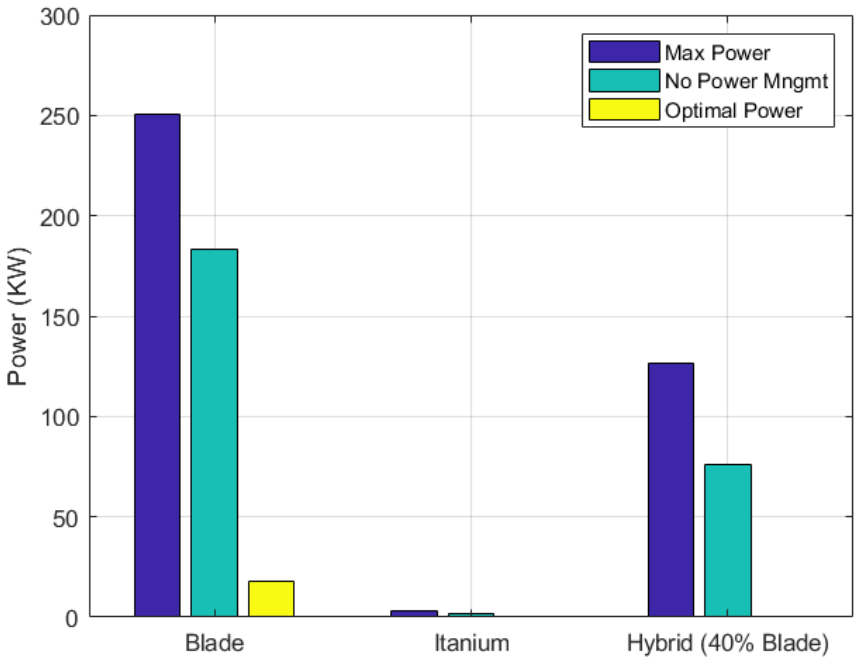

Figure 6.7: Experiment 1 with three types of PMs (Blade, Itanium and hybrid)

Table 6.6: Optimal VMP Results Comparisons

PM Types	Exp. No.	Max Power (KW)	No Power Mngmt (KW)	Optimal Power (KW)	No. of Iteration	Function count	Execution time (sec)	% of PM used
Blade	**1**	250.3	183.3	18.0	71	7,201	3.9	**25**
	2	751.0	617.8	100.9	87	8,801	6.0	**58**
	3	6,258.3	6,258.3	1,991.9	230	23,101	67.8	**100**
Itanium	**1**	2.8	2.0	0.7	58	5,901	3.2	**25**
	2	8.5	7.1	4.5	109	11,001	6.9	**58**
	3	70.6	70.6	64.9	160	16,101	49.8	**100**
Hybrid (40% blade)	**1**	126.6	76.2	0.7	60	6,101	3.2	**25**
	2	297.8	197.4	4.1	152	15,301	9.8	**58**
	3	2,545.6	2,528.6	853.4	352	35,301	111.3	**99**

In experiment 3, all PMs of the data centre used, 100% PM required. In this experiment, the number and size of VMs were massive and almost needed all PMs. The proposed algorithm saves power, particularly in hybrid, given that the algorithm uses the PMs with less power. For example, no power management and optimal power for Itanium were 70.6KW and 64.9KW, respectively, i.e. 8% saving. The power management and the optimal power for Hybrid were 2528.6KW and 853.4KW, respectively, i.e. 66% saving. However, both experiments almost used the same number of PMs. 100% and 99% of PMs were used in Itanium and Hybrid experiments.

Significant differences are found when using MCI to model PM power as compared to modeling based on CPU only. MCI takes the effect of server components and environments. These differences are identified when we compare optimal power usage in Table 6.6 with the actual power usage for each PMs' CPU. Additionally, there is a large difference in power usage between Blade and Itanium servers in all experiments. These differences are unequaled with the power characteristics of these processors. Table 6.7 depicts the architectural power and clock rate for the two processors using the McPAT package version 1.3[3]. McPAT is a micro-architectural multi-core power and area estimator. Further details about the package

[3]https://github.com/HewlettPackard/mcpat

can be found in [181].

Table 6.7: Blade and Itanium CPU characteristics using McPAT

Architecture Characteristics		**Blade (AMD Turion)**	**Itanium 2**
Clock Rate		**2.2 GHz**	**4 \times 1.5 GHz**
Memory		**512 MB SDRAM**	**1 GB DDR**
Dynamic	Peak Dynamic	68.07 W	61.72 W
	Runtime Dynamic	101460 W	29638 W
Static	Subthreshold Leakage	9.85 W	8.61 W
	Subthreshold Leakage with power gating	4.82W	4.23 W
	Gate Leakage	1.21 W	1.04 W
Total Power		**101,544 W**	**29,714 W**

These experiments tested MCI for hardware components only. However, as discussed in Section 6.2, MCI can be trained to include all qualitative and quantitative quantities pertaining to server power in a heterogeneous environment.

6.6 Chapter Summary

MCI is a unified index proposed and developed as a scalar value to rank hosts in cloud data centres for energy consumption on the basis of VM consolidation. For any host, MCI has many independent quantifiable and non-quantifiable parameters, and converts these parameters into a useful single and unique index value to measure the effectiveness with which the PM fits the new and migrated VMs in order to reduce energy. The suggested index showed the simplicity and feasibility of implementing the MCI as a benchmark to provide a sense of the required investment required for bringing the host to a certain level of energy saving. The suggested index also demonstrated the simplicity of converting the multi-objective function to a single objective function, which assumes the greatest significance in real-time VM consolidation with optimal displacement.

Chapter 7

Conclusion and Future Work

7.1 Conclusions

This thesis proposes a comprehensive real-time VM consolidation framework that focuses on energy consumption in large-scale heterogeneous data centres. In particular, thesis focused on proactive dynamic VM consolidations in data centres with heterogeneous environments.

The framework introduces and proposes various techniques and algorithms that could be summarized into three main subsystems. Upon submitting a VM request from end-users to a data centre, the first step is to cluster VMs based on users' behaviour, the number of VM requests, and the number of VM re-allocated by discovering the behaviour of the user and the VM. Second, the prediction subsystem based on modified ELM is used to predict the number of VMs and estimates the number of PMs required for the next period of time. Third, the optimal placement for new and migrated VMs algorithms has been undertaken through a multi-objective genetic-based algorithm using a novel index named MCI.

MCI hinges on measured and estimated quantitative and qualitative components; it was used to efficiently estimate the objective function in the VMP algorithm,. MCI can also be

used as a pricing model for cloud service providers and may include any types of components, including the infrastructure of the data centre, server hardware, server software, and service attribute.

Though the analysis and implementation, key observation:

1. We illustrated that dynamic VMs consolidation and resource allocation policies can play an important role in energy-efficient allocation of resources. To that end, a systematic framework was proposed to create a comprehensive review of components, algorithms, techniques, and tools that were used in previous publications in these fields.

2. Various clustering techniques used for analyzing workload characteristics do not provide a structured model that can be used for conducting simulation. Workload analysis needs to explore more than coarse-grain statistics and cluster centroids. In order to capture the patterns of clustered individuals it is also necessary to conduct an analysis of the parameters and examine the trends of each cluster characteristic.

3. Efficient VMs clustering process should not only use historical data only, but also include user behaviour, clustering indices, the current state of the data centre, and effective monitoring window size.

4. Identifying users' behaviours based on the number of VMs requests strongly influences the overall cloud workload prediction process. Under the proposed framework, a simple and fast filtering process through VM and users behaviour helps improve pre-processing, clustering, and prediction accuracy.

