DOCTOR OF PHILOSOPHY (PHD)

Optimization Approach for Green Cloud Computing

Al Mahrqi, Ali Abdullah Hamed

Award date:
2024

Awarding institution:
Glasgow Caledonian University

Licence:
CC BY-NC

Link to publication in ResearchOnline
Optimization Approach For Green Cloud Computing

Ali Abdullah Hamed Al Mahruqi (MSc)

Supervisor: Professor Brian Stewart
DoS: Professor Tuleen Boutaleb

A thesis submitted in partial fulfillment of the requirements of Glasgow Caledonian University for the degree of Doctor of Philosophy

School of Engineering and Built Environment

June 2024
ABSTRACT

With the increased use of computers and computing power, implementing cloud computing (CC) has become imperative in the present-day global scenario. While CC techniques give the user community and data centres many advantages, they also provide drawbacks regarding energy consumption and quality of service (QoS). These issues have paved the way for developing a new optimisation approach for green cloud computing (GCC). CC with its extensive infrastructure and a vast array of services, inherently necessitates a large amount of energy to operate effectively. The cumulative energy utilisation in CC can be immense, contributing to operational costs. Since cloud services grow to accommodate increasing data and computing requirements, finding methods to optimise energy consumption becomes vital for ensuring sustainability and cost-effectiveness. Research into energy-efficient practices and technologies is vital for addressing these challenges, aiming to maintain high performance and reliability. The contributions of the thesis mainly focus on optimising the energy efficiency of the cloud without compromising the QoS requirements. This research work is divided into two phases. The initial phase focuses on the analysis of energy models for VM consolidation on the practical servers using various load categories. Based on the observations of the initial phase, a novel hybrid heuristic algorithm-based energy-efficient cloud computing service (HH-ECO) is proposed that offers an effective and energy-efficient solution for optimising scientific workflows under various server load categories.

Understanding the impact of different load categories on energy consumption helps to reduce the energy wastage significantly. Initially, analysis of energy models is performed in CC using a realtime practical server with various load categories. By examining energy models using high-end and standard servers, each hosting multiple VMs for investigating load balancing and energy consumption. By monitoring server behaviour under various loads, key parameters affecting performance and energy usage are identified. The experimental setup comprises VMware ESXi hypervisors handling multiple VMs across one high-end server and three standard servers. Each standard server hosts 7 VMs, whereas the high-end server hosts 17 VMs. The practical testbed is implemented by consolidating VMs on physical servers. By emphasising CPU and memory utilisation as primary factors in energy consumption, the power consumption and performance are measured and verified using the testbed based on these load categories. Results in terms of CPU, memory, and power consumption for each server category are presented. An idle server with all VMs on proved low CPU and memory utilisation and consumed 60-75% of its overall power capacity. Likewise, an underloaded server had maximum utilisation (up to 40%) and slightly more power consumption. A balanced server showed an optimal performance with CPU utilisation of up to 63% and memory utilisation of about 58%, indicating that it could manage additional VMs. An overloaded server showed a high CPU of up to 80%, memory
utilisation of 75% and maximum power consumption, demonstrating the requirement for VM migration to prevent performance degradation.

By categorising server categories, this study aims to develop a novel approach for optimising resource utilisation, reducing power consumption, and improving QoS. Results proved that maintaining a balanced state through strategic VM migrations is an effective method for achieving these objectives. Considering the bottlenecks, cost, and risk associated with the practical environments, the results observed in the practical approach are further investigated by extending the research using the cloudsim tool to utilise the potential advantages such as cost-effectiveness, versatility and risk-free environment for validating and testing the energy efficiency. Thus, the next phase proposes a novel approach that uses a Hybrid Heuristic algorithm-based Energy-efficient Cloud computing service (HH-ECO) and offers an effective solution for resource allocation, task scheduling, and optimisation of scientific workflows. The complexity of scientific workflow increases energy consumption because of the huge computational and data processing demands they impose. Moreover, since scientific workflows often involve large-scale simulation and iterative processing tasks, substantial computational power and memory resources are required. Hence, achieving energy efficiency is significant in scientific workflow within a cloud environment.

A Metaheuristic algorithm called chaotic-based particle swarm optimisation (C-PSO) is used in CC to optimise energy management by dynamically adjusting resource allocation and load balancing. It finds near-optimal solutions for efficient energy utilisation by generating global best plans without local convergence. C-PSO with adaptive mutation prevents the decline of global optima by recognising the best host for VM placement and ensuring an effective resource allocation approach. Considering the workflow task precedence relationships during C-PSO-based task scheduling, the novel hybrid heuristic technique efficiently solves the multi-objective combinatorial optimisation problem without dominance among the workflow tasks. HH-ECO focuses on executing non-dominant workflow tasks through adaptive mutation and an energy-aware migration strategy. C-PSO with adaptive mutation avoids the deterioration of global optima while finding the best host to place the virtual machine and ensures an appropriate resource allocation plan. During the execution of workflow tasks, the load category-based resource migration maintains the estimated QoS even in a dynamic environment. A Cloudsim-based simulation study delivers superior results compared to the existing methods, such as the Hybrid heuristic workflow scheduling algorithm (HHWS) and Distributed dynamic vm management (DDVM). The proposed approach significantly improves the optimal makespan and reduces energy consumption when compared to the existing methods. It meets the criteria for minimum energy consumption and provides an improved QoS to cloud data centres.
Table of Contents

CHAPTER 1 INTRODUCTION .................................................................................................................................................... 1
  1.1 CLOUD COMPUTING .................................................................................................................................................... 1
    1.1.1 HISTORY OF CLOUD COMPUTING .................................................................................................................. 2
  1.2 DEPLOYMENT MODELS: .................................................................................................................................................. 4
  1.3 BENEFITS OF CLOUD COMPUTING: ......................................................................................................................... 6
  1.4 CLOUD COMPUTING ARCHITECTURE: ..................................................................................................................... 6
  1.5 MOTIVATION ................................................................................................................................................................. 8
  1.6 GREEN CLOUD COMPUTING ........................................................................................................................................ 10
    1.6.1 SERVER CONSOLIDATION .................................................................................................................................. 10
    1.6.2 PLACEMENT OF VIRTUAL MACHINES ............................................................................................................ 11
    1.6.3 VM MIGRATION ...................................................................................................................................................... 12
    1.6.4 LOAD BALANCING ............................................................................................................................................... 12
  1.7 METAHEURISTIC OPTIMIZATION FOR ENERGY EFFICIENCY IN CLOUD ............................................................. 13
  1.8 ENERGY EFFICIENCY IN SCIENTIFIC WORKFLOW ................................................................................................ 13
  1.9 PROBLEM THEME .......................................................................................................................................................... 14
  1.10 AIM AND OBJECTIVES ................................................................................................................................................. 16
  1.11 THESIS OUTLINE .......................................................................................................................................................... 17
  1.12 SUMMARY .................................................................................................................................................................... 18

CHAPTER 2 LITERATURE REVIEW ........................................................................................................................................ 19
  2.1 BACKGROUND ............................................................................................................................................................... 19
  2.2 TAXONOMY DESIGN ....................................................................................................................................................... 20
  2.3 STATE-OF-THE-ART .......................................................................................................................................................... 23
    2.3.1 Workflow Task Scheduling Approaches ........................................................................................................... 23
    2.3.2 Resource Allocation ............................................................................................................................................... 25
    2.3.3 VM Migration Approaches ................................................................................................................................ 29
    2.3.4 VM Migration with Server Consolidation ......................................................................................................... 33
    2.3.5 Service Level Agreement (SLA)-Based Approaches .......................................................................................... 39
  2.4 GAPS IDENTIFIED .......................................................................................................................................................... 40
  2.5 SUMMARY ....................................................................................................................................................................... 42

CHAPTER 3 METHODOLOGY .................................................................................................................................................... 43
  3.1 INTRODUCTION ............................................................................................................................................................... 43
  3.2 PROBLEM FORMULATION ............................................................................................................................................. 43
  3.3 PROPOSED METHODOLOGY ....................................................................................................................................... 44
  3.4 PHASE 1: ANALYSIS OF ENERGY MODELS USING VM CONSOLIDATION ON PRACTICAL SERVER WITH VARIOUS LOAD CATEGORIES ............................................................................................................. 45
    3.4.1 VM CONSOLIDATION .............................................................................................................................................. 45
    3.4.2 CONFIGURATION OF CONSOLIDATED SERVER .................................................................................................. 46
    3.4.3 ANALYSIS OF SERVER LOAD CATEGORIES ......................................................................................................... 46
  3.5 PHASE 2: HYBRID HEURISTICS-BASED ENERGY EFFICIENT OPTIMIZATION OF SCIENTIFIC WORKFLOW IN CLOUD ............................................................................................................................................. 46
    3.5.1 VM ALLOCATION ................................................................................................................................................... 46
    3.5.2 TASK SCHEDULING .............................................................................................................................................. 47
    3.5.3 VM MIGRATION ....................................................................................................................................................... 48
  3.6 SUMMARY ....................................................................................................................................................................... 48

CHAPTER 4 ANALYSIS OF ENERGY MODELS USING VM CONSOLIDATION ON PRACTICAL SERVERS WITH VARIOUS LOAD CATEGORIES ............................................................................................................. 49
  4.1 INTRODUCTION ............................................................................................................................................................... 49
    4.1.1 VIRTUALISATION USING ESXI AND V-SPHERE ................................................................................................. 49
    4.1.2 SYSTEM MODEL ................................................................................................................................................... 50
CHAPTER 5 HYBRID HEURISTICS-BASED ENERGY-EFFICIENT OPTIMIZATION OF SCIENTIFIC WORKFLOW IN CLOUD

5.1 OUTLINE OF HH-ECO APPROACH
5.1.1 OBJECTIVES OF HH-ECO APPROACH
5.1.2 CONTRIBUTIONS
5.1.3 PROCESS INVOLVED

5.2 PRELIMINARIES
5.2.1 SCIENTIFIC WORKFLOW
5.2.2 DOMINANCE PROBLEM
5.2.3 METAHEURISTIC ALGORITHMS
  5.2.3.1 PSO
  5.2.3.2 MODELLING PSO IN CLOUD
  5.2.3.3 C-PSO
  5.2.3.4 HYBRID-HEURISTIC ALGORITHM
  5.2.3.5 ADAPTIVE MUTATION

5.3 SYSTEM MODEL
  5.3.1 PSO-based Workflow Scheduling
  5.3.2 C-PSO-based Workflow Scheduling

5.4 PROBLEM FORMULATION
  5.4.1 Resource Allocation and Task Scheduling
  5.4.2 VM Migration

5.5 THE HH-ECO APPROACH

5.6 VM ALLOCATION WITH ADAPTIVE MUTATION

5.7 TASK SCHEDULING WITH ADAPTIVE MUTATION

5.8 VM MIGRATION WITH ADAPTIVE MUTATION

5.9 OVERALL ALGORITHM

5.10 SUMMARY
CHAPTER 6  EXPERIMENTAL EVALUATION

6.1 OVERVIEW OF CLOUDSIM ................................................................. 112
6.2 SCIENTIFIC WORKFLOWS ............................................................... 113
6.3 MODELLING THE CLOUD ENVIRONMENT ........................................... 114
6.4 EXPERIMENTAL SETUP ..................................................................... 116
6.4.1 PROCESS OF SIMULATION ............................................................. 118
6.5 EVALUATION METRICS ..................................................................... 118
6.6 EXPERIMENTAL RESULTS ................................................................. 120
    6.6.1 COMPARATIVE STUDY OF MAKESPAN, AVERAGE ENERGY CONSUMPTION, SLA VIOLATION, AND WAITING TIME RATIO ................................................................. 121
    6.6.2 IMPACT OF AN AVERAGE NUMBER OF ACTIVE HOSTS ......................... 126
    6.6.3 IMPACT OF THE PERCENTAGE OF RESOURCE UTILIZATION .................. 129
    6.6.4 IMPACT OF SERVER BEHAVIOR ................................................... 135
    6.6.5 COST-BENEFIT ANALYSIS ............................................................. 137
6.7 SUMMARY ......................................................................................... 137

CHAPTER 7  CONCLUSIONS ......................................................................... 139

CHAPTER 8  FUTURE WORK ......................................................................... 142

LIST OF REFERENCES ................................................................................ 143
## List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACO</td>
<td>Ant-Colony Optimization</td>
</tr>
<tr>
<td>AD</td>
<td>Active Directory</td>
</tr>
<tr>
<td>ARPANET</td>
<td>Advanced Research Projects Agency Network</td>
</tr>
<tr>
<td>CC</td>
<td>Cloud Computing</td>
</tr>
<tr>
<td>CCA</td>
<td>Cloud Computing Architecture</td>
</tr>
<tr>
<td>CCC</td>
<td>Cloud Computing Client</td>
</tr>
<tr>
<td>CCF</td>
<td>Cloud Computing Framework</td>
</tr>
<tr>
<td>CEAR</td>
<td>Carbon Emission Agency Regulator</td>
</tr>
<tr>
<td>CEAS</td>
<td>Cost and Energy-Aware Scheduling</td>
</tr>
<tr>
<td>CLPSO</td>
<td>Comprehensive Learning Particle Swarm Optimisation</td>
</tr>
<tr>
<td>C-PSO</td>
<td>Chaotic-based Particle Swarm Optimisation</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>CSP</td>
<td>Cloud Service Provider</td>
</tr>
<tr>
<td>CU</td>
<td>Cloud User</td>
</tr>
<tr>
<td>DAG</td>
<td>Directed Acyclic Graph</td>
</tr>
<tr>
<td>DARPA</td>
<td>Defense Advanced Research Projects Agency</td>
</tr>
<tr>
<td>DBMS</td>
<td>Database Management Software</td>
</tr>
<tr>
<td>DD</td>
<td>Degree of Disparity</td>
</tr>
<tr>
<td>DDVM</td>
<td>Distributed Dynamic VM Management</td>
</tr>
<tr>
<td>DHCP</td>
<td>Dynamic Host Configuration Protocol</td>
</tr>
<tr>
<td>DNS</td>
<td>Domain Name System</td>
</tr>
<tr>
<td>DVFS</td>
<td>Dynamic Voltage Frequency Scaling</td>
</tr>
<tr>
<td>EATTO</td>
<td>Energy-Aware, Time, and Throughput Optimisation</td>
</tr>
<tr>
<td>EEM</td>
<td>Energy-Efficient Hybrid</td>
</tr>
<tr>
<td>EMM</td>
<td>Enhanced Maximum-Minimum Algorithm</td>
</tr>
<tr>
<td>EMO</td>
<td>Evolutionary Multi-objective Optimization</td>
</tr>
<tr>
<td>EPP</td>
<td>Energy Power Provider</td>
</tr>
<tr>
<td>GA</td>
<td>Genetic Algorithm</td>
</tr>
<tr>
<td>GCC</td>
<td>Green Cloud Computing</td>
</tr>
<tr>
<td>GSA</td>
<td>Gravitation Search Algorithm</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphic User Interface</td>
</tr>
<tr>
<td>GWES</td>
<td>General Workflow Execution Service</td>
</tr>
<tr>
<td>HDPSO</td>
<td>Hybrid Discrete Particle Swarm Optimisation</td>
</tr>
<tr>
<td>HEFT</td>
<td>Heterogeneous Earliest Finish Time</td>
</tr>
<tr>
<td>HH-ECO</td>
<td>Hybrid Heuristic algorithm based Energy-efficient cloud Computing</td>
</tr>
<tr>
<td>HHWS</td>
<td>Hybrid Heuristic Workflow Scheduling</td>
</tr>
<tr>
<td>HRM</td>
<td>Human Resource Management</td>
</tr>
<tr>
<td>I_MaOPSO</td>
<td>Improved Many Objective Particle Swarm Optimisation</td>
</tr>
<tr>
<td>IaaS</td>
<td>Infrastructure as a Service</td>
</tr>
<tr>
<td>ICT</td>
<td>Information and Communication Technology</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>MBFD</td>
<td>Modified Best Fit Decreasing</td>
</tr>
<tr>
<td>Acronym</td>
<td>Full Form</td>
</tr>
<tr>
<td>---------</td>
<td>-----------</td>
</tr>
<tr>
<td>MDP</td>
<td>Markov Decision Process</td>
</tr>
<tr>
<td>MIPS</td>
<td>Millions of Instructions Per Second</td>
</tr>
<tr>
<td>MIS</td>
<td>Management Information System</td>
</tr>
<tr>
<td>MM</td>
<td>Min-Min</td>
</tr>
<tr>
<td>MOGA</td>
<td>Multi-Objective Genetic Algorithm</td>
</tr>
<tr>
<td>MR</td>
<td>Memory Reuse</td>
</tr>
<tr>
<td>MS</td>
<td>Makespan</td>
</tr>
<tr>
<td>NASA</td>
<td>National Aeronautics and Space Administration</td>
</tr>
<tr>
<td>NIC</td>
<td>Network Interface Card</td>
</tr>
<tr>
<td>NUR</td>
<td>None-Underutilised Rack</td>
</tr>
<tr>
<td>OMUR</td>
<td>Only Migrate Under-utilised Racks</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>PaaS</td>
<td>Platform as a Service</td>
</tr>
<tr>
<td>PM</td>
<td>Physical Machine</td>
</tr>
<tr>
<td>PMBL</td>
<td>Physical Machine Balanced</td>
</tr>
<tr>
<td>PMIS</td>
<td>Physical Machine Idle Server</td>
</tr>
<tr>
<td>PMOL</td>
<td>Physical Machine OverLoaded</td>
</tr>
<tr>
<td>PMUL</td>
<td>Physical Machine UnderLoaded</td>
</tr>
<tr>
<td>PSO</td>
<td>Particle Swarm Optimisation</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
</tr>
<tr>
<td>RBR</td>
<td>Rack by Rack</td>
</tr>
<tr>
<td>REEWS</td>
<td>Reliability and Energy Efficient Workflow Scheduling</td>
</tr>
<tr>
<td>SaaS</td>
<td>Software as a Service</td>
</tr>
<tr>
<td>S-PSO</td>
<td>Set-Based Particle Swarm Optimisation</td>
</tr>
<tr>
<td>SLA</td>
<td>Service Level Agreement</td>
</tr>
<tr>
<td>VM</td>
<td>Virtual Machine</td>
</tr>
<tr>
<td>VMC</td>
<td>Virtual Machine Consolidation</td>
</tr>
<tr>
<td>WTR</td>
<td>Waiting Time Ratio</td>
</tr>
</tbody>
</table>
List of Tables

TABLE 1-1 THESIS OUTLINE ................................................................................................................. 17

TABLE 4-1 SETUP REQUIREMENT ........................................................................................................... 55
TABLE 4-2 NORMAL SERVER (3 SERVERS) ............................................................................................. 55
TABLE 4-3 HIGHEND SERVER SETUP CRITERIA AND SERVER BEHAVIOUR OBSERVATION .......... 58
TABLE 4-4 NORMAL SERVERS (THREE SERVERS) SETUP CRITERIA AND SERVER BEHAVIOUR OBSERVATION ......................................................................................................................... 61
TABLE 4-5 LOAD CATEGORY AND TEST ............................................................................................. 66
TABLE 4-6 CATEGORISATION OF SERVER BEHAVIOR ........................................................................ 77

TABLE 5-1 COMPARISON OF THE STANDALONE TASK, WORKFLOW TASK, AND SCIENTIFIC WORKFLOW ............................................................................................................................. 85
TABLE 5-2 LOAD CATEGORIES .............................................................................................................. 94
TABLE 5-3 TIME COMPLEXITY OF PARTICLES OVER SOLUTION SPACE IN PSO ......................... 96
TABLE 5-4 CPU PERFORMANCE AND MEMORY OF VMS AND PMS - EXAMPLE SCENARIO .......... 102
TABLE 5-5 VM ALLOCATION - EXAMPLE SCENARIO ............................................................................ 102

TABLE 6-1 SPECIFICATIONS OF HOST AND VMS .................................................................................. 117
TABLE 6-2 PERFORMANCE COMPARISON OF MAKESPAN, AVERAGE ENERGY CONSUMPTION, SLA VIOLATION, AND WAITING TIME RATIO ...................................................................................... 121
TABLE 6-3 PERFORMANCE COMPARISON OF TOTAL COST, AVERAGE ENERGY CONSUMPTION, AND DEGREE OF DISPARITY .................................................................................................. 126
TABLE 6-4 PERFORMANCE COMPARISON OF TOTAL COST, AVERAGE ENERGY CONSUMPTION, AND DEGREE OF DISPARITY .................................................................................................. 129
TABLE 6-5 AVERAGE ENERGY CONSUMPTION AND AVERAGE CPU UTILIZATION IN TERMS OF SERVER BEHAVIOR .............................................................................................................................. 135
List of Figures

FIGURE 1-1 CLOUD DEPLOYMENT & SERVICE MODEL ................................................................. 5
FIGURE 1-2 CC FRONTEND SYSTEM [19] .................................................................................. 7
FIGURE 1-3 CC BACKEND SYSTEM ........................................................................................... 8
FIGURE 1-4 ENERGY CONSUMPTION IN DATA CENTRE [24] ...................................................... 9

FIGURE 2-1 TAXONOMY DIAGRAM .......................................................................................... 22
FIGURE 2-2 MIGRATING WITHOUT MEMORY REUSE (MR) ADAPTED FROM [57] .................... 36
FIGURE 2-3 MIGRATING WITH MEMORY REUSE (MR) ADAPTED FROM [57] ............................ 36

FIGURE 3-1 OVERALL BLOCK DIAGRAM .................................................................................... 45

FIGURE 4-1 TYPE 1 HYPERVISOR (BARE METAL) .................................................................. 51
FIGURE 4-2 TYPE 2 HYPERVISOR (HOSTED) ............................................................................. 52
FIGURE 4-3 TESTBED ARCHITECTURE (ESXI 5.5, VSPHERE CLIENT AND V-CENTER) FOR 4 ESXI HOSTS. ADAPTED FROM [151] ................................................................. 56
FIGURE 4-4 CLOUD TEST BED SETUP ON HIGH-END SERVER. ADAPTED FROM [161] ............ 57
FIGURE 4-5 CLOUD SETUP ON A NORMAL SERVER. ADAPTED FROM [161] ............................. 61
FIGURE 4-6 IDLE CPU UTILISATION WITH 17 VMS ON ......................................................... 67
FIGURE 4-7 IDLE MEMORY UTILISATION WITH 17 VMS ON .................................................. 68
FIGURE 4-8 IDLE POWER CONSUMPTION WITH 17 VMS ON ............................................... 69
FIGURE 4-9 UNDER LOADED CPU UTILISATION ..................................................................... 70
FIGURE 4-10 UNDER LOADED MEMORY UTILISATION ............................................................. 70
FIGURE 4-11 UNDER LOADED POWER CONSUMPTION ............................................................ 71
FIGURE 4-12 BALANCED CPU UTILISATION ............................................................................. 72
FIGURE 4-13 BALANCED MEMORY UTILIZATION ................................................................. 73
FIGURE 4-14 BALANCE POWER CONSUMPTION ..................................................................... 74
FIGURE 4-15 OVERLOADED CPU UTILISATION ...................................................................... 75
FIGURE 4-16 OVERLOADED MEMORY UTILIZATION ............................................................... 76
FIGURE 4-17 OVERLOADED POWER UTILISATION ................................................................. 77
FIGURE 4-18 ALGORITHM FLOWCHART .................................................................................... 78

FIGURE 5-1 EXAMPLE WORKFLOW OF TASKS .......................................................................... 93
FIGURE 5-2 HYBRID HEURISTIC-BASED ENERGY OPTIMIZED TASK SCHEDULING AND VM MIGRATION OF SCIENTIFIC WORKFLOW ...................................................... 99
FIGURE 5-3 C-PSO TASK SCHEDULING PLAN ........................................................................... 106
FIGURE 5-4 FLOWCHART OF HH-ECO METHODOLOGY ............................................................ 109

FIGURE 6-1 MAKESPAN VS. NO. OF WORKFLOW TASKS .......................................................... 122
FIGURE 6-2 AVERAGE ENERGY CONSUMPTION VS. NO. OF WORKFLOW TASKS .................. 123
FIGURE 6-3 SLA VIOLATIONS VS. NO. OF WORKFLOW TASKS ............................................... 124
FIGURE 6-4 WAITING TIME RATIO (WTR) VS. NO. OF WORKFLOW TASKS ............................... 125
FIGURE 6-5 TOTAL COST VS. NO. OF HOSTS ............................................................................ 127
FIGURE 6-6 DEGREE OF DISPARITY VS. NO. OF HOSTS ............................................................ 128
FIGURE 6-7 AVERAGE ENERGY CONSUMPTION VS. NO. OF HOSTS ....................................... 129
FIGURE 6-8 MAKESPAN VS. PERCENTAGE OF RESOURCE UTILIZATION ................................. 131
FIGURE 6-9 DEGREE OF DISPARITY (DD) VS. PERCENTAGE OF RESOURCE UTILIZATION .......... 132
FIGURE 6-10 MIGRATION EFFICIENCY VS. PERCENTAGE OF RESOURCE UTILIZATION .......... 133
FIGURE 6-11 SLA VIOLATIONS VS. PERCENTAGE OF RESOURCE UTILIZATION ......................... 134
FIGURE 6-12 AVERAGE CPU UTILIZATION VS. CATEGORIZATION OF SERVER BEHAVIOR .......... 135
FIGURE 6-13 AVERAGE ENERGY CONSUMPTION CATEGORIZATION OF SERVER BEHAVIOR ....... 136
ACKNOWLEDGMENT

First and foremost, I thank God Almighty for giving me health, patience, and blessing to complete this thesis. Though, I have gone through many challenges of hectic work, family, and health issues that arose while doing this research. With Almighty’s blessings, I overcame all challenges and got motivated to cope and finish my research and thesis.

I am incredibly grateful to my supervisor, Prof. Brian Stewart, for his irreplaceable advice, unlimited support, and extreme patience during this research. Again, I am very thankful for his understanding of my unfulfilled promises of submitting my thesis after all the motivation and support given multiple times. His immense experience and outstanding knowledge have enabled and encouraged me to publish multiple conference and journal papers on reputable publishers. His valuable comments have contributed to better work and pushed me to imitate the style of comments which I have continuously used with my students in their final year projects and MSc thesis.

The other appreciation goes to Professor Gordon Morison, who acted as Director of Studies (DoS) during my research. His valuable comments contributed to the success of some of the publications. His motivations have also encouraged me to push myself to finalize the remaining part of this thesis. I need to remember Professor Tuleen, who accepted to be my new DoS after the departure of Professor Gordon from Glasgow Caledonian University, Scotland, United Kingdom. I am grateful for her acceptance, assistance in viva rehearsal and motivation she has been providing.

I am also very grateful to my ex-colleague and friend, Dr. Sarma Mallela, who helped me elaborate on the comments given by my supervisor. I would turn to him if I needed a second opinion on the comments. My other appreciation goes to Professor Sudhir (Assistant Dean, Undergraduate), for assisting me in learning the error bars in excel, Dr Viajaya, Dr Ram & Ms Namitha (Colleagues) for checking the formats of the thesis. My other appreciation goes to my Dean (Professor Ahmed Al Bulushi) for his understanding and support during my research.

I also acknowledge Glasgow Caledonian University for supporting my Ph.D. work and waiving my fees during my studies. This resulted from our long-time collaboration with our College of Engineering, Caledonian, National University. I am also grateful for the extra time provided to me even after my delay.

Last but not least, I would like to thank my mother for her extreme support, prayer, and love provided to me while doing my research. Without forgetting my beloved wife and children for their understanding, patience, and support, it would be impossible for me to complete my study without their company.

(Ali Abdullah Hamed Al Mahruqi)
DECLARATION

This to confirm that this project thesis is my own original work and has not been submitted anywhere for any other fulfillment of the requirement or for any awards.

This thesis is available for everyone on the understanding that it is copyright material and that no quotation from this thesis may be published without proper acknowledgement.

I certify that all material in this project report, which is not my own work, has been identified and that no material has previously been submitted and approved for the award of a degree by this or any other University

Student Name: Ali Abdullah Hamed Al Mahruqi
Student ID of GCU: S1236594
| 02 | Ali Abdullah Hamed Al-Mahruqi, Brain Stewart and Brian Hainey (2015), A review of green cloud computing techniques and algorithms. 2nd International Conference on next generation of computing and communication technology, April 22-23, 2015, Dubai UAE. |
## PRESENTATION

<table>
<thead>
<tr>
<th>No.</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>PhD Related Cloud Computing Presentation in Caledonian College of Engineering Foundation students (October 2015)</td>
</tr>
<tr>
<td>03</td>
<td>PhD Related Cloud Computing Theory &amp; Virtualization Workshop in Caledonian College of Engineering (October 2016)</td>
</tr>
</tbody>
</table>

## AWARDS

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>02</td>
<td>Ali Abdullah Hamed Al-Mahruqi, The First best Presentation Award (2015), a review of green cloud computing techniques and algorithms. 2nd International Conference on next generation of computing and communication technology, April 22-23, 2015, Dubai UAE.</td>
</tr>
</tbody>
</table>

CHAPTER 1 INTRODUCTION

1.1 Cloud Computing

Cloud computing (CC) is a powerful technology that has attracted many Information and communication technology (ICT) companies to manoeuvre their business faster and at a lower cost. As stated by [1], the term "Cloud" originally comes from the concept of the world of telecommunication technology with the use of Virtual private networks (VPNs) where different organisations or employees can access resources from separate locations. This network is very similar to CC, which is defined as the delivery of computing services, servers, storage, databases, networking, software, analytics, and the Internet, for which it may not be necessary for the end-user to know the physical location of where the data is sourced [2]. CC is a combination of hardware and software accessed over the Internet from a different site with minimal effort from the end-users [3]-[6]. CC allows various organisations from other parts of the world to share and access an unlimited pool of resources on-demand, which provides the flexibility of pay-as-you-go. With this new IT trend, organisations and users are not required to set up a physical infrastructure to access their computing facilities. All computing facilities will be accessed on-demand over the Internet using the resources set up by a third party known as Cloud service providers (CSPs). CSPs are gargantuan syndicates that host massive cloud facilities and services for cloud consumers. CSPs operate the same way as electricity, water, and telephone service providers. They provide cloud consumers with the service they require, and payment is made depending on the usage.

Some commercial CC service providers are world-leading data centres such as Amazon Web Service, IBM, Microsoft (Azure), Oracle, Google, Verizon, and others. With the advent of increased users globally, the list of companies increases annually, thus providing more and more flexibility at the user end over time. Data centres are the backbone that handles all IT resources and are used by multiple CSPs to distribute and manage services to cloud consumers. They act as centre repositories that hoard and propagate bulky data to cloud clients. Data centres consist of servers, storage, and network facilities such as switches, routers, cables, and racks to interconnect all IT facilities [7]. Cloud data centres comprise sufficient infrastructure such as power systems to manage the distribution of power to the infrastructure, cooling systems to
cool equipment, ventilators to outlet polluted air and inlet fresh air, backup
generators to automatically provide power to the infrastructure in case of a power
failure and comprehensive firewalls and other security measures to protect from
intruders.

1.1.1 History of Cloud Computing
A summary of CC is provided below, focusing on crucial periods.

The concept of shared computer use by two or more people simultaneously was
established in 1963 as a precursor to what is today known as CC. This project was
undertaken by DARPA (Defense Advanced Research Projects Agency, USA). This
project was termed a primitive Cloud with two or three people accessing gigantic,
archaic computers using reels of magnetic tape memory, leading to virtualisation,
which is a process of creating more than one image of a computer while sharing the
same resources of the same hardware. It acts like a real computer with a complete
capability operating system (OS). John Mc Cathy, an American computer scientist and
one of the core founders of artificial intelligence, envisioned computers becoming a
public utility in the future [1]. In 1966, Parkhill wrote a book which explored the
characteristics of CC, "The Challenge of computer utility" [1]. In 1969, Licklider, a man
behind the project, helped to develop the ARPANET (Advanced Research Projects
Agency Network), a primitive version of the Intergalactic Computer Network (the
Internet), which has helped the birth and continuation of CC. The concept of
Virtualization evolved in the 1970s.

