An Optimum Compute Resources Consolidation Framework for Cloud Data Center

A Thesis

submitted in fulfillment of the requirement for the award of the degree of
DOCTOR of PHILOSOPHY

by

Sheetal Garg
(901903011)

under the guidance of

Dr. Rohit Ahuja
Assistant Professor, CSED
Thapar Institute of Engineering and Technology, Patiala -147004

Dr. Raman Singh
Lecturer
University of the West of Scotland, United Kingdom, G72 0LH

Dr. Ivan Perl
Associate Professor
ITMO University, Saint Petersburg, Russia, 197101

Computer Science and Engineering Department
Thapar Institute of Engineering and Technology,
Patiala - 147004, INDIA

October 2024

Candidate Declaration

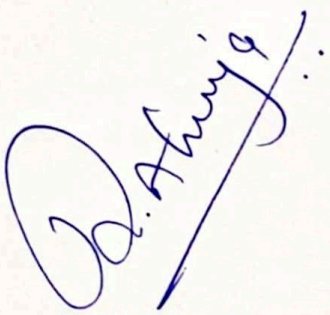
I hereby certify that the work, which is being presented in the thesis, entitled **An Optimum Compute Resources Consolidation Framework for Cloud Data Center**, in partial fulfillment of the requirements for the award of the degree of **Doctor of Philosophy** and submitted to the institution is an authentic record of my own work carried out during the period **August 2019** to **October 2024** under the supervision of **Dr. Rohit Ahuja, Dr. Raman Singh** and **Dr. Ivan Perl**. I have also cited the reference about the text(s)/figure(s)/table(s) from where they have been taken.

The matter presented in this thesis has not been submitted elsewhere for the award of any other degree or diploma from any institution.

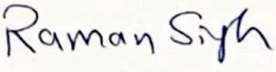
(**Sheetal Garg**)
Regn. No. 901903011

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

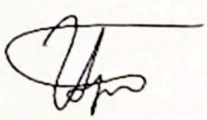
(**Dr. Rohit Ahuja**)
Assistant Professor
Department of Computer Science & Engineering
Thapar Institute of Engineering and Technology, Patiala, 147004
Punjab, INDIA



(**Dr. Raman Singh**)
Lecturer
School of Computing, Engineering and Physical Sciences
University of the West of Scotland, United Kingdom, G72 0LH



(**Dr. Ivan Perl**)
Associate Professor
Faculty of Software Engineering and Computer Systems
ITMO University, Saint Petersburg, Russia, 197101



.....dedicated to all young generation researchers

Acknowledgements

Before discussing my journey of Ph.D., I would like to thank the almighty God who gave me strength and courage to overcome all the obstacles and complete this endeavour. The aim of my life, to be called by the salutation of a 'Doctor', seems to become a reality when I got admission in Doctorate of Philosophy in Thapar Institute of Engineering & Technology. Research initiated with this startup in my life. Without acknowledging the people who supported me throughout this journey, this task would be incomplete. I know words are never enough to express the gratitude; I am just delivering the phrase for the acceptance of regards.

First, I would like to express my deep gratitude to my supervisor, Dr. Rohit Ahuja, for his insightful guidance and support. He was always there to steer me in the proper path whenever I became stuck with my thoughts. I am sincerely grateful to my supervisor, Dr. Raman Singh, for his unwavering support and invaluable guidance throughout this journey. Despite being geographically distant, his continual availability and insightful comments have been instrumental in shaping this work. I am also indebted to Dr. Ivan Perl for supervising me throughout my Ph.D journey. I am also grateful to the head of the department, Dr. Shalini Batra, Dr. Sushma Jain, Dr. H.S. Pannu and members of my doctoral committee, Dr. Inderveer Channa, Dr. Anju Bala, and Dr. Amit Mishra for their constructive suggestions and ensuring the correct pace of my work. I am also obliged to the Director, Prof. Prakash Gopalan, Dean (RSP) and the management of Thapar Institute of Engineering and Technology, who provided me with all the necessary resources and facilities to complete my work.

I would also like to extend my heartfelt gratitude to my dear friend, Reaya, who has consistently supported me, offering a compassionate ear and unwavering motivation during my most challenging moments. I also thank Namrata for her constant belief in me and for her much-needed joyful distractions throughout my PhD. Additionally, I am deeply thankful to Paluck for her unconditional assistance and support. I am also grateful to Govind, Krishan, and Niyaz for boosting my energy and providing much-needed support, especially towards the end of my submission.

The chain of gratitude will definitely be incomplete if I forget to thank my mother, Mrs. Manju Garg; my father, Mr. Sanjeev Garg; my mother-in-law, Mrs. Anita Gupta; my father-in-law, Mr. Bijendra Gupta, for their unconditional love, support and encouragement in every phase of my life. I am also grateful to all my family members and cousins. They all made this journey more comfortable with words of encouragement, which helped

me in finishing my work.

On a personal note, I'd like to extend my heartfelt gratitude to my husband, Dr. Mayank Gupta. His steadfast support and wise guidance have been indispensable to me every step of the way. His calming presence has always been a source of strength, helping me navigate through any challenges with ease. Moreover, his passion towards research has been truly inspiring, constantly encouraging me to strive for excellence in my research. Thank you so much for being my rock and my guiding light.

Sheetal Garg

Abstract

Cloud computing offers an efficient alternative for business enterprises as compared to traditional models for computing and data storage needs. The dynamic workload on cloud servers is one of the major reasons for the ineffective utilization of cloud resources. Therefore, a larger number of servers are active in cloud data centers (CDCs) to satisfy the cloud users' demands, which severely enhances energy consumption and heat dissipation. Hence, effective resource management within the CDCs has become crucial for cloud service providers.

To meet users' requests of cloud services, it is essential to optimally allocate requested resources on physical machines (PMs) through virtual machines (VMs). The dynamic nature of workloads complicates initial VM allocation, which often results in the overutilization or underutilization of PMs. This leads to performance degradation, wastage of resources, increased operational costs, higher active servers, and energy consumption. Therefore, to address these challenges, effective resource management strategies are required to ensure the effective utilization of PM resources.

With an intent to achieve effective resource management, this thesis proposed solutions to accurately predict the resource usage of machines and effectively utilize resources to balance the load of PMs, as well as reduce the total energy consumption of a data center. Firstly, in order to effectively utilize the resources of PMs in a data center, a hybridizing approach leveraging Gaussian Mixture Model (GMM) and Long Short-Term Memory (LSTM) model is proposed to predict resource usage of heterogeneous PMs. This research aims to capture the heterogeneity of available PMs in a data center using GMM based on mean CPU usage and memory usage of machines while capturing temporal dependencies and patterns in resource usage data using optimal hyperparameters of the LSTM model that enable more accurate prediction.

Next, to capture both long-range and short-term dependencies, as well as input sequences of data to predict the resource usage of PMs, rigorous experiments are carried out. Attention-based mechanism models, Transformer and Informer, along with LSTM, are employed for this purpose. Moreover, to rationally select heterogeneous machines based on mean CPU, memory, and hard disk usage, the Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) algorithm is utilized.

To effectively utilize the resources of PMs in the data center, it is essential that requested VMs are allocated to PMs in a manner that maximizes resource usage and reduces the overall energy consumption of the data center. To achieve this, we employed Best Fit, First Fit, Best Fit Decrease, First Fit Decrease, and Genetic Algorithm for optimal VM allocation (VMA) to minimize active PMs as an objective function and identify their effect on energy consumption.

Lastly, a three-tier architecture is proposed for the allocation and dynamic consolidation of VMs using current as well as predicted resource usage of machines in a CDC. The first tier carries out a Non-Dominated Sorting Genetic Algorithm (NSGA-II) for VMA by considering their resource constraints. The second tier employs LSTM for resource usage prediction of VMs, which enables the computation of PM current as well as predicted resource usage. The third tier aims to achieve dynamic consolidation of the VMs by detecting overloaded and underloaded PM using threshold-based criteria, VM selection using a proposed novel strategy based on Pareto-front and VM placement using NSGA-II. The objective of this architecture is to provide a holistic approach that reduces the active number of PMs, the number of migrations, and the energy consumption of the CDC with a minimum prediction error rate.

Google Cluster Trace usage dataset (GCT) is utilized to assess and validate the techniques employed in our study. A Sum-Average (SA) algorithm is proposed for data preprocessing to extract PM resource usage from task resource usage. It prepares the dataset and makes it suitable for providing input into clustering and prediction models.

List of Publications

Journal Publications (SCI/SCIE):

1. **Sheetal Garg**, Rohit Ahuja, Raman Singh and Ivan Perl, "GMM-LSTM: a component driven resource utilization prediction model leveraging LSTM and gaussian mixture model", Cluster Computing 26.6 (2023): 3547-3563 [**SCI Indexed, IF 3.6**]

2. **Sheetal Garg**, Rohit Ahuja, Raman Singh and Ivan Perl, "An effective deep learning architecture leveraging BIRCH clustering for resource usage prediction of heterogeneous machines in cloud data center",Cluster Computing (2024): 1-21. [**SCI Indexed, IF 3.6**]

3. **Sheetal Garg**, Rohit Ahuja, Raman Singh and Ivan Perl "A Three-Tier Energy Efficient Architecture Integrating Virtual Machine Allocation and Consolidation leveraging NSGA-II and LSTM for Cloud Data Center", IEEE Transactions on Emerging Topics in Computing [**SCI Indexed, IF - 5.1**] (Under Review)

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ACO	Ant Colony Optimization
ACS-VMC	Ant Colony System-Based VM Consolidation
ANN	Artificial Neural Network
API	Application Programming Interface
ARIMA	Auto Regression Integrated Moving Average model
AWS	Amazon Web Services
BAT	Bandwidth-Aware Divisible Task
BF	Best Fit
BFD	Best Fit Decrease
BIRCH	Balanced Iterative Reducing and Clustering using Hierarchies
BP	Bin Packing
BPTT	BackPropagation Through Time
BRR	Bayesian Ridge Regression
CDC	Cloud Data Center
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CSPs	Cloud Service Providers
DBC	Density-Based Clustering Method
DCMMT	Dynamic Consolidation with Minimization of Migration Thrashing
DVMC	Dynamic Virtual Machine Consolidation
EEMD	Ensemble Empirical Mode Decomposition
ELMs	Extreme Learning Machines
EMA	Exponential Moving Average
esDNN	efficient supervised learning-based Deep Neural Network
ESFCNN	Ensemble model and Subtractive-Fuzzy Clustering based Fuzzy Neural Network

FF	First Fit
FFD	First Fit Decrease
FFO-EVMM	FireFly Optimisation Energy Aware VM Migration
FLNN	Functional Link Neural Network
GA	Genetic Algorithm
GCP	Google Cloud Platform
GCT	Google Cluster Usage Trace
GMM	Gaussian Mixture Model
GPS	Growth Potential Aware VM Selection
GRU	Gated Recurrent Unit
HMM	Hidden Markov Modeling
I/O	Input Output
IaaS	Infrastructure as a Service
IMFs	Intrinsic Mode Functions
IQR	Interquartile Range
KNNR	K-Nearest Neighbor Regression
LP	Local Predictor
LR	Linear Regression
LSTM	Long Short Term Memory model
MA	Moving Average
MAD	Median Absolute Deviation
MAPE	Mean Absolute Percentage Error
MBFD	Modified Best Fit Decreasing
MC	Maximum Correlation
MFFD	Modified First Fit Decreasing
ML	Machine Learning
MMT	Minimal Migration Time
MP	Machine Predictor

MSE	Mean Square Error
MUUHD	Modified Underused Host Detection
NIST	National Institute of Standards and Technology
NP	Nondeterministic Polynomial Time
NSGA-II	Non-dominated Sorting Genetic Algorithm II
OED	Orthogonal Experimental Design
OHD-MUP	Overloaded Server Detection utilising Multi-User Prediction
OS	Operating System
PBC	Prototype-Based Clustering Method
PaaS	Platform as a Service
PBUHD	Prediction-Based Underutilized Host Detection
PHOD	Pearson Host Overload Load Detection Method
PM	Physical Machine
PSO	Particle Swarm Optimization
PABFD	Power-Aware Best Fit Decreasing algorithm
PUE	Power Usage Effectiveness
PWS	Prediction Window Size
QOS	Quality of Service
RA	Random Allocation
RAM	Random-Access Memory
RC	Random Choice
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
RPPS	Resource Prediction and Provisioning scheme
RR	Round Robin
SA	Sum Average
SaaS	Software as a Service
SANs	Storage Area Networks

SARIMA	Seasonal Autoregressive Integrated Moving Average
SLAV	Service Level Agreement Violation
SLB	Statistical-based Load Balance
SMA	Second Moving Average
SSDs	Solid-State Drives
STLF	Short-Term Load Forecasting
SVM	Support Vector Machines
SVMC	Static Virtual Machine Consolidation
SVR	Support Vector Regression
TCN	Temporal Convolutional Network
TPM	Two-phase Pattern Matching
TRU	Task Resource Usage table
TSP	Time Series Prediction
UPS	Uninterruptible Power Supply
UP-VMC	Utilization Prediction-aware VM Consolidation
VM	Virtual Machine
VMA	Virtual Machine Allocation
VMC	Virtual Machine Consolidation
VMD	Variational Mode Decomposition
VMM	Virtual Machine Monitor
VMP	Virtual Machine Placement
VMS	Virtual Machine Selection
VPC	Virtual Private Cloud
VPN	Virtual Private Network
WBDTH	Workload-Based Dynamic Threshold

List of Notations

μ	Mean
σ	Standard Deviation
Z	Standardization Score
H	Hopkins Statistic
$s(o)$	Silhouette Coefficient
a_t	Forget gate in LSTM model
b_t	Input gate in LSTM model
d_t	Output gate in LSTM model
h_t	Hidden State
Q	Query matrix in Transformer model
K	Key matrix in Transformer model
V	Value matrix in Transformer model
$FFN(x)$	Feed-Forward Layer with input x
p	Total number of PMs
v	Total number of VMs
C	Total CPU capacity vector of PMs
D	Requested CPU demand vector of VMs
R	Remaining CPU capacity vector of PMs
Total_usage_j	Total CPU usage of PM j
TEC	Total energy consumption of DC after VMA
TAM	Total number of active machines after VMA
$F1$	Fitness function for VMA
$\text{PM}_i^{\text{CPU}}(t)$	Current total CPU usage of PM i at time t
$\text{PM}_i^{\text{CPU}}(t+1)$	Predicted total CPU usage of PM i at time $t+1$
$\text{VM}_i(t)$	Workload of VM i at time t

$\text{VM}_i(t+1)$	Predicted workload of VM i at time $t+1$
$\text{CPU\_per}_i(t)$	Percentage of CPU usage by PM i at time t
$\text{CPU\_per}_i(t+1)$	Calculated percentage of CPU usage by PM i at time $t+1$
Usage_j	Maximum CPU usage of VM j between current and predicted value
Staying_usage	Calculated remaining CPU usage of PM
T	Threshold for overloaded PM
numberOfzero	Total number of remaining VMs on overloaded PM after migration
$F2$	Fitness function for virtual machine selection
PM_usage_i	Maximum CPU usage of PM i between current and predicted value
Available_i	Available CPU capacity of PM i to place any migrated VM
$F3$	Fitness function for VM placement
$TEC1$	Total energy consumption of DC post-VMC
$TAM1$	Total active number of machines in DC post-VMC

Chapter 1

Introduction

In today's digital era, the exponential growth in computing demands necessitates robust infrastructure, often posing difficulties like financial constraints, scalability, security, and adaptability for businesses and organizations. To address these challenges, an escalating number of entities turn to cloud computing. It is a technology that allows users to access computing resources, store data, retrieve data, and deploy applications over the web on a pay-per-use basis [1]. It provides flexibility, scalability, and cost-effectiveness by offering on-demand accessibility to a shared pool of configurable computing resources. Cloud users can access these services from any location without being constrained by geography, type of device (smartphone, laptop, desktop), or time limitations. Leading companies such as Amazon, Google, IBM, Microsoft, and many others build, operate, and maintain cloud data centers (CDCs) to offer cloud services and are known as cloud service providers (CSPs). These services typically include Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) [2]. A CDC houses servers, networking hardware, and storage systems to fulfil user requirements, while server resources encompass hardware components and processing capabilities such as central processing units (CPUs), random-access memory (RAM), storage drives, and network bandwidth.

The growing demand for cloud services has led to an increase in resource requirements to support the expanding range of user requests and operational needs. It has been analyzed that the CPU is the most crucial and in demand resource that acts as the main computational engine for carrying out various tasks and meeting user requirements [3, 4, 5]. Generally, the CPU utilization of a physical machine (PM) within a data center environment has been observed in the range between 15% to 20% for processing requests, with a significant portion of CPU resources remaining idle [6]. Moreover, findings indicate that during peak periods in university computer labs, only about 5% of CPU and 25% of RAM are utilized [7]. As per the Open Compute Project, Facebook's Oregon data center boasts a Power Usage Effectiveness (PUE) that signifies 93% of the energy utilized in the data center is dedicated to computing resources [8]. Additionally, findings reveal that

about 40% of the operational budget is allocated to electricity expenses and managing the generated heat within data centers. It reported that servers in U.S. data centers consume over 1.5% of the nation's total electricity, amounting to nearly $4.5 billion [9]. Recent studies indicate that global expenditure on enterprise power and cooling systems has exceeded $30 billion due to the growing number of servers [10]. Over the past decade, electricity consumption by servers has surged tenfold, raising concerns regarding escalating power consumption [11].

Inadequate resource management in the CDC leads to wasted investment and increased operational expenditures, primarily due to maintenance and electricity costs incurred by companies. Increased server loads exacerbate performance degradation, adversely affecting response times and overall system efficiency, ultimately undermining infrastructure reliability and user satisfaction. Moreover, a higher quantity of active servers can result in substantial environmental consequences, leading to excessive energy consumption contributing to carbon dioxide emissions and environmental degradation. Inefficient cooling systems and power distribution further exacerbate energy wastage, increasing the carbon footprint of data center [12]. Therefore, effective resource management has become imperative for optimizing performance, ensuring scalability, reducing expenses, and promoting environmental sustainability in cloud data centers.

Effectively managing data center resources poses significant challenges, including dynamically fluctuating workloads, optimal allocation of virtual machines (VMs) to physical servers, load distribution, the balancing act of overloaded and under-loaded servers and intricate decisions regarding VM migration and placement. Additionally, providers must determine the optimal number of active servers, provide Quality of Services (QoS) to users, reduce Service Level Agreement Violation (SLAV), minimize energy consumption while maintaining optimal server performance, and reduce the count of migrations to ensure efficient resource utilization.

Various computational approaches, including statistical methods, machine learning (ML), deep learning (DL), and meta-heuristic single-objective and multi-objective optimization techniques, have gained attention for their potential to efficiently allocate and consolidate VMs, thereby enhancing resource management in data centers. By developing a predictive and optimal resource allocation and consolidation framework for CDC utilizing these computational approaches, it becomes feasible to effectively utilize the resources of each active PM and mitigate the energy consumption of the data center.

Before outlining the research aim and specific thesis objectives, it is essential to introduce the overview of cloud computing, its architecture, cloud services, components of data

centers, virtualization, challenges encountered by CSPs in service provision, and strategies employed for resource management.

1.1 Overview of Cloud Computing

In the 1960s, researchers Douglas Parkhill and John McCarthy focused on developing the idea of utility computing. The primary objective of this model is to provide web-based services to customers like traditional utilities, such as natural gas, water, and power [13]. To put it in simpler terms, users just need access to the Internet to use a wide variety of services such as storage for files, deployment servers, and web-based applications hosted on the organization's computers accessible through the Internet [14, 15, 16]. Due to rapid growth in processing and storage technologies, along with the widespread adoption of the web or Internet, computing resources have become more affordable, more powerful, and more extensively prevalent compared to past periods. Therefore, the name of utility computing was converted into cloud computing [17]. Cloud computing is a modern paradigm in the information technology field, commonly recognized as a developing area in the discipline of computer science and known as "Internet computing" [18]. The term 'cloud computing' is derived from its association with cloud-like representations often used to represent networks and the Internet in pictures. It enables convenient access to data and applications through the Internet, allowing their access from any geographical location. It is imperative to acknowledge that cloud computing does not introduce completely new technologies; instead, it signifies a novel approach to utilizing pre-existing technologies and equipment [19]. Certainly, the lack of a universally accepted and standardized definition for cloud computing has not only increased excessive advertising claims but has also generated considerable confusion and ambiguity. [20] involved examining and analysing more than 20 distinct definitions obtained from various sources to validate a universally accepted definition. As per the National Institute of Standards and Technology (NIST), the standard definition that completely incorporates all fundamental characteristics is as follows:

> *"Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction"* [21]

This definition outlines five essential characteristics of cloud computing:

- **On-demand self-service** describes a user's convenient and automatic access to computing resources, including CPU, memory, network, and software usage. This

means that users can satisfy their urgent computing needs at specific times without needing to communicate with the companies that provide these services.

- **Broad network access** entails delivering computing resources over the network, such as the Internet. Different client programs running on heterogeneous platforms—such as PDAs, laptops, and phones that use these resources at the consumer's geolocation.

- **Resource Pooling:** In cloud computing, a provider combines its computing resources into a shared pool using either the multi-tenancy or virtualization approach. This process involves allocating and reallocating various physical and virtual resources according to consumer demand [22]. Establishing this computing paradigm centred around pools is motivated by two crucial factors: economies of scale and specialization.

- **Rapid elasticity** is a core feature of cloud computing that pertains to the capacity to rapidly and efficiently adjust computing resources in accordance with fluctuations in demand. This capability allows users to dynamically provision and release resources, often in an automated fashion, ensuring that the system can quickly expand or contract based on varying workloads.

- **Measurable Service:** The cloud infrastructure can effectively evaluate the utilization of computing resources for each consumer by employing appropriate mechanisms, even though these resources are shared and pooled across several customers (i.e. multi-tenancy).

Figure 1.1 illustrates that cloud computing provides seamless data retrieval across various devices. When connected to a reliable internet connection, users can utilize various services, such as servers, software platforms, storage, apps, and virtual desktops. The figure shows how cloud computing is compatible with laptops, desktop computers, printers, smartphones, and notepads.
Cloud computing offers a significant advantage by enabling both small and large enterprises to efficiently utilize modern software and hardware resources without the need for additional infrastructure investment, such as purchasing servers on a rental basis. This allows businesses to focus on their core operations rather than allocating time and financial resources to enhance their computer infrastructure. Moreover, cloud computing provides a conducive environment for enterprises to run applications faster with minimal maintenance requirements. This flexibility empowers IT teams to adjust resources dynamically in response to fluctuating business demands. By adopting a pay-as-you-go model, cloud computing provides customers with resources customized to their specific

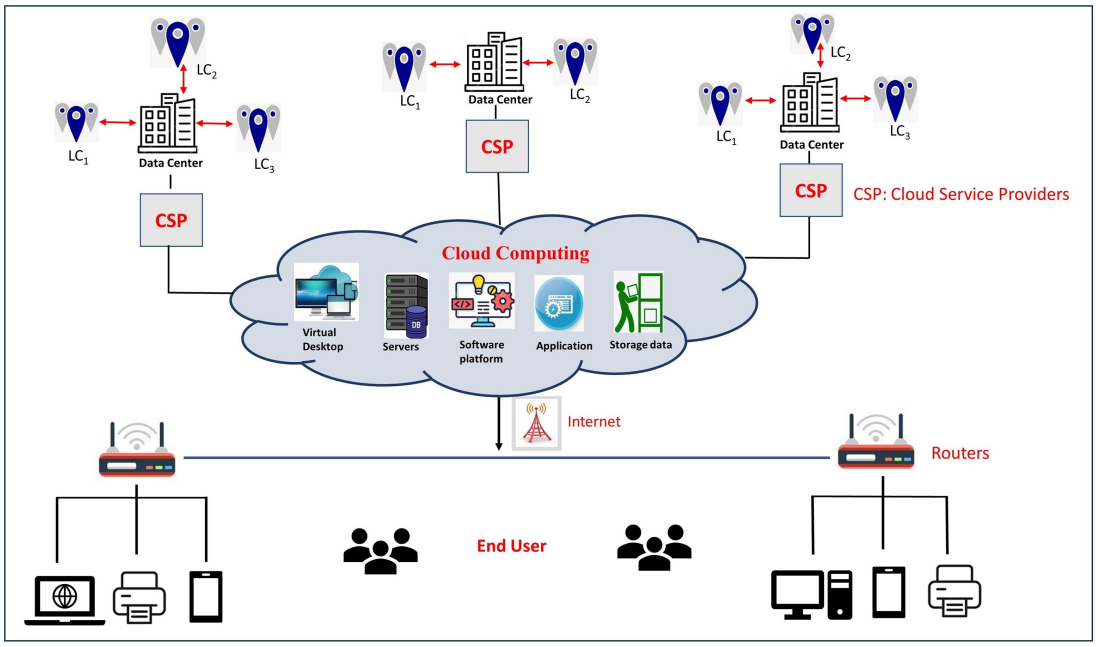


Figure 1.1: Overview of cloud computing

needs, thereby facilitating a more adaptable and cost-effective IT environment [23].

1.2 Cloud Computing Architecture

The architecture mainly consists of front-end and back-end components [24], as shown in Figure 1.2.

1.2.1 Front end - User Interface

It serves as the primary interface for clients, customers, and users to engage with the cloud environment, integrating the network-capable devices they use for access. The front-end architecture consists of three key components:

- **Web Browser:** This component allows cloud computing software to operate as a user terminal, adopting the form of either a web browser or a client application.
- **Google Docs:** Users can directly engage with the cloud through interfaces such as Gmail, Google Docs, or a text editor, facilitating seamless interaction with cloud resources.
- **Client PC and Networks:** This integral part of the front end encompasses the user's PC, input devices, and internet connections, enabling the execution of computing tasks in the cloud.

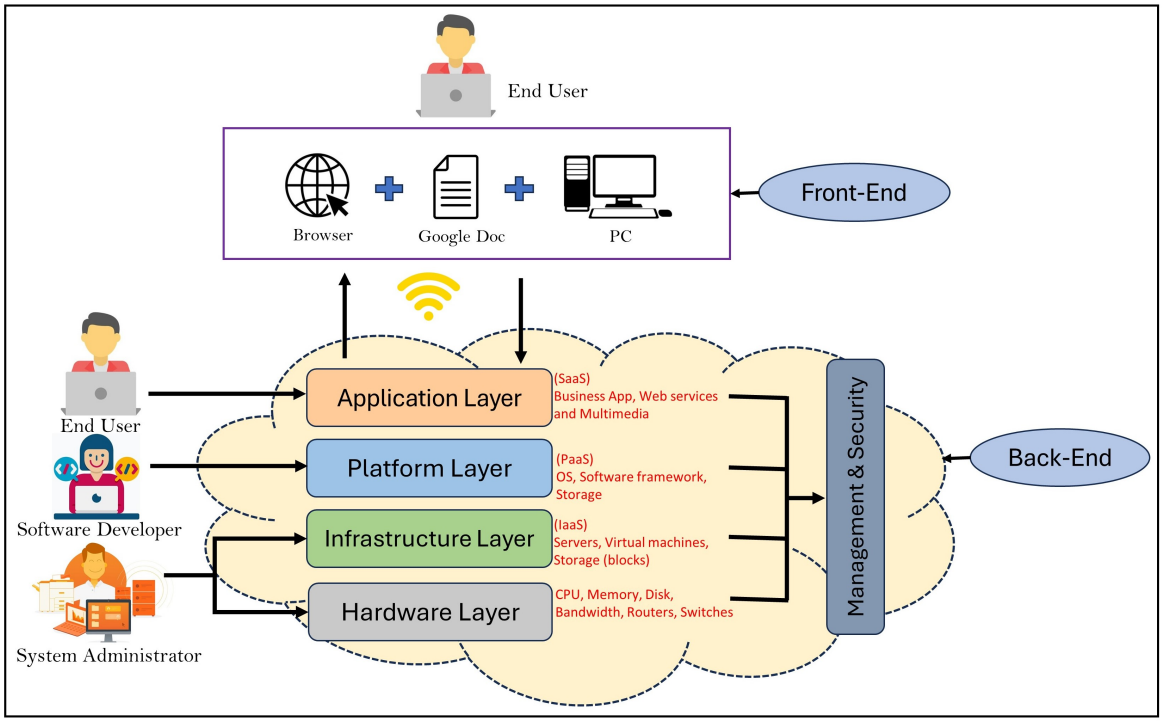


Figure 1.2: Cloud Computing Architecture

Various cloud computing systems employ distinct interfaces, offering a choice of web browsers like Chrome, Safari, Firefox and platforms like Google Docs.

1.2.2 Back end - Cloud Service Provider

CSPs manage the backend of cloud computing, which operates on remote servers. Typically, this architecture comprises four layers: Application layer/ Data center layer, Platform layer, Infrastructure layer, and Hardware layer. Additionally, it also incorporates security and management software [17].

- **Application layer:** The application layer, known as software as a service (SaaS), is an essential element in cloud computing architecture. The topmost layer comprises a range of software applications or online services enabled by SaaS to manage client requests and meet their requirements. The Application layer is mainly used as a distribution procedure, where a third party hosts the application and provides access to users via the Internet. SaaS eliminates the costs and demands associated with hardware, maintenance, licensing, installation, and support.

- **Platform layer:** This layer is characterized by the presence of operating systems and application frameworks designed to streamline the deployment of applications. Its primary purpose is to alleviate the challenges associated with directly deploy-

ing applications into VM containers. By providing a standardized environment and essential tools, the platform layer facilitates the development and execution of applications with greater efficiency. This layer abstracts much of the underlying infrastructure complexities, allowing developers to focus on coding and functionality rather than managing intricate details of the deployment environment. An exemplary instance of the platform layer in action is evident in platforms like Google App Engine, which operates within this stratum, furnishing robust API support for the seamless implementation of storage, database management, and business logic in typical web applications. Essentially, the platform layer serves as an intermediary that enhances the overall flexibility and accessibility of cloud-based application development.

- **Infrastructure Layer:** The Infrastructure Layer, also known as the virtualization layer, plays an essential role in cloud computing given that it utilizes virtualization technologies such as Xen [25], KVM [26], and VMware [27] to divide physical resources and establish a flexible pool of storage and processing capabilities. This layer comprises essential components such as CPU, Motherboard, GPU, VMs, virtualization software, and servers within the IaaS platform. It serves as the foundation driving entire cloud software services at application and network levels. System Administrators benefit from effortless access to scalable storage and computing power, enabling dynamic resource assignment and facilitating key features necessary for effective cloud computing environments.

- **Hardware layer:** This is the lowest level in cloud architecture, responsible for the physical control of essential components. This layer includes physical components such as database servers, routers, switches, memory discs, and other essential parts that are responsible for maintaining hardware configurations, assuring the ability to handle faults, and managing power supply and traffic within the cloud architecture. The hardware layer is primarily responsible for managing the actual components of the cloud infrastructure, such as servers, routers, switches, and power and cooling systems. This layer is commonly used in data centers and is responsible for coordinating numerous servers that are arranged in racks and connected through switches, routers, or other network fabrics. The hardware layer focuses on addressing key challenges such as hardware configuration, fault tolerance measures, traffic management, and the optimal utilization of power and cooling resources. These factors collectively contribute to ensuring the uninterrupted operation of cloud environments.

- **Management:** Management software is crucial in effectively assigning resources to

specific operations and maintaining the smooth operation of the cloud environment. It functions as an intermediary between the frontend and backend architecture, facilitating the cohesive functioning of the entire cloud computing system.

- **Security:** Security is an essential aspect of any cloud computing infrastructure since it is critical in effectively addressing possible security concerns through a well-coordinated debugging process. In the field of cloud security, the first step is to prioritize regular storage backups. In addition, virtual firewalls are crucial components used to maintain the overall security of the cloud computing system.

1.3 Cloud Services

Cloud computing offers a wide variety of services that address various demands and specifications. The main classifications of cloud services are usually known as the cloud service models and deployment models.

1.3.1 Service models

Service models in cloud computing relate to the different levels of abstraction and functionality that cloud service providers deliver to users. These models describe the nature of the services the level of user control over the foundational infrastructure, and the specific responsibilities of both the provider and the end user. The three primary service models in cloud computing are Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) [28].

- **IaaS:** It serves as the fundamental level, delivering virtualized computer resources via the internet. Customers have the option to lease VMs, storage, and networking components, paying only for the resources they use. This concept empowers users with enhanced control over the infrastructure and enables them to adjust the allocation of resources to meet their needs easily. Some examples of IaaS providers are GoGrid [29], Amazon Web Services (AWS) [30], Google Cloud Platform (GCP), IBM Cloud [31], and Flexiscale [32].

- **PaaS:** It simplifies the underlying infrastructure and offers a comprehensive framework for developers to create, deploy, and oversee applications. Users may concentrate on coding and application development without handling the complicated tasks of managing the underlying hardware and software infrastructure. Some examples of PaaS providers are Oracle Cloud Platform, Google App Engine [33], IBM Cloud Foundry and Windows Azure [34].

- **SaaS:** It provides entire software applications through the Internet, with users paying for access on a subscription basis. Users can conveniently access these applications via a web browser without the requirement of local program installation or maintenance. SaaS is specifically designed to be simple and easily accessible, making it an optimal solution for end-users. Some examples of SaaS providers are Salesforce [35], Zoom, Dropbox Business and Rackspace [36, 37].

According to the layered architecture of cloud computing, a PaaS provider can effectively use an IaaS provider's cloud infrastructure. However, it is common for IaaS and PaaS providers to be integrated within the same organization in current practice. This can be seen in companies such as Google and Salesforce. Therefore, vendors of PaaS and IaaS are often called infrastructure providers or cloud providers [38].

1.3.2 Deployment model

Several factors must be considered when shifting an enterprise application to a cloud environment. For example, some service providers may prioritize a reduction of costs in their operations, while others may prioritize achieving higher levels of reliability and security. Deployment models in cloud computing refer to the strategies by which cloud services and resources are allocated, organized, and delivered to users. These models determine the deployment and management of cloud infrastructure [39, 40]. The primary deployment models in cloud computing are:

- **Public cloud** deployment paradigm involves the offering of cloud services and resources to the public via the Internet. The third-party cloud service provider is responsible for owning, operating, and maintaining the infrastructure. Public clouds are often capable of being scaled up or down and provide a pricing structure where users only pay for the resources they use, making them economically advantageous for organizations with fluctuating workloads. However, public clouds have limits since they lack specific oversight over data, network, and safety parameters, which might affect their effectiveness in different business situations.

- **Private cloud**, often known as internal clouds, are specifically designed for exclusive usage by a particular organization. Whether built and regulated within the organization or outsourced to external vendors, a private cloud guarantees maximum control over performance, dependability, and security. However, these systems occasionally receive criticism for their similarities to traditional proprietary server farms and their lack of benefits, such as eliminating initial capital expenses.

- **Hybrid cloud** combine public and private cloud models to address the limitations

of each technique. The configuration involves a portion of the service infrastructure operating within private clouds, accompanied by another portion running on public clouds. The hybrid cloud approach offers increased flexibility in comparison to both public and private clouds. Significantly, it allows for stricter regulation and protection of application data compared to public clouds while also facilitating the ability to adjust service capacity as needed rapidly. However, creating a hybrid cloud requires a careful assessment to identify the best balance between public and private cloud elements.

- **Virtual private cloud (VPC)** is a solution that offers an alternative to solve the limitations found in both public and private clouds. It is a platform that operates on top of public clouds and sets itself apart by utilizing virtual private network (VPN) technologies. Service providers can customize their own topology and security settings, which include the configuration of firewall rules. A VPC, in contrast to standard configurations, has a more extensive architecture that not only virtualizes servers and applications but also extends this virtualization to the underlying communication network. Furthermore, a VPC provides a smooth shift for several enterprises from a dedicated service infrastructure to a cloud-based alternative, made possible by the virtualized network layer [41, 42].

1.4 CDC Components

A CDC is a physical building where computer systems and their related equipment are maintained. It includes the computational infrastructure needed to operate IT systems, including servers, data storage equipment, and network devices. These centralized locations are designed to provide scalable and on-demand access to computing resources that allow users to store, process, and manage data without needing on-premises infrastructure. Below are the primary components use in CDC [43, 44].

- **Server:** Servers in a CDC represent PMs that store, process, manage and run applications for cloud computing services. The servers are the foundation of the cloud architecture and store virtualized computing resources such as VMs, software packages, and storage solutions. CDC contain numerous servers that are connected by networking equipment to facilitate smooth communication and data exchange. CSPs manage and maintain these servers to provide reliable, scalable, and efficient computing services to individuals and enterprises.

- **Storage system:** It refers to the infrastructure that is responsible for effectively and efficiently managing and storing data. These systems leverage technologies such

as hard drives, solid-state drives (SSDs), and storage area networks (SANs) to offer storage solutions that are both scalable and dependable. Cloud storage systems are essential for securely storing and retrieving large volumes of information, thus guaranteeing data durability and availability.

- **Networking equipment:** Components such as routers, switches, and firewalls provide data transfer and communication within the data center and between the data center and third-party networks. An efficient network infrastructure is essential for guaranteeing fast response times and high data transfer rates.
- **Power & cooling equipment:** The equipment in a CDC is essential for maintaining optimal conditions for operation. This infrastructure comprises uninterruptible power supply (UPS) devices and backup generators to guarantee a faultless and continuous power supply. In addition, air conditioning or liquid cooling systems are used to control the temperature and prevent servers and networking equipment from overheating.
- **Backup and Recovery Systems:** These systems are essential procedures created to protect the accuracy of data and guarantee the uninterrupted operation of an organization. These systems feature periodic data backups, which require generating duplicate copies of critical data. Recovery techniques are established to return data to its original state in the event of data loss or system failures.
- **Load Balancers:** It is a hardware device or software which is used to evenly distribute incoming network traffic among numerous servers, leading to maximizing resource usage and preventing any single server from getting overloaded. Through the act of distributing the workload evenly, they optimize performance, reduce reaction times, and increase the overall dependability of applications.
- **Environmental Controls:** These systems are used to supervise and control the actual state of the facilities. This includes the maintenance of appropriate humidity, temperature levels, and air quality to guarantee the consistent functioning and durability of equipment. Environmental controls are essential to prevent excessive heat and reduce the likelihood of equipment malfunctions, hence enhancing the overall reliability and effectiveness of the data center structure.

1.5 Virtualization

Virtualization in the cloud refers to the process of creating virtual instances of computing resources such as servers, storage, networks, and applications within a cloud environment.

This technology allows multiple VMs or containers to run on a single PM to maximize resource utilization and flexibility. Through virtualization, users can abstract underlying hardware complexities and enable more efficient resource management. This approach allows scalable adjustments based on demand, without investing in physical infrastructure. It enables the consolidation of workloads, improves resource allocation, enhances security through isolation, and facilitates easier migration and disaster recovery processes. Cloud computing is built on virtualization technology, which enables service providers to deliver customized and readily available services. This foundation offers consumers increased flexibility, cost efficiency, and reliability in managing and deploying their software applications and data [45, 46].

Figure 1.3 depicts a virtualized environment. At its core is a PM, which serves as the

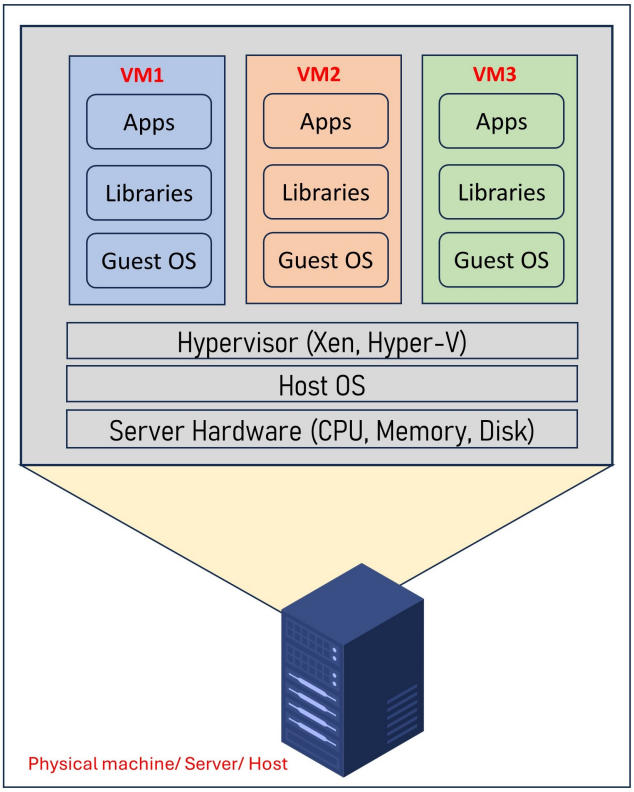


Figure 1.3: Virtualization

fundamental hardware support. This PM possesses its own set of hardware resources, such as CPU, memory, storage, and network interfaces. Above the PM, there is a layer known as a virtual machine monitor (VMM) or hypervisor. It acts as a thin software layer that abstracts and manages the physical hardware resources, which facilitates the creation and maintenance of several VMs on a single PM. Within this setup, the PMs' resources are partitioned or divided into multiple VMs. Each VM is an independent entity with its own virtualized hardware resources, including virtual CPUs, memory, disk space,

and network interfaces. These VMs are isolated from each other and operate independently, much like separate PMs. Each VM also runs its own guest operating system (OS) and associated applications or services. The guest operating system functions within the virtualized environment facilitated by the hypervisor and communicates with the virtual hardware as if running on dedicated physical hardware. Furthermore, each VM may have its own set of libraries, dependencies, and application software installed, tailored to its specific requirements. These components are encapsulated within the VM environment and do not interfere with other VMs running on the same PM.

1.6 Challenges to Cloud service providers

Despite the widespread adoption of cloud computing worldwide, there are still numerous research issues that need to be fully addressed to enhance the utilization and effectiveness of cloud services. CSPs deal with numerous issues in efficiently maintaining infrastructure resources and providing user services [47, 48].

- **Resource utilization:** CSPs must assure optimal utilization of their physical hardware resources, including CPU, memory, storage, and network bandwidth. This entails the dynamic allocation of resources to VMs or containers in response to demand while minimizing resource inefficiency [49, 50].
- **Energy consumption:** Data centers require substantial quantities of energy to operate and maintain servers and other infrastructure elements. CSPs strive to decrease energy consumption through the optimization of hardware utilization, the implementation of energy-efficient technology, and the utilization of modern cooling technologies [51, 52].
- **Load balancing:** It is the process of equitably dividing incoming network traffic or workload requests among multiple servers or VMs in order to avoid overwhelming any individual resource. CSPs employ load-balancing algorithms to optimize resource utilization, guarantee high resource availability, and enhance overall performance [53, 54].
- **Response time:** It is the duration that takes for a system to react to a user's request or inquiry. CSPs aim to reduce response times by optimizing network latency, minimizing processing overhead, and leveraging caching and content delivery networks to deliver content more efficiently to consumers [55, 56].
- **Throughput:** The speed at which data can be processed or transferred within a system. CSPs aim to maximize throughput by optimizing network bandwidth,

minimizing data transfer latencies, and improving data processing efficiency across their infrastructure [57].

- **Detect overloaded and underloaded hosts:** CSPs must consistently monitor their infrastructure to detect PMs or VMs that are either overloaded with resources or not being utilized to their full potential. Monitoring methods and algorithms are utilized to identify deviations in performance, forecast resource requirements, and adaptively allocate resources to uphold ideal system performance [58].

- **Virtual machine migration:** It refers to the process of migrating a functioning VM from one PM to another without causing any interruption to the availability of services. CSPs employ live migration methodologies and migration procedures to evenly distribute workloads, maximize resource utilization, and carry out hardware repairs or updates with minimal interruption [59].

1.7 Resource Management in CDC

To optimally utilize the resources of PMs in a CDC is paramount for maximizing performance, ensuring cost-effectiveness, and minimizing energy usage [60]. Efficient allocation and management of these resources are pivotal for enhancing the data center's operational efficiency. Several strategies have been investigated in the literature to accomplish this goal, including virtualization, load balancing, task scheduling, allocation policies, predictive analytics, VM migration, and server consolidation [61, 62, 63].

To effectively utilize the resources of a PM, this thesis focuses on investigating and developing innovative algorithms in three key areas: VM Allocation, Resource usage prediction, and Dynamic VM Consolidation approach [64, 65, 66, 67, 68, 69].

Let's consider a general scenario of a CDC where v VMs are requested by users using a cloud application environment, each with specific resource requirements. There are p PMs available in the data center, each with specific resource capacities.

$$V = \{1, 2, \ldots, v\}\text{: Set of VMs.}$$
$$H = \{1, 2, \ldots, p\}\text{: Set of PMs.}$$

When users request these v VMs, the requests are sent to the CSP for processing. Afterwards, the VM allocator determines how these v VMs will be allocated to the p PMs in the data center using the VM allocation strategy.

1.7.1 Virtual Machine Allocation (VMA)

VMA in cloud computing refers to the process of assigning and provisioning VMs to PMs within a cloud environment. As illustrated in Figure 1.4, several users request cloud services through a web-based cloud application, specifying their needs in terms of CPU, memory, and disk space. These requests are sent to the CSP, where the VM Allocator plays a crucial role. The VM Allocator is an optimization algorithm that determines which requested VMs should be assigned to which PMs to maximize the resource usage of each PM. The resource manager provides detailed information about the available resources of each PM to the VM Allocator. This ensures that the VM Allocator can adhere to resource constraints. The hypervisor then uses the VM Allocator's decisions to allocate VMs to PMs [66].
This optimal allocation strategy maximizes the utilization of each active PM and ensures that only the optimal number of PMs are active, which helps to conserve energy and enhance overall efficiency [66, 70]. The VMA process involves static allocation and dynamic allocation.

- **Static Allocation (Fixed Allocation):** This approach involves assigning a fixed quantity of CPU, memory, and storage to a VM when it is created, and these allocations remain unchanged regardless of the workload. This technique ensures a consistent and reliable supply of resources, but it may result in inefficient usage during times of decreased demand [70].

- **Dynamic Allocation:** This refers to the process of modifying the assigned resources for a VM in response to its current workload. This enables adaptability and quickness in response to fluctuating resource requirements. The system possesses the ability to dynamically adjust resource allocations, scaling them up or down based on demand. In addition, this approach optimizes resource utilization and ensures that VMs consistently have adequate resources available when required [71].

Traditional data centers typically function at fixed allocation, resulting in constant consumption of energy regardless of the actual workload requirements. In contrast, dynamic allocation enables flexible change in resource allocation, resulting in a more optimized distribution of computing jobs. This flexibility not only decreases the requirement for keeping extra capacity but also allows data centers to turn off or consolidate machines during periods of less demand. Consequently, the total energy usage of data centers is significantly reduced, thus supporting goals for sustainability and reducing operational costs. Hence, VMA plays a crucial role in optimizing performance and reducing environ-

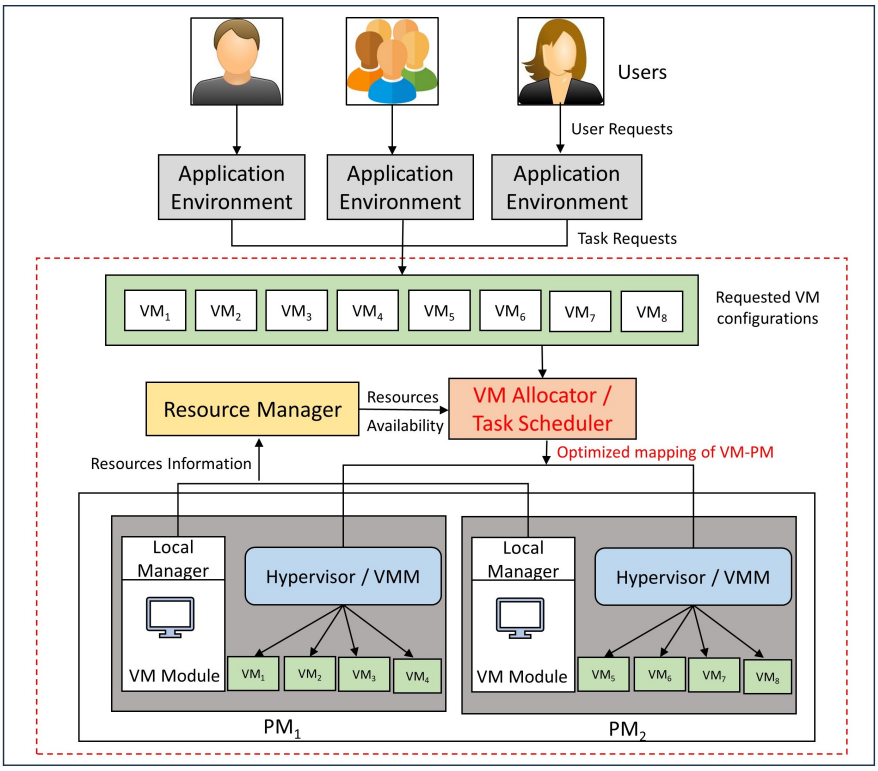


Figure 1.4: Virtual Machine Allocation in CDC

mental impact in cloud computing infrastructures by effectively utilizing resources and implementing energy-efficient procedures.

1.7.2 Resource Usage Prediction

Resource usage prediction in the CDCs refers to the process of estimating future resource demands and usage patterns based on historical data and current trends. It involves analyzing various parameters such as CPU utilization, memory consumption, network traffic, and application behaviour to predict how these workloads will evolve over time [72, 73].

As depicted in Figure 1.5, multiple users request cloud services through a web-based cloud application. These requests are sent to the CSP as VM configurations, specifying CPU, memory, and disk space requirements. Afterwards, the VM allocator optimally assigns the requested VMs to available PMs based on resource constraints using the hypervisor of a PM. A Resource Manager maintains comprehensive information on the resources used by VMs and PMs, including current usage, VM-to-PM assignments, and overall PM resource utilization. This information is provided to both the VM allocator and a prediction model. The prediction model leverages historical workload patterns to predict future resource usage for both VMs and PMs.

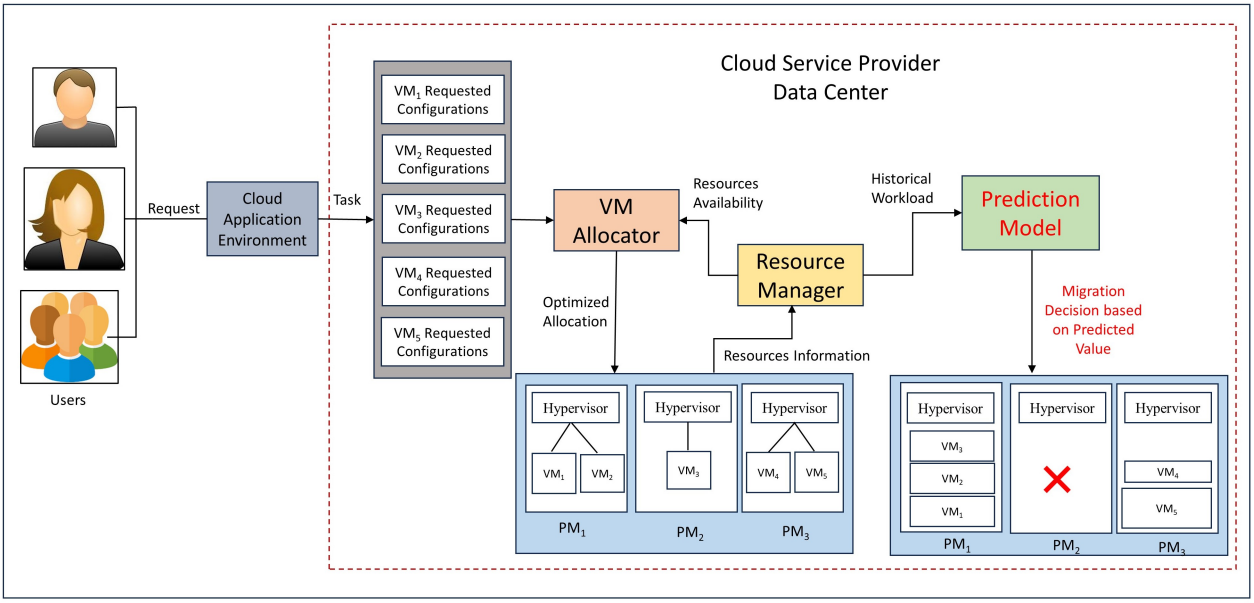


Figure 1.5: Resource usage Prediction in CDC

Accurate workload prediction is pivotal in resource management, as it empowers CSPs to make informed decisions about resource provisioning and allocation. By leveraging historical data in machine learning algorithms for predictive analytics, organizations can develop sophisticated models that predict future workload patterns with a high degree of accuracy. This foresight enables data center operators to anticipate peak periods, fluctuations in demand, and seasonal variations, allowing for proactive adjustments to resource allocations and capacities [72].