5. The key component of our proposed workload forecast is the modified multivariate time series ELM predictor. ELM predictor eliminates any restrictions on the observation

window sizes and the number of VM clusters for the ELM predictor.

6. VMs predictor sustains the strength of the ELM algorithm in fast prediction, but offsets its accuracy weakness as compared to other NNs. This accuracy is improved either by increasing the number of feedback (number of the previous state), or by reducing the prediction window size via the utilization of modified multivariate ELM.

7. The fast pre-processing, mapping, filtering, ELM and single-objective optimization algorithms constitute an efficiency real-time VMs consolidation system for a large-scale heterogeneous data centre.

8. Extensive simulations were carried out to evaluate the proposed framework components separately with both synthetic and realistic data traces. Simulation results show that proposed algorithms can achieve better and efficient results in comparison to some existing algorithms.

9. With the use of MCI, proposed VMP algorithm takes into account the heterogeneity of PMs in terms of capacity and power efficiency.

10. Unlike most algorithms that only consider CPU as their primary input to measure power, MCI was extended to consider other important resources as well, such as memory, storage, bandwidth, cooling, etc. This will lead to enhanced performance and more accurate power consumption.

11. Most VMP algorithms compare their proposed algorithms against trivial heuristics. To that end, a comparison against real data, Google Trace and simulated Amazon EC2 used in this work yielded more meaningful results.

12. The proposed energy-aware VMP based MCI scheme sustains the strength of the ge-

netic algorithm in optimality and efficiency, but offsets its weakness in slow convergence speed. By placing the VMs running the same application to closer cores on the multi-core system based on their traffic rates, energy consumption and delays in communication can be reduced significantly.

7.2 Future Work

1. Consideration of QoS parameters related to VMs performance such as availability, response time and reliability is imperative as part of the consolidation process. It is very important to ensure that the level of QoS is maintained in accordance with SLA, while attempts are made to fully utilize data centre resources. These element can be a part of the MCI.

2. Multi-criteria VMs selection models that take into consideration multiple infrastructure resources, can be used to improve the consolidation process, particularly for energy consumption. For example, it is useful to have a rule-based system, e.g. ANFIS, so as to improve the process of VMs selection according to environment states.

3. Additional implementation is required for the proposed MCI for different practical PMs and different cloud monitoring tools for different vendors. This will help identify a unique MCI for each type of machine and environment. This process can be automated using an unsupervised machine learning algorithm in order to estimate the MCI coefficient through continuous monitoring and adjustment.

4. Most of VMP algorithms and techniques have neglected the security-related objective in the VMP operations. Security is one of the crucial factors that must be considered in future VMP researches and studies.

5. VM migrations usually occur when there is over/under-utilization of the resources. Extra VMs migration may affect energy efficiency, thus resulting in further power consumption. Balancing should be done to avoid unnecessary VM migration. This balance results in energy savings due to turning off PMs and energy ascribed to the migration process. Adding migration cost on MCI and the VMP algorithm will also increase energy efficiency.

6. Failures due to power outages or network components are referred to as correlated failures. The impact of these failures can cause reliability to be overestimated by at least two orders of magnitude. Correlated failure impact on energy consolidation necessitates more attention in future researches.

7. Predict VM release times for submitted requests based on historical traces when VMs' release times are typically not specified by the client. It is useful to build a prediction process that is able to estimate VM release time and add that to VMP algorithm.

8. It is important to enable a collaboration link with the CloudSim group in order to integrate the proposed workload generator as an add-in of the existing framework implementation, which paves the way for its publicly availability.

Appendix A

Systematic Clustering Example

This example illustrates the detailed steps implementation of the systematic clustering algorithm, described in Section 4.4 for predication purposes.

In this example, we assume there are three user defined VMs/tasks with CPU and memory distributed values shown in table A.1 after pre-processing stage.

Table A.1: Task distribution, fixed length [168]

	Distribution (AD Value)	**Parameters**	**Distribution (AD Value)**	**Parameters**
Task	**Requested CPU**		**Requested Memory**	
T_1	Generalized Extreme	ξ= -0.016,	3P Lognormal	μ = -4.342,
		ρ = 0.02098,		σ = 0.569,
		μ = 0.01954		T = -2.399E-4
T_2	Weibull	k = 0.9594,	3P Weibull	k = 2.528,
		λ= 0.09795		λ = 0.0703
				T = -9.294E-3
T_3	3P Lognormal	μ = -6.120,	3P Lognormal	μ = -5.907,
		σ = 1.897,		σ = 0.877,
		T = 6.41E-6		T = -2.204E-4

Figure A.1 and A.2 reveal the CPU and memory of 1000 VMs requests based on distribution described in Table A.1.

Table A.2 represents a 3 predefined VMs with 650 requests for the same duration (sam-

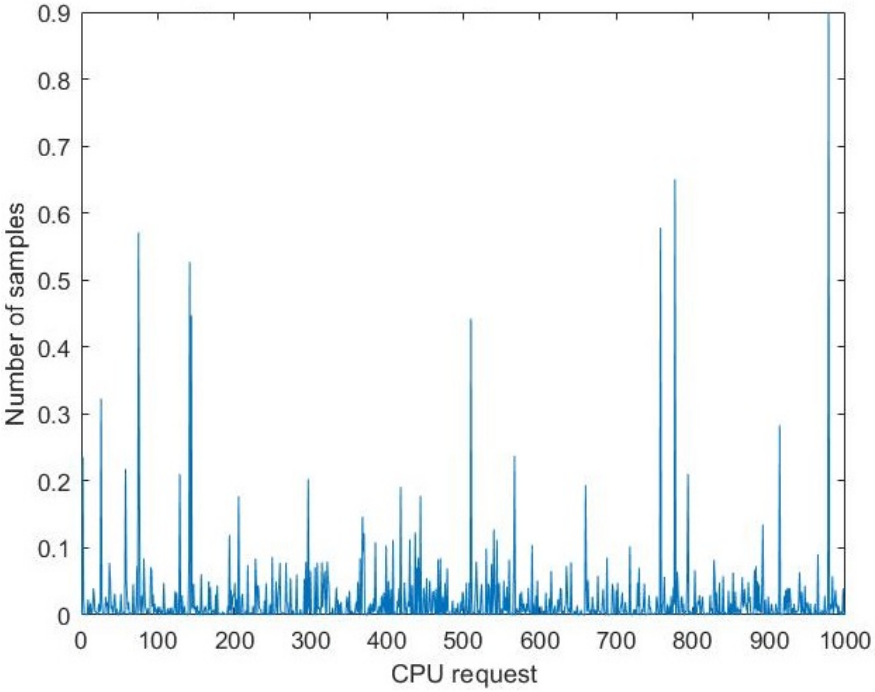

Figure A.1: CPU requests proposed distributions

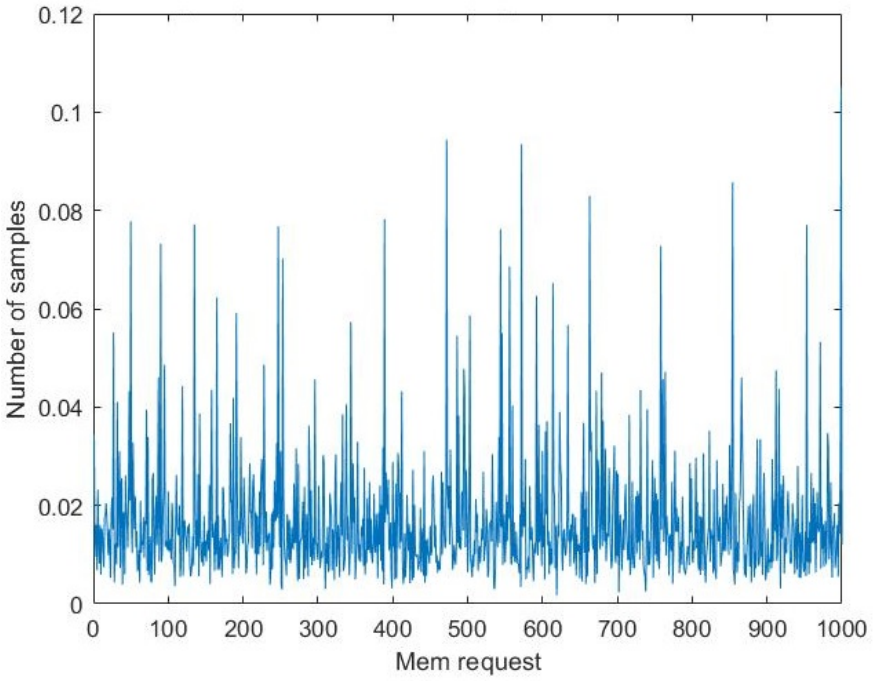

Figure A.2: Memory requests with proposed distributions

pling period).