As ideas progressed with the Internet, many businesses began offering VPNs as a
service to be rented. From the 1970s to the 1990s, many advancements in the
technology required for trustworthy CC occurred. Computer giant IBM, for example,
released an OS in 1972 called the VM (Virtual Machine) OS. A VM is software or an
emulator of a computer system, typically installed on computer hardware and
allowing the user to set up multiple OSs on the same computer that can be used
simultaneously. VM shares the number of resources available on a particular PM
(Physical Machine), and once installed, VM acts like a real computer and provides the
same functionality as a standalone OS. Virtualisation technology emerged and became
popular in the 1990s, prompting organisations to develop modern cloud infrastructure
[8].

The 1990s saw several telecommunications companies offer their versions of
virtualised VPNs. CC was already a vibrant and growing asset for companies,
educational facilities, and many others by 1996. In 1997, Chellappa from the University
of Emory came up with the new definition of CC as the new computing paradigm that
will be utilised for economic purposes rather than technical alone [8]. This computing
has become a reality because CC has attracted many organisations and contributed to
faster business activities and a booming economy.
As the years passed, CC became popular among many companies, and different companies began to research its benefits and possible usage. In 1999, Salesforce was one of the pioneers in delivering successful CC services to end users. They made use of the Internet to provide software programs to end users. This process attracted many with internet access to download or access software applications without a physical purchase.

In 2002, Amazon moved into the cloud and introduced online retail services where customers could purchase products. Cloud infrastructure enables Amazon to use its computer facilities more efficiently. This service encouraged and motivated the research and pushed further into the cloud.

In 2006, a new cloud web service was initiated by Amazon to provide comprehensive online services to clients and other websites [8]-[9]. Amazon provided two web services, "Amazon Mechanical Turk and Elastic Compute Cloud (EC2)". Amazon Mechanical Turk offers cloud services like storage computation and human intelligence. In contrast, Elastic Compute Cloud allowed clients to rent virtual computers and install and use their software programs.

In the same year, Google introduced the concept of Google Docs, which multiple clients could use on the Internet. Google Docs had two different products (Google Spreadsheet and Google Writely). The documents allowed users to write, develop, edit, and update spreadsheets and share them between teams online [8].

As a result of the CC expansion, in 2007, IBM, Google and numerous universities (University of Washington, Carnegie Mellon University, MIT, Stanford University, University of Maryland and the University of California at Berkeley) initiated the development of research projects that were dedicated to developing a server farm with fast processors which could handle large datasets. All the universities were motivated as they knew CC would improve work efficiency with less cost. In the same year, Netflix also came up with streaming online video by using the concept of cloud, allowing users to watch videos online.

In 2008, NASA’s (National Aeronautics and Space Administration, USA) Open Nebula released the first open-source software that could be used for deploying Private and hybrid deployment models. Private, installed exclusively by an individual organisation or leased by service providers dedicated to a specific organisation, and Hybrid is a combination of public and private clouds which provide flexibility for organisations to use both (private and public). The private cloud is used for sensitive data, and the public for non-sensitive data. These deployment models opened up innovative technologies for the effectiveness of important businesses. After 2008, many players realised the importance of CC and decided to move from traditional computing and start their businesses in a cloud environment [8][9].

From 2011 onwards, cloud concepts and their utilisation started to boom and attracted many businesses [10]. In 2017, most of these CSPs provided all services, such as software, platforms and infrastructures, to their clients. After that, many...
organisations entered the CC business, with some becoming CSPs and some migrating from traditional computing to CC [10].

1.2 Deployment Models:
All CSPs around the world have different types of deployment models which are available for cloud users. They provide flexibility of choices to users when joining CC. The four deployment models which are commonly used by CSPs are as follows [11] (Figure 1-1):

- **Private cloud**: This model can be deployed exclusively by an individual organisation or leased by service providers dedicated to a particular organisation. It is expensive to deploy as infrastructures and services are utilised by a single organisation.
- **Public cloud**: This is a type of deployment model which is shared by multiple organisations. It always consists of mega-scale infrastructure capable of providing services to various companies. It is cheap in terms of cost, as numerous companies share infrastructures and services.
- **Hybrid cloud**: This deployment model comprises multiple deployments (private, community and public). For example, an organisation can choose to lease a public deployment from a service provider while maintaining in-house infrastructures (private) or lease from a service provider.
- **Community cloud**: This allows the sharing of infrastructures for specific organisations or communities. For example, the Ministry of Health can deploy a community cloud for all ministries of health and hospitals around the country. It can also be managed solely by an organisation or leased by service providers.
All of the deployment models (private, public, hybrid and community) offer the following services: software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS) (Figure 1-1). SaaS enables cloud clients to access software applications over the Internet through CSPs without installing and maintaining the applications. There are many software applications which CSPs offer under SaaS, and some of them are office software such as Microsoft software packages, message software, Human resource management (HRM) software, Management Information System (MIS) Software, accounting software, payroll software, service desk management software, database management software (DBMS) etc. They can be accessed anytime, anywhere, with a simple connection to the Internet. PaaS provides the platform with an opportunity for the clients, enabling them to develop, execute and manage the database without having in-house facilities. Cloud users are provided with multiple platforms to work on their programmes based on the needs of the organisations. IaaS delivers resources such as PMs with VMs and network devices. Clients do not need to purchase or set up any infrastructure [11]-[14] on their premises. CSPs provide all necessary hardware, including storage to store the new unlimited quantity of data, firewalls that protect clients from unauthorised access and network connections to provide connectivity from CSPs to cloud consumers [4].

Though CC may appear to be a new trend in ICT, the concept has been practised on a small scale since the Internet was introduced. The idea is the same as email servers, search engines, YouTube, and so on. Users can sit in front of their computers and access unlimited data without having infrastructure set up in-house. An excellent example of the cloud concept is the Google search engine, where users are only
required to have their computers and perform a search without requiring extra resources for the retrieved data. Google hosts all server facilities that provide all the necessary data consumers need. A user only needs an internet connection and email account to get hooked up and use email facilities anywhere worldwide without hosting the information provided.

1.3 Benefits of cloud computing:
Many benefits encourage many businesses to join the world of the cloud. Some of them are as follows [15].

I. Cost saving: This is the primary benefit that attracts many organisations worldwide. Businesses can save tremendous money when buying, managing and maintaining IT facilities. Establishments are no longer required to purchase expensive facilities; instead, they will use the facilities provided by their service providers. Organisations are no longer bothered to plan for system upgrades. Human resource costs are also reduced as service providers take care of significant maintenance. Additionally, energy consumption is reduced as the organisation does not host server systems at their premises, reducing costs.

II. Scaling up and down: Based on needs, organisations can scale up and down (i.e., increase or decrease computational capacity) at any time without considering facilities. This flexibility helps to free organisation time and concentrate on the company’s productivity.

III. Business steadiness: Since all facilities, maintenance and data security are taken care of by the CSP, the organisation will not worry about system failures, power failures, natural disasters or other crunches.

IV. Efficacy and better collaboration: CC allows the organisation with multiple branches or partners to work more effectively as duplicate files and other facilities can be shared by various employees, facilitating focus on the common goals.

V. Work flexibility: CC overcomes the problem of working only at the office; work can be done quickly at home, on a journey or anywhere without being available physically at the workplace.

VI. Involuntary updates: Organisations should not worry about any updates associated with costs when it comes to processing power, software and other facilities.

1.4 Cloud Computing Architecture:
A Cloud computing architecture (CCA) is a system that consists of different components and subcomponents that allow both cloud users and CSPs to interconnect with each other via the Internet [1][16][17]. The architecture has been designed to accommodate two parties (cloud clients/users and cloud providers) and is divided into two parts: (i) frontend system and (ii) backend system. The frontend system refers to the clients or users who access cloud facilities and consists of
interfaces and applications that act as a gateway to cloud facilities (refer to Figure 1-2). The frontend system needs only desktop computers, tablets or mobile phones to connect to the cloud. Use of the facility provided by CSPs is undertaken per the Service Level Agreement (SLA), where the SLA is a contract between the CSPs and cloud users based on the service the consumers need. It contains all the rules and regulations that will protect both parties. SLA provides the list of services and level of standard that CSPs will deliver to cloud clients. It provides guidelines that benefit CSPs and cloud clients regarding performance and Quality of Service (QoS). For example, if the performance is degraded and QoS is not up to standard, the CSPs should pay the penalty to the cloud clients.

Moreover, if the cloud clients are misusing the services beyond the agreed terms, then cloud clients will also need to pay the fine to the CSPs. An SLA contract agreement is negotiated and agreed upon by both parties. It becomes a legal document once it is signed by both parties [18]. The backend system is the cloud itself and cannot be seen by cloud users. It consists of all cloud facilities, such as PMs, VMs, storage, and security mechanisms, to protect cloud facilities from intruders and a general cloud management system that manages the entire services of CC (refer to Error! Reference source not found.).
1.5 Motivation
The recent surge in CC platform adoption has significant implications for energy efficiency. Notably, several CSPs worldwide offer different services to cloud consumers like Amazon Web Services, Microsoft Azure, Google Cloud Platform, and Oracle Cloud. These CSPs invest heavily in building and maintaining energy-efficient data centres and potentially benefit from economies of scale. This expansion proves that CC shows no sign of slowing down. It continues to grow with a high percentage of participants. In 2018, there were around 95 data centres in Asia and the Pacific region and 79 in the United States and Canada. These two regions are considered to have 72% of the world's data centres, with 24% hosted by Europe and 4% in Latin America. The size of the public computing market worldwide has climbed from $5.8 billion in 2008 to $141.43 billion in 2018 and reached $151.71 billion in 2019, $159.28 billion in 2020 and over $278 billion in 2021 [20].

The CC market was valued at $545.8 billion in 2022, and it is projected to reach $1.240.9 billion by the end of the year 2027 [21]. By 2026, the CC market is estimated to be worth $947.3 billion, indicating that the market is expected to grow significantly, driven by increased adoption of cloud services across different industries worldwide. Amazon Web Services remains the biggest public cloud provider, contributing to 32% of the market, highlighting AWS's dominant position in the CC industry. By 2025, it is expected that there will be 200 ZB (a trillion GBs) of data in the world, highlighting the exponential progress of data worldwide, with an estimated 200 ZB of data expected to be generated and stored globally by 2025. This development is driven by factors like IoT, digital transformation, and growing digitalisation across different industries.
The energy consumption must be reduced to meet the increasing demand and minimise operational costs.

In 2023, the International Energy Agency stated that data centres are accountable for around 1% of energy-based greenhouse gas emissions worldwide. By 2030, data centres and networks will consume 18% of global electrical power. The Net Zero Emissions by 2050 Scenario is a standardised scenario showing a pathway for the global energy sector in order to attain net zero CO$_2$ emissions by 2050, with advanced economies attaining net zero emissions compared to others [23]. Minimising energy consumption is not enough to improve Green cloud computing (GCC) while reducing energy consumption and overall environmental impact should also be considered. Achieving this necessitates comprehensive efforts to improve energy efficiency, reduce resource utilisation, and adopt sustainable practices throughout the entire process of CC services. Figure 1-4 illustrates the growing trend of energy consumed by the data centre in the CC environment. Hence, it is essential to achieve energy efficiency in the cloud environment to meet the rapidly increasing energy consumption driven by the fast development of cloud services.

![Energy consumption in data center](image.png)

Figure 1-4 Energy consumption in data centre [24]

This thesis focuses on tackling energy consumption and performance issues. Firstly, deciding the exact threshold for capacity utilisation (the capacity like memory, bandwidth, or central processing unit (CPU) performance utilised by the host) is critical without degrading the hosts' performance to avoid violations in maintaining a trade-off between the performance level and resource consumption. Research and exploration will be done by setting up an initial real cloud environment scenario using real physical servers and VMs to test different load categories (i.e. when the server is underloaded, overloaded or idle). As a result of the research, a novel technique will
be proposed to help alleviate the problems of VM Consolidation (VMC) for PMs, contributing to a GCC environment.

1.6 Green Cloud Computing
In the cloud environment, most of the energy is utilised by servers that generate heat and need cooling. Owing to the exponential growth in data collection and usage, the demand for data centres is increasing day by day [7]. CC uses several data centres and servers for providing services on a pay-per-use basis, thus requiring extensive power for cooling systems, networking devices, and server farms. So, adopting green technology in CC is significant for many organisations. Efficiently utilising energy is a straightforward and cost-effective technique to save money as well as fulfil growing energy demands [10]. Cloud providers accommodating a variety of applications must follow SLAs, ensuring low access latency, fulfilling task deadlines, and providing secure, consistent, and efficient data management. Managing data centre energy is complex because of the requirement to continuously measure dynamic factors like resource allocation, workload allocation, and traffic conditions.

Moreover, increasing energy costs and a competitive cloud service pricing market are the driving factors that make cloud providers explore energy-saving strategies for their backend data centres [25]. Scheduling the computational tasks and workloads will minimise energy consumption while satisfying the performance objectives. By effectively scheduling the tasks on the basis of resource availability, workload characteristics, and energy efficiency considerations, cloud providers can attain substantial energy savings without affecting performance. Provisioning of the optimal number of PMs for VM allocation involves determining the appropriate number of PMs essential for the host in a cloud environment. Cloud providers can optimise the utilisation of the resources, minimise waste, and ensure efficient performance by precisely provisioning the PMs depending on the workload demands and resource requirements [26]. One of the approaches used to conserve energy is server consolidation, which reduces the number of active physical servers essential for handling workloads.

1.6.1 Server Consolidation
Virtualisation technology is extensively used for simplifying the management of PMs or servers. However, inappropriate implementation of PMs in VMs can negatively influence the performance of the data centre, causing more energy consumption and data centre sprawl. In spite of these potential limitations, virtualisation provides significant benefits like efficient resource allocation, VM resizing, live migration, and server consolidation. Server consolidation is mainly commonly utilised for reducing overall energy consumption in data centres. The ineffective utilisation of resources is the main problem in CC, causing energy waste. The estimated percentages of server
utilisation will be used to decide whether the server is overloaded, underloaded, balanced or idle. The server with average server utilisation levels ranging between 40% and 60% is considered a balanced server, indicating that the server is handling an appropriate number of tasks efficiently without being overburdened or underutilised. An underloaded server is not fully utilised but has minimal resource utilisation. It will occur when the workload allocated to the PM is less than expected.

In contrast, an overloaded server exceeds its resource utilisation, causing reduced performance. In cases of low demand or idle situations, PMs, which are not actively hosting the VMs can be switched off in order to conserve energy. This process helps to reduce energy consumption for cloud providers, aiding in reduced operating costs along with better service availability through workload consolidation on active servers. Server consolidation in CC includes combining multiple servers into a single powerful server through virtualisation technology to enhance the cost-effectiveness and efficiency of the cloud environment. It enables numerous virtual servers to run on a single physical server. The main advantage of server consolidation is the capability to perform migration of VMs, which minimises energy consumption by minimising the number of active servers. Migration allows VMs to move between servers with less system downtime, thus preventing SLA violations while maintaining QoS. It allows a running VM to be moved without service interruption. Server consolidation includes placing multiple VMs on fewer PMs to improve resource utilisation and minimising energy consumption [27]. Server consolidation requires effective VM placement, which is discussed in the next section.

1.6.2 Placement of Virtual Machines
The placement of VMs in the CC environment is vital as the number of cloud users is increasing. The scheduling of VMs significantly impacts the performance and throughput of the whole system. Virtualisation in a PM is performed by Hypervisor, which is discussed in Chapter 4 in detail. The distribution of VM requests adopts a two-tier approach. Data centres configure numerous PMs, and a CSP may operate across multiple data centres. So, initially, the VM requests’ optimal distribution spans across data centres. Then, requests in each data centre are allocated across PMs. When an established mapping of VMs onto PMs occurs, dynamic VM placement is used. The primary goal is to accomplish optimal solutions using the current VM mapping while lessening costs. Optimal parameters may range from minimising response time to reducing energy consumption or a combination of various factors. Decisions must account for the influx of user requests and the VMs and PMs’ status. Its objective is to accomplish optimal solutions from the current mapping while maintaining minimal costs. This procedure avoids shutting down or halting running VMs, necessitating the placement solution to outline live migrations required for shifting to the optimal state. Dynamic VM placement’s successful execution requires consideration of the states of
PMs [28]. After initial placement, VMs need to be migrated for balancing load across different PMs, which is explained in the next section.

### 1.6.3 VM Migration

VM Migration includes moving a VM from one host to another. It is frequently used to balance resource utilisation across hosts, thus optimising energy consumption and performance. When a host is overloaded, VMs on that host will be migrated to underloaded hosts. Migration of VMs can be performed offline or live. Offline migration includes some downtime since the VM operations are suspended and then restarted. On the other hand, live migration has less visible downtime. VM migration in CC will enhance scalability and flexibility. When compared to traditional physical servers that need time-consuming hardware modifications to scale up or down, VMs allow the service providers to allocate resources on the basis of demand dynamically. VM migration allows seamless workload movement between servers or even across diverse geographical regions, allowing industries to quickly adjust to changing needs without interruption. It also helps to improve resource optimisation in cloud environments. By dynamically reassigning VMs across servers, better workload distribution can be achieved, thus ensuring optimal resource utilisation. This adaptive and responsive method lessens the number of active servers required at any given time, thus minimising energy consumption and operational costs [29].

### 1.6.4 Load Balancing

CC loads fluctuate depending on the user demands and resource requirements, making load balancing a challenge that cannot be ignored. During load balancing, the load balancer gets client demands and utilises a load balancing algorithm for allocating these demands among VMs [30]. Load balancing encompasses the distribution and redistribution of the workload among available resources for optimising utilisation and reducing costs, energy consumption, and response times. It ensures effective resource utilisation, user fulfilment, fair resource allocation, and the prevention of bottlenecks and over-provisioning. When CC emerged, it encountered several challenges: QoS management, scaling, security, resource scheduling, data centre energy consumption, service availability, and effective load balancing. Therefore, handling energy consumption and server load becomes a primary concern in CC. Effectively Balancing the load will maximise throughput with reduced response times, costs, and energy consumption, thus enhancing resource utilisation and performance. Server consolidation is also significant in improving these metrics while maintaining SLAs and ensuring user satisfaction through effective load-balancing strategies. For addressing issues like unbalanced loads on PMs, it is essential to integrate server consolidation with load balancing, thus significantly enhancing the efficiency and success of CC environments. The approach utilised to assign VMs to servers in a cloud data centre is vital to achieve optimal results. It determines the QoS provided to end
users and the cloud data centre’s overall efficiency [31].

1.7 Metaheuristic Optimization for Energy Efficiency in Cloud

Metaheuristic algorithms are mainly used for optimisation to resolve complex problems that defy conventional methods [32]. Motivated by natural phenomena like swarm behaviour, genetics, and evolution, these algorithms discover massive search spaces to expose the global optimum of the problem. An intrinsic advantage of these algorithms is their capability to initiate optimisation without any predefined starting point, making them invaluable for solving problems with unknown preliminary conditions. Furthermore, they effectively manage extensive and intricate search spaces compared to the traditional methods. However, their stochastic nature suggests that they may not reliably produce the global optimum. The random feature introduces uncertainty without any guarantee of achieving the optimal solution. Besides, their computational requirements must be considered while confronting large search spaces. As a guiding strategy for the search process, the primary aim of metaheuristic algorithms is to navigate the search space to reveal nearly optimal solutions efficiently. Techniques used within these algorithms vary from straightforward local search procedures to intricate learning processes. These algorithms are non-deterministic and approximate. Although metaheuristics are not personalised to specific problems, they have been applied to several optimisation problems in diverse fields [32].

While achieving better energy efficiency in CC, metaheuristic optimisation includes using advanced algorithms for finding optimal or near-optimal solutions for workload distribution and resource allocation, thus reducing energy consumption. These algorithms use large and complex search spaces for identifying efficient configurations of VMs and physical servers. By adjusting the resources dynamically on the basis of real-time demands and energy utilisation patterns, metaheuristic optimisation improves the cloud data centres’ performance. This approach reduces energy consumption and operational costs while maintaining higher levels of service quality and user satisfaction, addressing key challenges in CC environments [33]. In the proposed approach, metaheuristics is applied in scientific workflow for efficient energy management.

1.8 Energy Efficiency in Scientific Workflow

Workflows act as the vital bridge between computer systems and scientists, comprising a predetermined number of computational tasks, each having a set of instructions that need specific input files. These tasks are dependent on each other. In a cloud application, a workflow task is an individual task or operation within a large process or workflow [34]. Each task in the scientific workflow indicates a distinct computational operation that is aimed at achieving a scientific goal. In recent times,
many scientists have encountered the challenge of handling vast amounts of data through large computational processes. Therefore, they need high-performance computing platforms to execute their data-processing applications efficiently. For this, convenient and robust approaches must be needed to express both the data requirements and computational aspects of their applications. One predominant approach is through scientific workflows [35]. These workflows describe a series of interconnected tasks and their dependencies, altering the main considerations for resource management. Scientific workflows ensure the optimal utilisation of computational resources by building complex distributed solutions. Given the scientific operations’ complexity, these workflow applications are characterised by their heavy dependence on data and computational resources. CC has developed as the main distributed computing environment for supporting the implementation of complex and extensive scientific workflow applications [36].

In CC, energy management is essential to the efficient execution of scientific workflow tasks due to the complex nature and computational cost of scientific workflow. It can be achieved through resource allocation [37], task scheduling [35], and VM migration [29]. Resource allocation involves assigning cloud resources to tasks depending on their particular demands, such as computational power and memory. In contrast, task scheduling schedules the task execution’s sequence and timing to optimise resource utilisation. When a PM is overloaded, VMs belonging to that PM must be migrated to other underloaded PMs to balance workloads and improve performance, thus ensuring the continuity and effectiveness of workflow execution. Achieving efficient execution of scientific workflows requires a centralised scheduling engine to manage workflows widely during their execution. As the entire workflow is known previously, scientific problems often operate based on either an offline or stochastic model. In addition, scientific applications frequently involve interdependencies among their tasks, emphasising the importance of scheduling engines to accommodate Directed Acyclic Graph (DAG)-style workflows. In intricate scientific settings, workflows can have significant challenges related to management. Tasks in these workflows have dependencies; that is, the completion of one task depends on another task, thus potentially causing bottlenecks or delays. Inappropriate workflow task allocation can lead certain parts of the workflow to become bottlenecks, thus affecting the overall efficiency [35].

1.9 Problem Theme
As CC continues to expand and many organisations are migrating from traditional computing, many servers with aggressive VMC are deployed in many data centres worldwide. VMC is installing and configuring multiple VMs in one or numerous PMs. This process aggressively deploys several VMs on a single or multiple PMs in data centres worldwide. This deployment causes resources to experience constraints because of high demand, thus reducing performance and suboptimal cloud service
quality for cloud clients. Moreover, VMC causes increased energy consumption, leading to environmental pollution. In spite of aiming to optimise resource utilisation, the aggressive consolidation approach of VMC can overload the PMs, causing ineffectiveness and performance reduction. This constraint emphasises the requirement for balanced VM consolidation approaches to alleviate performance issues and energy consumption in cloud environments [24].

There are reported cases of dramatic energy consumption escalation in the Asia Pacific and other parts of the world [10][25]-[27][38]. As a result of increased energy consumption, Greenpeace, the United States Environmental Protection Agency and the Climate Server Computing Initiatives [10] have taken inventive measures to make servers more energy efficient. In addition, other initiatives associated with power management techniques through dynamic voltage frequency scaling (DVFS) have also been proposed [39]. However, the involved technological solutions offered depend on hardware and are not controlled according to the needs of the cloud providers. This approach poses a problem as CSPs have aggressively consolidated thousands of VMs [40]-[42] on other PMs, with the majority of these PMs operating partially overloaded or sometimes idle, which in turn causes performance issues for cloud clients [43][44], increased consumption of energy from cloud provider data centres.

Research suggests that a lack of proper planning when using CC can cause severe problems in terms of energy consumption and performance degradation when delivering cloud services to clients [45]-[48]. Another significant research effort has been made by various authors [29][49]-[55] to reduce energy consumption and increase performance while ensuring a better QoS. However, most researchers have considered the CPU as the main component for utilisation. It is unrealistic to judge performance and power consumption without considering utilisation through primary memory or Random Access Memory (RAM), a type of computer memory accessed randomly while the computer is in use. Primary memory is an essential component that must be considered along with CPU usage [56]-[57]. Most energy-aware resource management techniques consider a trade-off between under-utilisation and over-utilisation of resources as a primary way to produce an energy-optimised GCC environment. However, the dynamic nature of workloads raises different issues and deteriorates the performance of existing resource management techniques [29][50]-[55].

In order to tackle the above problems and achieve a move towards GCC, which refers to the eco-friendly usage of computer facilities that minimises energy consumption and contributes to environmentally friendly CC, more research is still needed to improve the energy efficiency of data centres [54]. Furthermore, existing resource management techniques often involve shutting down idle server VMs to conserve
energy. However, this approach can result in additional energy consumption when rebooting the PMs. Moreover, current methods allocate CPU and memory resources to workflow tasks under the assumption that individual best-fit solutions do not negatively impact other solutions. This assumption leads to a multi-objective combinatorial optimisation problem for optimising the execution of multi-interdependent tasks.

Additionally, achieving energy-efficient resource allocation, scheduling, and VM migration requires addressing each non-dominated task while resolving conflicts among other tasks in the solution space. A task is non-dominated when no other task is better at all considered objectives simultaneously. In contrast to single-heuristic algorithms, hybrid heuristic algorithms leverage the strengths of different algorithms to address multi-interdependent task optimisation in cloud environments effectively. The load categories are analysed through practical experiments. Based on the outcomes, a novel approach is developed to achieve energy efficiency in scientific workflows through a metaheuristic algorithm. It optimises energy consumption and achieves high levels of QoS without violating SLAs.

The research questions are given below:

- How does VM consolidation influence energy consumption and resource utilisation across different load categories on practical servers?
- What are the real-time effects on CPU usage, memory utilisation, and power consumption while implementing VM consolidation in different load categories?
- How do metaheuristic-based approaches effectively optimise scientific workflows in a cloud environment with a focus on energy efficiency without violating QoS?
- What are the potential improvements achieved by the proposed approach over the state-of-the-art algorithms in terms of total cost, energy consumption, and overall performance?

1.10 Aim and Objectives

This research aims to propose an energy-efficient CC approach to improve performance and energy efficiency in GCC without violating SLAs.

The specific objectives of the research are:

- To analyse the energy models using VM consolidation on practical servers with various load categories
- To conduct real-time experiments and evaluate CPU usage, memory, and power consumption for each server category.
To develop a novel hybrid heuristics-based energy-efficient optimisation of scientific workflow in a cloud environment through resource allocation, task scheduling, and VM migration without violating SLAs.

To validate the proposed approach in Cloudsim and compare the results with the state-of-the-art algorithms in terms of total cost, energy consumption and overall performance.

1.11 Thesis outline

The Thesis is organised into eight chapters, and details of each chapter are presented in Table 1-1.

<table>
<thead>
<tr>
<th>Chapter #</th>
<th>Title of the chapter</th>
<th>Description of the chapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 2</td>
<td>Literature Review</td>
<td>This chapter covers a wide variety of existing state-of-the-art developments in the area of CC. It analyses the techniques, algorithms and frameworks in CC to improve energy consumption in data centres, QoS and performance from data centres to cloud consumers. The knowledge gap derived from the existing research is also highlighted.</td>
</tr>
<tr>
<td>Chapter 3</td>
<td>Methodology</td>
<td>This chapter covers the overall methodology used in this thesis. It includes the overall block diagram and the system model.</td>
</tr>
<tr>
<td>Chapters 4</td>
<td>Analysis of energy models using VM consolidation on practical servers with various load categories</td>
<td>This chapter covers the analysis of energy models using VM consolidation on practical servers with various load categories. It discusses the initial setup of real-time experiments using type-1 (Baremetal) hypervisor and the initial test of VMs consolidated into PMs. It also evaluates the results in terms of energy consumption, CPU usage, and memory on idle, underloaded, balanced and overloaded servers.</td>
</tr>
<tr>
<td>Chapter 5</td>
<td>Hybrid heuristics-based energy-efficient optimisation of scientific</td>
<td>This chapter covers the proposed HH-ECO approach for scientific workflow using C-PSO with adaptive mutation. It includes different processes like VM allocation, scheduling and VM migration.</td>
</tr>
</tbody>
</table>
1.12 Summary

Chapter 1 has covered the general background of CC, including its history and market size. The architecture of CC elaborates on the difference between frontend and backend systems. It has also covered service and deployment models that provide the bridge between CSPs and cloud clients. Energy consumption, QoS, and performance issues are the primary concerns discussed in this chapter. Research questions were derived from the problem theme, which has helped to provide a guideline for tackling the problems that CSPs and cloud clients still face. The aim and objectives are presented to provide a clear path for this research to address, develop and implement a novel energy-efficient algorithm to tackle the issues of optimal energy consumption and performance in GCC.
CHAPTER 2 LITERATURE REVIEW

2.1 Background
This chapter focuses on server VMC and SLAs for CSP and CC. Through server VMC, proper planning of consolidating VMs in a PM is a crucial step for every server administrator. Due to the aggressive consolidation of VMs, the data centre in a cloud system experiences high energy consumption and low QoS, which causes degradation of performance [42][43]. Though it is well understood that modern high-end servers come with DVFS, a high-speed CPU and high memory capacity, it is still not easy to control large numbers of VMs in data centres, as all of these technologies depend on hardware. In CC, SLA is a process by which a framework is designed to deal with service agreements between CSP and CC to ensure CSPs improve efficiency and provide better energy efficiency, QoS and improved performance [58]. The major concepts covered by the literature survey are discussed below.

- **Workflow optimisation**: It aims to improve the overall performance and efficiency of task execution in a workflow. It includes scheduling tasks, allocating resources, migrating VMs, and optimising the workflow's execution in order to satisfy performance objectives like minimising makespan or maximising resource utilisation [24].

- **VM resource allocation**: This includes provisioning the computational resources such as CPU performance, storage, memory, and network bandwidth to VMs, depending on the requirements of the hosted applications. This process aims to ensure that each VM has access to adequate resources to execute its tasks efficiently while maximising resource utilisation across the cloud infrastructure. Resource allocation techniques may include dynamic scaling, where resources are allocated or deallocated based on workload fluctuations, and load balancing, which distributes workload evenly across VMs to prevent resource bottlenecks and overloading [37].

- **Task scheduling**: Task scheduling in CC involves determining the optimal execution sequence and task allocation across VMs for achieving different performance objectives like reducing makespan, improving resource utilisation, and satisfying deadlines. Scheduling decisions consider several factors like resource availability, task dependencies, VM capacity, and SLA requirements. Methods for task scheduling comprise static scheduling, in
which the tasks are allocated to VMs during the initiation of the execution process, and dynamic scheduling, in which scheduling decisions are dynamically made on the basis of real-time workload conditions [31].

- **SLA**: It indicates the agreement between service provider and client that represents the expected level of performance, service quality, and availability. Violation of SLA happens when the service provider fails to meet the agreed-upon service standards. This violation causes a loss of reputation, financial penalties, or other consequences for the service provider [59].

- **Trade-offs between performance, energy consumption, and SLA violations**: It refers to the fundamental problems in balancing these competing goals. For instance, optimising performance may need more resource utilisation, causing increased energy consumption. On the other hand, reducing energy consumption may affect the performance, possibly leading to SLA violations. Maintaining the appropriate balance includes making strategic decisions and trade-offs on the basis of specific requirements and priorities of the system. The optimal framework must consider these trade-offs to optimise performance, lessen energy consumption, and mitigate SLA violations effectively [59].

Let us consider a scenario in which an approach is developed for achieving optimised resource allocation, task scheduling, and VM migration in a cloud environment. Its main objective is to improve the efficiency and performance of task execution while reducing energy consumption and adhering to SLAs using workflow optimisation techniques. This approach optimises the workflow execution by dynamically allocating resources, scheduling tasks, and migrating VMs between hosts for balanced resource utilisation, improved makespan, reduced energy consumption, and managing SLA compliance. While satisfying these objectives, several constraints include resource availability, SLA requirements, and increased waiting time ratio. Total cost must be carefully considered and accomplished to attain the desired optimisation level [59].

### 2.2 Taxonomy Design

A taxonomy used concerning CC revolves around energy savings and performance. Based on the literature review, the typical classification of VM consolidation techniques /algorithms and frameworks for SLA is as follows: see Figure 2-1, which depicts the Taxonomy Diagram.