Workload prediction optimizes resource utilization by aligning resource allocation with anticipated demand, minimizing shortages and avoiding over-provisioning. It supports dynamic scaling and load balancing, enhancing performance and user experience in cloud environments. Additionally, accurate workload prediction aids energy efficiency through strategic task scheduling, load balancing, and server consolidation [74, 75].

1.7.3 Virtual Machine Consolidation (VMC)

VMC in cloud computing refers to the process of consolidating multiple VMs into fewer PMs. As shown in Figure 1.6, after receiving the user's request, the CSP examines the available resources of PMs based on their resource usage and allocates VMs to suitable PMs according to their requested requirements. However, over time, the distribution of VMs across PMs can become uneven due to fluctuating resource demands. This imbalance results in inefficiencies in resource utilization and a reduce overall performance. Therefore, the VMC approach plays a crucial role in resolving this issue. Resource usage prediction

helps to make better decisions at each step of VMC, which maximizes resource usage and minimizes energy consumption.

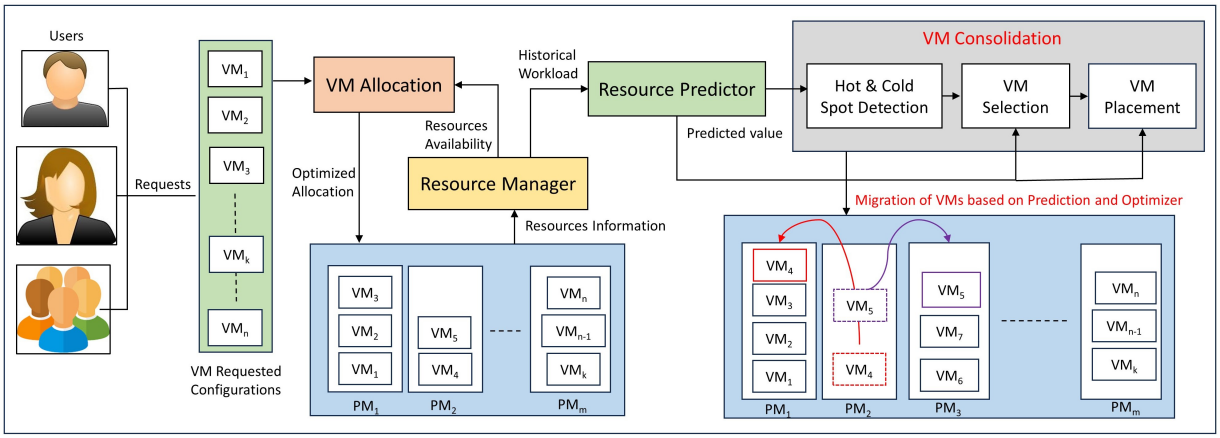


Figure 1.6: Virtual Machine Consolidation in CDC

VMC is required in the CDC to fulfil various important goals. First and foremost, it aids in optimizing the utilization of resources, which leads to cost reductions and enhanced efficiency. It decreases the number of underutilized or inactive servers, hence lowering wasted resources and decreasing operational expenses related to power, cooling, and hardware maintenance. It improves the ability to handle increasing workload needs and adapt to changing circumstances, enabling cloud providers to be more flexible and scalable. Moreover, the consolidation of VMs enhances resource management and their allocation. It ensures each VM receives adequate computing resources and prevents conflicts and performance degradation by strategically organizing VMs based on their resource requirements and usage patterns. By strategically planning and executing consolidation techniques, CSPs can successfully achieve a balance between resource utilization and workload demands. This enables the provision of trustworthy and economical cloud services to clients.

VMC is classified into two types: Static virtual machine consolidation (SVMC) and Dynamic virtual machine consolidation (DVMC).

- **DVMC:** Within a cloud environment, workloads are executed in VMs, which utilize the resources of the hosting PMs to complete their assigned tasks. As time progresses, existing workloads continue to evolve while new workloads are continuously accepted by CSP. Additionally, there is a process of replacing certain PMs due to technical issues, as well as the inclusion of new PMs. Hence, the total workload in the CDC, together with the related need for resources and the availability of resources, continuously changes over time. The DVMC algorithm incorporates the

current assignment of VMs to PMs when performing the VMC procedure. It should be noted that the workload or resource demand of a VM and its location, which refers to the PM hosting it, might be subject to change over time. The VMC procedure is referred to as the DVMC algorithm when it takes into account the dynamic workload and the current assignment of the VM to the server. DVMC algorithms provide a solution for redistributing existing VMs across a reduced number of PMs, aiming to minimize the number of active PMs in operation.

- **SVMC:** Unlike the DVMC method, SVMC algorithms, also known as consolidated VM placement algorithms, do not consider the existing assignment of VMs to PMs when selecting a new destination PM for any VM. The authors stated in [76] that SVMC algorithms operate using a collection of completely empty PMs and a collection of VMs with defined resource demands. The SVMC algorithm efficiently determines the optimal initial placement of VMs by minimizing the number of active PMs. This results in improved energy conservation and resource utilization for the CDC. However, it fails to address the issue of reallocating VMs to new PMs while considering the present VM-to-PM assignment. The SVMC algorithm is named as such because it does not consider the dynamic workload and placement of VMs. The SVMC technique, exemplified by [77, 78], does not consider the present assignment of VMs to servers when selecting a new destination PM for a VM.

SVMC algorithms are mostly suitable for the initial allocation of VMs or the migration of VMs between data center clusters. Over time, the workload and availability of resources in the CDC vary. Thus, in addition to the initial placement of VMs, DVMC becomes a vital method for upholding energy efficiency, enhancing utilization of resources, and maximizing revenue for CSPs.

1.8 Research Motivation

Numerous studies have focused on enhancing the effective management of resources in CDCs. This includes research efforts aimed at improving resource allocation and consolidation strategies. Various statistical, machine learning and deep learning methods have been developed to predict resource utilization patterns accurately and efficiently. These strategies utilize past data and continuous monitoring to enhance decision-making on the allocation of resources. Moreover, single-objective and multi-objective heuristic and metaheuristic algorithms were developed for virtual machine allocation and consolidation. These algorithms provide several methods for improving resource utilization while taking

into account varied goals like as performance, energy efficiency, and cost. The ongoing evolution of these methodologies highlights the importance of continuous innovation in addressing the complexities of resource management in cloud environments.

The existing resource usage prediction approaches often have difficulties adapting to dynamic patterns, leading to suboptimal performance in the cloud computing environment. The limitations of handling complex nonlinear patterns and capturing long-term dependencies can lead to inaccurate predictions and ineffective resource allocation. Additionally, the vanishing gradient problem poses a significant obstacle in training machine learning models on sequential data, thereby limiting their capability to capture intricate temporal relationships effectively.

Similarly, many existing approaches used in literature for VMA and VMC encounter significant challenges that impact their effectiveness in real-world cloud computing environments. One key issue is the sensitivity to local optima, where optimization algorithms may converge to sub-optimal solutions without comprehensively exploring the entire solution space. This limitation can result in inefficient resource utilization and performance degradation, especially in dynamic and evolving workload scenarios. Furthermore, the convergence rate of optimization techniques applied to VMA and consolidation can be problematic, leading to premature convergence and potentially missing better solutions. Another critical issue is the difficulty in balancing multiple conflicting objectives simultaneously, such as maximizing resource utilization, minimizing energy consumption, ensuring QoS guarantees, and optimizing server consolidation. Traditional approaches often struggle to achieve an optimal trade-off among these objectives, leading to sub-optimal allocation decisions and inefficient resource management.

Furthermore, there is a lack of research analysing the impact of short-term dependencies, long-range dependencies, and sequential data patterns on resource usage prediction. Additionally, many studies rely solely on the current resource usage of PMs for detecting overloaded and under-loaded machines, selecting machines for migration, and determining machine placement. However, this approach can lead to frequent migrations, resulting in higher energy consumption and operational costs. Moreover, there is a notable absence of research utilizing a single optimization technique capable of both allocating and consolidating resources seamlessly. Most studies focus on either allocation or consolidation separately without demonstrating how these strategies can effectively work in parallel to optimize resource management and minimize inefficiencies in cloud environments.

Consequently, there is a critical imperative to investigate more resilient and adaptable methodologies to improve the accuracy and responsiveness of resource usage prediction in heterogeneous environments within cloud computing and data center management. In addition, analysing the impact of dependencies and the role of sequential data helps in

achieving better accuracy in complex and dynamic resource usage patterns. Addressing optimization challenges requires the development of more sophisticated optimization methodologies, such as multi-objective and adaptive algorithms, capable of effectively navigating complex solution spaces and dynamically adjusting resource allocations based on real-time conditions. In addition, it is essential to develop comprehensive and integrated solutions that improve resource utilization, reduce energy consumption, and optimize operational costs in cloud computing. These problems and gaps motivate us to build an optimum unified framework for cloud data center to allocate and consolidate VMs using accurate predictive resource utilization.

1.9 Research Objectives

This research aims to propose an optimized resources consolidation framework for CDCs, which helps to utilize resources effectively and consolidates VMs to minimize energy consumption. The following are the specific objectives undertaken:

1. To study and analyse existing virtual machine placement, selection, consolidation and resource prediction techniques.
2. To propose workload prediction model for resources utilization of cloud data center.
3. To propose optimum compute resources consolidation framework (using virtual machine placement and selection) for energy-efficient cloud data centers.
4. To validate and evaluate the performance of the proposed techniques of workload prediction and virtual machine consolidation.

1.10 Thesis Organization

As seen in Figure 1.7, this thesis is organized into seven chapters, with brief descriptions summarizing the content of each chapter.

This chapter established the thesis's context and motivation. The purpose, research topics, and objectives have been discussed.

Chapter 2 provides a comprehensive review of the literature on resource usage prediction, VMA, and the methodologies employed in the VMC process. This includes the identification of overloaded and under-loaded hosts, as well as the selection and placement of VMs. Additionally, this chapter presents key observations identified as research gaps, which will be addressed in this thesis.

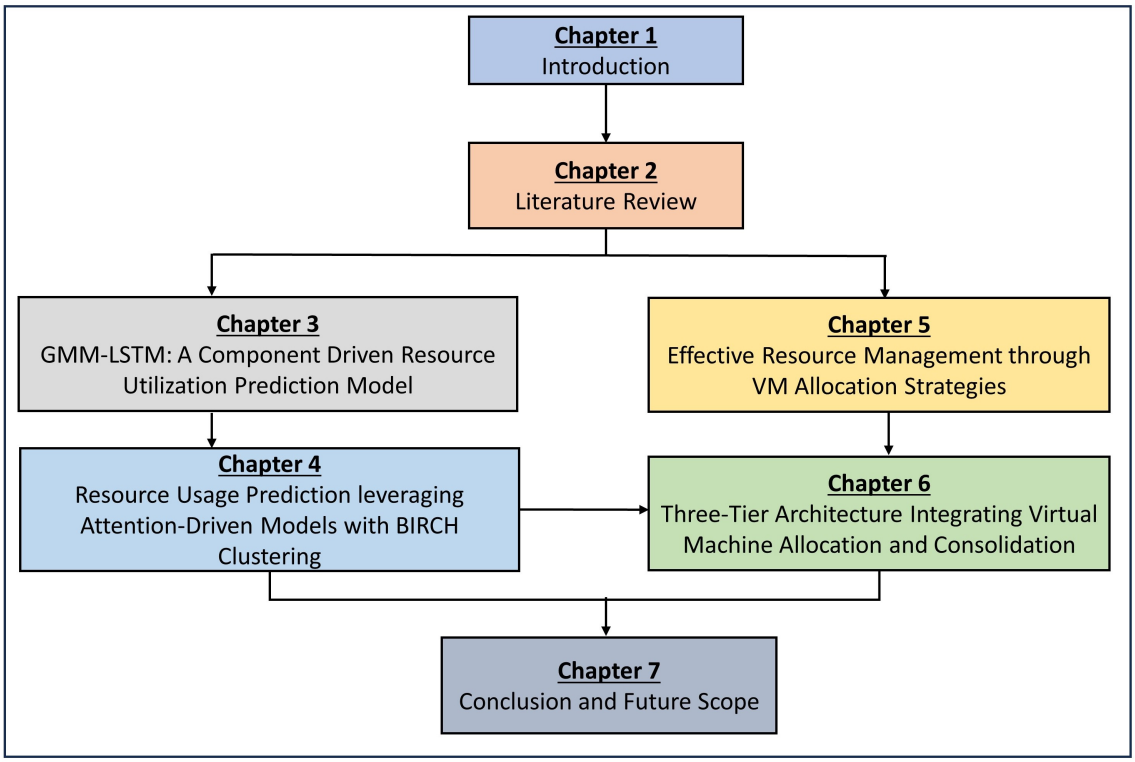


Figure 1.7: Flowchart describing the thesis structure

Chapter 3 presents the proposed GMM-LSTM prediction model designed to predict resource utilization in PMs. This approach begins with the proposed Sum-Average (SA) algorithm for preprocessing data. Afterwards, Gaussian Mixture Model (GMM) clustering will be employed to categorize PMs based on their average CPU and memory usage, facilitating the identification of a subset of heterogeneous machines. Lastly, employ the Long Short-Term Memory (LSTM) model to predict the CPU usage patterns of a PM. Evaluation using the Google cluster trace usage (GCT) dataset validates the effectiveness of our proposed model. Comparative analysis against linear regression, moving average, and autoregressive integrated moving average models demonstrates the superior performance of our approach, as evidenced by root mean square error (RMSE) analysis.

Chapter 4 explores deep learning attention-based mechanism models Transformer and Informer along with the LSTM model—within the context of a cloud computing environment. We comprehensively evaluate and compare their performance by predicting the CPU usage of a PM. In addition, leveraged the Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) algorithm to extract a subset of heterogeneous PMs from the dataset by considering average CPU, memory, and hard disk usage.

Chapter 5 employed single-objective metaheuristic algorithms, including first fit (FF), first fit decrease (FFD), best fit (BF), best-fit decrease (BFD), and genetic algorithm

(GA), to allocate VMs to PMs. The primary objective was to enhance resource utilization by minimizing the number of active machines. The obtained results are used to investigate whether a single objective is sufficient to manage resource usage and reduce energy consumption in data centers.

Chapter 6 presents a three-tier architecture designed for optimal allocation and consolidation of VMs using current and predicted workloads. This approach integrates the non-dominated sorting genetic algorithm (NSGA-II) algorithm to efficiently map VMs to PMs, while the LSTM model accurately predicts CPU usage of allocated PMs. We also present a Pareto front-inspired VM selection technique. This architecture aims to reduce active PMs, migration counts, and energy consumption of the cloud data center.

Finally, **Chapter** 7 summarises the thesis by highlighting the contributions of the proposed schemes. It also outlines future research to improve data center resource utilization and energy consumption.

Chapter 2

Literature Review

Our research focused on three key strategies aimed at optimizing resource utilization within cloud data centers (CDCs): Resource usage prediction, virtual machine allocation (VMA), and virtual machine consolidation (VMC). This chapter offers a detailed discussion of prior research aimed at optimising resource utilization in CDCs. Section 2.1 describes the existing research to predict resource utilization across physical machines (PMs) within CDCs. Section 2.2 offers a detailed exploration of VMA strategies. Section 2.3 presents various techniques utilized in previous studies, covering each stage of VMC, including the identification of overloaded and underloaded physical machines, VM selection, and placement. Section 2.4 outlines the identified gaps in the literature review process. Figure 2.1 shows the interconnected relationship between each strategy for resource management in CDCs using a Venn diagram.

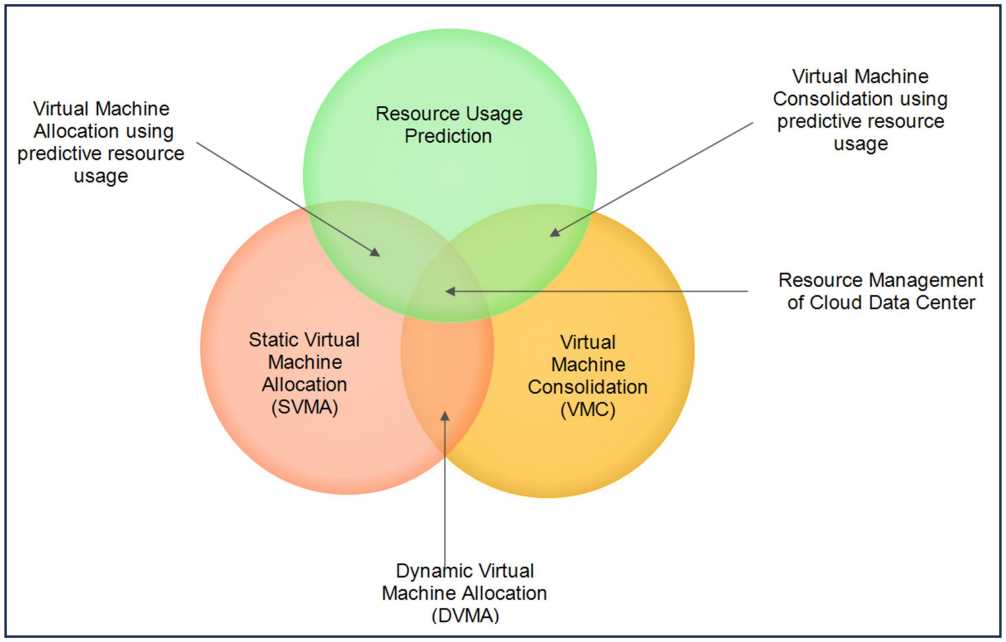


Figure 2.1: Relationship between Resource usage prediction, Virtual machine allocation, and Virtual machine consolidation

2.1 Resource Usage Prediction

As the demand for cloud services keeps rising, the dynamic nature of workloads also increases. This leads to overutilization and underutilization of PMs, resulting in resource usage imbalances within data centers. Therefore, to make better decisions for resource management, it becomes crucial for cloud service providers (CSP) to accurately predict resource usage of a data center that will help to efficiently allocate resources, optimize performance, prevent over-provisioning or under-utilization, and minimise energy consumption and expenditure [79].

Workload prediction of resources is a forward-looking method that gives an approximate idea of future demand for computing resources, storage, and network bandwidth by analysing previous usage patterns, application needs, and external variables. In cloud computing, the ideas of workload forecast and resource usage prediction are closely tied to one another. Workload prediction involves precisely forecasting the demand patterns based on various resource needs. On the other hand, resource usage prediction is a method that emphasises predicting the consumption of specific server resources such as central processing unit (CPU), memory, and storage. Both ideas work together to make it possible to allocate resources effectively, optimize costs, and improve performance inside cloud systems.

In the literature, various resource usage predictors have been employed by researchers to address specific challenges [65, 80]. In this section, we classify existing studies into four main categories: statistical models, machine learning models (ML), deep learning models (DL), and hybrid models. This taxonomy offers a comprehensive overview of the many approaches to forecasting cloud computing resource utilization.

2.1.1 Statistical models

An approach to predict cloud load specifically designed to accurately estimate the load on a host over long periods, up to 16 hours is introduced [81]. This approach primarily focused on CPU and memory by utilizing a Bayesian model to extract valuable information during variations in workload and noise. To verify the effectiveness, they conducted a thorough evaluation utilising a detailed dataset of one month's worth of load traces from a large Google data center, encompassing more than $10,000$ machines. The contributions were diverse, including precise forecasts of average load for future and successive periods and developing new characteristics for Bayesian prediction. This method demonstrated exceptional efficacy in predicting cloud computing load, as confirmed by thorough comparisons with state-of-the-art techniques. The Bayesian strategy emerged as the leading

method, outperforming competitors by significant margins ranging from 5.6% - 50%. It achieved a mean-squared error (MSE) of 0.0014 for a single interval and did not exceed 10^{-5} for a pattern. In the same year, [82] introduced RPPS (Cloud Resource Prediction and Provisioning Scheme) approach to proactively manage resources in CDCs. RPPS was based on the autoregressive integrated moving average (ARIMA) model for workload prediction, which is an example of effective resource allocation. RPPS effectively manages demand fluctuations in cloud settings by utilising coarse-grained and fine-grained resource scaling techniques and a deliberate virtual machine (VM)-complementary migration strategy. This approach achieves a prediction accuracy of around 90%, proving its effectiveness in dealing with the dynamic nature of cloud workloads. RPPS has been successfully implemented and tested on Xen and KVM virtualisation platforms. It has proven highly effective in real-world data center scenarios, providing a potential solution for optimising cloud resource provisioning and workload management.

Similarly, [83] proposed a Statistical-based Load Balance (SLB) that leverages statistical prediction and resource evaluation for online resource allocation. Unlike Service Level Agreements (SLA) based methods, SLB achieves load balancing by forecasting VM resource demand. It comprises two components: (1) Data analysis of historical performance to predict VM resource demand and (2) An algorithm for selecting an appropriate host in the resource pool. Their experiments demonstrated SLB's ability to achieve timely load balance and more equitable resource utilization. [84] introduced a method to address the increasing demand for resources in cloud systems. Their approach focused on employing the Grey Forecasting model as a method for workload prediction, offering a fresh perspective on cloud forecasting. By utilising the natural time-dependent characteristics of daily workload patterns, introduced prediction method allowed for the prediction of VM workload trends with exceptional accuracy, facilitating faster VM allocation and decreased overall energy usage. By conducting thorough simulations, the effectiveness of approach was demonstrated, highlighting the ability to predict workloads while minimising data consumption accurately. The resource distribution guided by model resulted in substantial power savings, highlighting the advantages of both efficiency and sustainability. Upon concluding their analysis, the authors highlighted the crucial importance of resource forecasting in the allocation process. They praised the Grey Forecasting model as an innovative and inventive CDC resource management method.

[85] presented a service cloud architecture focusing on scalability by utilizing a linear regression model for workload prediction. The auto-scaling mechanism represents a significant contribution by combining real-time and pre-scaling techniques to address self-healing, resource-level scaling, and VM-level scaling. Their approach effectively meets

SLA while minimising scaling costs. Contributions include the linear regression model, the service cloud architecture, and the auto-scaling approach, with experiments reveal superior accuracy, cost-effectiveness, and reduced SLA violations compared to other methods. The generic nature of the approach makes it adaptable to various service cloud scenarios, promising accurate predictions and efficient resource management. In the evolving landscape of cloud-based Software as a Service (SaaS) applications, the competition among providers necessitates ensuring high Quality Of Service (QoS) to retain user satisfaction and loyalty.

[86] addressed the challenge of maintaining QoS while optimising resource utilization in response to fluctuating workloads throughout time. Their proposed solution involves proactive, dynamic provisioning of resources by leveraging the ARIMA model for workload prediction. They did a thorough evaluation using real web server request traces, and the study demonstrates the effectiveness of the ARIMA-based predictions by achieving an average accuracy of up to 91%. [87] presented a univariate linear regression (LR) model for short-term load forecasting (STLF) utilising daily load cycles. This model facilitate the forecasting work by utilising patterns as input and output variables, thereby eliminating trends and seasonal changes that extend beyond daily cycles. Employing techniques such as stepwise and lasso regressions aimed to decrease the number of predictors, whereas principal components regression and partial least-squares regression utilize only a single predictor, allowing for the display of the regression function. Comparative assessments indicated that these models outperform standard techniques, such as ARIMA and exponential smoothing, especially regarding shorter forecasting periods. The local modelling technique prioritises simplicity and avoids overfitting to offer benefits compared to sophisticated models with many parameters. This study indicates that the suggested linear models outperform non-linear models such as Multilayer Perceptron (MLP) and N-WE in extrapolation. These linear models have an advantage since they require fewer parameters while simplifying estimation.

[88] assessed various prediction algorithms, such as First-order AutoRegressive model (AR(1)), First-order Moving Average model (MA(1)), Simple Exponential Smoothing (SES), Double Exponential Smoothing (DES), ETS (Error, Trend, Seasonal), ARIMA, and Neural Network AutoRegression method (NNA), to determine their effectiveness in accurately predicting real cloud workloads. The evaluation was conducted using Google data and Intel Netbatch logs. It has been discovered that no method can consistently provide correct predictions. [89] conducted simulations of an application using real-time workload patterns, billing models, and various predictive scaling approaches. Their evaluation revealed that no single predictor consistently yielded the best results across all

workload patterns. [90] developed an energy-aware expenditure prediction employing the ARIMA model. This framework assesses the power consumption and resource usage of a VM. [91] presented OnlineElastMan, a proactive elasticity manager for cloud infrastructure as a service (IaaS) resources that undergoes self-training and automatically activates whenever a workload change event occurs. [86] developed a workload prediction module utilising the ARIMA model. This module evaluates future workloads' accuracy by analysing web server trace demands.

The characteristics of Microsoft Azure's VM workload emphasise the potential of leveraging historical data for predicting future behaviour [92]. Their introduced system, resource central (RC), collects and learns from VM telemetry offline and provides online predictions to resource managers. Based on these predictions, Azure's VM schedules. This approach improves server utilization and prevention of resource exhaustion. The conclusion of their work highlights resource central's effectiveness in generating accurate predictions, enabling safe CPU oversubscription, and underscores the broader applicability of exploiting workload characteristics through machine learning for substantial improvements in resource management. Various scholars employed different methods such as ARFIMA, a modified version of the ARIMA model, weight moving average (WMA), MA, and AR for workload prediction [64].
The limitation of statistical prediction methods is the dependency on predetermined assumptions and linear associations between variables. This causes difficulties in handling complex, non-linear data patterns and relationships. Therefore, With the increasing size and diversity of datasets, traditional methods struggle to capture complex relationships and hidden patterns accurately.

2.1.2 Machine learning based model

Machine learning models have demonstrated superior performance as compared to traditional statistical and regression models in various prediction tasks. Studies emphasise the capacity of machine learning methods to analyse and summarise complex hidden patterns in data, which results in more accurate predictions [93, 94, 95, 96]. [97] introduced a method for long-term workload forecasting in the Grid environment, targeting network and CPU load fluctuations across various timeframes, from minutes to over a week. Their algorithm harnesses seasonal load variation for load variance and one-step-ahead predictions, employing a Markov model-based meta-predictor sensitive to late trends. Experimental results demonstrate the effectiveness of their approach, with mean error rates of 6.2% for CPU load and 9.4% for network load in one-step-ahead forecasts. Particularly noteworthy is the utilization of seasonal variation in network-related CPU load

for extended network load predictions. Experiments using real CPU and network load data from a Cisco router confirm the accuracy and practicality of the method, showcasing precise predictions and a low mean error rate. Continuing the exploration of resource allocation optimization in cloud computing, [98] investigated the optimal allocation of cloud computing resources by analysing and forecasting the workload of VMs carried out using actual data center traces. A co-clustering technique to discover groups of VMs that have correlated workload behaviours. Subsequently, it employs Hidden Markov Modeling (HMM) to determine temporal correlations and forecast fluctuations in workload. Their findings indicate an improved understanding of workload patterns at the group level and an increased precision in forecasting workload fluctuations within a cloud-based environment. The suggested method outperforms traditional prediction methods at the individual server level.

Similarly, to make a promising approach for workload prediction in cloud computing environments, [99] introduced an advanced data center prediction model, blending k-means clustering techniques and Extreme Learning Machines (ELMs) to forecast future loads, explicitly focusing on VM requests. The proposed model, tested on real Google traces featuring over 25 million tasks, demonstrates superior performance compared to existing models in its literature. The conclusion emphasises the innovative framework, combining a k-means clustering algorithm with an ELM predictor, effectively estimating the number of VM requests in a CDC. [100] focused on the critical issue of capacity planning in data centers, advocating for a proactive resource management system that integrates workload prediction with efficient resource allocation strategies. Their study aims to assess and compare the effectiveness of various time series prediction models, including AutoRegressive Moving Average (ARMA) variations, exponential smoothing models (Holt-Winters), and spectral estimating approaches. To enhance the performance of ARMA variations, they employ the Kalman filter and Wavelet decomposition techniques. Through analysis of real workload traces from Wikimedia Grid, the Kalman filter-based ARIMA and its seasonal versions outperform other methods, showcasing remarkable forecasting accuracy. These hybrid models achieve mean absolute percentage errors of less than 0.03% for RAM, under 0.8% for CPU, and less than 4% for the network. These models exhibit exceptional performance even in scenarios with limited system information.

Furthermore, the study identifies optimal input sizes for predicting outcomes beyond the sample, providing valuable insights for accurately forecasting workloads in dynamic data center environments. Building upon this research, [101] directed their efforts toward enhancing resource utilization in CDCs through the introduction of a multi-objective genetic algorithm (GA) for dynamic workload prediction of VMs. This innovative GA

considers the CPU and memory utilization of VMs and PMs and energy consumption to forecast resource requirements accurately. Additionally, they introduced a VM placement algorithm to allocate VMs based on GA predictions. Simulation results underscore the superior accuracy of the proposed GA compared to the Grey forecasting model, indicating enhancements in CPU and memory utilization alongside reduced energy consumption across stable and unstable utilization scenarios. The VM placement algorithm optimizes resource utilization and energy conservation, particularly under low utilization conditions. Moreover, it demonstrates increased accuracy with a higher ratio of PMs to VMs due to richer historical data. This study presents a holistic approach for effectively predicting workloads and optimising resources in CDCs, contributing significantly to cloud computing infrastructure management advancements.

[102] evaluated the performance of ARIMA, Support Vector Regression (SVR), Bayesian Ridge Regression (BRR), and Long short term memory (LSTM) in predicting the CPU usage of Google cluster hosts. This investigation, covering single and three-time steps ahead, reveals that BRR achieves the best results, particularly for highly fluctuating and large Google cluster traces. Addressed the limitations of SVR for large datasets, ARIMA for pathological data, the ability of LSTM to model non-linear patterns effectively, and the superior performance of BRR for highly dynamic Google cluster traces. They also introduced the m-gap prediction method for effective task scheduling. A clustering-based workload prediction method is proposed, enhancing accuracy by clustering tasks with similar workload patterns and employing dedicated prediction models for each cluster.

[103] addressed the pressing issue of power consumption in CDCs, stemming from the escalating demand for cloud computing services. Their study forecasts VM power consumption by employing regressive predictive analysis, particularly leveraging the multi-layer perceptron (MLP) regressor as a machine learning technique. This response is necessitated by the growing number of connected devices and the surge in data center power requirements. The study demonstrates that proactive techniques, leveraging historical performance data and various regression models, can accurately predict fluctuations in power usage. The suggested MLP regressor model achieves an impressive accuracy rate of 91%, furnishing cloud managers with a proactive approach to predict and manage VM power usage effectively. As a result, it ensures a reliable environment for consumers amidst the dynamic landscape of cloud computing. The limitation of machine learning methods is their dependence on feature engineering and manual extraction of specific patterns from data. This procedure is time-consuming, requiring a significant amount of effort, and may not fully capture all relationships among enormous, complex data sets. Moreover, traditional machine learning models may encounter difficulties when dealing

with unstructured data, such as images, sounds, or text, where the process of identifying significant characteristics manually is complex [104].

2.1.3 Deep learning based model

Deep learning is increasingly used in prediction models for cloud computing due to its ability to understand complex representations and features in hierarchical structures. Developing this ability is crucial for effectively handling the enormous quantity, various categories, rapid growth, and accuracy of large-scale data characteristic of cloud computing. Therefore, deep learning is considered a more advanced option than typical machine learning techniques for making predictions in cloud computing environments [105, 106, 107, 108]. [109] focused on improving host load prediction in computational grids by employing a neural network predictor, aiming for optimal performance and load balancing. This proposed neural network showcases exceptional accuracy, consistently surpassing linear and tendency-based models in experimental assessments, with mean prediction errors reduced by up to 79%. The performance of the 20:10:1 network is particularly noteworthy, which achieves superior results while requiring minimal training time, enabling tens of thousands of accurate predictions within seconds. Due to their low overhead and efficient prediction capabilities, this underscores the feasibility of employing neural predictors in dynamic computing environments such as computational grids and clouds.

[110] introduced a novel framework that integrates load demand prediction with stochastic state transition models to optimize cloud resource allocation. This research evaluates the performance of neural networks and autoregressive linear prediction algorithms in forecasting load in CDCs. The findings indicate that both models excel in predicting network loads, with the Linear Predictor yielding the most precise results. In the exploration of resource allocation efficiency in cloud computing, [111] introduced a neural network model coupled with a learning algorithm to predict cloud server workloads. This addresses the critical challenge of ensuring optimal resource allocation. By integrating a resource manager with a corresponding allocation algorithm, the model empowers cloud service providers to anticipate future server workloads, thereby facilitating efficient resource allocation and load balancing. Experimental findings underscore the effectiveness of this approach in averting resource inadequacies. Moreover, the results emphasise the pivotal role of the neural network model in aiding cloud providers in anticipating and managing resource requirements with greater accuracy and efficiency compared to traditional regression methods. [112] tackled the complexities of effective resource management in cloud computing, specifically focusing on host load prediction. Their approach involves

utilising an autoencoder as the pre-recurrent feature layer of echo state networks to forecast host load in future intervals based on Google cluster usage data. Experimental results validate the effectiveness of this method, showcasing superior performance as compared to the state-of-the-art approaches, particularly in CPU load prediction accuracy. The study underscores the innovation of this approach, which combines echo state networks with autoencoder neural networks to enhance input representations and improve load prediction accuracy, as evidenced by comparisons with other advanced techniques.

[113] tackled the challenge of maximising benefits and minimising computational costs in a cloud cluster by introducing a workload prediction approach utilising recurrent neural networks (RNN). This study incorporates CPU and RAM resource types and employs an orthogonal experimental design (OED) to identify optimal parameter sets for RNN. The study concludes by underscoring the utilization of Google Cloud CPU and RAM traces for training RNN models, emphasising the identification of optimal parameter combinations through OED analysis. Notably, the RNN-based method, with the smallest average mean squared error (MSE) of 2.761×10^{-5}, proves highly effective in accurately predicting workload states, particularly in resolving time sequences. [114] focused on achieving precise host load prediction in cloud computing, which is crucial for enhancing resource allocation and utilization. This proposed method harnesses the power of the LSTM) model to forecast the average load over successive time intervals and the actual load several steps ahead in both cloud and traditional grid contexts. An evaluation conducted on actual load traces obtained from Google data centers and a conventional distributed system underscores the exceptional flexibility and state-of-the-art performance of the LSTM model. It outperforms conventional approaches that may excel in grid environments but falter in cloud settings. The findings underscore the efficiency of the succinct yet robust LSTM-based method for forecasting host load across diverse computing environments.

[115] delved into time series forecasting of CPU usage, leveraging the LSTM Network and contrasting its performance with traditional ARIMA models. The LSTM model exhibits a 17-23% forecasting error range, notably surpassing the ARIMA model's 37-42% range. These findings highlight the LSTM model's consistency and efficacy in predicting nonlinear data patterns, showcasing its potential for enhancing resource usage forecasting in data center environments. Similarly, [116] introduced the CPW-EAMC framework, revolutionising multi-dimensional resource utilization forecasting in data centers beyond traditional unidimensional predictions in cloud computing. This innovative model integrates a noise reduction algorithm (CPW) for precise feature extraction and a neural network (EAMC) for accurate multi-dimensional predictions. The proposed CMES evaluation standard comprehensively assesses the model's performance. Experimental results

demonstrate a notable 2% to 17% improvement over other popular approaches. The conclusion highlights the critical importance of predicting physical machine resource utilization for efficient scheduling decisions, highlighting the novel contributions of the noise reduction algorithm and the multi-dimensional prediction model [117].

2.1.4 Hybrid model

Hybrid models are a combination or amalgamation of many statistical, machine learning and deep learning models or methodologies aimed at utilizing the strengths of each component and obtaining enhanced performance or accuracy in addressing a specific problem. These models are created to address the limitations of individual models by employing the complementary properties of different techniques. The concept is to integrate the advantages of many approaches to enhance the ability to make accurate predictions, increase adaptability, or optimize efficiency. Therefore, these models demonstrated superior performance compared to other employed approaches in the field of cloud computing.
[118] introduced and assessed a hybrid model for workload prediction in dynamic grid environments. This model merges an autoregressive model with confidence interval estimations and integrates signal processing filters to enhance prediction accuracy. Experimental findings from a real grid environment reveal the superior predictive capability of the proposed model in forecasting load several steps ahead, outperforming traditional AR models with notably fewer prediction errors. Particularly noteworthy is the hybrid model's effectiveness in predicting up to 50 minutes in advance, presenting a satisfactory interval length for task scheduling. This study emphasises the significance of confidence interval estimates in capturing load variability and furnishing valuable insights to computational grid task schedulers. Moreover, noise elimination techniques like the Kalman filter and Savitzky-Golay filter substantially enhance prediction accuracy, achieving the effectiveness of the model in striking a balance between accuracy and confidence interval length for workload prediction in grid environments.

[119] tackled precisely predicting host load in dynamic cloud computing systems to optimize resource utilization. This approach blends Phase Space Reconstruction (PSR) with the Evolutionary Algorithm-based Group Method of Data Handling (EA-GMDH). Evaluation using real-world load traces from a traditional distributed system and a Google data center demonstrates superior prediction performance compared to state-of-the-art methods. This method surpasses traditional approaches, ensuring satisfactory performance in traditional distributed systems and cloud computing environments. Similarly, [120] addressed the crucial challenge of accurately forecasting resource demands in Infrastructure as a Service (IaaS) cloud environments to dynamically allocate resources

effectively for users. The technique ESFCFNN (Ensemble model and Subtractive-Fuzzy Clustering based Fuzzy Neural Network) employs an ensemble model and subtractive-fuzzy clustering to achieve self-adaptive resource prediction. This architecture integrates user preferences analysis, base prediction models, and a fuzzy neural network with self-adjusting learning rates and momentum weights. [121] introduced the fuzzy-subtractive clustering algorithm that optimizes the convergence performance of the FNN. Experimental results validate the accuracy and effectiveness of the proposed method in predicting resource demands, leading to enhanced performance in resource provisioning and utilization within the cloud environment compared to the exponential moving average (EMA), second Moving Average (SMA) model, and ARM.

[122] implemented statistical and machine learning methods under the Advanced Model for Efficient Workload Prediction in the Cloud (AME-WPC) to address the challenges of accurately predicting cloud application workloads. They proposed domain-specific database extensions to enhance learning capabilities, considering factors influencing workload. The Two-phase pattern matching method (TPM) was employed to identify workload patterns based on value and fluctuation. Workload prediction was addressed through classification and regression and validated with random forest (RF) on basic and extended training data. Evaluation of the AuverGrid workload data series, including the KNN method, demonstrated that TPM and RF with extended datasets significantly improved prediction accuracy over time. This approach effectively enabled enhanced resource management, optimal service provisions, reduced operational costs, and a more stable cloud environment.

[123] proposed an approach that categorises workloads and assigns different prediction models based on workload features, effectively framing workload classification as a 0–1 programming problem to address the crucial need for accurate workload prediction in service clouds to enable auto-scaling resource management. The optimization algorithm maximizes prediction precision, enhancing accuracy compared to single-model methods, particularly in platform cumulative absolute prediction error. Their significant contribution lies in introducing a categorical prediction approach and establishing an optimal method for classifying workloads to allocate adaptive prediction models. They utilize LR and SVR models to predict workload with the intent to enhance the comprehensiveness of the proposed approach.

[124] focused on efficient resource provisioning in dynamic cloud computing environments, where fluctuating user demands adaptive strategies for optimal resource utilization. They introduced a prediction model based on linear regression to forecast resource usage, leveraging function points computed from users' requests. Integrating an artificial neural net-

work enhances prediction accuracy and facilitates a proactive approach to virtual machine allocation or release. The resource pool manager deployed an efficient load-balancing algorithm to optimize cloud usage cost and distribute the load evenly across providers. Integrating LR with ANN turns out to be effective in achieving adaptive resource provisioning, which helps CSPs in optimizing resource allocation and cost efficiency. The approach is well-suited for timely virtual machine allocation or release decisions in response to dynamic resource requirements. [125] implemented a stacking structure that integrates the RNN and Autoencoder, amalgamating multiple prediction algorithms. Experimental assessments across various datasets confirmed the model's exceptional performance, achieving a lower average normalised root mean squared error (NRMSE) compared to a fixed weighted optimal combination value. Moreover, it improved from 7.43% to 12.45% compared to individual component algorithms. The recurrent neural network (RNN) leverages the gated recurrent unit (GRU), which is pivotal in integrating algorithms, while the Autoencoder enhances the representation layer.

The dynamic resource scaling and power consumption challenges in cloud computing by developing a precise workload prediction model tackled [80]. Leveraging a neural network and a self-adaptive differential evolution algorithm, their model showcased significant advancements over existing methods, achieving error reductions of up to 168 times compared to a backpropagation-based model. Experimental evaluations conducted on HTTP traces from NASA and Saskatchewan servers across different prediction intervals demonstrated remarkable accuracy, with prediction errors minimised to 0.013 and 0.001 for NASA and Saskatchewan, respectively. The proposed approach, utilising evolutionary techniques, outperformed traditional gradient-based learning algorithms, requiring fewer iterations (less than 80) for convergence compared to the 250 iterations needed by the backpropagation model.

[126] addressed the challenge of timely and accurate resource provision in cloud computing, exacerbated by ever-changing and fluctuating customer demands. Their EEMD-ARIMA approach utilizes ensemble empirical mode decomposition (EEMD) to dissect non-stationary resource needs, leading to improved forecast accuracy compared to the conventional ARIMA model. The findings underscore the significance of proactive resource allocation amidst cloud resource requirements' unpredictable and fluctuating nature. Experimental results validate the efficacy of the EEMD-ARIMA method, demonstrating superior prediction accuracy in short-term cloud resource demand forecasts relative to the traditional ARIMA model. In the following year, a hybrid approach was introduced, EEMD-RT-ARIMA, which merges ensemble empirical mode decomposition (EEMD), Runs Test (RT), and autoregressive integrated moving average (ARIMA) for

real-time host utilization prediction in cloud computing. Their method effectively forecasts dynamic resource scheduling by decomposing non-stationary host utilization into stable Intrinsic Mode Function (IMF) and residual components, thereby enhancing prediction accuracy. Selective and efficient IMF components are reconstructed to minimise prediction time and error accumulation. The overall predictions are then generated by combining these components using the ARIMA method. Experiments conducted on real host utilization data from a cloud platform validate the cost-effectiveness and suitability of the proposed approach for short-term host utilization prediction, demonstrating the superior performance compared to the ARIMA model and the EEMD-ARIMA method in terms of error, effectiveness, and time-cost analysis [127].

[128] tackled the challenge of accurately modelling server load in dynamic and intricate environments characterised by scaling Cloud workloads and heterogeneous infrastructure. Their proposed pCNN-LSTM hybrid prediction approach integrates 1-dimensional convolutional neural networks (1D CNN) with LSTM networks to forecast CPU utilization on Cloud servers across multiple consecutive time steps. The model effectively learns and predicts intricate, noisy variations in host CPU usage. Leveraging Google cluster trace, Alibaba trace, and Bitbrains data, pCNN-LSTM outperforms alternative models, yielding improvements of up to 15%, 13%, and 16% in host load prediction, showcasing its multi-scale learning capability. The study underscores the significance of load forecasting in facilitating effective capacity planning and provisioning. By leveraging 1D CNN for local pattern extraction and LSTM for learning temporal dependencies, pCNN-LSTM surpasses other deep learning models, demonstrating superior generalisation capability and modelling skills essential for informed Cloud server capacity management.

[129] merges variational mode decomposition (VMD) with temporal convolutional network (TCN) to address the critical aspect of workload prediction in cloud computing, highlighting its pivotal role in enhancing resource management and impacting various factors such as quality of service, elasticity, SLA, and power consumption. Their model outperforms existing deep learning models in accuracy, showcasing its state-of-the-art performance. The innovative approach employs VMD to decompose workload into IMFs, mitigating variance impact and stabilising prediction accuracy. TCN then predicts IMFs individually, and their summation provides the final predicted workload demands. Comparative analyses with existing methods, including RNN and TCN, achieve the superiority of the proposed model in real-world cloud computing workload prediction.

[125] introduced BHyPreC, a novel hybrid RNN model tailored for forecasting CPU utilization workload in cloud VMs. Combining Bidirectional Long Short-Term Memory (Bi-LSTM) with layered Long Short-Term Memory (LSTM) and Gated Recurrent Unit

(GRU), BHyPreC achieves remarkable precision compared to alternative statistical models like ARIMA, LSTM, GRU, and Bi-LSTM, excelling in both short-term and long-term workload prediction. The study highlights the importance of accurate forecasting in optimising VM migration, resource allocation, and job scheduling tasks. By leveraging deep learning techniques, BHyPreC effectively addresses the non-linearity of time series data and exhibits resilience in handling sudden spikes and drops in workload patterns. The findings suggest its potential applicability in resource allocation, scheduling, load balancing, and VM migration tasks, providing valuable insights into future CPU consumption data. While regression approaches and recurrent neural networks currently dominate cloud workload prediction, they struggle to capture long-term workload variance. Therefore, [130] proposed an efficient supervised learning-based Deep Neural Network (esDNN) approach that overcomes this limitation by employing a sliding window to convert multivariate data into a supervised learning time series. This facilitates accurate prediction using a revised GRU. Experimental results, based on traces from Alibaba and Google CDCs, showcase esDNN's superior accuracy and efficiency, reducing mean square errors by 15% compared to RNN, GRU, and Bi-LSTM models. Moreover, the application of esDNN for auto-scaling further optimizes resource usage, demonstrating its potential for cost savings in CDCs. The study concludes by emphasising the opportunities for resource provisioning optimization in cloud computing through the proposed deep learning-based approach, which effectively addresses high-dimensionality challenges and achieves high prediction accuracy for varied workloads.

[131] focused on predicting resource utilization to mitigate over and under-provisioning issues directly affecting power consumption and operational costs. Given the oversight of resource correlations in existing methods, the study introduces a Functional Link Neural Network (FLNN) combined with a hybrid Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) approach for multi-resource utilization prediction. Evaluation based on Google cluster traces reveals the model's superiority over conventional FLNN, FLGANN, and FLPSONN, offering higher accuracy. The conclusion underscores the significance of multi-resource utilization prediction in cloud computing for dynamic scaling, energy conservation, and cost-effectiveness. Despite the challenges of rapid fluctuations in resource utilization, the hybrid GA-PSO and FLNN model demonstrates remarkable accuracy in prediction.

To summarise, the literature analysis on workload prediction in cloud computing has emphasised the significance of precise resource consumption forecasting for effective resource management and optimization in dynamic environments. Various techniques and models have been proposed, ranging from statistical methods to advanced machine learning

algorithms. These studies have shown the importance of accurate workload prediction in optimising resource allocation, reducing energy usage, improving quality of service, and decreasing operational expenses. Furthermore, evaluating several prediction methodologies has yielded valuable observations regarding the strengths and limitations of each technique. Overall, the research in accurate resource usage prediction underscores the ongoing efforts to develop robust and effective workload prediction models that meet cloud computing systems' evolving dynamic needs.

2.2 Virtual Machine Allocation (VMA)

In cloud computing, resource allocation techniques are essential for ensuring the efficient utilization of resources to meet consumer demands. These techniques encompass a range of methodologies aimed at fulfilling consumer requirements effectively. [132] introduced five allocation strategies: random allocation, sequence allocation and greedy technique. They include three varieties of sequence allocation, i.e., full sequence allocation, sequence allocation based on task execution time sorting, and sequence allocation based on virtual machine execution speed sorting. The objective is to minimise the execution time of all tasks. The simulations and experimental studies are performed utilising the Cloudsim platform. The results demonstrate that the greedy method outperforms execution time, whereas the random technique is the least efficient. The three sequence techniques demonstrate a moderate level of performance. [133] employed the Krill Herd algorithm to minimise the data center's energy consumption by optimising the aggregation of virtual machines and managing server shutdowns during periods of idleness while ensuring service quality is maintained. The proposed method compared with a genetic and modified best-fit decrease (MBFD) algorithm, where Interquartile range (IQR) and median absolute deviation (MAD) were used for overloaded host detection, random selection to select the migrated machine and minimal migration time algorithms for virtual machine placement. The results conclude that their proposed algorithm reduced energy consumption by 35% and 17%.

Similarly, for task allocation to virtual machines, a hybrid approach incorporating a hierarchical process and a scheduler to manage and prioritise requested tasks effectively. They modified the Bandwidth-Aware divisible Task (BAT) scheduling model with the Bar system model. This proposed hybrid model performance is evaluated through a comparison with BAT and Ant Colony Optimization (ACO). Their results demonstrated that the hybrid model outperforms resource usage and bandwidth allocation [134] suggested. The initial placement of virtual machines, alongside an innovative virtual machine selection algorithm, serves to optimize the existing allocation based on factors such as memory us-

age, bandwidth utilization, and virtual machine size; [135] proposed a Modified Discrete Particle Swarm Optimization (DPSO). The results stated that integrating the suggested allocation and selection algorithms reduces energy consumption significantly. [136] formulated the VM placement problem as a constraint optimization problem and solved it using a GA. They concluded that their approach saves 8% of energy compared to the First Fit Decrease (FFD) algorithm and a 66% reduction in execution time from standard GA. [137] proposed an evolutionary approach to allocating VMs capable of optimizing the energy efficiency of a CDC while integrating an increased number of reserved VMs. This approach quickly finds an effective allocation mechanism for a batch of reserved VMs and consolidates more VMs using fewer PMs. The solution can increase profit by 24% and save energy by 41%, surpassing FFD and MBFD algorithms. [138] employed a Hybrid Genetic Cat Swarm Optimization algorithm with multi-objectives to save energy and reduce resource wastage. They further consolidate VMs by migrating and shutting down the idle machine to minimise the number of active machines. In another work [139], they allocated VMs using a GA and PSO approach hybrid. This time, consider SLA, energy consumption, and resource wastage metrics.

[70] developed an optimal VMA model to optimize VM usage for providers while reducing the time spent on user tasks. They introduced an enhanced differential evolution technique to address the optimization challenge when confronted with a given set of user tasks. Their experimental results demonstrate that their approach provides faster convergence than differential evolution. Furthermore, this method can potentially achieve resource allocation plans with a shorter duration than the Round-Robin and Min-Min methods. [140] provided a novel allocation approach that prioritises thermal considerations in CDCs. Their approach analyses host temperature factors, significantly decreasing energy usage and migration incidents while simultaneously upholding the SLAs within CDCs. [141] used a novel energy-efficient multi-resource allocation model and PSO to find the optimal resource utilization-energy consumption equilibrium using the total euclidean distance fitness function. Their approach saves energy, optimizes system resources in CDCs compared to MBFD, and modifies the best-fit heuristic algorithm.

[142] used Google's SLA agreement to calculate penalties based on violation severity to evaluate QoS as a cost factor. They handled the issue by considering the network switch and host power usage. They evaluated three main costs: host power consumption, network switch power consumption, and penalty cost. A prediction-based and power-aware virtual machine allocation technique was provided for three-tier CDCs. Their simulations show that the ensemble prediction-based VM allocation approach beats other benchmark algorithms. [143] broadly categorised VMA techniques into strategic allocation, which

adapts to the dynamic nature of consumer demands. Target resource allocation, which concentrates on fulfilling specific resource requests. Auction-based allocation, where resources are allocated through a bidding process. Optimization-driven allocation, which aims to maximise resource utilization. Scheduling techniques, which prioritise tasks to enhance overall performance. Power-aware allocation strategies, which optimize resource allocation while minimising power consumption.

2.3 Virtual Machine Consolidation (VMC)

Dynamic VMC encompasses a structured process that is divided into four essential steps. (1) Identifying overloaded or hotspot PMs inside the data center infrastructure. The detection technique is essential for recognising PMs undergoing excessive resource use, which may result in performance degradation or system instability. (2) Detect the underloaded PMs or cold spots in the environment. Underutilized PMs allow migrating or distributing all the assigned VMs to other active PMs so that they can be switched off and reduce power consumption. (3) Selection of appropriate VMs for migration from the overloaded physical machines. The selection procedure involves considering several factors, such as the resource demands of VMs, the overhead of migration, and the overall optimization target of the VM selection consolidation algorithm. (4) Selection of appropriate PMs to place the migrated VMs. This step is called VM placement. To achieve the best VM placement and maximise resource usage in the data center, it is crucial to carefully examine resource constraints, network structure, and performance requirements during the placement process.
Figure 2.2 displays all the methods used for these four steps of VMC employed in the literature review.

2.3.1 Host Overloaded Detection Algorithm

When a host experiences overload, migrating one or more VMs from the host becomes necessary to address the overload condition. The literature on DVMC algorithms shows that upper and lower threshold values are used in most situations to determine if a PM is overloaded or under-loaded. This determination is based on the PM's resource usage ratio. The critical factor is the evaluation of utilization in relation to these threshold values, which can be either static or adaptable.