Table A.2: Pre-defined VMs

VM Predefined size	**Requested CPU**	**Requested Memory**	**No. of request**
Small	0.1	0.1	300
Medium	0.2	0.2	200
Large	0.3	0.3	150

To select the best algorithm and number of clusters based proposed systematic clustering described in Algorithm 2, we should follow the following steps:

1. Using known VMs during this period (Table A.2), find the average number of clusters for all candidate clustering algorithms after find optimal number of clusters based on clustering indices. We select k-means and k-means++ for simplicity, and we found the average for number of clusters in both k-means and $k$means++ are 3 and 5.67, respectively. Details can be found in Table A.3. Compare with active number of pre-defined VMs (e.g. 3), we select k-means for this interval (monitoring period).

Table A.3: Pre-defined VMs optimal number of cluster for different clustering indices

	k-means		**k-means ++**	
Index	**Value**	**Best**	**Value**	**Best**
Calinski_Harabasz	4.61E+30	3	2.06E+31	15
Davies_Bouldin	1.37E-14	3	1.37E-14	4
Dunn	1	3	1	3
Silhouette	1	3	1	4
Wemmert_Gancarski	1	3	1	4
Xie_Beni	5.22E-29	3	5.22E-29	4
Average		3		**5.67**

2. Filter unrepeated VMs/tasks using DBSCAN. Figure A.3 shows the user defined VMs before and after DBSCAN filtering process. We notice that only 7 VMs will be excluded from clustering process. These 7 VMs assumed to be consolidated in a separate PMs for each.

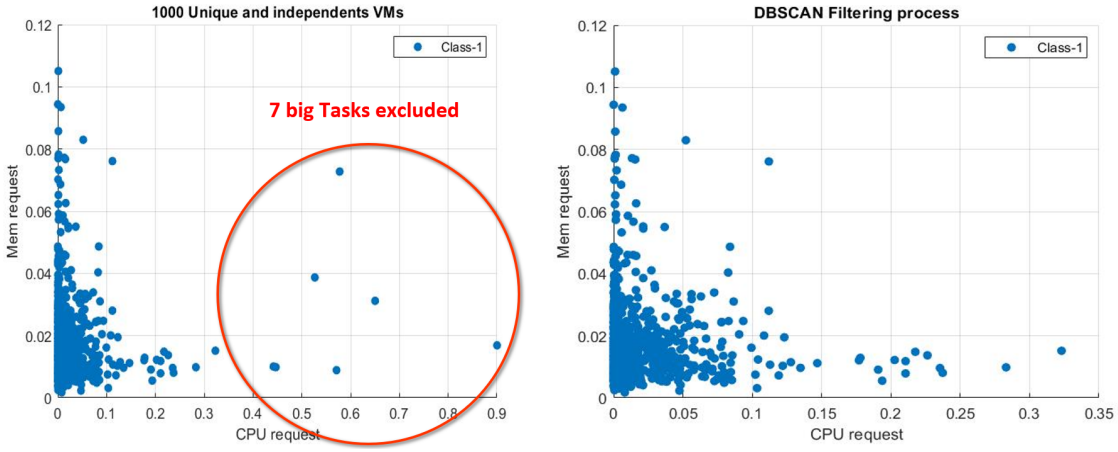

Figure A.3: DBSCAN filtering for the 1000 unique independents VMs

3. For accurate predication, we need to identifies dissimilar clusters that are far from each other and compact, thus Davies-Bouldin (DB) and Dunn (DI) are best candidate indices for prediction application.

4. The last step is to find optimal number of clusters for the candidate clustering algorithms (k-means in this example) using selected clustering indices (DB and DI indices), Table A.4.

Table A.4: Candidate clustering indices based selected algorithm for user defined VMs/tasks

	$k-$**means**	
Index	**Value**	**Best**
Davies_Bouldin	0.59	4
Dunn	0.02	2
Average		**3**

The optimal number of the unknown VMs/tasks during this monitoring period will be 3, and candidate clustering algorithm will be k-means.

Bibliography

[1] S. Ismaeel and A. Miri, "A universal unit for measuring clouds," in *IEEE International Conference on Humanitarian Technology Conference (IHTC)*, pp. 1–4, May 2015.

[2] S. Ismaeel, R. Karim, and A. Miri, "Proactive dynamic virtual-machine consolidation for energy conservation in cloud data centres," *Journal of Cloud Computing*, vol. 7, no. 1, pp. 1–28, 2018.

[3] K. Hwang, J. Dongarra, and G. C. Fox, *Distributed and cloud computing: from parallel processing to the internet of things*. Morgan Kaufmann, 2013.

[4] S. Petrović, A. Colangelo, O. Balyk, C. Delmastro, M. Gargiulo, M. B. Simonsen, and K. Karlsson, "The role of data centres in the future danish energy system," *Energy*, vol. 194, pp. 1–14, 2020.

[5] Natural Resources Defense Council (NRDC), "Data Center Efficiency Assessment," tech. rep., Natural Resources Defense Council (NRDC), August 2014.

[6] D. Sitaram, H. L. Phalachandra, G. S, S. H V, and S. TP, "Energy efficient data center management under availability constraints," in *2015 Annual IEEE Systems Conference (SysCon) Proceedings*, pp. 377–381, April 2015.

[7] J. Prevost, K. Nagothu, M. Jamshidi, and B. Kelley, "Optimal calculation overhead for energy efficient cloud workload prediction," in *World Automation Congress (WAC), 2014*, pp. 741–747, Aug 2014.

[8] X. Fu and C. Zhou, "Virtual machine selection and placement for dynamic consolidation in cloud computing environment," *Frontiers of Computer Science*, vol. 9, no. 2, pp. 322–330, 2015.

[9] A. Beloglazov, *Energy-efficient management of virtual machines in data centers for cloud computing*. PhD thesis, Department of Computing and Information Systems, The University of Melbourne, 2013.

[10] L. Zhang, Z. Li, and C. Wu, "Dynamic resource provisioning in cloud computing: A randomized auction approach," in *Proceedings IEEE INFOCOM*, pp. 433–441, 2014.

[11] A. Abdelsamea, E. E. Hemayed, H. Eldeeb, and H. Elazhary, "Virtual machine consolidation challenges: A review," *International Journal of Innovation and Applied Studies*, vol. 8, no. 4, pp. 1504–1516, 2014.

[12] 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.

[13] E. Arianyan, H. Taheri, and S. Sharifian, "Novel energy and SLA efficient resource management heuristics for consolidation of virtual machines in cloud data centers," *Computers & Electrical Engineering*, vol. 47, pp. 222–240, 2015.