- **General utilisation of the server's components**: This considers the utilisation of CPU, storage and other input/output (I/O) activities. Though the CPU is considered the primary component, other components like storage and I/O activities are included to ensure all issues related to the QoS and performance in CC are tackled. There is much research which has been done in [60][61] to prove the utilisation of these components to contribute to performance
CHAPTER 2. LITERATURE REVIEW

degradation, energy consumption and QoS in CC.

- **CPU utilisation threshold**: This checks the CPU utilisation based on the CPU threshold (lower and higher). Different researchers have done comprehensive research on CPU utilisation by considering the threshold usage; when the threshold is underutilised, algorithms have been designed to migrate the VMs to the underutilised CPUs. Similarly, when the thresholds are overutilised the VMs are dynamically migrated to other CPUs with enough capacity.[62]-[66] The threshold value for CPU utilisation to trigger migration can differ on the basis of specific characteristics, requirements, and workload patterns of the system. Usually, the threshold value is determined based on factors like performance objectives and workload characteristics.

- **CPU utilisation based on millions of instructions per second (MIPS)**: This checks the utilisation of CPU based on MIPS. This utilisation is another area where many researchers have used MIPS to check the performance and calculate the utilised and non-utilised processor capacity. Based on the calculation of MIPS, algorithms have been developed to predict the underloaded and overloaded processors for migrating VMs from overloaded PMs to underloaded PMs. This method is easily understood by helping calculate the speed Cycle per second, average clock cycle per instruction and execution time. However, it is not always very accurate as different CPUs of different capacities will calculate the MIPS differently [67][68].

- **Network topology (rack utilisations)**: This checks and detects the rack utilisation (underutilised and non-underutilised). Racks are metal frames which house the network hardware, such as switches, routers, access points and modems. In a cloud environment, the network topology of routers and switches is very comprehensive and massive, depending on the capacity of CSPs. This rack utilisation is another area where some researchers have considered it very important regarding performance degradation and energy consumption.

- **Memory utilisation**: Memory is another crucial area to be analysed when checking servers’ performance in CC. Though it is not considered a significant component that contributes to high energy consumption like the CPU, it is an essential resource that is likely to cause performance degradation if it is not consolidated wisely. Only some existing research has been done by utilising memory usage in order to improve performance in CC.

- **Framework for SLA**: SLAs are very important for both CSPs and clients; SLAs allow the two parties to govern their understanding of resource usage in CC. This framework maintains the agreement between the CSP and cloud consumers regarding energy optimisation, maintaining stable performance
and QoS. Any CSP that wishes to host cloud services successfully for organisations should have a robust SLA framework to ensure a good balance of resource distribution from CSPs to clients.

![Figure 2-1 Taxonomy Diagram](image)

Figure 2-1 above describes the literature review which has been done by the author since the research of this thesis is about the optimisation of energy-efficient CC, which aims to reduce energy consumption, and minimise performance issues in CC. The taxonomy above has mainly concentrated on the existing research which focuses on GCC. This literature has reviewed the main components to have a broader picture of GCC that results in energy consumption and degrade CC performance. Different
techniques and algorithms have also been reviewed, enabling the author to pinpoint the significant areas of research that can contribute to GCC.

2.3 State-of-the-Art

In CC, heterogeneity, the arrival of dynamic workloads over time, and workflow tasks encounter problems in resource allocation, task scheduling and VM migration. Several researchers have proposed better heuristics, techniques and algorithms that control the utilisation of VMs on servers to improve CC and increase its efficiency. MIPS is a characteristic of the hardware in PM. The estimation of the CPU performance (MIPS) for a VM at any given time is performed by analysing the behaviour and resource utilisation patterns of the VM under different workloads. It can either be analysed by performance profiling that identifies CPU-intensive tasks and calculates their resource utilisation based on the execution patterns or by performance monitoring tools that keep track of the CPU usage, and memory usage of the VM in real-time. There are several resource management techniques which have been proposed and implemented in a framework for virtualised cloud data centres [69]-[71]. This section surveys energy-efficient resource management techniques and frameworks along with their strengths and limitations. The detailed analysis determines the essential characteristics of resource management algorithms and discusses various issues and solutions to optimisation for GCC.

2.3.1 Workflow Task Scheduling Approaches

The authors in [72] have proposed a multi-objective task scheduling algorithm and assigned tasks to VMs to improve throughput and reduce the cost of processing time. The assignment of task priority relies on the QoS requirements of the applications. The tasks and VMs are sorted using non-dominated sorting based on the running capacity before task scheduling. This algorithm is suitable for QoS parameters such as execution time and cost but is not appropriate for energy efficiency. A meta-heuristic Particle Swam Optimization (PSO) algorithm is exploited to minimise the execution cost of the application in [73]-[75]. However, the earlier standstill of the PSO particles before reaching an optimal global value and the low-level diversification among the particles create premature convergence when many jobs are running on the cloud. An HHWS in [74] determines the optimal solution for the workflow tasks by combining a PSO algorithm and a Gravitation Search Algorithm (GSA). Even though the method provides scheduling of workflows by considering cost and deadline constraints, it did not achieve the globally optimal solution while applying the GSA to tasks. This approach may cause a detrimental effect on the top-k particles in the PSO, leading to local convergence. Hence, the execution of workflow tasks is dominated by factors like cost and deadlines without efficiently considering other critical aspects essential for
attaining the best overall solution.

The research work [75] presents a discrete Comprehensive Learning PSO (CLPSO) approach for scheduling workflow tasks based on a set-based PSO (S-PSO) to address the user-defined QoS constraints. However, the meta-heuristic algorithms for workflow applications reduce the speed to reach the global optima. EnReal model [76] categorises the VMs into splittable and non-splittable VMs and effectively utilises the slack time of the workflow tasks. Consequently, it schedules the online scientific workflows in the cloud using different energy-efficient dynamic scheduling methods. A multi-objective list-based workflow scheduling heuristic [77] maintains a trade-off between the Makespan and energy efficiency, which is the extension of the Heterogeneous Earliest Finish Time (HEFT), named MOHEFT. Applying the empirical models predicts the execution time and energy consumption for the workflow tasks based on the historical data of the actual workflow task execution.

However, historical execution-based workflow scheduling leads to computational complexity. The Evolutionary Multi-objective Optimization (EMO) model [78] resolves the problem of workflow scheduling on an Infrastructure as a Service (IaaS) platform. It presents the encoding, population initialisation, fitness evaluation, and genetic operators concerning the problem, which ensures higher stability of task-instance assignments of workflows with minimised Makespan and cost. Even though it mutates the tasks randomly, it needs to perform the mutation during the execution of the workflow tasks, leading to ineffective results in higher energy consumption for the dynamic cloud environment. A Multi-Objective Genetic Algorithm (MOGA) ensures the minimised Makespan based on the deadline and budget constraints and the improved energy efficiency for workflow scheduling in the cloud by considering the conflicting interests of the cloud stakeholders and employing the DVFS. Even though the MOGA model improves resource utilisation during workflow scheduling in the cloud, it fails to improve user satisfaction [79]. A Cost and Energy-Aware Scheduling (CEAS) algorithm [80] minimises the energy consumption and execution cost while considering the deadline by incorporating sub-algorithms, including VM selection, task merging, VM reuse, and slack time reclamation algorithms. To schedule the workflow applications, the research work [81] applies a hybridisation of GSA and HEFT to accomplish the bi-objective optimisation of minimising both the Makespan and cost. However, it provides ineffective results for the complex tasks that arrive in the cloud environment.

The Reliability and Energy Efficient Workflow Scheduling (REEWS) method [82] maximises reliability. It minimises energy consumption while maintaining workflow dependency and considering the QoS and deadline constraints specified by the users. It achieves the objectives by applying four main phases: priority computation, task clustering, target time distribution, and cluster assignment of the resources based on the voltage or frequency levels. The research work [83] presents an energy-efficient,
cost-effective, and QoS-aware scheduling model for workflow applications in the cloud. It employs the per-core DVFS method on multi-core heterogeneous processors and considers the effects of input errors on the execution time of the tasks.

A hybrid algorithm-based scientific workflow scheduling approach [84] considers execution time, load balancing, and monetary costs while scheduling workflow tasks on CC resources. It applies preprocessing to prepare the tasks for the scheduling algorithm based on the dependencies and employs the PSO algorithm for workflow scheduling. A hybrid meta-heuristic algorithm [85] resolves the discrete task-scheduling problem in the distributed computing environment using the hybrid discrete PSO (HDPSO) algorithm. Applying the hill-climbing algorithm with the local search trend balances exploitation and exploration and avoids the suboptimal trap. The energy-aware workflow scheduling approach [86] employs the bat algorithm. It presents the energy-aware, time, and throughput optimisation heuristic (EATTO) algorithm to minimise the execution time and energy consumption of the compute-intensive workflows in the cloud environment.

A firefly-based multi-objective scheduling strategy [87] considers multiple objectives such as Makespan, the workload of servers, reliability, and resource utilisation in the cloud during the scheduling of the workflow tasks. It assigns the workflow tasks to the appropriate VM instances, balancing the workloads and utilisation and minimising the Makespan with the reference of the deadline. An improved many-objective PSO (I_MaOPSO) approach [88] addresses the workflow-scheduling problem in the cloud, considering four objectives: cost minimisation, energy consumption minimisation, makespan minimisation, and reliability maximisation. The four greedy heuristic methods maintain the equilibrium between exploitation and exploration during the scheduling. It tends to the convergence of the non-dominated solutions along with the generation of a high-quality initial population, social leader selection, and cognitive leader selection to balance the intensification and diversification capabilities of the scheduling algorithm. Moreover, it maximises the throughput without compromising the QoS while resolving the multi-objective optimisation problem.

2.3.2 Resource Allocation

Resource allocation is the process of allotting VMs to PMs. The user establishes SLAs with the CSP [28], and resource management techniques attempt to ensure the SLA as well as reduce energy consumption [59][89]. Managing the SLA is formulated as a multi-objective combinatorial problem in most of the recent applications [28][23]. The research work [23] applies an ant colony optimisation algorithm-based VM placement strategy to reduce resource waste and energy consumption. Applying the Ant-Colony Optimization (ACO) algorithm obtains a Pareto set to resolve the multi-objective constraints. A DDVM mechanism in [90] allocates and reactively reallocates the VMs
using a first-fit heuristic algorithm to accomplish the distributed and dynamic adaptation, eliminating single-point failure. While considering the restart time of the inactive server, energy consumption may increase. However, its reactive VM management leads to a cumbersome optimal solution selection process during the arrival of numerous similar applications requiring VM management.

A dynamic resource allocation technique in [91] optimises the allocated resources in VMs based on user demands. This technique adaptively multiplexes the VMs to PMs according to the dynamic workload over time. However, heterogeneity among PMs and VMs encounters allocation issues within the above techniques since it tends to imbalance the load of PMs. The research work [92] optimises energy efficiency during communication and computation by enabling intelligent resource management for real-time vehicular cloud services. It accomplishes the QoS requirements by distributing the intensive computation to the distributed fog computing infrastructures. However, this adaptive resource management model fails to focus on reducing energy consumption during VM migration. In these situations, the energy-aware resource allocation algorithms presented in [93] and [94] consider resource utilisation and energy consumption. Energy-efficient resource allocation has been proposed for traditional data centres with distinguishing features such as heterogeneous workload, average load rate, and intensive completion time [95].

Research on reducing energy consumption in cloud infrastructure, which moves VMs to a limited number of servers to allow the concentrated physical servers to utilise the maximum resources and set other servers to a low power mode, has been attempted [52]. An algorithm was developed [96]. It was based on a gossiping protocol that occasionally runs to spot the VM allotment's current situation. It migrates the VMs from an empty server to maximise power consumption. It uses the live migration features the VM monitor management provides to complete the migrations. The algorithm mainly considers the number of VMs operating on each server. It checks the capability of a particular server through gossiping to determine if it can allow migration from a source server, which will have to ensure the destination server is not full before the migration. It uses peer-to-peer (P2P) service to gossip to determine the current VMs hosted by a particular server. Each server on the cloud will operate by using two threads [96], "active thread and passive thread," whereby a particular node can send the number of VMs to each neighbour in order to determine which server runs the higher number of VMs to allow the VMs to be migrated when the maximum is reached.

The authors in [97] have proposed an idea on resource allocation based on negotiations to analyse the utilisation of cloud resources and the percentage of job execution in CC. This research proposes involving multiple CSPs in distributing cloud services to cloud consumers. They have claimed that a single CSP may need more resources to allow multiple cloud consumers to use the service of CC efficiently. The
bargaining model is one of the central concepts proposed by the authors, whereby the model will initiate contract establishment to tackle the scarcity of resources in CSPs, ensuring better service and optimal management of resources from CSPs to cloud consumers. The sharing and allocation of resources among CSPs is called a multi-objective optimisation problem.

They [62] have used the real-world data trace for simulation in a cloud sim tool, and they have set parameters, application, and utilisation of resources to define the ratio of allocated resources to requested resources. For the experiment to work correctly, the authors have simulated three CSPs to provide a multi-objective optimisation solution. They randomly used the number of resources, such as CPU and CPU cores, primary memory (RAM), and storage, from three different CSPs and ran the algorithm to distribute the resources to cloud users (CUs).

After running the experiment with the real-world data trace, they claimed to be getting improved resource sharing from multiple CSPs with a quick response time. This technique will provide stability for resource sharing among the CSPs and ensure a very smooth allocation of resources to CUs. As a result, they prove that CSPs with resource scarcity can improve their service and avoid daily performance issues.

The research done by the authors in [62] provides a good solution because it addresses a persistent challenge in CC like improved resource allocation and improved services. Involving multiple CSPs will improve resource allocations to CUs; however, the idea will need an improved and comprehensive framework to convince CSPs from different domains to initiate a coalition. Most CSPs, such as Amazon, Microsoft Azure, Google Cloud, Alibaba, and others, have comprehensive resources and multiple data centres worldwide, now operating with improved services to their clients around the globe. Additionally, most of them are competing with each other and may prefer to avoid an alliance, which will jeopardise business secrets and raise security issues.

Additionally, this technique solves one resource allocation problem with performance but needs to look at the central issue: energy consumption and carbon emissions. The algorithm can consider parameters and thresholds that can help tackle the energy consumption issue in CC. The idea could also consider energy awareness and solutions to save energy and the environment.

The authors in [98] have proposed an idea for resource allocation based on a Genetic Algorithm (GA) for CC. Their study found resource allocation to be unbalanced, resulting in longer processing time; this is a critical problem in a CC environment. Due to this, they have proposed two phases; the first phase is the extrapolation of the workload in the VMs, as they believe this to be more substantial than the threshold setting. Moreover, the second proposal is reusing GAs to allot the tasks and find an ideal solution for the allotment of cloud resources. They have initiated a problem theme and parameters to be used as the system model for CC.
In the system model, they have considered intra-migration and intermigration costs, which they believe many researchers still need to consider. Intra-migration is migration within a PM, and intermigration is the migration process between PMs. In order to envisage the state of each VM in PMs, they have used the Markov Decision Process (MDP) [99] model. Then, they used GA [98] to estimate resources for each VM to decide which task must be moved from one VM to another. In each VM, CPU and memory resources have been used along with the transmission time for the task migration.

In the experiment and simulation, they used Matlab 2020b and set the number of PMs to 5 and the number of VMs to 30. They have tested several times using GA and found the result to provide a good load balance and less processing time in the cloud. Even though the research achieves effective load balancing, it is more efficient only when using a few PMs and VMs. It is not enough while considering real-time cloud data that can offer more accurate estimations of the result.

It is also imperative to consider the overloaded and underloaded PMs and VMs, which will predict the accurate migration of tasks with VMs and PMs.

The research done by [100] has proposed an idea for resource allocation in CC using an enhanced maximum-minimum algorithm (EMM) due to load imbalance and starvation in CC. They have compared the EMM algorithm with the original Min-Min (MM) algorithm. They experimented on Cloudsim, which used 150 tasks assigned to four VMs, and all the VMs have been allotted to two data centres. The experiment was executed using both algorithms, MM and EMM.

In the MM algorithm, the tasks are scanned and determined as short and longer tasks. The algorithm will first deal with shorter tasks until their completion and assign them to the VMs. Then, it moves to more extensive task execution and is assigned to the VMs. The process takes a very long time, as larger tasks will have to wait for smaller tasks to be completed. As a result, it causes inefficient resource utilisation with a long duration.

In the EMM algorithm, the tasks are selected and take an average or nearer average instead of categorising between more minor and more extensive tasks then a decision is made to assign the tasks. By doing this, they have claimed to have reduced the time to complete the task with minimal delay. After verifying the results, they found that EMM utilisation is 70% to 90% compared to the MM, which ranges from 60% to 80%. In addition, the makespan in EMM takes a shorter time than in MM. The research has compared previous work well and provided proof to provide better utilisation of resources. However, the authors need to check the reality of data centres by considering the complexity of VM consolidation in CC. A simulation needs to consider more VMs within multiple PMs and set multiple tasks to check the overloading and underloading of data centres. In addition, the research did not discuss any issues
related to energy consumption or environmental awareness, a critical challenge many global CSPs face.

### 2.3.3 VM Migration Approaches

However, the assumption of allocating heavy-load applications to new-generation servers and light-load applications to older-generation servers deteriorates the efficiency of load balancing. Most resource management techniques exploit heuristic optimisation algorithms to solve the energy-efficient placement of VMs in a cloud environment [101]. However, only a few techniques consider VM migration during running time, and these techniques result in significant time overhead. A scheduling method for multi-VM migration in the cloud is proposed in [102] to reduce VM migration time. However, the lack of correlation between migrated VMs considerably degrades performance and increases migration time [103]. Determining VM migration time is a significant challenge in CC services [104]. It is necessary to consider the load category (i.e., idle, underloaded, balanced and overloaded) in attaining a better trade-off between resource utilisation and Makespan without violating the SLAs.

The passive thread will only listen to the other peers' communication and decide whether the VMs should be pushed to the new destination or pulled from their original location. They have systematically reduced energy consumption by migrating the VMs from the underloaded PMs to PMs, which have space for migrating VMs from other servers. Though the algorithm can detect the number of VMs in each PM, it is challenging to judge when and how they become overloaded or underloaded without setting up parameters such as CPU, memory, and network bandwidth usage. There will be a very high chance of overloading the destined PMs, which may disrupt service to consumers due to the unpredicted workload by cloud consumers. Authors [30][105][106] have identified various performance problems in data centres as an unpredicted workload sent by cloud consumers; most of the performance issues discussed were caused by unpredicted CPU, memory, and network bandwidth usage. Switching the PMs to power-saving mode is a positive idea, as it will save energy while simultaneously ensuring the QoS of the end users of CC. The power-saving mode or low-power state minimises the power drawn from idle servers [107]–[109]. For example, the power-saving mode can draw from 2W–5W compared to 200W when operated in normal 'on' mode [110].

The study [111]–[112] suggests a similar migration technique in the VMs, taking the number of PMs available in the data centres and squeezing them into other PMs, which are not fully packed. Although the concepts look similar, their techniques vary [50], one allowing the idle machine to be put in a low-power mode and the other allowing the idle machines to be powered down [111].

VM migration is considered a strategy for saving energy and improving the QoS
These authors have reviewed the general power model that helps identify which CC components consume energy when the workload increases. Different components, such as PMs with VMs, storage, and network components, have been reviewed regarding energy consumption. It mainly focused on CPU usage, which is considered to be the most crucial factor in terms of utilisation. They have proposed VM placement and used a modified best-fit decreasing algorithm. All the VMs will be sorted based on CPU usage in decreasing order, which is quite different from research based on MIPS. Two steps were introduced, which claim to be proper and valid. The first step is using a threshold system, which is the better solution to prevent the resources from being overutilised. Higher and lower utilisation have been considered rather than just one alone (higher utilisation), which will always allocate the VMs to PMs. The authors suggested using dynamic migration of VMs, which will fit the changing behaviour of VMs.

Three methods that will decide which VMs will need to migrate are as follows:

- Minimisation of migration: a few numbers of VMs migrate to reduce migration overhead.
- Highest potential growth: the VMs that seem to have the lowest CPU usage are migrated.
- Random choice: choosing the actual number of VMs randomly.

As per the algorithm, they have suggested a gossiping protocol similar to that decides the number of VMs a particular PM can handle. The protocol consists of two threads: an active thread and a passive thread. The active thread sends the message to the neighbouring host, while the passive thread listens and transmits the updates.

The concentration of power consumption has been based more on CPU threshold utilisation compared to MIPS. The work of [50] allowed the idle server to be switched to power mode, whereas [70] was to switch off all idle servers. The idea allows VMs to be consolidated and migrated based on two factors (high threshold and low threshold utilisation) [114] that will reduce the over-utilisation of resources. The algorithms allow equal distribution of resources and prevent migration overhead. Based on this idea, it is evident that the utilisation of resources will be minimised, reducing energy consumption. The algorithm also used the same idea as [52] (active and passive threads). It allows the VMs to be migrated or not, depending on the availability of resources on the PMs. The positive point is that it may improve the QoS if all the VMs work with minimal allocation, as the techniques have been designed.

Moreover, the idea will only work perfectly if the loads of all CUs are stabilised. The techniques and algorithms will also harm quality and energy consumption if the VMs
are turned off [115]. Turning on and off the VMs results in more power consumption and inhibits the resources' capability to work. As mentioned earlier, switching off the idle servers may produce three negative points: the energy consumption might be increased if there is a frequent increase of "on and off" within a short time, the hardware might get damaged, and the QoS will be decreased upon boot-up of the servers. The boot-up of servers can be judged based on the application and the number of processes it would need to boot. The research does not only consider servers and VMs as sources of energy consumption. It goes beyond that by considering a cooling system and the racks of network structure that hold the physical servers.

The authors [50] have also formulated the power model based on three components—servers, networks, and cooling devices—where the equation is based on the total power consumed by all servers on the network layers. These are referred to as top rack switches, aggregation layer switches and core routers, which link the entire network. Based on the above idea, the authors have used the best method of minimising energy consumption by considering resources such as network topology (network racks), PMs, and VMs. The energy may be reduced dramatically, but it is unclear how the algorithms determine the unpredictable tasks that the CUs send over a longer period. The sorting of VMs from underloaded and overloaded servers within the racks is not specified, nor is it specified when the server becomes overloaded or underloaded. Since the algorithm has been set to operate on minimum and maximum levels, the prediction utilisation is not precise, and this may cause the systems to either become more overloaded or leave them idle. This process increases energy consumption and degrades performance; if the unexpected tasks overload the powered servers and the racks, it will take double the energy and time to pull the new tasks to the VMs from power-saving mode servers and racks. The VM utilisation prediction is not based on the utilisation percentage of the CPU. It is based on the MIPS, which makes it difficult to predict the cloud task even though there is an SLA.

Authors in [116] used a similar concept as used in [41]. The authors in [41] have used automation of thresholds (dynamically adjusted based on various conditions), whereas the authors in [116] have used fixed thresholds (constant values without any automatic adjustments). Both of these methods will minimise the number of idle servers and the number of migrations, which are claimed to have cost implications. They claimed that these approaches help to optimise resource utilisation and have cost implications like energy consumption or management overhead due to the reduction in the number of migrations and idle servers.

The research in [41] has taken the positive path of migrating VMs from the least loaded server to other servers with enough resources and turning the idle servers into power-saving mode. The other strength is that they have considered the total amount of memory and network bandwidth used during the live migration. This process helps to limit the number of migrations and helps cloud systems operate smoothly. They
have also taken QoS as an important issue rather than energy consumption alone, as they have used an algorithm that considers performance degradation caused by host overloading and VM migration. However, CPU utilisation is at 100% in order to measure the performance degradation due to migration. Based on the practical experiment done by the work of this thesis using ESXi Hypervisor, the utilisation can degrade to its performance when it reaches 75% or above server overloading (which is discussed in Chapter 4). As that is the case, the estimation of 100% is very high, as it leaves us without space to accommodate more tasks. The authors have used threshold automation that will depend on the workload pattern; as the cloud load is not easily predictable, improving the performance, as claimed, might be very difficult. It also needs to be clarified how the system decides the automation of the threshold.

Other researchers [117] have proposed an algorithm for a greener cloud by mimicking the behaviour of honeybees in their research. They have observed overloaded, underloaded, and idle CPUs. They have introduced proper management of overloaded, idle, and underloaded CPUs, intending to turn off the idle CPUs and schedule the task of the overloaded CPU to the underloaded CPU to save energy and prevent QoS violations. This algorithm’s major problem is finding the proper way of managing the idle CPU, the underloaded CPU, and the overloaded CPU. The research has divided the jobs into two parts; the first deals with managing underloaded and overloaded CPUs to reschedule tasks. This part has been compared to a bee colony, classifying poor-quality nectar as low-threshold CPU utilisation (underloaded CPU) and high-quality nectar as high-threshold CPU utilisation (overloaded CPU). The second part deals with idle CPUs to reduce power consumption. This part is compared to an ant colony algorithm, which will be used to find the shortest path and detect the idle CPU.

The research done by [102] is good since they have done practical experiments using Ubuntu (Linux). The framework proposed is well-organised and linked to the algorithms proposed. The framework is designed in stages and allows cloud consumers to utilise the service based on the limitations and agreements with the cloud providers. Based on the two algorithms (bee and ant colony algorithms), properly managing overloaded and underloaded servers has been handled to allow smooth service operation with proper balancing techniques.

The threshold value of the CPU is the best way to judge whether the servers are becoming overloaded or underloaded. It is better to use a threshold value based on the utilisation of the CPU, which is much more practical and faster than MIPS. This idea will be much more difficult, especially when it is undeniable that idle machines with large numbers of VMs may consume 50%–70% of energy utilisation [118][42][54][119]-[121]. However, they have not mentioned how the idle CPUs will be triggered. Moreover, turning off the idle CPUs will not be a perfect idea; if the CPU is turned off for a short time and rebooted, it tends to consume more electricity. The
other issue is that if the underloaded CPUs become overloaded by an unexpected request, how will an idle server ensure QoS without interruption since it has to take the process of rebooting into account? The authors in [122] propose an energy-aware analysis, which is based on RAM utilisation (RAM-based host overloading detection), which is quite different from [41][113][116], which is concentrated on the usage of CPU through its utilisation and MIPS.

Let us consider the technique [122] of RAM-based overloading detection. The algorithm and the equation suggested will help improve the data centre’s performance. The technique is familiar, using similar techniques [41] of minimal utilisation. Though an algorithm can migrate the VMs from one PM to another with minimal utilisation, it needs to be clarified how the over-utilisation of resources will match the under-utilisation to allow migration. Since the utilisation of VMs is calculated by adding the required amount of RAM of the cloudlets, which operate in the VMs and is then divided by the total capacity of RAM in a particular VM, it may not be that easy when applied to the vast environment of data centres, where millions of users are involved. The task of each user is unpredictable. Since the CPU is considered an essential component in any computer system, judging the degree of performance improvement from the primary memory is impractical.

During the intensive load on the VMs, the CPU is overused more than the RAM (refer to Chapter 4 for test bed results). As that is the case, performance cannot be guaranteed based on using RAM alone.

Based on the research [40], it has been proven that the migration of VMs from one PM to another causes a violation of service as it degrades performance. They have considered the cost of live migration to be 10% of the total performance. Based on this idea, their technique intends to reduce the number of migrations. Though it is very accurate that migration of VMs degrades performance and increases power consumption due to the need for extra memory, CPU speeds are to be sacrificed during migration. It may be challenging to limit the migration of VMs in an environment of cloud usage, such as Amazon or Google. However, if migration is limited, there will be performance problems based on the over-utilisation of the CPU. The research has mostly considered CPU utilisation and avoided memory utilisation, which again may cause performance degradation.

2.3.4 VM Migration with Server Consolidation

An Energy-Efficient Hybrid (EEH) framework [123] performs request scheduling and server consolidation to improve the efficiency of the data's electrical energy consumption. Initially, it sorts the user requests for the time and power needed to execute the tasks before scheduling. It determines the underloaded and overloaded servers to migrate the VMs to the new servers to reduce energy consumption. The
idea [113] claims improved VM consolidation, which covers the node components that seem to consume more energy—the CPU, memory, hard disk, motherboard, and Network Interface Card (NIC).

The authors in [50] have also proposed an idea that seems beyond the standard VM migration techniques based on components such as memory and CPU utilisation in a particular PM. This type of consolidation employs four different algorithms to achieve less energy consumption, improved QoS and optimal placement of VMs. The four different algorithms detect the underutilised and non-underutilised server by selecting the number of VMs from the VM list and by using a Modified Best Fit Decreasing algorithm (MBFD), which places VMs in a rack-by-rack algorithm, VMs in a non-underutilised rack algorithm, and migrates VMs on underutilised racks consecutively. Therefore, an algorithm functions as follows:

- The first algorithm (MBFD) sorts the VMs in decreasing order using MIPs. The VMs are sorted from the server list and find non-empty or non-underutilised servers to decide where they should be migrated.
- The second algorithm (Place VMs Rack by Rack (RBR)) sorts the VMs and then racks them in decreasing order, according to the user request of MIPS. If the racks are identified as underutilised, the algorithm will search and look for the racks with enough resources to migrate the VMs of that server into underutilised racks.
- The third algorithm (Place VMs in None-Underutilized Rack (NUR)) sorts the VM list after the search. It searches to find the best server amongst all the non-underutilised racks.
- The fourth algorithm (Only Migrate Underutilised Racks (OMUR)) uses a technique to consolidate the racks and place them in the VMs. Towards the end, migrate the VMs from underutilised racks to non-underutilised racks with enough resources.

The objective is to take care of consolidation issues, which would minimise the number of active racks, which will, in turn, switch off the unused racks, cooling systems and idle servers after proper migrations. The research has claimed to improve energy by an average of 14.7% in the entire data centre.

Another research work [41] has proposed an algorithm that will allow dynamic VM consolidation by using online live migration. The algorithm allows the idle PMs to be switched to sleep mode to optimise power consumption and performance. The research has discussed essential aspects. The first aspect is the cost of the live migration of the VMs. The authors have demonstrated that the degradation of performance during VM migration depends on the application's behaviour at a particular time. As that is the case, they have estimated the average web application degradation and downtime to be 10% of CPU utilisation, which means each migration may cause an SLA violation, so limiting the number of migrations is imperative.
They also considered QoS a vital issue and reflected the minimum throughput and maximum time response that the systems give. Based on this, two metrics have been proposed in order to measure violations of service in IaaS:

- The percentage of time taken by the active host to reach 100% CPU utilisation;
- Degradation of performance caused by migration

The team also suggested several heuristic algorithms for dynamic consolidation based on the analysis and usage of VMs. The reason for using 100% utilisation is that if a particular host happens to utilise the CPU fully (100%), VMs will have a problem regarding performance level. The problem of dynamic VM consolidation is divided into four parts as follows [41]:

- Considering when the VMs will be overloaded and which one is required to be migrated from a particular host
- Considering when the VMs will be underloaded and how the decision will be made to migrate all the VMs from the host in order to put it in sleep mode
- When selecting the VMs that will be migrated from overloaded hosts
- Decide the new allocation of VMs to overloaded and underloaded hosts (PMs).

It is difficult to predict the workload and to assure QoS. Thus, the research has changed the concept by introducing automation of thresholds instead of fixed thresholds, which depends on the workload pattern.

The other research has taken a different approach, which involves reusing memory mechanisms for dynamic VM consolidation [57] to reduce the amount of transferred data during migration. They have claimed that the existing live migration techniques take a long time to migrate since they have to transfer the entire image of the memory from one host to another. They have two types of physical hosts: a shared server that handles VMs, which is not taking on a heavy computation load (close to an idle server), and dedicated servers, which take active VMs. All VMs will be consolidated to the shared servers if they are idle. If all the VMs are active, they will require higher performance, so they must be transferred to the dedicated servers to maintain performance.