As shown in [144], if the resource usage ratio exceeds the upper threshold value, the PM is said to be overloaded or over-utilized. Therefore, VMs are transferred from the PM until the resource use decreases below the higher threshold. It is crucial to monitor the

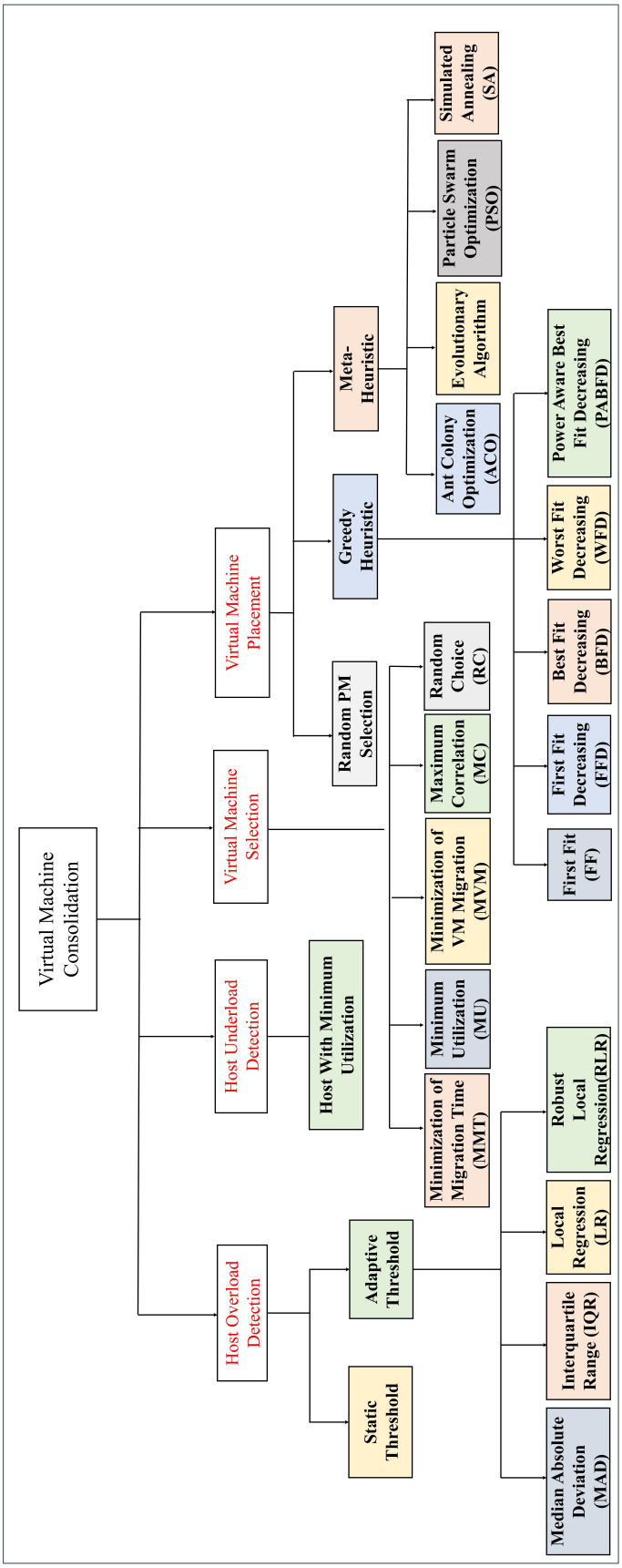


Figure 2.2: Classification of Virtual machine consolidation techniques

R-value since it indicates the possibility of QoS degradation or violation of SLA. This is due to the increased demand for resources from the hosted VMs. On the other hand, if R drops below the lower utilization criterion, the PM is identified as being underloaded or under-utilized. In such instances, the search is conducted for suitable destination PMs where VMs from the PM can be transferred, enabling the PM to go into sleep mode. Threshold-based DVMC algorithms can be classified into two distinct categories based on the type of thresholds used to determine if a PM is overloaded or underloaded: Static threshold and Adaptive threshold-based host overloaded detection.

Static threshold methods (STDVMC) utilize fixed higher and lower thresholds to classify a machine as either overloaded or underloaded. The term "static threshold" describes thresholds that remain constant across time. The STDVMC algorithms used in these studies [144, 145, 146]. In [146], the authors consider 100% CPU utilization as the upper threshold and 50% CPU utilization as the lower threshold. If the CPU usage of a PM turns out to be 50%, then that PM is classified as underutilized, and all the VMs are transferred from that PM to other PMs. If the combined resource demand of all the VMs hosted on a certain PM exceeds the CPU capacity of that PM, it is classified as an overloaded PM. In such cases, the select some VMs to be transferred from the overloaded PM to other PMs. In contrast, Adaptive threshold (ATDVMC) methods dynamically adjust the threshold values that determine whether a PM is considered overloaded or under-loaded based on the PM's resource utilization ratio. Basically, the threshold value adjusts according to variations in resource consumption as used in [147, 148, 149, 150].

In [151] proposed four policies, which can be classified into two groups. The initial category comprises adaptive use threshold-based algorithms that consist of two policies: MAD and IQR. These rules enable automatic adjustment of usage criteria by analysing previous data collected throughout the lifespan of the virtual machines. The second kind of algorithm is regression-based, including two policies: LR and robust local regression (LRR). These are dependent upon predicting future CPU usage. They have superior performance in predicting host overloading but possess a higher degree of complexity. This category is influenced by the presence of outliers and does not fully depict the behaviour of the bulk of the data. A technique for detecting host overload based on temperature was proposed [152]. The ideal server is selected among all servers in a data center based on its largest possible operations per second. A CPU utilization of up to 10% is required to execute VM live migration successfully. Hence, a CPU usage of 90% can be considered as the maximum limit, and its temperature is measured when the CPU is operating at 90% capacity. This temperature is referred to as the threshold temperature. The selection of the overloaded server is solely determined by temperature,

without considering any other resources.

The authors in [153] propose a novel method for calculating a lower threshold that takes into account the total usage of the data center and the utilization of all active hosts in it. Using the comprehensive workload analysis of the data center, it forecasts the maximum potential number of hosts that could be left unused. The estimated maximum number of hosts that can be removed is used to determine the lower threshold. Experimental results demonstrated that the utilization-aware host (HUA) approach is effective in identifying under-loaded hosts and subsequently freeing up a more substantial number of hosts. This leads to reduced energy consumption while still maintaining compliance with SLAs.

In [154], the authors proposed an approach based on prediction. The Markov chain technique is utilized to predict the CPU usage of a host. If the current CPU usage of a host is beyond the upper level, it is deemed over-utilized. Conversely, it is considered underutilized if CPU usage falls below the lower threshold. Alternatively, a host is deemed to be over-utilized or under-utilized solely if it is expected to remain in that state in the immediate future. [155] proposed an overload detection system that considers the elevated threshold and anticipated future demand of the host CPU utilization. The MAD technique is used to compute the higher threshold. They employ single exponential smoothing and double exponential smoothing techniques to forecast future demand. This method eliminates current and potential hosts that are being excessively used from the list of probable destination hosts. This enables us to avoid unnecessary virtual machine migrations.

Multiple regression was suggested by [156] to solve VMC problems. Due to its reliance on anticipating host utilization, LR is better for host overload detection in dynamic cloud systems. Multiple parameters are employed in regression analysis to estimate host utilization. This method accurately assesses host usage. MRHOD and HLRHOD perform better than single-element algorithms in energy usage. Several researchers propose using the ARIMA model to forecast host workload to detect overload [157]. This method prevents SLA violations and host overload by detecting host overload. The researchers developed Overloaded Server Detection utilising multi-user prediction (OHD-MUP) to identify overloaded servers [158].

In a CDC with numerous machines and VM workloads, making VM migration decisions based on a fixed threshold may lead to hundreds or thousands of hot spots over time. proposed OHDMUP uses present and expected usage to reduce hot spots to essentials. In [159] researchers introduced a Dynamic Voltage and Frequency Scaling (DVFS) and VM migration-based technique for consolidating VMs to improve energy efficiency. The proposed workload-based dynamic threshold (WBDTH) method identifies overloaded or

underloaded machines. VM migration was employed to equilibrate the underutilized and overburdened PMs. The DVFS is applied to the underloaded host where the job is near completion; shifting them at this time leads to a performance decrease.
In essence, most current detection techniques rely on monitoring the system's current CPU utilization. VM migration is initiated promptly whenever a host is deemed overcrowded, although this may not always be the optimal solution. It's important to note that each VM migration incurs a certain level of performance degradation, consequently elevating the SLA violation rate. It's widely acknowledged that the timing of initiating VM migration is closely linked to the cost incurred by the increase in the SLA violation rate.

2.3.2 Host Underloaded Detection Algorithm

Host underutilization, also known as cold spot, occurs when specific servers or hosts continually operate significantly below the considering underutilization threshold capacity. Host overload detection methods assist in minimising SLA violations during the VM consolidation process, while host underload detection methods aid in reducing energy consumption by data center servers. Consequently, numerous researchers have proposed various solutions for host underload detection algorithms in the literature, recognising their critical role in decreasing energy usage. This discussion delves into the diverse works authors propose for host underload detection methods [160].

The system finds the compute host with the lowest utilization and transfers all VMs from it to other servers to avoid overloading them. VMs are prepared for migration to destination hosts if possible. After migrations, the source host sleeps to conserve energy. If the source host cannot move all VMs to another host, it will continue functioning. This is repeated for all non-overloaded hosts [151]. The researchers in [161] devised a modified and underutilized technique for detecting hosts, known as MUUHD (Modified Underused Host Detection). An extra metric value representing the number of VMs on a specific server is also employed to choose the under-loaded hosts. The evaluation will consider the host with the smallest CPU utilization or the server with the fewest VMs and no VMs being migrated to or from the host.
The authors propose a host uderloading detection approach in [162], which is based on a recommended robust SLR model. The purpose of this method is to minimise power consumption and prevent SLA breaches. This approach implemented an adaptive lower utilization threshold using the IQR to detect underloading in the host. The under-threshold host method is proposed [163]. The Pearson Host Overload Load Detection Method (PHOD) calculates the lower threshold. The PHOD method utilizes the under-

threshold host algorithm to calculate the lower threshold, known as LowerT, to identify hosts that are experiencing low load after a global change in host load. The method proposed by the authors of [154] is called Prediction-Based Underutilized Host Detection (PBUHD). The Markov chain model is utilized to predict the CPU usage of a host. The decision that a host is over-loaded or under-loaded is made by comparing its current CPU usage to the predefined thresholds for higher and lower CPU utilization.

Instead, a host is regarded as over or under-loaded only if it is projected to stay so in the near future. [152] developed a technique for detecting when a host is underloaded based on temperature. Assuming that there are no alternatives available in the remaining two categories. In the above example, the algorithm selects the least busy server first from the low-performance machines, then from the medium-performance servers, and finally from the high-performance machines. The objective of this strategy is to maximize the amount of time that low- and medium-performance machines spend in a sleep state, as they consume more energy than high-performance servers. The technique outlined by the authors in [164] pertains to the identification of a server experiencing excessive workload. They employ total data center usage to identify the maximum expected quantity of servers that are not being fully utilized in order to detect underloaded servers. Subsequently, that number is utilized to compute a lower threshold, which is calculated periodically and is dependent on the upper threshold.

The authors propose the identification of underloaded hosts, as mentioned in [157]. This approach detects underloaded candidate hosts that meet two criteria: (1) the server has not performed a VM migration, and (2) the server has not experienced overload. Subsequently, the algorithm evaluates the potential hosts and selects the host with the greatest head value as the underloaded server. The algorithm's time complexity is O(n), where n is the amount active hosts. The authors propose a Workload-Based Dynamic Threshold (WBDTH) [159]. In order to calculate the threshold, the tasks of predicting the workload and determining the magnitude of the workload were performed. To assess if the machine is underloaded, compare the current CPU use to the lower threshold value. If the number of values is less than the threshold value, the host is underloaded, or it is balanced.
In summary, the majority of current underload detection approaches rely on the system's present CPU use. Identifying underutilized hosts is crucial for optimising energy usage and resource utilization. Current approaches employ either a fixed or adaptive method to calculate lower thresholds. However, an efficient algorithm is still required to determine when a host is underloaded, which would result in reduced energy consumption and enhanced resource efficiency in data centers.

2.3.3 Virtual machine selection algorithms

After migrating VMs from a specific host, a particular VM selection method should be used to select one or more VMs among the entire collection of VMs currently running on the host. The task is to determine the most ideal subset of VMs to transfer in order to achieve the optimal system reorganisation.

The authors of [165] proposed the growth potential aware VM selection (GPS) approach, which involves selecting virtual machines for migration from overloaded servers. If the average utilization exceeds the present utilization, the VM will probably overload the server in the future. Therefore, the VMs with the highest average value are selected for migration to a server with a moderate load. A VM is randomly chosen from the source PMs [144, 166]. This can also be called a random selection and choosing a VM in O (1) time [151]. [151] define three VM selection criteria. Unlike other server VMs, minimal migration time (MMT) chooses the shortest migration time. The Random Choice (RC) algorithm uses a uniformly distributed discrete random variable to move a VM. However, the Maximum Correlation (MC) approach shows that an oversubscribed server becomes overloaded as the correlation across application resources increases. Review and relocate the VMs with the highest CPU use correlation first. Different correlation coefficients were used to estimate correlation.

The minimum number of VMs that need to be transferred to reduce the present resource consumption of a PM is below the upper usage threshold. The MVM algorithm, as introduced by [144], initially arranges the VMs into descending order based on their CPU consumption. and selects the VM that meets the two criteria: Firstly, the CPU utilization of the VM should exceed the difference between the current overall CPU utilization of the host and the upper threshold for utilization. Secondly, the chosen VM has the smallest difference between the upper threshold and the updated utilization compared to all the other VMs. If a VM meeting the specified criteria is not discovered, the VM with the greatest utilization is chosen, and the procedure continues until the new utilization falls below the higher utilization threshold. [157] proposed a VM selection approach that assesses the decrease in CPU capacity resulting from VM migration and compares it to the host overload. Their comparison evaluates whether VMs should be migrated off the overloaded host. When moving a VM, select the one with the minimal reduction in CPU capacity. This method prevents unnecessary VM migration while reducing CPU utilization and improving QoS satisfaction.

[167] proposed energy-aware dynamic VM selection approaches which combine VMs from hot spot or cold spot hosts, aiming to decrease total energy consumption and optimize

QoS. The objective of this strategy is to optimize energy utilization while minimising instances of SLA violations. [168] presented three VM selection methods, which are as follows. Policy for Selecting the CPU with the Highest Priority (HCPS). Following the HCPS policy, VM with a higher CPU usage is prioritised for migration. Lowest CPU Priority Selection (LCPS) is an approach where VMs with lower CPU utilization are given more priority throughout the migration process. The Random CPU Selection (RCS) policy selects the VM for migration in a random manner. The virtual machine with the lowest ratio of real resource utilization to its first estimated resource demand is picked. Several authors [169, 148] have labelled this approach as Minimum usage (MU) while focusing solely on resource usage and disregarding resource demand. The algorithm has a time complexity of O(n) in the asymptotic sense. The VM with the least migration time is chosen for migration. The migration time is calculated by dividing the amount of RAM the VM uses by the available spare network bandwidth of the hosting PM [147]. This approach named as Minimization of Migration Time (MMT) has a time complexity of O(n).

VMs with the strongest relationship in resource use with other VMs are chosen [151]. The multiple association Coefficient, as suggested by [170], is employed to find the association between the resource utilization of VMs. Using the RC may facilitate the identification of the globally best solution. However, if the problem space is predetermined, such as selecting the source PM based on the heuristic that the PM with the highest or lowest resource consumption will be chosen and then randomly selecting a VM from that PM, the RC method may not yield the globally optimal solution. The other greedy heuristics, including MVM, HPG, MMT, and MC, yield the optimal solution within a local context. RC can select a solution based on likelihood, even if it is not the best solution inside a specific region. Furthermore, VMs encounter a decline in QoS while undergoing the migration process. Hence, choosing the VM with the shortest MMT will help in minimizing SLA violations [171]. On the other hand, RC may select for a virtual machine with a longer migration period. As a result, the rate of SLA violations may be larger for RC compared to the other approaches. The methodology known as UOVM, presented by the authors of [163], is a used optimization VM selection method. The primary principle underlying this strategy is to select a VM from a heavily burdened host with the greatest mean deviation utilization (MDU) and lowest migration delay (MD) to minimise the number of migrations, energy usage, and SLA breaches.

[172] proposed the utilization of modified VM Selection. Migration control has been incorporated into the enhanced VM selection procedure. We will not move a VM if it constantly utilizes a significant amount of server resources for a prolonged duration. When

migrating a VM, there is a strong probability that the VM will overwhelm the destination server, leading to the need for another migration. This will result in increased network traffic, violation of SLA for migration, and greater energy use due to a failed effort at VMC. [173] proposed a Fuzzy VM selection approach with a migration control technique. A VM will be eligible for transfer with the least CPU utilization and the greatest fuzzy output value. If the total CPU usage of all VMs on a heavily burdened host surpasses the migration control threshold, the machine with the greatest fuzzy output value is relocated.

In summary, the existing VM selection approaches were developed to accurately identify the VMs that must be migrated from overloaded PM. The current solutions have several constraints, such as relying only on CPU and RAM usage for decision-making, lacking proactive criteria for controlling VM migration, and not considering the uneven workload distribution on hosts. It causes service interruption, increased network resource usage, and decreased performance. Therefore, advanced techniques are required to take into consideration different resources, manage the migration of VMs, and address workload imbalances during decision-making.

Table 2.1 presents different resource management strategies categorized by their primary focus, including resource prediction, virtual machine allocation (VMA), overloaded detection, VM selection (VMS), and VM placement (VMP). The table also details the techniques employed to achieve these objectives, the datasets utilized, evaluation metrics, and the parameters considered, such as resource usage, energy consumption (EC), number of PMs, and VM migrations.

2.3.4 Virtual machine placement algorithm

After the process of VM selection is finished, the next step is to determine the placement of the VM. Choosing the appropriate destination PM is a crucial factor in improving the energy efficiency of the CDC. The objective is to decrease the overall quantity of active PMs while guaranteeing that no PM's resource limitations have been violated throughout VM migration. The selection of destination PM is a difficult task as it is classified as an NP-Hard problem, which leads to the formulation of multiple heuristic and meta-heuristic algorithms. These algorithms aim to optimize VMC of VMs and reduce energy consumption.

Within the context of First Fit (FF), VMs are arranged in a specific order, and for every VM, the first suitable PM from an ordered set of PMs is chosen. Simply, the search for the target PM always begins with the first PM for each VM. The search for an appropriate

Table 2.1: Comparative analysis of existing survey articles on Resource Management in CDCs.

Reference	Resource Prediction	VMA	Overloaded Detection	VMS	VMP	Techniques used	Dataset	Evaluation metrics	Parameters: Resource usage, Energy Consumption (EC), Number of PMs, VM migrations
[129]	✓	×	×	×	×	Variational Mode Decomposition with Temporal Convolutional Network (VMD-TCN)	Alibaba trace	Accuracy	Resource usage
[98]	✓	×	×	×	×	Hidden Markov Modeling	Real trace	Accuracy	Resource usage
[123]	✓	×	×	×	×	Linear Regression (LR), Support Vector Regression (SVR)	Self-collected	Mean Absolute Percentage Error (MAPE)	Resource usage
[103]	✓	×	×	×	×	Multi-layer Perceptron (MLP)	Azure VM workload	Root Mean Square Error (RMSE), Number of VM	EC
[101]	✓	✓	×	×	✓	Genetic Algorithm (GA)	Self-collected	Prediction error	Resource usage, EC
[84]	✓	×	×	×	×	Grey Forecasting Model	Self-collected	Accuracy	EC, Number of PMs
[102]	✓	×	×	×	×	PBC and DBC	Google dataset	Accuracy	Resource usage
[127]	✓	×	×	×	×	Ensemble Empirical Mode Decomposition, Runs Test, and Auto-Regressive Integrated Moving Average (EEMD-RT-ARIMA)	Alibaba	MAPE	Number of PMs
[86]	✓	×	×	×	×	Auto-Regressive Integrated Moving Average (ARIMA)	Wiki media Foundation Web Server Request Traces	Accuracy	Resource usage
[122]	✓	×	×	×	×	Advanced Model for Efficient Workload Prediction in the Cloud (AME-WPC)	Auver Grid	Mean Square Error(MSE), Normalized MSE	Resource usage
[172]	×	×	✓	✓	×	Migration control Modified overload detection	PlanetLab	Service Level Agreement Violation (SLAV)	EC
[151]	×	×	✓	✓	✓	Interquartile Range, Mean Absolute Deviation, Local Regression, Robust Local Regression	Planetlab	—	Resource usage, EC, Number of PMs
[174]	×	×	×	×	✓	FireFly Optimization	Random	—	EC

Reference	Resource Prediction	VMA	Overloaded Detection	VMS	VMP	Techniques used	Dataset	Evaluation metrics	Parameters: Resource usage, EC, Number of PMs, VM migrations
[175]	×	×	×	×	✓	Energy and Carbon-Efficient Placement	Random	—	EC
[164]	×	×	✓	✓	×	Performance-to-Power ratio	PlanetLab	SLAV	EC,VM migrations
[163]	×	×	×	✓	✓	Utilization Optimization VM, Pearson PABFD	PlanetLab,Random	SLAV	EC, VM migrations
[162]	✓	×	✓	×	✓	Robust simple LR model	PlanetLab,Random	SLAV	Resource usage, EC
[157]	×	×	✓	✓	✓	Energy-efficient and QoS dynamic Virtual Machine Consolidation (EQVC)	Bitbrain	SLAV	EC, VM migrations
[161]	×	×	×	×	✓	Utility Aware Best Fit Decrease	Random	SLAV	EC
[165]	×	×	✓	✓	✓	Growth Potential Aware VMP	PlanetLab	SLAV	EC
[154]	✓	×	✓	✓	✓	Prediction based on Markov chain model	Random	SLAV	EC, VM migrations
[132]	×	✓	×	×	×	Random, Sequence, Greedy technique	Planetlab	Execution time	—
[133]	×	✓	×	×	×	Krill Herd algorithm	CloudSim simulator	SLAV	EC
[134]	×	✓	✓	×	×	Bandwidth-Aware divisible Task model	Self - collected	Response time, Turnaround time	Resource usage
[135]	×	✓	✓	✓	✓	Discrete Particle Swarm Optimization	CloudSim configuration	—	EC, Number of PMs, VM migrations
[136]	×	×	×	×	✓	GA	Self collected	Execution time	EC
[176]	×	×	×	×	✓	Glowworm Swarm Optimisation	PlanetLab	SLAV	EC
[177]	×	×	×	×	✓	Cost and Renewable-Aware	Random	Power Usage Effectiveness (PUE), Carbon Footprint	EC
[178]	×	×	×	✓	✓	Most available Renewable energy	Google Cluster, meteorological data	Carbon Footprint, Brown Energy	Resource usage, EC, VM migrations

Reference	Resource Prediction	VMA	Overloaded Detection	VMS	VMP	Techniques used	Dataset	Evaluation metrics	Parameters: Resource usage, EC, Number of PMs, VM migrations
[179]	×	✓	✓	✓	✓	Power Efficient First-Fit Decreasing, Power Efficient Best-Fit Decreasing and Medium-Fit Power Efficient Decreasing	PlanetLab and Bitbrains	SLAV	EC, Number of PMs
[180]	×	×	×	×	✓	GA with the tabu search (GATA)	Google Cluster	maximizing load balance, SLAV	EC
[181]	×	✓	×	×	×	Resource ranking And utilization Factor (ERVS)	Random	makespan, cost, flow time, and emission rate of carbon dioxide	EC
[182]	×	×	×	✓	×	I/O and CPU Intensive VM selection	Planet lab	Quality of service (QoS), SLAV	EC, VM migrations
[167]	×	×	×	✓	×	Energy-Aware Dynamic VM Selection	CloudSim	SLAV	EC, VM migrations
[152]	×	×	✓	✓	✓	Heuristic Energy and Temperature aware-based VM consolidation (HET-VC) and FET-VC (Fire-Fly Energy and temperature-based VM Consolidation)	PlanetLab	SLAV, time and space complexities	EC
[153]	×	×	✓	×	✓	Host Utilization Aware (HUA)	PlanetLab	Minimal Migration costs	Resource usage, Number of PMs
[159]	✓	×	✓	×	×	Workload-based Dynamic threshold (WBDTH)	Self Collected	SLAV	EC, VM migrations
[183]	×	✓	✓	×	×	Self-adaptive Threshold method, grey correlation degree model	PlanetLab	SLAV	EC, VM migrations
[184]	×	×	×	✓	✓	Virtual Machine Selection Policy (MP)	Cloudsim	Migration Time, SLAV	EC, Resource usage
[185]	✓	×	×	×	×	Seasonal Auto-Regressive Integrated Moving Average (SARIMA)	Bitbrain	MAE, MAPE	Resource usage
[186]	×	✓	×	×	×	Binary Whale Optimization Approach (BWOA)	Random	—	Resource usage, Number of PMs
[137]	×	✓	×	×	×	Evolutionary approach	Coarse-Grained Simulation Engine	Instruction energy ratio	Resource usage, EC
[138]	×	✓	×	×	✓	GA, Cat Swarm Optimization	Self-collected	CPU usage percentage	Resource usage, EC, Number of PMs

Reference	Resource Prediction	VMA	Overloaded Detection	VMS	VMP	Techniques used	Dataset	Evaluation metrics	Parameters: Resource usage, EC, Number of PMs, VM migrations
[187]	×	×	×	✓	×	Power-aware VM Selection policy	PlanetLab	SLAV	EC, VM migrations
[99]	✓	×	×	×	×	Extreme Learning Machines (ELM)	GCT	RMSE	—
[100]	✓	✓	×	×	×	Kalman filter and Wavelet decomposition	Wikimedia Grid	Mean Absolute Error (MAE)	Resource usage
[110]	✓	✓	×	×	×	Neural Network, Auto-Regressive Filter	NASA, EPA	RMSE	EC
[188]	✓	×	✓	✓	✓	Virtual Machine Consolidation Utilization Prediction (VMCUP-M)	Synthetic and real-world workloads	SLAV	EC, VM migrations
[146]	×	×	✓	✓	✓	Ant Colony System (ACS-VMC)	CloudSim	QOS	EC, VM migrations
[189]	✓	×	✓	✓	✓	Utilization Prediction-VMC (UP-VMC)	CloudSim	RMSE	EC, VM migrations
[173]	×	×	✓	✓	×	Fuzzy VMS with migration control	PlanetLab	Migration time, Overload time fraction, SLAV	EC

PM with sufficient resource capacity follows a sequential order, starting with the first PM. If the first PM cannot fit a VM, the second PM is checked. If the second PM also can't fit it, the search continues to the third PM, and so on, always adhering to the initial order of PMs. Due to the possibility of a VM requiring more resources than what is currently available on a PM, the average runtime of FF is O(vp), in which v is the number of VMs and p represents the total number of PMs.

First Fit Decreasing (FFD) is a variant of the FF algorithm where the VMs are arranged in descending order based on their resource demand. First, the FF algorithm is used to search for the destination PM for the VM with the highest resource demand. The search then continues for the VM with the second highest resource demand, and so on. The time complexity of FF is O (vlogv+ vp), where v represents the total number of virtual machines and p represents the total number of physical machines. It should be noted that the running time of the sorting algorithm is (nlogn). Like FF, Next Fit (NF) also uses a sequential search method, which commences with the most recently chosen server in the preceding placement. In further elaboration, if the previous VM was allocated to the second PM, subsequent VM placements will commence the checking process from the second PM onwards. Conversely, the checking process would always initiate from the first PM for any VM in the FF and FFD algorithms. NF is commonly known as Round Robin (RR) [190]. The asymptotic time complexity of the NF algorithm is equivalent to that of

the FF algorithm. In Best Fit (BF), the destination PM has the lowest residual resource [189]. The residual resource of a PM is computed by subtracting the resource demand of its VMs and the target VM being searched for from its overall resource capacity. If the PMs are first sorted by resource consumption ratio, the Best-Fit Decreasing (BF) algorithm takes the same time as the FFD algorithm. However, without sorting, the Brute Force technique has a temporal complexity of $O(vp^2)$. For Best Fit Decrease (BFD), VMs are arranged by resource demand in decreasing order. Then BF algorithm searches for the destination physical machine for the initial VM with the highest resource demand, then the second, and so on. BFD has the same asymptotic running time as FFD. The power-aware best fit decreasing (PABFD) VM placement algorithm was proposed by the authors of [151]. The algorithm organises all VMs based on their current CPU usage, from highest to lowest, and allocates each VM to a host that results in the smallest increase in power consumption caused by the allocation. This approach allows the algorithm to exploit host heterogeneity by initially selecting the most energy-efficient hosts. The authors propose the fireFly optimization energy-aware VM migration (FFO-EVMM) technique in [174]. This approach aims to transfer the most heavily utilized VM from a currently operational node that satisfies a minimum energy consumption requirement to another operational node that has the lowest energy consumption.

[184] proposed a novel VM selection strategy, which incorporates resource satisfaction levels as a key factor in minimising energy consumption, VM migration, and SLAV. Further, a VM placement policy is also proposed to identify target hosts demonstrating the least correlation coefficient with the VMs eligible for migration. [191] presented the Energy-efficient and Traffic-managing Ant Colony Optimization (ETA-ACO) algorithm. They offer three novel methods to improve ETA-ACO. Energy and bandwidth-conscious tactics are used to choose the best PM to host a VM. Initially, maintain PMs with lower power consumption, then select the one with the least bandwidth resource demand for the VM. Second, ETA-ACO arranges VMs by traffic demand in descending order to optimize traffic handling. The third method distributes the best solution's components over a set of new solutions. This approach improves ETA-ACO performance for VMP.

[192] proposed a Dynamic Consolidation with Minimization of Migration Thrashing (DCMMT) approach that prioritises high-capacity VMs, aiming to substantially decrease migration thrashing and minimise the number of migrations to uphold SLAs. Rather than migrating VMs susceptible to migration thrashing, DCMMT strategically places them on the same physical servers. Their results were compared to aggressive-migration-based solutions and concluded that DCMMT improved by 28% in migration thrashing, a 21% reduction in the number of migrations, and a 19% enhancement in SLAV.

[193] proposed a VM utilization-based overload detection algorithm. This algorithm forecasts host overload by analysing the historical utilization of VMs and determining whether it surpasses the available host capacity. To minimise energy consumption while avoiding SLAV, [144] proposed a static double threshold technique which selects the overloaded and underloaded host based on the CPU utilization of the host that comes in between decided upper and lower threshold values. Further, they proposed random choice, minimum migrations, and the highest potential growth policy for VM selection. Lastly, for energy aware VM placement, they modified the BFD algorithm. Prioritising system resources such as CPU, RAM, or bandwidth, [194] have focused on establishing a threshold based on the host's temperature metric. Their results were compared with LrMmt and IqrMc, concluding that they offer substantial enhancements in energy consumption, SLA compliance, and the number of live migrations.

[195] introduced the VMCUP-M algorithm, a VMC solution that incorporates multiple usage prediction using linear regression to enhance the energy efficiency of CDCs, where multiple usages mean multiple prediction steps for multiple resource types. Their results concluded that by combining the assessment of current and forecasted resource utilization, they achieved a dependable means of identifying overloaded and underloaded servers. This, in turn, leads to a reduction in both server load and power consumption after consolidation. [196] employed a predictive model that integrates grey modeling with ARIMA to detect host status. In addition, they introduced a novel virtual machine placement strategy that prioritises resource utilization and dynamically manages energy usage to determine the most appropriate host. Further, it introduced a VM selection policy known as AUMT, which prioritises the selection of virtual machines with minimal average CPU utilization and migration time. Compared to benchmark approaches, it reduces energy usage by 56.07%, migrations by 79.21%, SLAV by 91.01%, and ESV by 84.34%. The AUMT policy reduces energy consumption, migrations, and ESV by 15.46%, 28.11%, and 3.96%. [146] introduced a novel and dynamic VMC method, ACS-VMC. This method consolidates VMs onto fewer active PMs while ensuring QoS to reduce data center energy consumption. Since the VM consolidation problem is NP-hard, they employed the Ant Colony System to find near-optimal solutions. A multi-objective function that minimises dormant PMs and reduces migrations is used in this method. Compared to dynamic VM consolidation methods (AVVMC, IQR, MADTHR and IR), this approach reduces energy consumption, SLA violations, and migrations.
In another work, [189] presented a Utilization Prediction-aware VM Consolidation (UP-VMC) approach. This approach employed LR and KNNR to predict short-term CPU and memory usage and rely on current and future resource utilization information to detect overloaded and underloaded hosts. Further, to select VMs from detected overloaded

hosts, they employed VM allocation policies (Minimum Migration Time, Maximum Load and Minimum Load). The approach involves migrating all VMs from the cold spots to the heavily loaded PMs and aims to decrease the energy consumption of data centers by freeing up resources in the cold spots. Achieved results are better than MBFD, MFFD, PM Utilization Prediction-aware VM Consolidation, VM Utilization Prediction-aware VM Consolidation, Sercon, and ACS-VMC in terms of minimising the energy consumption, count of VM migrations and SLAV. [197] employed Glowworm Swarm Optimization for VM. Their method outperforms power-aware best fit decreasing in terms of energy efficiency, SLAV reduction, the combined improvement in energy efficiency and SLA compliance, and a reduction in the number of virtual machine migrations.

2.4 Research Findings and Gaps

Based on a comprehensive analysis of the existing literature survey presented in the above sections, it is evident that significant research gaps persist in the domains of resource usage prediction, virtual machine allocation, and dynamic consolidation within cloud environments. These gaps have substantial implications for cloud systems' overall performance and efficiency. The identified gaps are outlined below:

1. In most existing research, data for time series prediction is typically prepared using sum or average techniques. However, tasks on machines often overlap and share resources, these methods may not accurately reflect the actual resource usage at any given time interval. Therefore, it is crucial to preprocess the data to preserve its integrity and originality to ensure that the resulting representation reflects the actual resource utilization patterns without distortion or loss of information.

2. In literature, there is no organized procedure in the machine selection from the dataset to train and test the accuracy of the employed model. Most studies on resource usage prediction have utilized randomly selected machine data, potentially resulting entirely in homogeneous machine selection. This approach overlooks the critical necessity of evaluating prediction models in heterogeneous machine environments that mirror real-world cloud scenarios. Therefore, an organized procedure to select machines from the dataset is essential for training and testing employed prediction models across various machine configurations.

3. To evaluate the performance of employed models, the choice of dataset plays a pivotal role in analyzing and comparing the research findings, as they directly influence the outcomes of employed algorithms. Several studies rely on random or partial datasets for experimentation. Therefore, employing extensive, real-time datasets is

imperative to accurately evaluate the proposed algorithms' performance.

4. Most of the employed resource usage prediction techniques have a limited ability to capture long sequences, long-range dependencies and parallel processing. This highlights the need for advanced models that are capable of handling complex sequential data with extended dependencies, emphasizing the importance of exploring novel approaches to enhance resource usage prediction accuracy and efficiency.

5. Most of the existing studies focus on individual aspects of VM consolidation, such as host overload detection, host underload detection, VM selection, or VM placement. This necessitates the development of comprehensive strategies that integrate these challenges to manage the resources of data centers effectively and reduce energy consumption.

6. Several researchers used only current resource usage during VM consolidation. This often results in high SLA violations, increased migration costs, and machines being repeatedly selected as overloaded hosts due to insufficient consideration of historical usage trends. Therefore, developing and validating dynamic consolidation strategies that use comprehensive historical data, predictive analytics, and smart decision-making algorithms is crucial to reducing SLAV, enhancing migration efficiency, and preventing recurring allocation problems.

7. Several existing VM placement algorithms commonly select the host with the least resource usage as the destinated host. This method leads to more operational hosts responsible for managing user requests, resulting in higher energy consumption. Hence, enhancing VM placement policies to reduce the number of hosts and minimize energy consumption is not getting enough attention.

8. In the existing literature, the integration of virtual machine allocation, prediction, and consolidation has not been given enough attention. Therefore, it is essential to develop a holistic solution that can optimally allocate VMs and dynamically consolidate based on both current and predictive resource usage of PMs. This holistic approach is crucial for effectively managing data center resources and reducing energy consumption.

2.5 Summary

In this chapter, a comprehensive literature review on resource usage prediction was conducted, which categorized into statistical, machine learning, deep learning, and hybrid approaches. Then examined the literature on virtual machine allocation strategies, fol-

lowed by an in-depth analysis of virtual machine consolidation techniques, including the algorithms used to detect under-loaded and overloaded states, virtual machine selection, and optimal virtual machine placement. Lastly, several unsolved issues have been extracted from the literature review that will help in efficient resource management of the cloud environment.

In the next chapter, a prediction model GMM-LSTM is introduced to predict resource usage across heterogeneous machines. In addition, a data preprocessing algorithm, Sum Average (SA) is proposed which enables the extraction of consistent time series data from machines to train the prediction model.

Chapter 3

GMM-LSTM: A Component Driven Resource Utilization Prediction Model

In this chapter, a prediction model (GMM-LSTM) is introduced to predict PM resource utilization, enabling effective utilizing the entire data center's resources to reduce energy consumption. First, the raw time series workload is processed to enhance the value of its features for better training and prediction of mean resource utilization in the CDC using the proposed Sum-Average (SA) algorithm. Afterwards, the Gaussian Mixture Model (GMM) is employed to cluster heterogeneous machines of data center based on their mean CPU and memory usage, which helps to analyze the prediction for each kind of configured machine available in a data center. Finally, the Long Short Term Memory model (LSTM) is employed to predict the mean resource usage of PMs for every clustered machine. The effectiveness of the proposed model is evaluated using the Google cluster trace usage (GCT) dataset and compared with Linear Regression (LR), Moving Average (MA), and Auto Regression Integrated Moving Average model (ARIMA). The proposed model outperforms the other compared techniques in terms of achieved RMSE.

3.1 Introduction

Cloud computing has captivated users across the globe due to its pay-per-use model. It provides various services, such as computation, storage as well as software. Virtualization is an enabling technology to scale resources based on customers' demands. Nowadays, the rapid growth of CDCs results in heavy energy consumption due to a huge number of active PMs to process users' requests, primarily attributed to poor resource management of cloud infrastructure. The high energy consumption in data centers has serious economic as well as environmental concerns.

The most widely adopted strategy to reduce energy consumption in data centers is to minimize the number of active PMs. This can be achieved through the effective utilization of the available resources on each PM [198, 199]. Accurate prediction of resource utilization enables CSPs to manage PMs' resources efficiently. Hence, it helps to reduce the active number of PMs in a data center. However, dynamic changes involved in workload requests and resource usage of heterogeneous infrastructure are challenging tasks to predict resource consumption accurately.

Several statistical time series, machine learning models, deep learning and hybrid algorithms are employed to predict VM workload, VM requests, resource demands, and resource utilization in the data center, as we discussed in chapter 2.1. Time series models such as LR, Auto Regression, MA, ARIMA, Exponential Smoothing, and others are widely employed for the prediction of resource utilization. However, these models do not support accurate prediction for a longer duration [200, 201]. To address this issue, various ML models such as Support Vector Machine, Random Forests, K-Nearest Neighbors [202], Self-Organizing Feature Map, GA, and RNN have been employed to improve accuracy. However, these models also struggle with the vanishing gradient problem, which hinders their ability to achieve precise accuracy.

This chapter presents a resource utilization prediction model which helps to minimize the active number of PMs and energy consumption of a data center. Firstly, we employed a probabilistic model, i.e., the Gaussian Mixture Model (GMM), to cluster heterogeneous machines of a data center based on its resource consumption into an optimal number of components. Furthermore, an Artificial Recurrent Neural Network, i.e., Long Short Term Memory (LSTM), is employed to predict the resource usage of PMs. Finally, the Google Cluster Usage Trace (GCT) dataset [203] is used to evaluate the effectiveness of our proposed methodology. In summary, the contributions of the proposed methodology can be listed as below:

- Proposed a Sum-Average algorithm for data preprocessing that enhances the raw data view for effective clustering, training, and prediction of mean resource utilization of PMs in a data center. The proposed preprocessing technique converts the dataset into fixed interval time series data as well as extracts machine information from task data to predict the resource usage of a PM. The proposed algorithm cleans raw data from missing values, unequal time intervals, and overlapping of resource usage by multiple tasks and prepares it for a machine learning model, which reduces data size as well as improves the prediction model's accuracy and effectiveness.

- Introduced a novel method for categorization of PMs into different components or clusters based on their resource usage. This technique of categorization helps to process huge data and can save training time. To predict resource usage for all available PMs in a data center is a time-consuming as well as computationally expensive process. However, this technique enables the selection of a set of heterogeneous PMs to represent the entire data center.

- The proposed methodology employs the univariate LSTM model to efficiently predict resource usage on fixed interval time series preprocessed data. The proposed methodology accurately predicts the mean CPU usage of PM using the optimal number of hidden layers, epoch, and prediction window size. This will help CSPs to make better decisions during VM allocation and dynamic consolidation.

- To evaluate the performance of the proposed methodology, Several experimental simulations are conducted on real-world workloads using various time series models such as LR, MA, ARIMA, and LSTM. The performance of the proposed approach outperforms.

The rest of the chapter is organized as follows. In Section 3.2 Google cluster trace dataset is summarized. All the techniques used in this chapter are described in Section 3.3. The proposed methodology steps, i.e., data processing, clustering-based machine selection and predictor, are presented in Section 3.4. Finally, Section 3.5 and 3.6 provide the experimental results discussion and summary.

3.2 Dataset Description

Google Cluster Trace (GCT) usage dataset [203] has been employed to evaluate the proposed scheme. Google Cloud SDK tool is used to download all the dataset files. GCT comprises 29 days of workload information corresponding to 12.5K machines. In Google, data center workload is identified as a job that is further subdivided into single or multiple

tasks to be processed on various PMs. The entire information corresponding to machines and tasks is collected within six different tables in the dataset. The machine events table and machine attribute table describe the information of the machine, while the job event table, task event table, and task constraint table expound the job and task information. The task resource usage (TRU) table describes the number of resources consumed by a task every 5 minute time interval. TRU table consists of 500 unzip .csv files with 20 attributes that are mentioned in table 3.1. This table mainly consists of task information, i.e. which task is running on which machine ID, average CPU, canonical memory, and local hard disk space utilized by a particular task in a specific interval of time.

Table 3.1: TRU table of GCT dataset features with description

Sr. No.	**Feature name**	**Description**
1	Start time	Start time of a task measurement period (in seconds)
2	End time	End time of a task measurement period (in seconds)
3	Job ID	Unique ID for each user's arriving request
4	Task index	Task index within the respective Job ID
5	Machine ID	Machine ID on which the task is processing
6	Mean CPU usage rate	Average CPU usage over a uniformly chosen one-second sample from a 5-minute measurement period
7	Canonical memory usage	Average memory usage over a uniformly chosen one-second sample from a 5-minute measurement period
8	Assigned memory usage	Number of pages accessed by users
9	Unmapped page cache memory usage	Page cache not mapped into any user space process
10	Total page cache memory usage	Total Linux page cache
11	Maximum memory usage	Maximum canonical memory usage measurement amount that has been achieved during the time
12	Mean disk I/O time	Mean across all disks on the machine, in units of disk-time seconds per second
13	Mean local disk space used	Mean disk space recorded in the trace
14	Maximum CPU usage	Maximum CPU usage measurement amount that has been achieved during the time
15	Maximum disk IO time	Maximum across all disks on the machine
16	Cycles per instruction (CPI)	The average amount of CPU cycles needed to execute an instruction

Table 3.1: TRU table of GCT dataset features with description

Sr. No.	Feature name	Description
17	Memory accesses per instruction (MAI)	Measurement of cache misses at the last-level cache are utilized to estimate memory accesses.
18	Sample portion	The ratio of the predicted sample size to the actual sample size
19	Aggregation type	1 if maximums from sub containers were summed
20	Sampled CPU usage	Average CPU utilization for a randomly selected 1-second interval within the specified measurement period.

3.3 Techniques Used

- **Feature Scaling:** Feature scaling is the best way to give an equal contribution to all the features which are used during the model estimation i.e. also known as normalization. Min-Max scaler and standardization are the two most commonly used methods for scaling purposes. Min-Max scaler transforms all the features in the range [0,1], which means that for all the features, the minimum value is mapped to 0, the maximum value is mapped to 1, and all other values are mapped to decimal numbers between 0 and 1.

$$z = \frac{x - min(x)}{max(x) - min(x)} \tag{3.1}$$

  standardization is the process of rescaling the individual features so that they will have the Gaussian distribution properties with $\mu = 0$ and $\sigma = 1$. The standardization score is also known as the Z score, and it is calculated as:

$$z = \frac{x - \mu}{\sigma} \tag{3.2}$$

  where x is the value of each variable, μ is the mean and σ is the standard deviation.

- **Hopkins statistic (H):** This statistic is used to get the clustering tendency of any dataset. In Hopkins, an artificial data point is generated as the original data point, and distance is calculated for each data point with its nearest neighbour, i.e., denoted by w. The same process is repeated for the artificially generated data

points, i.e., denoted as u. H is calculated as:

$$H = \frac{\sum_{i=1}^{p} w}{\sum_{i=1}^{p} u + \sum_{i=1}^{p} w} \tag{3.3}$$

If the H value is nearest to 0.5, that means the dataset is not effective for clustering purposes.

- **Silhouette Score:** This is a metric that calculates the goodness of the clustering technique. The value of the silhouette score is in the range of -1 to 1. If the score is 1 that means all the data points within a cluster are compact and far away from the other clusters. 0 denotes the overlapping of clusters and -1 is the worst. Here is the equation which is used to calculate the silhouette score or silhouette coefficient.

$$s(o) = \frac{b(o) - a(o)}{max\{a(o), b(o)\}} \tag{3.4}$$

The silhouette coefficient for a data point o is denoted as $S(o)$. It is calculated based on two measures: $a(o)$, which represents the average distance between point o and all other points within the same cluster, and $b(o)$, which represents the minimum average distance between point o and all other clusters that it does not belong to.

- **Gaussian Mixture Model:** This model is a probabilistic distribution model and uses a soft computing approach to distribute the data points in a specific cluster. The Gaussian distribution is also known as the normal distribution. In this, data points are symmetrically distributed around the mean value and form a bell-shaped curve. In one-dimensional space, i.e., only one variable, the Gaussian distribution probability density function is given by:

$$f(x, \mu, \sigma^2) = (\frac{1}{\sqrt{2\Pi\sigma^2}}) e^{\frac{-(x-\Pi)^2}{2\sigma^2}} \tag{3.5}$$

where μ is the mean and σ^2 is the variance.

3.4 Proposed Scheme

With a notion to proffer an accurate prediction model for resource consumption of PMs in a data center, the proposed scheme employs the GMM along with the LSTM model. Figure 3.1 depicts procedures of the proposed methodology. Firstly, the Sum-Average (SA) algorithm is introduced for the preprocessing of the dataset. This algorithm calculates the mean resource usage of a PM from task resource usage to map each PM into

a specific cluster as well as it helps to convert the mean resource usage of an individual PM into 5 minute time interval for resource utilization prediction. Afterwards, GMM is employed to cluster available PMs of the data center based on the output of the SA algorithm. Clustering helps to figure out heterogeneous configuration PMs present in the data center [204, 205]. Finally, the univariate LSTM model is employed on selected machines of each component/cluster to predict the mean resource usage with minimum root mean square error (RMSE). The procedures followed in the proposed methodology

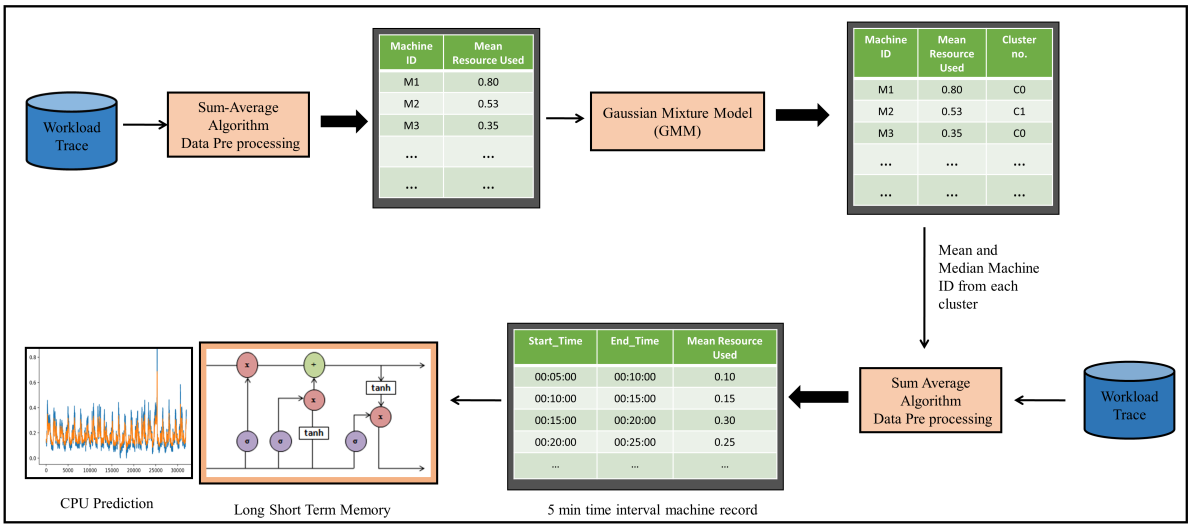


Figure 3.1: Working of the Proposed Scheme

are divided into three steps: data preprocessing, clustering-based machine selection, and predictor. All these steps are discussed in the subsections.

3.4.1 Step1: Data Preprocessing

Data preprocessing is required to prepare the dataset for the clustering technique as well as the prediction model. To cluster entire PMs of the data center, data preprocessing calculates the average resource usage of each PM from task resource usage attributes. For prediction, firstly, extract the resource usage of individual machines. Afterwards, the SA algorithm is employed to calculate the mean resource usage of every 5 minute time interval for 29 days.

In the GCT dataset, several tasks are executed over a single machine with a maximum of 5 minutes. Figure 3.2 depicts mean CPU usage for an interval of 5 minutes (300 seconds) over a machine M. The percentage value denotes the mean CPU usage along with its task number, and the x-axis represents the time in seconds. Five different tasks are executed with a different time span, i.e. $task_1$ and $task_2$ for 50 sec exclusively using 30% and 40% of CPU, respectively. $Task_3$ is using 55% of CPU sharing 20 sec with $task_1$ and 50 sec

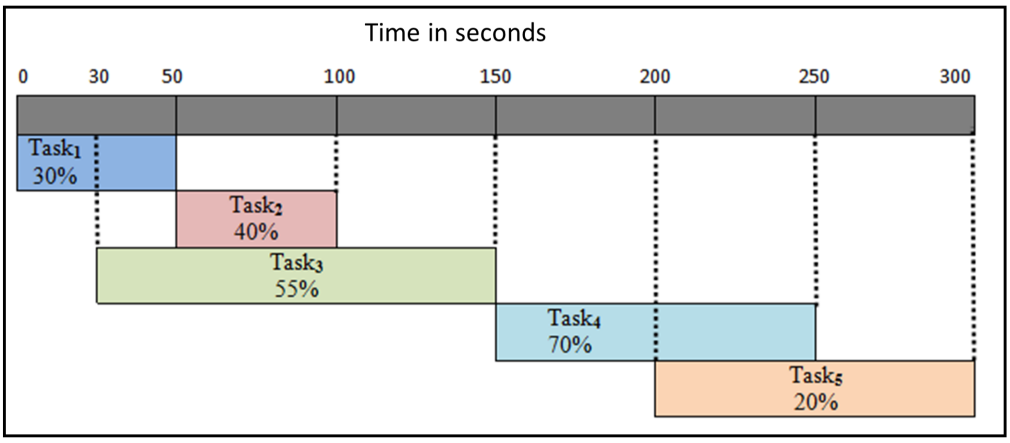


Figure 3.2: Machine M task CPU usage (%) for 5 minute time interval

with $task_2$, while 50 seconds exclusively. 70% of CPU is consumed for a time duration of 100 seconds by $task_4$, while $task_5$ executes for a total duration of 100 sec with 20% of CPU usage with 50 seconds sharing with $task_4$ rest 50 sec exclusive. With a notion to figure out the exact CPU usage by a machine, neither the sum nor average method suits the dataset because the dataset defines the CPU consumed by tasks for a different period overlapping with each other. The sum method may result in the overutilization of resources, while the average may result in the underutilization of resources. Hence, the amalgamation of the Sum Average algorithm is proposed that enables us to figure out the exact mean CPU usage for a single machine for a specific duration.