[14] S. Ismaeel, A. Al-Khazraji, and A. Miri, "An efficient workload clustering framework for large-scale data centers," in *IEEE 8th International Conference on Modeling, Simulation and Applied Optimization (ICMSAO)*, April 2019.

[15] T. Mastelic, A. Oleksiak, H. Claussen, I. Brandic, J.-M. Pierson, and A. V. Vasilakos, "Cloud computing: Survey on energy efficiency," *Acm computing surveys (csur)*, vol. 47, no. 2, pp. 1–36, 2014.

[16] E. Feller, *Autonomic and energy-efficient management of large-scale virtualized data centers*. PhD thesis, Université Rennes 1, 2012.

[17] V. De Maio, G. Kecskemeti, and R. Prodan, "A workload-aware energy model for virtual machine migration," in *2015 IEEE International Conference on Cluster Computing*, pp. 274–283, Sep. 2015.

[18] D. Breitgand, D. M. Da Silva, A. Epstein, A. Glikson, M. R. Hines, K. D. Ryu, and M. A. Silva, "Dynamic virtual machine resizing in a cloud computing infrastructure," Jan. 2 2018. US Patent 9,858,095.

[19] Z. Gong, X. Gu, and J. Wilkes, "Press: Predictive elastic resource scaling for cloud systems," in *2010 International Conference on Network and Service Management*, pp. 9–16, Oct 2010.

[20] S. Sotiriadis, N. Bessis, C. Amza, and R. Buyya, "Elastic load balancing for dynamic virtual machine reconfiguration based on vertical and horizontal scaling," *IEEE Transactions on Services Computing*, vol. 12, pp. 319–334, March 2019.

[21] D. Breitgand, D. M. Da Silva, A. Epstein, A. Glikson, M. R. Hines, K. D. Ryu, and M. A. Silva, "Dynamic virtual machine resizing in a cloud computing infrastructure," Sept. 17 2012. US Patent App. 13/621,526.

[22] F. Larumbe and B. Sansò, "Elastic, on-line and network aware virtual machine placement within a data center," in *IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pp. 28–36, 2017.

[23] W. Fang, X. Liang, S. Li, L. Chiaraviglio, and N. Xiong, "Vmplanner: Optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centers," *Computer Networks*, vol. 57, no. 1, pp. 179–196, 2013.

[24] F. Ramezani, J. Lu, and F. Hussain, "An online fuzzy decision support system for resource management in cloud environments," in *2013 Joint IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS)*, pp. 754–759, June 2013.

[25] M. Tang and S. Pan, "A hybrid genetic algorithm for the energy-efficient virtual machine placement problem in data centers," *Neural Processing Letters*, vol. 41, no. 2, pp. 211–221, 2014.

[26] X.-F. Liu, Z.-H. Zhan, K.-J. Du, and W.-N. Chen, "Energy aware virtual machine placement scheduling in cloud computing based on ant colony optimization approach," in *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, GECCO'14, (New York, NY, USA), p. 41–48, Association for Computing Machinery, 2014.

[27] J. Ortigoza, F. López-Pires, and B. Barán, "Dynamic environments for virtual machine placement considering elasticity and overbooking," *arXiv preprint arXiv:1601.01881*, 2016.

[28] A. Amarilla, "Scalarization methods for many-objective virtual machine placement of elastic infrastructures in overbooked cloud computing data centers under uncertainty," *arXiv preprint arXiv:1802.04245*, 2018.

[29] F. López-Pires, B. Barán, L. Benítez, S. Zalimben, and A. Amarilla, "Virtual machine placement for elastic infrastructures in overbooked cloud computing datacenters under uncertainty," *Future Generation Computer Systems*, vol. 79, pp. 830 – 848, 2018.

[30] J.-p. Luo, X. Li, and M.-r. Chen, "Hybrid shuffled frog leaping algorithm for energy-efficient dynamic consolidation of virtual machines in cloud data centers," *Expert Systems with Applications*, vol. 41, no. 13, pp. 5804–5816, 2014.

[31] X. Li, P. Garraghan, X. Jiang, Z. Wu, and J. Xu, "Holistic virtual machine scheduling in cloud datacenters towards minimizing total energy," *IEEE Transactions on Parallel and Distributed Systems*, vol. 29, pp. 1317–1331, June 2018.

[32] N. Quang-Hung, N. T. Son, and N. Thoai, "Energy-saving virtual machine scheduling in cloud computing with fixed interval constraints," in *Transactions on Large-Scale Data- and Knowledge-Centered Systems* (A. Hameurlain, J. Küng, R. Wagner, T. K. Dang, and N. Thoai, eds.), (Berlin, Heidelberg), pp. 124–145, Springer Berlin Heidelberg, 2017.

[33] Y.-h. Zhu, D. Chen, and Y. Zhuang, "Virtual machine scheduling algorithm based on energy-aware in cloud data center," *Computer and Modernization*, vol. 4, pp. 17–25, 2016.

[34] Y.-C. Shim, "Performance evaluation of static vm consolidation algorithms for cloud-based data centers considering inter-vm performance interference," *International Journal of Applied Engineering Research*, vol. 11, no. 24, pp. 11794–11802, 2016.

[35] M. Masdari, S. S. Nabavi, and V. Ahmadi, "An overview of virtual machine placement schemes in cloud computing," *Journal of Network and Computer Applications*, vol. 66, pp. 106–127, 2016.

[36] S. Ismaeel and A. Miri, "Multivariate time series elm for cloud data centre workload prediction," in *Human-Computer Interaction. Theory, Design, Development and Practice* (M. Kurosu, ed.), (Cham), pp. 565–576, Springer International Publishing, 2016.

[37] Y. Shoaib and O. Das, "Performance-oriented cloud provisioning: Taxonomy and survey," *arXiv preprint arXiv:1411.5077*, 2014.

[38] K. L. LaCurts, *Application workload prediction and placement in cloud computing systems*. PhD thesis, Massachusetts Institute of Technology, 2014.

[39] S. Ismaeel and A. Miri, "Using ELM techniques to predict data centre VM requests," in *IEEE 2nd International Conference on Cyber Security and Cloud Computing*, pp. 80–86, November 2015.

[40] C. Vazquez, R. Krishnan, and E. John, "Time series forecasting of cloud data center workloads for dynamic resource provisioning," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, vol. 6, no. 3, pp. 87–110, 2015.

[41] M. Dabbagh, B. Hamdaoui, M. Guizani, and A. Rayes, "Energy-efficient resource allocation and provisioning framework for cloud data centers," *IEEE Transactions on Network and Service Management*, vol. 12, pp. 377–391, Sept 2015.

[42] T. Marian, H. Weatherspoon, K.-S. Lee, and A. Sagar, "Fmeter: Extracting indexable low-level system signatures by counting kernel function calls," in *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*, pp. 81–100, Springer, 2012.

[43] X. Jin and J. Han, "Partitional clustering," *Encyclopedia of Machine Learning*, pp. 766–766, 2010.

[44] S. Ismaeel, A. Miri, and A. Al-Khazraji, "Energy-consumption clustering in cloud data centre," in *IEEE 3rd MEC International Conference on Big Data and Smart City (ICBDSC)*, pp. 1–6, March 2016.

[45] M. R. Chowdhury, M. R. Mahmud, and R. M. Rahman, "Implementation and performance analysis of various vm placement strategies in cloudsim," *Journal of Cloud Computing*, vol. 4, no. 1, pp. 1–21, 2015.

[46] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: format+ schema," *Google Inc., Mountain View, CA, USA, Technical Report*, 2011.