As VM migration happens many times in cloud systems, they claimed there is a high chance that VMs will migrate to the original host, as has been done before. For instance, if the VMs happen to move from an idle state to a busy state, there is a need for them to be moved from the shared server to a dedicated server to maintain performance. In order to avoid the total transfer of memory from one host to another, an image of memory from one host (i.e., from host Y) can be left and reused when the VMs are transferred back, as shown in Figures 2-2 & 2-3 [57].
Each memory page within the VM (V) is assigned a generation value (the number of updates made to a page since the VM’s boot). On VM boot, all page generations are reset to zero to form the VM’s generation table (V, A), where A is the host. These generation tables are managed by the generation server, which is responsible for the generation tables of all VMs. The process of VM migration is shown in Figure 2-2 for the initial migration from host A to host B. It includes temporarily halting V on A, identifying updated memory pages since V’s boot utilising dirty page tracking, forwarding this information to the generation server for updating (V, A), resuming V after a certain time, and finally, migrating V to B while maintaining the memory image on A for potential reuse. On the other hand, Figure 2-3 demonstrates the process of migrating V back from B to A. Prominent differences in this process include stopping V on B, detecting updated pages up to the last migration to B, updating (V, B), comparing (V, A) and (V, B) for identifying reusable pages, sending these page numbers to the hosts, resuming V, and migrating it back to A without sending reusable memory pages [57].
The authors in [57] have taken a different approach to memory reuse than all other research. Since the proposed idea is unique and works in two ways, the system can improve its performance better than a complete image transfer if the cloud systems work as the authors predicted. VMs can migrate back to their original host since the VM consolidation process is executed as soon as required, implying that this consolidation process is flexible and dynamic, thereby allowing VMs to be reallocated across hosts to optimise resource utilisation and satisfy performance objectives. This idea may work in an environment with minimal hosts and where the original host migration will be quickly returned. Suppose a comparison is made to data centres like Google and Amazon. In that case, returning to the original location may only sometimes be possible due to the massive availability of servers.

Moreover, the existing live migration technology will always use random paths from different numbers of servers for migration, making it very difficult for the proposed algorithm to survive in its intended operation. Another setback must be considered; the authors have stated that "the comparisons will be done between VM on host Y and VM on host Z by the generation server to find reusable memory." Reusable memory indicates the memory pages in a VM that can be transferred or reused during migration. This part of the algorithm will compare based on any of the selected servers, but how will the VMs judge the reusable memory if the VM has more resource demand than the original memory capacity?

The other researchers in [124]–[127] have also proposed a similar idea for an ant and bee colony system, which aims to perform dynamic VM consolidation to improve the utilisation of PM resources and reduce energy consumption. Through the research [122], it has been claimed to have improved energy consumption by 37.26% and 70% of performance due to the migration of VMs.

They have developed a policy for energy awareness of RAM-consolidated VM selection. Data centre brokers and cloudlets have been created based on the utilisation values. VMs with minimal RAM utilisation are considered to have been migrated to other servers. RAM utilisation is the percentage of the total available RAM currently being used by a VM.

In order to ensure that energy and QoS are maintained, [122] has developed an algorithm that will detect the over- and under-utilisation of hosts and implement RAM consolidation with local regression techniques, which will evaluate the performance of the host when combined with the VM selection algorithm. They have redesigned the schedule for the cloudlets to get each application’s information. The RAM of each VM has been sampled and added up in each host from the VMs scheduler.

The authors in [42] proposed a novel technique for the dynamic consolidation of VMs based on the adaptive utilisation of the CPU threshold. They have used dynamic threshold utilisation, which means the system will be able to adjust based on the
behaviour of the workload sent by the users. Thus, the system utilisation threshold will be based on a history of utilisation during the lifetime of the VM operation. This technique is different from the other research, which considers the fixed-value utilisation of the CPU. This novel technique reduces energy consumption and ensures QoS for CUs. The SLA violation claims to be as low as 1%.

Due to the live migration of VMs, the authors in [42] have considered the service violation and predicted the cost of live migration to be 10%. Based on this claim, they have proposed a dynamic threshold with a heuristic algorithm that minimises the number of VM migrations. The research has also proposed a power model that considers power utilisation when the servers are idle. They claim that idle servers consume 70% of energy compared to when they are fully utilised [41].

The revised idea of fixed-threshold CPU utilisation has been discussed by different researchers, such as [49][124][128]-[131] regarding the VM consolidation issue. The proposal of dynamic threshold utilisation [42] has provided a different view that will enable better performance by adjusting the system speed based on the load given by the users. Though dynamic threshold utilisation is presented, doubt may arise about how and when the CPU is considered overloaded. It needs to be clarified what percentage of utilisation is considered the maximum and what is considered the minimum.

As the energy efficiency of multi-server consolidation becomes an issue, meeting QoS for CUs has come up with additional obstacles. The authors [27] have devised a proposal that may help evaluate server consolidation to improve power efficiency while ensuring that QoS is maintained, particularly in large data centres. The proposed model addresses the application workload and considers the effect of energy and QoS while turning on and off the servers and migrating VMs from one data centre to another. It mainly concentrates on resource usage (CPU, memory usage, and I/O activities). According to the simulation results from the two data centres, which involve 400 concurrent VMs, they claim to have reduced the energy consumption by 50% while ensuring that the QoS is maintained.

The multi-agent strategy proposed will be able to negotiate the allocation of resources in CC. Four agents have been introduced: CSP, CU, energy power provider (EPP), and carbon emission agency regulator (CEAR). The primary purpose of CEAR is to regulate the carbon emissions emitted by CSP and EEP at any given time during cloud operation.

The model can improve performance and energy consumption if the allocation of resources in each data centre is done systematically. According to the idea [27], it provided a scenario of two data centres negotiating the load allocation (DC1 and DC2); according to the scenario, DC1 is considered overloaded. The technique allowed the workload to be sent to DC1 to see if DC1 can accommodate the load; if not, it will send
it to DC2 to see if the accommodation of the workload will be taken. If not, it will be sent to DC1 again, and DC1 will look to see if resources are available. If not, DC1 will negotiate with agents, and if the load is still not accommodated, then DC1 will check the service violation to terminate the process.

The process is very long, which will consume more energy and reduce performance. There will be no good purpose for terminating the service due to an SLA violation after all the processes have failed. The designed agents are acceptable, but the technique takes a longer time, thus making it difficult to match the reality of CC. The technique may work better if the migration decision is made after the maximum point where the load is not acceptable.

2.3.5 Service Level Agreement (SLA)-Based Approaches

The other part of this literature is frameworks for SLA, which help to maintain the QoS between the CSP and cloud client. Several researchers [132]–[135] have suggested monitoring CSP concerning meeting the SLA requirement for their consumers. The authors in the [136] cloud framework paper proposed a framework that could be used to monitor the efficiency of SLAs. The research analysed SLA parameters, such as response time, job execution time, threat limits, and runtime data, to isolate the sources of SLA violations. Based on the prediction of SLA violations, implementing improvements to mitigate the violations was introduced through adaptive resource allocation using the results of SLA violations. The adaptive resource allocation system does provide cloud applications with additional computer resources, which will help reduce the SLA violation. The authors claimed that such a method could reduce the occurrence of SLA violations. Per their test results, the research claimed to produce efficient outcomes, reducing the occurrence of SLA violations and satisfying cloud clients and providers. A SLA-based workload scheduling (SLA-WS) technique [137] is designed for performing real-time workloads on multi-cloud platforms. It focuses on multi-objective parameters like energy optimisation and processing efficiency using a soft-computing-based dragonfly algorithm.

Other research by [138] proposed a framework algorithm that offered a load-balancing technique and QoS improvements among multiple servers. This framework aided in producing good response times and virtuous utilisation of cloud resources. In their methodology, they have congregated servers with different processing powers (high, medium and low) into different clusters and proposed an algorithm divided into two stages. The first stage used an SLA scheduling algorithm to determine the highest priority tasks allocated to the available server. The second stage concentrated on a monitoring algorithm for idle servers, balancing each working server's load. The algorithm selects and sorts the tasks based on priority and length. If the tasks are found to be high, medium, or low in length, they are assigned to high, medium or low...
processing power servers in a specified cluster consecutively. The algorithms were implemented and tested in a cloud simulation environment and compared with two existing algorithms (the throttled load balancing algorithm and the round-robin algorithm) by considering the waiting time, response time, and utilisation of resources as the primary performance measures. The results claimed to have improved response time, reduced waiting time and provided effective resource utilisation with better load balancing among multiple servers compared to other existing algorithms.

2.4 Gaps identified

It is well known that energy consumption and QoS delivery hinder the smooth operation of services in CC. Many researchers [139]-[141] have devised several ideas and suggestions for server consolidation to alleviate these problems. Though many researchers have done their level best to create techniques and algorithms, the problems of energy consumption and performance remain unsolved to meet the expected standards or requirements.

This report's detailed literature review and analysis revealed specific gaps that must be addressed. Some of the gaps identified are listed below:

- **CPU threshold limit:** There is no consensus among researchers regarding the optimal CPU threshold value for VM consolidation, with suggested values between 50% and 75% in several studies. This inconsistency highlights the requirement for a more definitive and universally accepted threshold value that balances VM consolidation and performance. Therefore, further research is essential for identifying the optimal CPU threshold value by considering diverse workloads and system configurations to improve both performance and consolidation efficiency [119]. The claim by the other authors of 100% utilisation [47] needs to be re-evaluated.

- **CPU and memory utilisation:** In the case of online live migration in cloud environments, previous studies have focused on CPU and memory utilisation or included network bandwidth utilisation. A detailed analysis of how network bandwidth affects performance degradation is absent, particularly in scenarios with multiple PMs. Hence, further investigation is required to develop reliable methods for assessing the performance impact of network bandwidth during live migrations involving several PMs.

- **Switching off the idle servers:** It has a negative impact on energy consumption and performance degradation. Most of the research formulated the algorithm that allows the idle server to be switched off. The research and practical test bed have proved more energy consumption and performance problems upon rebooting the system [26] [143]. It is challenging to predict the nature of cloud usage [144]. The rebooting time will depend on the availability of applications...
on the VMs. Stability after rebooting may vary from 6-12 minutes, depending on the server's capacity. Some algorithms allowed idle servers to be diverted to sleep mode; how the degradation time was measured is not specified. The algorithm mentioned that the sleep mode server might take a few seconds to a few minutes to operate fully. It may be better to create an algorithm to trigger the overloaded server for 1 to 2 minutes before it reaches maximum utilisation. This process alleviates the strain on the server and prevents potential interruptions to system performance.

- No specific server categories: No specific server categories are identified to judge when the servers become overloaded, underloaded and idle. Based on threshold or simulation tests, most research determines that the server is overloaded and underloaded. During a threshold problem, determining the appropriate threshold value is difficult. However, performing a practical experiment with different loads is imperative to predict the exact or close metrics that determine server load categories.

- Placement of VMs in PMs: Cloud providers consist of many physical servers, which have different memory capacities and CPU speeds. Again, judging which PM can be placed with the VMs that will fit with the original host capacity or higher is challenging to ensure better performance. It is necessary to consider these issues in a real cloud environment. However, the research done so far has yet to cover this area. Since the VMs will have a complete dynamic migration, an algorithm is proposed to explore the destination PMs to have the same or higher setup capacity for better performance maintenance.

- Lack of experimental data: Most of the research is done based on the available simulated data in the literature rather than using the data obtained from practical experiments. It is proposed to set up a series of experiments to obtain experimental data for initial data collection. Simulation and actual hardware setup results may differ widely in terms of power consumption and performance since simulations often depend on simplified models and assumptions for representing complex real-world scenarios and will not capture all the details present in actual hardware setups. The difference between simulation and real-time results can happen because of differences in the accuracy of input parameters and validation procedures.

- Optimise SLA compliance and energy efficiency: It is essential to balance the trade-off between meeting an appropriate SLA and minimising the energy consumption of PMs. It is imperative to identify a proper framework and integrate it with server load categories, and this helps to appropriately manage all the activities in cloud data centres, which will minimise energy consumption and improve performance and QoS for cloud consumers.
The above research [132]–[135] contributes to QoS improvements that help reduce SLA violations. However, most of the proposed techniques, algorithms, and metrics do not consider different server load categories, such as server idle, underloaded, balanced, and overloaded conditions, which are essential to balance the trade-off between meeting an appropriate SLA and minimising the energy consumption of PMs. Additionally, most of the above developments have focused primarily on simulations, with no testing in the real environments of PMs and VMs.

2.5 Summary

This chapter has focused on the state-of-the-art techniques used for CC and frameworks to enhance performance, minimise energy consumption, and provide QoS to both cloud providers and consumers. The taxonomy of the research has been designed to focus on GCC, where most of the research has been done by different scholars to tackle the problems of energy consumption, performance degradation, and QoS. Several techniques and algorithms have been reviewed regarding their strengths and limitations. After a comprehensive review of the existing state of the art, several gaps have been identified. The identified research gap has helped the author perform further research to tackle the issues still faced by service providers and cloud consumers in CC.
CHAPTER 3 METHODOLOGY

3.1 Introduction
This chapter initially discussed the problem formulated for developing the research solution. Moreover, the proposed methodologies, including the analysis of energy models, are performed using VM consolidation on practical servers with various load categories, and a novel HH-ECO approach for scientific workflow is presented along with the overall block diagram.

3.2 Problem Formulation
Cloud data centres utilise a large amount of energy; hence, improving energy efficiency is vital for reducing operational costs and environmental impact. VM consolidation plays a vital role in minimising energy consumption by optimising the utilisation of physical servers. Also, the cloud data centres maximise their hardware utilisation by consolidating VMs based on their load ‘L’ and resource requirements. Various consolidation strategies focus on analysing the energy consumption ‘E’ and performance ‘P’ under various load conditions, facilitating the designing of scalable solutions for efficiently handling the constraints in CC [145]. Energy modelling aids in tracing the energy consumption of servers under different load conditions like idle (E_{IL}), underloaded (E_{UL}), balanced (E_{BL}), and overloaded (E_{OL}), measured by energy utilisation ‘E’ of servers along with CPU performance and memory utilisation. Hence, it is essential to focus on the VM consolidation that intends to reduce the number of active servers S_{active} without compromising performance. Consequently, the design of energy-aware VM consolidation saves the energy E_{save} and thus, improves the overall energy efficiency of the data centre.

Scientific workflows in CC involve complex computations and large-scale data processing tasks that require significant energy consumption and computational resources. Optimising scientific workflows for energy efficiency while achieving satisfied performance and SLA becomes a vital task. In order to handle the trade-offs between energy consumption minimisation and performance maximisation, a metaheuristic algorithm is adopted due to its advantage of multi-objective optimisation. To solve the problem of VM allocation, task scheduling, and VM migration as a multi-objective optimisation problem, the focus of the proposed approach is to find an optimal solution for ensuring energy efficiency and performance. In particular, energy-efficient optimisation of scientific workflow in CC is based on the energy model and load categories.
The cloud environment comprises PMs with capacity $C_P$ (CPU, Memory, Bandwidth), VMs with capacity $C_V$ (CPU, Memory, Bandwidth), and tasks. The primary goal of this research work is to allocate VM resources within each PM to execute tasks efficiently. Each PM is characterised by its Makespan (MS), SLA violations, and Degree of Disparity (DD). The research work, HH-ECO operates in problem and solution spaces, in which particles direct through these spaces to find optimal solutions. This approach alleviates local convergence problems by preventing dominance and balancing exploration and exploitation. Moreover, HH-ECO targets the resource allocation and task scheduling phases by minimising execution time and DD by the advantage of heuristic algorithms, such as the C-PSO. Also, the integration of VM migration resolves the PM overloading or underloading situations.

3.3 Proposed Methodology

With the target of optimising the energy consumption in the cloud, the methodology of this thesis exemplifies the proposed HH-EECO model during the deployment on a real-time test bed and simulation environment as the two phases. The initial phase is the analysis of energy models using VM consolidation on a practical server with various load categories. It involves the real-time experiment using ESXi hypervisor for different load categories across servers. Based on the server load categories, the proposed HH-ECO is developed for the scientific workflows using C-PSO and adaptive mutation. This phase includes VM allocation, task scheduling, and VM migration. However, setting a real-time cloud computing testbed necessitates the complex and frequently intricate infrastructure that includes cutting-edge networking hardware, reliable storage systems, and high-performance servers. In addition, establishing a real-time testbed consumes high upfront costs, including those associated with purchasing technology and obtaining software licenses. In addition to the initial cost, the operation cost becomes high due to the expenses for staffing, maintenance, cooling, energy, and other services. Furthermore, scalability is a major constraint in the real-time test environment due to the arrival of large-scale dynamic workloads, which affects the scope of energy modelling in the HH-ECO without the dynamic scaling on the workload testing. The second phase of this work describes the proposed HH-ECO that is simulated in Cloudsim. Due to the complex nature of scientific workflow, the task execution consumes more energy in the cloud. Hence, the proposed HH-ECO focused on optimising scientific workflow in terms of energy efficiency and performance. Figure 3.1 shows the overall block diagram of the proposed approach. The techniques to be developed aim to reduce energy consumption by considering scheduling and VM provisioning. The aim is to ensure a better QoS by taking care of VMC issues and prioritising utilising resources (CPU and primary memory) that will satisfy cloud users.
3.4 Phase 1: Analysis of Energy Models Using VM Consolidation on Practical Server with Various Load Categories

In this phase, the actual cloud environment is designed based on the existing system that is discussed in Chapter 4. The energy models are analysed using VM consolidation with different load categories. The test is done by giving the loads to all the VMs. Based on the load given, the result is recorded and analysed in terms of CPU utilisation/performance, memory utilisation, and energy consumption.

3.4.1 VM Consolidation

VM consolidation is the process of planning to accommodate the number of VMs in a particular PM to minimise the number of PMs. This consolidation analyses the energy models under various load conditions. The setup for this experiment is done by using a hypervisor. A hypervisor is the VM monitoring management that allows multiple VMs to run on a particular PM while sharing resources such as processors, storage, memory and other related computer resources. Two types of hypervisors are commonly used: Type 1 (Bare Metal) and Type 2 (Hosted). Type 1 hypervisors, preferred for their superior performance and security, operate directly on hardware without an underlying OS. V-Sphere serves as the management console for Type 1 hypervisors, facilitating VM and resource management. ESXi, selected for the test bed, stands out for its direct installation onto PMs, robust management functionality, and flexibility in resource management. The test bed includes one high-end server and
three regular servers consolidated with VMs, each with allocated RAM and CPU based on their OS and role in the environment.

### 3.4.2 Configuration of Consolidated Server

After setting up the servers, all VMs are configured and allocated shared resources from the servers, including RAM, CPU, and storage capacity. After that, the setup criteria are established for each consolidated server. These criteria are designed to organise load types and monitor the behaviour of the consolidated servers.

### 3.4.3 Analysis of Server Load Categories

Through this testbed setup and configuration, the test was performed to determine when the servers are considered to be idle, balanced, underloaded, or overloaded. The results were taken after testing individual VMs in a server for number of hours. The results are obtained in terms of CPU, energy, and memory utilisation. This analysis paved the way for developing an energy-efficient workflow optimisation approach in the cloud environment.

### 3.5 Phase 2: Hybrid Heuristics-based Energy Efficient Optimization of Scientific Workflow in Cloud

The second phase of this work is the experiment of the proposed HH-ECO approach on the simulation environment in which the load categories are obtained from the real-time testbed experiment. The proposed HH-ECO approach aims to maintain the optimal balance between energy consumption and task performance while executing scientific workflow applications. The primary objective of this work is to improve energy efficiency by executing the non-dominant workflow tasks with the help of the meta-heuristic algorithm. In the HH-ECO, a metaheuristic approach refers to the C-PSO that is responsible for handling VM allocation, task scheduling, and VM migration. Moreover, the C-PSO with adaptive mutation leverages the energy-efficient workflow task execution based on the task precedence relationships in the workflow. Thus, the proposed approach addresses the multi-objective combinatorial optimisation problem without dominance among the workflow tasks using the hybrid heuristic algorithm in the cloud environment.

### 3.5.1 VM Allocation

To improve the energy efficiency in the cloud, optimally allocating the VMs is crucial due to the VM resource allocation relies on the potential utilisation of CPU, memory, and storage resources. Hence, to improve the overall execution performance, underutilised or overutilised hosts are to be identified. The energy consumption of cloud data centres heavily depends on the energy consumed by the cause of virtual machine allocation. Hence, the proposed HH-ECO targets the VM
allocation for reducing energy consumption while satisfying SLAs. This approach initially categorises VMs and PMs based on their capacities and workloads. By applying the C-PSO with adaptive mutation, VMs are assigned to suitable PMs with the consideration of energy efficiency, performance, and workload balancing. In the C-PSO-based VM allocation context, each particle denotes a potential VM allocation solution, whereas the velocity and position determine its VM allocation plan. The HH-ECO dynamically adjusts the inertia weight for balancing exploration and exploitation by guiding these particles through the search space in terms of available PMs. By considering resource utilisation, energy efficiency, and SLA compliance, the updated inertia weight and particle velocity-based proposed algorithm returns the fitness score for an effective VM allocation. Thus, the C-PSO-based VM allocation algorithm ensures energy-efficient VM allocation across heterogeneous PMs and VMs and facilitates energy-efficient workflow task execution.

3.5.2 Task Scheduling

In addition to the VM allocation, task scheduling plays a vital role in optimising the energy level in the cloud. In particular, task scheduling becomes essential for workflow scheduling to reduce downtime as well as optimise resource utilisation. Also, the dominance or task completion time and dependency-aware task scheduling further improve energy efficiency in the large-scale cloud environment. Accordingly, the proposed task scheduling helps to schedule the tasks on on-demand VMs depending on their workflow dependencies and task requirements. By leveraging C-PSO with the inertia weight update, HH-ECO optimises task scheduling by considering the static PM resource utilisation and dynamically estimated execution costs and makespan of the workflow tasks. The fitness value calculated by C-PSO indicates total execution cost and adherence to SLAs, targets to schedule workflow tasks on the VMs with minimum costs and makespan within deadlines. The proposed C-PSO algorithm enforces the selection of the best scheduling plan for the workflow tasks without the dominant task execution constraint, depending on chaotic sequences and the inertia weight factor. However, the dominance of individual workflow tasks causes suboptimal decisions, which is solved by the adaptive mutation in task scheduling. Thus, HH-ECO reduces makespan without violating SLAs by considering task dependencies and resource utilisation, thus ensuring efficient scientific workflow task execution.
3.5.3 VM Migration

To maintain energy efficiency during the workflow execution, the proposed approach targets the VM migration phase that relies on energy modelling based on the workload conditions of the PMs. After effectively scheduling the workflow tasks with adaptive mutation, each PM's state is monitored to categorise them as idle, overloaded, under-loaded, or balanced, in which the load categories are determined from the real-time testbed conducted in Phase 1. VM migration phase initiates when there are fluctuations in the workloads for energy optimisation during the workflow task execution. By applying the C-PSO algorithm, the proposed approach selects the destination PMs for the VMs in either underloaded or overloaded source PM based on the load balancing condition. The fitness function leverages the VM migration to high-fitness destination PMs by considering the VM's predicted capacity utilisation. During VM migration, idle PMs are switched to sleep mode to reduce energy consumption, with HH-ECO estimating the waking time point for reducing resource wastage. Thus, the HH-ECO optimises VM migration strategies, enhances cloud resource utilisation and avoids performance degradation during workflow task execution.

3.6 Summary

This chapter presented a detailed discussion of the proposed methodology concentrating on VM consolidation, configuration, categorising the load, VM allocation, task scheduling, and VM migration. Through practical experiments on cloud infrastructure, the analysis of energy models is performed using VM consolidation under different load categories like idle, underloaded, balanced and overloaded. Based on these categories, the proposed HH-ECO is implemented. The HH-ECO prioritises the workflow task based on the dominance criteria in the execution time. In the HH-ECO, the C-PSO-based energy-efficient VM allocation with adaptive mutation optimally balances the exploration and exploitation of the available cloud resources. Also, task scheduling optimises workflow execution performance by considering the workflow task dependencies and resource utilisation. Moreover, the VM migration ensures load balancing and energy efficiency during the execution of workflow tasks. The proposed methodology is briefly explained in the subsequent chapters.
CHAPTER 4 ANALYSIS OF ENERGY MODELS USING VM CONSOLIDATION ON PRACTICAL SERVERS WITH VARIOUS LOAD CATEGORIES

4.1 Introduction
This chapter discusses the designing of the practical experiment using a high-end server and standard servers with several VMs to analyse energy utilisation and consolidation. This experiment intends to analyse and pinpoint consolidation problems, load balancing, and energy consumption in virtual machines and physical server environments. Through this experiment, the investigation of different loads from virtual machines will be analysed. The server load categories and specific parameters will be identified and analysed after monitoring the server behaviour at each test stage. The experimental setup validates various load categories for servers to evaluate their performance and energy consumption under various conditions. Potential results of CPU, memory, and power consumption for each server category are also presented.

4.1.1 Virtualisation Using ESXi and V-Sphere
The virtualisation platforms used were operated through VMware’s suite of virtualisation products. So, VMware and its virtualisation solutions, such as vSphere and ESXi, are explained. VMware is specialised in virtualisation technology. VMware’s ESX server hypervisor enabled the consolidation of physical servers into VMs, thus minimising the number of physical servers required. VMware’s vSphere platform is extensively utilised for virtualisation. ESXi of VMware is a type-1 (bare-metal) hypervisor, whereas vCenter is the central management hub used for resource and VM management. The distributed resource scheduler balances the resource utilisation by migrating VMs between different hosts. vMotion facilitates live migration with no downtime. Storage vMotion transfers the VM files between datastores for balancing the utilisation. HA technology in VMware automatically resumes VMs after host failures. VMware’s vSphere utilises virtual networking elements such as port groups and vSwitches. The current platform of the client comprises three ESXi hosts, a storage array, a virtual vCenter server, and networking components, for migrating to a new platform with 15 ESXi hosts. Some popular alternatives are Hyper-V, Xen, Proxmox VE, and Citrix Hypervisor. Among these, vSphere and ESXi are recognised for their performance and scalability, which can efficiently manage large numbers of VMs and
workloads across distributed environments. VMware offers updates and technical support, which is vital for reliable virtualisation infrastructure management.

4.1.2 System Model

This section discusses a practical experiment considering VMC in high-end and standard servers. ESXi hypervisor will be configured as a layer that helps to manage and monitor multiple numbers of VMs in all PMs. The bare-metal hypervisor will be set up in one high-end server with 17 VMs and three regular servers with 7 VMs. The number of servers and VMs is selected based on practical constraints like available hardware resources and the specific requirements of the applications and workloads being tested. To interact with all the VMs by providing all the setup and adjustments within the hypervisor, the setup will use a V-sphere. V-centre will also be configured to help manage and monitor all ESXi hosts, including the VMs.

The whole setup of ESXi hosts (PMs) will be configured within the Active Directory (AD) environment. AD is a domain name in the Microsoft server’s operating system that acts as the central database that connects all the clients’ computers to the server. Dynamic Host Configuration Protocol (DHCP) server will also be configured within ESXi hosts to manage the clients easily. When the clients reboot, DHCP automatically distributes the Internet Protocol (IP) addresses to client computers. All the servers within this setup will be installed with a Windows server which will provide AD and DHCP full functionality, and all client computers will be installed with different operating systems (clients and server). Mixing multiple operating systems with the ESXi host refers to running different operating systems concurrently on virtual machines (VMs) hosted on the ESXi hypervisor. This approach offers flexibility in adjusting the memory and CPU core sizes allocated to each VM based on the system’s graphic user interface (GUI). Essentially, each operating system may have varying resource requirements depending on its workload and usage patterns. By running multiple operating systems simultaneously, administrators can dynamically allocate resources such as memory and CPU cores to each VM as needed through the ESXi host’s GUI. It allows for efficient resource management, ensuring that each VM receives the necessary resources to operate optimally while maximising the utilisation of the underlying hardware.

Each VM in the ESXi host will be allotted a specific amount of RAM (memory capacity) to allow each VM to function properly. The memory size will be allotted based on the capacity of each VM. Two or three GB of RAM will be allotted for a server VM, and for a client VM, one or two GB will be allotted. Server VMs will typically require more capacity than clients since they are used to link clients and are configured with DHCP for proper management of clients. A server VM hosts server applications or infrastructure components, like web servers, database servers, and so on.
designed to offer backend functionality and services to client systems. On the other hand, a client VM emulates a client system like a computer or mobile device. Client VMs are utilised for running the client applications, accessing services offered by server VMs, and executing user-centric tasks.

4.2 VM Consolidation Setup

As discussed in Chapter 1, VMs consolidation is done by using a hypervisor. Most organisations and IT industries around the world use two types of hypervisors. The type of hypervisors is as follows:

4.2.1 Type 1 Hypervisor (Bare metal)

According to IBM and other leading ICT industries, Bare metal hypervisor provides better performance, security and availability than type 2 (Hosted) since it does not contain any OS at its base [146]. Type 1 Hypervisor is a type of hypervisor that is installed directly to computer hardware (i.e. server) without any Operating System (OS). A Type 1 hypervisor does not have a GUI to guide the user on a configuration of VMs. It needs additional software to act as a management console for all the VMs. In this setup, we used a V-sphere to interact with the hypervisor to manage and maintain VMs and needed resources such as processors, storage, and memory. The full explanation of V-sphere management will be discussed in another section of this chapter.

Figure 4-1 Type 1 hypervisor (Bare Metal)

Figure 4-1 shows a type 1 hypervisor (Bare Metal). It has three layers; the first layer, in light blue colour, is a hardware layer. This layer does not have any operating system in it. It should be an empty computer with enough capacity to handle the type 1 hypervisor to be installed. The second layer in light grey is a hypervisor layer, where
an organisation or an individual chooses the type of hypervisor to be installed on empty hardware. The hypervisor will be installed directly on the first layer. The third layer in dark orange is the VM layer. Any operating system can be installed in each VM, and they can run simultaneously. Multiple VMs can be installed with operating systems based on the server’s capacity. In order to manage all the VMs in a type 1 hypervisor, a management console software such as V-Sphere is needed.

### 4.2.2 Type 2 Hypervisor (Hosted)

Type 2 hypervisor is the type of hypervisor that is installed on top of the OS. Most type 2 hypervisors use an OS application to control and manage the VMs compared to a type 1 hypervisor, which requires additional software to act as a VMs management console. An operating system must be installed before installing type 2 hypervisors.

![Figure 4-2 Type 2 Hypervisor (Hosted)](image)

Figure 4-2 shows a type 2 hypervisor. It has four layers compared to the three layers found in the type 1 hypervisor. The first layer in light blue is a hardware layer; this layer will need to be installed with any operating system compared to the first layer of type 1 hypervisor, where no operating system is needed. The second layer in light grey is the operating system layer, which will need to be installed in layer one. The third layer in light green is a hypervisor layer. A type of hypervisor for type 2 will be installed in an operating system in layer two. Layer two will be the base for running all the hypervisors compared to the type 1 hypervisor, whereby layer one will be the base for running the hypervisor and all the VMs. The fourth layer in dark orange is the VM layer, where the VMs will be configured and operating systems installed on each VM. This is different from type 1 hypervisor. The VMs in the type two hypervisor are monitored and managed by layer three through layer two.
Each VM within a Type 1 hypervisor can run its own OS concurrently, managed through V-Sphere. The layers of a Type 1 hypervisor setup are the hardware layer, hypervisor layer, and VM layer. Type 2 hypervisors are installed on top of an operating system (OS) and rely on OS applications for VM management. Unlike Type 1 hypervisors, which require separate management software, Type 2 hypervisors necessitate an OS installation before deployment. Comparing type 1 and type 2 hypervisors, they provide the same functionality of running the same kind of VMs, which can easily be moved from a system that runs type 1 or 2 hypervisors. However, Type 1 Hypervisor is used for this thesis to set up a test bed that will provide a platform for all VMs. The reason for selecting Type 1 hypervisor is because it provides enhanced performance and greater flexibility. Since it runs directly on hardware without OS, it provides a thin layer that helps to represent the entire hardware resources to VMs which helps to reduce the overhead that is needed to run the hypervisor itself. The type 1 hypervisor provides better security as no known OS runs below its layer [146], and it is very difficult for intruders to penetrate the systems. Type 1 hypervisors run directly on the physical hardware without any underlying OS, thus eliminating the potential vulnerabilities caused by having an extra layer (called the host OS) between the hardware and the hypervisor. As there is no traditional OS running below the hypervisor layer, the attack surfaces and points of entry for potential intruders will be considerably reduced. Hence, it is intrinsically more difficult for attackers to enter the systems. Most importantly, the type 1 hypervisor supports hardware virtualisation compared to the type 2 hypervisor, which supports software virtualisation [146]. It becomes very important and suitable for this test bed as it is intended to perform consolidation of VMs and test the hardware used to check performance power consumption and quality of service.