Figure 3.3 depicts the working of the proposed SA algorithm-based preprocessing technique. The percentage denotes the mean CPU usage along with its task number. + and μ denote aggregation and mean of CPU resource, respectively. The x-axis represents the time in seconds. Firstly, it extracts the unique start as well as end timestamp of each task. Next, it segregates the task based on overlapping time intervals with other tasks. As depicted in figure 3.2, $task_1$ executes on machine M for a duration of 50 sec of which the first 30 seconds it exclusively operates on M while for the rest 20 seconds resources of M is shared with $task_3$. Thus, to identify the overall utilization for the said duration, the proposed algorithm 3.1 aggregates the resource consumption by individual tasks for the overlapped time. Similarly, the resource consumption of $task_2$ and $task_3$ is aggregated for the next 50 seconds. Afterwards, $task_3$ operates for 50 seconds consuming 55% of CPU, and $task_4$ operates for the next 50 seconds consuming 70% of CPU with no overlapping time due to which aggregation is not required. However, $task_4$ and $task_5$ inclusively operate on M for 50 seconds, consuming 70% and 20% CPU, respectively, due to which we aggregate the CPU consumption for this 50 sec. Lastly, $task_5$ exclusively operates on M consuming 20% of CPU. Finally, algorithm 3.1 calculates the average CPU consumption

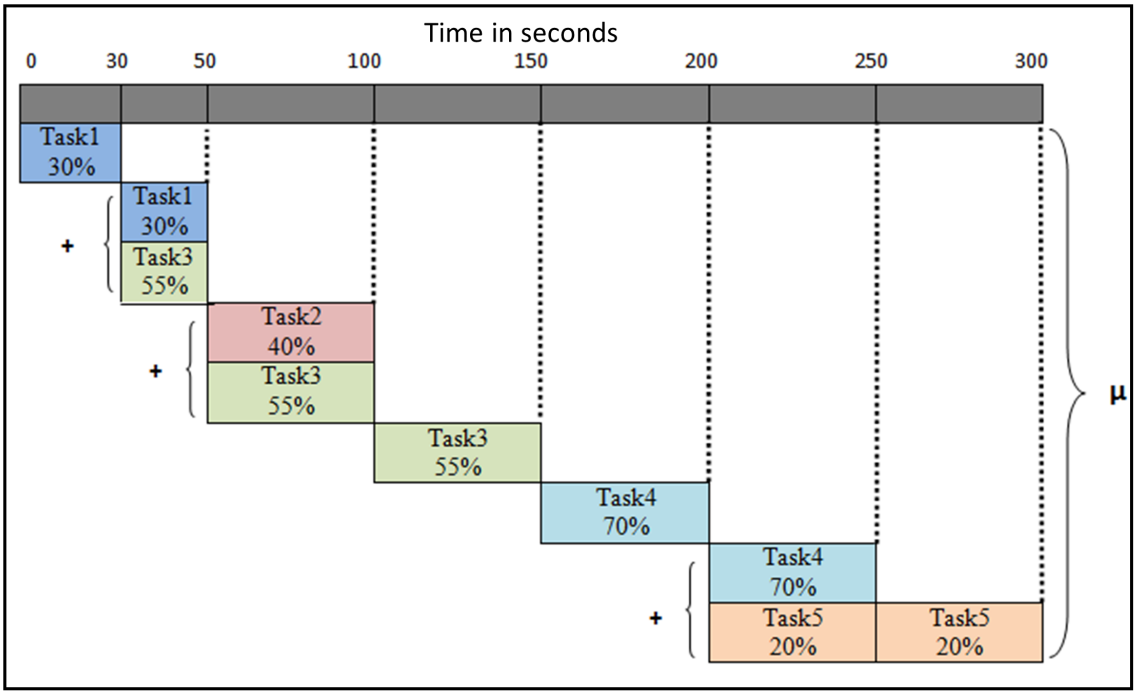


Figure 3.3: Calculate machine M CPU usage (%) from task CPU usage(%) using Sum Average Algorithm for 5 minute time interval

of each PM for step 2 and also converts 5 minutes time duration for step 3 by taking an average of all the aggregated as well as exclusive CPU consumption values.

Algorithm 3.1 $Sum - Average$: Retrieving machine information from task information using TRU table of GCT dataset

Input: $[G] = (g_{ij})_{i=1,j=1}^{i=p,j=a}$ where, $g_{i,j} = j^{th}$ feature of i^{th} file, p = Total number of files in TRU table, a = Total number of features in p. **Output:** MID_j where, MID is Machine ID with relevant features

1: $t_{ij} = (x_1, x_2, x_3, x_4, x_5)$ where, x_1= Start_Time, x_2= End_Time, x_3= Machine ID, x_4= Mean CPU usage, and x_5= Canonical memory usage.
2: $group(x_3) \rightarrow N$ where, N = Total number of machines.
3: **for** i in N **do**
4: $\sum_{r_i}(x_1, x_2)(x_3, x_4, x_5)$ where, r_i is the number of records for machine i
5: Extract unique $(x_1, x_2) \rightarrow p$ where, p is an array
6: Break x_1 and x_2 according to p for each task
7: Assign x_4, x_5 according to x_1, x_2
8: $group(x_1, x_2).sum() \rightarrow$ Final {Prepare data for clustering}
9: $Final.mean() \rightarrow D$ {Prepare data for prediction}
10: **end for**

Algorithm 3.1 shows a step-by-step procedure to calculate the mean CPU usage of PM from running task information of that machine. Here, we are going to extract and calculate PM features due to the requirement of PM resource usage for prediction. We prepare

data for clustering of available PMs of the entire data center and prediction of resource usage of PMs. To divide all the available PMs of a data center into a fixed number of clusters, the raw data set is processed using the Sum - Average algorithm and calculates each PM's mean resource usage. The output of pre-processing for clustering is machine ID with its corresponding mean resource usage. To predict the resource usage of PM, firstly extract a particular machine information for the entire data set. Afterwards, the SA algorithm is applied, and the 5 minute mean resource usage is calculated.

3.4.2 Step 2: Clustering based Machine Selection

GCT dataset provides information associated with $12.5K$ machines [203]. The prediction of the resource consumption of all the machines is a time-consuming process. Clustering is a widely employed approach to group similar elements. Hence, we proposed to employ clustering to group similar types of machines based on their resource consumption. The clustering enables us to figure out the few sets of machines (S) that represent the entire machines of the data center. Thus, we can apply prediction on the (S) set of the machine to predict the CPU consumption of PMs for the entire data center.

Figure 3.4 depicts the proposed process of selecting a representation of machines from a large group. All features of the TRU table are not required for clustering, so extraction of the relevant features from the TRU table for cluster PMs is based on average CPU as well as canonical memory usage. SA was employed for the data preprocessing and extracting mean CPU as well as canonical memory usage of PM from task mean CPU and canonical memory usage. Afterwards, the min-max technique is employed to normalize the processed data. To verify whether clustering is a well-suited approach for the data or not, we measure its probability using Hopkins statistics [206]. The Hopkins statistics for the dataset used for experiments is greater than 0.5, which suggests that clustering is a suitable approach for the dataset. Further, the Silhouette score is computed to figure out the optimal number of clusters for GCT [207]. GMM is employed to assign all the operational machines from the GCT dataset to a specific cluster based on mean as well as variance [208]. Thus, every cluster is comprised of machines with similar resource usage. Clustering itself means that all the machines within the cluster are having a similar kind of resource usage. Hence, two machines from each cluster that represent those clusters are extracted. Those machines which are nearest to the mean and median value of mean CPU usage of each cluster are selected to represent the entire cluster. These extracted machine IDs are further used for the prediction of mean CPU usage. Finally, two PMs were used to represent the individual cluster and the entire data center is represented by twice the number of optimal clusters.

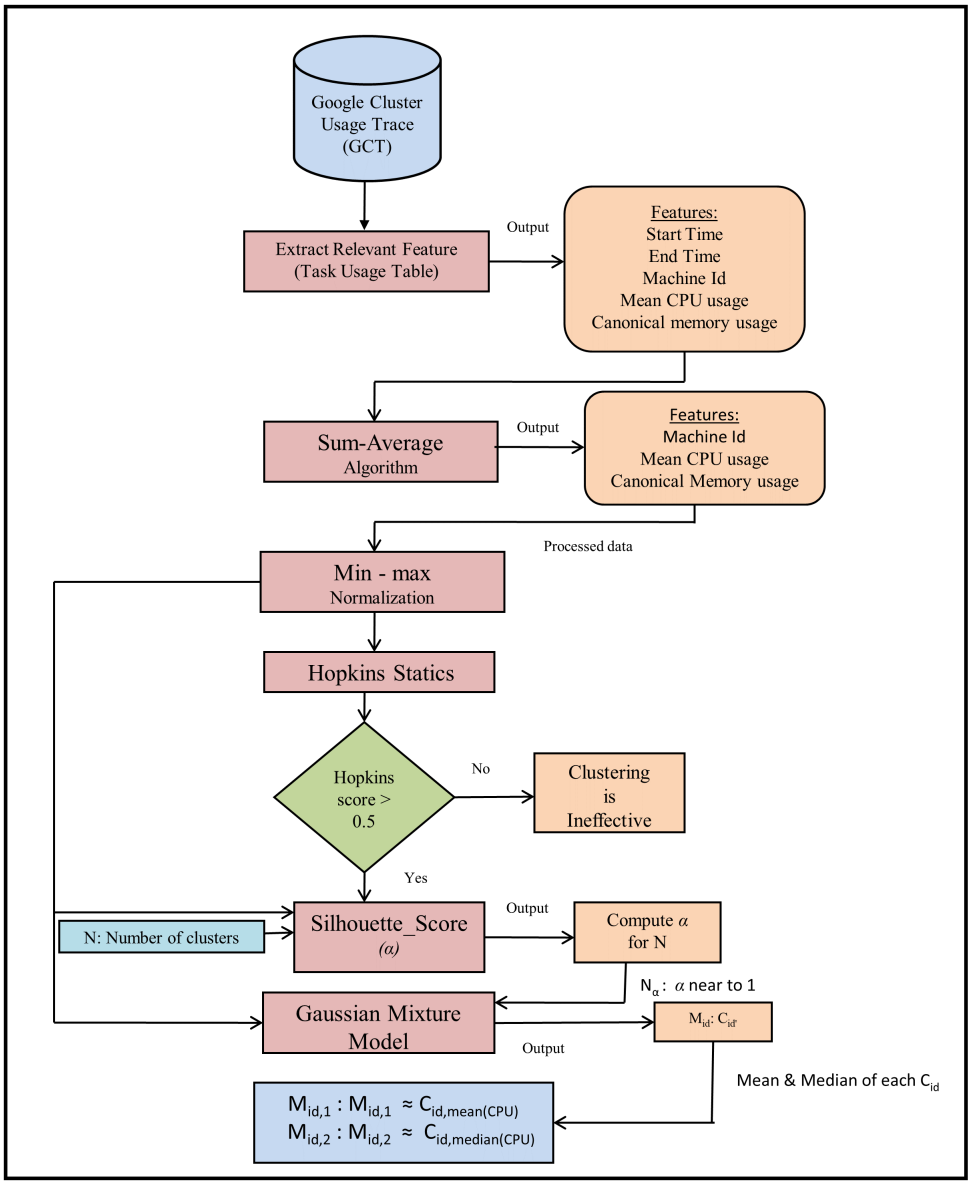


Figure 3.4: The proposed machine selection approach

Algorithm 3.2 demonstrated the procedure used to cluster available PMs in a data center. G denotes all files of the GCT dataset. t_{ij} are the selected features from the TRU table for the further procedure. Normalized processed data is taken as Hopkins' input, and the output value is stored in variable H. The H value finalized whether clustering is an effective method or not for a particular data. Afterwards, the silhouette score is calculated, and the GMM is applied. Here in algorithm 3.2, V_k variable denotes the optimal number of clusters possible. Two machine IDs from each cluster are extracted using an iterative loop. In each loop process, firstly calculate the mean and median CPU usage of each cluster and store the value in α and β. Afterwards, find the nearest neighbouring machine whose CPU usage is possible to α and β.

Algorithm 3.2 SlctMch: To select a set of machines from the GCT dataset representing the entire available machines in the Google data center.

Input: $[G] = \left(g_{ij}\right)_{i=1,j=1}^{i=p,j=a}$ Where, $g_{i,j} = j^{th}$ feature of i^{th} file, p = Total no. of files in TRU table, a= Total number of features in p. **Output:** $M_{\{i,1\}} = \left\{M_{Id_{\{C_i,me\}}}\right\}_{i=1}^{n}$, $M_{\{i,2\}} = \left\{M_{Id_{\{C_i,med\}}}\right\}_{i=1}^{n}$ Where, M_{Id} = Machine Id, C_i = Cluster Id_i, me = Mean machine of cluster, med = Median machine of cluster, n = number of clusters.

$t_{ij} = (x_1, x_2, x_3, x_4, x_5)$ Where, x_1=Start_Time, x_2= End_Time, x_3= Machine ID, x_4= Mean CPU usage and x_5= Canonical memory usage.
$SA() \rightarrow D$
$minmaxscaler() \rightarrow S$
$S.fit(D) \rightarrow D1$
$Hopkins(D_1) \rightarrow H$
if $H > 0.5$ **then**
    $Silhouette\_score(D_1, V) \rightarrow C$ Where, V = Number of components
    $max(C) \rightarrow V_k$ Where, V_k is component number with max C
    $GMM(D_1, V_k)$
    for j in V_k **do**
        $D_1[x_4].mean() \rightarrow \alpha$
        $(D_1[x_4] - \alpha).abs().sort() \rightarrow M_{j,me}$
        $D_1[x_4].median() \rightarrow \beta$
        $(D_1[x_4] - \beta).abs().sort() \rightarrow M_{j,med}$
    end for
else
    D_1 is ineffective for clustering
end if

3.4.3 Step 3: Predictor

The accurate prediction of the resource consumption of PMs is a challenging problem for CSPs. The prediction of PM resource consumption enables the effective utilization of the machines' resources, which will help to figure out the overloaded and under-loaded machines in a data center. It helps to reduce the number of migration overhead as well as an active number of PMs to conserve energy. The mathematical representation of the mean CPU usage prediction of machine M for the next 5 minutes i.e., $P_{k+1}(M)$ by using the previous K mean CPU usage $P_1(M)$ ——$P_k(M)$.

Long Short Term Memory Model

We employed the LSTM model for CPU usage prediction of PMs.. LSTM is a type of RNN architecture commonly used for time series prediction (TSP). It is essential to remember

the previous words to predict the next term; therefore, RNN takes input sequences using a loop structure, as represented in figure 3.5(a). One limitation of RNNs is that they can struggle to capture dependencies in time series data, as the gradient can become vanishingly small or explode over time [209]. LSTM has been developed to address this issue.

LSTM is designed to model sequential data by selectively remembering or forgetting information from previous inputs. The LSTM network uses a series of memory cells, gates, and activation functions to learn and predict patterns in sequential data [210]. The mathematical representation of an LSTM cell can be expressed as follows:

First, the input to the LSTM cell is a vector $x(t)$ at time t. The input is transformed by applying a linear transformation using a weight matrix W and a bias vector b:

$$a_t = \sigma(W_a \left[x_t, h_{t-1}\right] + b_a) \tag{3.6}$$

$$b_t = \sigma(W_b \left[x_t, h_{t-1}\right] + b_b) \tag{3.7}$$

$$d_t = \sigma(W_d \left[x_t, h_{t-1}\right] + b_d) \tag{3.8}$$

Here, a_t,b_t, and d_t are the forget, input, and output gates, respectively. They control the flow of information in and out of the memory cell. They are computed by applying a sigmoid activation function to a linear combination of the input vector x_t, the previously hidden state h_{t-1}, and bias terms b_a, b_b, and b_d as depicted in figure 3.5 (b).

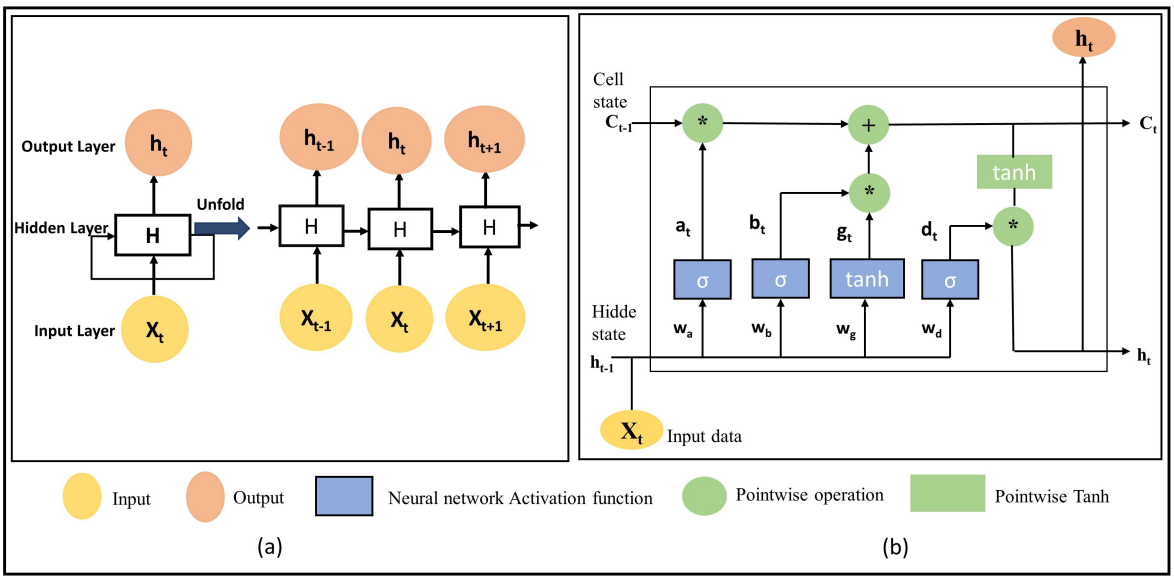


Figure 3.5: (a) Architecture of Recurrent neural network (b) Architecture of Long short term memory model

$$g_t = \tanh(W_g \left[x_t, h_{t-1}\right] + b_g) \tag{3.9}$$

g_t is the candidate value, which is computed by applying the tanh activation function of a neural network to a linear combination of input vector x_t, previously hidden state h_{t-1}, and bias term b_g. This new candidate is used to update the memory cell state. Next, the memory cell state C_t is updated by selectively forgetting or remembering information from the previous memory cell state C_{t-1}, current input x_t, based on values of forget gate a_t and the input gate b_t:

$$C_t = a_t * C_{t-1} + b_t * g_t \tag{3.10}$$

Finally, the hidden state h_t is computed by applying the output gate d_t to the memory cell state C_t, and passing the result through the tanh operation:

$$h_t = d_t * \tanh(C_t) \tag{3.11}$$

The output of the LSTM cell is the hidden state h_t at time t. This hidden state can be used for prediction or passed to the next LSTM cell in the sequence.

In this regard, the proposed methodology employs the univariate LSTM to predict the mean resource usage of the PM. Firstly, relevant features such as start time, end time, machine ID, and mean CPU usage are extracted corresponding to machines M_{mean} and M_{median} associated with each cluster (the result of step 3.4.2). Next, the SA algorithm is employed to calculate PM mean CPU utilization with a time interval of 5 minutes of 29 days. Afterwards, the univariate LSTM model is employed to predict the mean CPU usage of a machine. To converge the LSTM model faster, it is important to scale the input data. A larger value in the input slows down the model learning. Hence, a standard scaler is employed to scale the data. After scaling, the data is transformed into an appropriate format that is suitable for the LSTM model. The output of the LSTM model, i.e., predicted value depends on the last sequence provided as input in the network. Lastly, it calculates the RMSE to check the error accuracy between actual and predicted resource usage.

The flow chart in figure 3.6 represents all prediction steps using a directional arrow. The direction of the arrow helps to find out the input and output of each process. It shows that to extract the required features for prediction, the input should be the original TRU table of the GCT dataset, and the output will be some selected features only (i.e., start time, end time, machine ID, as well as mean CPU usage). Afterwards, the output of the feature extraction process along with Machine IDs are retrieved from the clustering

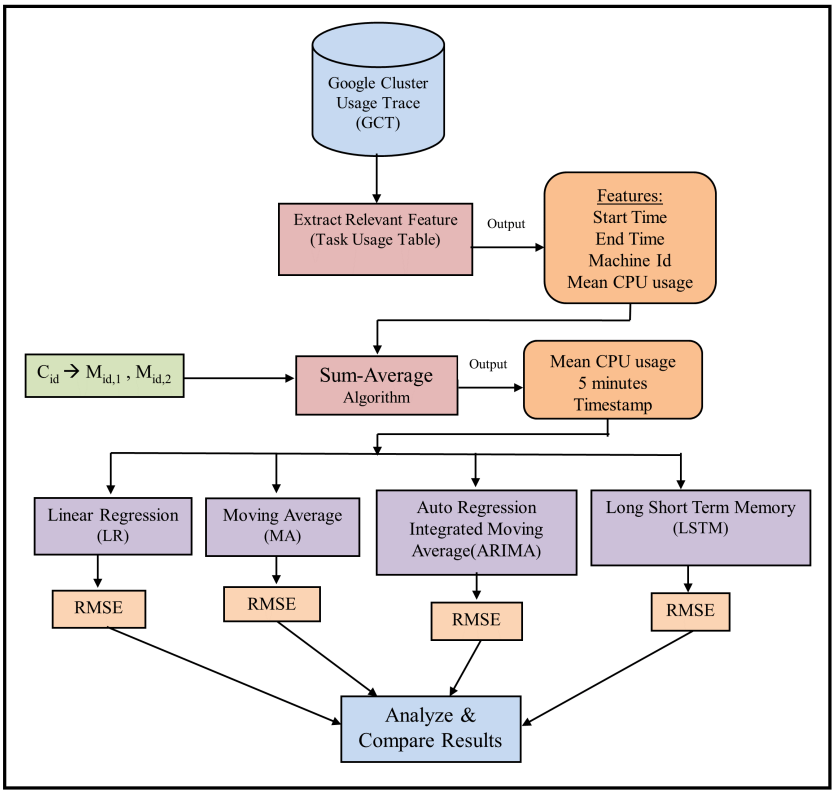


Figure 3.6: Prediction of mean CPU usage

process(Figure 3.4) and taken as the input to the proposed SA algorithm, and the output will be the mean CPU usage of a given PM for each 5 minutes time interval. Afterwards, LR, MA, ARIMA, as well as LSTM, were applied to predict the mean CPU usage. We compared the RMSE of the LSTM model with linear regression, moving average as well as ARIMA-RMSE results. The same process was repeated for the number of clusters * 2.

3.5 Results and Discussion

The proposed technique is implemented using Python programming language (version 3.8.8) using Jupyter notebook [211]. The computer system used in all the experiments is configured with an Intel Xeon 2.60 GHz processor, Microsoft Windows 10 Professional 64-bit operating system, and 16 GB physical memory. The experiments are performed on the task resource usage table of the Google cluster trace version 2.0 dataset.

In this section, the results of the proposed model are demonstrated, and the obtained results are compared with other traditional techniques like LR, MA, and ARIMA models. GCT dataset consisting of mean resource usage corresponding to tasks of each PM. To predict the resource utilization of a PM, the mean CPU usage of each PM is calculated

using the SA algorithm. In step 3.4.1 of the proposed methodology, i.e., data preprocessing, takes some selected attributes of task resource usage table, i.e, Start_time, End_Time, Machine_ID, Mean_CPU_usage and Canonical_memory_usage as input and gives a record of mean CPU as well as memory usage corresponding to every PM id.

In step 3.4.2 of the proposed methodology, Hopkins statistic is applied to normalized data and figure out its clustering tendency. The achieved Hopkins score of the normalized data is 0.96, i.e., (> 0.5), which shows the evidence that data is more likely to be clustered than uniformly distributed random values. Afterwards, the silhouette score is applied to the normalized dataset for a different number of components with k ranging from 2 to 19 to figure out an optimal number of possible clusters in the dataset. With the help of figure 3.7, we can observe that the silhouette score for k = 3 is 0.416, for k = 4, is 0.460, and for k = 5 is 0.387. The highest Silhouette score i.e. nearest to 1, indicates the optimal number of clusters. Here, K = 4 obtains the highest Silhouette score. Due to that, we consider 4 to be the optimal number of clusters to divide all the PMs of the dataset.

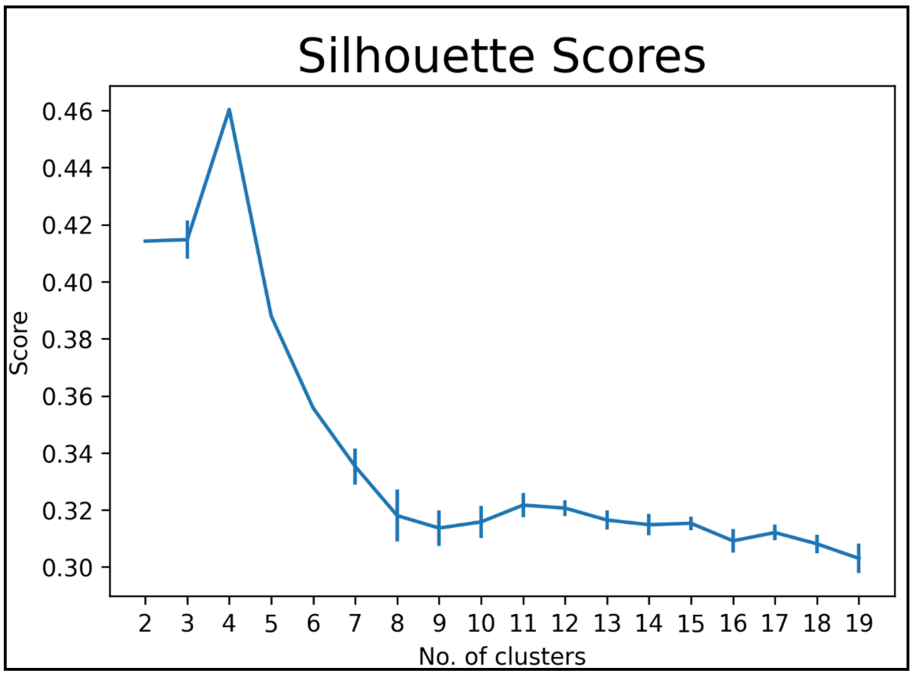


Figure 3.7: Silhouette Score

Afterwards, GMM is employed to divide all the PMs of the data center into four groups based on mean CPU and canonical memory usage. GMM helps to assign each PM of the data center to one specific cluster. Figure 3.8 depicts the division of four clusters using yellow, blue, violet, and teal colors. Each color of figure 3.8 represents a cluster. In the GCT dataset, a total of 12497 PMs are in active states, and GMM helps to divide all active

PMs of the data center into clusters. Two machines are selected from each group based on mean and median CPU usage. These two selected machines are expected to represent the entire cluster because of the similar characteristics of the clustering technique. Table 3.2

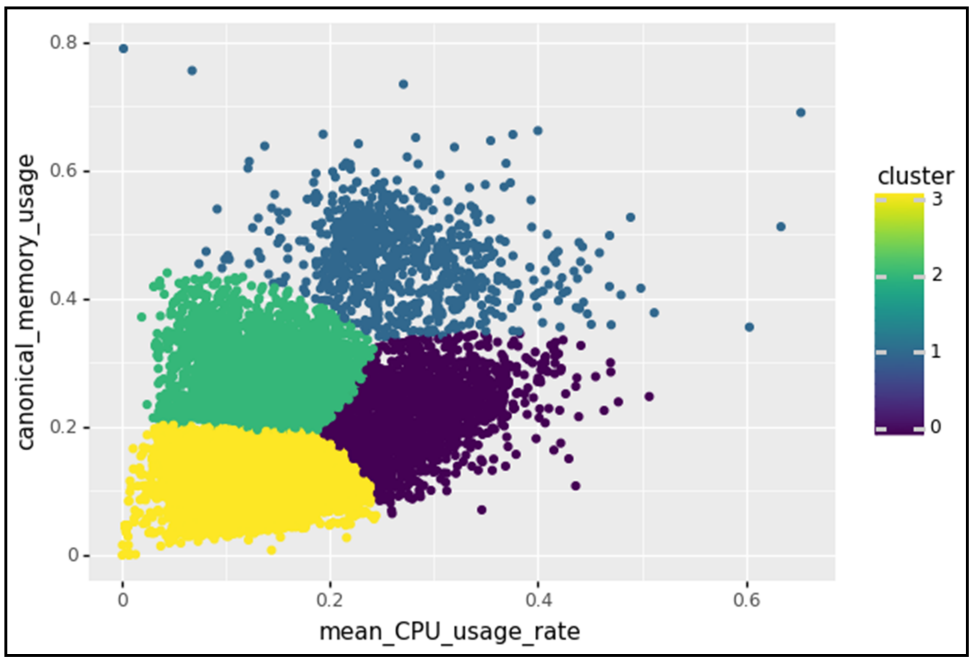


Figure 3.8: Clusters of GCT dataset PMs

represents cluster id, the number of machines mapped into the corresponding cluster with mean and median selected machine-id. Finally, eight machine IDs are collected from the entire dataset with different CPU usage for further mean CPU usage prediction process. The TRU data is processed and extracted from selected machine mean CPU usage along with a time stamp from each cluster for prediction using the SA algorithm.

Table 3.2: Selected Machine Id corresponding to Cluster Id

Cluster ID	**Total machines in Cluster**	**Machine 1**	**Machine 2**
0	1669	257348765	2893768826
1	777	4820254605	4820099550
2	3763	7716048	3137836022
3	6288	4337998263	317486421

In step 3.4.3 of the proposed methodology, the univariate LSTM model is employed to predict mean CPU usage for eight selected PMs mentioned in table 3.2. To predict the mean CPU usage of each selected PM, firstly, extract task-related information of a

particular machine from the historical workload, i.e., the TRU table of the GCT dataset. Afterwards, extract relevant features which are required to predict the mean CPU usage of the machine, i.e., Start_Time, End_Time, Machine_ID, mean_CPU_usage. The proposed SA algorithm (algo 3.1) is then applied to convert task mean CPU usage information into machine mean CPU usage for every 5 minute time interval of 29 days. All machine-related data is divided into 7:3, i.e., 70% of data is employed to train the model, while the rest 30% of information is used for testing. LSTM model is used on 5,15,30,60, and 120 minutes prediction window size (PWS) for each selected PM and noticed that the best result, i.e., minimum RMSE achieved with 30 minutes of PWS. LSTM achieved the best results by using the following parameters for all PMs, i.e., one layer of LSTM with 'tanh' as an activation function, 30 hidden neurons, and 80 epochs.

Figure 3.9(a) and Figure 3.9(b) depict mean CPU usage prediction carried out with a PWS of 30 minutes training and testing of machine id 257348765 and 2893768826 from cluster-id 0 with testing RMSE of 0.061 and 0.047 respectively. Figure 3.10(a) and Figure 3.10(b) show the best prediction results of machine id 4820254605 and 4820099550 associated with cluster-id 1 with testing RMSE of 0.058 and 0.059 respectively. Figure 3.11(a) and Figure 3.11(b) represent mean CPU usage prediction of cluster-id 2 selected machines, i.e., 7716048 and 3137836022 with testing RMSE of 0.042 and 0.061 respectively. Figure 3.12(a) and 3.12(b) represents prediction results corresponding to cluster 3 machine ids 4337998263 and 317486421 with testing RMSE of 0.055 and 0.022 respectively.

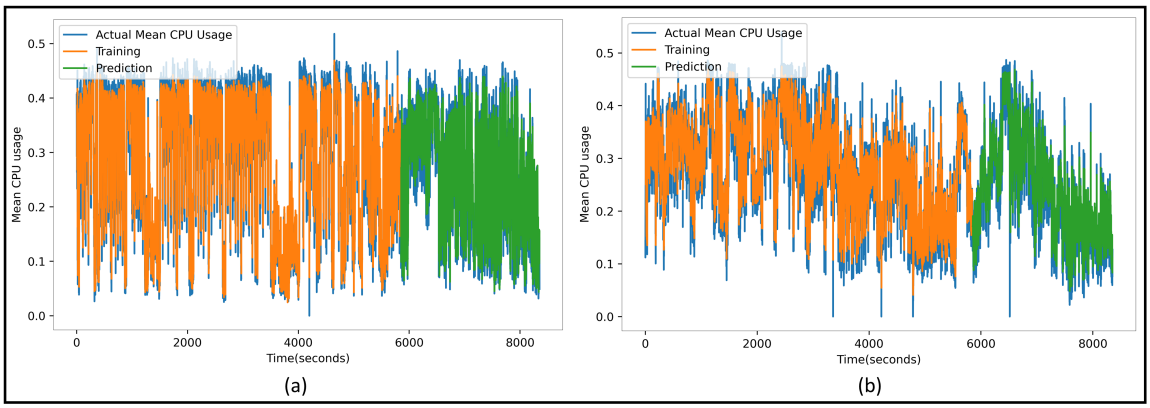


Figure 3.9: Mean CPU usage prediction for Machines of Cluster Id 0 with sliding size = 6(a) Machine Id 257348765 (b) Machine Id 2893768826

The x-axis of figures [3.9-3.12] represents the total time duration in seconds, i.e., 29 days, and the y-axis depicts the mean CPU usage of PM, which is measured in units of CPU-core seconds per second. The blue colour represents the actual PM mean CPU usage while the orange colour represents the prediction result on 70% of training data, and the

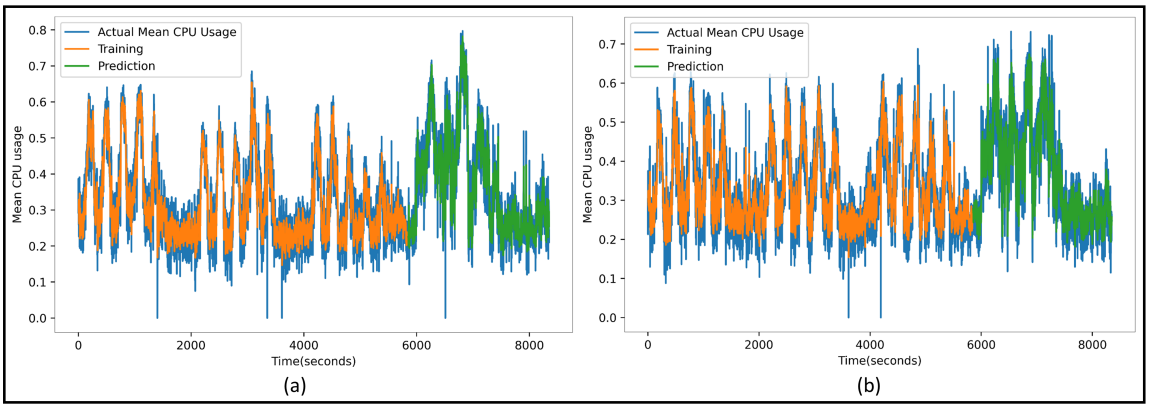


Figure 3.10: Mean CPU usage prediction for Machines of Cluster Id 1 with sliding size = 6(a) Machine Id 4820254605 (b) Machine Id 4820099550

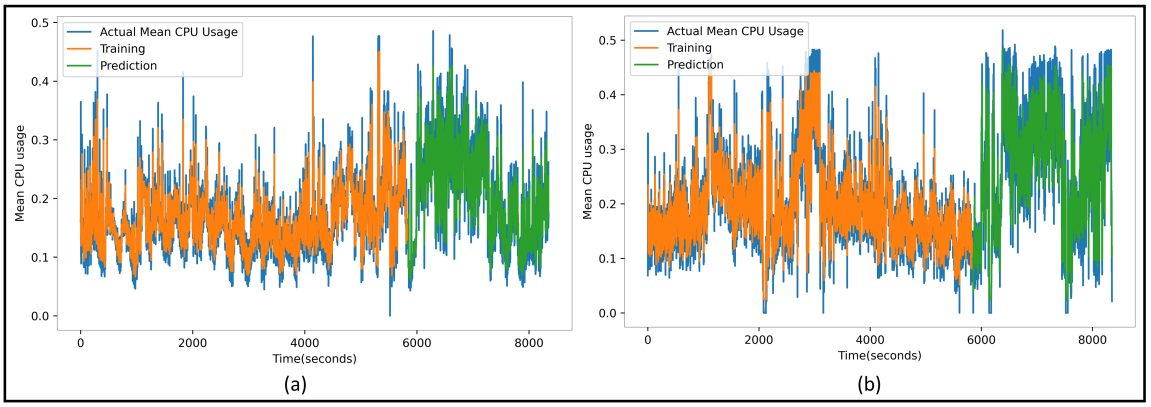


Figure 3.11: Mean CPU usage prediction for Machines of Cluster Id 2 with sliding size = 6(a) Machine Id 7716048 (b) Machine Id 3137836022

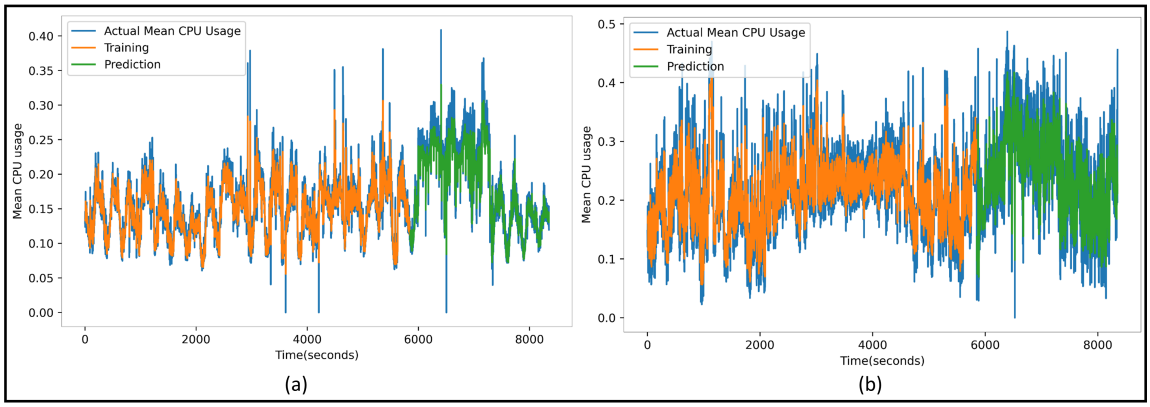


Figure 3.12: Mean CPU usage prediction for Machines of Cluster Id 3 with sliding size = 6(a) Machine Id 4337998263 (b) Machine Id 317486421

green colour represents the mean CPU usage prediction result on 30% of testing data with 30 minutes PWS.

Table 3.3: Prediction RMSE comparison

Machine ID	**LR**	**MA**	**ARIMA**	**The proposed model**
257348765	0.1997	0.1227	0.1124	0.06126
2893768826	0.1108	0.1428	0.1099	0.04733
4820254605	0.1870	0.1877	0.2259	0.05861
4820099550	0.1495	0.1528	0.1783	0.05911
7716048	0.0911	0.1091	0.1112	0.04266
3137836022	0.1696	0.1743	0.2343	0.06110
4337998263	0.1862	0.0988	0.1141	0.05510
317486421	0.1722	0.1075	0.1026	0.02210

RMSE of the proposed model is computed and compared with other time series models, i.e., LR, MA and ARIMA models. Table 3.3 states computed RMSE over test data for all sets of experiments. The analysis of experimental results for the compared model concluded that LR gives a better prediction result (less RMSE value) for four machines, i.e., 4820254605, 4820099550, 7716048 and 3137836022 compared to MA and ARIMA. For machines 257348765, 2893768826, and 317486421, ARIMA prediction is close to the actual prediction compared to MA and LR. For machine 4337998263, MA RMSE is lesser than other compared models. However, if compared to the proposed model, it is observed that the proposed model yielded the least RMSE values for all configured PMs, which means the proposed model outperforms the other tested models like LR, MA, and ARIMA.

3.6 Summary

The prediction of the forthcoming workload of a data center is the foremost requirement for efficient resource management. To achieve this, the GMM-LSTM approach is introduced in this chapter. First, the dataset is processed using a proposed SA algorithm to extract intact machine resource usage from task resource usage. Next, all the machines in the dataset are divided into clusters based on mean CPU and memory usage to find a rational set of heterogeneous machines from the dataset using the GMM algorithm. Lastly, the LSTM model is used with its optimal hyperparameter to predict the CPU usage of the selected machines. For this study, the GCT dataset is used and the achieved results demonstrate that the LSTM model effectively predicts short-term resource usage with minimal RMSE values compared to LR, MA, and ARIMA models across all selected

heterogeneous PMs. This capability facilitates the allocation and consolidation of VMs and contributes to reducing energy consumption in a CDC.

In this chapter, we delve into the LSTM model, renowned for its ability to capture short and medium-term dependencies in sequences. In the next chapter, attention-driven models Transformer and Informer are carried out to capture the impact of long-range dependencies and input sequence order for achieving higher accuracy of resource usage prediction. In addition, Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) clustering is used to capture the heterogeneity of PMs for building models

Chapter 4

Resource Usage Prediction leveraging Attention-Driven Models with BIRCH Clustering

Several machine learning approaches exist for the prediction of resource usage in a cloud environment, but they do not consider long-range dependencies. In this chapter, the LSTM model and attention-based mechanism models, i.e., Transformer and Informer, are used to predict the resource usage of PMs. The GCT dataset utilizes the BIRCH algorithm to intelligently select a set of diverse machines. Evaluating three models reveals that the Transformer architecture, which accounts for long-range dependencies in time series data, achieves a 14.2% *reduction in RMSE compared to the LSTM model. Despite this, the LSTM model performs better for certain machines, highlighting the significance of input sequence order. However, the Informer model, which combines the strengths of both LSTM and Transformer by considering dependencies and input sequence order, outperforms the other models. It achieves a* 21.7% *reduction in RMSE compared to LSTM and a* 20.8% *reduction compared to the Transformer. The results consistently demonstrate that the Informer model performs better across all dataset subsets. This indicates that incorporating both long-range dependencies and sequence ordering in resource usage predictions significantly enhances prediction accuracy.*

4.1 Introduction

With growing demand, a CSP must strive for infrastructure capacity planning and effective resource allocation for improved service performance and availability. Therefore, accurate resource usage prediction is crucial information for resource allocation, which helps in better managing the organization, improving system performance, and reducing operational costs [212, 213, 214, 215].

Recent studies on resource usage prediction used statistical, machine learning and deep-learning approaches. ARIMA [86, 102, 126], SARIMA [185, 216, 217], Exponential Smoothing [188, 218, 219], MA [220], Random Forest [221, 103], Support Vector Machines [222, 223, 224], Decision Trees [223], Neural Networks [225, 226, 227], Autoencoders [228, 229, 230], short term prediction models and LSTM are widely employed for time series prediction [185, 231]. However, these have a limited ability to capture non-linear trends and long-range dependencies. Further, they are sensitive to outliers, difficult to parallelize for fast processing, and tend to overfit noisy data [232, 233, 234]. Thus, It is vital for a model to handle the above shortcomings and should be tested on an extensive heterogeneous dataset for robust evaluation.

Transformer is a DL model that has recently gained importance due to its ability to capture the different parts of an input sequence. This model is based on self-attention mechanisms that allow it for efficient parallel processing and training on the long-range dependencies of a time series [235]. Such behaviour was missing in RNN and CNNs like LSTM [236]. These are trained sequentially and on backpropagation through time (BPTT), which can suffer from vanishing and exploding gradients [237]. However, in time series prediction, especially in resource usage of a PM, the input sequence order over time steps is an important consideration. During training, LSTM takes input at the current time step and the hidden state from the previous time step. The output at each time step is the sequence predicted value for the next observation. Thus, LSTM offers an advantage by preserving the input order over the Transformer, which is much more effective when the order is unimportant. Informer, another deep learning algorithm, offers both benefits by combining a self-attention mechanism, as in Transformer and a convolutional layer, as in LSTM. This allows the model to capture local and global dependencies and makes it suitable for time series prediction [238].

Accurately predicting the resource usage of PMs in the cloud is essential for efficient cloud resource management. This practice facilitates capacity planning, cost optimization, and performance enhancement. By accurately predicting resource needs, CSPs can avoid overprovisioning, optimize energy consumption, and meet service-level agreements [146].

Predictive analytics also aids in fault detection and prevention and contributes to a better user experience. Additionally, it enables auto-scaling mechanisms, ensuring that cloud resources dynamically adjust to meet fluctuating workloads, leading to improved overall reliability and responsiveness [189].

The contributions of the suggested approach can be summed up as follows:

- An analysis is presented to identify a best-suited model for time-series resource usage prediction in a cloud environment by understanding the influence of long-range dependencies and input sequence order on accuracy. This study compares the widely used LSTM model with the transformer and informer model.
- Identified the optimal hyperparameters for each model by a randomized search.
- Presented the methodology to extract a rational set of heterogeneous PMs based on mean CPU, memory and hard disk usage using the BIRCH clustering algorithm from the GCT dataset for evaluation and comparison of employed models.
- Identified the best window size with minimum RMSE by performing experiments with different window sizes.

The rest of the chapter is organized in the following manner. Section 4.2 describes the theoretical background of applied models. Section 4.3 describes the suggested methodology, which includes data pre-processing, selection of heterogeneous machines, and model implementation. Finally, Sections 4.4 and 4.5 present the results discussion of results and summary.

4.2 Theoretical Background

This section describes the DL models employed for resource usage prediction of PMs.

4.2.1 LSTM

The LSTM model architecture, its structure and operational mechanisms are described in subsection 3.4.3 of Chapter 3.

4.2.2 Transformer

The transformer utilizes a self-attention mechanism, enabling the model to assign varying degrees of priority to distinct segments of the input sequence during prediction [239]. This paradigm comprises two primary components: the encoder and the decoder. The

encoder analyses the input sequence and produces a concealed representation, which is subsequently transmitted to the decoder to generate the output sequence, as depicted in figure 4.1.

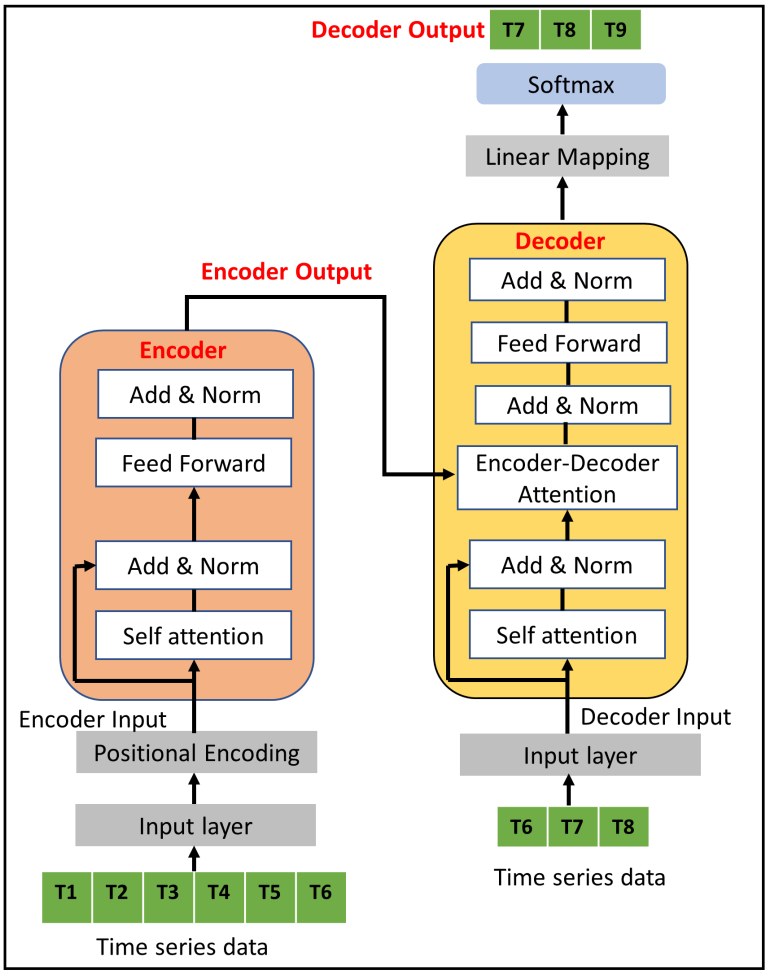


Figure 4.1: Architecture of Transformer model

The encoder layer is built up of several identical layers that are layered on top of one another. Each layer has two sublayers: a self-attention layer and a feed-forward layer. The self-attention layer determines the relative significance of various segments within the input sequence, while the feed-forward layer applies a non-linear conversion to the output of the self-attention layer. The equation defines the self-attention mechanism:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{4.1}$$

The Q (Query matrix) comprises queries derived from the input data. These queries serve the purpose of retrieving information from key-value pairs, where each row of Q represents a vector corresponding to the current input. The K (Key matrix) represents keys, also

derived from the input data in a manner similar to queries. Key vectors play a vital role in determining the degree of attention allocated to different segments of the input sequence when making predictions for a specific time step. Each row of K represents a key vector. The V (Value matrix) represents values, also derived from the input data. Key vectors encapsulate information from each position in the input sequence, which the model learns and utilizes for predictions. Each row of V corresponds to a value vector. The dimensionality of the key vector denoted as d_k, signifies the dimensionality of the key vectors. It is a scalar value representing this dimensionality. In the equation 4.1, d_k is employed to scale the dot product of Q and K before the application of the Softmax function. Dividing by the square root of d_k serves the dual purpose of stabilizing the training process and controlling the magnitude of the dot products.

The resulting weights are then used to compute a weighted sum of the values. After the self-attention layer, the output is passed through a feed-forward layer. The feed-forward layer is defined as follows:

$$FFN(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{4.2}$$

Here, x is the input to the feed-forward layer, W_1, b_1, W_2, and b_2 are learned parameters, and $\max(0, x)$ is the ReLU activation function.

Similarly, each layer in the decoder also consists of two sub-layers: a multi-head self-attention mechanism and a multi-head attention mechanism that attends over the encoder's hidden representation. The output of these sub-layers is passed through a feed-forward network. In addition, it includes a positional encoding mechanism that injects information about the position of each token in the input sequence into the model using equation 4.3 & 4.4.

$$PE(pos, 2i) = sin(\frac{pos}{2L^{(\frac{2i}{d}}})) \tag{4.3}$$

$$PE(pos, 2i+1) = sin(\frac{pos}{2L^{(\frac{2i}{d}}})) \tag{4.4}$$

The variable *pos* represents the position of the token in the input sequence. The variable i is the index of the dimension, and d_{model} represents the dimensionality of the hidden representations. Finally, in order to guarantee that the predicted value in a time series is solely based on previous data points, the decoder employs look-ahead masking and a one-position offset between the decoder input and target output.

4.2.3 Informer

The Informer model was introduced to overcome the shortcomings of the Transformer model with three distinct characteristics [238]. i) The ProbSparse self-attention mechanism is used to replace canonical self-attention, achieving a time complexity and memory usage of $O(L \log L)$, where L represents the input sequence length. This complexity is a significant improvement compared to the traditional $O(L^2)$ complexity that is associated with canonical self-attention. The reduction in complexity is accomplished through the introduction of probabilistic sparsity, wherein only a subset of elements is considered during attention computation. This innovative design choice enables the model to efficiently handle long input sequences, making the Informer model well-suited for scenarios where computational efficiency is paramount. In addition, the ProbSparse self-attention mechanism enhances the scalability of the architecture, enabling the model to handle sequences of different lengths while achieving better performance efficiently. ii) By reducing cascade layer input in half, the self-attention distillation effectively handles extremely long input sequences while highlighting dominating attention. iii) The generative style decoder utilizes a single forward operation to predict large time-series sequences rather than a step-by-step approach. This greatly improves the speed of inference for long-sequence predictions. The Informer architecture comprises two major parts: encoder and decoder, as shown in figure 4.2.