[47] A. Khan, X. Yan, S. Tao, and N. Anerousis, "Workload characterization and prediction in the cloud: A multiple time series approach," in *2012 IEEE Network Operations and Management Symposium*, pp. 1287–1294, April 2012.

[48] D. Xu, S. Yang, and H. Luo, "A fusion model for CPU load prediction in cloud computing," *Journal of Networks*, vol. 8, no. 11, pp. 2506–2511, 2013.

[49] C. Canali and R. Lancellotti, "Improving scalability of cloud monitoring through pca-based clustering of virtual machines," *Journal of Computer Science and Technology*, vol. 29, no. 1, pp. 38–52, 2014.

[50] C. Canali and R. Lancellotti, "Exploiting ensemble techniques for automatic virtual machine clustering in cloud systems," *Automated Software Engineering*, vol. 21, no. 3, pp. 319–344, 2014.

[51] I. Moreno, P. Garraghan, P. Townend, and J. Xu, "Analysis, modeling and simulation of workload patterns in a large-scale utility cloud," *IEEE Transactions on Cloud Computing*, vol. 2, pp. 208–221, April 2014.

[52] S. Arora and I. Chana, "A survey of clustering techniques for big data analysis," in *2014 5th International Conference - Confluence The Next Generation Information Technology Summit (Confluence)*, pp. 59–65, Sep. 2014.

[53] G. Semeraro, G. Magklis, R. Balasubramonian, D. H. Albonesi, S. Dwarkadas, and M. L. Scott, "Energy-efficient processor design using multiple clock domains with dynamic voltage and frequency scaling," in *Proceedings Eighth International Symposium on High Performance Computer Architecture*, pp. 29–40, Feb 2002.

[54] M. Zaman, A. Ahmadi, and Y. Makris, "Workload characterization and prediction: A pathway to reliable multi-core systems," in *2015 IEEE 21st International On-Line Testing Symposium (IOLTS)*, pp. 116–121, July 2015.

[55] R. Sarikaya, C. Isci, and A. Buyuktosunoglu, "Runtime workload behavior prediction using statistical metric modeling with application to dynamic power management," in *IEEE International Symposium on Workload Characterization (IISWC'10)*, pp. 1–10, Dec 2010.

[56] T. Liang, S. Wang, and I. Wu, "Using frequent workload patterns in resource selection for grid jobs," in *IEEE Asia-Pacific Services Computing Conference*, pp. 807–812, Dec 2008.

[57] R. Karim, C. Ding, and A. Miri, "End-to-end performance prediction for selecting cloud services solutions," in *IEEE Symposium on Service-Oriented System Engineering (SOSE)*, pp. 69–77, March 2015.

[58] R. Calheiros, E. Masoumi, R. Ranjan, and R. Buyya, "Workload prediction using ARIMA model and its impact on cloud applications' QoS," *IEEE Transactions on Cloud Computing*, vol. PP, no. 99, pp. 1–11, 2014.

[59] W. Fang, Z. Lu, J. Wu, and Z. Cao, "RPPS: A novel resource prediction and provisioning scheme in cloud data center," in *IEEE Ninth International Conference on Services Computing*, pp. 609–616, June 2012.

[60] Shengming Li, Ying Wang, Xuesong Qiu, Deyuan Wang, and Lijun Wang, "A workload prediction-based multi-vm provisioning mechanism in cloud computing," in *IEEE 15th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pp. 1–6, Sep. 2013.

[61] J. Cui, S.-f. Liu, B. Zeng, and N.-m. Xie, "A novel grey forecasting model and its optimization," *Applied Mathematical Modelling*, vol. 37, no. 6, pp. 4399–4406, 2013.

[62] Jhu-Jyun Jheng, Fan-Hsun Tseng, Han-Chieh Chao, and Li-Der Chou, "A novel VM workload prediction using grey forecasting model in cloud data center," in *IEEE International Conference on Information Networking 2014 (ICOIN2014)*, pp. 40–45, Feb 2014.

[63] M. R. Lotfalipour, M. A. Falahi, and M. Bastam, "Prediction of CO2 emissions in iran using grey and ARIMA models," *International Journal of Energy Economics and Policy*, vol. 3, no. 3, pp. 229–237, 2013.

[64] J. Benesty, J. Chen, Y. A. Huang, and S. Doclo, *Study of the Wiener filter for noise reduction*, pp. 9–41. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005.

[65] M. Dabbagh, B. Hamdaoui, M. Guizani, and A. Rayes, "Efficient datacenter resource utilization through cloud resource overcommitment," *Memory*, vol. 40, no. 50, pp. 1–6, 2015.

[66] R. G. Brown, P. Y. Hwang, *et al.*, *Introduction to random signals and applied Kalman filtering*, vol. 3. Wiley New York, 1992.

[67] S. Ajila and A. Bankole, "Cloud client prediction models using machine learning techniques," in *Proceedings of the 2013 IEEE 37th Annual Computer Software and Applications Conference (COMPSAC)*, pp. 134–142, July 2013.

[68] J. Cao, J. Fu, M. Li, and J. Chen, "CPU load prediction for cloud environment based on a dynamic ensemble model," *Software: Practice and Experience*, vol. 44, no. 7, pp. 793–804, 2014.

[69] G. Kousiouris, A. Menychtas, D. Kyriazis, S. Gogouvitis, and T. Varvarigou, "Dynamic, behavioral-based estimation of resource provisioning based on high-level application terms in cloud platforms," *Future Generation Computer Systems*, vol. 32, pp. 27–40, 2014.

[70] Z. Chen, Y. Zhu, Y. Di, and S. Feng, "Self-adaptive prediction of cloud resource demands using ensemble model and subtractive-fuzzy clustering based fuzzy neural network," *Computational intelligence and neuroscience*, vol. 2015, pp. 1–14, 2015.

[71] S. Sahi and V. Dhaka, "Study on predicting for workload of cloud services using Artificial Neural Network," in *Proceedings of the 2015 2nd International Conference on*

Computing for Sustainable Global Development (INDIACom), pp. 331–335, March 2015.

[72] J. J. Prevost, K. Nagothu, B. Kelley, and M. Jamshidi, "Prediction of cloud data center networks loads using stochastic and neural models," in *Proceedings of the 2011 6th International Conference on System of Systems Engineering (SoSE)*, pp. 276–281, June 2011.

[73] Y.-C. Chang, R.-S. Chang, and F.-W. Chuang, "A predictive method for workload forecasting in the cloud environment," in *Advanced Technologies, Embedded and Multimedia for Human-centric Computing* (Y.-M. Huang, H.-C. Chao, D.-J. Deng, and J. J. J. H. Park, eds.), (Dordrecht), pp. 577–585, Springer Netherlands, 2014.

[74] S. Islam, J. Keung, K. Lee, and A. Liu, "Empirical prediction models for adaptive resource provisioning in the cloud," *Future Generation Computer Systems*, vol. 28, no. 1, pp. 155–162, 2012.

[75] K. B. Bey, F. Benhammadi, A. Mokhtari, and Z. Guessoum, "CPU load prediction model for distributed computing," in *Proceedings of The 8th IEEE International Symposium on Parallel and Distributed Computing. (ISPDC'09)*, pp. 39–45, 2009.

[76] K. Cetinski and M. B. Juric, "Ame-wpc: Advanced model for efficient workload prediction in the cloud," *Journal of Network and Computer Applications*, vol. 55, pp. 191–201, 2015.

[77] S. Ismaeel and K. Al-Jebory, "Adaptive fuzzy system modeling," *Eng. Technology*, vol. 20, no. 4, pp. 201–212, 2001.