Many enterprises offer type 1 hypervisors to data centres around the world; however, the best well-known are as follows [147]:

- **KVM (Kernel-Based Virtual Machine):** This is a Linux open source-based hypervisor that supports most Linux operating systems such as Red Hat Enterprise, SUSE and Ubuntu. In addition to supporting Linux, KVM supports Solaris and Windows as well. It has an additional management tool that helps manage the VM operation of VMs.

- **Microsoft Hyper-V Server 2019:** This is a Microsoft virtualised platform that provides a cost-free solution, offering enterprise-grade virtualisation capabilities for hybrid cloud environments and data centres. It has a range of improved features developed to satisfy the performance requirements and scalability of the most crucial workloads.

- **Xen/Citrix Xen Server:** This type 1 hypervisor uses the Xen server for commercial purposes, and it is one of the most popular open-source hypervisors. Xen has many features such as management of power and
• memory, migration of storage, monitoring and alerting all the functions in a server environment, Xen management console to manage all the resources including VMs and PMs, and live migration.

• Red Hat Enterprise Virtualization: This is a type 1 hypervisor which is open source; it is built on Red Hat Enterprise Linux and KVM. It is a commercial implementation of KVM type 1 hypervisor. It offers similar features as other types one hypervisors.

• ESXi Hypervisor: This is one of the most robust types of hypervisor that consists of various versions from ESXi 3.5 to the current release of ESXi 8.0a. ESXi provides features such as VM replication, host profiles, auto deploy, V-center single sign-in, V-center operation manager, web-based management console, C-center virtual appliance deployment, distributed resource scheduler, load balancing, power management alert, and many other features. ESXi has free and commercial versions, widely used by many industries. A free version can offer up to 32 GB of RAM.

In this thesis, the ESXi hypervisor is used for the test bed. Using multiple hypervisors would increase the complexity and necessitate additional resources. The ESXi hypervisor is selected because its leading industrial base metal is directly installed into a PM without an underlying operating system, reducing the footprint and security threats. It has robust management functionality built into VMKernel and offers extensive VMs capabilities [148]. ESXi provides much flexibility, allowing users to efficiently manage the resources through its robust management console, which tolerates clone and pasting VMs with the same or different setup requirements [150].

The setup for testing includes one high-end server and three regular servers (
Table 4-1 & Table 4-2), which are consolidated with several VMs. Each VM in all four servers is installed with an operating system (Windows Server and Windows clients) and consolidated with an amount of RAM and a CPU. VMs installed with the servers’ OS are given higher RAM (2 to 3 GB). The client and member servers are given 1 GB in high-end servers. VMs in regular servers (3 servers) have been consolidated with 2 GB of RAM in the server environment and 1 GB in the client environment. Allotting more RAM capacity to servers than client computers is because server computers are configured with active directories and DHCP to connect, assign, delegate and distribute IP addresses automatically to client computers and overload the PMs. Thus, an additional amount of RAM is required to function correctly. Client computers in this test will only be used for overloading and contribute to overall loads of PMs. They do not have any active directory or DHCP.
Table 4-1 Setup Requirement

<table>
<thead>
<tr>
<th>Description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Setup for high-end server</strong></td>
<td></td>
</tr>
<tr>
<td>Manufacturer</td>
<td>IBM</td>
</tr>
<tr>
<td>Model:</td>
<td>IBM System x3500 M4 Server -[738325Z]-</td>
</tr>
<tr>
<td>Processors:</td>
<td>4 CPU x 2.399 GHz</td>
</tr>
<tr>
<td>Processor type:</td>
<td>Intel(R) Xeon(R) CPU E5-2609 0 @ 2.40GHz</td>
</tr>
<tr>
<td>H/Disk</td>
<td>Capacity: 551.50 GB</td>
</tr>
<tr>
<td>Hyper-threading:</td>
<td>Inactive</td>
</tr>
<tr>
<td>Total memory:</td>
<td>36 GB</td>
</tr>
<tr>
<td>Number of NICs:</td>
<td>5</td>
</tr>
<tr>
<td>Hypervisor</td>
<td>ESXi 5.5</td>
</tr>
<tr>
<td>State:</td>
<td>Connected</td>
</tr>
<tr>
<td>VMs:</td>
<td>17</td>
</tr>
</tbody>
</table>

Table 4-1 shows a setup requirement for a high-end server. It shows all the descriptions for a setup. This server has been configured with 17 VMs directly linked through an active directory and automatically assigned IP addresses by a DHCP server. Since the server has enough memory capacity and CPU cores, 17 VMs have been configured to load the PM. The server can accommodate 17 PMs and more without any problem.

Table 4-2 Normal Server (3 servers)

<table>
<thead>
<tr>
<th>Description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Setup for normal server</strong></td>
<td></td>
</tr>
<tr>
<td>Manufacturer</td>
<td>Lenovo</td>
</tr>
<tr>
<td>Processors:</td>
<td>2.0 GHz Intel Core i7-4765T Processor</td>
</tr>
<tr>
<td>H/Disk</td>
<td>250 GB H/Disk</td>
</tr>
<tr>
<td>Hyper-threading:</td>
<td>Inactive</td>
</tr>
<tr>
<td>Total memory:</td>
<td>8GB + 4 1600MHz DDR3 RAM</td>
</tr>
<tr>
<td>Number of NICs:</td>
<td>1</td>
</tr>
<tr>
<td>State:</td>
<td>Connected</td>
</tr>
<tr>
<td>VMs:</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 4-2 shows a setup requirement for three regular servers. It shows all the descriptions for a setup. The servers initially had 8 GB of RAM; due to the test requirement, they were upgraded to 12 GB to accommodate more VMs. 7 VMs were consolidated in each server.
Figure 4-3 Testbed architecture (ESXi 5.5, VSphere Client and V-center) for 4 ESXi hosts. Adapted from [151]

Figure 4-3 shows a test bed architecture which consists of four servers (one high-end server and three regular servers) This is an architecture that is specifically designed for the test bed of this experiment. All of the servers are configured with ESXi 5.5 hypervisors. The entire architecture is linked to V-Center, deployed on top of ESXi 5.5 as a VM. V-centre is deployed to provide a central management interface to manage all four ESXi hosts of this experiment in Figure 4-3. The reason for deploying V-center is that ESXi Graphic User Interface (GUI) can only manage one single host at a time and does not allow supervision and multiple hosts on a network. V-Centre provides
advanced features like VM cloning, which has helped to add multiple VMs on the four EXSi without installing the new OS every time a new VM is added.

<table>
<thead>
<tr>
<th>APPLICATION</th>
<th>APPLICATION</th>
<th>APPLICATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM 1 Active Directory Server</td>
<td>VM 2 DHCP Server</td>
<td>VM 3 V-CENTER Member Server</td>
</tr>
<tr>
<td>VM 4 Member</td>
<td>VM 5 Member</td>
<td>VM 6 Member</td>
</tr>
<tr>
<td>VM 7 Member</td>
<td>VM 8 Member</td>
<td>VM 9 Member</td>
</tr>
<tr>
<td>VM 10 Member</td>
<td>VM 11 Member</td>
<td>VM 12 Member</td>
</tr>
<tr>
<td>VM 13 Member</td>
<td>VM 14 Member</td>
<td>VM 15 Member</td>
</tr>
<tr>
<td>VM 16 Member</td>
<td>VM 17 Member</td>
<td></td>
</tr>
</tbody>
</table>

HARDWARE (IBM System x3500 M4 Server - [738325Z])

<table>
<thead>
<tr>
<th>CPU</th>
<th>Memory</th>
<th>Storage</th>
<th>NIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.40GHz</td>
<td>36 GB</td>
<td>552 GB</td>
<td>5</td>
</tr>
</tbody>
</table>

Figure 4-4 Cloud test bed setup on high-end server. Adapted from [161]

Figure 4-4 shows the system requirements and the setup on a high-end server for our implementation. The framework has been set up using a type 1 “bare metal” hypervisor. It consists of 17 VMs, which are directly configured in an active directory (AD) environment. An AD is a domain network developed by Microsoft to link all the clients and member servers to one or more domain names. It allows the clients and other member servers to be easily monitored and managed. The above setup also uses the DHCP (Dynamic Host Configuration Protocol) server as one of the VMs. The DHCP server is used in an AD environment to automatically distribute all the IP addresses to all VM clients and member servers. It ensures all VMs are connected to the networks, thus assisting in better testing all VMs using different loads. On the same setup, VM 3 is set up as V-Center, a package feature that comes with VMware ESXi type 1 hypervisor. As explained in the architecture (figure 4-3), V-Center is a centralised management utility that can easily manage all VMs in a particular PM or all ESXi hosts in a cloud environment from one location. The remaining VMs (14) are configured as clients and member servers and linked directly to one domain.

After setting up the servers (1 high-end and three regular servers), all VMs are
configured and given the shared resources from servers, such as RAM (Random Access Memory), CPU (Central Processing Unit) and storage capacity. The next stage is to arrange setup criteria for each consolidated server. This criterion is arranged to plan load types to observe consolidated servers’ behaviour. Error! Reference source not found. and Table 4-4 have illustrated high-end and standard server setup criteria for performance behaviour observation. Performance behaviour observation will depend on the type of load given to the VM, as per the table below. Based on the load given, the experiment is done to determine when the PMs are Idle, underloaded, balanced or overloaded. The loads are determined on the basis of changing levels of workload intensity and resource utilisation. The specific requirements of each workload decide the memory of the VM. The load given are categorised as follows.

i. **Low load**: This involves playing 3D video on all of the VMs and opening multiple applications on some of the VMs.

ii. **Medium load**: This involves playing 3D video on some VMs, disk defragmentation on some VMs, and opening multiple applications on all VMs.

iii. **Intensive load**: This involves Playing 3D video on all VMs, disk defragmentation on all VMs, opening multiple applications on all VMs and scanning all VMs by antivirus. The physical memory of the system is inadequate for accommodating all active processes, thereby causing the swapping of data between disk storage and physical memory for managing memory demands efficiently.

<table>
<thead>
<tr>
<th>VM Name</th>
<th>OS installed</th>
<th>Status/Purpose</th>
<th>Load given</th>
<th>Load type</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM 1 active directory server</td>
<td>Windows Server</td>
<td>Active Directory with DNS controller. All other VMs except DHCP are joined to VM1 as client and member servers.</td>
<td>Low, medium and intensive (refer to Table 4-5 Load Category and Test).</td>
<td>3D video, antivirus full scan and disk defragment (refer to Table 4-5 Load Category and Test).</td>
<td>Low and medium loads showed stable operation but reduced performance in medium loads. Intensive load showed sluggishness by slowing the operation of the VM.</td>
</tr>
<tr>
<td>VM 2 DHCP Server</td>
<td>Windows Server</td>
<td>Active Directory, DNS and DHCP server. To distribute IP addresses automatically.</td>
<td>Low, medium and intensive</td>
<td>3D video, antivirus full scan and disk defragment</td>
<td>Same as above</td>
</tr>
<tr>
<td>------------------</td>
<td>----------------</td>
<td>---------------------------------------------------------------------------------</td>
<td>--------------------------</td>
<td>-------------------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>VM 3 V-CENTER Member Server</td>
<td>Windows Server</td>
<td>Member server of VM1 (active directory) controls all VMs.</td>
<td>Low, medium and intensive</td>
<td>3D video, antivirus full scan and disk defragment</td>
<td>Same as above</td>
</tr>
<tr>
<td>VM 4 Member</td>
<td>Windows Client</td>
<td>Member client of VM1 (active directory)</td>
<td>Low, medium and intensive</td>
<td>Antivirus full scan and disk defragment</td>
<td>Low load showed average performance and medium load degraded the speed of the VM. Intensive load results in shutting down the VM.</td>
</tr>
<tr>
<td>VM 5 Member</td>
<td>Windows Client</td>
<td>Member client of VM1 (active directory)</td>
<td>Low, medium and intensive</td>
<td>Antivirus full scan and disk defragment</td>
<td>Same as above</td>
</tr>
<tr>
<td>VM 6 Member</td>
<td>Windows Server</td>
<td>Member server of VM1 (active directory)</td>
<td>Low, medium and intensive</td>
<td>3D video, antivirus full scan and disk defragment</td>
<td>Same as VM 1</td>
</tr>
<tr>
<td>VM 7 Member</td>
<td>Windows Client</td>
<td>Member client of VM1 (active directory)</td>
<td>Low, medium and intensive</td>
<td>Antivirus full scan and disk defragment</td>
<td>Same as VM 4</td>
</tr>
<tr>
<td>VM 8 Member</td>
<td>Windows Client</td>
<td>Member client of VM1 (active directory)</td>
<td>Low, medium and intensive</td>
<td>Antivirus full scan and disk defragment</td>
<td>Same as VM 4</td>
</tr>
<tr>
<td>VM 9 Member</td>
<td>Windows Client</td>
<td>Member client of VM1 (active directory)</td>
<td>Low, medium and intensive</td>
<td>Antivirus full scan and disk defragment</td>
<td>Same as VM 4</td>
</tr>
<tr>
<td>VM 10</td>
<td>Member</td>
<td>Windows Client</td>
<td>Member client of VM1 (active directory)</td>
<td>Low, medium and intensive</td>
<td>Antivirus full scan and disk defragment</td>
</tr>
<tr>
<td>-----</td>
<td>--------</td>
<td>----------------</td>
<td>------------------------------------------</td>
<td>--------------------------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>VM 11</td>
<td>Member</td>
<td>Windows Client</td>
<td>Member client of VM1 (active directory)</td>
<td>Low, medium and intensive</td>
<td>Antivirus full scan and disk defragment</td>
</tr>
<tr>
<td>VM 12</td>
<td>Member</td>
<td>Windows Client</td>
<td>Member client of VM1 (active directory)</td>
<td>Low, medium and intensive</td>
<td>Antivirus full scan and disk defragment</td>
</tr>
<tr>
<td>VM 13</td>
<td>Member</td>
<td>Windows Client</td>
<td>Client (not a member of any server)</td>
<td>Low, medium and intensive</td>
<td>Antivirus full scan and disk defragment</td>
</tr>
<tr>
<td>VM 14</td>
<td>Member</td>
<td>Windows Client</td>
<td>Member client of VM1 (active directory)</td>
<td>Low, medium and intensive</td>
<td>Antivirus full scan and disk defragment</td>
</tr>
<tr>
<td>VM 15</td>
<td>Member</td>
<td>Windows Client</td>
<td>Client (not a member of any server)</td>
<td>Low, medium and intensive</td>
<td>Antivirus full scan and disk defragment</td>
</tr>
<tr>
<td>VM 16</td>
<td>Member</td>
<td>Windows Client</td>
<td>Member client of VM1 (active directory)</td>
<td>Low, medium and intensive</td>
<td>Antivirus full scan and disk defragment</td>
</tr>
<tr>
<td>VM 17</td>
<td>Member</td>
<td>Windows Client</td>
<td>Member client of VM1 (active directory)</td>
<td>Low, medium and intensive</td>
<td>Antivirus full scan and disk defragment</td>
</tr>
</tbody>
</table>
Figure 4-5 shows the system requirements and the setup on a regular server for this experiment. Three regular servers are linked to AD, and each server is configured with seven VMs, including clients, servers and member servers. In each setup, there is a DHCP server configuration which helps to distribute IP addresses automatically. As explained in Figure 4-1, all three servers are managed by V-Center together with a high-end server in Figure 4-4.

Similar to Figure 4-4, All VMs are configured and given the shared resources from servers, such as RAM, CPU and storage capacity. The next stage is to arrange setup
criteria for each of the three consolidated servers. This criterion is arranged to plan load types to observe consolidated servers’ behaviour. The actual test on performance, energy consumption and QoS is available in Chapter 4.

Table 4-4 Normal servers (three servers) setup criteria and server behaviour observation

<table>
<thead>
<tr>
<th>VM Name</th>
<th>OS installed</th>
<th>Status/Purpose</th>
<th>Load given</th>
<th>Load type</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM 1 Active directory server</td>
<td>Windows Server</td>
<td>Active Directory with DNS controller. /All other VMs except DHCP are joined to VM1 as clients to the server</td>
<td>Low, medium and intensive</td>
<td>3D video, antivirus full scan and disk defragment</td>
<td>Low and medium loads showed stable operation but reduced performance in medium loads. Intensive load showed the sluggish operation of the VM.</td>
</tr>
<tr>
<td>VM 2 DHCP Server</td>
<td>Windows Server</td>
<td>Active Directory, DNS and DHCP server. To distribute IP addresses automatically</td>
<td>Low, medium and intensive</td>
<td>3D video, antivirus full scan and disk defragment</td>
<td>Same as VM1</td>
</tr>
<tr>
<td>VM 3 V-CENTER Member server</td>
<td>Windows Server</td>
<td>The member server of VM1 (active directory) Acts as a controller for all VMs.</td>
<td>Low, medium and intensive</td>
<td>3D video, antivirus full scan and disk defragment</td>
<td>Same as VM1</td>
</tr>
<tr>
<td>VM 4 Member</td>
<td>Windows Client</td>
<td>Member client of VM1 (active directory)</td>
<td>Low, medium and intensive</td>
<td>Antivirus full scan and disk defragment</td>
<td>Low load showed typical performance, and medium load degraded the speed of the VM. Intensive load results in shutting down the VM.</td>
</tr>
<tr>
<td>VM 5 Member</td>
<td>Windows Client</td>
<td>Member client of VM1 (active directory)</td>
<td>Low, medium and intensive</td>
<td>Antivirus full scan and disk defragment</td>
<td>Same as above</td>
</tr>
<tr>
<td>VM 6 Member</td>
<td>Windows Server</td>
<td>Member server of VM1 (active directory)</td>
<td>intensive</td>
<td>defragment</td>
<td>Same as VM 1</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------</td>
<td>-----------------------------------------</td>
<td>-----------</td>
<td>-----------</td>
<td>--------------</td>
</tr>
<tr>
<td>Low, medium and intensive</td>
<td>3D video, antivirus full scan and disk defragment</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VM 7 Member</td>
<td>Windows Client</td>
<td>Member client of VM1 (active directory)</td>
<td>Low, medium and intensive</td>
<td>Antivirus full scan and disk defragment</td>
<td>Same as VM 4</td>
</tr>
</tbody>
</table>
4.3 Energy Model

Based on the above setup and test on real PMs, it is practical to think of the components that make up CC. The components that contribute to the consumption of energy and performance are[42][50]:

i. The PMs, which include the CPU, memory (RAM), motherboard, and network interface card

ii. Networking components such as switches, routers and ports included within the switch and routers

iii. Storages, which include disk arrays and storage controllers

The above are reasonable due to the many existing research reviewed and published. Some researchers [42] consider the CPU to be one of the more essential components, which may cause a massive deal in consuming energy. At the same time, some others [50][57] have also considered memory to be one of the critical components for the consumption of energy and performance. Since CC is not about consuming energy alone, it becomes imperative to ensure that consumers of cloud services are satisfied with the service. As that is the case, the primary memory of any data centre should be another vital component that will positively contribute to a better QoS by improving performance and lower energy consumption.

Research has considered other network components, such as switches, routers and ports included within the routers and switches [50], contributing to energy consumption and performance degradation in the cloud environment. However, the percentage of utilisation of these components is tiny compared to CPU and memory [50]. In the proposed research of this thesis, we will be using the equations, which will include the physical nodes with CPU and memory utilisation as follows:

Equations (4.1) and (4.2) check CPU and memory usage while all the VMs are switched ON in a PM within the cloud environment. Our experiment considers CC to consist of PMs consolidated with VMs (adapted from [113]).

\[
G_{CC} = G_{PMserver} + VM_{Usage} \tag{4.1}
\]

\[
G_{PMserver} = G_{CPUusage} + G_{Memoryusage} \tag{4.2}
\]

As that is the case, all VMs will compete to utilise CPU and Memory, which will help to judge the utilisation of energy consumption and performance. \(G_{CC}\) stands for global usage of CC, \(G_{PMserver}\) stands for Global usage of PMs server, \(VM_{Usage}\) stands for usage of VMs, \(G_{CPUusage}\) stands for Global usage of CPU and \(G_{Memoryusage}\) stands for global usage of memory.
Based on the literature search [83], efforts are made in this thesis to improve techniques and heuristic algorithms that will ensure energy efficiency and the assurance of the QoS to the cloud users. After considering the power consumption components, the next step is to design an algorithm to control CPU and memory utilisation. As mentioned above, memory and CPU usage alone will be considered. Most of the research [94] has claimed to have achieved improvements in terms of saving energy and assurance of the quality of service. For example, techniques like workload consolidation and dynamic resource allocation aim to optimise QoS and lessen energy consumption in data centres. However, most of the research is either based on theoretical concepts from the gaps identified by other researchers or in the simulation setup. The results might contribute to better consumption and performance, but proving the result is challenging since actual experiments were not involved. Although simulation setups [89] are very helpful in designing sophisticated cloud environments and policy implementation, testing the energy consumption and the need for QoS may not be efficient since the setup environment is not real.

At this stage, this thesis includes the design of the actual cloud environment based on the existing system (refer to Error! Reference source not found.). Figure 4-2 and Figure 4-3). The test is done by giving the loads to all the VMs (refer to Error! Reference source not found. & Table 4-4). Based on the load given, the result is recorded, analysed and reported in Chapter 4.

4.4 Testing
Load balancing is an essential technique in utilising the resources in CC. The aim and objectives of chapter four are to test load balancing to reduce power/energy consumption and improve performance and QoS in the cloud environment based on the test bed. The testing methodology, criterion and rationale for server classification are presented along with the results. The main objective is to test and validate the utilisation of the following.

i. CPU utilisation/performance:
ii. Memory Utilisation
iii. Power Consumptions

Through this test-bed setup and configuration, the test was done to determine when the servers are considered to be idle, underloaded or overloaded. Many researchers have attempted these three categories using simulations to achieve load balance in CC [117][122][152]. Though many other researchers have used Idle and overloaded alone or underloaded and overloaded, it is imperative to consider both categories to achieve a better load balancing which will help develop a better technique to migrate VMs from one PM to another. An underloaded PM is not fully utilised but has some level of resource consumption. In contrast, an overloaded PM exceeds its resource...
capacity, causing degraded performance and potential resource contention. Due to the test in Table 4.5, another category was named balanced, which falls between underloaded and overloaded server categories. The balanced category is crucial since it will help to avoid unnecessary usage of the resources in the cloud systems. It allows the underloaded virtual machines to be transferred to it to become idle and switch to power-saving mode. It will also benefit the overloaded servers as it may allow some of the VMs to be transferred to balance and leave the overloaded server to become balanced with a reasonable load. Introducing the balance category is one of our initial contributions based on the existing literature gap and the experiment performed in our test bed.

Table 4-5 presents the load categories and tests performed. It also explained how the results were taken and the conditions that were considered. The test in Table 4.5 was carried out by switching ON all the VMs and sending the load simultaneously. The load sent was done for many hours to check the system’s stability in both categories. Based on the test, all these loads have proved to cause resource activities contributing to performance degradation and energy consumption. Below are the types of loads given which were used for this study at Caledonian (University) College of Engineering:

1. Playing 3D video on all VMs: Since the videos were played on all VMs continuously, a promising result on performance degradation and energy consumption was observed to be true. Though there are several ways of stressing the server, such as MAXON and Heavy Load, the 3D video was found to be suitable. This is because the failure of MAXON [153] and Heavy Load [154] to run on the VMs recommends that they are not appropriate for stress testing in this specific virtualised environment. Instead, playing 3D videos in all VMs is considered a more effective method for monitoring performance degradation and energy consumption, likely due to its compatibility and lower resource requirements compared to the other stress testing tools.

2. Scanning using antivirus on all VMs: Scanning all VMs with antivirus proves suitable since it utilises system resources [155] such as CPU and RAM. It becomes evident and indisputable that continuous scanning can cause a burden on performance [156]

3. Disk defragments deal with fragmented files, which cause excessive utilisation of resources (CPU and RAM) [157]. As that is the case, disk defragmentation can cause multiple activities to all the VMs, which will again help to overload the entire system.

4. Opening multiple applications on all the VMs: Opening multiple applications or processes, such as audio and video, can cause performance degradation since they compete for shared resources [158]. This again proves to be another
method that will contribute to overloading all the VMs within the server.

Table 4-5 Load Category and Test

<table>
<thead>
<tr>
<th>Sl No.</th>
<th>Load Category</th>
<th>Load Test</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Idle server with all VMs on</td>
<td>All VMs switched on, and no load was given to VMs</td>
<td>Test outcomes were recorded after all the VMs were stable from booting.</td>
</tr>
<tr>
<td>2</td>
<td>Under loaded server</td>
<td>All VMs switched on, and the following tests were done:</td>
<td>Test outcomes were recorded based on the load test. The test was done to check if the behaviour of VMs and servers changed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1. Playing 3D video on all of the VMs</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. Multiple applications were open on some of the VMs</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Balanced server</td>
<td>1. Playing 3D video on 14 VMs</td>
<td>The test was repeated to observe the behaviour of VMs and servers. Test outcomes were recorded.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. Disk defragmentation on 15 VMs</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3. Multiple applications on all VMs</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Overloaded server</td>
<td>1. Playing 3D video on all VMs</td>
<td>Test observed and repeated. Test outcomes were recorded.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. Disk defragmentation on all VMs</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>3. Multiple applications on all VMs</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>4. Scanning of all VMs by antivirus</td>
<td></td>
</tr>
</tbody>
</table>

4.5 Result and analysis

Based on Table 4-5, the test outcomes were recorded, and the analysis will be performed based on the results obtained. The results were obtained after testing individual VMs in a server for about an hour. They are based on the utilisation of the CPU and memory, which shows the percentage of CPU usage, memory usage and power consumption in all of the four categories as follows.
4.5.1 Idle Server Utilization (with all VMs on)

4.5.1.1 CPU utilisation (Idle server with all VMs ON)

The CPU and memory utilisation percentages recorded over an hour for individual VMs on a server reflect the usage of virtual CPU and memory resources in a virtualised cloud environment. Utilisation may rise because of higher resource contention, workload demands, or ineffective resource management. Higher utilisation specifies more CPU cycles and memory required since the tasks increase within VMs. Resource limitations or inappropriate configuration can lead to increasing load over time. In Figure 4-6, the scale in MHz represents the CPU’s speed, providing CPU utilisation percentages by representing its maximum capacity. Even though data collection might have continued for 150 minutes, trends can be inferred from the observed data. Further analysis is needed to infer the trends beyond 150 minutes by considering variability in workload. The CPU utilisation shown in Error! Reference source not found. corresponds to the idle case with all the VMs switched ON and no activity after rebooting. The percentage utilisation at the end of the test is below 25%. Some amount of activity is seen because all 17 VMs were switched ON. As such, the average CPU utilisation is much below 25%.
**4.5.1.2 Memory Utilisation (Idle server with all VMs ON)**

![Memory Performance (Idle)](image)

Figure 4-7 shows the performance in terms of memory utilisation for the idle case as a function of Time, representing the amount of memory used by the system. In this case, the consumed memory utilisation includes the activity in VMs and also due to the switching on of the VMs. If any applications such as audio, video and so on are opened to any of the VMs, the utilisation increases. As for the idle server, the observed memory utilisation is in line with the results obtained in [159] that encompass various aspects of memory utilisation in idle server scenarios. This comparison helps validate the reliability and consistency of the current study’s results and provides insights into the behaviour of memory utilisation in idle server environments.
4.5.1.3 Power Consumption (Idle server with all VMs ON)

Figure 4-8 Idle Power consumption with 17 VMs ON

Figure 4-8 depicts the power consumption (Watts) as a function of Time. While considering the power consumption at full utilisation to be 150-160 Watts, the average power consumption for this idle case falls in the band of 60 to 75%. Different types of servers or machines have varying power consumption characteristics, based on the factors like hardware specifications, workload categories, and their configurations. For testing purposes, the power consumption of 150-160 Watts is considered. The results obtained here corroborate the claims by other researchers that an idle server will consume between 50 to 70 % of power as compared to full utilisation [26][42][50]. Even when the server is considered idle, there are often background processes or system tasks running within the VMs. These processes may periodically consume additional power, causing temporary spikes in power consumption. Hence, the blip up at the end was observed after 2 hours, however, the result was almost the same without further increase.
4.5.2 Underloaded Server Utilisation

4.5.2.1 CPU utilisation (Under loaded server)

Figure 4-9 Under loaded CPU utilisation
An underloaded server means that all VMs are not loaded. Some VMs are given loads, while the remaining are not. The CPU performance of this case is shown in Figure 4-9 as a function of Time. This case’s maximum Percentage CPU utilisation at the end of the test is 40%, the average performance being about half of the maximum. Since all VMs are not loaded, the maximum performance is not expected.

4.5.2.2 Memory Utilisation (Under-loaded server)

Figure 4-10 Under loaded Memory utilisation
An underloaded server means that only some VMs are loaded while others are not. The memory performance for the underloaded server is shown in Figure 4-10 as a function of Time. The minimum memory utilisation is at 13%, while the maximum memory utilisation is close to 40%. As such, the memory utilisation is nearly half of its capacity compared to the idle case.

4.5.2.3 Power Consumption (Underloaded server)

Figure 4-11 shows the power consumption as a function of Time for the underloaded server. The power consumption pattern for this case is quite different from that of an idle server, as shown earlier in Figure 4-8. The activities and their patterns are different for these two cases. However, only about a 20% increase in power consumption is seen in the underloaded server compared to the idle server case. This increase is in line with the results obtained by other researchers [26][41][42].

![Power consumption(Underloaded)](image-url)
4.5.3 Balanced Server Utilisation

4.5.3.1 CPU utilisation (Balanced server)

A balanced server case is introduced between underloaded and overloaded server categories. CPU performance for the balanced server case is shown in Figure 4-12 as a function of Time. It is observed that while VMs are working during the test time, the maximum utilisation is only around 63%. A margin of 37% is still available. This implies that some more VMs can be added without causing any degradation in performance. As such, some VMs can be transferred to this balanced server from under-loaded or over-loaded servers. Therefore, we have proposed an algorithm that will allow the underloaded or overloaded server to transfer the VMs to the balanced server to allow the under-loaded servers to become idle and the overloaded server to be balanced consecutively.
4.5.3.2 Memory Utilisation (Balanced server)

Figure 4-13 Balanced Memory Utilization

Figure 4-13 shows the utilisation of memory for the balanced server case. The maximum percentage of memory utilisation is around 58%, which is higher by 15% compared to under loaded server case. A balanced server can handle more memory utilisation to alleviate the problems with both underloaded and overloaded servers. It avoids unnecessary wastage of resources and thus allows the server to become well-balanced by accepting the migration of VMs from other servers. (Under loaded or overloaded).
4.5.3.3 Power Consumption (Balanced server)

The Power consumption for the balanced server case as a function of Time is shown in Figure 4-14. As expected, power consumption is marginally increased by 10 to 20% compared to Idle and underloaded server cases. It enables some more VMs to be transferred from an underloaded server to a balanced server and become idle without causing any load problems, keeping the overall power consumption well below the threshold of more than 75% of server utilisation. In addition, some of the idle servers can be either switched off or put into sleeping mode.

![Power consumption(Balanced)](image)
4.5.4 Overloaded Server Utilisation

4.5.4.1 CPU utilisation (Overloaded Server Utilisation)

The CPU performance for the case of an overloaded server is shown in Figure 4-15. It is seen that all CPUs are busy with hectic activities, starting from 58% to a maximum of up to 80%. More than 80% load will result in slower speed and inefficient operation of the CPU. As such, the plot depicts that PM is overloaded, and hence, some VMs may fail to operate [160]. It may be noted that some important data centres use a strategy of 50% CPU utilisation to enable the best performance of VMs migration [54]. 25% of the remaining utilisation is reserved to avoid performance degradation. However, 70% - 75% is still good enough as the stability of server performance is very convincing [51]. The CPU takes so long to reach stability because of workload variability and is influenced by the dynamic and complex nature of virtualised environments.
4.5.4.2 Memory Utilisation (Overloaded Server Utilisation)

Figure 4-16 shows the memory performance as a function of Time in the case of an overloaded server. The memory consumption from the beginning is relatively high and near constant, with an average value of around 75%. When the server’s memory is heavily burdened, it will cause slow response times and decreased efficiency. Due to this, the performance becomes sluggish and unsatisfactory. In line with CPU performance for an overloaded server, as shown in Figure 4-15, a need will arise to migrate some of the CPUs (VMs) to another type of server, preferably a balanced server.
4.5.4.3 Power Consumption (Overloaded Server Utilisation)

Figure 4-17 shows the power consumption vs Time for the case of an overloaded server. It is seen that the power consumption is very high, starting at 100 Watts and reaching about 160 Watts peak during the time VMs are busy with several activities. The power consumption of around 150-160 Watts obtained above is relatively high, resulting in higher energy wastage in the target server. Hence the need for migration of VMs arises here.