Firstly, input is prepared by employing token, positional, and temporal embedding on an encoder's original time series input sequence. The prepared data is provided to the ProbSpare self-attention block, which helps remove canonical self-attention and reduce the network size. Instead of using equation 4.1, ProbSpare self-attention uses equation 4.5.

$$Attention(Q, K, V) = softmax(\frac{\overline{Q}K^T}{\sqrt{d}})V \tag{4.5}$$

Afterwards, the encoder layer is designed to extract reliable long-range dependencies from lengthy sequential inputs. Encoder layers add convolution, activation, and maximum pooling operations between each encoder and decoder layer and cut the input sequence of the preceding layer into half to eliminate the issue of too much memory utilization. Similarly, the decoder receives a long input sequence with 0 padding elements. The procedure follows equation 4.6.

$$X_{j+1}^t = MaxPool(ELU(Conv1d([X_j^t]_{AB}))) \tag{4.6}$$

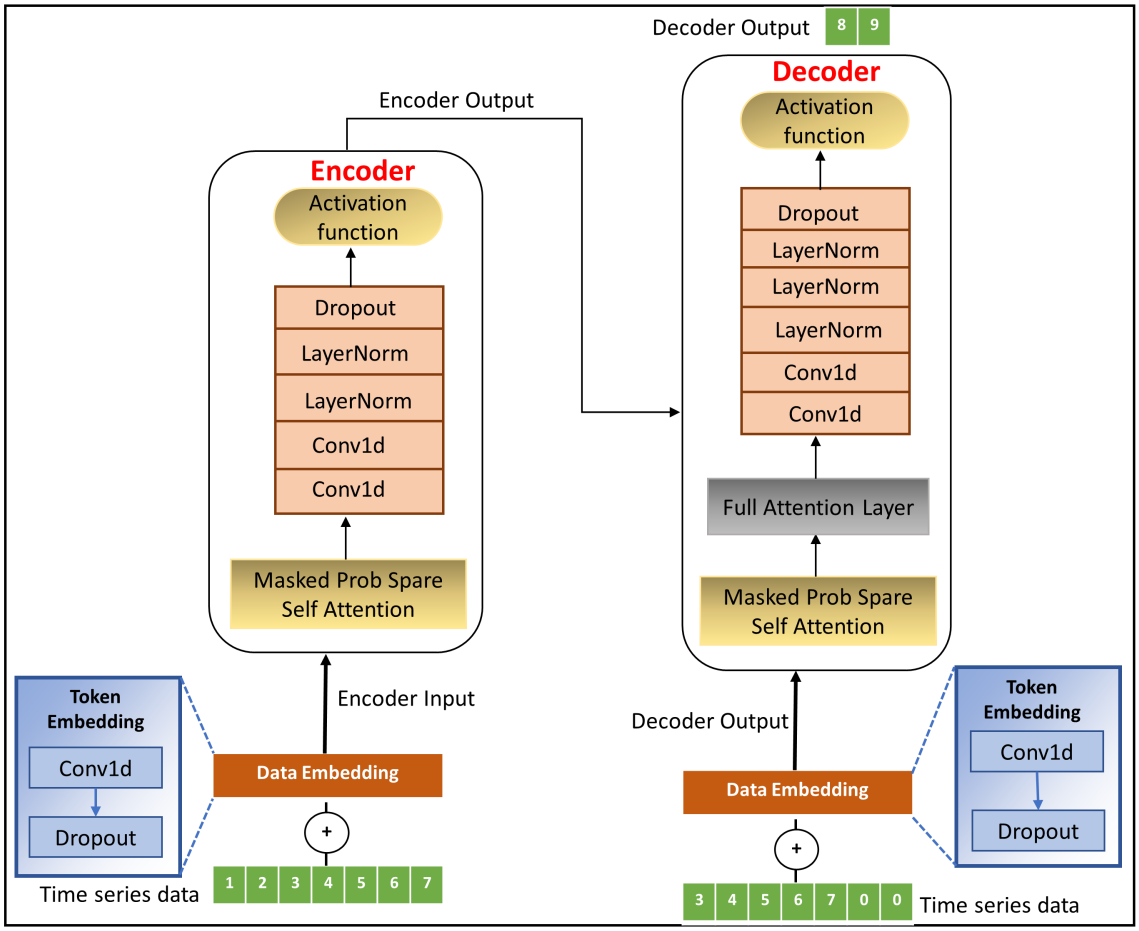


Figure 4.2: Architecture of Informer model

Where $[X_j^t]_{AB}$ is the calculation result of the multi-headed ProbSparse self-attention layer in the previous layer; ELU is the activation function; X_{j+1}^t is the current layer output of the multi-headed ProbSparse self-attention layer.

4.3 Proposed Scheme

This section provides a detailed explanation of the methodology used to predict the resource utilization of PMs in a CDC. As depicted in Figure 4.3, the process proceeds in the following manner: Initially, the dataset is pre-processed to meet the clustering requirements, facilitating the extraction of heterogeneous PMs based on their mean resource usage. Afterwards, with the selected machines identified, the data is further processed in accordance with the time series models. Finally, we employed LSTM, Transformer, and Informer models to predict the CPU usage of the machine. The primary objective of this methodology is to identify the most suitable models that can accurately predict CPU usage with minimal error, regardless of the configuration of PM.

The methodology is divided into three subsections: Data pre-processing, Selection of heterogeneous machines, and Model implementation.

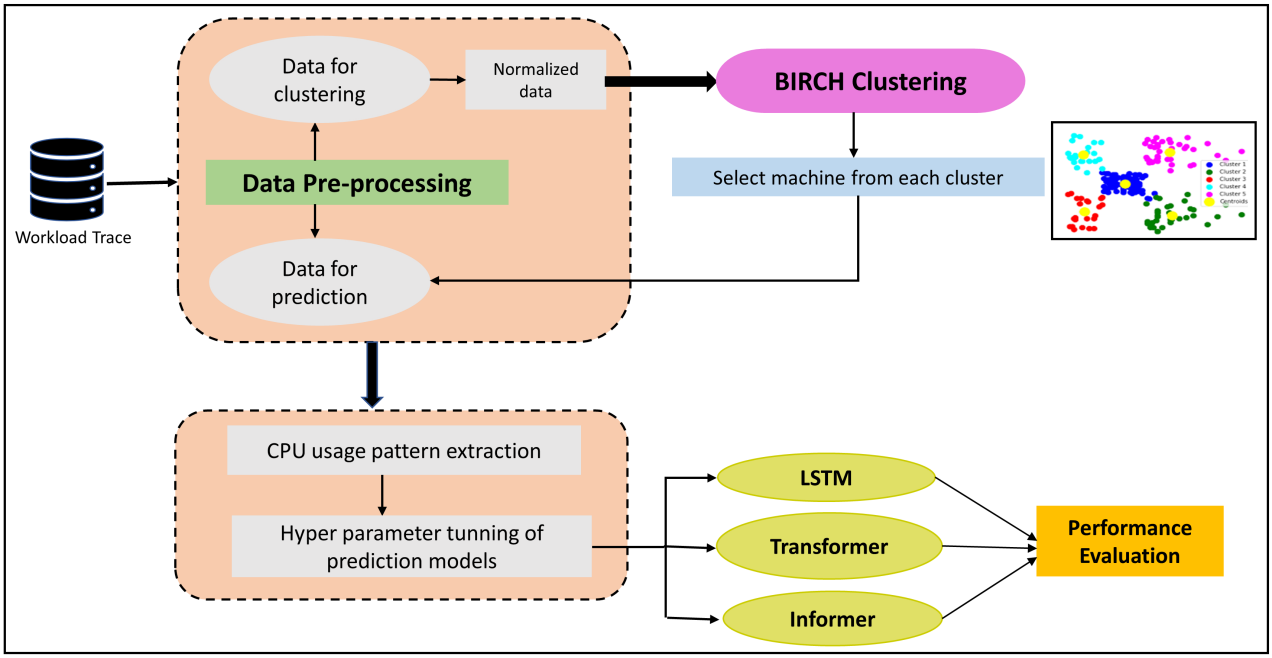


Figure 4.3: Methodology to predict CPU usage of PMs

4.3.1 Data Pre-processing:

The employed prediction models use the GCT dataset that is described in section 3.2. The GCT dataset encompasses attributes associated with resource categories, including CPU, memory, and hard disk specifications.

To initiate the process, firstly, extract a subset of heterogeneous machines from the dataset, with heterogeneity defined by the average resource usage of the machines. Select six features from the dataset: Start time, End time, Machine ID, mean CPU usage, canonical memory usage, and mean local hard disk space usage. The inclusion of Start time and End time is crucial for calculating 5-minute time intervals, as detailed in Section 4.3.1. Machine IDs were incorporated to provide unique identifiers, streamlining the grouping of machines efficiently. The features mean CPU usage, canonical memory usage, and mean local hard disk space usage are fundamental for computing the total average utilization of the corresponding resources. Only these features are required to serve as the basis for dividing machines into clusters.

Subsequently, to predict the mean CPU usage of PMs, we refined our feature selection further. For this objective, focused on three key features: Start time, End time, and

mean CPU usage. The 5-minute time intervals were computed using Start time and End time, while mean CPU usage was utilized to train prediction models. In pursuing this objective, we exclusively focus on predicting mean CPU usage. Hence, our feature selection is centered solely around the inclusion of the mean CPU usage feature.

Regarding the remaining features in the dataset, we intentionally omitted certain attributes that were not directly aligned with our study's objectives. Our focus on the selected six features ensures that we prioritize the most relevant information for clustering machines based on mean resource usage and three features for predicting mean CPU usage.

Pre-Processing

Data pre-processing is the primary step towards the accurate resource prediction of a PM. Adequate pre-processing helps to clean the data and keep it consistent and meaningful [240]. The resource usage table consists of task resource usage information instead of PM information, but the notion of research is to predict the CPU usage of a PM. It is also observed that resources are shared between multiple tasks in the same period.

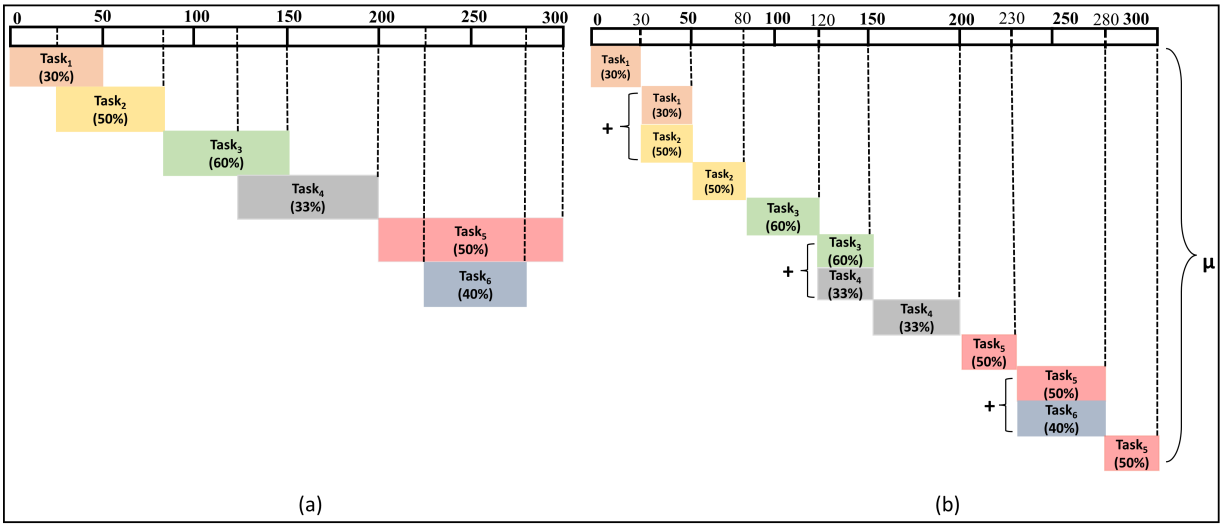


Figure 4.4: (a) Utilization of CPU by a task (b) Procedure use by Sum-Average algorithm

Therefore, calculate the mean resource usage of each PM available in the dataset for clustering and calculate the mean CPU usage of the PM over a 5-minute interval for prediction. First, extract the dataset's start time, end time, canonical memory usage, mean CPU usage, and mean disk usage features for pre-processing and employ the SA. Figure 4.4 depicts the procedure followed to calculate the mean CPU usage of a machine over 5 minutes (300 seconds) using the SA algorithm. Figure 4.4 (a) shows six tasks

running over a single machine. Here, $task_1$, $task_2$, and $task_6$ consume 30%, 50%, and 40% CPU, respectively, for 50 sec at the different time stamp. CPU usage of $task_3$ is 60%, and $task_4$ is 33% for 70 sec and 80 sec, respectively. $task_5$ consumes 50% of the CPU for 100 sec to complete it. Figure 4.4 (b) shows how the SA algorithm divides time intervals based on the sharing of CPU resources between multiple tasks. The benefit of division is calculating the intact value of total mean CPU consumption. Here, $task_1$ uses the CPU for 30 sec individually out of 50 sec, and the rest of 20 sec is shared with $task_2$. Hence, total CPU consumption from 30 sec to 50 sec is 80%. Similarly, $task_3$ processes for 40 sec individually, i.e., from 120 sec to 150 sec, and the rest of the 30 sec CPU is shared with $task_4$. Similarly, $task_5$ and $task_6$ share the CPU from 230 sec to 280 sec. Lastly, after aggregating shared CPU usage, SA calculates the mean of total CPU utilization corresponding to the PM.

4.3.2 Selection of Heterogeneous Machines

The dataset comprises $12.5K$ heterogeneous machine information, so predicting resource usage for all available PMs in the data center is computationally expensive and time-consuming. Thus, applying prediction models to every PM to check model performance is a cumbersome task for academia and researchers. This step also helps to train and evaluate prediction models' performance on complex heterogeneous patterns. Therefore, employ the clustering strategy and select a rational set of heterogeneous configured PMs from an entire dataset based on CPU, RAM, and hard disk utilization [241]. Figure 4.5 depicts the procedure for selecting machine IDs from the available PMs. Firstly, the start time, end time, machine ID, canonical memory usage, mean CPU, and hard disk usage are extracted, and the SA algorithm is employed to find the mean resource usage of all the machines for 29 days. The processed data is normalized using Z normalization, which helps to ensure that all features are on the same scale by considering the mean of all data points as 0 and the standard deviation as 1 [242]. Afterwards, Hopkins statistics is employed on normalized data to check whether clustering is an effective approach for the data or not, using probability estimation by checking the clustering tendency. If Hopkins statistics $(H) > 0.5$, it means clustering is a practical approach; otherwise, the data does not qualify for clustering [243]. Hopkin's result is more significant than 0.5, suggesting that we can create groups to divide entire machines based on CPU, memory, and hard disk usage.

Finally, the BIRCH clustering algorithm is employed to assign each available to a specific cluster. Therefore, analyze the data using silhouette analysis to find the optimal number for clustering and select two machines from each cluster to represent that cluster [244].

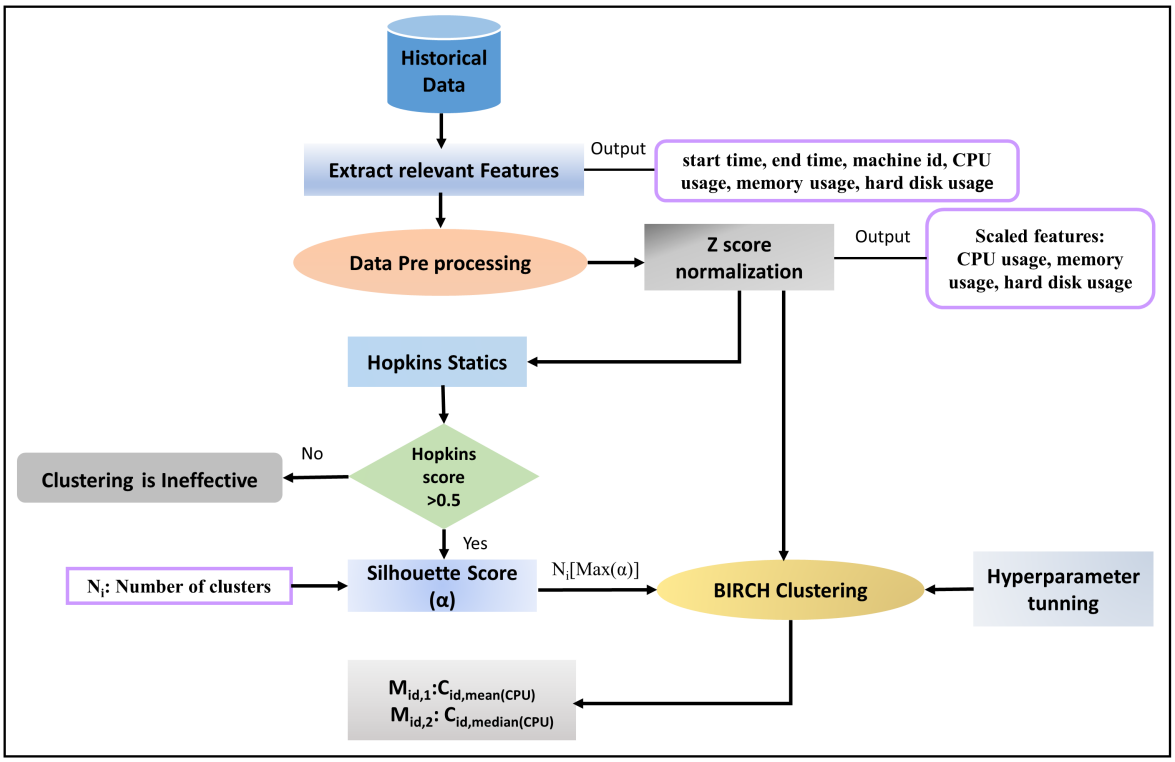


Figure 4.5: Selection of heterogeneous machines

Each cluster is represented by the machines that are closest to its mean and median of mean CPU usage. Further, the prediction of mean resource usage is conducted using these retrieved machine IDs. As a result, we collected two PMs from each cluster; therefore, gathered the total number of PMs is twice the number of optimal clusters.
Algorithm 4.1 demonstrates the procedure followed to divide all PMs into clusters and which machines are selected for the prediction step.

4.3.3 Model Implementation

For implementation, firstly, extract the features from data F={start time, end time, and mean CPU usage} of collected heterogeneous machines using clustering. Afterwards, the SA algorithm is used to pre-process the data to calculate the mean CPU usage of PMs over a 5-minute interval. Lastly, the employed LSTM, Transformer, and Informer are trained and tested with processed data, and their performance is analysed based on the RMSE. The experimental environment setup utilized in this study remains consistent as in Chapter 3.5.

Algorithm 4.1 clsMch: Machine selection to represent the entire available machines in a data center using clustering technique

Input: $[D] = \left(d_{pq}\right)_{p=1,q=1}^{p=k,q=l}$ Where, $d_{p,q} = q^{th}$ feature of p^{th} file, k = files in task resource table table, l= number of features in k. **Output:** $ID_{\{c,1\}} = \left\{M_{\{C_p,mean\}}\right\}_{p=1}^{k}$, $ID_{\{c,2\}} = \left\{M_{\{C_p,median\}}\right\}_{p=1}^{k}$ Where, C_p = Cluster number, M = Machine Id, $mean$ = machine nearest to the mean of cluster, $median$ = machine nearest to the median of cluster, k = total number of clusters.

Initialize an empty matrix G and store features of time series data from D data set
$G = (y_1, y_2, y_3, y_4, y_5, y_6)$ where, y_1 = start_time, y_2 = end_time, y_3 = machine_id, y_4 = mean_CPU_usage, and y_5 = canonical_memory_usage, y_6 = mean_diskspace_used.
$Datapre - processing(G) \rightarrow$matrix(T)
$(T_{ij} - \mu)\sigma \rightarrow T_{\text{new}}$ where, i-th row and j-th column
$Hopkins(T_{new}) \rightarrow H$
if $H > 0.5$ **then**
    $Silhouette\_score(T_{new}, n) \rightarrow S$ where, n = 4,5,6,7,8,9,10,11
    Find n corresponding to max(S) $\rightarrow N$
    $BIRCH(T, branching\_factor, N, threshold) \rightarrow C_k$ *where,* $k = 1$—N
    for j from 1 to N **do**
        Extract $C_j(T) \rightarrow F_j$
        Calculate $F_j(y_4).mean() \rightarrow \alpha_j$
        Find $F_j(y_3)$ whose $F_j(y_4)$ equal to α_j or nearest to α_j
        Calculate $F_j(y_4).median() \rightarrow \beta j$
        Find $F_j(y_3)$ whose $F_j(y_4)$ equal to βj or nearest to βj
    end for
else
    Clustering is an ineffective approach for T data
end if

Model Setup

The processed data of PMs are divided into a 7 : 3 ratio, where 70% of the data is used to train models and the rest 30% is used for testing the performance of applied models. We implemented each model over 2 (10 minutes), 6 (30 minutes), 12 (1 hour), 18 (1.5 hours), 24 (2 hours), and 48 (4 hours) input window sizes to investigate the effect of input sequence dependencies and compared their performance to determine the optimal size. Further, to determine the optimal hyperparameters for each model, a randomized search is conducted over a predefined range of values for learning rate, batch size, epoch and the number of hidden units. We examine multiple combinations of hyperparameters on the test set of models and choose the configurations that produce the best results. In addition, we trained the Informer model on combinations of token, positional and temporal embedding layers and compared their results to determine the optimal arrangement. The significance of the proposed approach is to provide insight into the impact of input sequences and long-range dependencies on accurate prediction of CPU utilization, regardless of the machine's configurations. By implementing this methodology, the objective is to determine the most efficient model for accurately predicting CPU utilization across various machine configurations. This analysis enables the assessment of the effectiveness of LSTM, Transformer, and Informer models in capturing the complexity of input sequences and long-range dependencies. Ultimately, it helps to identify the most suitable model for accurate prediction of CPU usage on diverse PMs.

4.4 Results and Discussion

4.4.1 Selection of heterogeneous machines

The achieved Hopkins statistic of 0.9549 on the normalized resource usage data confirms the possible grouping of PMs. We calculate the silhouette score from k= 2 to 12 range to determine how closely a machine resembles its group compared to other clusters. According to silhouette statistics, the number of clusters corresponding to the highest score is considered an optimal number for clustering. Figure 4.6(a) depicts that a silhouette score of 0.61 is highest at $k = 5$. Therefore, the BIRCH algorithm is employed to divide the entire PMs of the Google cluster into five groups based on CPU, memory, and hard disk usage. Six features are used to divide the machines into clusters: start time, end time, machine ID, mean CPU usage, canonical memory usage, and mean local disk space usage. Figure 4.6(b) depicts that all available PMs are seamlessly divided into five clusters using a 3D image where each colour denotes a specific cluster, and the axis indicates the resource's type.

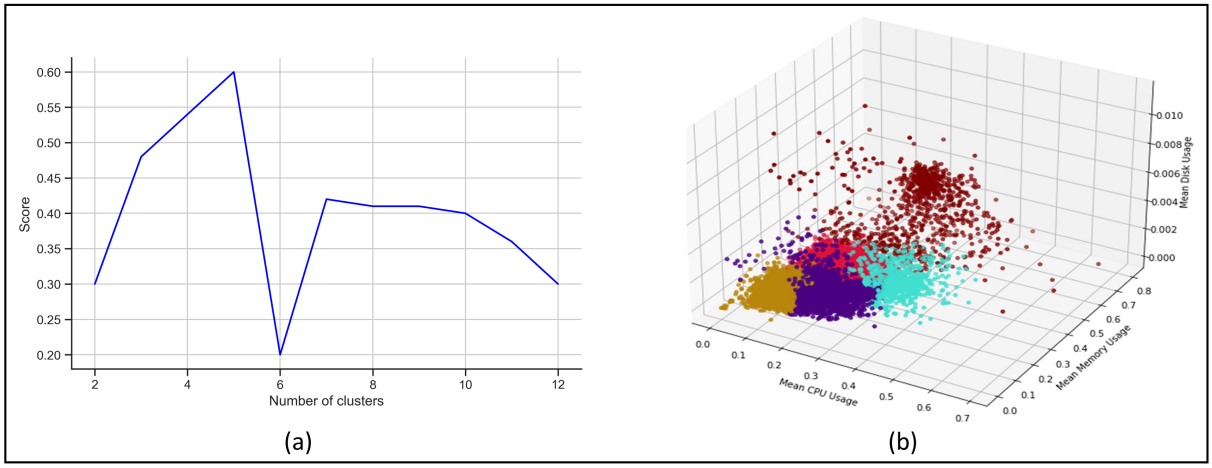


Figure 4.6: (a) Number of clusters vs Silhouette score (b) Clusters formed using BIRCH clustering technique

Afterwards, calculate each cluster's mean and median CPU usage and extract the machine ID whose CPU usage is near the computed values. Lastly, ten machine IDs are extracted, i.e., two machine IDs from each cluster for further analysis. Table 4.1 represents the selected machine IDs corresponding to their cluster number. Here, machine 1 and machine 2 attributes denote machine ID near the mean and median of CPU usage, respectively.

Table 4.1: Selected machine ids corresponding to cluster id

Cluster ID	**Machine 1**	**Machine 2**
1	4820094424	257335556
2	38708463	306881505
3	1676250332	1338630
4	351653579	3365958041
5	1335688	317486393

4.4.2 Analysis of Informer embedding layers

Firstly, the Informer model is employed with a token, positional, and temporal embedding layer to predict the mean CPU usage of PMs and achieve high RMSE, as shown in

figure 4.7. Further, we tested the Informer model with a token and positional embedding layer and observed improved results, as the predicted value is too close to the actual value. At last, the Informer model is employed with a token embedding layer, and the results outperformed each selected machine. Therefore, we consider only the token embedding layer of the Informer model to predict the mean CPU usage of PM. Figure 4.7 represents the achieved RMSE value of mean CPU usage prediction for machine ID 4820094424 after testing with 3 (token, positional, and temporal), 2 (token and positional), and 1 (token) embedding layers over 2, 6, 12, 18, 24, and 48 input window sizes.

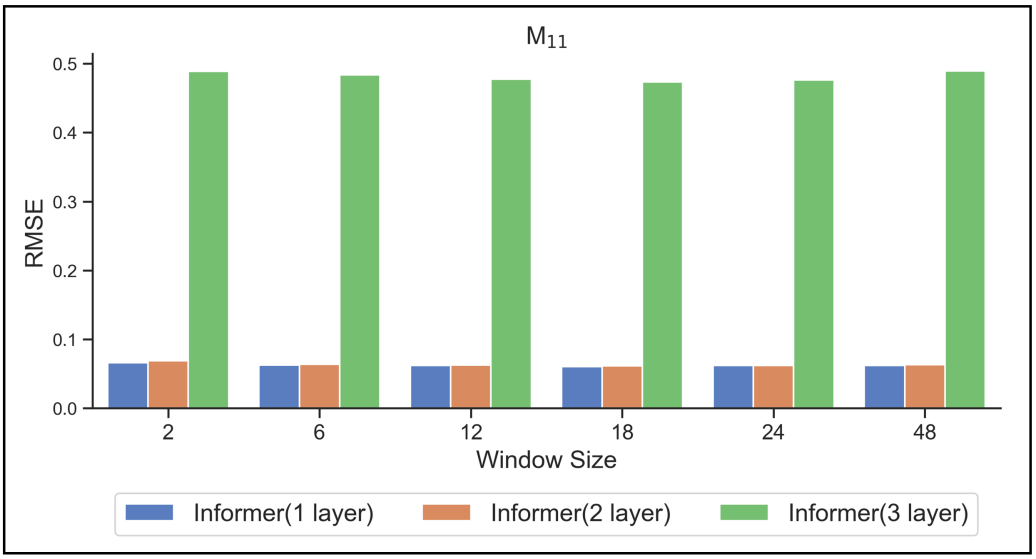


Figure 4.7: RMSE comparison between token, positional, and temporal embedding layer of Informer

4.4.3 Optimal hyperparameter of models

The experiments are carried out with various hyperparameters for LSTM, Transformer, and Informer to find the optimal parameters for the prediction of mean CPU usage. After conducting multiple trials, the following hyperparameter values were identified as resulting in the best performance. The optimal hyperparameters for LSTM, Transformer and Informer are described in table 4.2, 4.3, and 4.4, respectively.

Table 4.2: LSTM hyperparameters

Parameter	**Value**	**Parameter**	**Value**
Epoch	80	Batch size	64

Hidden layers	30	Optimizer	Adam
Activation function	relu	Dropout	0.01

Table 4.3: Transformer hyperparameters

Parameter	**Value**	**Parameter**	**Value**
Epoch	80	Batch size	64
Feature size	250	Number of layers	1
Dropout	0.01	Number of heads	10
Max length	5000	Learning rate	0.0001

Table 4.4: Informer hyperparameters

Parameter	**Value**	**Parameter**	**Value**
Epoch	10	Batch size	64
Time features encoding	minute	Features	S
Encoder layer	1	Decoder layer	1
dmodel	256	nhead	8
Device	GPU	Dropout	0.05
Learning rate	0.0001	Attention	Prob
Embedding Layer	Token embedding	Patience	5
Sequence length	2/6/12/18/24/48	Label length	2/6/12/18/24/48

4.4.4 Performance comparison of implemented models

The achieved RMSE values for each model during mean CPU usage prediction were plotted to compare their performances. Here, M_{ij} denotes j^{th} machine of i^{th} cluster where, $i \in \{1, 2, 3, 4, 5\}$ and $j \in \{1, 2\}$.

Figure 4.8 shows the achieved RMSE of LSTM, Transformer and Informer over 2,6,12,18,24, and 48 input window sizes for selected machines of cluster 1. Figure 4.8 (a) depicts that for machine 4820094424, LSTM and Transformer achieved the lowest RMSE score of 0.068562 and 0.066298 at window size 12, whereas, Informer got RMSE of 0.06185. The Informer performs best at window size 18 with an RMSE of 0.060492. Figure 4.8 (b) represents the RMSE difference between Transformer and Informer with respect to the LSTM model in percentages ($\Delta RMSE_{Model-LSTM}$) for machine 4820094424. The result shows that at window size 2, the change in RMSE of the Transformer is almost negligible. It shows that the Transformer and Informer reduced RMSE at each input window size whereas, at window size 18, the Informer performs best by reducing RMSE 14% compared to LSTM. Figure 4.8 (c) depicts the results of machine id – 257335556. Here, LSTM and

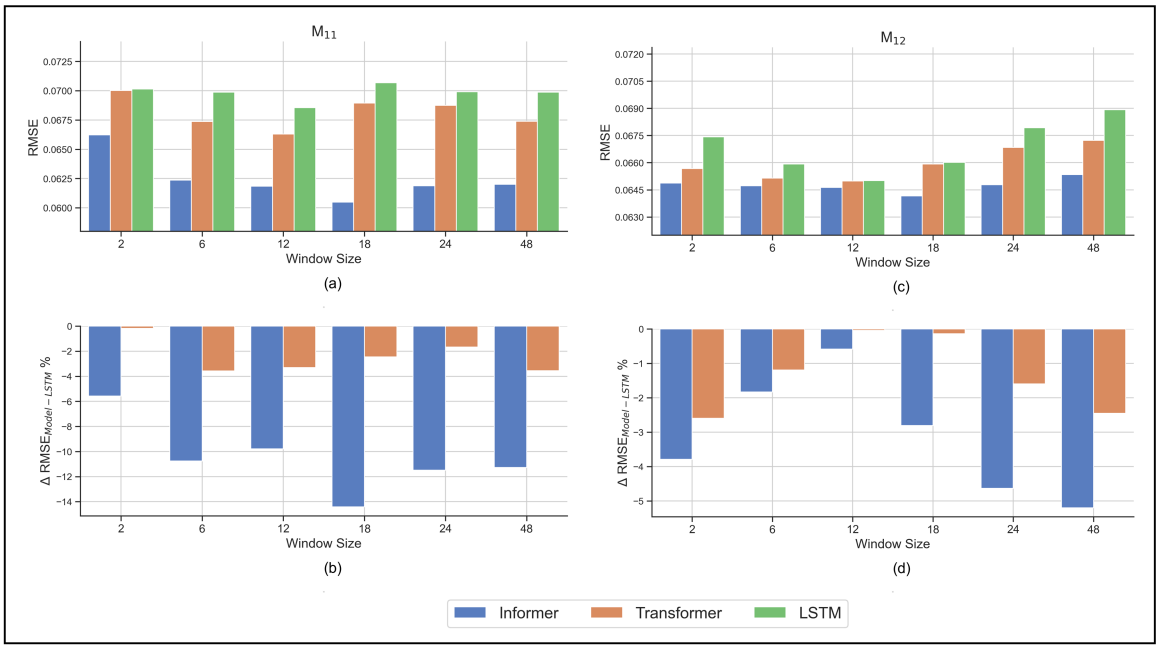


Figure 4.8: Comparison between the models using RMSE for different window sizes (a,c) RMSE for machines 4820094424 and 257335556; (b,d) Change in RMSE of Informer and Transformer with respect to LSTM for machines 4820094424 and 257335556

Transformer achieved the lowest RMSE at window size 12, with 0.065014 and 0.064987, respectively, and Informer achieved 0.064635. The Informer performs better at window size 18 with an RMSE of 0.06416. Figure 4.8 (d) shows that for machine 257335556, the RMSE difference between Transformer and LSTM is small at window sizes 12 and 18. The Informer and Transformer reduce RMSE for each window size, whereas, at window size 48, they reduced the most by 5.5% and 2.8% compared to LSTM. Figure 4.9 (a)–(b)

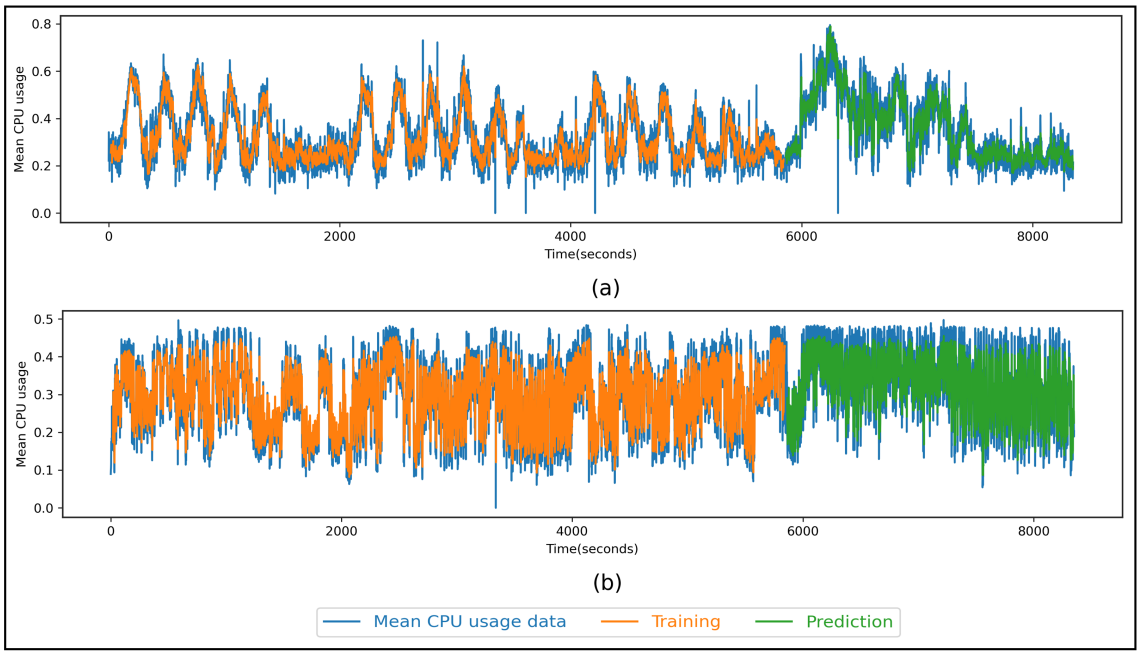


Figure 4.9: Actual and predicted mean CPU usage by Informer at window size = 18 (a) Machine 4820094424 (b) Machine 257335556

shows the achieved pattern of training and predicting mean CPU usage by Informer over actual observation at window size 18 for machines 4820094424 and 257335556, respectively.

Figure 4.10 represents the RMSE of employed models over 2,6,12,18,24 and 48 input window sizes for selected machines of cluster 2. Figure 4.10(a) shows the results of machine 38708463; LSTM achieved the lowest RMSE of 0.000592 at window size 6, whereas Transformer performed best at window size 12 with an RMSE of 0.000518. The Informer outperforms with an RMSE of 0.000473 at window size 18. It is observed that the RMSE of Informer decreases as the window size increases to 18 and then keeps increasing. Figure 4.10 (b) shows that at every window size, Informer and Transformer reduced RMSE compared to the LSTM model for machine 38708463, and Informer performed best by lowering the RMSE by 28%. However, at window sizes 6 and 18, the RMSE difference between Transformer and LSTM is small.Figure 4.10 (c) depicts the achieved RMSE of machine 306881505, where LSTM and Transformer perform best at window size 12 with

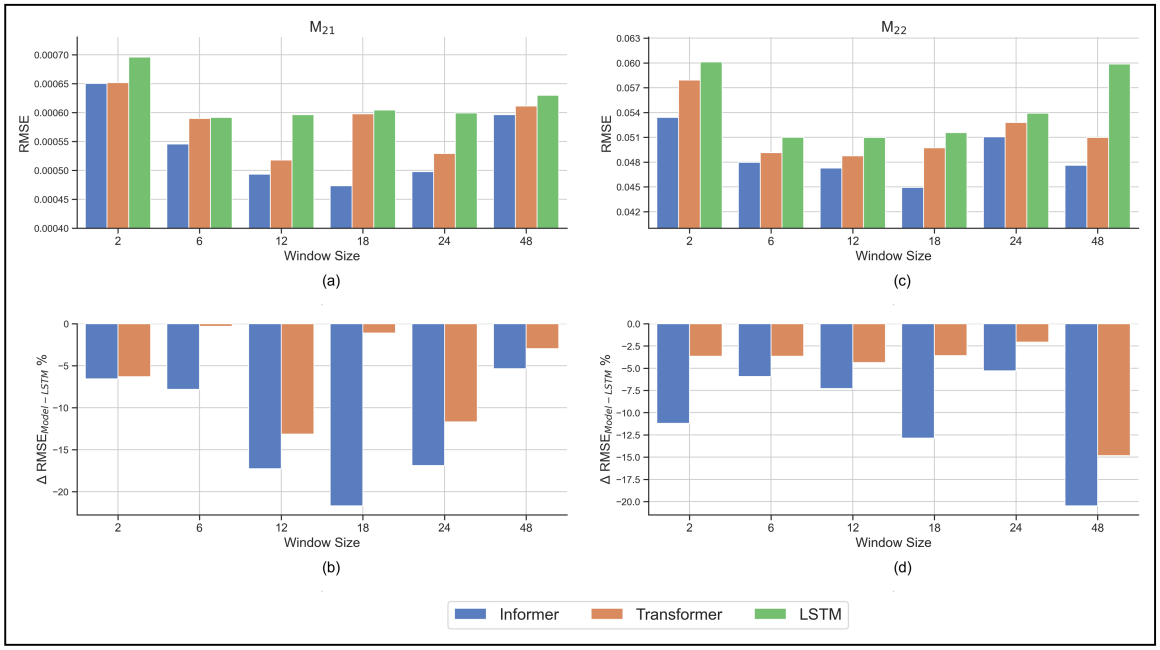


Figure 4.10: Comparison between the models using RMSE for different window sizes (a,c) RMSE for machines 38708463 and 306881505; (b,d) Change in RMSE of Informer and Transformer with respect to LSTM for machines 38708463 and 306881505

an RMSE of 0.050983 and 0.048754, respectively, and the Informer model gets 0.047271. At window size 18, the Informer model outperforms with an RMSE of 0.04493. Figure 4.10 (d) shows that the RMSE of the Informer and Transformer model is less than the LSTM model at each window size for machine 306881505. The Informer model decreases RMSE by 22% at window size 48. Figure 4.11 (a)-(b) displays the actual, trained and predicted mean CPU usage patterns achieved by Informer at window size 18 for machines 38708463 and 306881505, respectively.

Figure 4.12 represents the RMSE of applied models over input window sizes of 2, 6, 12, 18, 24, and 48 for selected machines of cluster 3. Figure 4.12 (a) represents LSTM performs best at window size 12 with an RMSE of 0.06593, and the Transformer model performs best at window size 6 with an RMSE of 0.06419 for machine 1676250332. Meanwhile, Informer outperforms other models in terms of window size, 18, with an RMSE of 0.06300. Figure 4.12 (b) shows that Transformer and Informer reduced RMSE at every input window size compared to the LSTM. Transformer and Informer deduced RMSE by 5.7% and 6.8%, respectively. At window size 12, the difference of Transformer and LSTM is negligible. Figure 4.12 (c) depicts that LSTM performs best at window size 6 and Transformer at window size 12 with RMSE of 0.05316 and 0.05189, respectively, for machine 1338630. Here Informer performs best compared to other employed models at window 18 with an RMSE of 0.04959. Figure 4.12 (d) shows that the Transformer

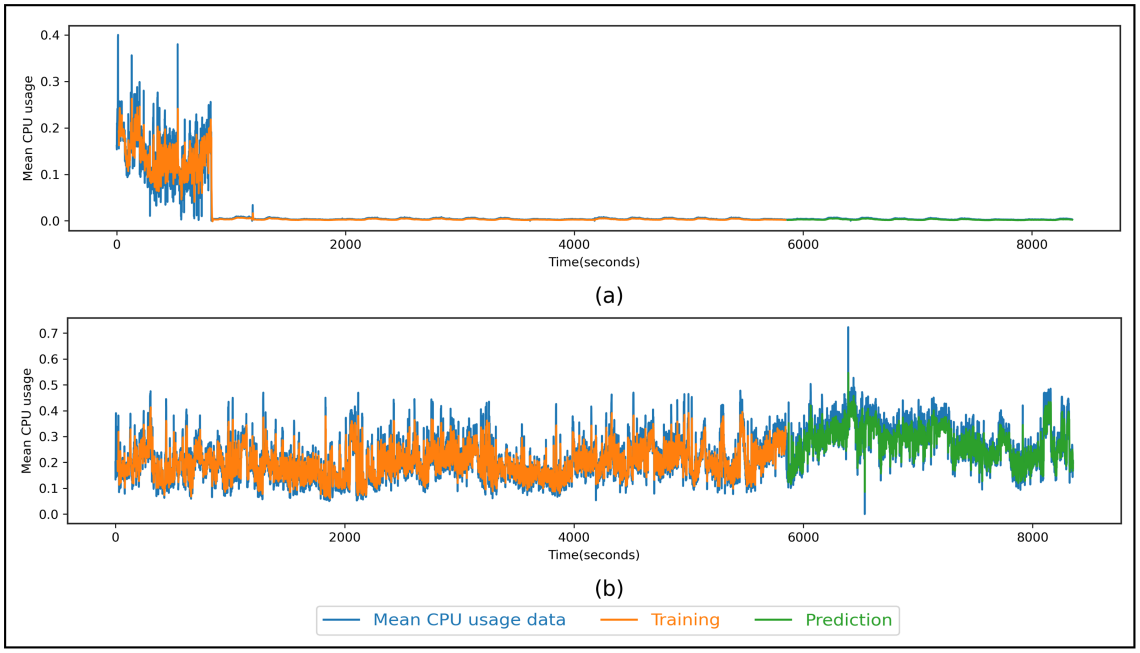


Figure 4.11: Actual and predicted mean CPU usage by Informer model at window size= 18 (a) Machine 38708463 (b) Machine 306881505

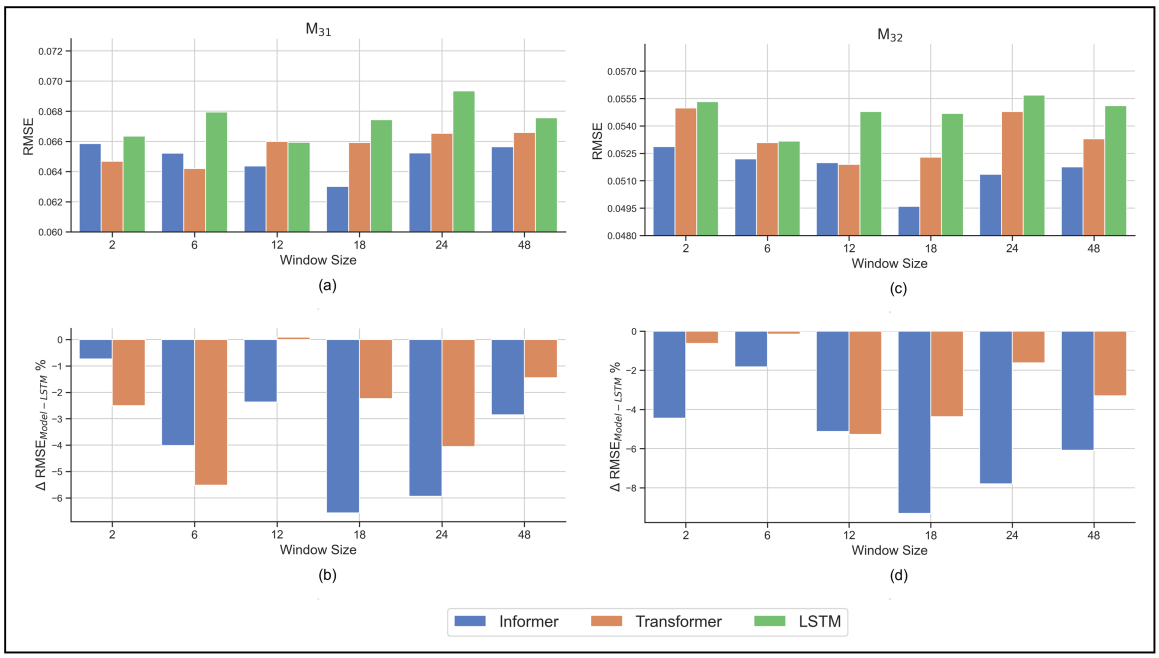


Figure 4.12: Comparison between the models using RMSE for different window sizes (a,c) RMSE for machines 1676250332 and 1338630; (b,d) Change in RMSE of Informer and Transformer with respect to LSTM for machines 1676250332 and 1338630

achieved a 5.3% deduction at window size 12, and the Informer achieved a 9.8% deduction at window size 18 in RMSE of machine 1338630 compared to the LSTM. But at window sizes 2 and 6, Transformer model RMSE is almost equivalent to LSTM.

Figure 4.13 (a)-(b) depicts that the prediction pattern of mean CPU usage is close to actual mean CPU usage for machines 1676250332 and 1338630, respectively, by employing the Informer model at window size 18. Figure 4.14 represents the RMSE of applied

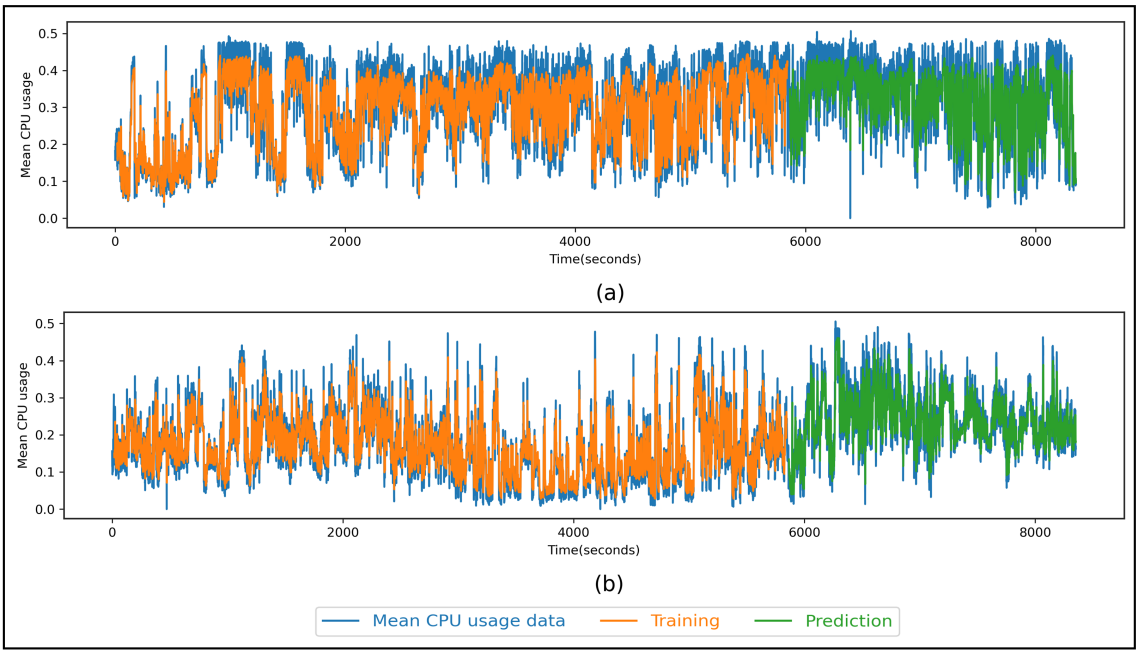


Figure 4.13: Actual and predicted mean CPU usage by Informer model at window size= 18 (a) Machine 1676250332 (b) Machine 1338630

models over input window sizes of 2, 6, 12, 18, 24, and 48 for selected machines of cluster 4. Figure 4.14 (a) represents that LSTM achieved the best mean CPU usage prediction at window size 12, and Transformer performed best at window size 6 with an RMSE of 0.06523 and 0.06378, respectively, for machine 351653579. The Informer outperformed at a window size of 18 with achieved an RMSE of 0.06286. Figure 4.14 (b) shows that the Transformer and Informer outperform at each window size of machine 351653579. Whereas the transformer reduced RMSE by 3.5% at window size 6 and the Informer reduced by 5.9% at window size 24 compared to the LSTM. At window size 12, the Transformer is achieving almost similar RMSE as LSTM. Figure 4.14 (c) depicts the results corresponding to machine 3365958041 and represents that the LSTM, Transformer and Informer achieved the best performance at window size 18 with an RMSE of 0.054399, 0.053734, and 0.052976, respectively. Figure 4.14 (d) represents that at window size 12, the Informer reduced the RMSE by 3.9% compared to LSTM for machine 3365958041. However, at window sizes 2 and 6, the difference between Transformer and LSTM RMSE is small.

Figure 4.15(a)–(b) depicts the trained and predicted CPU usage pattern by employing the Informer at window size 18 for machines 351653579 and 3365958041, respectively.