[78] K. Aljebory, S. Ismaeel, and A. Alqaissi, "Implementation of an intelligent SINS navigator based on ANFIS," in *Systems, Signals and Devices, 2009. SSD '09. 6th International Multi-Conference on*, pp. 1–7, March 2009.

[79] S. Mazumdar, A. Scionti, and A. S. Kumar, "Adaptive resource allocation for load balancing in cloud," in *Cloud Computing*, pp. 301–327, Springer, 2017.

[80] S. Di, D. Kondo, and W. Cirne, "Host load prediction in a google compute cloud with a bayesian model," in *IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pp. 1–11, Nov 2012.

[81] R. Karim, S. Ismaeel, and A. Miri, "Energy-efficient resource allocation for cloud data centres using a multi-way data analysis technique," in *Human-Computer Interaction. Theory, Design, Development and Practice* (M. Kurosu, ed.), (Cham), pp. 577–585, Springer International Publishing, 2016.

[82] Z. A. Mann and M. Szabó, "Which is the best algorithm for virtual machine placement optimization?," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 10, 2017.

[83] F. Rossigneux, L. Lefevre, J.-P. Gelas, D. Assuncao, and M. Dias, "A generic and extensible framework for monitoring energy consumption of openstack clouds," in *Proceedings of The 4th IEEE International Conference on Big Data and Cloud Computing (BdCloud)*, pp. 696–702, Dec 2014.

[84] A. Hameed, A. Khoshkbarforoushha, R. Ranjan, P. P. Jayaraman, J. Kolodziej, P. Balaji, S. Zeadally, Q. M. Malluhi, N. Tziritas, A. Vishnu, *et al.*, "A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems," *Computing*, vol. 98, no. 7, pp. 751–774, 2016.

[85] B. Jennings and R. Stadler, "Resource management in clouds: Survey and research challenges," *Journal of Network and Systems Management*, vol. 23, no. 3, pp. 567–619, 2015.

[86] K. Fatema, V. C. Emeakaroha, P. D. Healy, J. P. Morrison, and T. Lynn, "A survey of cloud monitoring tools: Taxonomy, capabilities and objectives," *Journal of Parallel and Distributed Computing*, vol. 74, no. 10, pp. 2918–2933, 2014.

[87] A. Hameed, A. Khoshkbarforoushha, R. Ranjan, P. P. Jayaraman, J. Kolodziej, P. Balaji, S. Zeadally, Q. M. Malluhi, N. Tziritas, A. Vishnu, *et al.*, "A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems," *Computing*, vol. 98, no. 7, pp. 751–774, 2016.

[88] G. Aceto, A. Botta, W. De Donato, and A. Pescapè, "Cloud monitoring: A survey," *Computer Networks*, vol. 57, no. 9, pp. 2093–2115, 2013.

[89] S. Ismaeel, A. Miri, D. Chourishi, and S. M. R. Dibaj, "Open source cloud management platforms: A review," in *IEEE 2nd International Conference on Cyber Security and Cloud Computing*, pp. 470–475, November 2015.

[90] J. Gutierrez-Aguado, J. M. Alcaraz Calero, and W. Diaz Villanueva, "Iaasmon: Monitoring architecture for public cloud computing data centers," *Journal of Grid Computing*, vol. 14, no. 2, pp. 283–297, 2016.

[91] G. Da Cunha Rodrigues, R. N. Calheiros, V. T. Guimaraes, G. L. d. Santos, M. B. de Carvalho, L. Z. Granville, L. M. R. Tarouco, and R. Buyya, "Monitoring of cloud computing environments: Concepts, solutions, trends, and future directions," in *Proceedings of the 31st Annual ACM Symposium on Applied Computing*, SAC '16, pp. 378–383, ACM, 2016.

[92] U. Ismail, "Comparing 7 options for docker container monitoring," January 2020. Online; accessed 03 March 2020.

[93] A. Beloglazov and R. Buyya, "Openstack neat: a framework for dynamic and energy-efficient consolidation of virtual machines in openstack clouds," *Concurrency and Computation: Practice and Experience*, vol. 27, no. 5, pp. 1310–1333, 2015.

[94] S. Masoumzadeh and H. Hlavacs, "Dynamic virtual machine consolidation: A multi agent learning approach," in *IEEE International Conference on Autonomic Computing (ICAC)*, pp. 161–162, July 2015.

[95] R. Kashyap, S. Chaudhary, and P. Jat, "Virtual machine migration for back-end mashup application deployed on openstack environment," in *International Conference on Parallel, Distributed and Grid Computing (PDGC)*, pp. 214–218, Dec 2014.

[96] A. Horri, M. S. Mozafari, and G. Dastghaibyfard, "Novel resource allocation algorithms to performance and energy efficiency in cloud computing," *The Journal of Supercomputing*, vol. 69, no. 3, pp. 1445–1461, 2014.

[97] 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.

[98] M. Monil and R. Rahman, "Implementation of modified overload detection technique with vm selection strategies based on heuristics and migration control," in *IEEE/ACIS 14th International Conference on Computer and Information Science (ICIS)*, pp. 223–227, June 2015.

[99] D. Minarolli, A. Mazrekaj, and B. Freisleben, "Tackling uncertainty in long-term predictions for host overload and underload detection in cloud computing," *Journal of Cloud Computing*, vol. 6, p. 4, Feb 2017.

[100] A. Beloglazov and R. Buyya, "Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, pp. 1366–1379, July 2013.

[101] F. Farahnakian, P. Liljeberg, and J. Plosila, "LiRCUP: Linear regression based CPU usage prediction algorithm for live migration of virtual machines in data centers," in *IEEE 39th Euromicro Conference on Software Engineering and Advanced Applications*, pp. 357–364, Sep. 2013.

[102] F. Farahnakian, T. Pahikkala, P. Liljeberg, and J. Plosila, "Energy aware consolidation algorithm based on k-nearest neighbor regression for cloud data centers," in *IEEE/ACM 6th International Conference on Utility and Cloud Computing (UCC)*, pp. 256–259, 2013.

[103] S. S. Masoumzadeh and H. Hlavacs, "An intelligent and adaptive threshold-based schema for energy and performance efficient dynamic vm consolidation," in *Energy Efficiency in Large Scale Distributed Systems* (J.-M. Pierson, G. Da Costa, and L. Dittmann, eds.), (Berlin, Heidelberg), pp. 85–97, Springer Berlin Heidelberg, 2013.

[104] T. L. Saaty, *The Analytic Hierarchy and Analytic Network Processes for the Measurement of Intangible Criteria and for Decision-Making*, pp. 363–419. New York, NY: Springer New York, 2016.

[105] F. Farahnakian, A. Ashraf, T. Pahikkala, P. Liljeberg, J. Plosila, I. Porres, and H. Tenhunen, "Using ant colony system to consolidate vms for green cloud computing," *IEEE Transactions on Services Computing*, vol. 8, pp. 187–198, March 2015.

[106] M. A. H. Monil and R. M. Rahman, "Vm consolidation approach based on heuristics, fuzzy logic, and migration control," *Journal of Cloud Computing*, vol. 5, p. 8, Jul 2016.

[107] G. F. Shidik, K. Mustofa, *et al.*, "Evaluation of selection policy with various virtual machine instances in dynamic vm consolidation for energy efficient at cloud data centers," *Journal of Networks*, vol. 10, no. 7, pp. 397–406, 2015.

[108] A. Song, W. Fan, W. Wang, J. Luo, and Y. Mo, "Multi-objective virtual machine selection for migrating in virtualized data centers," in *Pervasive Computing and the Networked World* (Q. Zu, B. Hu, and A. Elçi, eds.), (Berlin, Heidelberg), pp. 426–438, Springer Berlin Heidelberg, 2013.