Table 4-6

<table>
<thead>
<tr>
<th>Categorisation of Server Behavior</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Overloaded</td>
<td>The server becomes overloaded (CPU and memory utilisation) and consumes maximum power. Some VMs can be transferred to underloaded, balanced or idle servers. (&gt; 75%)</td>
</tr>
<tr>
<td>2 Under loaded</td>
<td>The server has minimum tasks to perform. It can receive a load from an overloaded or send it to the balanced server so that some VMs are left in Idle mode. (15%-40%)</td>
</tr>
<tr>
<td>3 Balanced</td>
<td>The server is neither underloaded nor overloaded or idle. It has a partial workload and may have space to accommodate VMs from overloaded or underloaded servers (41%-60%).</td>
</tr>
<tr>
<td>4 Idle</td>
<td>The server is without any load. It can either receive a load from overloaded or be turned to power saving mode (5%-15%).</td>
</tr>
</tbody>
</table>
4.6 Server Categorisation Flow Chart

Based on the above analysis, a flow chart is prepared to migrate VMs to other servers which have enough resources. The flow chart is shown in Figure 4-18 Algorithm Flowchart:

4.6.1 Working of the Algorithms

The algorithm in Figure 4-18 verifies the utilisation based on CPU and primary memory usage, providing comprehensive energy consumption and performance results. The
server behaviour classification is based on whether the servers become overloaded, underloaded, balanced or idle. The condition has been formulated based on the test bed results achieved. The steps are explained below:

(i) Servers know the other server’s state using their utilisation percentages. When a server becomes overloaded, the algorithm searches for balanced or underloaded servers to perform VM migrations. If no servers are found, idle servers will be activated. The algorithm verifies whether the utilisation percentage is between 73% to 75%. If the condition becomes true, it will check if balanced or underloaded servers can accommodate other VMs from the overloaded servers. As that is the case, the algorithm will scan the availability of balanced and underloaded servers to migrate the VMs to avoid the servers becoming overloaded. If there are no balanced and underloaded servers, the idle server will be triggered to wake up from sleeping mode. Here, the idle servers are not turned off to avoid the delay when the server is rebooted, which may depend on the server’s capacity and applications. The reason for triggering at 73% is to enable the idle server to be alert to wake up from sleeping mode before it becomes fully loaded (>75%). It allows the incoming VMs to be migrated with better performance and low energy consumption.

(ii) The algorithm verifies if the percentage of utilisation is within 41%-60%. If the condition becomes true, the server will become balanced; in that case, the server is neither underloaded nor overloaded. The reason for this balance is that there will be a wastage of energy and resources; to overcome this problem, the algorithm will always ensure that the server will be utilised with a very reasonable utilisation of all the resources. If the server becomes balanced, it should receive other VMs from overloaded or underloaded servers. The reason for receiving VMs from overloaded servers is to free some space from overloaded servers to enable the server to operate in a stable condition (not overloaded). Moreover, the reason for receiving the VMs from underloaded servers is to allow underloaded servers to become idle and be put into power-saving mode. Thus, the energy and quality of service will be improved.

(iii) Underloaded servers decide whether to wait based on utilisation percentage (15% to 40%) or offload VMs. If the condition becomes true, then the server will become underloaded. If it is underloaded, it should receive other VMs from overloaded servers or transfer VMs to balanced servers and become idle. The reason for receiving VMs from overloaded servers is to free some space so that they can operate within a stable condition. Furthermore, the reason for transferring the VMs to balanced servers is to enable the balanced servers to become balanced and leave the underloaded server idle. If it becomes idle, it will automatically be put in power-saving mode. All of these conditions will be
scanning the availability of server classification at each stage to migrate the VM to a proper place with enough resources. When there are two underloaded servers, they will not be offloaded to each other. Instead, they will prioritise receiving VMs from overloaded servers or transferring VMs to balanced servers to maintain balance and efficiency.

(iv) The algorithm verifies if the percentage of utilisation is between 5%-15%. If it is, then the server will become idle. Furthermore, if idle, it should receive VMs from an overloaded server. If not, the servers should be put in sleep mode. It should be noted that the waking up of sleeping mode condition will only be triggered before it reaches the maximum overloading condition. An idle server waits for some time to check for VMs before going to sleep, indicating the server enters sleep mode only when it will not receive any VMs from an overloaded server and its utilisation is within the given range. The idle servers are triggered to wake up from sleep mode before the other servers reach maximum overloading, thus assisting in load balancing.

4.7 Summary
This chapter discussed a practical design of a test bed used to configure a real high-end server and regular servers for energy efficiency and consolidation. A type 1 Hypervisor (Bare metal) manages and monitors multiple VMs on PMs. Consolidating VMs into PMs helps study better utilisation of resources through load balancing of PMs. Setup criteria and server behaviour observation are validated through the test bed to observe the server’s behaviour after overloading with different load types. All four categories (idle, underloaded, balanced, and overloaded) were identified in the initial testing, and tests were done for 1 hour. The CPU, Memory, and power consumption performance were plotted for each category. Based on the results and analysis, migration of VMs from underloaded or overloaded servers to balanced servers was proposed, subject to the availability of resources. In order to mitigate the problem of inefficient resource utilisation and energy consumption identified during the initial testing of the CC infrastructure, the next phase proposes a unique optimisation technique, which is expected to balance power consumption and performance optimisation in cloud data centres and users for executing the scientific workflow. The next phase is planned to utilise the cloud simulation tool for evaluation as the real-time experiments require substantial computational resources and extensive hardware, which are not feasible for study purposes.
CHAPTER 5 HYBRID HEURISTICS-BASED ENERGY-EFFICIENT OPTIMIZATION OF SCIENTIFIC WORKFLOW IN CLOUD

This chapter describes the overall procedure of the Hybrid Heuristic algorithm-based Energy-efficient cloud Computing service (HH-ECO) approach that balances the energy consumption and performance of the system while executing the scientific workflow applications. This proposed approach is developed based on the load categories obtained using practical experiments (discussed in Chapter 4). The proposed HH-ECO approach is mainly designed for scientific workflow applications. For the execution of scientific workflow tasks, the HH-ECO approach performs the VM allocation, scheduling, and VM migration using the Chaotic Particle Swarm Optimization (C-PSO) algorithm. The rest of the chapter is organised as follows. This chapter begins with the outline of the HH-ECO approach in section 5.1. The preliminaries are discussed in section 5.2, and the system model of the proposed HH-ECO approach in section 5.3. Section 5.4 defines the problem formulation, which describes the issues in CC and key measures of the proposed work to enhance its performance. Sections 5.4-5.6 display the procedure of VM allocation, workflow task scheduling, and VM migration with the adaptive migration. Finally, section 5.7 summarises the HH-ECO approach and the block diagram of that approach.

5.1 Outline of HH-ECO Approach

Owing to the complex nature of scientific workflow, energy consumption is high for executing the workflow in the cloud environment. Hence, the proposed approach aims to improve energy efficiency by focusing on resource allocation, task scheduling, and resource migration. The proposed HH-ECO algorithm is accompanied by the C-PSO algorithm that dynamically handles the scientific workflow tasks based on resource availability. Resource allocation with energy management by considering the SLAs is a significant research issue in CC because of the dynamicity, scalability, performance uncertainties, and heterogeneous application workloads that demand changing resource requirements [162]. Energy-efficient management of virtualized resources ensures the QoS constraints of the end-user by improving the energy efficiency of operations like efficient VM allocation [163], task scheduling [164][33] and resource
migration. In high-performance computing environments, scientific workflow applications’ QoS requirements include ensuring efficient resource utilization, energy conservation, and performance, considering task characteristics, scheduling the task execution, and accomplishing cost-effectiveness and optimal energy consumption.

In the cloud environment, idle PMs consume a substantial quantity of resources (that is, 40% to 70% of their peak load power), which have been addressed by several cloud implementations by switching OFF the idle machines to conserve energy [165]. However, energy consumption is higher while rebooting PMs because of the time taken to stabilize all the applications after rebooting. So, maintaining the trade-off between performance and energy consumption is often challenging while offering the end users the best cloud service. Several researchers have recommended that changing a physical server into sleep mode will improve energy efficiency as it takes minimum time to stabilize all the applications [166] compared to the idle state. On the other hand, overloaded PMs significantly weaken the quality of cloud services and reduce the execution speed, primarily due to the over-utilization of the available resources on a specific PM. Therefore, optimally allocating the virtual resources to PM and migrating overloaded VMs to the best hosts are essential to optimize resource utilization [57] significantly. Owing to the dynamic nature of the cloud, VMs accept dynamic workloads while running applications that demand highly intensive computation. Hence, the utilization of the CPU varies with time, necessitating the dynamic allocation to optimally utilize the cloud service in terms of accomplishing energy efficiency and SLA. For the energy-efficient execution of scientific workflow tasks, the proposed HH-ECO approach aims to enhance the VM allocation, task scheduling, and VM migration phases in the cloud.

5.1.1 Objectives of HH-ECO Approach

The overall objectives of the HH-ECO approach are

- To address the multi-objective combinatorial optimization problem without dominance amid the workflow tasks using the hybrid heuristic algorithm in the cloud environment.

- To perform energy-efficient resource allocation, task scheduling, and VM migration using C-PSO with adaptive mutation.

The detailed objectives of the HH-ECO approach are

- To prevent premature convergence by improving the convergence rate of the PSO algorithm with the chaotic procedure during multi-objective optimization in a dynamic cloud environment.

- To improve the global-optimum solution in the search space by controlling the dominance area of one task over others during resource allocation and task
scheduling.

- To build an energy-efficient virtualized environment with high reliability and efficiency by the optimal resource allocation using a hybrid heuristic workflow algorithm.

- To effectively perform the scheduling of workflow tasks even when there are dominance tasks using C-PSO with an adaptive mutation in a green cloud environment.

- To develop globally optimal migration plans using C-PSO with adaptive mutation to optimize makespan and lessen costs.

5.1.2 Contributions

Due to the aggressiveness of CC infrastructure installation worldwide, complex scientific operations and workflow applications have become data-intensive and computation-intensive. Since CC is emerging as the widely utilized distributed computing environment that supports the execution of complex and sizeable scientific workflow applications, it becomes imperative for the proposed system model to consider the complexity of scientific workflow in a cloud environment. The scientific workflows are briefly discussed in the following section. While executing scientific workflow applications, effective resource utilization is crucial to the success of robust problem-solving environments in high-performance computing. The traditional CC approaches [71] do not consider the dominant characteristics of scientific workflow tasks, which intricate the execution of interdependent tasks that demand substantially varying resource requirements. Thus, the violations of the assured quality of cloud services to the end-user have significantly increased in the scientific workflow. It is essential to focus on the orchestration of the workflow tasks execution on the VMs with the consideration of the dominant characteristics. By leveraging these considerations, QoS must be ensured at a reasonable cost for the end users while sustaining the optimal energy consumption of the resources.

To handle the dynamic and large-scale cloud environment, conventional research works have applied population-based meta-heuristic algorithms, such as PSO, to improve efficiency. Although PSO is beneficial for combinatorial and multi-objective optimization in the absence of gradient information, it frequently confronts premature convergence. Without this problem, it is difficult to increase the convergence rate of PSO, particularly for the multi-objective optimization problem. Hence, to resolve this shortcoming, this work targets adopting the chaotic sequences during the implementation of the PSO algorithm, thereby ensuring global convergence for multi-objective cloud workflow tasks. In particular, this work contributes to the C-PSO impact on resource allocation, task scheduling, and VM migration with the concept of adaptive mutation for the execution of workflow tasks with the awareness
of the dominance problem.

5.1.3 Process Involved

This thesis proposes a Hybrid Heuristic algorithm-based energy-efficient CC service, HH-ECO approach. HH-ECO adopts C-PSO [167], which accelerates the optimisation of resource allocation, task scheduling, and resource migration by generating global best plans without local convergence. It accomplishes efficient resource management and energy optimisation. Particle Swarm Optimisation (PSO) is an intelligent swarming technique that helps to obtain a feasible solution using the position and the velocity [168]. Although, it comprises several drawbacks, including premature convergence and a slow convergence rate. Hence, the proposed C-PSO algorithm-based energy-efficient approach attempts to tackle the dominance issue among the workflow tasks while scheduling and migrating the workflow tasks due to the ergodicity and disciplinarian characteristics of the chaotic method. The chaotic Particle Swarm Optimisation (C-PSO) ensures the exploration and exploitation of the particle by improving its searching behaviour and avoiding its premature convergence. In HH-ECO, an enhanced C-PSO algorithm with an adaptive mutation solution controls the dominance area of one task to the other tasks. It avoids the deterioration of global optimum performance. It ensures the appropriate resource allocation plan to reduce the Makespan and, thus, ensure a green cloud environment. HH-ECO employs the C-PSO algorithm [167] with significant enhancements to optimise energy and non-dominant execution among the workflow tasks during the execution of workflow tasks in the dynamic cloud environment, which addresses the problem of stagnation. An enhanced C-PSO algorithm with an adaptive mutation mechanism facilitates solving ongoing problems and meets dynamic requests with a rapid convergence rate and less computation effort. Additionally, HH-ECO fine-tunes the scheduling results by globally exploring the cloud resources with the help of a self-adaptive global optimisation technique of C-PSO with the adaptive mutation.

5.2 Preliminaries

This section explains the fundamental concepts related to the proposed methodology. It helps to gain basic insights into the proposed HH-ECO approach.

5.2.1 Scientific Workflow

Cloud applications typically include intricate processes that are described across several steps, each of these steps having a workflow task. These tasks are interconnected and may include various services within the cloud application [34]. On the other hand, a standalone task in a cloud application can operate independently
and is not dependent on other tasks within the application. The comparison of these tasks is summarized in Table 5.1. These tasks are self-sufficient and can be executed without particular dependencies or sequences. These workflows are mainly applied in domains including science (bioinformatics, data mining, high-energy physics, astronomy, and neuroscience) and business (forecasting, economics, and oil exploration). In scientific fields, cloud services are mainly used to store, retrieve, and execute experiments [169].

Table 5-1 Comparison of the standalone task, workflow task, and scientific workflow

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Standalone Task</th>
<th>Workflow Task</th>
<th>Scientific Workflow Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>Definition</td>
<td>A task in a structured process with a series of steps to attain a specific goal.</td>
<td>A sequence of tasks organized in a structured manner within the workflow</td>
<td>Each task within the scientific workflow represents a distinct computational operation aimed at attaining a scientific goal</td>
</tr>
<tr>
<td>Task Dependency</td>
<td>Not dependent on other tasks</td>
<td>Dependent on other tasks</td>
<td>Dependent on other tasks</td>
</tr>
<tr>
<td>Flexibility</td>
<td>More flexible due to its independence</td>
<td>Less flexible since variations in one task may impact other tasks.</td>
<td>Less flexible due to task dependency and execution within a larger process</td>
</tr>
<tr>
<td>Cloud Resource Requirement</td>
<td>Minimal computing resources</td>
<td>Significant computing resources</td>
<td>Highly complex and needs more computing resources</td>
</tr>
<tr>
<td>Cloud Resource Utilization</td>
<td>Low</td>
<td>Moderate</td>
<td>High</td>
</tr>
<tr>
<td>Execution in Cloud</td>
<td>Executed independently</td>
<td>Based on task dependencies and the workflow's design</td>
<td>Based on scientific protocol, complex computation</td>
</tr>
<tr>
<td>Execution pattern</td>
<td>Linear sequence</td>
<td>Sequential when the completion of one task initiates the next task in the workflow</td>
<td>Parallel execution during large datasets or complex simulations.</td>
</tr>
<tr>
<td>Optimization</td>
<td>Fine-tuning</td>
<td>Dependency</td>
<td>Workflow</td>
</tr>
<tr>
<td>Parameters, algorithm optimization</td>
<td>Management</td>
<td>Optimization algorithms</td>
<td></td>
</tr>
<tr>
<td>------------------------------------</td>
<td>------------</td>
<td>------------------------</td>
<td></td>
</tr>
<tr>
<td>Application</td>
<td>Science (bioinformatics, data mining, high-energy physics, astronomy, and neuroscience) and business (forecasting, economics, and oil exploration)</td>
<td>Climate science, astrophysics simulations, medical imaging, ecological modelling, bioinformatics</td>
<td></td>
</tr>
</tbody>
</table>

Existing scientific workflow models are of two main classes [169], including script-like systems and graphical systems. In script-like systems, workflow descriptions are specified by a textual programming language that is defined by grammar in a similar way to conventional programming languages. In a graphical-based system, workflows are specified as graphical elements that are parallel to the graph components, like nodes and edges. Graphical-based systems are easier and more intuitive for the user than script-based systems due to their graphical representation. Some of the popular scientific workflow management systems are Taverna, Triana, Pegasus, Kepler, Askalon, Weka4WS, General Workflow Execution Service (GWES), DVega, Karajan, and DIS3GNO [169].

The workflow splits complex and data-intensive applications into small tasks executed serially or parallelly on the basis of the application’s nature. The workflow application’s graphical representation is in the form of a DAG [170] by demonstrating the dependencies among these tasks. In DAG, each node represents the computational tasks, whereas directed edges indicate the dependencies between them. In high-performance computing, effective resource utilization during the execution of scientific workflow applications is crucial for robust problem-solving environments. However, traditional CC methods often manage the unique features of scientific workflow tasks. These tasks include executing interdependent tasks with different resource requirements, causing challenges in maintaining the promised quality of cloud services to end-users. Consequently, scheduling the execution of workflow tasks on VMs by considering their main characteristics becomes essential. By using these considerations, QoS must be assured for end-users at a reasonable cost by optimizing power consumption [171]. Scientific workflow tasks find applications across various domains of scientific research, facilitating complex data analysis and computations in fields such as climate science [172], astrophysics simulations [173], medical imaging [174], ecological modelling [175], and bioinformatics [176]. Thus, with the emergence of scientific workflow applications in the cloud-based environment, building an energy-efficient cloud service solution is essential, which is
accomplished by the enhancement of VM allocation, task scheduling, and VM migration phases in this research work.

5.2.2 Dominance Problem

Dominance problems arise among workflow tasks when certain tasks in a workflow excessively impact the execution time or resource allocation of the whole workflow. It causes difficulties in managing and optimizing workflows. Dominant tasks utilize a majority of available resources, including memory, CPU, or storage, affecting the execution of other tasks in the workflow in terms of increasing the waiting time for succeeding tasks. As a result of this, increased delay, high makespan, and even task failures may occur due to high waiting time. When few of the tasks dominate any other tasks in workflow during execution, they are early prioritized over others during scheduling, resulting the increased delay. Also, the dominance problem causes the inefficient utilization of resources. It is challenging in cloud environments in which multiple users share resources. Also, dominance issues negatively impact the QoS, as delays in a few tasks can impact the overall workflow outcomes, thereby causing violations of SLAs. To overcome this dominance problem, non-dominant execution is implemented in workflow optimization. Instead of considering a single objective like reducing execution time or improving resource utilization, non-dominant execution simultaneously considers multiple objectives like energy consumption, cost, and performance, thus making balanced decisions.

5.2.3 Metaheuristic Algorithms

Metaheuristic algorithms can efficiently navigate this complexity for identifying optimal or near-optimal configurations. Moreover, it can be scaled for handling large and dynamic server environments, adapting to variations in real-time, which is vital for maintaining energy efficiency in fluctuating conditions. It is briefly discussed in the following sections. Hence, it is applied in the proposed approach for attaining energy efficiency and performance. Prominent metaheuristic algorithms include genetic algorithms [177], PSO [73], ACO [178], simulated annealing [179], and tabu search [180]. These algorithms are extensively implemented in domains like business, engineering, and computer science to solve intricate challenges. Among these algorithms, a PSO-based algorithm called C-PSO is used in the proposed methodology due to its simplicity, scalability, faster convergence, and global optimization. Server environments encompass several variables and constraints like changing workloads, dynamic resource allocation, and varied energy consumption patterns.
5.2.3.1 PSO

Among various heuristic algorithms, PSO is one of the most adaptable techniques for complex and multi-objective optimization problems because of its potential advantage of optimal balance between exploration and exploitation. Moreover, PSO [73] is one of the most commonly used metaheuristic algorithms that mimic the bird flocks’ swarm behaviour for doing their tasks and discovering an optimal solution depending on an objective function. PSO algorithm is simple and easy to implement with fewer parameters. It can attain a faster convergence. PSO has been used in several domains, such as electric power systems, route planning, wireless sensor networks, and robotics. However, PSO’s performance still has space for enhancement. The performance of the PSO is improved by adjusting the parameters and procedure of the PSO and optimizing the particle’s update strategy.

In the PSO, Each individual in the swarm recognizes its optimal location stored as its local best position ($P_{\text{best}}$). Moreover, the best location exposed by the entire swarm ($P_{\text{best}}$) is kept in the global best position ($G_{\text{best}}$). These values are continuously updated at every step. Then, the position and velocity of every individual in the swarm are adapted by estimating the current individual velocity and the distances to the $G_{\text{best}}$ and $P_{\text{best}}$ positions. The velocities and distances of the agents from the $P_{\text{best}}$ and $G_{\text{best}}$ locations are modified accordingly. An appropriate choice of inertia weight ($W_1$) can help in balancing between exploring the local and global spaces.

$$W_1 = W_{\text{mx}} - \left(\frac{W_{\text{mx}} - W_{\text{mn}}}{m_{x_i}}\right) \cdot I$$

(5.1)

Where $W_{\text{mx}}$ and $W_{\text{mn}}$ indicate the upper and lower value of weight, respectively, I and $m_{x_i}$ indicate the current iteration and maximum iteration, respectively.

5.2.3.2 Modelling PSO in Cloud

In a cloud environment, a particle in the PSO algorithm refers to a potential solution for a given problem. In the proposed work, resource allocation, task scheduling, and VM migration are considered as the problem. A particle is a vector used to encode a set of parameters (like CPU allocation, memory usage, network bandwidth) for a particular solution within the problem space. The modelling of a particle includes representing the configurations that control the allocation and management of cloud resources. A particle's position indicates its current solution (VM allocations or schedule of tasks) within the search space. The fitness function measures the quality of a particle’s position, that is, the effectiveness of the configuration. A particle’s velocity indicates how its parameters move over iterations toward better solutions based on $P_{\text{best}}$ and $G_{\text{best}}$. $G_{\text{best}}$ helps all particles to move towards the optimal solution (optimal resource allocation, optimal task scheduling or optimal VM migration). The particles work together to explore the search space in order to converge on optimal
or near-optimal solutions over iterations. During workflow scheduling, the number of tasks is denoted by the dimension of the particles. Each particle’s position denotes a mapping between the VMs and tasks.

### 5.2.3.3 C-PSO

The proposed C-PSO method uses the chaos search technique that replaces the random initialization and perturbation in traditional PSO with chaos-based initialization and perturbation. Chaos search employs random movement with elements of pseudorandomness, ergodicity, and regularity governed by a deterministic equation. Through chaos iteration, a series of random sequences with ergodicity and pseudorandomness are produced. The logistic mapping equation is used to create pseudorandom sequences. Chaos initialization involves the process of a chaotic variable from the logistic map, which selects an initial particle value at random. Another study outlined the logistic map equation for the hybrid C-PSO algorithm.

\[
\gamma^{k+1} = \delta \gamma^k (1 - \gamma^k), \quad 0 \leq \gamma^1 \leq 1
\]  

Where \( k \) is the number of iterations, and the control parameter \( \delta \) set within a range from 0.0 to 4.0. The magnitude of \( \delta \) decides whether \( \gamma \) stabilizes at a constant area, oscillates within restricted limits, or behaves chaotically in an unpredictable form. The new inertia weight factor (\( W_2 \)) was calculated by multiplying the \( W_1 \) and the logistic map [181].

\[
W_2 = W_1 \times \gamma^{k+1}
\]  

To enhance the behaviour of the simple PSO, this work presents a novel velocity update by blending the inertia weight factor \( W_{PSO} \) with the logistic map equation (\( \gamma \)).

In the updating process of particles, their velocities and positions will be perturbed sufficiently and the search space will be traversed as sufficient as possible. In the initialization phase, C-PSO optimizes the initial particles according to the characteristics of combination optimization problems. Via item classification, similar items are grouped into the same category, thus reducing the number of combinations. Therefore, it is possible to enumerate all combination schemes and improve the search efficiency. In the chaos perturbing phase, a new set of perturbing rules is designed to perturb the velocities and positions of particles sufficiently, so that C-PSO has good global search capability and adaptability, and the premature convergence problem of particles is also effectively solved. In the above two phases, C-PSO controls the number of selected items in each category to ensure the diversity of the final combination scheme. The fitness function of C-PSO utilizes the concept of personalized constraints and general constraints to get a personalized interface, which can be used to solve the corresponding personalized combinatorial optimization
problem. In the task of resource allocation, workflow scheduling, and migration, the chaotic behaviour in the PSO enables the particles to escape from local optima easily. By randomly varying the search direction over time, C-PSO increases the likelihood of locating the global optimum in the search space.

5.2.3.4 Hybrid-Heuristic Algorithm

A hybrid heuristic algorithm in the context of CC refers to a combination of different heuristic techniques used to solve multi-interdependent task execution optimization problems related to resource allocation, scheduling, load balancing, or other challenges within a CC environment for providing effective and efficient solutions and improving resource management in the cloud. In this work, the hybrid algorithm is C-PSO [167] for optimizing resource allocation, task scheduling and VM migration through best global solutions without local convergence. By combining various heuristic approaches and leveraging the strengths of C-PSO and adaptive mutation, HH-ECO provides an energy-efficient and performance-optimized CC service. C-PSO is used, being a variant of PSO, which introduces chaotic sequences into the optimization process, which enhances the exploration capability of the algorithm. It allows C-PSO to effectively search through complex solution spaces, making it suitable for scientific workflows that often involve high-dimensional and nonlinear optimization problems. Chaotic sequences help diversify the search process, preventing the algorithm from getting stuck in local optima. By promoting exploration, C-PSO can more effectively converge to near-optimal solutions in scientific workflows, even in challenging optimization landscapes.

5.2.3.5 Adaptive Mutation

In PSO, particles are significantly affected by their own preceding best positions as well as the global best particle. If the global best particle is fixed at a local optimum, other particles rapidly converge to their location. By changing the search space around the global best particle in every iteration, the global best particle avoids getting stuck in local minima. Hence, the adaptive mutation makes the global best particles help other particles towards the best positions. Even though this method mutates the global best particle in each iteration, controlling the mutation size becomes difficult. It is accomplished by adjusting the mutation size parameters based on the objective function [182].

In PSO, a random mutation process depends on an adaptive mutation probability and distribution functions. Adaptive mutation can be performed by dynamically changing the mutation size depending on the present search space. This adaptive mutation mechanism improves the global search ability of PSO and fastens the convergence. When a particle is chosen for mutation, a Gaussian random disturbance is added to its
present position. This disturbance consists of a variable step size that reduces dynamically based on the fitness of the present best solution. Adaptive mutation aids in maintaining diversity in the solution search space, thus avoiding premature convergence [183]. In the proposed system, the adaptive mutation intends to improve energy efficiency, which is associated with task scheduling and VM migration in the cloud environment.

5.3 System Model

Based on the initial investigations that have been done in Chapter 4, the categorisation of the servers with their parameters was identified as (idle, underloaded, balanced and overloaded). Through the categorisations, it is proposed to have a system model and algorithms that will help to tackle the problem of performance, energy consumption and carbon emission in CC environments.

This section provides a system setup for executing the proposed HH-ECO approach over the cloud environment, including VM allocation, workflow task scheduling, and migration. Let the virtualised cloud server consist of a set of m physical computing hosts defined by \( H = \{ \text{PM}1, \text{PM}2, \ldots, \text{PM}m \} \). For the \( i \)th PM \((i = 1, 2, \ldots, m)\), the processing Capability \((C_i)\) is defined as the CPU performance in Million Instruction Per Second (MIPS), which varies for each PM. Let \( n_i \) be the number of VMs within any given \( \text{PM}_i \) having different capacity, i.e., \( \text{PM}_i = \{ \text{VM}_1, \text{VM}_2, \ldots, \text{VM}_{n_i} \} \). Each VM\(_{ji}\) has the processing capability \( C_{ji} \) that is subject to \( \sum_{j=1}^{n_i} (C_{ji}) \leq C_i \). There are two states for each PM\(_i\): active, i.e., APM\(_i\) when \( C_i > 15\% \), and idle, i.e., IPM\(_i\) when \( 5\% \leq C_i \leq 15\% \). These values are decided from real-time practical experiment focusing on server performance on a practical high-end server.

5.3.1 PSO-based Workflow Scheduling

Three main steps involved in PSO are computing the fitness of every particle, computing local as well as the global best fit, and updating the position and velocity of each particle, which are iterated until the stopping criteria are met [184]. PSO algorithm optimises the execution of task-resource mapping according to its principles. Initially, the heuristic model computes the mapping for all the tasks without considering the workflow dependencies to optimise the overall computation cost of the workflow application. PSO-based mapping is used to map the tasks into resources in a workflow application. In PSO, tasks are considered as particles, whereas the resources are considered as the available PMs. The mapping helps to assign tasks to the most appropriate resources for optimizing the performance and overall computation cost of the workflow application. In the PSO-based task-resource mapping, the scheduling heuristic validates the dependencies (relationship) among the workflow tasks by confirming that the tasks are executed on the appropriate
resources in the proper order. When a task is based on the completion of another task, then the scheduling heuristic considers these dependencies while mapping tasks to resources. In subsequence, it imposes these scheduling steps until all the workflow tasks are completed [185]. In the first step, the PSO algorithm randomly initialises the position and velocity of the particles. In the context of workflow scheduling, the task-resource mapping process represents the particles, and the number of workflow tasks represents the dimension of the particles. By performing the fitness function, the evaluation of each particle is computed. In the computational method, PSO identifies the solution in the region of a given problem. By applying a simple mathematical formula over the position and velocity of particles, PSO moves these particles in the region and identifies the best position based on the suitable PM selection for the workflow task from the possible placement.

5.3.2 C-PSO-based Workflow Scheduling

In PSO, the placement of workflow tasks to available VMs is represented by a sequence of possible mappings. This random sequence of task placement aids in simulating different potential solutions that help in analysing and making decisions for reducing computation time. By updating their velocity, the random sequence utilized in PSO influences the movement of the particles that refer to the possible solutions for task-resource mapping. This sequence helps the algorithm explore diverse regions of the solution space and make changes to particle positions depending on their performance. Chaos is considered a state of a dynamic system with random yet bounded behaviour. Chaotic sequences provide a form of randomness with regularity that fastens the generation of new sequences and enhances the performance of the algorithm [169].

Unlike the typical PSO algorithm, Chaotic-based PSO (C-PSO) [186] improves global convergence and its ability to determine the optimal solution. C-PSO is performed by using chaotic system-generated sequences instead of using purely random sequences as in the PSO algorithm. This replacement offers a more efficient and robust search procedure, offering better results. In the scheduling model, C-PSO exploits the chaotic sequence for initialisation and updation, which differs from the PSO algorithm. Incorporating chaotic sequences, the C-PSO algorithm [187] determines a better scheduling plan and attains minimum time and cost than the PSO algorithm. It is assumed that all the PMs and VMs are in idle state (s).