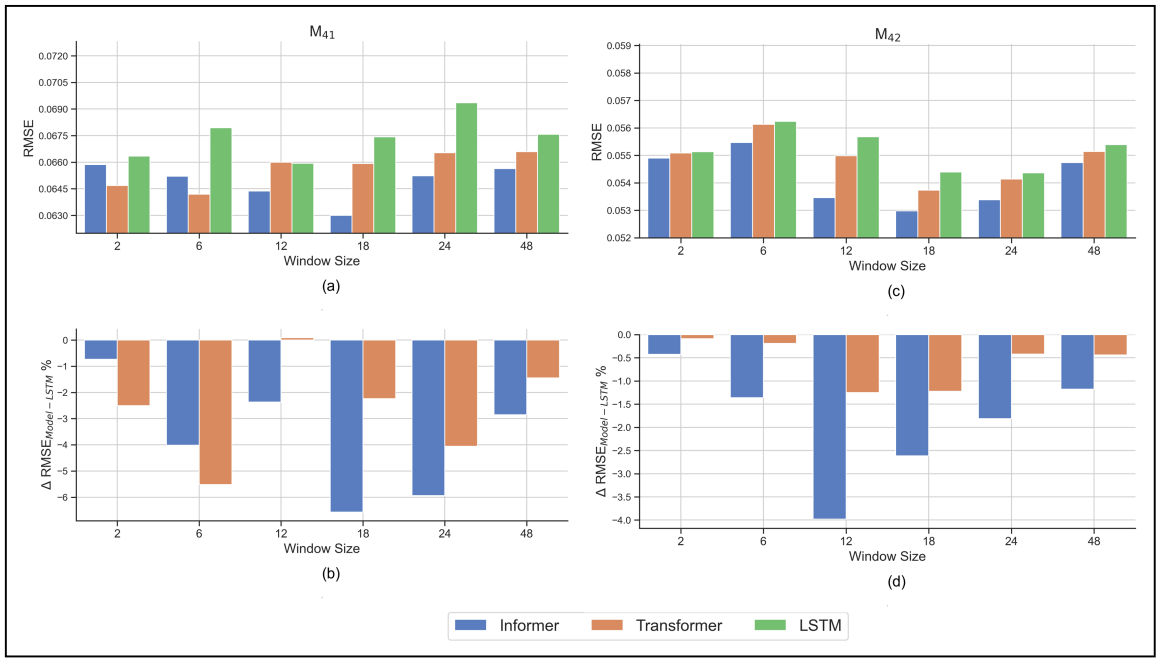


Figure 4.14: Comparison between the models using RMSE for different window sizes (a,c) RMSE for machines 351653579 and 3365958041; (b,d) Change in RMSE of Informer and Transformer with respect to LSTM for machines 351653579 and 3365958041

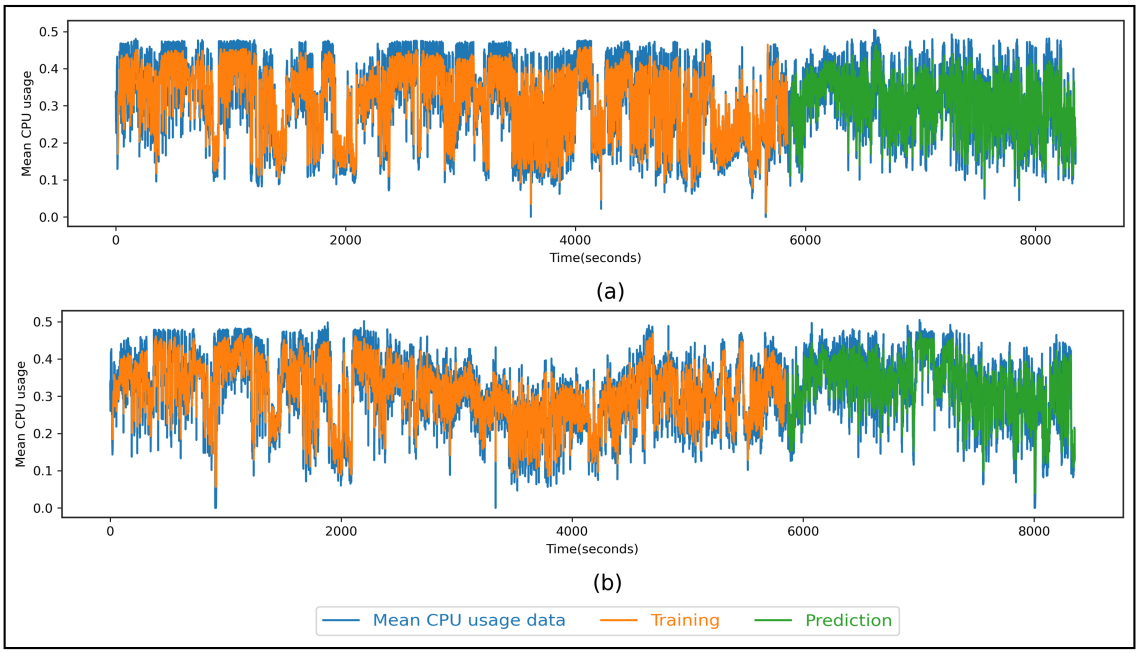


Figure 4.15: Actual and predicted mean CPU usage by Informer model at window size= 18 (a) Machine 351653579 (b) Machine 3365958041

Figure 4.16 represents the RMSE of applied models over input window sizes of 2, 6, 12, 18, 24, and 48 for selected machines of cluster 5. Figure 4.16 (a) shows LSTM, Transformer,

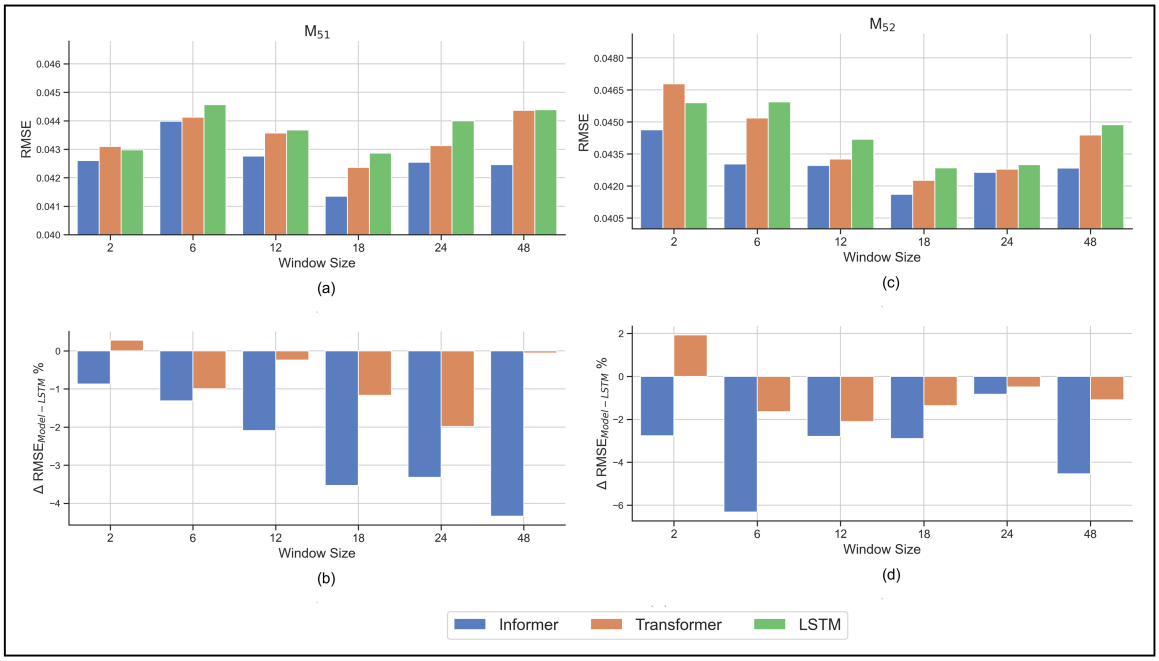


Figure 4.16: Comparison between the models using RMSE for different window sizes (a,c) RMSE for machines 1335688 and 317486393; (b,d) Change in RMSE of Informer and Transformer with respect to LSTM for machines 1335688 and 317486393

and Informer performed best at a window size 18 with an RMSE of 0.04286, 0.04236, and 0.04135, respectively, for machine 1335688; hence, Informer outperforms. Figure 4.16 (b) denotes that for machine 1335688, at window size 2, LSTM performs better than Transformer. Further, at window size 48, the RMSE of the Transformer and LSTM are almost similar, and at window size 12, the RMSE difference between the Transformer and LSTM is low. Whereas, Informer outperforms at each window size and reduces RMSE by 4.8% compared to LSTM at window size 48. Figure 4.16 (c) depicts that prediction performance enhances correspondingly to the increment of window size up to 18 for machine 317486393. The best achieved RMSE of LSTM at window size 18 and Transformer at window size 12 is 0.04285 and 0.04325, respectively. The Informer outperforms with an RMSE of 0.04161 at window size 18. Figure 4.16 (d) shows that for machine 317486393, at window size 2, LSTM beat than Transformer and at window size 24, the RMSE difference between Transformer and LSTM is slight. However, Transformer reduces RMSE by 2.1% at window size 12, and Informer by 6.2% at window size 6 compared to LSTM. Figure 4.17 (a)-(b) depicts the trained and predicted mean CPU usage value by employing the Informer at an input window size 18 for machines 1335688 and 317486393, respectively.

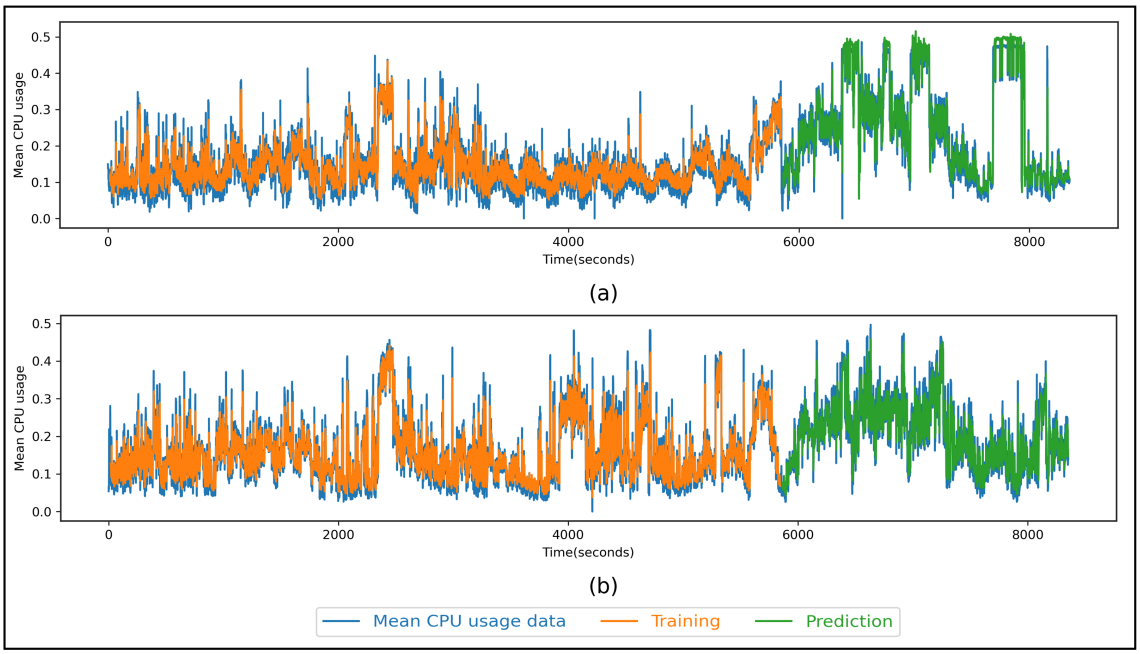


Figure 4.17: Actual and predicted mean CPU usage by Informer model at window size= 18 (a) Machine 1335688 (b) Machine 317486393

4.5 Summary

Deep learning, attention-based mechanism models, such as Transformer and Informer, and LSTM, are employed to predict the CPU usage of PMs. LSTM considers the sequence order of the input data when making predictions. Specifically, the model processes the input data sequentially, one element at a time, and updates its internal state at each step. The transformer works well on long-range dependencies and processes the input data in parallel without considering the order of the input sequence. Meanwhile, the Informer model works well for both long-range and input sequence order. The experiment results showed that the Transformer model predicts better than LSTM for most of the machines. This indicates that long-range dependencies significantly improve the prediction. For some machines, LSTM displayed better RMSE than Transformer, which concluded that the input sequence order is also relevant for resource usage predictions. This is highlighted by the better performance of the Informer model in all cases considering both aspects. However, this approach is limited to PM resource usage prediction. In the next chapter, optimally allocate VMs to PMs using a single objective function to maximize the resource utilization of PMs. We implemented the First Fit, First Fit Decrease, Best Fit, Best Fit Decrease and Genetic Algorithm to achieve this optimization. Following the allocation, compute the total energy consumption achieved by each employed algorithm and analyze the correlation between the minimum number of active

PMs and the overall energy consumption of the CDC.

Chapter 5

Effective Resource Management through VM Allocation Strategies

Efficient VMA is critical for optimizing resource utilization and minimizing energy consumption in CDCs. Reducing the number of active PMs will result in minimizing the overall energy consumption of the CDC. In this chapter, we utilized single-objective heuristic algorithms, First Fit (FF), First Fit Decrease (FFD), Best Fit (BF), Best Fit Decrease (BFD), and Genetic Algorithm (GA) to assign VMs to PMs and achieve optimal mappings. In addition, we assessed total energy consumption based on the resource utilization of PMs needed to operate the assigned VMs. The experimental findings indicate that achieving an optimal number of active PMs is not always sufficient for reducing total energy consumption. GA successfully achieves an optimal number of active PMs, but it does not consistently result in the lowest total energy consumption in the data center compared to other employed algorithms. This highlights the importance of considering energy consumption as an independent objective in VMA and consolidation strategies, beyond solely minimizing the number of active PMs, to reduce the overall energy consumption of the data center effectively.

5.1 Contributions

The presented work yields the following listed contributions:

- An experiment is conducted to optimally allocate VMs to PMs using FF, FFD, BF, BFD, and GA, with the objective of maximizing the resource utilization of PMs. The comparison of these algorithms is based on their effectiveness in minimizing the number of active PMs, thus decreasing energy usage.
- The experimental results suggest that achieving an optimal quantity of active PMs does not always lead to the lowest overall energy usage. The variations in the correlation between active PMs and energy usage are highlighted.
- The examination of the GA specifically demonstrates its efficacy in optimizing the number of active PMs. However, it also highlights that GA does not consistently achieve the minimum total energy consumption compared to other heuristic approaches.
- By demonstrating that minimizing active PMs alone is insufficient for optimal energy efficiency, This work underscores the importance of developing energy-aware VM allocation strategies. This involves considering energy consumption as a distinct and critical metric alongside resource utilization and performance optimization.

5.2 Problem Statement

To assign or map VMs to PMs is a challenging combinatorial optimization problem similar to the well-known bin packing (BP) problem [245, 246]. Like its counterpart, this task falls under the category of NP-hard problems, where finding the optimal solution becomes increasingly computationally intensive as the size of the problem grows. In the bin packing problem, items of different sizes need to be packed into a fixed number of bins or containers, minimizing the number of bins used while ensuring that the total size of items in each bin does not exceed its capacity [247].

In the context of VM allocation, the bin packing problem translates into the challenge of efficiently assigning VMs with different resource demands to a number of PMs while optimizing resource utilization and minimizing the number of PMs used. Each VM represents an item with specific resource requirements (such as CPU, memory, and storage), and each PM corresponds to a bin with limited capacities for these resources. The goal is to allocate VMs to PMs to maximize the utilization of PM resources, minimize resource

wastage, and ensure that the constraints imposed by PM capacities are not violated.

Let $P = \{p_1, p_2, ..., p_p\}$, P represent the set of p PMs with different CPU capacities, where p_i denotes the capacity of the i^{th} PM. Similarly, let $V = \{v_1, v_2, ..., v_v\}$, V denote the set of v VMs with different CPU demands, where v_j represents the CPU demand of the j^{th} VM. The total number of possible mappings can be calculated by considering that each of the V VMs can be mapped to any one of the P PMs. This yields n choices for each VM. Therefore, the total number of possible mappings is obtained by raising the number of PMs (p) to the power of the number of VMs (v): Total number of mappings $= p^v$. This problem requires heuristic and metaheuristic approaches to find near-optimal solutions. The objective function for a single objective shown in 5.1 i.e., the minimum number of PMs and its constraints:

$$obj1 = \min(\text{NumberOfActivePM}) = \sum_{i=1}^{n} PM_i \tag{5.1}$$

Subject to
NumberOfActivePM ≥ 1
$\sum_{j=1}^{v} a_j x_{ij} \leq p_i PM_i, \quad \text{where } i \in \{1, 2, 3, \ldots, p\}$
$\sum_{i=1}^{p} x_{ij} = 1, \quad \text{where } j \in \{1, 2, 3, \ldots, v\}$
$PM_i \in \{0, 1\}, \quad i = 1, 2, \ldots, n$
$x_{ij} \in \{0, 1\}, \quad i = 1, 2, \ldots, p; \quad j = 1, 2, \ldots, v$

where $PM_i = 0$ if PM i is not used, otherwise $PM_i = 1$, and $x_{ij} = 0$ if VM j is not put into PM i, otherwise $x_{ij} = 1$.

In addition, calculate the total energy consumption of data center (TEC) using 5.2

$$\text{TEC} = \sum_{i=1}^{p} E \left[\frac{\text{Total\_usage}_i}{p_i} \cdot 100 \right], \quad \text{where } i \in \{1, 2, 3, \ldots, p\} \tag{5.2}$$

Where, Total_usage_i is the summation of CPU usage by the allocated VMs on PM i. E is energy in watt taken from the table 5.1, as per the CPU usage percentage of PM and MIPS configuration of PM.

5.3 Algorithms Used

Given that 'p' PMs have been provided with resource capacities measured in terms of CPU, we need to allocate 'v' VMs to ensure the overall resource requirement of the VMs assigned to a PM does not exceed its capacity. This section discusses the BP-based VMA

algorithms employed to obtain the optimal VMA.

5.3.1 First Fit (FF)

The FF algorithm, employed in the allocation of VMs, operates by assigning each VM in a sequential manner to the first available PM that can accommodate it within its capacity constraints, as shown in algorithm 5.1. The process starts by analyzing the initial VM and then continues by systematically searching through each succeeding machine, aiming to locate a suitable PM with sufficient capacity. Once an appropriate match is found, the VM is assigned to a particular PM, and the capacity of the PM is updated accordingly. This process continues until all VMs are assigned. An error is raised when there are no available PMs to accept a VM. The FF strategy prioritizes filling PMs in the order they are encountered, with the goal of minimizing resource wastage and reducing the number of active PMs needed for allocation [248, 249].

Algorithm 5.1 First Fit (FF) Algorithm for VM Allocation

Require: p: number ofPMs, v: number of VMs, a: array of VM resource demands, S: array of PM capacities
Ensure: Assignment of VMs to PMs

```
1: Initialize array assignment of length v with -1 {-1 means VM is unassigned}
2: for j = 0 to v − 1 do
3:    for i = 0 to p − 1 do
4:       if a[j] ≤ S[i] then
5:          Assign VM j to PM i
6:          S[i] ← S[i] − a[j] {Update remaining capacity of PM i}
7:          break {Move to the next VM after assigning to the first suitable PM}
8:       end if
9:    end for
10: end for
11: return assignment
```

5.3.2 First Fit Decreasing (FFD)

The FFD method is based on the same foundation as the FF algorithm but includes an extra step to enhance efficiency. The process begins by arranging the VMs in descending order according to their resource requirements as shown in algorithm 5.2. This sorting method assists in prioritizing larger VMs, which might potentially decrease fragmentation and enhance resource utilization. Next, it carries out the allocation procedure by systematically allocating each VM to the earliest accessible PM that can accommodate it within its capacity limitations. Similar to the FF algorithm, it aims to minimize resource

wastage and the number of active PMs utilized for allocation. However, prioritizing the larger VMs when using the FFD algorithm can result in more efficient allocations, especially in situations when there is a significant difference in the sizes of VMs [250, 251].

Algorithm 5.2 First Fit Decrease (FFD) Algorithm for VM Allocation

Require: p: number ofPMs, v: number of VMs, a: array of VM resource demands, S: array of PM capacities
Ensure: Assignment of VMs to PMs

```
1: Sort VMs in array a in non-increasing order {Sort VMs by resource demands}
2: Initialize array assignment of length v with -1 {-1 means VM is unassigned}
3: for j = 0 to v − 1 do
4:    for i = 0 to p − 1 do
5:       if a[j] ≤ S[i] then
6:          Assign VM j to PM i
7:          S[i] ← S[i] − a[j] {Update remaining capacity of PM i}
8:          break {Move to the next VM after assigning to the first suitable PM}
9:       end if
10:   end for
11: end for
12: return assignment
```

5.3.3 Best Fit (BF)

The BF algorithm operates by iteratively assigning each VM to the PM with the closest capacity to accommodate it without exceeding its capacity constraints, as shown in algorithm 5.3. The process involves allocating each VM by evaluating all the available PMs and choosing the one with the least remaining capacity after accommodating the VM. This excess capacity reflects the most suitable allocation for the VM, guaranteeing optimal use of resources and minimizing inefficiency. The Best Fit method tries to optimize resource allocation and prevent fragmentation by prioritizing the PM that has the closest capacity match to each VM. However, this approach may lead to slightly higher computational complexity compared to FF, and FFD as it requires searching through all available PMs for each allocation [252, 253].

5.3.4 Best Fit Decrease (BFD)

The BFD algorithm for VM allocation follows the basic principles of the BF algorithm but introduces an additional step where VMs are sorted in descending order of their resource demands before allocation as shown in algorithm 5.4. This method prioritizes

Algorithm 5.3 Best Fit (BF) Algorithm for VM Allocation

Require: p: number ofPMs, v: number of VMs, a: array of VM resource demands, S: array of PM capacities
Ensure: Assignment of VMs to PMs

```
Initialize array assignment of length v with -1 {-1 means VM is unassigned}
for j = 0 to v − 1 do
    bestFitIndex ← −1
    bestFitGap ← ∞
    for i = 0 to p − 1 do
        if a[j] ≤ S[i] and S[i] − a[j] < bestFitGap then
            bestFitIndex ← i
            bestFitGap ← S[i] − a[j]
        end if
    end for
    if bestFitIndex ≠ −1 then
        Assign VM j to PM bestFitIndex
        S[bestFitIndex] ← S[bestFitIndex] − a[j] {Update remaining capacity of PM}
    end if
end for
return assignment
```

allocating larger VMs first, which can help reduce fragmentation and optimize resource utilization. By sorting VMs based on their demands, the BFD algorithm aims to find the best fitting allocation for each VM while minimizing resource wastage [254, 255].

5.3.5 Genetic Algorithm (GA)

The GA for VM allocation is a population-based optimization technique designed to minimize the number of active machines while efficiently assigning VMs to PMs. Initially, a population of potential solutions is generated randomly, each represented as a chromosome encoding VM-to-PM mappings. These chromosomes undergo an iterative, evolutionary process comprising several key steps shown in algorithm 5.5. Firstly, each chromosome is evaluated based on a fitness function that quantifies its quality in terms of factors like the number of active machines and resource utilization. Then, a selection process is employed to choose promising chromosomes for reproduction. During crossover, selected chromosomes exchange genetic information to produce offspring with new VM allocation configurations. Mutation introduces random variations to maintain diversity within the population. The offspring, along with potentially mutated individuals, replace less fit members of the population. This iterative process continues over multiple generations until termination conditions are met, such as a maximum number of generations or the attainment of a satisfactory solution. Throughout the algorithm's execution, it

Algorithm 5.4 Best Fit Decrease (BFD) Algorithm for VM Allocation

Require: p: number of PMs, v: number of VMs, a: array of VM resource demands, S: array of PM capacities
Ensure: Assignment of VMs to PMs

Sort VMs in array a in non-increasing order {Sort VMs by resource demands}
Initialize array *assignment* of length v with -1 {-1 means VM is unassigned}
for $j = 0$ **to** $v - 1$ **do**
  bestFitIndex $\leftarrow -1$
  bestFitGap $\leftarrow \infty$
  for $i = 0$ **to** $p - 1$ **do**
    if $a[j] \leq S[i]$ **and** $S[i] - a[j] <$ bestFitGap **then**
      bestFitIndex $\leftarrow i$
      bestFitGap $\leftarrow S[i] - a[j]$
    end if
  end for
  if bestFitIndex $\neq -1$ **then**
    Assign VM j to PM bestFitIndex
    S[bestFitIndex] $\leftarrow S$[bestFitIndex] $- a[j]$ {Update remaining capacity of PM}
  end if
end for
return *assignment*

systematically explores the solution space, aiming to discover VM allocation schemes that minimize the number of active machines while efficiently utilizing resources and adhering to system constraints [256, 257].

Algorithm 5.5 Genetic Algorithm (GA) for VM Allocation

Require: p: number of PMs, v: number of VMs, a: array of VM resource demands, S: array of PM capacities, N: population size, G: maximum number of generations
Ensure: Best assignment of VMs to PMs

Initialize population P with N random solutions {Each solution represents a VM-to-PM assignment}
for $g = 1$ **to** G **do**
  Evaluate fitness of each solution in P based on resource utilization and constraints
  Select parents for crossover and mutation (e.g., tournament selection, roulette wheel selection)
  Apply crossover and mutation operators to create offspring
  Evaluate fitness of offspring
  Select new population P using a replacement strategy (e.g., elitism, generational replacement)
end for
return Best solution (assignment) found in P

5.4 Experiment

This section outlines the system configuration used for conducting all experiments. It includes details regarding the setup configuration for VMs and PMs. Additionally, energy consumption is considered based on the PM configuration and CPU usage

Simulation setup

The same environment set-up is used as described in Chapter 3.5 for the experiment. We created a simulation environment comprising n heterogeneous PMs. One-quarter of the PMs are configured with 1000 MIPS, another quarter with 2000 MIPS, a subsequent quarter with 4000 MIPS, and the remaining machines are outfitted with 6000 MIPS while assuming all PMs have a single core. The 'm' VM instances are equally categorized into five MIPS configurations, namely 200, 700, 1000, 1500, and 2000.

Energy consumption

As depicted in Table 5.1, assume that when a PM is equipped with 1000 MIPS and operates at 50% usage after allocating all VMs, it consumes 13 (w) energy. Likewise, when a PM has a configuration of 4000 MIPS, and its CPU usage reaches 80% post VM allocation, its contribution to the total energy consumption amounts to 66 (w), and so forth. When a PM is inactive, and in a sleeping stage, regardless of its configuration, its energy consumption is considered to be 0.

Table 5.1: PM MIPS and Energy Consumption in Percentage

MIPS	**Sleep**	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
1000	0	5	7	9	11	13	15	17	19	21	23
2000	0	9	12	15	18	21	24	27	30	33	36
4000	0	24	30	36	42	48	54	60	66	72	80
6000	0	36	46	56	66	76	86	96	106	116	126

We employed FF, FFD, BF, BFD, and GA by considering the single objective, minimizing the active number of PMs after allocating all the requested VMs. In addition, calculate total energy consumption using equation 5.2 for each employed algorithm corresponding to considered VM and PM instances.

5.5 Results and Discussion

The experiments conducted to evaluate the performance of different VM allocation algorithms yielded insightful results. This section presents the findings obtained from each algorithm: FF, FFD, BF, BFD, and GA.

The evaluation involved testing each employed model across a range of scenarios with varying numbers of PMs and requested VMs. The experiments were conducted by incrementally increasing the number of PMs from 100 to 1000 and VMs from 200 to 1000. Figure 5.1 illustrates the relationship between the number of requested VMs and the corresponding number of PMs utilized by each employed algorithm. The x-axis represents the varying numbers of requested VMs, while the y-axis indicates the number of PMs required for allocation. The results indicate that when a data center comprising 100 PMs requests allocation for 200 VMs, as discussed in Section 5.4, the FF, FFD, BF, BFD, and GA algorithms require 95, 91, 97, 95, and 80 active PMs, respectively. Upon increasing the number of PMs to 200 and the requested VMs to 300, the active PMs required by FF, FFD, BF, BFD, and GA algorithms are 136, 132, 170, 165, and 120, respectively. Similarly, setting the number of PMs and required VMs in the data center to 900 and 1000, respectively, results in a minimum requirement of 478, 463, 671, 653, and 412 PMs by the FF, FFD, BF, BFD, and GA algorithms for their allocation.

The observations across all scenarios revealed that the GA consistently allocated requested VMs to the minimum number of PMs compared to other algorithms. Conversely, the BF algorithm consistently required the maximum number of PMs to assign the same number of VMs. The BFD algorithm required fewer PMs than BF, although it utilized more PMs than the FFD algorithm. These findings underscore the efficiency of GA in resource allocation, the suboptimal performance of BF in resource utilization, and the intermediate performance of BFD relative to BF and FFD.

Based on the minimum number of required active machines corresponding to the requested number of VMs and available PMs in a data center, we conducted energy consumption calculations for employed algorithms, and the obtained results are depicted in Figure 5.2. Here, the x-axis represents the number of required VMs. At the same time, the y-axis depicts the total energy consumption according to the active number of PMs used by the employed algorithms.

Upon analysis, we observed that when the number of PMs and VMs were set to 200 and 300, respectively, the FF algorithm utilized 136 active PMs, resulting in a total energy consumption of 9123 watts in the data center. Meanwhile, the FFD algorithm utilized

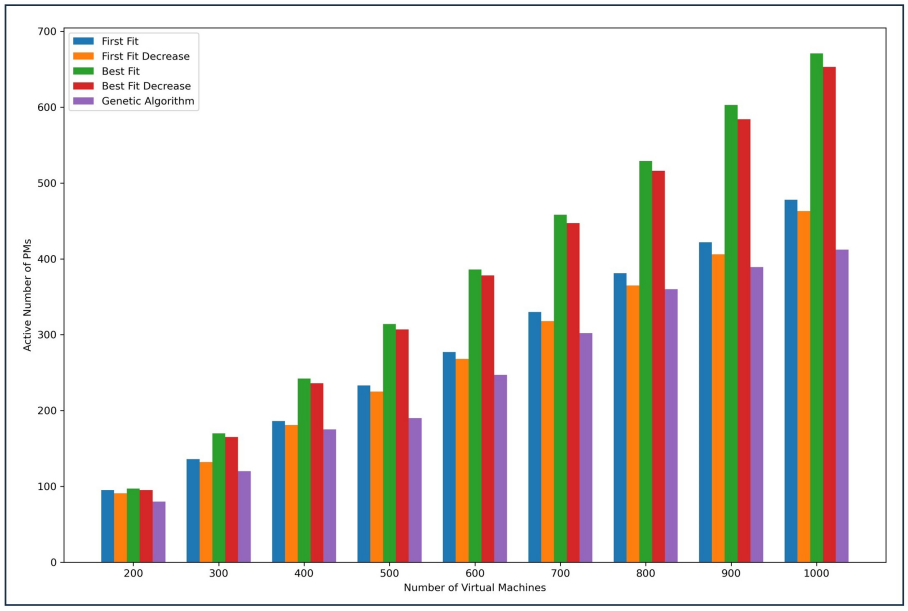


Figure 5.1: Number of active PMs corresponding to the number of VM requests during VMA process

132 PMs, consuming 8909 watts. The BF algorithm required 170 PMs and consumed 8992 watts, while the BFD algorithm used 165 PMs and 8762 watts. The GA required 120 PMs and consumed 8991 (w) of energy. Notably, GA exhibited the most efficient resource allocation, requiring the least number of machines, but the energy consumption was higher than that of FFD and BFD. However, it shows that BFD utilizes more active PMs than FF, FFD, and GA algorithms but still achieves the least energy consumption. In a similar scenario where there are 700 PMs and 800 VMs, the active PMs utilized by the FF, FFD, BF, BFD, and GA algorithms are 381, 365, 528, 516, and 360, respectively. However, their corresponding energy consumption values are 24315, 23772, 23477, 23161, and 24321, respectively. Here, we can observe that GA requires the least number of PMs compared to other employed algorithms, but it consumes the maximum amount of total energy after allocation. Meanwhile, BFD consumes the least amount of energy but requires more PMs than FF, FFD, and GA.

In a scenario where there are 900 PMs and 1000 VMs, the energy consumption by GA after utilizing 412 PMs is 30509, which is the highest among all other compared values. Conversely, BFD utilized 653 PMs, exceeding the requirements of FF, FFD, and GA, yet it consumed the least energy, i.e., 28851.

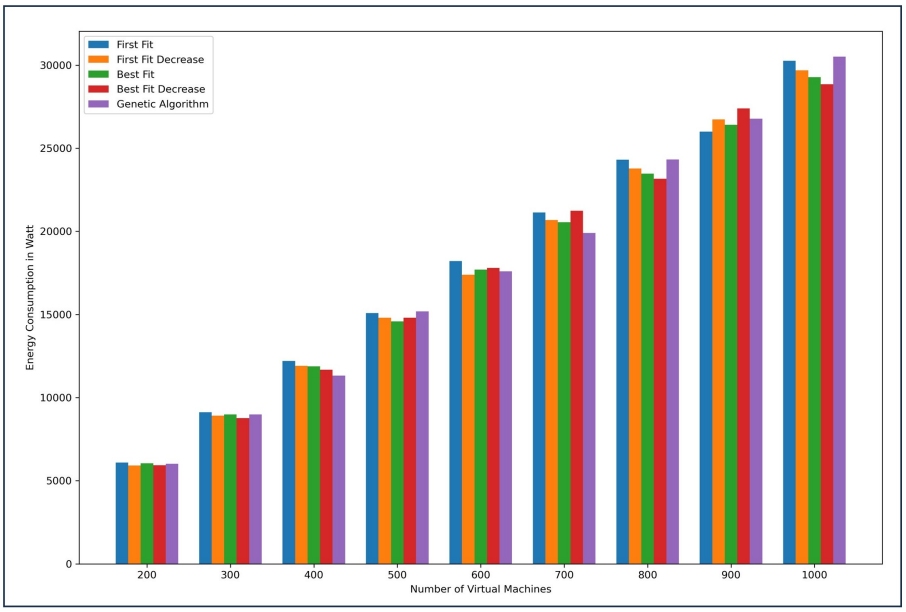


Figure 5.2: Energy consumption corresponding to the number of requested VMs during VMA process

5.6 Summary

In this chapter, requested VMs are allocated to available PMs in the data center using FF, FFD, BF, BFD and GA to maximize the resource usage of active PMs while minimizing the total number of active machines in the data center. Ten scenarios were created by changing the number of physical and virtual machines and analyzing how an active number of PMs impacts the total energy consumption of the data center. We observed that GA outperforms in achieving the minimum number of active machines but does not necessarily consume the least energy for all cases. Our investigation has shown that the energy usage of a data center cannot be entirely dependent on the quantity of active machines. Instead, it should be considered an independent objective to achieve the lowest possible energy usage within the data center.

In the next chapter, presents a Three-Tier architecture that effectively manages cloud environment resources by strategically allocating and consolidating VMs in data centers. Our strategy utilizes a multi-objective optimization algorithm, a non-dominated sorting genetic algorithm (NSGA-II), to optimally allocate initial requests to VMs to minimize the number of active servers and energy usage. In addition, integrate predicted resource usage of VMs and PMs to consolidate VMs effectively. This strategy enables the minimization of the number of active PMs, reduces the VM migration count, and decreases the total energy consumption of the data center.

Chapter 6

Three-Tier Architecture Integrating Virtual Machine Allocation and Consolidation

VMA and consolidation both play a vital role in efficiently utilizing the resources of a data center. This chapter simultaneously tackles both aspects and proposes a Three-Tier architecture to optimally allocate and consolidate VMs by using current and predicted workload. Tier 1 *optimally maps VMs to PMs using the Non-dominated Sorting Genetic Algorithm II-based VMA and consolidation (NSGAII-VMAC) mechanism. Tier* 2 *utilizes an LSTM model to predict VM workload and compute the predicted workload of PMs. Finally, Tier* 3 *consolidates VMs in three steps. Firstly, overloaded and underloaded PMs are identified using current and predicted workloads. Next, select VMs from overloaded PMs for migration using a proposed selection method inspired by the Pareto front strategy. Lastly, the destination PM is determined for the placement of migrated VMs using the NSGAII-VMAC. Extensive experiments show a significant reduction in energy consumption up to* 50.93% *while the number of active servers and migrations are reduced up to* 60.39% *and* 30.57%*, respectively, as compared to existing approaches. In addition, improvements were achieved in RMSE and MAE compared to state-of-the-art approaches using the GCT dataset.*

6.1 Introduction

To achieve the maximum benefit of each server's resources, virtualization technology leverages a VMM or hypervisor to consolidate multiple VMs into a single PM. This optimization strategy aims to enhance resource utilization, minimize the data center's hardware footprint, and reduce the overall energy consumption of the data center [258].

Along with resource consolidation, virtualization facilitates effective VMA and virtual machine consolidation (VMC), further enhancing the overall energy efficiency and performance of the data center. VMA is a mapping approach used to initially place newly provisioned VMs onto PMs in the CDC. The allocation is based on the resource demands of VMs and the available capacity of PMs, thereby creating the conditions for effective resource utilization and workload distribution [66]. The dynamic requirements of users lead to variations in PMs' resource utilization over time, resulting in under-loaded and overloaded machines. VMC is solving this by optimizing system performance and reducing the number of active PMs. The VMC process involves three key steps to dynamically reorganize and redistribute running VMs among available PMs within a virtualized environment. Firstly, it detects which PMs are in over-loaded (hotspot) and under-loaded (cold spot) conditions. Next is the selection of VMs that should be migrated from the overloaded PMs (referred to as VM selection). Lastly, find the most suitable PMs where to place the selected VMs from overloaded PMs (referred to as VM placement) and all VMs from under-loaded PMs to switch off [259, 189].

Most studies define VMA and VMC as fundamentally rooted in the BP problem, where VMs are viewed as items and PMs are bins; therefore, it is an NP-Hard problem and combinatorial optimization challenge [260, 261]. In recent works, many heuristic algorithms such as BF, FF, Next Fit, Round Robin, Greedy, FFD, BFD [262, 263, 264, 265], metaheuristic algorithms such as PSO [266, 267, 268], GA [269, 270, 271], ACO [146, 272, 273, 274], Simulated Annealing [275], Artificial Bee Colony [276], Interlinear Programming [277], Game Theory [278, 279], ML [280, 281, 189] and various hybrid approaches have been employed to improve resource utilization and QoS, detect overloaded and underloaded PMs, reduce energy consumption while minimizing number of active servers. In a cloud computing environment, most research focuses on either VM allocation or VM consolidation. However, a significant gap remains in the comprehensive analysis of how these two processes collectively operate and optimize the overall energy consumption of a data center. Furthermore, we observed that most researchers concentrated on the current resource utilization of VM and PM for VM consolidation. The presented work yields the following listed contributions:

- A Three-Tier architecture is proposed to enhance the optimal allocation of user requests within a cloud computing environment. Also, the proposed architecture dynamically consolidates VMs by leveraging current and predictive CPU usage data from PMs. In addition, it facilitates efficient resource allocation and reduces energy consumption, contributing to the overall optimization of the cloud infrastructure.

- In this study, our contribution expands to the application of a fast non-dominating sorting algorithm uniquely customized for the optimal allocation of VMs to PMs within a cloud computing environment, referred to as NSGAII-VMAC. The algorithm considers a bi-objective optimization approach, aiming to simultaneously minimize energy consumption and the optimal number of active machines.

- This study incorporates current and predictive resource usage for enhanced decision-making in the status detection of PMs (i.e., overloaded, normal, underloaded) and VM consolidation. The LSTM model is employed to approximate future resource requirements and compare its accuracy with state-of-the-art models.

- Pareto front of NSGA-II driven approach is proposed to select VMs from overloaded PMs for migration (VM selection). This method is a multi-objective optimization technique which minimizes migration count and effectively manages the resources of overloaded PMs to enhance the overall performance of cloud infrastructures.

- To place the migrated VM to the targeted PM (VM placement), our proposed methodology considers the current and predicted resource usage of VMs. This approach guarantees that the chosen targeted PM not only fulfils the immediate resource needs of the migrated VMs but also anticipates future demands. By integrating predictive resource usage, our methodology acts as a safeguard against potential spikes in demand for the assigned VMs post-migration, thereby minimizing the necessity for frequent migrations. To implement this methodology, we utilized the NSGAII-VMAC algorithm, aiming to identify the optimal target PM. The primary objectives of this algorithm include reducing energy consumption, minimizing the count of migrations, and optimizing the number of active servers.

- This study takes into account both optimal VMA and dynamic VMC on total energy consumption, migration count, and the number of active PMs. Our investigation sheds light on the interplay between these crucial aspects, providing a comprehensive understanding of their collective influence on the efficiency and performance of cloud computing environments.

6.2 Energy Efficiency Aware Architecture of CDC

This section describes the general process followed in CDCs to execute any requested task and proposed energy efficiency-aware three-tier architecture.

6.2.1 System architecture

A CDC is an infrastructure that contains 'p' heterogeneous PMs and 'v' VMs, denoted as $P = \{p_1, p_2, p_3, --, p_p\}$ and $V = \{v_1, v_2, v_3, --, v_v\}$, respectively. Each PM offers 'r' distinct resources, such as CPU, memory, storage and network I/O, denoted as $D = \{d_1, d_2, d_3, --, d_r\}$ with its capabilities. Cloud users submit requests for VMs, describing the desired VM configuration regarding resources (CPU, memory, storage, and I/O). After receiving the user's request, the cloud management examines the available resources of PMs and allocates requested VMs to suitable PMs. However, due to fluctuating resource usage, PMs may become unbalanced, leading to resource inefficiency and reduced performance. Therefore, VM migration is employed to resolve this issue, which involves transferring a running VM from one PM to another without affecting its operation. Hence, the accurate overloaded and underloaded PM detection plays a vital role in identifying which VM should be migrated. When a PM is overloaded, its resources become constrained, which could result in performance degradation. On the other hand, when the PM is underloaded, its resources are not fully utilized, leading to wastage of available resources, a higher number of active PMs and energy consumption. So, the cloud management periodically monitors PM and VM resource usage to detect overloading and underloading PMs and select which VM should be migrated and where to place it at a specific time interval. The collective decision of periodic detection and migration (i.e., selection and placement) process is called dynamic VM consolidation, as shown in the Figure 1.6.

6.2.2 Proposed 3-tier architecture

A Three-Tier architecture consists of VMA, workload prediction, and VMC. This architecture aims to enhance resource utilization, reduce energy consumption, minimize migration count and number of active PMs. Figure 6.1 depicts all the components and relations between tiers. The role of each tier and its component is described as follows:

1. Tier 1 receives the configuration of VMs based on users' requests and the available capacity of each PM in the data center, which is periodically updated and stored in the Resource monitor component. The NSGAII-VMAC algorithm is employed

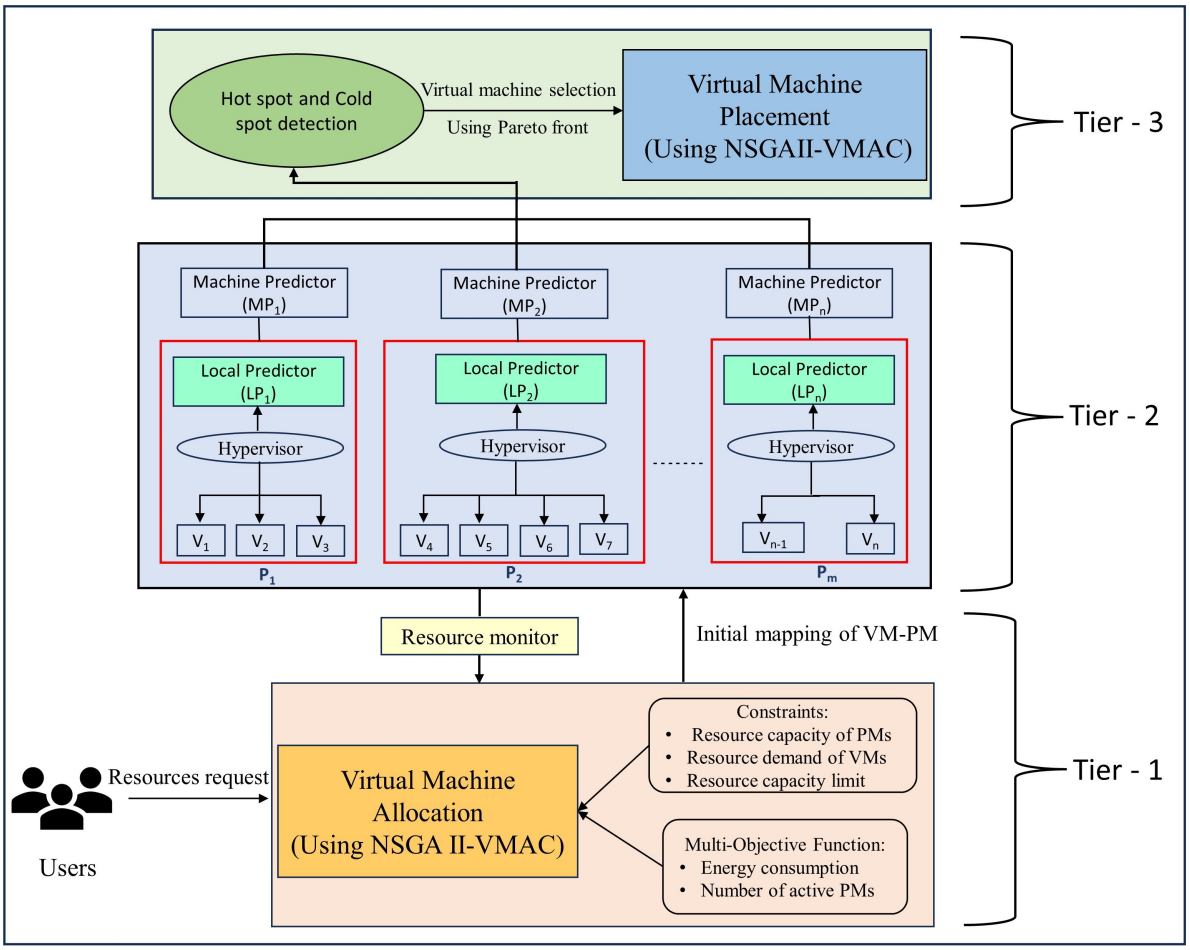


Figure 6.1: Proposed Three-Tier Architecture

for optimal VMA, described in section 6.3.2.

2. Tier 2 predicts the workload of VMs and PMs. The Local Predictor (LP) of each PM monitors the current usage of their assigned VMs and leverages the LSTM networks to predict the workload of VMs accurately. The Machine Predictor (MP) calculates the current and predicted resource utilization of PMs, illustrated in section 6.3.3.

3. Tier 3 identifies overloaded and underloaded PMs by using current and predicted usage of PMs as shown in section 6.3.4. Then, select a subset of VM from the overloaded machines for migration using the proposed VM selection algorithm demonstrated in section 6.3.5.

4. Tier 3 also finds the destination PMs to migrate selected VMs of overloaded PMs and VMs of underloaded PMs using the NSGAII-VMAC algorithm while adhering to the constraint that the destination machine should not be overloaded currently or shortly after placement of migrated machines as shown in 6.3.6.

6.3 Research Methodology

The proposed methodology for optimizing data center energy consumption is structured into three tiers (VMA-Prediction-VMC), as visually depicted in Figure 6.1, each strategically designed to maximize resource usage and minimize the overall energy usage in the data center. This section comprehensively explains the functionality of each tier.

6.3.1 NSGA II - based VM allocation and consolidation (VMAC):

NSGA-II is a multi-objective evolutionary optimization approach designed to tackle complex optimization problems involving multiple conflicting objectives. This approach imitates natural evolution, where solutions evolve and improve over generations. In any multi-objective problem, there are two goals: convergence and diversity. We integrate advanced features of NSGA-II in each generation, as shown in algorithm 6.1.

Initially, define the population size denoted as 'Q' and generate a random population while adhering to specified constraints as per the problem. Subsequently, each generated population's fitness or objective functions are evaluated. Once fitness evaluation is calculated, the Fast Non-Dominated Sorting approach is used to efficiently categorize solutions into fronts based on their dominance relationships and assign ranks to each solution according to fronts, which helps in improving the convergence of the algorithm. The first front consists of solutions that are not dominated by any other solutions; the second front is made of solutions that are solely dominated by the first front, and so on. It reduces the computational complexity of dominance depth from $\mathcal{O}(FQ^2)$ to $\mathcal{O}(FQ \log F)$, and the storage requirement is $\mathcal{O}(Q^2)$, where F is the number of objectives being optimized. Afterwards, to preserve the diversity in the solutions, calculate the Crowding Distance, which measures the crowdedness of a solution with respect to its neighbors lying on the same front.

After evaluating and assigning the rank to each solution, proceed for selection to choose a good (above-average) solution using a Crowded Binary Tournament Selection Operator. It is a modified version of the Tournament selection operator to incorporate rank and diversity (crowding distance) into selection. After selection, the selected solutions undergo variation, i.e., crossover and mutation operator, to create a new offspring population. These new solutions explore the search space. Further, the offspring population (size=Q) merge with the parent population (size=Q), which means it becomes a '2Q' solution. To choose the best 'Q' solution for the next generation, evaluate fitness, fast non-dominated sorting and crowding distance on the combined population. At the end of the current generation, select the best set of 'Q' solutions according to the front, and

the rest will be rejected. The repeated generation of NSGA-II helps converge towards a set of solutions known as the Pareto front, representing the optimal trade-offs between conflicting objectives [282].

In this chapter, we employed algorithm 6.1 for the initial mapping of VMs to PMs, i.e., VMA, and finding the destination PMs to place to migrated VMs.

Algorithm 6.1 NSGA-II using fast non dominated method

Input: i= 0 (Generation counter), Maximum allowed generation= I, population size = Q; %real number **Output:** Set of solutions that represents the best Q populations generated after I iteration

1: Initialize random population P of size Q; Apply constraints Parent population
2: Evaluate (P(i)) Evaluate objective, and assign fitness value
3: Assign rank using fast non-dominated sorting method and calculate diversity using crowding distance to P(i)
4: **while** $i < I$ **do**
5: S(i) := Selection(P(i)) (Crowded binary tournament selection) Survival of the fittest
6: V(i) := Variation (S(i)); one point crossover and mutation
7: Evaluate V(i);
8: Merge population $\hat{P}(i)$ = (P(i) U V(i)); Offspring population
9: Assign rank using fast non-dominated sorting method and calculate diversity using crowding distance to $\hat{P}(i)$;
10: P(i+1) := Survivor($\hat{P}(i)$); Survival of the fittest
11: i=i+1; increment in generation
12: **end while**

The total CPU capacity vector of PMs denoted as $C = \{c_1, c_2, - - -, c_p\}$, and the requested CPU demand vector of VMs represented as $D = \{d_1, d_2, - - -, d_v\}$ in a data center where, p and v are the total numbers of PMs and VMs, respectively. Although we are just considering one resource dimension, i.e., CPU, adding multiple dimensions for total and demand resources is possible. In addition, the remaining CPU capacity vector of PMs is represented as $R = \{r_1, r_2, - - -, r_p\}$, whereas initially R = C.

6.3.2 Virtual machine allocation (VMA):

Tier 1 employs the NSGAII-VMAC for carefully chosen PMs by considering the constraints that each VM needs to satisfy while respecting the PM's resource limitations efficiently. The aim during allocation is to optimize energy consumption and resource utilization by reducing the number of active PMs. To achieve the objectives of optimal VMA algorithm 6.1 requires establishing parameters, such as population definition, constraints, and fitness function.

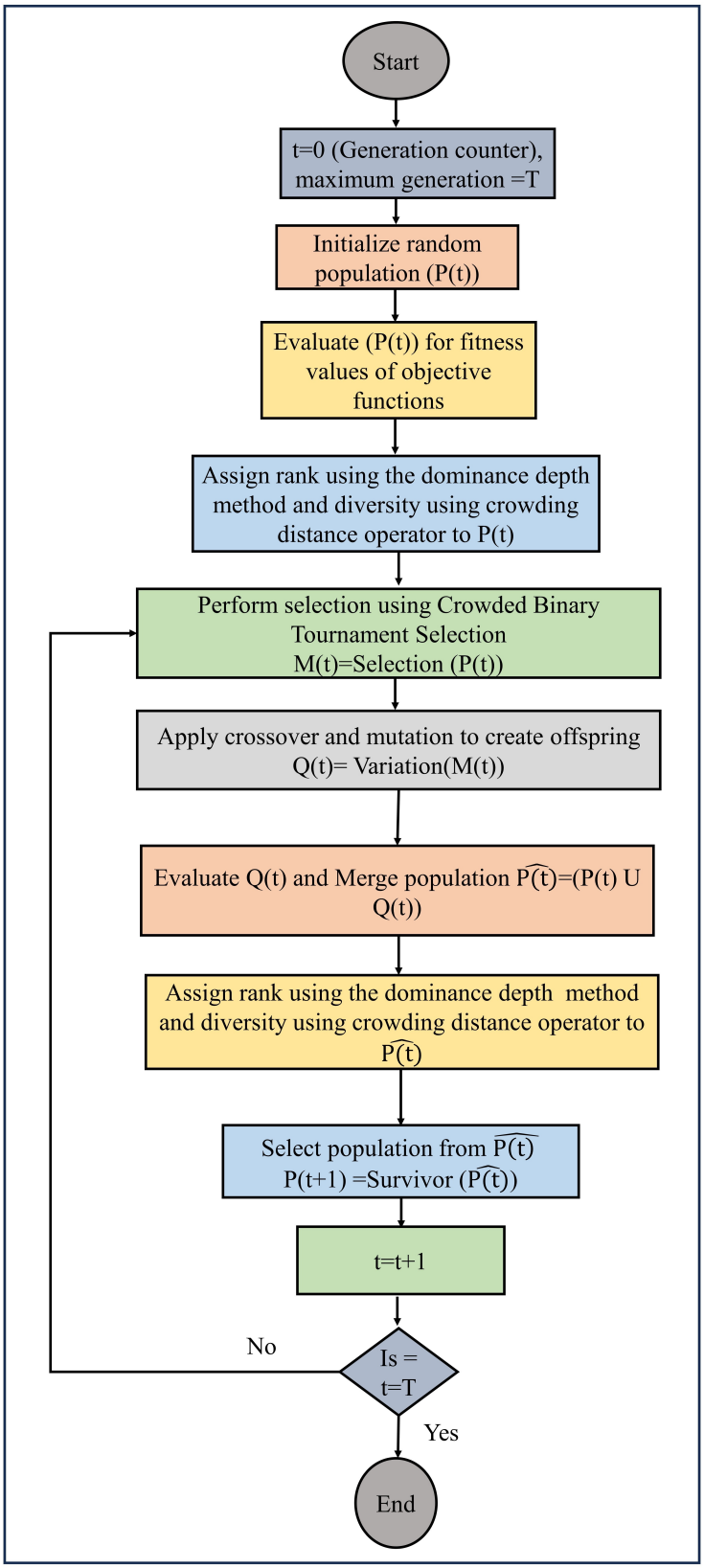


Figure 6.2: NSGAII algorithm

For initial D to C mapping, we generate the random population of size 'Q' (line 1 of algorithm 6.1). Each chromosome in the population has a unique mapping with a vector length of v. Each chromosome gene has a random number between 1 and p. If the chromosome meets the constraints that the total CPU usage of all the assigned VMs on PM should be less than or equal to that PM's total CPU capacity, and after mapping all the demanded VMs to PMs, the remaining CPU usage of PMs should be greater than or equal to 0, it will be regarded a population chromosome. Otherwise, it will be rejected, and the process will be repeated. Finally, the total chromosome count should equal Q.

$$\text{Total\_usage}_j = \sum_{i=1}^{v} d_i \cdot a, \tag{6.1}$$

where a =

$$\begin{cases} 1, & \text{if } i^{th} \text{ VM map to } j^{th} \text{ PM} \\ 0, & \text{otherwise} \end{cases}$$

Total_usage_j is the summation of CPU usage by the allocated VMs on PM j, d_i is the requested demand of VM i, and a is the binary constant.

$$r_j = c_j - \text{Total\_usage}_j \tag{6.2}$$

r_j is the remaining CPU capacity of PM j after assigning VMs. If there is no allocation of VM on PM j, then it will be equal to c_j.