[109] S. S. Masoumzadeh and H. Hlavacs, "Integrating vm selection criteria in distributed dynamic VM consolidation using fuzzy Q-learning," in *IEEE 9th International Conference on Network and Service Management (CNSM)*, pp. 332–338, Oct 2013.

[110] M. A. H. Monil and R. M. Rahman, "Fuzzy logic based energy aware VM consolidation," in *Internet and Distributed Computing Systems* (G. Di Fatta, G. Fortino, W. Li, M. Pathan, F. Stahl, and A. Guerrieri, eds.), (Cham), pp. 31–38, Springer International Publishing, 2015.

[111] S. Ismaeel, A. Al-Khazraji, and K. Al-delimi, "Fuzzy information modeling in a database system," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 6, no. 1, pp. 1–7, 2017.

[112] S. Chaisiri, B.-S. Lee, and D. Niyato, "Optimal virtual machine placement across multiple cloud providers," in *IEEE Asia-Pacific Services Computing Conference APSCC*, pp. 103–110, 2009.

[113] F.-H. Tseng, C.-Y. Chen, L.-D. Chou, H.-C. Chao, and J.-W. Niu, "Service-oriented virtual machine placement optimization for green data center," *Mobile Networks and Applications*, vol. 20, no. 5, pp. 556–566, 2015.

[114] C. Ghribi, *Energy efficient resource allocation in cloud computing environments*. PhD thesis, Institut National des Télécommunications, 2014.

[115] D. Jiankang, W. Hongbo, and C. Shiduan, "Energy-performance tradeoffs in iaas cloud with virtual machine scheduling," *IEEE China Communications*, vol. 12, no. 2, pp. 155–166, 2015.

[116] M. Dabbagh, B. Hamdaoui, M. Guizani, and A. Rayes, "Online assignment and placement of cloud task requests with heterogeneous requirements," in *IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2015.

[117] B. C. Ribas, R. M. Suguimoto, R. A. Montano, F. Silva, and M. Castilho, "Pbfvmc: A new pseudo-boolean formulation to virtual-machine consolidation," in *IEEE Brazilian Conference on Intelligent Systems*, pp. 201–206, 2013.

[118] X. Liu, H. Gu, H. Zhang, F. Liu, Y. Chen, and X. Yu, "Energy-aware on-chip virtual machine placement for cloud-supported cyber-physical systems," *Microprocessors and Microsystems*, vol. 52, pp. 427–437, 2017.

[119] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[120] W. Yue and Q. Chen, "Dynamic placement of virtual machines with both deterministic and stochastic demands for green cloud computing," *Mathematical Problems in Engineering*, vol. 2014, 2014.

[121] F. Farahnakian, T. Pahikkala, P. Liljeberg, J. Plosila, and H. Tenhunen, "Multi-agent based architecture for dynamic VM consolidation in cloud data centers," in *40th EUROMICRO Conference on Software Engineering and Advanced Applications*, pp. 111–118, Aug 2014.

[122] F. Farahnakian, T. Pahikkala, P. Liljeberg, J. Plosila, and H. Tenhunen, "Utilization prediction aware VM consolidation approach for green cloud computing," in *IEEE 8th International Conference on Cloud Computing*, pp. 381–388, June 2015.

[123] H. Goudarzi and M. Pedram, "Energy-efficient virtual machine replication and placement in a cloud computing system," in *IEEE 5th International Conference on Cloud Computing (CLOUD)*, pp. 750–757, 2012.

[124] A. Dalvandi, M. Gurusamy, and K. C. Chua, "Time-aware VM-placement and routing with bandwidth guarantees in green cloud data centers," in *IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom)*, vol. 1, pp. 212–217, 2013.

[125] A. C. Adamuthe, R. M. Pandharpatte, and G. T. Thampi, "Multiobjective virtual machine placement in cloud environment," in *IEEE International Conference on Cloud and Ubiquitous Computing and Emerging Technologies (CUBE)*, pp. 8–13, 2013.

[126] E. Feller, L. Rilling, and C. Morin, "Energy-aware ant colony based workload placement in clouds," in *12th IEEE/ACM International Conference on Grid Computing (GRID)*, pp. 26–33, Sept 2011.

[127] F. López-Pires and B. Barán, "Multi-objective virtual machine placement with service level agreement: A memetic algorithm approach," in *IEEE/ACM 6th International Conference on Utility and Cloud Computing (UCC)*, pp. 203–210, Dec 2013.

[128] F. López-Pires and B. Barán, "Virtual machine placement literature review," *arXiv preprint arXiv:1506.01509*, 2015.

[129] W. Song, Z. Xiao, Q. Chen, and H. Luo, "Adaptive resource provisioning for the cloud using online bin packing," *IEEE Transactions on Computers*, vol. 63, no. 11, pp. 2647–2660, 2014.

[130] A. Alahmadi, A. Alnowiser, M. M. Zhu, D. Che, and P. Ghodous, "Enhanced first-fit decreasing algorithm for energy-aware job scheduling in cloud," in *IEEE International Conference on Computational Science and Computational Intelligence (CSCI)*, vol. 2, pp. 69–74, 2014.

[131] U. Ismail, "IBM ILOG CPLEX optimization studio," 2020. Online; accessed 03 March 2020.

[132] Y. Gao, H. Guan, Z. Qi, Y. Hou, and L. Liu, "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing," *Journal of Computer and System Sciences*, vol. 79, no. 8, pp. 1230–1242, 2013.

[133] S. K. Bose and S. Sundarrajan, "Optimizing migration of virtual machines across data-centers," in *International Conference on Parallel Processing Workshops (ICPPW'09)*, pp. 306–313, Sept 2009.

[134] A. Mohan and S. Shine, "Survey on live vm migration techniques," *International Journal of Advanced Research in Computer Engineering and Technology*, vol. 2, no. 1, pp. 155–157, 2013.

[135] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*, pp. 273–286, 2005.

[136] B. Hu, Z. Lei, Y. Lei, D. Xu, and J. Li, "A time-series based precopy approach for live migration of virtual machines," in *IEEE 17th International Conference on Parallel and Distributed Systems (ICPADS)*, pp. 947–952, 2011.

[137] S. Sahni and V. Varma, "A hybrid approach to live migration of virtual machines," in *IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, pp. 1–5, Oct 2012.

[138] A. Strunk, "A lightweight model for estimating energy cost of live migration of virtual machines," in *IEEE Sixth International Conference on Cloud Computing (CLOUD)*, pp. 510–517, June 2013.

[139] X. Guan, B. Y. Choi, and S. Song, "Topology and migration-aware energy efficient virtual network embedding for green data centers," in *IEEE 23rd International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–8, Aug 2014.

[140] J. Huang, K. Wu, and M. Moh, "Dynamic virtual machine migration algorithms using enhanced energy consumption model for green cloud data centers," in *IEEE International Conference on High Performance Computing Simulation (HPCS)*, pp. 902–910, July 2014.

[141] C. Ghribi, M. Hadji, and D. Zeghlache, "Energy efficient VM scheduling for cloud data centers: Exact allocation and migration algorithms," in *13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pp. 671–678, May 2013.

[142] S. Akiyama, T. Hirofuchi, and S. Honiden, "Evaluating impact of live migration on data center energy saving," in *IEEE 6th International Conference on Cloud Computing Technology and Science (CloudCom)*, pp. 759–762, Dec 2014.

[143] T. Yang, Y. C. Lee, and A. Y. Zomaya, "Energy-efficient data center networks planning with virtual machine placement and traffic configuration," in *Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on*, pp. 284–291, Dec 2014.