In the proposed system, resource allocation is a prerequisite task before the workflow scheduling. Initially, the C-PSO algorithm allows for the optimal allocation of VMs to PMs during ‘s’ when there is no load allocated. It utilizes the processing capability (Ci) of every PM to help in making the allocation decisions. C-PSO can effectively explore potential allocations and find the best match between PMs and VMs by utilizing chaotic sequences, ensuring efficient resource utilization even during idle state. The value of Ci in terms of MIPS can be varied under different VM load conditions. Since Ci can change depending on the load conditions and other factors, it will not be a fixed value. For example, when a VM requires more CPU resources, Ci for
other VMs in that PM will reduce, causing a reduction in $C_i$. An example is given in Figure 5.1, where the scientific workflow application is represented as a DAG model $G = \{T, E_T\}$ [93][188]. In Error! Reference source not found., where $T$ defines a set of tasks, the processes in the execution order are named tasks, and $E_T$ refers to a set of dependencies between Tasks $\in T$ of an application. Error! Reference source not found. describes processes with the following notations $(V, W, X, Y, Z)$.

An edge is defined as a set of dependencies between Tasks [90]. In Error! Reference source not found., the dependencies are in the form of edges $E_T = ((T_v)^1, (T_x, T_w)^2, (T_y)^3, (T_z)^4), \in T$ in which the superscripts denote the level of task processing and $T_v, T_x, T_w, T_y$, and $T_z$ are the tasks. For instance, the succeeding tasks, $T_x$ and $T_w$, depend on data generated by the preceding task ($T_v$) for execution. So, they are executed in parallel after completing task $V$. Task $T_v$ is in the first level of task processing, and after completing the $T_v$, the second level of tasks $(T_x, T_w)^2$ can be executed. Moreover, the exit task $T_z$ cannot start before the completion of $T_w$ and $T_y$. In other words, task $T_z$ depends on the data generated by the preceding tasks with the same priority ($T_w$ and $T_y$) for its execution. For a task $T_v$, the parameters are denoted as $T_v = \{AT_v, TS_v, DL_v\}$, where $AT_v, TS_v$, and $DL_v$ indicate the arrival time, task size, and deadline of task $T_v$, respectively. The task size is the length of the task. Makespan is the execution time of an initial task and its succeeding tasks, whereas the execution cost is the time expense of the allocated VM for a task completed. The Makespan defines the completion time of tasks from $T_v$ to $T_z$. C-PSO reduces execution cost and makespan by considering the precedence relationships between workflow tasks to schedule them efficiently. It employs adaptive task mutation that involves the adaptive scheduling plan depending on these task dependencies for optimizing resource utilization. The workflow task precedence relationship, such as a set of Preceding Tasks and Succeeding Tasks of $T_v$, is denoted as $\{PT\}T_v$ and $\{ST\}T_v$, respectively. According to the disequilibrium state of resource utilisation, C-PSO categorises the active PM into Over Loaded (PM$_{OL}$), Under Loaded (PM$_{UL}$), and balanced (PM$_{BL}$) conditions. C-PSO takes the PM$_{OL}$ and PM$_{UL}$ as
input and determines the best suitable destination PMIBL to execute the migrated
tasks energy-efficiently. The idle server utilisation is in the range of 5% to 15%, which
is denoted as (PMIs). The parameters below are introduced after stressing a high-end
server in each category in an ESXi 8.5 Hypervisor setup on PMs. The PM was
consolidated with 17 VMs, and the test was conducted by sending the load to the VMs
for about an hour, with results retrieved and recorded. The parameters may change
slightly due to the nature of the load being considered and tested in a physical server.
However, based on the behaviour of the server in terms of performance, these "fixed"
values were considered in this thesis for each category.

Table 5-2 Load Categories

<table>
<thead>
<tr>
<th>Load category</th>
<th>Notation</th>
<th>Variation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overloaded PM</td>
<td>PM_{OL(i)}</td>
<td>$C_i \geq 75%$</td>
</tr>
<tr>
<td>Underloaded PM</td>
<td>PM_{UL(i)}</td>
<td>$15% \leq C_i \leq 40%$</td>
</tr>
<tr>
<td>Balanced PM</td>
<td>PM_{BL(i)}</td>
<td>$41% \leq C_i \leq 60%$</td>
</tr>
<tr>
<td>Idle server</td>
<td>PM_{IS(i)}</td>
<td>$5% \leq C_i \leq 15%$</td>
</tr>
</tbody>
</table>

On the basis of the performance behaviour of the cloud server, specific thresholds are
determined for each category. Google data centres use a policy of 50% maximum CPU
usage, aiming to optimize performance and reduce VM migrations, driven by their
decision to have zero tolerance in performance to meeting customer demands
without compromise. This strategy is feasible for Google, given its abundant resources
to maintain all PMs at this level. However, evidence suggests that server stabilit y is
robust even in the utilization levels ranging between 70% and 75%. It indicates that
slightly higher utilization of the cloud resources ensures stable server performance
with enery efficiency.

5.4 Problem Formulation
This section provides the fundamental issues in CC processes and explains how HH-
ECO performance is high through a small example. Let the cloud consist of PM, VM
and Tasks. The VM resources must be allocated within each PM to execute the tasks.
Each PM is modelled using its Makespan (MS), SLA violations, and Degree of Disparity
(DD). The Makespan of the workflow application is the total time taken to complete
all the tasks allocated to the cloud resources. DD defines the uneven workload among
all the allocated hosts in which the minimum DD represents the situation when the
system is balanced. In other words, DD is defined as the amount of load distribution
among VMs based on their execution competencies. The SLA violation needs to be
ensured with DD. DD is derived by estimating the execution time across all PMs and
the average execution time across all PMs, defined by equation (5.4) in [189], i.e.

$$DD = \frac{(ET_{max} - ET_{min})}{ET_{avg}} \quad (5.4)$$

Where $ET_{max}$, $ET_{min}$, and $ET_{avg}$ denote the maximum, minimum, and average task
execution time of tasks assigned to each PM for the entire workflow, respectively,
among all the hosts. DD arises when the improper allocation of VMs and tasks occurs across different physical hosts in the computing phases, causing an uneven workload distribution. While using PSO, local convergence that leads to an inappropriate solution is also a primary reason behind this problem since local convergence violates the SLA. DD must be very low or almost close to 0 for an evenly distributed workload across all PMs with less deviation in execution times.

For a cloud environment, HH-ECO aims to provide Energy-efficient CC processes such as resource allocation, task scheduling, and migration. This approach ensures that these processes are executed in the shortest possible time without violating the SLAs. However, the cloud includes heterogeneous PMs and VMs, creating server overloading and underloading issues. The Energy-efficient CC problem may be considered a problem space $P_S$ in each process. The particles ‘p’ are flown through the solution space, $S^S$. Problem space is a space in which the energy-efficient problem exists, whereas solution space is the space in which the solutions to those problems exist. Every individual particle ‘p’ flies through $P_S^S$. Instead of providing the best solution to each particle, the best possible solution to all the particles without dominance is defined as the global optima. The impact of the best solution of a particle over other particles is named the dominance relationships. Based on the dominant relationship between two particles, the time taken to run the HH-ECO approach to the best, average, or worst case may vary, leading to higher time complexity. If the dominant particle rapidly makes other particles converge to an optimal solution, time complexity becomes low, which is referred to as the ‘best case’. If the dominant particle leads a balanced exploration and exploitation of $S^S$, time complexity will be moderate which is referred to as the ‘average case’. If the dominant particle causes premature convergence, time complexity will be high, which is referred to as the ‘worst case’. This can be clearly explained using an example given in Table 5.3.

Table 5-3 illustrates the time complexity of particles over the solution space in PSO. There are two solutions in the solution space, $S_{o1}$ and $S_{o2}$; that is, two different possibilities being explored are considered. It helps to improve the diversity in the solution space, which supports the algorithm to evade premature convergence on a single solution. Providing the best solution $s_1 \in S^S$ for particle ‘i’, problem space of i ($p_i$) tends the time complexity of another particle k to the average solution. The problem space of k is denoted by $p_k$. Selecting the best or average solutions for each particle without allowing any one particle to dominate the solution space reduces the time complexity (O). This is because the algorithm balances the exploration of diverse areas of $S^S$ to optimize the entire process. Table 5-3The task size represents the length of the task, and the Waiting Time represents the amount of time a task is in the idle state before it is processed. In this table, the best-case time complexity is considered as an estimate of the time taken to complete the task under ideal conditions without
any delay caused by overloaded VMs and PMs. In contrast, the average case considers the time taken to complete the task when a delay is caused by VM overloading. The worst case happens when the time taken to complete the task is increased due to delays caused by VM overloading and migration time.

### Table 5-3 Time Complexity of Particles over Solution Space in PSO

<table>
<thead>
<tr>
<th>Problem Space</th>
<th>Solution So1</th>
<th>Solution So2</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p_i )</td>
<td>Best: ( O((\text{Task Size}/C_{VM(i)})) )</td>
<td>Average: ( O((\text{Task Size}/C_{VM(i)}) + \text{Waiting Time due to VM(i) overloading}) )</td>
</tr>
<tr>
<td>( p_k )</td>
<td>Average: ( O((\text{Task Size}/C_{VM(i)}) + \text{Waiting Time due to VM(i) overloading}) )</td>
<td>Worst: ( O((\text{Task Size}/C_{VM(i)}) + \text{Migration Time to VM(j) + Waiting Time due to VM(j) overloading}) )</td>
</tr>
</tbody>
</table>

Local convergence is referred to as the convergence of a particle for satisfying SLA before reaching its appropriate solution. Local convergence violates the SLA during the finding of the best solution in a suboptimal state hence, the local convergence avoids the optimal solution. An optimal solution balances the global and local search space in the available cloud resources for the execution of scientific workflow tasks, eliminating premature convergence.

To avert local convergence in PSO, the C-PSO method applies the process of mutating the particle to another position in \( S^S \), which controls the dominance area (i.e., time complexity) of one solution to the other and avoids the deterioration of a global optimum. By mutating the particles, C-PSO explores the diversified and appropriate positions of the solution space by reducing the probability of prematurely finding a suboptimal solution. It facilitates the identification of global optimum by preventing the deterioration of the solution quality. The mutation is the process of particle movement towards the global best position in which the movement relies on the particle velocity, previous best position, and inertia weight. In PSO, the inertia weight controls the velocity for maintaining the trade-off between the global and local search abilities; and minimises premature convergence. Applying C-PSO with a mutation in each phase in cloud-based workflow permits HH-ECO to provide a time-computation-efficient solution.

### 5.4.1 Resource Allocation and Task Scheduling:

The mutation algorithm reduces the total execution time of all PMs and the DD value by identifying the best global solution using C-PSO. The optimal C-PSO algorithm in HH-ECO derives a solution for the VM-PM resource pair as well as the task-resource pair for the execution of workflow tasks. In a cloud environment, tasks are executed
on VMs that are hosted on PMs. The cloud environment handles the allocation of tasks to VMs and the allocation of VMs to PMs depending on workload and available resources. Examining energy-efficient resource allocation relies on the assumption of evenly distributed tasks among resources. Consider an equal load distributed among various PMs. The execution time of tasks decides the total cost of PM and the DD value. The total cost is the summation of the execution time of \( l \) tasks executed by the PM, i.e. \( \sum_{tasks=1}^{l} \text{Execution time}_{\text{task}} \). To utilise the advantage of energy-efficient resource allocation, HH-ECO deals with task scheduling pragmatically based on the even distribution of tasks. The task scheduling measures the fitness using Makespan and total execution time, where the fitness is defined as the condition of being an optimal solution without dominance (i.e., a balanced solution in which the workload across all PMs and VMs in the cloud environment accomplishes optimal performance and energy efficiency). Based on the workflow, the tasks are mutated among PMs to reduce the waiting time, resulting in energy-efficient task scheduling. It is because a mutation in C-PSO involves shifting the workloads in terms of tasks among energy-efficient PMs based on their current workload and performance metrics, ensuring efficient resource allocation and task scheduling with reduced execution time and energy consumption. Usually, heuristic algorithms (like C-PSO) are designed to find near-optimal solutions in minimum time, but they will not find the absolute global optimum in every condition. The time taken for C-PSO to converge optimally depends on the complexity of the problem space and the parameters used. The convergence time upper bound may significantly vary based on the size and nature of the task scheduling problem.

5.4.2 VM Migration:
Due to the lack of knowledge in overall PM capacity utilisation, PM can be overloaded or under-loaded at any time. When a PM is overloaded, the VM is migrated using the load category and C-PSO. Waking up the PM for migration increases the total cost. The waking-up process is likely to increase the responding time of the application, in which the waking-up time varies according to the state of the resource, such as idle, sleep, or shut down. The decision to VM migration along with the task without considering the domination of the succeeding task may affect the MS. Traditional migration decision that did not consider the dependencies of the succeeding task, the execution time of the overall workflow application becomes high due to improper allocation that leads to increased delay in the task completion. Three cases of VM migration are considered.

- Case (i): if the resource is in an idle state, there is no need for a wake-up time because the idle resource is in an active state,
- Case (ii): if the resource is in sleep mode, it takes a wake-up time of 300ms to become an active resource,
• Case (iii): if the resource is in a shutdown state, it requires a wake-up time of 480ms for the transition [190].

Thus, the consumption of wake-up time is idle < sleep < shut down. Also, applying the pre-wake-up policy without sacrificing energy efficiency is necessary to decrease the response time. Hence, HH-ECO focuses on implementing the pre-wake-up policy with consideration of the Makespan, computational cost, and energy consumption. The following section explains the processes of the HH-ECO in detail.

5.5 The HH-ECO approach

Figure 5-2 HH-ECO includes three CC processes, namely, VM allocation, task scheduling, and migration. In the VM allocation phase, each VM has a predefined CPU, memory, storage, IP address, etc. VM placement aims to find the best PM to run the appropriate VMs. The second phase is task scheduling, in which hierarchical task scheduling plays a significant role in workflow applications. Scheduling the tasks on available VMs, based on the Workflow precedence, is essential. Thirdly, the overloaded VM must be migrated to idle PMs in the VM migration phase. VM migration is performed either by the HH-ECO system or locally based on the PM conditions. Considering workflow relationships are vital during task-VM mapping and task execution to ensure energy efficiency in the cloud. Task scheduling is performed based on workflow precedence, ensuring that tasks are scheduled on available VMs based on dependencies and relationships between tasks. Proper VM mapping and migration using workflow relationships helps to obtain optimal solutions.

HH-ECO approach combines the Chaotic and PSO algorithms and adaptively applies the C-PSO to the CC processes. PSO incurs an earlier standstill of the particles before reaching the global optima due to the low-level diversification among particles. This problem is called premature convergence. Thus, the proposed work applies a chaotic mapping function to assist the particles in breaking away from the local optima when it reaches premature convergence in each iterative searching process and improves the accuracy of the HH-ECO. To prevent the PSO algorithm from being trapped into a premature convergence, it is essential to apply control over local exploitation, i.e., taking advantage of previous best solutions, and global exploration, i.e., the heuristic search over new regions. PSO performance significantly relies on its parameters, especially the velocity of the previous particle. The previous velocity provides the necessary momentum for particles to find the best-fit solution for the search space. Thus, proper control of the particle velocity is essential to determine the optimum solution accurately and efficiently. HH-ECO employs a chaotic sequence with high randomness and regularity to ensure global convergence, improving the solutions' diversity. Moreover, the adaptive inertia weight factor-based proposed model effectively balances the local and global exploration to avoid premature convergence.
HH-ECO operates dynamically depending on cloud environment conditions, thereby allowing it to adjust the processes as needed to maintain optimal performance. It operates continuously since it iteratively searches for the best solutions.

Figure 5-2 Hybrid Heuristic-Based Energy Optimized Task Scheduling and VM Migration of Scientific Workflow
The C-PSO adaptively controls the inertia weight using the fitness measurement and moves the particle toward the best solution without local optima. In essence, the randomness of the chaotic theory improves the PSO algorithm with the chaotic mutation operator. The C-PSO algorithm prevents the convergence of the particle position. It leads other particles to move away from the local optimal solutions within a short period. Figure 5.2 shows the hybrid heuristic-based energy-optimized task scheduling and VM migration of scientific workflow.

In the HH-ECO approach, several components are used for the workflow execution, which are discussed below.

**Chaotic iteration:** The HH-ECO approach performs VM allocation, task scheduling, and migration using the chaotic-based PSO algorithm to avoid premature convergence.

**PSO-based scheduling:** The PSO algorithm detects the best allocation and scheduling plan iteratively for reducing energy consumption. In addition, it devises the migration plan based on the minimisation of cost and time consumption that is, allocation of the VM to the suitable PM iteratively.

**VM allocation:** The VM allocation aims to run the workflow task in the suitable PM to achieve the objective function, where the VM contains the predefined CPU, memory, storage, and IP address.

**Task-resource mapper:** The task-resource mapper matches the workflow task with the suitable PM that executes it with minimum time and cost consumption.

**Workflow task dependency maintenance:** It is essential to ensure that it is a contributed equation and that other authors are not involved in the precedence of the task during the execution of workflow application due to the data dependencies.

**Workflow task dependency-based scheduler:** The task scheduler schedules the task based on the best scheduling plan without changing the workflow precedence.

**Workload status predictor:** The status predictor obtains the current load status of the PM and VM for making decisions regarding the VM migration.

**Dynamic Makespan Measurement:** It is the measure of the execution time of scientific workflow tasks on the allocated resources during the estimation as well as after completion to resolve the multi-objective optimization problem in the cloud.

**SLA-aware VM migration:** VM migration dynamically monitors the PM during the execution of workflow tasks. If the PM is overloaded or underloaded, it migrates the VM to the idle or balanced PM using the C-PSO algorithm.

**Workflow task Precedence relation:** It maintains the workflow precedence even after the migration of the VM

### 5.6 VM Allocation with Adaptive Mutation

To ensure performance and energy efficiency, the HH-ECO allocates the VMs to minimise energy consumption instead of meeting SLAs. The HH-ECO approach implements the VM allocation on heterogeneous PMs and VMs. CPU and memory
utilisation are the most critical components since they contribute significantly to servers' performance [70]. Consider that the 'n' number of VMs has to be allocated to the 'm' number of PMs. According to their capacity (CPU performance, memory, storage, and workload), the HH-ECO categorises the VMs and PMs into Small, Medium, and Large types [191] and then allocates the VMs to their appropriate PMs using the C-PSO algorithm. During VM allocation, a particle is considered as a potential solution to the VM allocation on the PM resources. Particularly, each particle denotes an appropriate VM to PMs in the cloud environment. The PSO models the simple mathematical formulae using particle velocity, position, and inertia weight. Note that, in every iteration, the C-PSO generates 'k' solutions (particles), which indicates the possible way to allocate VMs to PMs by considering the performance, energy efficiency, and workload balancing. Among these solutions, the best solution (best PM to allocate VM) is considered G_{Best}. Each particle has a position indicating the current allocation solution and a velocity indicating the direction and rate at which the particle varies its allocation plan in every iteration. The previous positions of all particles in a j^{th} solution of the i^{th} iteration act as P_{Best} for the j^{th} solution of the i+1^{th} iteration. The chaotic algorithm applies an inertia weight W to control the impact of the previous velocity on the current one and balance between exploitation and exploration during optimal VM allocation. W influences the particles' movement in the search space. A more significant inertia weight guides the particles to the global search, while a smaller factor leads particles to a local search through the previously estimated best solutions. The HH-ECO estimates the adaptive inertia weight factor (W) (using equation 5.5) and new velocity (V_{j}^{ITER+1}) (using equation 5.7), where W_{max} and W_{min} refer to the maximum and minimum values of inertia weight (W), and Fitness_{max}, Fitness_{min}, and Fitness_{avg} represent the maximum, minimum, and average fitness of the particle in all solutions of the i^{th} iteration respectively. In Equation 5.5, when the fitness value is high, the inertia weight is reduced, allowing for exploration in the search space. Otherwise, the inertia weight is increased, enabling exploitation around the appropriate available resources that fulfil the multiple objectives. Equation (5.6) computes the weighted threshold (W_{α}) based on the inertia weight and the fitness values of particles in the corresponding iteration to adaptively compute the inertia weight. The fitness function is used to evaluate particles by measuring the effectiveness of the VM allocation plan in terms of resource utilization, energy efficiency, and SLA compliance. The HH-ECO approach applies the C-PSO algorithm in optimising each CC phase using equations (5.5) - (5.7).

$$\text{Inertia Weight, } W = \begin{cases} W_{min} + W_{α}, & \text{if } \text{Fitness}_{max} \leq \text{Fitness}_{avg} \\ W_{max}, & \text{if } \text{Fitness}_{max} > \text{Fitness}_{avg} \end{cases}$$ (5.5)
Where $W_{\alpha}=\frac{(W_{\text{max}}-W_{\text{min}})(\text{Fitness}_{\text{max}}-\text{Fitness}_{\text{min}})}{\text{Fitness}_{\text{avg}}-\text{Fitness}_{\text{min}}}$ (5.6)

$$V_{j}^{\text{ITER}+1}=W*V_{j}^{\text{ITER}}+c_{1}r_{1}(P_{\text{Bestj}}-X_{j}^{\text{ITER}})+c_{2}r_{2}(G_{\text{Bestj}}-X_{j}^{\text{ITER}})$$ (5.7)

Where $c_{1}$ and $c_{2}$ are positive constant parameters, and $r_{1}$ and $r_{2}$ represent the random value generated from the random function.

According to the participation of a single type of PM over the total $C_{i}$ of all the PMs, the HH-ECO approach allocates the VMs. An example of the VM allocation to S, M, and L types of PMs is depicted in Table 5-4 and 5.5. The total CPU performance and memory values of VMs and PMs are given in the table. VMs are allocated to appropriate PMs using C-PSO. The VMs allocated for small PM are VM1 and VM2, which results in a utilization of 91.67% of the PM's capacity. The VMs allocated for medium PM are VM3, VM4, and VM5, utilizing 86.21% of their capacity. The large PM is allocated with VM6, VM7, VM8, and VM9, utilizing 84.55% of its capacity. A VM allocation strategy is performed efficiently by utilizing the resources of different PMs while satisfying the requirements of VMs related to CPU and memory capacities.

Table 5-4 CPU performance and memory of VMs and PMs - Example scenario

<table>
<thead>
<tr>
<th>VM and PM</th>
<th>CPU Performance (MIPS)</th>
<th>Memory (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>VM1</td>
<td>50</td>
<td>40</td>
</tr>
<tr>
<td>VM2</td>
<td>80</td>
<td>50</td>
</tr>
<tr>
<td>VM3</td>
<td>100</td>
<td>40</td>
</tr>
<tr>
<td>VM4</td>
<td>120</td>
<td>50</td>
</tr>
<tr>
<td>VM5</td>
<td>130</td>
<td>60</td>
</tr>
<tr>
<td>VM6</td>
<td>140</td>
<td>70</td>
</tr>
<tr>
<td>VM7</td>
<td>160</td>
<td>80</td>
</tr>
<tr>
<td>VM8</td>
<td>150</td>
<td>70</td>
</tr>
<tr>
<td>VM9</td>
<td>200</td>
<td>60</td>
</tr>
<tr>
<td>Small PM (S)</td>
<td>140</td>
<td>100</td>
</tr>
<tr>
<td>Medium PM (M)</td>
<td>400</td>
<td>180</td>
</tr>
<tr>
<td>Large PM (L)</td>
<td>800</td>
<td>300</td>
</tr>
</tbody>
</table>

Table 5-5 VM Allocation - Example scenario

<table>
<thead>
<tr>
<th>PM type</th>
<th>VM Allocated</th>
<th>VM Allocation (%)</th>
<th>PM Capacity utilized by VMs (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small (S)</td>
<td>VM1, VM2</td>
<td>12.5%</td>
<td>91.67%</td>
</tr>
</tbody>
</table>


CHAPTER 5  HYBRID HEURISTICS-BASED ENERGY-EFFICIENT OPTIMIZATION
OF SCIENTIFIC WORKFLOW IN CLOUD

<table>
<thead>
<tr>
<th>Medium (M)</th>
<th>VM3, VM4, VM5</th>
<th>30.21%</th>
<th>86.21%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Large (L)</td>
<td>VM6, VM7, VM8, VM9</td>
<td>57.29%</td>
<td>84.55%</td>
</tr>
</tbody>
</table>

In order to demonstrate the steps of the C - PSO procedure in the cloud environment, two case examples are used, explaining the C-PSO processes in VM allocation.

Case 1

Initially, HH-ECO decides the participation level of each type of VM and executes the fitness function. Notably, the total participation level of all types of VMs is equal to 12.5%, as per Table 5.5. Different combinations of VMs are created to reach a 12.5% participation level. Each combination is denoted as a particle and the position vector of particles. According to PSO, the values of W and \( V_{j}^{ITER} \) are initially randomly chosen from 0 to 1, using the \( W \) and \( V_{j}^{ITER} \), the value of \( V_{j}^{ITER+1} \) is estimated using equation (5.5). In the initial case, the value of \( P_{Bestj} \) and \( G_{Best} \) equal the position vector of the first particle. \( X_{j}^{ITER} \) also represents the first particle’s position vector. Thus, the terms \( c_{1}r_{1}(P_{Bestj} - X_{j}^{ITER}) \) and \( c_{2}r_{2}(G_{Best} - X_{j}^{ITER}) \) tend to zero, and the new velocity value equals \( W * V_{j}^{ITER} \). The \( W \alpha \) value is identified using equation (5.5) in the following solution of the same iteration.

Case 2

After the first iteration, \( W \alpha \) is estimated. Moreover, the \( P_{Bestj} \) and \( G_{Best} \) values of the particle are also identified. In the previous iteration, if the fitness value is maximum than fitness\(_{avg}\), \( W \alpha \) is high, so the value of \( V_{j}^{ITER+1} \) is also high, according to equation (5.7). This scenario represents that the location of the best solution is at a long distance from the previous solution due to the high difference between them (Fitness\(_{max} \)−Fitness\(_{min}\)). A high value of (Fitness\(_{avg} \)−Fitness\(_{min}\)) represents the closeness of the best solution. Thus, C-PSO reduces its velocity significantly. Moreover, if fitness is more than the average fitness, the high values of W and \( V_{j}^{ITER+1} \) are used to identify the suitable solution for CC processes. A significant inertia weight guides the particles to the global search, while the smaller factor leads particles to the local search through the previously estimated best solutions.

To reach the total capacity of a single type of PM, C-PSO selects different types of VMs (A, B, and C) in each iteration. For the selected VMs, the HH-ECO applies equation (5.8) to measure the fitness of the VM allocation plan. The best Allocation Plan (\( A_p = \{A_g, B_g, C_g\} \)) is decided using the fitness measurement according to the Capacity (\( C_i \)) of VM and Memory utilisation (\( M_i \)) of VM in many iterations, where \( A_g \) represents the fraction of the number of VMs allocated to the ‘A’ type. The allocation plan iteration continues
until fitness reaches the condition of \(0.8 \leq \text{Fitness}_{A_p} < 1\), assuming that 80% of the suitability of solutions avoids balanced PMs. This condition infers that the fitness of the solutions must be within the range of 0.8 to 1, representing a high level of efficiency or suitability of the allocation plan. For the capacity of a single type of PM, the VMs are allocated. For instance, the participation level of a large-capacity PM is 57.29%. Equation (5.8) decides the allocation of the VMs required in each type to reach 57.29% of the total PM capacity, where the total capacity and memory are 57.29% of the total available PM capacity and memory, respectively. The C-PSO algorithm executes equation (5.8) for small, medium, and large capacity PMs individually to reach an optimal VM allocation.

\[
\text{Fitness}_{A_p} = \left( \frac{\sum_{i=1}^{n} C_i}{\text{total capacity}} \right) \times \left( \frac{\sum_{i=1}^{n} M_i}{\text{total memory}} \right) 
\]  

(5.8)

Where \(A_p\) → Fitness for allocation  
\(C_i\) → capacity utilization of VM  
\(M_i\) → memory utilization of VM  
Total capacity → Total capacity of PM  
Total memory→ Total memory of PM  
n → number of VMs

According to equation (5.8), the PM, which has several VMs more than its capacity, i.e., fitness is greater than \(\text{Fitness}_{avg}\), is classified under \(\{q\}\). Otherwise, the PM is grouped under \(\{p\}\). The VM resource allocation defines 'p' and 'q' as the number of VMs in the \(\{p\}\) and \(\{q\}\), respectively, as mentioned in equation (5.9).

\[
\text{Particle Type}= \begin{cases}  
\{p\} & \text{if } \text{Fitness} < \text{Fitness}_{avg} \\
\{q\} & \text{if } \text{Fitness} > \text{Fitness}_{avg}  
\end{cases} 
\]  

(5.9)

HH-ECO aims to balance the utilization of PMs by reallocating VMs among them. Certain particles grouped under \(\{q\}\) indicates an imbalance that is VMs with high capacity. Hence, the mutation process is initiated, which involves transferring a VM from an overloaded PM to an underloaded PM with the satisfaction of multiple objectives, thus aiming to balance the workload across PMs. Hence, HH-ECO ensures that all PMs function closer to their maximum capacity by optimizing resource utilization and overall system performance.

5.7 Task Scheduling with Adaptive Mutation

In Workflow applications, hierarchical task scheduling plays a significant role. The Workflow task relationship must satisfy the quality constraints to plan the task scheduling in a cloud environment effectively. The proposed task scheduling includes
a change in focus of optimization from allocating VMs to PMs to scheduling tasks to VMs in workflow applications in a cloud environment. In other words, without considering the allocation of VMs to PMs, the optimization considers scheduling workflow tasks to on-demand VMs based on their requirements and dependencies. At the task level, scheduling on-demand VMs to the tasks with knowledge of workflow can minimize the total cost without violating the SLA requirements. By applying C-PSO, the HH-ECO considers the static resource utilization of the PMs under idle state and dynamically estimated Makespan and execution cost of scheduled tasks by C-PSO as input to the fitness measurement. Total execution cost denotes the summation of the execution cost of all the subtasks that helps to evaluate the performance of subtasks. When the cost of the scheduling plan is minimal, the HH-ECO schedules the tasks to the selected resource based on fitness score in the C-PSO.

The HH-ECO selects the best scheduling plan using equations (5.5)-(5.7) based on the chaotic sequence and the 'W' factor. When the scheduling plan returns a higher fitness value, it converges to the optimum. The high fitness value is returned only when the tasks are allocated to a particular VM with a minimum Makespan than the deadline and with minimum total execution cost using equation (5.10). In equation (5.10), t1 and t2 denote the weighted parameters for two objectives of Makespan and total execution cost, where 0 ≤ t1, t2 ≤ 1 and t1 + t2 = 1. If the makespan is within the deadline, t1 is set to 1 and t2 to 0; otherwise, t1 is 0 and t2 is 1.

\[
\text{Fitness } TS_c = t1 \times \left(1 - \frac{\text{Makespan}}{\text{Deadline } TS_c}\right) + t2 \times \left(\frac{1}{\text{Total execution cost}}\right) \quad (5.10)
\]

where
- \(TS_c\) → Fitness for task scheduling
- \(t1\) → weighted parameter of makespan
- \(t2\) → weighted parameter of execution cost
- \(\text{Deadline } TS_c\) → Deadline for task scheduling

Both the cost and makespan deadline of VM is considered for computing task-resource fitness score in which the makespan is to be comparatively less than the deadline, along with the consideration of efficient resource utilization to improve the task execution performance. Consequently, the C-PSO-based scheduling algorithm decreases the scheduling space and generates the best scheduling plan. However, the generated plan only satisfies the individual particle but does not consider other particles' dominance. For example, according to the C-PSO, the Workflow tasks in Error! Reference source not found. are scheduled for small, medium, and large capacity VMs, as shown in Figure 5-3.
According to the C-PSO algorithm, task X is scheduled in a small capacity VM after the execution of task V. However, when the succeeding task of X (Y) runs after the completion of task W, the Makespan will be increased. In essence, the decision to schedule task W without considering the dominance of Y (dependency relationship between task Y and its preceding task X) affects the Makespan. As Y does not have dependencies other than X, its execution can be affected by the scheduling decisions made for X. To avoid this, HH-ECO applies mutation to the task scheduling plan. Each task has its preceding task (PT)Ti and succeeding task (ST)Ti. If a task Ti dominates any of its (ST) Ti, the HH-ECO applies the adaptive mutation. Accordingly, task X dominates task Y. Consequently, such a scheduling plan destroys the resource utility due to the delayed start time of task Y by task W even after the completion of its preceding task X. Hence, the HH-ECO approach mutates task W on another best fit free VM based on the high value of the mutation-selection, which reduces the Makespan without wasting the resources and violating SLAs.