Afterwards, calculate the fitness values for each chromosome. Therefore, define two fitness values (line 2 of algo 6.1), i.e., minimize total energy consumption (TEC) using equation 6.4 and reduce the number of active machines (TAM) using 6.5 as shown in equations 6.3, where F_1 is fitness function:

$$F_1 = (\text{TEC, TAM}) \tag{6.3}$$

$$\text{TEC} = \sum_{j=1}^{p} E\left[\frac{\text{Total\_usage}_j}{c_j} \cdot 100\right] \quad \text{where } j \in \{1, 2, 3, \ldots, p\} \tag{6.4}$$

$$\text{TAM} = |\{a_1, a_2, a_3, \ldots, a_v\}| \tag{6.5}$$

TEC is calculated by summing up the total load percentage of each PM after VM allocation and, according to the load percentage, considering the energy consumption (E) value

provided in Table 6.1. TAM is the number of unique values (i.e., PMs) in a population chromosome. After evaluating the fitness step, algorithm 6.1 is followed. The constraints 1 and 2 are applied during the variation step (line 6 of algo. 6.1). Finally, we get a D to C mapping sequence that ensures minimal energy consumption and effectively utilizes the CPU of each PM while reducing the number of active PMs.

Once the VMs have been allocated in the data center environment, Tier 2 will be initiated and predict the CPU usage of machines. Accurate prediction serves two purposes within our methodology: first, it aids in the early detection of overloaded PMs and allows proactive resource allocation adjustments to prevent violations. Second, it helps in identifying continuously underutilized PMs, enabling the timely shutdown of these machines to optimize resource usage and reduce energy consumption. Therefore, each PM is equipped with an LP, which includes a prediction model to predict the CPU usage of their assigned VMs.

6.3.3 Resource usage prediction:

Tier 2 employs a specialized RNN, i.e., the LSTM model, which captures and learns complex temporal patterns in time series data. LSTM helps to solve the vanishing gradient problem by maintaining long-term dependencies in sequences. This is accomplished through a network of specialized gates, such as input, forget, and output gates, which control the information flow as shown in 3.4.3.

The historical data of individual VMs are used to train the model and periodically estimate their short-term CPU usage due to the dynamic workload. The predictive CPU usage for VM "k" is represented in equation 6.6.

$$Y_k(t+1) = f(Y_k(t), Y_k(t-1), Y_k(t-2), \ldots, Y_k(t-n)) \tag{6.6}$$

$Y_k(t+1)$ represents the predicted CPU usage of VM "k" for the next time step, which is evaluated based on the information contained in the historical CPU usage data from time stamp t to t-n. Here, n is the input sequence length in the employed LSTM model. Working alongside the local predictor, the machine predictor of each PM calculates the current and predicted CPU usage of PM by aggregating the usage of its allocated VMs, as shown in Equations 6.7 and 6.8.

$$\mathrm{PM}_i^{\mathrm{CPU}}(t) = \sum \mathrm{VM}_j(t) \quad \text{when the } j\text{th VM is assigned to the } i\text{th PM} \tag{6.7}$$

$$\mathrm{PM}_i^{\mathrm{CPU}}(t+1) = \sum \mathrm{VM}_j(t+1) \quad \text{when the } j\text{th VM is assigned to the } i\text{th PM} \tag{6.8}$$

$\mathrm{PM}_i^{\mathrm{CPU}}(t)$ is the total workload at PM i during time interval t, and $\mathrm{VM}_j(t)$ is the workload of VM j at time interval t.

Dynamic fluctuations in current and predicted VM workloads lead to variations in CPU usage within PMs and create a challenging scenario where some PMs are pushed to their limits, experiencing overload while others remain underutilized. This uneven distribution of resources can lead to unused capacity, decreased performance, and increased operational costs. Therefore, Tier 3 consolidates VMs, as shown in the following sections.

6.3.4 Detect overloaded and underloaded PM:

The first step toward VMC is finding hotspots and coldspots. We used threshold criteria and considered a PM as a hotspot; either the current ($\mathrm{PM}_i^{\mathrm{CPU}}(t)$) or predicted CPU usage ($\mathrm{PM}_i^{\mathrm{CPU}}(t+1)$) of a PM exceeds 80% of its total CPU capacity (c_i). And if the current and predicted CPU usage of PM is below 40% of the c_i or least-loaded PM in the data center, it is classified as an underloaded machine. The remaining PMs are considered to be running at a normal load, and this classification is calculated as equation 6.9 and 6.10.

$$\mathrm{CPU\_per}_i(t) = \left(\frac{\mathrm{PM}_i^{\mathrm{CPU}}(t)}{c_i}\right) \times 100 \tag{6.9}$$

$$\mathrm{CPU\_per}_i(t+1) = \left(\frac{\mathrm{PM}_i^{\mathrm{CPU}}(t+1)}{c_i}\right) \times 100 \tag{6.10}$$

Where $\mathrm{CPU\_per}_i(t)$ and $\mathrm{CPU\_per}_i(t+1)$ is the total percentage of CPU usage by PM i at time interval t and t+1 respectively.

An overloaded PM has trouble with effective resource allocation, which affects system performance and causes SLAV. Following the detection of overloaded PMs, the subsequent step of Tier 3 is the selection of VMs for migration so that the PMs can return to their normal state currently and in the near future. This step aims to reduce the number of migrations and maximize the resource usage of PMs.

6.3.5 Virtual machine selection (VMS):

We proposed a VM selection algorithm inspired by the Pareto front of the NSGA-II algorithm [282]. The objective of this step is to move as few VMs as possible to minimize the disruptions caused by migrations while ensuring the maximum utilization of PM CPU resources (closest and below 80%) and mitigating the state of overloading, both in the present and near future.

We are taking into account the highest CPU usage value between the current ($\mathrm{VM}_j(t)$)

and the predicted future CPU usage ($\mathrm{VM}_j(t+1)$) of each assigned VM on the detected PM, as the CPU usage of VM (equation 6.11). This strategy helps to avoid the need for migration in the foreseeable future and efficiently manage resources, especially in cases where resource demands may increase.

$$\mathrm{Usage}_j = \max(\mathrm{VM}_j(t), \mathrm{VM}_j(t+1)) \tag{6.11}$$

Where Usage_j represents the maximum CPU utilization between the current and forecasted future CPU workloads of VM j.

Firstly, random chromosomes of the population are created. Each chromosome in the population has a binary vector denoted as $S = \{s_1, s_2, - - -, s_a\}$, where a is the total number of assigned VMs on detected overloaded PM. '0' represents a VM staying on a machine, and '1' represents a VM migrating to another machine. To create a diverse population, we generate a population of size $N = 2^a - 2$ encompassing all possible combinations, except those where either all VMs migrate (represented as all '1's) or none migrate (indicated by all '0's). Afterwards, evaluate the fitness value to achieve the objectives using equation 6.12, proceed with the chromosomes that satisfy the constraint that the total CPU utilization of the VMs staying on the machine does not exceed 80% (considered as the threshold for overloading) of the total PM capacity.

$$\mathrm{F2} = \max(\text{staying\_usage, numberOfzero}) \tag{6.12}$$

$$\text{staying\_usage} = \sum_{i=1}^{a} (\mathrm{Usage}_i \cdot s_i) \quad \text{if } s_i = 0 \tag{6.13}$$

Where staying_usage is the total CPU usage of overloaded PM after migrating some of their allocated VMs, and s_i is the i^{th} chromosome gene.

$$\text{numberOfzero} = \sum_{i=1}^{a} f(s_i) \tag{6.14}$$

Where numberOfzero is the total number of VMs that stay in PM after migration, $f(s_i)$ is a function that equals 1 if s_i equals 0 and 0 otherwise.

Lastly, non-dominated sorting is employed and categorizes a set of solutions into different fronts based on their dominance relationships of fitness values, as shown in algorithm 6.2. After identifying all Pareto optimal fronts, sort solutions within each front based on the first and second fitness values in descending order. This sorting ensures that solutions with better fitness values come first within each front. Hence, the optimal chromosome for VM selection, which successfully satisfies both objectives, is located at the forefront

as the first value within the initial front.

Algorithm 6.2 VM Selection using Non-Dominated Sorting

Input: A list of fitness values for a set of solutions. Each fitness value is represented as a pair $(staying\_usage, numberOfzero)$ **Output:** A list of fronts, where each front contains indices of solutions in that front.

```
num_individuals ← len(fitness_values)
domination_count ← [0] * num_individuals
dominated_solutions ← [[] for _ in range(num_individuals)]
fronts ← []
for i ← 0 to num_individuals − 1 do
    for j ← i + 1 to num_individuals − 1 do
        if fitness_values[i] dominates fitness_values[j] then
            if fitness_values[i] is strictly better than fitness_values[j] then
                Increase domination_count[i] by 1
                Add i to the list of solutions dominated by j (dominated_solutions[j])
            end if
        else if fitness_values[j] dominates fitness_values[i] then
            if fitness_values[j] is strictly better than fitness_values[i] then
                Increase domination_count[j] by 1
                Add j to the list of solutions dominated by i (dominated_solutions[i])
            end if
        end if
    end for
end for
current_front ← List of solutions where domination_count[i] == 0
while current_front is not empty do
    next_front ← []
    for each solution i in current_front do
        for each solution j in dominated_solutions[i] do
            Decrease domination_count[j] by 1
            if domination_count[j] == 0 then
                Add j to next_front
            end if
        end for
    end for
    Append current_front to fronts
    Set current_front to next_front
end while
for each front in fronts do
    Sort solutions in the front based on fitness values in descending order
end for
return The sorted fronts as the final result
```

6.3.6 Virtual machine placement (VMP):

The final step towards VMC is to find destinated PMs to place selected VMs from overloaded machines and migrate all the VMs of underloaded PM. We denoted the total number of VMs needed to migrate as 'm'. Tier 3 employed the proposed NSGA II-VMAC algorithm to achieve minimum energy consumption and enhance resource utilization by reducing the active PMs in a data center.

During the VMP, we calculate the usage of PMs by taking into account the maximum CPU usage between the current and predicted values of assigned VMs (equation 6.15). This ensures that we maintain sufficient capacity to accommodate any potential increase in demand for the assigned VMs even after placement of the migrated VM, allowing us to avoid migration shortly and prevent SLAVs.

$$\text{PM\_usage}_j = \sum_{i=1}^{v} \text{Usage}_i \text{ if VM } i \text{ is assigned to PM } j \tag{6.15}$$

Where PM_usage_j is the maximum possible CPU usage of PM j between t and t+1 time interval. For VMP, each chromosome gene of length 'm' is a random number within the range of 1 to 'p', excluding the PM number from where it is migrated, i.e., source PM (line 1 of algorithm 6.1). If the chromosome satisfies the specified constraints described in equations 6.18, it will be considered part of the population; otherwise, it will be discarded. The process is iterated to generate chromosomes and achieve the population size. A PM can qualify as the target PM if it meets the constraint that the PM has available capacity (equation 6.17) greater than and equal to the threshold (T) of the total CPU capacity of the PM, even after taking into account the possibility of higher CPU needs for that PM and after placement of migrated VM (equation 6.16).

$$\text{Total\_usage}_j^{\text{aftermigration}} = \sum_{i=1}^{m} \text{Usage}_i \cdot a \tag{6.16}$$

where,

$$\text{a} = \begin{cases} 1 & \text{if VM } i \text{ migrated to PM } j, \\ 0 & \text{otherwise} \end{cases}$$

$$\text{Available}_j = \frac{T \cdot c_j}{100} - \text{PM\_usage}_j \tag{6.17}$$

$$\text{Total\_usage}_j^{\text{aftermigration}} \leq \text{Available}_j \text{ if VM } i \text{ migrates to PM } j \tag{6.18}$$

Afterwards, evaluate the fitness value of an objective function (line 2 of algorithm 6.1) to select targeted PMs that minimize the total energy consumption of a data center after the migration and minimum count of PMs (equation 6.19).

$$\text{F3} = \min(\text{TEC1, TAM1}) \tag{6.19}$$

$$\text{TEC1} = \sum_{j=1}^{p} E\left[\left(\frac{\text{Total\_usage}_j(t)}{c_j}\right) \cdot 100\right], \quad \text{where } j \text{ belongs to } \{1, 2, 3, \ldots, p\} \tag{6.20}$$

$$\text{TAM1} = |\{b_1, b_2, b_3, \ldots, b_k\}|, \quad \text{where } b_1, b_2, \ldots, b_k \text{ is the subset of PMs} \tag{6.21}$$

TEC1 is the total energy consumption of a data center after migrating the selected VMs to the target PMs by aggregating energy consumed by every PM at time intervals t according to population chromosome. E is the % of energy from Table 6.1. TAM1 is the total number of active machines after consolidation. Once the fitness value of each chromosome has been computed, Algorithm 6.1 is executed. In addition, constraint 19 is applied in the variation step. At last, we will get the optimal targeted PM to place the migrated VM with minimum energy consumption within the data center and reduce the number of active machines.

6.4 Experimental Setup

In this section, we provide an outline of the simulation setup environment used to evaluate our proposed approach and describe the Google cluster usage trace dataset along with evaluation metrics.

6.4.1 Simulation setup

To analyze the effectiveness of our proposed three-tier architecture, we used the Python programming language, specifically version 3.9.12, over the Pycharm Python Integrated Development Environment. The implementation is carried out on a computer system equipped with a 12th Generation Intel(R) Core(TM) i7-12700H processor running at 2.30 GHz, an Nvidia GeForce RTX 3060 GPU with 16 GB of physical memory, and Microsoft Windows 10 Professional as the operating system, running on a 64-bit architecture. We created a simulation environment comprising P heterogenous PMs. Half of the PMs are

HP ProLiant ML110 G4 servers equipped with 1,860 MIPS cores, while the remaining half are HP ProLiant ML110 G5 servers comprising 2,660 MIPS cores and assume that all PMs have a single core. The 'V' VM instances, with three-fourths configured at 600 MIPS and one-fourth operating at 800 MIPS. We have conducted all comparison techniques within the same simulation environment.

6.4.2 Google cluster trace usage data

For our experiment, we calculated the cumulative CPU usage of the PMs by collecting and combining the workloads generated by tasks running on each specific PM over the initial ten days using the Sum-Average algorithm described in 3.4.1. This aggregation process was conducted at 5-minute intervals, allowing for a periodic assessment of how PMs actually utilized their CPU. The attributes we used to gather information on PM are: start time, end time, machine ID, task index, job ID and average CPU usage. Start and end times represent the time when the task began and completed its execution, respectively, measured in microseconds. Task index and Machine ID are unique identifiers for the task within the job and the machine on which the task was executed, respectively. Average CPU usage is the normalized amount of CPU used by a task corresponding to the machine. In this work, machine information is regarded as being represented by VM information.

6.4.3 Evaluation metrics

The objectives of our proposed VMA and VMC approach are to optimize the total energy consumption, reduce the number of VM migrations and active PMs of the data center. In addition, the objective of the employed algorithm for prediction is to improve the accuracy of workload forecasting while reducing the RMSE and MAE between the actual and predicted workload values.

- Energy Consumption: We consider the total energy consumption in a data center as the amount of electrical power used by the servers or PMs to process assigned applications workload. The energy usage of PMs within a data center is determined by how extensively their CPU, disk, memory and network are utilized. Most studies [283, 284] consistently demonstrate that the CPU consumes more electrical power compared to other components. Consequently, CPU utilization is frequently used as a metric to assess how well PM resources are used because it offers vital insight into the energy consumption profile of the machine. The energy consumption is calculated using real power consumption data from the SPECpower benchmark results. Table 6.1 shows the energy consumed, measured in watts (W), for two

distinct HP G4 and G5 server models at various load percentages.

Table 6.1: The energy consumption of PMs at load percentage of CPU

PM	Sleep	0	10	20	30	40	50	60	70	80	90	100
G4	10	86	89.4	92.6	96	99.5	102	106	108	112	114	117
G5	10	93.7	97	101	105	110	116	121	125	129	133	135

- Number of Migrations: The 'number of migrations' is the count of VM movements within the virtualized environment. These migrations encompass two scenarios: firstly, the selection of VMs from overloaded PMs to migrate and bring those PMs back to a normal operational state, and secondly, the migration of VMs from underloaded PMs to facilitate their shutdown. Throughout our study, we continually monitor all PMs and record the frequency of such VM migrations at both the VMS and VMP stages.

- Active number of PMs: We determine the number of active PMs within a cloud data center by assessing their CPU usage. A PM is classified as active when its CPU is in use, signifying that tasks are actively running on it and power is being consumed. Importantly, even if a machine is using a relatively small percentage of its CPU, we still classify it as active. This approach ensures that any degree of CPU utilization, no matter how minimal, qualifies the machine as active. When the count of active PMs in a data center is optimized, it indicates that the CPUs of all active PMs are effectively utilized, promoting efficient resource allocation and workload management.

- RMSE: This statistic is used to evaluate the accuracy of the employed model (i.e., LSTM) for resource usage prediction. RMSE is a widely used statistical metric that evaluates the error amount or difference between observed and predicted values. It measures the degree to which a predictive model matches real-world data, where lower RMSE values signify higher prediction accuracy. The following equation is used to calculate RMSE:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N}(P_i - A_i)^2}{N}} \tag{6.22}$$

  Where N represents the total number of data, P_i is the predicted value for the ith data, while A_i is the actual observed value for the ith data. RMSE is computed by taking the mean of the squared deviations between the predicted and observed values.

- MAE: Mean Absolute Error is another statistical metric used to evaluate the accuracy of the employed predictive model for resource usage of VM. In contrast to RMSE, MAE does not involve squaring the differences between predicted and actual values. Instead, it computes the absolute differences and subsequently computes their average. The following equation is used to calculate MAE:

$$\text{MAE} = \frac{\sum_{i=1}^{N} |P_i - A_i|}{N} \tag{6.23}$$

Let N be the number of data points, P_i denotes the predicted value for the ith data, while A_i represents the actual observed value for the ith data.

6.5 Results and Discussion

In this section, we have initially presented the results obtained for VMA using NSGA-II in Tier 1, comparing them with other existing algorithms. Subsequently, we have displayed the results of the implemented LSTM model in Tier 2. Finally, we present the results achieved by the proposed VMC approach and compare them with other benchmark algorithms.

6.5.1 Virtual machine allocation (Tier 1):

We conducted multiple experiments to evaluate the performance of our proposed VMA approach. The optimal number of active PMs indicates that the resources of PMs are appropriately distributed among VMs and effectively managed. Therefore, we tested our proposed approach over ten situations to analyze and compare it with existing techniques. For this experiment, we created a simulation environment of 100 heterogenous PMs and obtained results for allocating VM requests ranging from 10 to 100 using the proposed NSGAII-VMAC algorithm. In addition, we have also assigned VM requests using random allocation (RA), genetic algorithm (GA), best-fit decrease (BFD), and ant colony optimization (ACO) and compared all the achieved results. Figure 6.3 shows that if there were requests for 10 VMs, using RA, GA, BFD, and ACO requires 8,6, 4, and 4 PMs to allocate them. At the same time, using NSGAII-VMAC, they were assigned to 3 machines. As the number of VM requests increases to 50, the RA, GA, BFD, and ACO allocation results in 32, 30, 19, and 18 PMs, respectively. However, NSGAII-VMAC will assign them to 14 PMs. Moreover, when the requests further increase to 100, RA, GA, BFD, and ACO will require 56, 52, 38, and 36 PMs, respectively. Remarkably, even in this scenario, NSGAII-VMAC efficiently allocates them to 30 PMs.

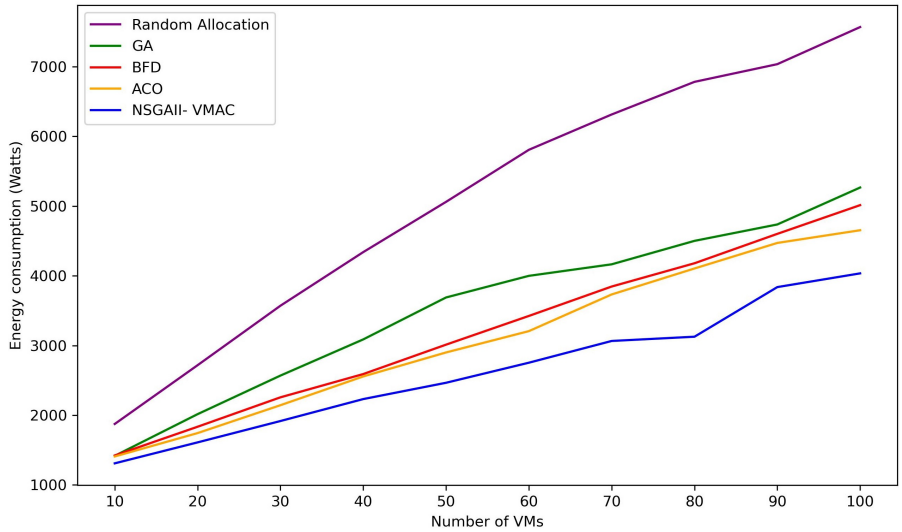


Figure 6.3: Energy consumption corresponding to the number of VM requests for VMA

Following our Tier 1 goals, we have also examined the total amount of energy consumed by PMs in a data center that depends on the type of active machines (G4 and G5) along with the number of PMs. Figure 6.4 depicts the energy consumption of a data center measured in Watts with respect to the number of requested VMs. Most of the studies employed BFD for the initial allocation of VMs requested [189, 146, 272]; therefore, we analyzed that when dealing with 50 VM requests, RA and GA consume 68.03% and 22.47% respectively more energy than BFD, while ACO reduces energy consumption by 3.72%. However, the most remarkable reduction is 18.12%, achieved by NSGAII-VMAC. Similarly, when handling 100 VM requests, RA and GA consume 50.96% and 5.04% more energy than BFD, with ACO achieving a 7.17% reduction. NSGAII-VMAC leads with the most significant energy reduction, reaching 19.52%. Hence, NSGAII-VMAC consistently outperforms by achieving the lowest energy consumption in all observed scenarios. In addition, we assess the collective performance of Tier 1 and Tier 3 using our proposed methodology. To do so, we established a data center comprising 100 heterogeneous PMs and 50 VMs, as outlined in 6.4.1.

6.5.2 Workload prediction (Tier 2):

We used random 50 machine information from the GCT dataset as a workload of 50 assigned VMs. The LSTM model is employed to predict the next step (5-minute) CPU usage of VMs for a duration of 10 hours, where 70% of the data is used to train the model, and the remaining 30% is used to test the model's performance. To assess the accuracy of the prediction model, we used Root mean square error and Mean absolute error metrics. These metrics offered valuable insights into the model's performance and capacity to

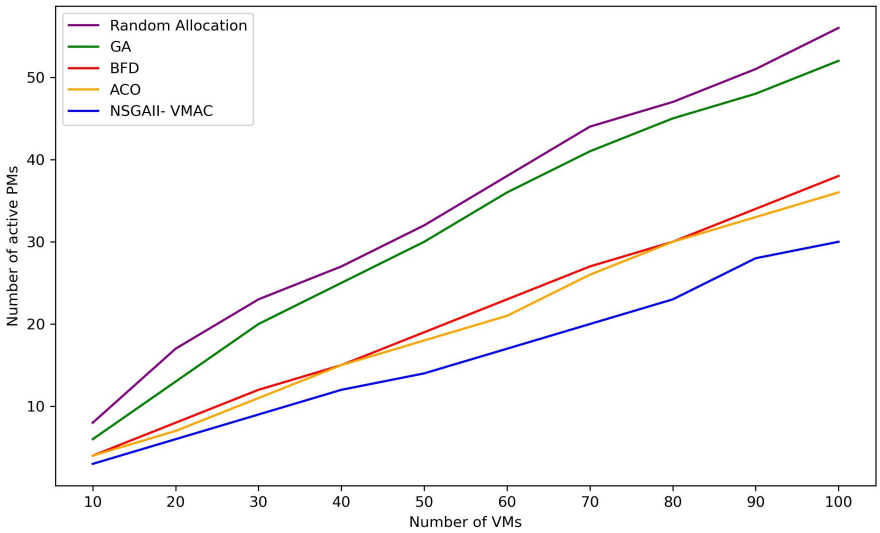


Figure 6.4: Number of active PMs corresponding to the number of VM requests during VMA process

generate precise predictions of CPU usage for VMs. To evaluate the performance of the LSTM and other benchmark models, we tested every individual collected 50 VMs.

Figure 6.5 represents the achieved RMSE value of LR, ARIMA, K-Nearest Neighbor Regression (KNNR) and LSTM using a box plot. The box shows the interquartile range (IQR), which covers the middle 50% of the RMSE. A horizontal line within the box symbolizes the median RMSE when arranged in ascending order. The lines that extend from the box are the whiskers, which indicate the range of achieved RMSE, excluding outliers. Outliers are RMSE values that are significantly different from the majority of the values. The results show that the RMSE values for the LR model fall within the range of 0.052 to 0.368, for ARIMA between 0.043 and 0.337, and for KNNR from 0.047 to 0.325, whereas the LSTM model consistently outperforms on all the machines with its lower range from 0.028 to 0.132.

Similarly, the achieved MAE value of all the models also depicts that LSTM outperforms, as shown in Figure 6.6. The MAE values for all machines under different models are as follows: for LR, ranges from 0.040 to 0.303; for ARIMA, it falls within the range of 0.029 to 0.280; for KNNR, the range spans from 0.037 to 0.270; and for LSTM, ranging from 0.021 to 0.111.

6.5.3 Virtual machine consolidation (Tier 3):

We conducted an analysis and performance evaluation of the proposed VMC approach, employing three key metrics: Energy consumption, number of active PMs and number of VMs migration. We compared the obtained metric results with those of existing ap-

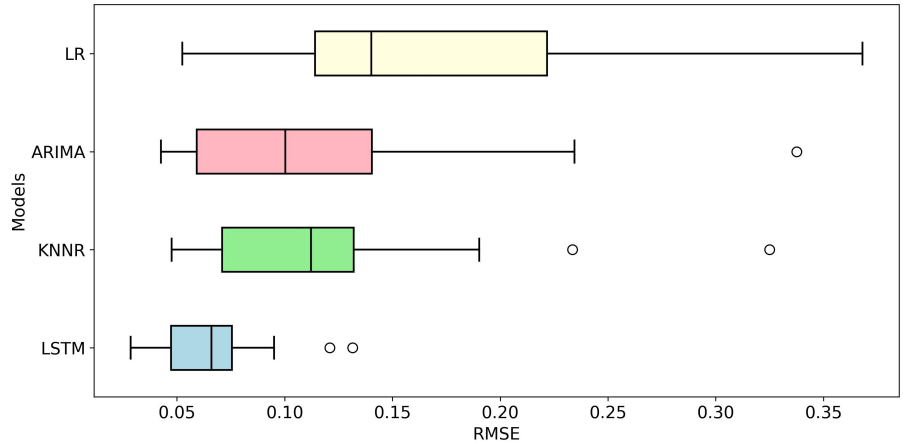


Figure 6.5: Model Performance for workload prediction based on RMSE

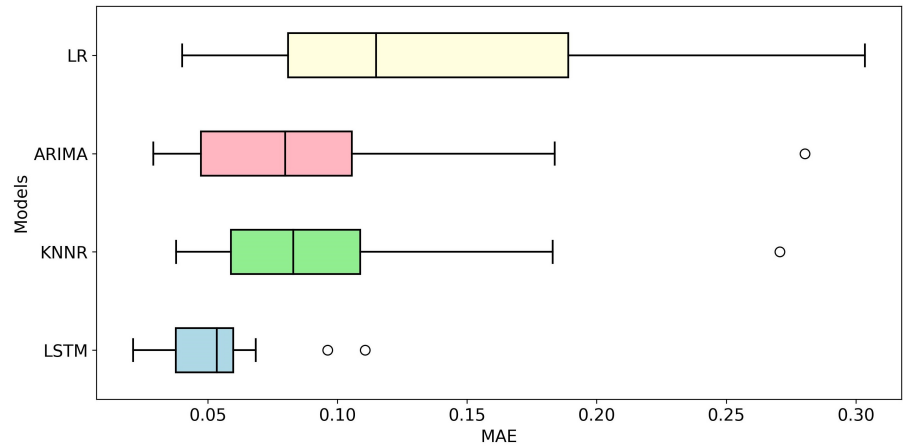


Figure 6.6: Model Performance for workload prediction based on MAE

proaches, including Utilization Prediction-aware VM Consolidation (UP-VMC), Modified Best Fit Decreasing (MBFD), Modified First Fit Decreasing (MFFD), and ACS-based VM Consolidation (ACS-VMC).

For this experiment, we calculated the average energy consumption and the number of active PMs for each hour and aggregated the count of active PMs over 10 hours. Figure 6.7 shows that the NSGAII-VMAC algorithm (proposed approach) leads to significantly less energy consumption in each hour than the other four benchmark algorithms. This arises from the fact that we commence VM allocation and placement using the NSGA-II algorithm, which prioritizes assigning VMs to PMs based on minimizing energy consumption. Consequently, this approach leads to optimizing overall energy consumption in a data center. The proposed 3-tier methodology significantly reduces average energy consumption by 37.49%, 41.81%, 45.40%, and 50.934% compared to UP-VMC, ACS-VMC, MBFD, and MFFD, respectively.

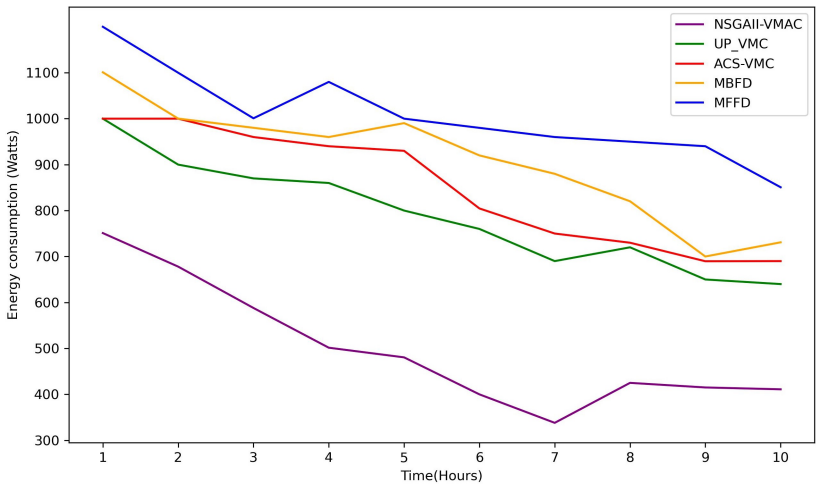


Figure 6.7: Hourly total energy consumption by NSGAII-VMAC and benchmark methods

Figure 6.8 represents that NSGAII-VMAC also outperforms in reducing the total number of active PMs for each hour compared to the other four algorithms. This is because NSGAII-VMAC is a multi-objective approach that also focuses on minimizing the count of active PMs in both the VMA and VMP phases. The reduction in the number of active PMs is further enhanced by accurate workload predictions, enabling the identification of presently underloaded PMs and those expected to be underloaded in the near future. This, in turn, facilitates more informed decision-making and allows for migrating all VMs from the identified underloaded PMs to optimize system shutdowns. In addition, we also consider that during changing a PM state from overloaded to normal, we ensure that its CPU usage is maximized after migrating selected VMs. This approach enables us to activate an optimal number of PMs by fully utilizing their capabilities. The proposed 3-tier methodology, compared to UP-VMC, ACS-VMC, MBFD, and MFFD, achieves remarkable reductions in the average total number of active servers by 39.62%,46.84%,49.86%, and 60.39%, respectively.

Figure 6.9 depicts that NSGAII-VMAC is reducing the total number of VM migrations every hour compared to other benchmark algorithms. This became possible because, during VM migration from overloaded PMs, our objective was to select the minimum number of VMs for migration by utilizing information on current and predicted workloads of VM and PM. The proposed 3-tier methodology reduces the average number of VM migrations by 7.85%, 20.64%, 28.83%, and 30.57% compared to UP-VMC, ACS-VMC, MBFD, and MFFD, respectively.

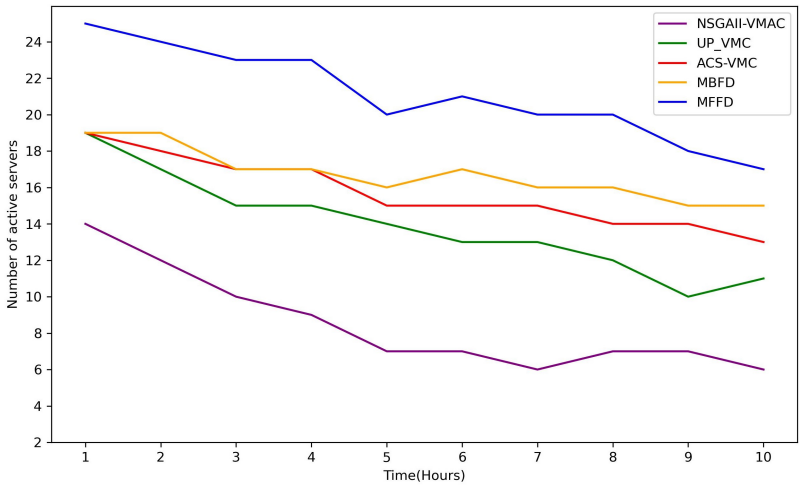


Figure 6.8: Hourly number of active servers by NSGAII-VMAC and benchmark methods

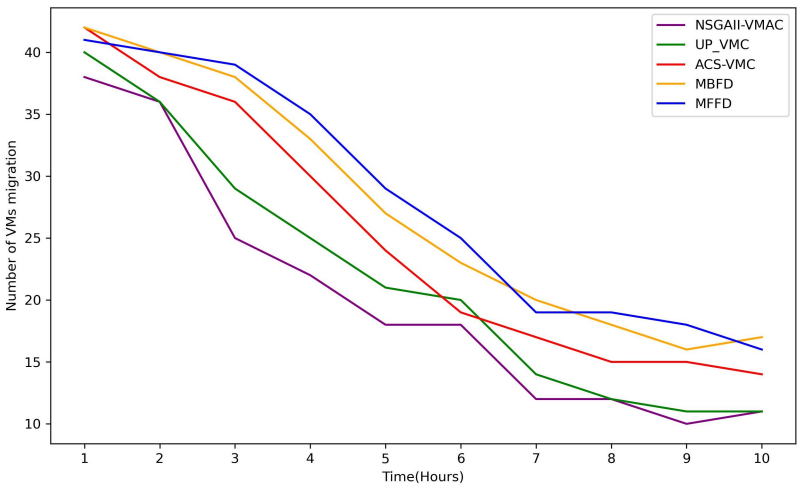


Figure 6.9: Hourly VM migrations count by NSGAII-VMAC and other methods

6.6 Summary

This chapter introduces a three-tier architecture that optimizes both key aspects of the cloud computing environment: VM allocation and dynamic consolidation. The architecture helps to optimally allocate requested VMs, accurately predict the resource usage of machines, and dynamically consolidate VMs that achieve minimum energy usage, optimal number of active servers, minimum number of VM migrations, and less resource usage prediction error. To address VMA and VMC, we employed an NSGA-II-based virtual machine allocation and consolidation algorithm. Tier 1 utilizes NSGAII-VMAC for the optimal mapping of VM to PM. The algorithm considers resource usage and user request constraints, with the dual objective of reducing energy usage and the number of active servers. Subsequently, Tier 2 employed the LSTM model to predict the accurate resource

usage of VMs and calculate the predicted usage of PM by aggregating the predicted values of assigned VMs on each respective PM. At last, Tier 3 dynamically consolidates VMs to reduce the energy consumption of the data center, minimize migration count, and optimize resource utilization on every active PM. This Tier follows a three-step process: firstly, it identifies overloaded and underloaded PMs using threshold-based criteria. Secondly, it selects VMs for migration utilizing a proposed novel method inspired by the Pareto front strategy. Finally, it places VMs to designated PM using NSGAII-VMAC. This Tier relies on current and predicted resource usage to inform decision-making at each step. We evaluated the performance of each Tier individually as well as the entire architecture as a cohesive unit. The presented architecture outperforms state-of-the-art approaches.

Chapter 7

Conclusions & Future Scopes

This chapter concludes the discussion on effective resource management using deep learning and optimization algorithms in CDCs. Furthermore, it explores unresolved issues that could serve as potential areas for future research in these domains.

7.1 Conclusions

The thesis provides a literature survey describing various methodologies, taxonomies, and comparative analyses related to resource usage prediction, virtual machine allocation, and consolidation. We have identified research gaps within existing approaches, such as the need for a preprocessing method to extract intact machine resource usage from a shared resource environment, a defined method to capture heterogeneous machines from the dataset to evaluate the employed models, the impact of dependencies and input sequence order during resource usage prediction, and the development of an effective strategy to manage data center resources with the aim to reduce energy consumption.

To confront these gaps, a fusion strategy integrating the GMM with the LSTM model is carried out. GMM is employed to select a set of heterogeneous machines, which divides available machines into components based on the mean CPU usage and canonical memory usage of PMs. Afterwards, the LSTM model was employed with optimal hyperparameters that gave the best CPU usage prediction for all the selected sets of machines. The proposed model demonstrates an improvement of approximately 66.78%, 63.13% and 62.80% compared to LR, ARIMA, and MA, respectively.

Furthermore, to capture the long-range, short-term and input sequence order on resource usage of PM prediction, LSTM, Transformer, and Informer models have been carried out. In addition, BIRCH is employed to rationally select the set of heterogeneous machines based on mean CPU, canonical memory and local hard disk usage. The evaluation of the three models reveals that the Transformer model, which accounts for long-range dependencies and addresses dataset limitations, demonstrates improved forecasting performance with a 14.2% reduction in RMSE compared to LSTM. However, LSTM exhibits superior results for certain machines compared to the Transformer, highlighting the signif-

icance of input sequence order. The Informer model, which integrates both dependencies and input sequence in order, surpasses both models with a 21.7% reduction in RMSE compared to LSTM and a 20.8% reduction compared to Transformer. This concludes that to predict the long-term usage of machine resources, the informer model gives accurate results compared to the LSTM and Transformer models.

With the notion of designing a holistic solution for the CDC that could serve users' requests from initial allocation to dynamic VM consolidation, rigorous experiments are carried out using meta-heuristic single-objective optimization techniques. FF, FFD, BF, BFD, and GA are used for initial VM allocation with an aim to minimize the number of active PMs while maximizing resource usage on each machine. The results depicts that the GA algorithm excelled in allocating requested VMs to the fewest PMs compared to other algorithms. However, it did not consistently result in reduced overall energy consumption of the data center. The outcome of this experiment explicitly states that considering both the number of active machines and energy usage as independent objectives during the allocation process provides a more effective strategy for resource management and reduces overall energy consumption in CDCs.

Finally, a multi-objective three-tier architecture has been presented, which aims to provide a solution for effective resource management while reducing the overall energy consumption of CDC. This architecture enables the CDC from VMA to dynamic consolidation using current and predicted resource usage of machines. Tier-1 is dedicated towards optimal VMA using the NSGA-II, which benefits by achieving up to 46.43%, 21.05%, 16.67% and 42.31% reduction in the number of active machines compared to RA, BFD, ACO and GA. In addition, it achieved an approximately 46.72%, 19.00%, 14.01% and 22.28% reduction in energy consumption compared to RA, BFD, ACO and GA. Tier-2 centred on predicting resource usage for all assigned VMs using an LSTM model. This tier achieves more accurate current as well as predicted resource usage of PMs validated from RMSE and MAE. Tier-3 dynamically consolidates VMs through a 3-step procedure: detection of overloaded and underloaded PM, VMS and VMP. This study investigates the intricate relationship between VM allocation and consolidation, ultimately reducing the number of active PMs, lowering energy consumption, minimizing migration counts, and maximizing resource usage by accurately predicting resource demands with minimal RMSE. Extensive experiments show a significant reduction in energy consumption up to 50.93% while the number of active machines and migrations are reduced to 60.39% and 30.57%, respectively, as compared to UP-VMC, ACS-VMC, MBFD, and MFFD.

A real-world GCT was utilized to assess and validate the techniques employed in our study. This dataset was chosen for its ability to capture the dynamic resource utilization

characteristics across diverse heterogeneous machines. To address the challenge of overlapping resource usage between jobs with differing time intervals, a Sum-Average (SA) algorithm was introduced for data pre-processing. This algorithm effectively resolves the issue by extracting the precise mean resource usage of machines within uniform time intervals, thereby enhancing the accuracy of our analyses. Additionally, this algorithm played a crucial role in preparing the data used as input for the employed clustering and predicting techniques.

7.2 Future Scopes

The research can be expanded in the following areas:

1. Algorithms could be further validated by simulating them with larger real-time datasets containing higher numbers of jobs to ensure stability across various scenarios.

2. Implementing all proposed algorithms in a real-time cloud environment, such as CloudSim, with more complex workload models would allow for a comparison of simulation results against those generated in a real cloud setting. This validation process is essential for confirming the effectiveness and practical applicability of the algorithms in the real-world cloud environment.

References

[1] Devesh Lowe and Bhavna Galhotra. An overview of pricing models for using cloud services with analysis on pay-per-use model. *International Journal of Engineering & Technology*, 7(3.12):248–254, 2018.

[2] Chnar Mustafa Mohammed and Subhi RM Zeebaree. Sufficient comparison among cloud computing services: Iaas, paas, and saas: A review. *International Journal of Science and Business*, 5(2):17–30, 2021.

[3] Yuanyuan Zhang, Wei Sun, and Yasushi Inoguchi. Predict task running time in grid environments based on cpu load predictions. *Future Generation Computer Systems*, 24(6):489–497, 2008.

[4] Peter A Dinda and David R O'Hallaron. Host load prediction using linear models. *Cluster Computing*, 3:265–280, 2000.

[5] K Beghdad Bey, Farid Benhammadi, Aicha Mokhtari, and Zahia Guessoum. Cpu load prediction model for distributed computing. In *2009 Eighth International Symposium on Parallel and Distributed Computing*, pages 39–45. IEEE, 2009.

[6] Sun-Yuan Hsieh, Cheng-Sheng Liu, Rajkumar Buyya, and Albert Y Zomaya. Utilization-prediction-aware virtual machine consolidation approach for energy-efficient cloud data centers. *Journal of Parallel and Distributed Computing*, 139:99–109, 2020.

[7] Carlos E Gómez, César O Díaz, César A Forero, Eduardo Rosales, and Harold Castro. Determining the real capacity of a desktop cloud. In *High Performance Computing: Second Latin American Conference, CARLA 2015, Petrópolis, Brazil, August 26-28, 2015, Proceedings 2*, pages 62–72. Springer, 2015.

[8] Anton Beloglazov and Rajkumar 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*, 24(7):1366–1379, 2012.

[9] Anshul Gandhi, Mor Harchol-Balter, Rajarshi Das, and Charles Lefurgy. Optimal power allocation in server farms. *ACM SIGMETRICS Performance Evaluation Review*, 37(1):157–168, 2009.

[10] Ramya Raghavendra, Parthasarathy Ranganathan, Vanish Talwar, Zhikui Wang, and Xiaoyun Zhu. No" power" struggles: coordinated multi-level power management for the data center. In *Proceedings of the 13th international conference on Architectural support for programming languages and operating systems*, pages 48–

59, 2008.

[11] US Environmental Protection Agency. Epa report on server and data center energy efficiency. *ENERGY STAR Program*, 2007.

[12] I Gartner. Gartner estimates ict industry accounts for 2 percent of global co2 emissions. *Press Releases. Available online: http://www. gartner. com/it/page. jsp (accessed on 15 April 2021)*, 2007.

[13] Frederico Durao, Jose Fernando S Carvalho, Anderson Fonseka, and Vinicius Cardoso Garcia. A systematic review on cloud computing. *The Journal of Supercomputing*, 68:1321–1346, 2014.

[14] Vinicius Cardoso Garcia. Frederico durao, jose fernando s. carvalho, anderson fonseka &. *J Supercomput*, 68:1321–1346, 2014.

[15] John McCarthy. Reminiscences on the history of time-sharing. *IEEE Annals of the History of Computing*, 14(1):19–24, 1992.

[16] Martin Campbell-Kelly, William F Aspray, Jeffrey R Yost, Honghong Tinn, and Gerardo Con Díaz. *Computer: A history of the information machine*. Routledge, 2023.

[17] Qi Zhang, Lu Cheng, and Raouf Boutaba. Cloud computing: state-of-the-art and research challenges. *Journal of internet services and applications*, 1:7–18, 2010.

[18] Matthew NO Sadiku, Sarhan M Musa, and Omonowo D Momoh. Cloud computing: opportunities and challenges. *IEEE potentials*, 33(1):34–36, 2014.

[19] Mark Spreeuwenberg, Marko van Eekelen, Ben Dankbaar, and René Schreurs. Cloud computing. *Unpublished Thesis), Radboud University, Faculty of Science, Computing Science, Nijmegen*, 2016.

[20] Luis M Vaquero, Luis Rodero-Merino, Juan Caceres, and Maik Lindner. A break in the clouds: towards a cloud definition, 2008.

[21] Hannah Wald. Cloud computing for the federal community. *Cloud Computing: Silver Lining or*, page 10, 2010.

[22] Anupama Prasanth. Cloud computing services: A survey. *International Journal of Computer Applications*, 46(3):25–29, 2012.

[23] Maricela-Georgiana Avram. Advantages and challenges of adopting cloud computing from an enterprise perspective. *Procedia Technology*, 12:529–534, 2014.

[24] Yashpalsinh Jadeja and Kirit Modi. Cloud computing-concepts, architecture and challenges. In *2012 international conference on computing, electronics and electrical technologies (ICCEET)*, pages 877–880. IEEE, 2012.

[25] David E Williams. *Virtualization with Xen (tm): Including XenEnterprise, XenServer, and XenExpress*. Elsevier, 2007.

[26] Yasunori Goto. Kernel-based virtual machine technology. *Fujitsu Scientific and*

Technical Journal, 47(3):362–368, 2011.

[27] Song Wu, Li Deng, Hai Jin, Xuanhua Shi, Yankun Zhao, Wei Gao, Jianyin Zhang, and Jin Peng. Virtual machine management based on agent service. In *2010 International Conference on Parallel and Distributed Computing, Applications and Technologies*, pages 199–204. IEEE, 2010.

[28] Eugene Gorelik. *Cloud computing models*. PhD thesis, Massachusetts Institute of Technology, 2013.

[29] Simon Ostermann, Radu Prodan, and Thomas Fahringer. Resource management for hybrid grid and cloud computing. *Cloud Computing: Principles, Systems and Applications*, pages 179–194, 2010.

[30] Rammohan Narendula. Amazon web services-a case study. 2012.

[31] Andrzej Kochut, Yu Deng, Michael R Head, J Munson, Anca Sailer, Hidayatullah Shaikh, Chunqiang Tang, Alexander Amies, Murray Beaton, David Geiss, et al. Evolution of the ibm cloud: Enabling an enterprise cloud services ecosystem. *IBM Journal of Research and Development*, 55(6):7–1, 2011.

[32] Filippo Lorenzo Ferraris, Davide Franceschelli, Mario Pio Gioiosa, Donato Lucia, Danilo Ardagna, Elisabetta Di Nitto, and Tabassum Sharif. Evaluating the auto scaling performance of flexiscale and amazon ec2 clouds. In *2012 14th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, pages 423–429. IEEE, 2012.

[33] Alexander Zahariev. Google app engine. *Helsinki University of Technology*, pages 1–5, 2009.

[34] Roger Jennings. *Cloud computing with the Windows Azure platform*. John Wiley & Sons, 2010.

[35] Can Özcanli. A proposed framework for crm on-demand system evaluation: Evaluation salesforce. com crm and microsoft dynamics online, 2012.

[36] Marek Jelen. *Platform for deploying web applications*. PhD thesis, Masarykova univerzita, Fakulta informatiky, 2012.

[37] Zeeshan Pervez, Asad Masood Khattak, Sungyoung Lee, and Young-Koo Lee. Dual validation framework for multi-tenant saas architecture. In *2010 5th International Conference on Future Information Technology*, pages 1–5. IEEE, 2010.

[38] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy H Katz, Andrew Konwinski, Gunho Lee, David A Patterson, Ariel Rabkin, Ion Stoica, et al. Above the clouds: A berkeley view of cloud computing. Technical report, Technical Report UCB/EECS-2009-28, EECS Department, University of California . . . , 2009.

[39] Tinankoria Diaby and Babak Bashari Rad. Cloud computing: a review of the concepts and deployment models. *International Journal of Information Technology*

and Computer Science, 9(6):50–58, 2017.

[40] Hiral B Patel and Nirali Kansara. Cloud computing deployment models: A comparative study. *International Journal of Innovative Research in Computer Science & Technology (IJIRCST)*, 2021.

[41] Brian Beach, Steven Armentrout, Rodney Bozo, Emmanuel Tsouris, Brian Beach, Steven Armentrout, Rodney Bozo, and Emmanuel Tsouris. Virtual private cloud. *Pro Powershell for Amazon Web Services*, pages 85–115, 2019.

[42] F John Krautheim. Private virtual infrastructure for cloud computing. *HotCloud*, 9:1–5, 2009.

[43] Caesar Wu and Rajkumar Buyya. *Cloud Data Centers and Cost Modeling: A complete guide to planning, designing and building a cloud data center*. Morgan Kaufmann, 2015.

[44] Kapil Bakshi. Considerations for cloud data centers: Framework, architecture and adoption. In *2011 Aerospace Conference*, pages 1–7. IEEE, 2011.

[45] Susanta Nanda Tzi-cker Chiueh and Stony Brook. A survey on virtualization technologies. *Rpe Report*, 142, 2005.

[46] Michael Pearce, Sherali Zeadally, and Ray Hunt. Virtualization: Issues, security threats, and solutions. *ACM Computing Surveys (CSUR)*, 45(2):1–39, 2013.

[47] Raoul Hentschel, Christian Leyh, and Anne Petznick. Current cloud challenges in germany: the perspective of cloud service providers. *Journal of Cloud Computing*, 7:1–12, 2018.

[48] Muthu Ramachandran and Victor Chang. Towards performance evaluation of cloud service providers for cloud data security. *International Journal of Information Management*, 36(4):618–625, 2016.

[49] Saad Mustafa, Babar Nazir, Amir Hayat, Sajjad A Madani, et al. Resource management in cloud computing: Taxonomy, prospects, and challenges. *Computers & Electrical Engineering*, 47:186–203, 2015.

[50] Andrew J Younge, Gregor Von Laszewski, Lizhe Wang, Sonia Lopez-Alarcon, and Warren Carithers. Efficient resource management for cloud computing environments. In *International conference on green computing*, pages 357–364. IEEE, 2010.

[51] Awada Uchechukwu, Keqiu Li, Yanming Shen, et al. Energy consumption in cloud computing data centers. *International Journal of Cloud Computing and Services Science (IJ-CLOSER)*, 3(3):31–48, 2014.

[52] Zheng Li, Selome Tesfatsion, Saeed Bastani, Ahmed Ali-Eldin, Erik Elmroth, Maria Kihl, and Rajiv Ranjan. A survey on modeling energy consumption of cloud applications: deconstruction, state of the art, and trade-off debates. *IEEE Transactions on Sustainable Computing*, 2(3):255–274, 2017.

[53] Sambit Kumar Mishra, Bibhudatta Sahoo, and Priti Paramita Parida. Load balancing in cloud computing: a big picture. *Journal of King Saud University-Computer and Information Sciences*, 32(2):149–158, 2020.

[54] Shahbaz Afzal and Ganesh Kavitha. Load balancing in cloud computing–a hierarchical taxonomical classification. *Journal of Cloud Computing*, 8(1):22, 2019.

[55] Jitendra Singh. Study of response time in cloud computing. *International journal of information engineering and electronic business*, 6(5):36, 2014.