[144] Y. Liao, D. Yin, and L. Gao, "Dpillar: Scalable dual-port server interconnection for data center networks," in *IEEE 19th International Conference on Computer Communications and Networks (ICCCN)*, pp. 1–6, 2010.

[145] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 63–74, 2008.

[146] G. Qu, Z. Fang, J. Zhang, and S.-Q. Zheng, "Switch-centric data center network structures based on hypergraphs and combinatorial block designs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 1154–1164, 2015.

[147] A. Erickson, I. A. Stewart, A. Kiasari, and J. Navaridas, "An optimal single-path routing algorithm in the datacenter network dpillar," *arXiv preprint arXiv:1509.01746*, 2015.

[148] T. Wang, Z. Su, Y. Xia, and M. Hamdi, "Rethinking the data center networking: Architecture, network protocols, and resource sharing," *IEEE Access*, vol. 2, pp. 1481–1496, 2014.

[149] A. Erickson, A. E. Kiasari, J. Navaridas, and I. A. Stewart, "An efficient shortest-path routing algorithm in the data centre network dpillar," in *Combinatorial Optimization and Applications* (Z. Lu, D. Kim, W. Wu, W. Li, and D.-Z. Du, eds.), (Cham), pp. 209–220, Springer International Publishing, 2015.

[150] T. H. Duong-Ba, T. Nguyen, B. Bose, and T. T. Tran, "A dynamic virtual machine placement and migration scheme for data centers," *IEEE Transactions on Services Computing*, 2018.

[151] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Towards understanding heterogeneous clouds at scale: Google trace analysis," *Intel Science and Technology Center for Cloud Computing, Tech. Rep*, vol. 84, pp. 1–21, 2012.

[152] S. Ismaeel and A. Miri, "Real-time energy-conserving vm-provisioning framework for cloud-data centers," in *IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 0765–0771, Jan 2019.

[153] Y. Yu, V. Jindal, I.-L. Yen, and F. Bastani, "Integrating clustering and learning for improved workload prediction in the cloud," in *IEEE 9th International Conference on Cloud Computing (CLOUD)*, pp. 876–879, 2016.

[154] J. V. Guttag, *Introduction to computation and programming using Python*. MIT Press, 2013.

[155] A. Fahad, N. Alshatri, Z. Tari, A. Alamri, I. Khalil, A. Y. Zomaya, S. Foufou, and A. Bouras, "A survey of clustering algorithms for big data: Taxonomy and empirical analysis," *IEEE transactions on emerging topics in computing*, vol. 2, no. 3, pp. 267–279, 2014.

[156] M. C. Thrun, *Projection-Based Clustering Through Self-Organization and Swarm Intelligence: Combining Cluster Analysis with the Visualization of High-Dimensional Data*. Springer, 2018.

[157] C. Hennig, "Cluster validation by measurement of clustering characteristics relevant to the user," *arXiv preprint arXiv:1703.09282*, 2017.

[158] J. Hämäläinen, S. Jauhiainen, and T. Kärkkäinen, "Comparison of internal clustering validation indices for prototype-based clustering," *Algorithms*, vol. 10, no. 3, p. 105, 2017.

[159] R. Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Transactions on neural networks*, vol. 16, no. 3, pp. 645–678, 2005.

[160] B. Desgraupes, "Clustering indices," *University of Paris Ouest-Lab Modal'X*, pp. 1–34, 2013.

[161] S. Salvador and P. Chan, "Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms," in *IEEE 16th International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 576–584, 2004.

[162] A. Amini, T. Y. Wah, and H. Saboohi, "On density-based data streams clustering algorithms: A survey," *Journal of Computer Science and Technology*, vol. 29, no. 1, pp. 116–141, 2014.

[163] R. Xu, J. Xu, and D. C. Wunsch, "A comparison study of validity indices on swarm-intelligence-based clustering," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 4, pp. 1243–1256, 2012.

[164] S. Saitta, B. Raphael, and I. F. Smith, "A comprehensive validity index for clustering," *Intelligent Data Analysis*, vol. 12, no. 6, pp. 529–548, 2008.

[165] N. Williams, S. Nasuto, and J. Saddy, "Method for exploratory cluster analysis and visualisation of single-trial erp ensembles," *Journal of neuroscience methods*, vol. 250, pp. 22–33, 2015.

[166] S. Ismaeel and A. Miri, "A systematic cloud workload clustering technique in large scale data centers," in *IEEE World Congress on Services (SERVICES)*, vol. 2642-939X, pp. 362–363, July 2019.

[167] Q. Xia, Y. Lan, L. Zhao, and L. Xiao, "Energy-saving analysis of cloud workload based on K-means clustering," in *The IEEE Computing, Communications and IT Applications Conference (ComComAp)*, pp. 305–309, Oct 2014.

[168] M. Rasheduzzaman, M. Islam, T. Islam, T. Hossain, and R. Rahman, "Task shape classification and workload characterization of google cluster trace," in *Proceedings of the 2014 IEEE International Advance Computing Conference (IACC)*, pp. 893–898, Feb 2014.

[169] Z. I. Botev, J. F. Grotowski, D. P. Kroese, *et al.*, "Kernel density estimation via diffusion," *The annals of Statistics*, vol. 38, no. 5, pp. 2916–2957, 2010.

[170] W. M. Campbell, D. E. Sturim, and D. A. Reynolds, "Support vector machines using gmm supervectors for speaker verification," *IEEE Signal Process. Lett.*, vol. 13, no. 5, pp. 308–311, 2006.

[171] M. Dabbagh, B. Hamdaoui, M. Guizani, and A. Rayes, "Toward energy-efficient cloud computing: Prediction, consolidation, and overcommitment," *IEEE Network*, vol. 29, no. 2, pp. 56–61, 2015.

[172] A. Khoshkbarforoushha, R. Ranjan, R. Gaire, P. P. Jayaraman, J. Hosking, and E. Abbasnejad, "Resource usage estimation of data stream processing workloads in datacenter clouds," *CoRR*, vol. abs/1501.07020, 2015.

[173] I. Sarji, C. Ghali, A. Chehab, and A. Kayssi, "Cloudese: Energy efficiency model for cloud computing environments," in *Proceedings of The International Conference on Energy Aware Computing (ICEAC)*, pp. 1–6, Nov 2011.

[174] N. ming Xie and S. feng Liu, "Discrete grey forecasting model and its optimization," *Applied Mathematical Modelling*, vol. 33, no. 2, pp. 1173 – 1186, 2009.

[175] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1-3, pp. 489–501, 2006.

[176] M. Dabbagh, B. Hamdaoui, M. Guizani, and A. Rayes, "Energy-efficient cloud resource management," in *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 386–391, 2014.

[177] A. Grosfeld-Nir, B. Ronen, and N. Kozlovsky, "The pareto managerial principle: when does it apply?," *International Journal of Production Research*, vol. 45, no. 10, pp. 2317–2325, 2007.

[178] F. López-Pires and B. Barán, "Many-objective virtual machine placement," *Journal of Grid Computing*, vol. 15, no. 2, pp. 161–176, 2017.

[179] D. Economou, S. Rivoire, C. Kozyrakis, and P. Ranganathan, "Full-system power analysis and modeling for server environments," in *Workshop MOBS*, pp. 70–77, 2006.

[180] V. Shrivastava, P. Zerfos, K. Lee, H. Jamjoom, Y. Liu, and S. Banerjee, "Application-aware virtual machine migration in data centers," in *Proceedings IEEE INFOCOM*, pp. 66–70, April 2011.

[181] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *42nd Annual IEEE/ACM International Symposium on Microarchitecture MICRO 42*, pp. 469–480, 2009.