[56] Mohammed Alhamad, Tharam Dillon, Chen Wu, and Elizabeth Chang. Response time for cloud computing providers. In *Proceedings of the 12th International Conference on Information Integration and Web-based Applications & Services*, pages 603–606, 2010.

[57] Jaideep Moses, Ravi Iyer, Ramesh Illikkal, Sadagopan Srinivasan, and Konstantinos Aisopos. Shared resource monitoring and throughput optimization in cloud-computing datacenters. In *2011 IEEE International Parallel & Distributed Processing Symposium*, pages 1024–1033. IEEE, 2011.

[58] Rahul Yadav, Weizhe Zhang, Keqin Li, Chuanyi Liu, and Asif Ali Laghari. Managing overloaded hosts for energy-efficiency in cloud data centers. *Cluster Computing*, pages 1–15, 2021.

[59] Pankajdeep Kaur and Anita Rani. Virtual machine migration in cloud computing. *International Journal of Grid and Distributed Computing*, 8(5):337–342, 2015.

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

[61] Srimoyee Bhattacherjee, Sunirmal Khatua, and Sarbani Roy. A review on energy efficient resource management strategies for cloud. *Advanced Computing and Systems for Security: Volume Four*, pages 3–15, 2017.

[62] Bhupesh Kumar Dewangan, Amit Agarwal, Tanupriya Choudhury, Ashutosh Pasricha, and Suresh Chandra Satapathy. Extensive review of cloud resource management techniques in industry 4.0: Issue and challenges. *Software: Practice and Experience*, 51(12):2373–2392, 2021.

[63] Mohammad Aldossary. A review of dynamic resource management in cloud computing environments. *Computer Systems Science & Engineering*, 36(3), 2021.

[64] Avneesh Vashistha and Pushpneel Verma. A literature review and taxonomy on workload prediction in cloud data center. In *2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, pages 415–420. IEEE, 2020.

[65] Mohammad Masdari and Afsane Khoshnevis. A survey and classification of the workload forecasting methods in cloud computing. *Cluster Computing*, 23(4):2399–

2424, 2020.

[66] Absalom E Ezugwu, Seyed M Buhari, and Sahalu B Junaidu. Virtual machine allocation in cloud computing environment. *International Journal of Cloud Applications and Computing (IJCAC)*, 3(2):47–60, 2013.

[67] Swapnil M Parikh. A survey on cloud computing resource allocation techniques. In *2013 Nirma University International Conference on Engineering (NUiCONE)*, pages 1–5. IEEE, 2013.

[68] Rahmat Zolfaghari and Amir Masoud Rahmani. Virtual machine consolidation in cloud computing systems: Challenges and future trends. *Wireless Personal Communications*, 115(3):2289–2326, 2020.

[69] Jaspreet Singh and Navpreet Kaur Walia. A comprehensive review of cloud computing virtual machine consolidation. *IEEE Access*, 2023.

[70] Peiyun Zhang, MengChu Zhou, and Xuelei Wang. An intelligent optimization method for optimal virtual machine allocation in cloud data centers. *IEEE Transactions on Automation Science and Engineering*, 17(4):1725–1735, 2020.

[71] Anton Beloglazov and Rajkumar Buyya. Energy efficient allocation of virtual machines in cloud data centers. In *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, pages 577–578. IEEE, 2010.

[72] Eduard Zharikov, Sergii Telenyk, and Petro Bidyuk. Adaptive workload forecasting in cloud data centers. *Journal of Grid Computing*, 18(1):149–168, 2020.

[73] Shobhana Kashyap and Avtar Singh. Prediction-based scheduling techniques for cloud data center's workload: a systematic review. *Cluster Computing*, 26(5):3209–3235, 2023.

[74] Xiaoyong Tang, Xiaoyi Liao, Jie Zheng, and Xiaopan Yang. Energy efficient job scheduling with workload prediction on cloud data center. *Cluster Computing*, 21(3):1581–1593, 2018.

[75] Ibrahim Alzamil and Karim Djemame. Energy prediction for cloud workload patterns. In *Economics of Grids, Clouds, Systems, and Services: 13th International Conference, GECON 2016, Athens, Greece, September 20-22, 2016, Revised Selected Papers 13*, pages 160–174. Springer, 2017.

[76] Md Hasanul Ferdaus and Manzur Murshed. Energy-aware virtual machine consolidation in iaas cloud computing. *Cloud Computing: Challenges, Limitations and R&D Solutions*, pages 179–208, 2014.

[77] Liuhua Chen, Haiying Shen, and Stephen Platt. Cache contention aware virtual machine placement and migration in cloud datacenters. In *2016 IEEE 24th International Conference on Network Protocols (ICNP)*, pages 1–10. IEEE, 2016.

[78] Luis Carlos Jersak and Tiago Ferreto. Performance-aware server consolidation with

adjustable interference levels. In *Proceedings of the 31st Annual ACM symposium on applied computing*, pages 420–425, 2016.

[79] John Panneerselvam. *A prescriptive analytics approach for energy efficiency in datacentres*. University of Derby (United Kingdom), 2017.

[80] Jitendra Kumar and Ashutosh Kumar Singh. Workload prediction in cloud using artificial neural network and adaptive differential evolution. *Future Generation Computer Systems*, 81:41–52, 2018.

[81] Sheng Di, Derrick Kondo, and Walfredo Cirne. Host load prediction in a google compute cloud with a bayesian model. In *SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, pages 1–11. IEEE, 2012.

[82] Wei Fang, ZhiHui Lu, Jie Wu, and ZhenYin Cao. Rpps: A novel resource prediction and provisioning scheme in cloud data center. In *2012 IEEE Ninth International Conference on Services Computing*, pages 609–616. IEEE, 2012.

[83] Z Zhenzhong, W Haiyan, X Limin, and R Li. A statistical based resource allocation scheme in cloud. In *2011 International Conference on Cloud and Service Computing*.

[84] 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 *The International Conference on Information Networking 2014 (ICOIN2014)*, pages 40–45. IEEE, 2014.

[85] Jingqi Yang, Chuanchang Liu, Yanlei Shang, Bo Cheng, Zexiang Mao, Chunhong Liu, Lisha Niu, and Junliang Chen. A cost-aware auto-scaling approach using the workload prediction in service clouds. *Information Systems Frontiers*, 16:7–18, 2014.

[86] Rodrigo N Calheiros, Enayat Masoumi, Rajiv Ranjan, and Rajkumar Buyya. Workload prediction using arima model and its impact on cloud applications' qos. *IEEE transactions on cloud computing*, 3(4):449–458, 2014.

[87] Grzegorz Dudek. Pattern-based local linear regression models for short-term load forecasting. *Electric power systems research*, 130:139–147, 2016.

[88] Carlos Vazquez. *Time series forecasting of cloud data center workloads for dynamic resource provisioning*. The University of Texas at San Antonio, 2015.

[89] Keunsoo Kim, Changmin Lee, Jung Ho Jung, and Won Woo Ro. Workload synthesis: Generating benchmark workloads from statistical execution profile. In *2014 IEEE International Symposium on Workload Characterization (IISWC)*, pages 120–129. IEEE, 2014.

[90] Mohammad Aldossary, Ibrahim Alzamil, and Karim Djemame. Towards virtual

machine energy-aware cost prediction in clouds. In *Economics of Grids, Clouds, Systems, and Services: 14th International Conference, GECON 2017, Biarritz, France, September 19-21, 2017, Proceedings 14*, pages 119–131. Springer, 2017.

[91] Ying Liu, Daharewa Gureya, Ahmad Al-Shishtawy, and Vladimir Vlassov. Onlineelastman: self-trained proactive elasticity manager for cloud-based storage services. *Cluster Computing*, 20:1977–1994, 2017.

[92] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 153–167, 2017.

[93] Brijesh Patel and Partho Sengupta. Machine learning for predicting cardiac events: what does the future hold? *Expert review of cardiovascular therapy*, 18(2):77–84, 2020.

[94] Marko Grebovic, Luka Filipovic, Ivana Katnic, Milica Vukotic, and Tomo Popovic. Machine learning models for statistical analysis. *Int. Arab J. Inf. Technol.*, 20(3A):505–514, 2023.

[95] Vitor Cerqueira, Luis Torgo, and Carlos Soares. Machine learning vs statistical methods for time series forecasting: Size matters. *arXiv preprint arXiv:1909.13316*, 2019.

[96] Sardar Khaliq uz Zaman, Ali Imran Jehangiri, Tahir Maqsood, Nuhman ul Haq, Arif Iqbal Umar, Junaid Shuja, Zulfiqar Ahmad, Imed Ben Dhaou, and Mohammed F Alsharekh. Limpo: Lightweight mobility prediction and offloading framework using machine learning for mobile edge computing. *Cluster Computing*, 26(1):99–117, 2023.

[97] Sayaka Akioka and Yoichi Muraoka. Extended forecast of cpu and network load on computational grid. In *IEEE International Symposium on Cluster Computing and the Grid, 2004. CCGrid 2004.*, pages 765–772. IEEE, 2004.

[98] Arijit Khan, Xifeng Yan, Shu Tao, and Nikos Anerousis. Workload characterization and prediction in the cloud: A multiple time series approach. In *2012 IEEE Network Operations and Management Symposium*, pages 1287–1294. IEEE, 2012.

[99] Salam Ismaeel and Ali Miri. Using elm techniques to predict data centre vm requests. In *2015 IEEE 2nd international conference on cyber security and cloud computing*, pages 80–86. IEEE, 2015.

[100] Somnath Mazumdar and Anoop S Kumar. Forecasting data center resource usage: An experimental comparison with time-series methods. In *Proceedings of the Eighth International Conference on Soft Computing and Pattern Recognition (SoCPaR 2016)*, pages 151–165. Springer, 2018.

[101] Fan-Hsun Tseng, Xiaofei Wang, Li-Der Chou, Han-Chieh Chao, and Victor CM Leung. Dynamic resource prediction and allocation for cloud data center using the multiobjective genetic algorithm. *IEEE Systems Journal*, 12(2):1688–1699, 2017.

[102] Jiechao Gao, Haoyu Wang, and Haiying Shen. Machine learning based workload prediction in cloud computing. In *2020 29th international conference on computer communications and networks (ICCCN)*, pages 1–9. IEEE, 2020.

[103] T Deepika and P Prakash. Power consumption prediction in cloud data center using machine learning. *Int. J. Electr. Comput. Eng.(IJECE)*, 10(2):1524–1532, 2020.

[104] Ishana Attri, Lalit Kumar Awasthi, and Teek Parval Sharma. Machine learning in agriculture: a review of crop management applications. *Multimedia Tools and Applications*, 83(5):12875–12915, 2024.

[105] Qingchen Zhang, Laurence T Yang, Zhikui Chen, and Peng Li. A survey on deep learning for big data. *Information Fusion*, 42:146–157, 2018.

[106] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 43(12):4338–4364, 2020.

[107] Imran Ahmed, Misbah Ahmad, Gwanggil Jeon, and Francesco Piccialli. A framework for pandemic prediction using big data analytics. *Big Data Research*, 25:100190, 2021.

[108] Mohammad Ahsan, Madhu Kumari, and TP Sharma. Detection of context-varying rumors on twitter through deep learning. *Int. J. Adv. Sci. Technol*, 128:45–58, 2019.

[109] Truong Vinh Truong Duy, Yukinori Sato, and Yasushi Inoguchi. Improving accuracy of host load predictions on computational grids by artificial neural networks. *International Journal of Parallel, Emergent and Distributed Systems*, 26(4):275–290, 2011.

[110] John J Prevost, KranthiManoj Nagothu, Brian Kelley, and Mo Jamshidi. Prediction of cloud data center networks loads using stochastic and neural models. In *2011 6th International Conference on System of Systems Engineering*, pages 276–281. IEEE, 2011.

[111] Yao-Chung Chang, Ruay-Shiung Chang, and Feng-Wei Chuang. A predictive method for workload forecasting in the cloud environment. In *Advanced Technologies, Embedded and Multimedia for Human-centric Computing: HumanCom and EMC 2013*, pages 577–585. Springer, 2014.

[112] Qiangpeng Yang, Yu Zhou, Yao Yu, Jie Yuan, Xianglei Xing, and Sidan Du. Multi-step-ahead host load prediction using autoencoder and echo state networks in cloud computing. *The Journal of Supercomputing*, 71:3037–3053, 2015.

[113] Weishan Zhang, Bo Li, Dehai Zhao, Faming Gong, and Qinghua Lu. Workload

prediction for cloud cluster using a recurrent neural network. In *2016 International Conference on Identification, Information and Knowledge in the Internet of Things (IIKI)*, pages 104–109. IEEE, 2016.

[114] Binbin Song, Yao Yu, Yu Zhou, Ziqiang Wang, and Sidan Du. Host load prediction with long short-term memory in cloud computing. *The Journal of Supercomputing*, 74:6554–6568, 2018.

[115] Deepak Janardhanan and Enda Barrett. Cpu workload forecasting of machines in data centers using lstm recurrent neural networks and arima models. In *2017 12th international conference for internet technology and secured transactions (ICITST)*, pages 55–60. IEEE, 2017.

[116] Yongde Zhang, Fagui Liu, Bin Wang, Weiwei Lin, Guoxiang Zhong, Minxian Xu, and Keqin Li. A multi-output prediction model for physical machine resource usage in cloud data centers. *Future Generation Computer Systems*, 130:292–306, 2022.

[117] Ishana Attri, Lalit Kumar Awasthi, Teek Parval Sharma, and Priyanka Rathee. A review of deep learning techniques used in agriculture. *Ecological Informatics*, page 102217, 2023.

[118] Yongwei Wu, Yulai Yuan, Guangwen Yang, and Weimin Zheng. Load prediction using hybrid model for computational grid. In *2007 8th IEEE/ACM International Conference on Grid Computing*, pages 235–242. IEEE, 2007.

[119] Qiangpeng Yang, Chenglei Peng, He Zhao, Yao Yu, Yu Zhou, Ziqiang Wang, and Sidan Du. A new method based on psr and ea-gmdh for host load prediction in cloud computing system. *The Journal of Supercomputing*, 68:1402–1417, 2014.

[120] Zhijia Chen, Yuanchang Zhu, Yanqiang Di, and Shaochong Feng. Self-adaptive prediction of cloud resource demands using ensemble model and subtractive-fuzzy clustering based fuzzy neural network. *Computational intelligence and neuroscience*, 2015:17–17, 2015.

[121] Vassilis S Kodogiannis, Mahdi Amina, and Ilias Petrounias. A clustering-based fuzzy wavelet neural network model for short-term load forecasting. *International journal of neural systems*, 23(05):1350024, 2013.

[122] Katja Cetinski and Matjaz B Juric. Ame-wpc: Advanced model for efficient workload prediction in the cloud. *Journal of Network and Computer Applications*, 55:191–201, 2015.

[123] Chunhong Liu, Yanlei Shang, Li Duan, Shiping Chen, Chuanchang Liu, and Junliang Chen. Optimizing workload category for adaptive workload prediction in service clouds. In *Service-Oriented Computing: 13th International Conference, ICSOC 2015, Goa, India, November 16-19, 2015, Proceedings 13*, pages 87–104. Springer, 2015.

[124] Sandeep K Sood. Function points-based resource prediction in cloud computing. *Concurrency and Computation: Practice and Experience*, 28(10):2781–2794, 2016.

[125] Md Ebtidaul Karim, Mirza Mohd Shahriar Maswood, Sunanda Das, and Abdullah G Alharbi. Bhyprec: a novel bi-lstm based hybrid recurrent neural network model to predict the cpu workload of cloud virtual machine. *IEEE Access*, 9:131476–131495, 2021.

[126] Jing Chen, Yinglong Wang, et al. A hybrid method for short-term host utilization prediction in cloud computing. *Journal of Electrical and Computer Engineering*, 2019, 2019.

[127] Jing Chen and Yinglong Wang. A resource demand prediction method based on eemd in cloud computing. *Procedia Computer Science*, 131:116–123, 2018.

[128] Eva Patel and Dharmender Singh Kushwaha. A hybrid cnn-lstm model for predicting server load in cloud computing. *The Journal of Supercomputing*, 78(8):1–30, 2022.

[129] Amine Mrhari and Youssef Hadi. Workload prediction using vmd and tcn in cloud computing. *Journal of Advances in Information Technology Vol*, 13(3), 2022.

[130] Minxian Xu, Chenghao Song, Huaming Wu, Sukhpal Singh Gill, Kejiang Ye, and Chengzhong Xu. esdnn: deep neural network based multivariate workload prediction in cloud computing environments. *ACM Transactions on Internet Technology (TOIT)*, 22(3):1–24, 2022.

[131] Sania Malik, Muhammad Tahir, Muhammad Sardaraz, and Abdullah Alourani. A resource utilization prediction model for cloud data centers using evolutionary algorithms and machine learning techniques. *Applied Sciences*, 12(4):2160, 2022.

[132] Xing Xu, Hao Hu, Na Hu, and Weiqin Ying. Cloud task and virtual machine allocation strategy in cloud computing environment. In *Network Computing and Information Security: Second International Conference, NCIS 2012, Shanghai, China, December 7-9, 2012. Proceedings*, pages 113–120. Springer, 2012.

[133] Minoo Soltanshahi, Reza Asemi, and Nazi Shafiei. Energy-aware virtual machines allocation by krill herd algorithm in cloud data centers. *Heliyon*, 5(7), 2019.

[134] G Sreenivasulu and Ilango Paramasivam. Hybrid optimization algorithm for task scheduling and virtual machine allocation in cloud computing. *Evolutionary Intelligence*, 14:1015–1022, 2021.

[135] V Dinesh Reddy, GR Gangadharan, and G Subrahmanya VRK Rao. Energy-aware virtual machine allocation and selection in cloud data centers. *Soft Computing*, 23:1917–1932, 2019.

[136] Elham Hormozi, Shuwen Hu, Zhe Ding, Yu-Chu Tian, You-Gan Wang, Zu-Guo Yu, and Weizhe Zhang. Energy-efficient virtual machine placement in data centres

via an accelerated genetic algorithm with improved fitness computation. *Energy*, 252:123884, 2022.

[137] Xinqian Zhang, Tingming Wu, Mingsong Chen, Tongquan Wei, Junlong Zhou, Shiyan Hu, and Rajkumar Buyya. Energy-aware virtual machine allocation for cloud with resource reservation. *Journal of Systems and Software*, 147:147–161, 2019.

[138] Neeraj Kumar Sharma and Ram Mohana Reddy Guddeti. On demand virtual machine allocation and migration at cloud data center using hybrid of cat swarm optimization and genetic algorithm. In *2016 Fifth International Conference on Eco-friendly Computing and Communication Systems (ICECCS)*, pages 27–32. IEEE, 2016.

[139] Neeraj Kumar Sharma and G Ram Mohana Reddy. Multi-objective energy efficient virtual machines allocation at the cloud data center. *IEEE Transactions on Services Computing*, 12(1):158–171, 2016.

[140] Jing V Wang, Chi-Tsun Cheng, and K Tse Chi. A power and thermal-aware virtual machine allocation mechanism for cloud data centers. In *2015 IEEE International Conference on Communication Workshop (ICCW)*, pages 2850–2855. IEEE, 2015.

[141] An-ping Xiong, Chun-xiang Xu, et al. Energy efficient multiresource allocation of virtual machine based on pso in cloud data center. *Mathematical Problems in Engineering*, 2014, 2014.

[142] Mehran Tarahomi and Mohammad Izadi. A prediction-based and power-aware virtual machine allocation algorithm in three-tier cloud data centers. *International Journal of Communication Systems*, 32(3):e3870, 2019.

[143] Muhammad Faraz Manzoor, Adnan Abid, Muhammad Shoaib Farooq, Naeem A Nawaz, and Uzma Farooq. Resource allocation techniques in cloud computing: A review and future directions. *Elektronika ir Elektrotechnika*, 26(6):40–51, 2020.

[144] Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future generation computer systems*, 28(5):755–768, 2012.

[145] Seyed Saeid Masoumzadeh and Helmut Hlavacs. A gossip-based dynamic virtual machine consolidation strategy for large-scale cloud data centers. In *Proceedings of the Third International Workshop on Adaptive Resource Management and Scheduling for Cloud Computing*, pages 28–34, 2016.

[146] Fahimeh Farahnakian, Adnan Ashraf, Tapio Pahikkala, Pasi Liljeberg, Juha Plosila, Ivan Porres, and Hannu Tenhunen. Using ant colony system to consolidate vms for green cloud computing. *IEEE transactions on services computing*, 8(2):187–198, 2014.

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

[148] Dongyan Deng, Kejing He, and Yanhua Chen. Dynamic virtual machine consolidation for improving energy efficiency in cloud data centers. In *2016 4th international conference on cloud computing and intelligence systems (CCIS)*, pages 366–370. IEEE, 2016.

[149] Fahimeh Farahnakian, Rami Bahsoon, Pasi Liljeberg, and Tapio Pahikkala. Self-adaptive resource management system in iaas clouds. In *2016 IEEE 9th international conference on cloud computing (CLOUD)*, pages 553–560. IEEE, 2016.

[150] Fahimeh Farahnakian, Pasi Liljeberg, and Juha Plosila. Lircup: Linear regression based cpu usage prediction algorithm for live migration of virtual machines in data centers. In *2013 39th Euromicro conference on software engineering and advanced applications*, pages 357–364. IEEE, 2013.

[151] Anton Beloglazov. Energy-efficient management of virtual machines in data centers for cloud computing. 2013.

[152] Maede Yavari, Akbar Ghaffarpour Rahbar, and Mohammad Hadi Fathi. Temperature and energy-aware consolidation algorithms in cloud computing. *Journal of Cloud Computing*, 8(1):13, 2019.

[153] Nimisha Patel and Hiren Patel. Energy efficient strategy for placement of virtual machines selected from underloaded servers in compute cloud. *Journal of King Saud University-Computer and Information Sciences*, 32(6):700–708, 2020.

[154] Anurina Tarafdar, Mukta Debnath, Sunirmal Khatua, and Rajib K Das. Energy and quality of service-aware virtual machine consolidation in a cloud data center. *The Journal of Supercomputing*, 76:9095–9126, 2020.

[155] Anita Choudhary, Mahesh Chandra Govil, Girdhari Singh, Lalit K Awasthi, and Emmanuel S Pilli. Energy-efficient fuzzy-based approach for dynamic virtual machine consolidation. *International Journal of Grid and Utility Computing*, 10(4):308–325, 2019.

[156] Amany Abdelsamea, Ali A El-Moursy, Elsayed E Hemayed, and Hesham Eldeeb. Virtual machine consolidation enhancement using hybrid regression algorithms. *Egyptian Informatics Journal*, 18(3):161–170, 2017.

[157] Yaqiu Liu, Xinyue Sun, Wei Wei, and Weipeng Jing. Enhancing energy-efficient and qos dynamic virtual machine consolidation method in cloud environment. *IEEE Access*, 6:31224–31235, 2018.

[158] Nguyen Trung Hieu, Mario Di Francesco, and Antti Ylä-Jääski. Virtual machine

consolidation with multiple usage prediction for energy-efficient cloud data centers. *IEEE Transactions on Services Computing*, 13(1):186–199, 2017.

[159] Parminder Singh, Pooja Gupta, and Kiran Jyoti. Energy aware vm consolidation using dynamic threshold in cloud computing. In *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*, pages 1098–1102. IEEE, 2019.

[160] Amany Abdelsamea, Elsayed E Hemayed, Hesham Eldeeb, and Hanan Elazhary. Virtual machine consolidation challenges: A review. *International Journal of Innovation and Applied Studies*, 8(4):1504, 2014.

[161] Jyotsna Sengupta, Pardeep Singh, and PK Suri. Energy aware next fit allocation approach for placement of vms in cloud computing environment. In *Advances in Information and Communication: Proceedings of the 2020 Future of Information and Communication Conference (FICC), Volume 2*, pages 436–453. Springer, 2020.

[162] Lianpeng Li, Jian Dong, Decheng Zuo, and Jin Wu. Sla-aware and energy-efficient vm consolidation in cloud data centers using robust linear regression prediction model. *IEEE Access*, 7:9490–9500, 2019.

[163] Jean Pepe Buanga Mapetu, Lingfu Kong, and Zhen Chen. A dynamic vm consolidation approach based on load balancing using pearson correlation in cloud computing. *The Journal of Supercomputing*, 77(6):5840–5881, 2021.

[164] Youssef Saadi and Said El Kafhali. Energy-efficient strategy for virtual machine consolidation in cloud environment. *Soft Computing*, 24(19):14845–14859, 2020.

[165] Swasthi Shetty and B Annappa. Growth potential aware virtual machine consolidation framework. *Authorea Preprints*, 2023.

[166] Gamal Eldin I Selim, Mohamed A El-Rashidy, and Nawal A El-Fishawy. An efficient resource utilization technique for consolidation of virtual machines in cloud computing environments. In *2016 33rd national radio science conference (NRSC)*, pages 316–324. IEEE, 2016.

[167] Rahul Yadav, Weizhe Zhang, Huang Chen, and Tao Guo. Mums: Energy-aware vm selection scheme for cloud data center. In *2017 28th International Workshop on Database and Expert Systems Applications (DEXA)*, pages 132–136. IEEE, 2017.

[168] Hui Xiao, Zhigang Hu, and Keqin Li. Multi-objective vm consolidation based on thresholds and ant colony system in cloud computing. *IEEE Access*, 7:53441–53453, 2019.

[169] Farhad Ahamed, Seyed Shahrestani, and Bahman Javadi. Security aware and energy-efficient virtual machine consolidation in cloud computing systems. In *2016 IEEE Trustcom/BigDataSE/ISPA*, pages 1516–1523. IEEE, 2016.

[170] Herve Abdi. Multiple correlation coefficient. *Encyclopedia of measurement and statistics*, 648(651):19, 2007.

[171] Chengyu Yan, Zhihua Li, Xinrong Yu, and Ning Yu. Bayesian networks-based selection algorithm for virtual machine to be migrated. In *2016 IEEE International Conferences on Big Data and Cloud Computing (BDCloud), Social Computing and Networking (SocialCom), Sustainable Computing and Communications (SustainCom)(BDCloud-SocialCom-SustainCom)*, pages 573–578. IEEE, 2016.

[172] Mohammad Alaul Haque Monil and Rashedur M Rahman. Implementation of modified overload detection technique with vm selection strategies based on heuristics and migration control. In *2015 IEEE/ACIS 14th International Conference on Computer and Information Science (ICIS)*, pages 223–227. IEEE, 2015.

[173] Mohammad Alaul Haque Monil and Rashedur M Rahman. Vm consolidation approach based on heuristics, fuzzy logic, and migration control. *Journal of Cloud Computing*, 5:1–18, 2016.

[174] Nidhi Jain Kansal and Inderveer Chana. Energy-aware virtual machine migration for cloud computing-a firefly optimization approach. *Journal of Grid Computing*, 14:327–345, 2016.

[175] Atefeh Khosravi, Saurabh Kumar Garg, and Rajkumar Buyya. Energy and carbon-efficient placement of virtual machines in distributed cloud data centers. In *Euro-Par 2013 Parallel Processing: 19th International Conference, Aachen, Germany, August 26-30, 2013. Proceedings 19*, pages 317–328. Springer, 2013.

[176] Dabiah Ahmed Alboaneen, Huaglory Tianfield, and Yan Zhang. Glowworm swarm optimisation algorithm for virtual machine placement in cloud computing. In *2016 Intl IEEE Conferences on Ubiquitous Intelligence Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, pages 808–814, 2016.

[177] Atefeh Khosravi, Lachlan LH Andrew, and Rajkumar Buyya. Dynamic vm placement method for minimizing energy and carbon cost in geographically distributed cloud data centers. *IEEE Transactions on Sustainable Computing*, 2(2):183–196, 2017.

[178] Atefeh Khosravi, Adel Nadjaran Toosi, and Rajkumar Buyya. Online virtual machine migration for renewable energy usage maximization in geographically distributed cloud data centers. *Concurrency and Computation: Practice and Experience*, 29(18):e4125, 2017.

[179] Fikru Feleke Moges and Surafel Lemma Abebe. Energy-aware vm placement algorithms for the openstack neat consolidation framework. *Journal of Cloud Computing*, 8(1):2, 2019.

[180] Da-Ming Zhao, Jian-Tao Zhou, and Keqin Li. An energy-aware algorithm for virtual

machine placement in cloud computing. *IEEE Access*, 7:55659–55668, 2019.

[181] Mohamed Abdel-Basset, Doaa El-Shahat, Mohamed Elhoseny, and Houbing Song. Energy-aware metaheuristic algorithm for industrial-internet-of-things task scheduling problems in fog computing applications. *IEEE Internet of Things Journal*, 8(16):12638–12649, 2020.

[182] Nagma Khattar, Jaiteg Singh, and Jagpreet Sidhu. An energy efficient and adaptive threshold vm consolidation framework for cloud environment. *Wireless Personal Communications*, 113:349–367, 2020.

[183] H Li, T Li, and Z Shuhua. Energy-performance optimisation for the dynamic consolidation of virtual machines in cloud computing. *International Journal of Services Operations and Informatics*, 9(1):62–82, 2018.

[184] Xiong Fu and Chen Zhou. Virtual machine selection and placement for dynamic consolidation in cloud computing environment. *Frontiers of Computer Science*, 9:322–330, 2015.

[185] KC Anupama, BR Shivakumar, and R Nagaraja. Resource utilization prediction in cloud computing using hybrid model. *International Journal of Advanced Computer Science and Applications*, 12(4), 2021.

[186] Ankita Srivastava and Narander Kumar. Multi-objective binary whale optimization-based virtual machine allocation in cloud environments. *International Journal of Swarm Intelligence Research (IJSIR)*, 14(1):1–23, 2023.

[187] Riman Mandal, Manash Kumar Mondal, Sourav Banerjee, and Utpal Biswas. An approach toward design and development of an energy-aware vm selection policy with improved sla violation in the domain of green cloud computing. *The Journal of Supercomputing*, 76:7374–7393, 2020.

[188] Amol C Adamuthe, Rajendra A Gage, and Gopakumaran T Thampi. Forecasting cloud computing using double exponential smoothing methods. In *2015 International Conference on Advanced Computing and Communication Systems*, pages 1–5. IEEE, 2015.

[189] Fahimeh Farahnakian, Tapio Pahikkala, Pasi Liljeberg, Juha Plosila, Nguyen Trung Hieu, and Hannu Tenhunen. Energy-aware vm consolidation in cloud data centers using utilization prediction model. *IEEE Transactions on Cloud Computing*, 7(2):524–536, 2016.

[190] Jose Antonio Pascual, Tania Lorido-Botrán, José Miguel-Alonso, and Jose Antonio Lozano. Towards a greener cloud infrastructure management using optimized placement policies. *Journal of Grid Computing*, 13:375–389, 2015.

[191] Huanlai Xing, Jing Zhu, Rong Qu, Penglin Dai, Shouxi Luo, and Muhammad Azhar Iqbal. An aco for energy-efficient and traffic-aware virtual machine placement in

cloud computing. *Swarm and Evolutionary Computation*, 68:101012, 2022.

[192] Xialin Liu, Junsheng Wu, Gang Sha, and Shuqin Liu. Virtual machine consolidation with minimization of migration thrashing for cloud data centers. *Mathematical Problems in Engineering*, 2020, 2020.

[193] Sudibyajyoti Jena, Likhet Kashori Sahu, Sambit Kumar Mishra, and Bibhudatta Sahoo. Vm consolidation based on overload detection and vm selection policy. In *2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, pages 252–256. IEEE, 2021.

[194] Seyed Yahya Zahedi Fard, Mohamad Reza Ahmadi, and Sahar Adabi. A dynamic vm consolidation technique for qos and energy consumption in cloud environment. *The Journal of Supercomputing*, 73(10):4347–4368, 2017.

[195] Nguyen Trung Hieu, Mario Di Francesco, and Antti Ylä-Jääski. Virtual machine consolidation with usage prediction for energy-efficient cloud data centers. In *2015 IEEE 8th International Conference on Cloud Computing*, pages 750–757. IEEE, 2015.

[196] Jinjiang Wang, Hangyu Gu, Junyang Yu, Yixin Song, Xin He, and Yalin Song. Research on virtual machine consolidation strategy based on combined prediction and energy-aware in cloud computing platform. *Journal of Cloud Computing*, 11(1):1–18, 2022.

[197] Dabiah Ahmed Alboaneen, Huaglory Tianfield, and Yan Zhang. Glowworm swarm optimisation algorithm for virtual machine placement in cloud computing. In *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)*, pages 808–814. IEEE, 2016.

[198] Mehboob Hussain, Lian-Fu Wei, Abdullah Lakhan, Samad Wali, Soragga Ali, and Abid Hussain. Energy and performance-efficient task scheduling in heterogeneous virtualized cloud computing. *Sustainable Computing: Informatics and Systems*, 30:100517, 2021.

[199] Ahmed Osman, Assim Sagahyroon, Raafat Aburukba, and Fadi Aloul. Optimization of energy consumption in cloud computing datacenters. *International Journal of Electrical & Computer Engineering (2088-8708)*, 11(1), 2021.

[200] Jitendra Kumar, Rimsha Goomer, and Ashutosh Kumar Singh. Long short term memory recurrent neural network (lstm-rnn) based workload forecasting model for cloud datacenters. *Procedia Computer Science*, 125:676–682, 2018.

[201] Raja Majid Ali Ujjan, Zeeshan Pervez, Keshav Dahal, Ali Kashif Bashir, Rao Mumtaz, and Jonathan González. Towards sflow and adaptive polling sampling for

deep learning based ddos detection in sdn. *Future Generation Computer Systems*, 111:763–779, 2020.

[202] Mohd Rafi Lone and Ekram Khan. A good neighbor is a great blessing: Nearest neighbor filtering method to remove impulse noise. *Journal of King Saud University-Computer and Information Sciences*, 34(10):9942–9952, 2022.

[203] Charles Reiss, John Wilkes, and Joseph L Hellerstein. Google cluster-usage traces: format+ schema. *Google Inc., White Paper*, pages 1–14, 2011.

[204] Shreelekha Pandey and Pritee Khanna. Content-based image retrieval embedded with agglomerative clustering built on information loss. *Computers & Electrical Engineering*, 54:506–521, 2016.

[205] Gopal Chand Gautam, TP Sharma, Vivek Katiyar, and Anil Kumar. Time synchronization protocol for wireless sensor networks using clustering. In *2011 International Conference on Recent Trends in Information Technology (ICRTIT)*, pages 417–422. IEEE, 2011.

[206] Amit Banerjee and Rajesh N Dave. Validating clusters using the hopkins statistic. In *2004 IEEE International conference on fuzzy systems (IEEE Cat. No. 04CH37542)*, volume 1, pages 149–153. IEEE, 2004.

[207] Godwin Ogbuabor and FN Ugwoke. Clustering algorithm for a healthcare dataset using silhouette score value. *International Journal of Computer Science & Information Technology (IJCSIT)*, 10(2):27–37, 2018.

[208] Eva Patel and Dharmender Singh Kushwaha. Clustering cloud workloads: k-means vs gaussian mixture model. *Procedia Computer Science*, 171:158–167, 2020.

[209] T Ryan. Lstms explained: A complete, technically accurate, conceptual guide with keras.

[210] P Pranav. Recurrent neural networks, the vanishing gradient problem, and long short-term memory.

[211] Jupyter notebook. `https://jupyter.org/`. Last accessed in December, 2021.

[212] Adnan Abid, Muhammad Faraz Manzoor, Muhammad Shoaib Farooq, Uzma Farooq, and Muzammil Hussain. Challenges and issues of resource allocation techniques in cloud computing. *KSII Transactions on Internet and Information Systems (TIIS)*, 14(7):2815–2839, 2020.

[213] Syed Hamid Hussain Madni, Muhammad Shafie Abd Latiff, Yahaya Coulibaly, and Shafi'i Muhammad Abdulhamid. Recent advancements in resource allocation techniques for cloud computing environment: a systematic review. *Cluster Computing*, 20:2489–2533, 2017.

[214] Abdul Hameed, Alireza Khoshkbarforoushha, Rajiv Ranjan, Prem Prakash Jayaraman, Joanna Kolodziej, Pavan Balaji, Sherali Zeadally, Qutaibah Marwan Malluhi,

Nikos Tziritas, Abhinav Vishnu, et al. A survey and taxonomy on energy efficient resource allocation techniques for cloud computing systems. *Computing*, 98:751–774, 2016.

[215] Rohit Ahuja and Sraban Kumar Mohanty. A scalable attribute-based access control scheme with flexible delegation cum sharing of access privileges for cloud storage. *IEEE Transactions on Cloud Computing*, 8(1):32–44, 2017.

[216] PP Dabral and Mharhoni Z Murry. Modelling and forecasting of rainfall time series using sarima. *Environmental Processes*, 4(2):399–419, 2017.

[217] Paluck Arora, Rajesh Mehta, and Rohit Ahuja. An adaptive medical image registration using hybridization of teaching learning-based optimization with affine and speeded up robust features with projective transformation. *Cluster Computing*, pages 1–21, 2023.

[218] Xiaona Ren, Rongheng Lin, and Hua Zou. A dynamic load balancing strategy for cloud computing platform based on exponential smoothing forecast. In *2011 IEEE international conference on cloud computing and intelligence systems*, pages 220–224. IEEE, 2011.

[219] Jinhui Huang, Chunlin Li, and Jie Yu. Resource prediction based on double exponential smoothing in cloud computing. In *2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*, pages 2056–2060. IEEE, 2012.

[220] Zia Ur Rahman, Omar Khadeer Hussain, and Farookh Khadeer Hussain. Time series qos forecasting for management of cloud services. In *2014 Ninth International Conference on Broadband and Wireless Computing, Communication and Applications*, pages 183–190. IEEE, 2014.

[221] Abraham Chandy et al. Smart resource usage prediction using cloud computing for massive data processing systems. *J Inf Technol*, 1(02):108–118, 2019.

[222] Akindele A Bankole and Samuel A Ajila. Predicting cloud resource provisioning using machine learning techniques. In *2013 26th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 1–4. IEEE, 2013.

[223] Tajwar Mehmood, Seemab Latif, and Sheheryaar Malik. Prediction of cloud computing resource utilization. In *2018 15th International Conference on Smart Cities: Improving Quality of Life Using ICT & IoT (HONET-ICT)*, pages 38–42. IEEE, 2018.

[224] Muhammad Tanveer, Bharat Richhariya, Riyaj Uddin Khan, Ashraf Haroon Rashid, Pritee Khanna, Mukesh Prasad, and Chin-Teng Lin. Machine learning techniques for the diagnosis of alzheimer's disease: A review. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 16(1s):1–

35, 2020.

[225] Martin Duggan, Karl Mason, Jim Duggan, Enda Howley, and Enda Barrett. Predicting host cpu utilization in cloud computing using recurrent neural networks. In *2017 12th international conference for internet technology and secured transactions (ICITST)*, pages 67–72. IEEE, 2017.

[226] Michael Borkowski, Stefan Schulte, and Christoph Hochreiner. Predicting cloud resource utilization. In *Proceedings of the 9th International Conference on Utility and Cloud Computing*, pages 37–42, 2016.

[227] Karl Mason, Martin Duggan, Enda Barrett, Jim Duggan, and Enda Howley. Predicting host cpu utilization in the cloud using evolutionary neural networks. *Future Generation Computer Systems*, 86:162–173, 2018.

[228] Szu-Yin Lin, Chi-Chun Chiang, Jung-Bin Li, Zih-Siang Hung, and Kuo-Ming Chao. Dynamic fine-tuning stacked auto-encoder neural network for weather forecast. *Future Generation Computer Systems*, 89:446–454, 2018.

[229] Hengheng Shen and Xuehai Hong. Host load prediction with bi-directional long short-term memory in cloud computing. *arXiv preprint arXiv:2007.15582*, 2020.

[230] Raja Majid Ali Ujjan, Zeeshan Pervez, and Keshav Dahal. Suspicious traffic detection in sdn with collaborative techniques of snort and deep neural networks. In *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, pages 915–920. IEEE, 2018.

[231] Soukaina Ouhame, Youssef Hadi, and Arif Ullah. An efficient forecasting approach for resource utilization in cloud data center using cnn-lstm model. *Neural Computing and Applications*, 33:10043–10055, 2021.

[232] Xuanyi Song, Yuetian Liu, Liang Xue, Jun Wang, Jingzhe Zhang, Junqiang Wang, Long Jiang, and Ziyan Cheng. Time-series well performance prediction based on long short-term memory (lstm) neural network model. *Journal of Petroleum Science and Engineering*, 186:106682, 2020.

[233] José F Torres, Dalil Hadjout, Abderrazak Sebaa, Francisco Martínez-Álvarez, and Alicia Troncoso. Deep learning for time series forecasting: a survey. *Big Data*, 9(1):3–21, 2021.

[234] Antonio Rafael Sabino Parmezan, Vinicius MA Souza, and Gustavo EAPA Batista. Evaluation of statistical and machine learning models for time series prediction: Identifying the state-of-the-art and the best conditions for the use of each model. *Information sciences*, 484:302–337, 2019.

[235] Neo Wu, Bradley Green, Xue Ben, and Shawn O'Banion. Deep transformer mod-

els for time series forecasting: The influenza prevalence case. *arXiv preprint arXiv:2001.08317*, 2020.

[236] Neha Gour and Pritee Khanna. Multi-class multi-label ophthalmological disease detection using transfer learning based convolutional neural network. *Biomedical signal processing and control*, 66:102329, 2021.

[237] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. A review of recurrent neural networks: Lstm cells and network architectures. *Neural computation*, 31(7):1235–1270, 2019.

[238] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 11106–11115, 2021.

[239] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[240] Preeti Khanna and M Sasikumar. Bimodal emotion recognition: A comparative study of rule based system vs classification algorithms. *International Journal*, 3(8), 2013.

[241] Richa Sharma, Amit M Joshi, Chitrakant Sahu, and Satyasai Jagannath Nanda. Detection of false data injection in smart grid using pca based unsupervised learning. *Electrical Engineering*, 105(4):2383–2396, 2023.

[242] S Patro and Kishore Kumar Sahu. Normalization: A preprocessing stage. *arXiv preprint arXiv:1503.06462*, 2015.

[243] RF Lachlan, L Verhagen, S Peters, and C ten Cate. Are there species-universal categories in bird song phonology and syntax? a comparative study of chaffinches (fringilla coelebs), zebra finches (taenopygia guttata), and swamp sparrows (melospiza georgiana). *Journal of Comparative Psychology*, 124(1):92, 2010.

[244] Hong Bo Zhou and Jun Tao Gao. Automatic method for determining cluster number based on silhouette coefficient. *Advanced materials research*, 951:227–230, 2014.

[245] Azaria Paz and Shlomo Moran. Non deterministic polynomial optimization problems and their approximations. *Theoretical Computer Science*, 15(3):251–277, 1981.

[246] Hamid Talebian, Abdullah Gani, Mehdi Sookhak, Ahmed Abdelaziz Abdelatif, Abdullah Yousafzai, Athanasios V Vasilakos, and Fei Richard Yu. Optimizing virtual machine placement in iaas data centers: taxonomy, review and open issues. *Cluster Computing*, 23:837–878, 2020.

[247] Jangha Kang and Sungsoo Park. Algorithms for the variable sized bin packing problem. *European Journal of Operational Research*, 147(2):365–372, 2003.

[248] Norman Bobroff, Andrzej Kochut, and Kirk Beaty. Dynamic placement of virtual machines for managing sla violations. In *2007 10th IFIP/IEEE International Symposium on Integrated Network Management*, pages 119–128. IEEE, 2007.

[249] Weiming Shi and Bo Hong. Towards profitable virtual machine placement in the data center. In *2011 Fourth IEEE International Conference on Utility and Cloud Computing*, pages 138–145. IEEE, 2011.

[250] Ankit Anand, J Lakshmi, and SK Nandy. Virtual machine placement optimization supporting performance slas. In *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*, volume 1, pages 298–305. IEEE, 2013.

[251] Hao Jin, Deng Pan, Jing Xu, and Niki Pissinou. Efficient vm placement with multiple deterministic and stochastic resources in data centers. In *2012 IEEE Global Communications Conference (GLOBECOM)*, pages 2505–2510. IEEE, 2012.

[252] Jiankang Dong, Hongbo Wang, Xing Jin, Yangyang Li, Peng Zhang, and Shiduan Cheng. Virtual machine placement for improving energy efficiency and network performance in iaas cloud. In *2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops*, pages 238–243. IEEE, 2013.

[253] Shuo Fang, Renuga Kanagavelu, Bu-Sung Lee, Chuan Heng Foh, and Khin Mi Mi Aung. Power-efficient virtual machine placement and migration in data centers. In *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, pages 1408–1413. IEEE, 2013.

[254] Dapeng Dong and John Herbert. Energy efficient vm placement supported by data analytic service. In *2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, pages 648–655. IEEE, 2013.

[255] Shao-Heng Wang, Patrick P-W Huang, Charles H-P Wen, and Li-Chun Wang. Eqvmp: Energy-efficient and qos-aware virtual machine placement for software defined datacenter networks. In *The International Conference on Information Networking 2014 (ICOIN2014)*, pages 220–225. IEEE, 2014.

[256] Kyungdaw Kang, Ilkyeong Moon, and Hongfeng Wang. A hybrid genetic algorithm with a new packing strategy for the three-dimensional bin packing problem. *Applied Mathematics and Computation*, 219(3):1287–1299, 2012.

[257] Julia A Bennell, Lai Soon Lee, and Chris N Potts. A genetic algorithm for two-dimensional bin packing with due dates. *International Journal of Production Economics*, 145(2):547–560, 2013.

[258] Nancy Jain and Sakshi Choudhary. Overview of virtualization in cloud computing. In *2016 Symposium on Colossal Data Analysis and Networking (CDAN)*, pages 1–4. IEEE, 2016.

[259] Md Hasanul Ferdaus, Manzur Murshed, Rodrigo N Calheiros, and Rajkumar Buyya. Virtual machine consolidation in cloud data centers using aco metaheuristic. In *Euro-Par 2014 Parallel Processing: 20th International Conference, Porto, Portugal, August 25-29, 2014. Proceedings 20*, pages 306–317. Springer, 2014.

[260] Benjamin Speitkamp and Martin Bichler. A mathematical programming approach for server consolidation problems in virtualized data centers. *IEEE Transactions on services computing*, 3(4):266–278, 2010.

[261] Zoltán Ádám Mann. Approximability of virtual machine allocation: much harder than bin packing. 2015.

[262] Pradeep Kumar, Dilbag Singh, and Ankur Kaushik. Power and data aware best fit algorithm for energy saving in cloud computing. *International Journal of Computer Science and Information Technologies*, 5(5):6712–6715, 2014.

[263] Goyal Sudhir, Bawa Seerna, and Singh Bhupinder. Experimental comparison of three scheduling algorithms for energy efficiency in cloud computing [c]. In *2014 IEEE International Conference on Cloud Computing in Emerging Markets, Bangalore,, India*, pages 15–17, 2014.

[264] Saad Mustafa, Kashif Bilal, Sajjad A Madani, Nikos Tziritas, Samee U Khan, and Laurence T Yang. Performance evaluation of energy-aware best fit decreasing algorithms for cloud environments. In *2015 IEEE International Conference on Data Science and Data Intensive Systems*, pages 464–469. IEEE, 2015.

[265] Abdulrahman Alahmadi, Abdulaziz Alnowiser, Michelle M Zhu, Dunren Che, and Parisa Ghodous. Enhanced first-fit decreasing algorithm for energy-aware job scheduling in cloud. In *2014 International Conference on Computational Science and Computational Intelligence*, volume 2, pages 69–74. IEEE, 2014.

[266] SP Usha Kirana and Demian Antony D'Mello. Energy-efficient enhanced particle swarm optimization for virtual machine consolidation in cloud environment. *International Journal of Information Technology*, 13(6):2153–2161, 2021.

[267] Archana Kollu and Sucharita Vadlamudi. Eagle strategy with cauchy mutation particle swarm optimization for energy management in cloud computing. *International Journal of Intelligent Engineering & Systems*, 13(6), 2020.

[268] Poornima Singh Thakur, Pritee Khanna, Tanuja Sheorey, and Aparajita Ojha. Trends in vision-based machine learning techniques for plant disease identification: A systematic review. *Expert Systems with Applications*, 208:118117, 2022.

[269] Maha Bakalla, Hadeel Al-Jami, Heba Kurdi, and Shada Alsalamah. A qos-aware and energy-efficient genetic resource allocation algorithm for cloud data centers. In *2017 9th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, pages 244–249. IEEE, 2017.

[270] Dilip Kumar, Bibhudatta Sahoo, Bhaskar Mondal, and Tarni Mandal. A genetic algorithmic approach for energy efficient task consolidation in cloud computing. *International Journal of Computer Applications*, 118(2):1–6, 2015.

[271] Ola E Elnaggar, Rabie A Ramadan, and Magda B Fayek. Wsn in monitoring oil pipelines using aco and ga. *Procedia Computer Science*, 52:1198–1205, 2015.

[272] Fahimeh Farahnakian, Adnan Ashraf, Pasi Liljeberg, Tapio Pahikkala, Juha Plosila, Ivan Porres, and Hannu Tenhunen. Energy-aware dynamic vm consolidation in cloud data centers using ant colony system. In *2014 IEEE 7th International Conference on Cloud Computing*, pages 104–111. IEEE, 2014.

[273] Xiao-Fang Liu, Zhi-Hui Zhan, Jeremiah D Deng, Yun Li, Tianlong Gu, and Jun Zhang. An energy efficient ant colony system for virtual machine placement in cloud computing. *IEEE transactions on evolutionary computation*, 22(1):113–128, 2016.

[274] Asra Aslam, Ekram Khan, and MM Sufyan Beg. Multi-threading based implementation of ant-colony optimization algorithm for image edge detection. In *2015 Annual IEEE India Conference (INDICON)*, pages 1–6. IEEE, 2015.

[275] Mehdi Rajabzadeh, Abolfazl Toroghi Haghighat, and Amir Masoud Rahmani. New comprehensive model based on virtual clusters and absorbing markov chains for energy-efficient virtual machine management in cloud computing. *The Journal of Supercomputing*, 76:7438–7457, 2020.

[276] Awais Ahmad, Murad Khan, Anand Paul, Sadia Din, M Mazhar Rathore, Gwanggil Jeon, and Gyu Sang Choi. Toward modeling and optimization of features selection in big data based social internet of things. *Future Generation Computer Systems*, 82:715–726, 2018.

[277] Zhibo Cao and Shoubin Dong. Dynamic vm consolidation for energy-aware and sla violation reduction in cloud computing. In *2012 13th International Conference on Parallel and Distributed Computing, Applications and Technologies*, pages 363–369. IEEE, 2012.

[278] Xuan Xiao, Yunni Xia, Feng Zeng, Wanbo Zheng, Xiaoning Sun, Qinglan Peng, Yu Guo, and Xin Luo. A novel coalitional game-theoretic approach for energy-aware dynamic vm consolidation in heterogeneous cloud datacenters. In *Web Services–ICWS 2019: 26th International Conference, Held as Part of the Services Conference Federation, SCF 2019, San Diego, CA, USA, June 25–30, 2019, Proceedings 26*, pages 95–109. Springer, 2019.

[279] Xialin Liu, Junsheng Wu, Lijun Chen, and Lili Zhang. Energy-aware virtual machine consolidation based on evolutionary game theory. *Concurrency and Computation: Practice and Experience*, 34(10):e6830, 2022.

[280] Seyedhamid Mashhadi Moghaddam, Michael O'Sullivan, Cameron Walker, Sareh Fotuhi Piraghaj, and Charles Peter Unsworth. Embedding individualized machine learning prediction models for energy efficient vm consolidation within cloud data centers. *Future Generation Computer Systems*, 106:221–233, 2020.

[281] Anjum Mohd Aslam and Mala Kalra. Using artificial neural network for vm consolidation approach to enhance energy efficiency in green cloud. In *Advances in Data and Information Sciences: Proceedings of ICDIS 2017, Volume 2*, pages 139–154. Springer, 2019.

[282] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and Tanaka Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. In *Parallel Problem Solving from Nature PPSN VI: 6th International Conference Paris, France, September 18–20, 2000 Proceedings 6*, pages 849–858. Springer, 2000.

[283] Aqeel Mahesri and Vibhore Vardhan. Power consumption breakdown on a modern laptop. In *Power-Aware Computer Systems: 4th International Workshop, PACS 2004, Portland, OR, USA, December 5, 2004, Revised Selected Papers 4*, pages 165–180. Springer, 2005.

[284] Jóakim von Kistowski, Hansfried Block, John Beckett, Cloyce Spradling, Klaus-Dieter Lange, and Samuel Kounev. Variations in cpu power consumption. In *Proceedings of the 7th ACM/SPEC on International Conference on Performance Engineering*, pages 147–158, 2016.