5.8 VM Migration with Adaptive Mutation

The HH-ECO can successfully schedule the tasks according to the proposed task scheduling with the adaptive mutation. However, due to a lack of knowledge of overall PM capacity utilisation, each PMi can be overloaded or under-loaded at any time. So HH-ECO monitors each PM after the tasks are scheduled and decides the state of PM (PM_{OL(i)}, PM_{UL(i)}, and PM_{IL(i)}). In the context of VM migration, consider PM_{OL(i)} and PM_{UL(i)} are source PMs. A set of predicted underloaded and overloaded PMs is denoted as Source_{UPM} = \{S_{UPM1}, S_{UPM2}, \ldots, S_{UPMk}\} and Source_{OPM} = \{S_{OPM1}, S_{OPM2}, \ldots, S_{OPMk}\} respectively. A set of balanced (active), idle servers are represented as destination PMs, Active Dest_{APM} = \{D_{APM1}, D_{APM2}, \ldots, D_{APMk}\} and (Idle Dest_{IPM} = \{D_{IPM1}, D_{IPM2}, \ldots,
D_{IPMk}, respectively. Moreover, Dest_{PM} represents a combination of Dest_{APM} and Dest_{IPM}. PMs are balanced when their resource utilization is optimally balanced, achieved by monitoring each PM’s utilization status during the execution of scheduled workflow tasks. From source to destination, the VMs are moved until all the PMs are balanced. Consider that (n-k) VMs are allocated in the i^{th} PM. If the ratio of all the (n-k) VMs over the total capacity of PM exceeds the ¾^{th}, then the PM is categorised as Overloaded. Offloading the workflow task to the PM, whose utilisation rate ranges between 80-100% capacity range, causes performance degradation. Hence, the optimal 75% is fixed as the minimum value for the workflow execution. Otherwise, it is categorised as Under Loaded, which is finalised from the process and result of experiments discussed in [192] and [131].

To attain the energy-efficient cloud, the proposed model performs this experiment as the prerequisite process that evaluates the performance of these factors for the different threshold range values.

\[
\text{Load Category = } \begin{cases} 
\text{Under Load} & \text{if } 0.15C_i < \left( \frac{\sum_{j=1}^{n-k} C_{ji}}{C_i} \right) \leq \frac{3}{4} C_i \\
\text{Over Load} & \text{if } \left( \frac{\sum_{j=1}^{n-k} C_{ji}}{C_i} \right) > \frac{3}{4} C_i \\
\text{Balanced} & \text{if } 0.41\% C_i < \left( \frac{\sum_{j=1}^{n-k} C_{ji}}{C_i} \right) \leq \frac{3}{3} C_i \\
\text{Idle} & \text{if } 0.05\% C_i < \left( \frac{\sum_{j=1}^{n-k} C_{ji}}{C_i} \right) \leq \frac{0.6}{4} C_i 
\end{cases}
\]  

(5.11)

PM_{i} is a PM chosen from the set of destination PMs (DestPM) for potential VM migration. \(C_{VM}\) is the predicted capacity utilisation of the VM that creates the host either over or under-loaded. The Fitness value is equal to the inverse of the difference between the capacity of the destination or target PM (\(C_{i\in\text{DestPM}}\)) and the capacity utilisation of the utilization of ith PM associated with the resource utilization or capability of the VM to be migrated. VM migration to the high-fitness destination PM favours reduced under-utilisation and over-utilisation of PMs. When migrating the VM from the source PM into the new destination PM, the capacity of the destination PM must be less than or equal to its total capacity after migration happens. Otherwise, the fitness value is zero. The VMs are migrated according to the high fitness destination measured using C-PSO. The high-fitness destination is considered a suitable solution.

\[
\text{Fitness Source}_{PM(i)} = \begin{cases} 
\left( C_{i\in\text{DestPM}} - (U_{PM\in\text{DestPM}} + C_{VM}) \right)^{-1} & \text{if } \{U_{PM\in\text{DestPM}} + C_{VM}\} \leq C_{i\in\text{DestPM}} \\
0 & \text{Otherwise}
\end{cases}
\]  

(5.12)
$U_{PM_i \in \text{DestPM}}$ represents the utilization of the $i$th PM associated with resource utilization.

$PM_i \in \text{DestPM}$ represents the $i$th PM in a set of DestPM.

Where $C_{VM} \rightarrow$ capacity utilization of VM.

If the capacity of the DestPM exceeds its total capacity after migration, the fitness value will be zero. It indicates that DestPM is not suitable for migration. During the VM migration, HH-ECO allows the PM to enable sleep mode when all its loaded VMs migrate. Moreover, it allows the PM in sleep mode to switch on only when migrating a VM to an already active PM is impossible. However, the sleep mode PM might take time to become an active server. In such a time, the VM possibly executes the task in the source PM per the scheduling plan and moves it to the overloaded state. HH-ECO tunes the idle server to sleep mode to reduce energy consumption if a task is not allocated to the idle server. If a VM with the task ‘$i$’ is migrated due to an overloaded or underloaded PM, it ensures that the preceding task of ‘$i$’ is completed. After completing the preceding task, the task ‘$i$’ can be executed in the migratable VM. According to this procedure, HH-ECO estimates the triggering time point ($\tau_P$) for waking up an active server from sleep mode, triggers the server at the time of $\tau_P$ and avoids unnecessary resource wastage. In equation (5.14), $C_o$ represents a constant value for the resource type, and it is assigned as 2 sec for large-capacity resources. As large-capacity resources can handle heavier workloads, they may require longer time intervals. Assigning a constant value of 2 sec helps to standardize the timing parameters within the system. Timepoint refers to the time in geometric space. $MT_P$ denotes the migration time point, which is the time point at which the VM migration process occurs. $CT_P$ denotes the current time points, which is the current state of the system.

\[
MT_p = CT_p + \text{Completion Time} \{(PT)\}_{v} \tag{5.13}
\]

\[
\tau_P = MT_p - \{Waking-up Time + C_o\}_{VM} - MT_{VM} \tag{5.14}
\]

Where $Waking-up Time =$ Time is taken to wake up the resource from the idle state.

After completing the preceding task $(PT)_{v}$, the succeeding task can be executed in a migratable VM. $MT_{VM}$ is the time consumed to migrate the VM from the source to the target resource. $\tau$ is the subtraction of $(Waking-up Time + C_o)_{VM}$ and $MT_{VM}$ from $MT_P$. 


5.9 Overall Algorithm

The process of allocating VMs to PMs primarily considers the fixed capacities of VMs, thus ensuring efficient resource utilization. It optimizes the initial distribution of

Figure 5-4 Flowchart of HH-ECO Methodology
workload among PMs depending on their capacity constraints. When the workflow progresses, its demands may change and VM capacity remains unchanged. Hence, VM migration is essential for maintaining optimal resource utilization and preventing PM from overloading or underloading for efficient task execution. The changes in workflow and running C-PSO for tasks depend on the dynamic nature of the workload and its requirements. It occurs periodically or when substantial changes in the workload happen. It is executed within the proposed HH-ECO approach to handle resource allocation and task scheduling adaptively. The process is rapidly performed by the integration of chaotic and PSO algorithms for making efficient decisions. The process iterates until achieving fitness conditions that are favorable for balanced VM allocation and optimal task scheduling to achieve high performance.

Figure 5-4 illustrates the overall sequential steps of the HH-ECO approach. Initially, the HH-ECO approach categorises the available PMs and VMs into the small, medium and large types based on their capacity, where the VMs have predefined CPU, memory, storage, and IP address. Consequently, the VM allocation decides the limitation of the number of VMs and their types to specific PMs based on their capacity. The proposed approach combines the chaotic and PSO algorithms for effective decision-making regarding the allocation to detect the suitable PM for the VM allocation. HH-ECO approach reduces 80% of VM allocation to the balanced PM by continuing its iteration until the fitness achieves the condition of $0.8 < \text{fitness}_{A} < 1$. This fitness function aims to efficiently allocate VMs to PMs by iteratively adjusting VM allocation until fitness reaches the desired threshold, indicating energy-efficient resource utilization. After the completion of VM allocation, the PMs are categorised into three types (small, medium, and large) based on the number of allocated VMs. Therefore, the HH-ECO approach adapts the mutation mechanism if the fitness value of the PM is higher than the threshold value. When the fitness value of a PM does not meet the threshold, HH-ECO focuses on the workflow dependency of the incoming tasks to enable the migration process that is accomplished by the adaptive mutation in the C-PSO. The mutation mechanism targets optimally balancing the resource utilisation level of all the PMs by mutating the underloaded or overloaded VM from the corresponding PM to the energy-efficient destination PM. The proposed approach continues with effective task scheduling to the appropriate VMs using the C-PSO algorithm. It makes the best scheduling plan for the workflow task using the fitness value measured by estimating makespan and cost consumption. Allocating VMs to PMs is not only for load equalization but also for ensuring workflow task dependencies using mutation mechanisms since it also considers the workflow’s incoming tasks.

Moreover, the quality of the scheduling plan is mainly based on the value of the fitness score (given in equation 5.8). With the high fitness score, the best scheduling plan minimises the Makespan. During the execution of workflow tasks, it is necessary to monitor the load status of the PM to avoid performance degradation. The VM migration to the balanced or idle PM occurs if the PM is overloaded or underloaded.
If the VM migration for the succeeding task is required, then utilise the C-PSO to mutate the task within the PM before its migration. Moreover, the workflow precedence relationship maintenance occurs if the VM is migratable. Thus, the proposed approach executes the workflow task with maximum energy efficiency, minimum Makespan, and without SLA violation.

5.10 Summary

The HH-ECO approach performs VM allocation, task scheduling and VM migration using the C-PSO algorithm in CC. The C-PSO algorithm obtains the optimal solution to improve the Cloud system's service quality. With the help of the C-PSO algorithm, the HH-ECO approach achieves an effective execution of the scientific workflow application within the minimum time complexity. Moreover, it enhances resource utilisation, and it leads to the minimisation of energy and cost consumption. The proposed approach is implemented in Cloudsim and compared with the existing approaches, which are discussed in Chapter 6.
This chapter presents the experimental results using the WorkflowSim toolkit to evaluate the efficiency of the proposed approach. It compares its performance with two proposed and published existing cloud-computing techniques. The proposed HH-ECO approach is implemented in the CloudSim simulator rather than in real-time since simulation provides a controlled, flexible, and cost-effective environment for testing and validating the performance and behaviour of the model under various scenarios without any complexities and computational costs associated with real-time deployment. CloudSim enables us to experiment with different workloads, percentages of resource utilization, and hosts for evaluating the performance before implementing the model in a real-time environment.

6.1 Overview of CloudSim
Simulation tools provide valuable solutions for evaluating the performance of cloud computing tasks. It is vital to predict the data-intensive workflows’ performance before positioning. It ensures that the cloud infrastructure can provide services in line with SLAs and required QoS levels. However, choosing the appropriate simulation tool necessitates the potential consideration of numerous factors. For instance, the simulator provides real-time models for cloud environments and applications. Considering the capabilities of simulators, CloudSim is selected for the following reasons. Typically, CloudSim provides a detailed model of cloud hardware specifications, rendering it proficient at modelling and simulating intricate workflows without substantial gaps in representing crucial cloud environment components. It stands out as a valued option for creating frameworks with optimal efficiency. WorkflowSim provides the benefit of performing repeatable and reproducible experiments without experiencing any costs for testing environments. Also, both simulators depend on Java, featuring open-source code, allowing for custom class writing and code development when required. Although CloudSim outshines at simulating large-scale cloud environments, it faces scalability issues while dealing with very large and complex cloud environments. Implementing simulations in CloudSim can take considerable computational resources, mainly for simulations with vast numbers of VMs, hosts, and tasks.
CloudSim [193] acts as a multipurpose and scalable simulation framework developed to enable the modelling, simulation, and experimentation of developing cloud computing environments. It provides capabilities for modelling and simulating large-scale data centres, allowing for the creation of multiple VMs with personalized policies for providing host resources to these VMs. CloudSim uses discrete event-driven simulation techniques that involve the exchange of events among entities in the simulation framework. It supports several core functions like event queuing and processing, inter-component communication, entity creation, and simulation clock management. The components of CloudSim abstract various elements of the cloud computing environment, including data centres, VMs, CPUs, and bandwidth.

To implement the proposed HH-ECO approach for scientific workflows, the WorkflowSim [194] is the widely utilized simulation tool for scientific workflows, providing a comprehensive cloud platform for modelling and simulating the complex dependencies in the workflows. It is an extended model of CloudSim that supports workflow-level simulation and enables seamless cloud modelling, simulation, and experimentation for workflow tasks. The simulator follows the methodology delineated by the Pegasus workflow management system [195], including vital components like a workflow mapper, scheduler, engine, and features. These components allow the users of WorkflowSim to evaluate and improve various algorithms and techniques related to resource allocation and workflow execution. Compared to Cloudsim, an extension of CloudSim is WorkflowSim particularly handles the interdependent tasks in complex workflows, comprising the workflow scheduling component based on the task dependencies. Also, WorkflowSim can manage the data-intensive tasks in the cloud. Some other simulators that are similar to CloudSim are iFogsim2 [196], CloudNetSim++ [197], Cooja [198] and iCanCloud [199].

6.2 Scientific Workflows

Workflows include different scientific and computational domains, ranging between astronomy and bioinformatics. They act as valuable test cases for assessing the performance of workflow scheduling algorithms in a cloud environment. The primary workflows that exhibit varying makespan times are MONTAGE, LIGO, EPGENOMICS, CYBERSHAKE, INSPIRAL, and SIPHT. These workflows are briefly discussed below.

- The Laser Interferometer Gravitational-Wave Observatory (LIGO) emphasizes the observation and investigation of astrophysical gravitational waves, providing vital data for research in physics and astronomy [200].
- MONTAGE includes the mosaics’ computational generation from sky images, initiating the preliminary step of reprojecting gathered images depending on their corresponding coordinates [201].
• EPIGENOMICS functions as an information parallel workflow that receives data from the IlluminaSolexa Genetic Analyzer as DNA sequence paths [202].
• SIPHT includes the comprehensive prediction and annotation of RNA sequences by using a sequence of individual programs implemented in a particular order with the help of Pegasus [202].
• CYBERSHAKE acts as a seismology application concentrated on Southern California, particularly calculating Probabilistic Seismic Hazard curves for geographical areas [203]. Using the Probabilistic Seismic Hazard Analysis (PSHA) method, CYBERSHAKE performs finite difference simulations for generating Strain Green Tensors (SGTs) that are utilized to calculate synthetic seismograms for predicted ruptures. This process leads to the generation of spectral acceleration values and probabilistic hazard curves. The workload of CYBERSHAKE is known for its scalability and complexity, making it a perfect choice for assessing the proposed HH-ECO approach under challenging conditions.

Among the numerous workflow applications, the experimental model utilizes the Cybershake workflow to simulate the HH-ECO approach due to the composition of diversified domain tasks and complex workflow tasks. Moreover, the Cybershake acts as a collaborative platform in terms of collaborative workflow tasks regarding seismic hazards. Hence, to assess the performance of the proposed HH-ECO, the workflow tasks in the Cybershake are exploited as the data input in the cloud environment.

6.3 Modelling the Cloud Environment
Cloud computing resources are intricately interrelated and are dynamically scalable. CloudSim, in response, summarises these resources into a set of units, in which each unit denotes an example of a component in the data center. It has several PMs capable of using different virtualization methods to accommodate multiple VMs. Applications can be implemented into these VMs. The number of VMs can be dynamically adjusted over time [204][205]. The components that model the cloud environment within CloudSim:

• Data center: This unit includes a group of computing hosts that may differ in their hardware configurations, comprising cores, memory, capacity, as well as storage. Every host in CloudSim denotes a physical node in the cloud infrastructure, with predefined processing capacities expressed in MIPS.
• Cloud Coordinator: It acts as an extension of the data centre. The cloud coordinator manages the cloud structure and platform at a federation level. It observes the data centre elements’ internal state continuously and is involved in negotiations with cloud providers in the federation to overcome any unexpected spikes in resource requirements. Moreover, it confirms that agreed-upon SLAs are met by observing the execution lifecycle of applications.
• VM: This unit represents the VM in CloudSim, which is managed and hosted by the host components in the data centre. Every VM has particular characteristics like available storage size, processing capacity, memory, and internal provisioning policies (for example, time-shared or space-shared). Hosts can simultaneously instantiate several VMs and assign cores on the basis of these policies.

• Cloudlet: It represents the cloud-based application services containing tasks deployed and executed in the data centre. Each task is considered as the predefined data volume and instruction length, vital for effective hosting and execution.

• VM Provisioning: It models the policy for assigning VMs to hosts in the data centre. It selects an available host depending on VM deployment requirements, including storage, memory, and processing elements. CloudSim depends on a simple VM Provisioner policy that chooses the first available host while satisfying the VM deployment conditions.

• Memory Provisioning: It is accountable for allocating memory to VMs. It determines the physical memory space essential for deploying challenging VMs. When the memory provisioner recognizes the essential space for new VM deployment, the VM can be deployed and implemented.

• BW Provisioner: It simulates the provisioning policy for the bandwidth needed to deploy a VM on a host. It assigns network bandwidth for deployed VMs in the data centre, assisting the researchers in modifying it with specific policies like a priority and QoS to fulfil application requirements.

• Storage Allocation: It models the storage area in CloudSim intended for storing large data chunks. SANStorage acts as an interface for simulating retrieval and data saving. Bandwidth is crucial in assessing data transmission speed, in which extra time may be added to task execution because of data transfer delays across the network.

• Network Topology: It simulates the network behavior in CloudSim, using information generated by the BRIT topology generator for simulating the network topology.

• Sensor: It models a sensor that allows the cloud coordinator to observe specific parameters like resource utilization and energy consumption, providing valuable understanding for effective cloud management.

WorkflowSim has several other components designed to effectively simulate workflow applications, including the clustering engine, workflow engine, workflow mapper, clustering engine, workflow scheduler, failure generator, and failure monitor.

• Workflow Mapper: It enables the import of DAG files formatted in DAX, in addition to supplementary metadata like file size, from a workflow generator.
• Workflow Engine: It is responsible for handling task states. It releases tasks to the Clustering Engine and confirms that tasks are held until their parent tasks have effectively completed execution.

• Clustering Engine: It combines tasks into jobs to diminish scheduler overhead. Typically, a job is an atomic unit by the execution system that is made up of a set of tasks executed either in parallel or sequentially.

• Workflow Scheduler: Responsible for scheduling the jobs to available resources depending on predefined criteria set by users, supporting an accurate and reliable job-level execution model, including both space-shared and time-shared models.

• Failure generator: The failure generator produces the task failure randomly at the execution site during the job execution based on the condition of the user.

• Failure monitor: It supports the adjustment of the scheduling strategy using the records of the failed task during the execution.

The workflow model is based on the DAG model, enabling static and dynamic workflow schedulers. Initially, the workflow mapper brings the XML-formatted files and other metadata for mapping the abstract workflows to the concrete workflows and allocates the entire workflow into the execution site. The clustering engine groups the multiple tasks into a job for executing them in parallel or sequential form. After the job formation, the workflow engine allows the execution of the workflow task based on the parent-child relationship. Using the workflow scheduler, these workflows are matched with the appropriate resources based on users’ specifications. Moreover, the workflow supports the simulation of the failed tasks using the failure generator and monitor.

As a result, to implement the scientific workflow tasks with the phases of resource allocation, task scheduling, and VM migration, the experimental model simulates the HH-EECCO approach by the components above in the WorkflowSim tool.

6.4 Experimental Setup

The experiment employs Cyber Shake workflow applications [195] with sizes of 30, 50, 100, 150, and 180 workflow tasks. Even though the Cybershake workflow application comprises a huge number of tasks, WorkflowSim needs to be configured for effectively managing and arranging the numerous tasks. The configuration of WorkflowSim according to the task complexity is critical to adjust the parameters governing task scheduling and resource allocation properly. In particular, virtual machines and data centres are organized to manage the workload, referring to the establishment of available resource quantity for the simulation. In addition, resource management and task scheduling strategies confront the scalability constraint due to the voluminous workflow tasks. Hence, the experimental model employs the minimal size of 30, 50, 100, 150, and 180 workflow tasks for the simulation. Furthermore, practical constraints, such as time limitations
or hardware limitations, make it infeasible to process all 800,000 tasks at once. Focusing on a smaller subset allows researchers to make progress within these constraints. Both CloudSim and WorkflowSim provide parameterization abilities, enabling the researchers to adjust simulation parameters to reflect different cloud and workflow characteristics. The proposed HH-ECO method is developed to be flexible and configurable, thereby enabling it to possibly work across diverse cloud environments with changing configurations and requirements. Pricing for cloud services over different subscription periods is known as pricing policies. VM servers and pricing policies are based on Amazon EC2 IaaS configuration, and on-demand VM instances are small, medium, and large. The proposed implementation model represents the processing capacity of the processor using MIPS. The simulated cloud data centre consists of 10 host machines and 40 VMs involving three types such as small, medium, and large. Each type of host machine includes 3, 3, and 4 PMs, respectively, and the specifications of the host and VMs used in the simulated data centre are given in Table 6-1. The capacity of VMs refers the i) CPU capability as 500, 1000, and 1500 MIPS, ii) RAM capability as 256, 512, and 768 MB, and iii) bandwidth capability as 1 MB. To accomplish the simulation efficiency with minimal resource availability, the experimental model designs the cloud environment with a low range of resource capabilities compared to the real-time setup. The execution environment modelling becomes simple when a minimal number of resources are created with low resource capabilities. Instead of dealing with managing a lot of memory or processing power, modelling low resource capabilities enables the researchers to concentrate on the algorithm behaviors within the cloud environment. In particular, in shared or multi-tenant settings, VMs are frequently assigned with limited resources, which reflects the real-time infrastructure with low RAM, bandwidth, and CPU resources, as well as the quantity.

Table 6-1 Specifications of Host and VMs

<table>
<thead>
<tr>
<th>Cloud Resources</th>
<th>No.of machines</th>
<th>CPU/MIPS</th>
<th>RAM (MB)</th>
<th>Bandwidth (MB/s)</th>
<th>Storage (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host Machines</td>
<td>10</td>
<td>1000, 2000, 3000</td>
<td>1024, 2048, 3072</td>
<td>100</td>
<td>1</td>
</tr>
<tr>
<td>VMs</td>
<td>40</td>
<td>500, 1000, 1500</td>
<td>256, 512, 768</td>
<td>1</td>
<td>-</td>
</tr>
</tbody>
</table>

In Table 6-1, the Million Instruction Per Second (MIPS) measures the speed of the CPU, that is, the capability of the CPU to execute the total number of instructions within a second. RAM is faster to read and write content to and from the other storage due to its volatile nature. Moreover, it tends to access the storage locations in random orders. Bandwidth is defined as the maximum transmission capacity of the specific
connection in the unit of time. In other words, the bandwidth measures the maximum data flow through the connection and is measured as Megabits per second (Mbps).

### 6.4.1 Process of Simulation

To initialize the simulation, the goals and objectives of the simulation in the cloud are investigated. In this work, goals and objectives involve optimizing VM allocation, workflow scheduling, and VM migration for energy efficiency in a cloud environment. HH-ECO takes as input a subset of workflow tasks from the CyberShake to generate realistic workload scenarios and evaluate its performance. The HH-ECO algorithm is simulated within the simulation tools, such as the WorkflowSim, to experiment with the simulations. The simulation results are compared with the existing approaches to exhibit the improvement in performance. The parameters that impact the performance of the HH-ECO algorithm and the simulation results, like workflow characteristics, cloud resource availability, and optimization criteria, are selected. The parameters are adjusted based on empirical data to improve the accuracy of the simulations.

### 6.5 Evaluation metrics

The evaluation system employs a set of evaluation metrics to validate the efficiency of the proposed HH-ECO approach as follows:

**Makespan**

The total time is taken from the start time of the initial task to the end time of the final task of a sequence of tasks, which refers to the overall application completion time.

**Degree of disparity**

The measurement of unevenness among all the allocated hosts. The minimum value of the degree of disparity represents that the system is more balanced. Disparity among PMs can be expected due to their inherent differences in capacity and performance characteristics. This disparity arises because PMs vary in their hardware configurations, such as CPU, memory, and storage capacities. Consequently, when allocating tasks or VMs to these heterogeneous PMs, the workload distribution may not be uniform, leading to varying levels of utilization across the hosts. The heterogeneous PMs also enable the opportunity for a degree of disparity due to their diversified capabilities even when VMs are migrated. The degree of disparity is not limited to the homogeneous PMs, which is irrespective of the resource characteristics. The workload execution on the resources leverages the load imbalance among the resources.
Average energy consumption

During the simulation, the average consumed energy to run the complete application on the allocated cloud resources. The energy consumption of the computing server is based on the utilisation of the resources. Although an idle server, it also consumes energy to retain the running of memory, disks, and I/O resources. The remaining consumption linearly escalates when increasing the workload of resources.

SLA Violations

It depends on the number of violating tasks from the task deadline and the number of violating hosts from the optimal energy consumption level while executing the workflow application. The deviation from optimal energy consumption can lead to an SLA violation because it directly affects the performance and efficiency of the cloud infrastructure. Also, it impacts the overall QoS provided to users.

Total cost

The total cost spent to complete the entire application includes the amount spent on processing, memory, storage and transfer costs along with the Makespan.

Migration efficiency

The ratio between the number of useful migrations and the number of migrations throughout the execution of the application. Useful migrations refer to (i) when the underloaded host becomes idle and (ii) when the overloaded host becomes a balanced host after the migration completes.

Waiting Time Ratio (WTR)

The ratio between the waiting time of the tasks during the execution of the application and the overall completion time of the application. The minimum value indicates that the system is efficient. Compared to the overall execution time, the waiting time should be a minimum. High waiting time leads to unnecessary power consumption on both the user and cloud sides. The efficient allocation of VMs and tasks, as well as the workload, decides the waiting time. For example, a cloud-based application is assumed to have five tasks (T1, T2, T3, T4, and T5) that must be executed sequentially. Assume that the waiting time of each task during execution is 2 sec, 3 sec, 1 sec, 4 sec, and 2 sec for T1, T2, T3, T4, and T5, respectively, and the total execution time for the overall application is 100 sec. The total waiting time for all the tasks is 12 minutes. WTR is estimated as 12 sec / 100 sec = 0.12 sec. It indicates that 12% of the total execution time is consumed during waiting for tasks during execution. Ideally, this ratio must be close to zero, indicating less waiting time.
**SD Error**

SD error denotes the standard error related to the mean of the Makespan values for every system and size of the workflow. It indicates the inconsistency of the data points around the mean.

**P value**

P value denotes the statistical tests’ significance level that are performed for comparing the systems’ performance. When the P value is low, it shows a substantial difference in performance between the systems. Otherwise, there is no evidence to discard the null hypothesis of no difference in performance.

**Standard deviation**

Standard deviation is a statistical measure used for quantifying the amount of variation in a set of data points. It specifies the amount of deviation of the individual data points in a dataset from the mean of the dataset. If the standard deviation is low, data points are very much closer to the mean. Otherwise, they are spread out over an extensive range of values.

**6.6 Experimental results**

The simulation scenario conducts the following four tests: Test 1: Impact of the number of workflow tasks, Test 2: Impact of an average number of active hosts, Test 3: Impact of the percentage of resource utilisation, and Test 4: Impact of server behaviour. The experimental framework compares the results obtained by the HH-ECO approach with the existing approaches, such as the HHWS algorithm [76] and DDVM [59].

The comparative work [59] targets resource allocation in the cloud, which is similar to the first objective of the HH-ECO approach, and the comparative work [76] targets workflow task scheduling in the cloud, which is similar to the second objective of the HH-ECO approach. Hence, to comparatively exemplify the performance improvement of the proposed approach, the experimental model compares these two conventional approaches. The experiments of both the existing research works and the proposed approach follow the same evaluation scenario, and data inputs vary in their algorithm design in the context of resource allocation and task scheduling.

Even though HHWS determines the optimal solution for the workflow task by combining a PSO algorithm with a GSA, it partially applies the mutation to the workflow tasks regardless of the optimal global solution. In addition, DDVM merely takes into account reactive VM management with the concern of minimising SLA violations and power consumption. HH-ECO differentiates itself from HHWS and DDVM by overcoming a broader spectrum of challenges in cloud environments. Even
though HHWS and DDVM emphasise particular aspects such as workflow scheduling and dynamic VM management, HH-ECO includes VM allocation, task scheduling, and VM migration. By integrating optimization techniques and heuristics, the proposed HH-ECO approach optimizes the allocation of VMs to PMs, efficiently schedules workflow tasks, and dynamically performs VM migration. It ensures the workflows’ timely execution and the optimal utilization of cloud resources while meeting SLAs. On the other hand, HHWS and DDVM may outperform in their corresponding domains. Still, it may lack the HH-ECO’s holistic perspective, thereby possibly limiting its capability to solve the multifaceted challenges faced in cloud computing environments. In addition, both the existing researches [59, 76] fail to focus on the dominant tasks during the task execution in the cloud environment. They are also lacking in focussing on the adaptive workflow execution during the scheduled or allocated tasks in the corresponding resources, leading to the increased energy cost. Hence, the proposed HH-ECO approach is compared against these approaches to show how it outperforms HHWS and DDVM.

6.6.1 Comparative Study of makespan, average energy consumption, SLA violation, and waiting time ratio

Table 6-2 shows the performance comparison and error analysis of the proposed and existing approaches in terms of makespan, average energy consumption, SLA violation, and waiting time ratio.

Table 6-2 Performance comparison of makespan, average energy consumption, SLA violation, and waiting time ratio

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Sub-metrics</th>
<th>HH-ECO</th>
<th>HHWS</th>
<th>DDVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Makespan (sec)</td>
<td>Workflow</td>
<td>30</td>
<td>239</td>
<td>375</td>
</tr>
<tr>
<td></td>
<td></td>
<td>50</td>
<td>372</td>
<td>603</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>413</td>
<td>658</td>
</tr>
<tr>
<td></td>
<td></td>
<td>150</td>
<td>458</td>
<td>718</td>
</tr>
<tr>
<td></td>
<td></td>
<td>180</td>
<td>508</td>
<td>783</td>
</tr>
<tr>
<td>SD error</td>
<td></td>
<td>44.04461</td>
<td>78.8286</td>
<td>609.6206</td>
</tr>
<tr>
<td>P value</td>
<td></td>
<td>0.008842</td>
<td>0.011381</td>
<td>0.85453</td>
</tr>
<tr>
<td>Standard deviation</td>
<td></td>
<td>102.3255</td>
<td>156.2508</td>
<td>1335.504</td>
</tr>
<tr>
<td>Average Energy Consumption</td>
<td>Workflow</td>
<td>30</td>
<td>83291</td>
<td>189947</td>
</tr>
<tr>
<td>(Watt-sec)</td>
<td></td>
<td>50</td>
<td>238479</td>
<td>1229315</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>680834</td>
<td>3598444</td>
</tr>
<tr>
<td></td>
<td></td>
<td>150</td>
<td>1943717</td>
<td>5361327</td>
</tr>
<tr>
<td></td>
<td></td>
<td>180</td>
<td>5549130</td>
<td>8524210</td>
</tr>
<tr>
<td>SD error</td>
<td></td>
<td>1279789.909</td>
<td>672894.9092</td>
<td>619172.9419</td>
</tr>
<tr>
<td>P value</td>
<td></td>
<td>0.297732041</td>
<td>0.097731709</td>
<td>0.050149151</td>
</tr>
<tr>
<td>Standard deviation</td>
<td></td>
<td>2272866.314</td>
<td>3333000.345</td>
<td>3121617.868</td>
</tr>
</tbody>
</table>
Performance measures with varying numbers of workflow tasks are shown in Figure 6-1, Figure 6-2, Figure 6-3 and Figure 6-4.

![Figure 6-1 Makespan Vs. No. of workflow tasks](image)

Figure 6-1 exhibits the comparison of the makespan taken from the proposed HH-ECO approach and the existing approaches. The results for the Makespan represent the significant performance improvements attained by the proposed HH-ECO approach when compared to both HHWS and DDVM across all workflow sizes (30, 50, 100, 150, 180). For the 30-task workflow, HH-ECO lessens the Makespan by 36.27% when compared to HHWS and 63.08% when compared to DDVM. Likewise, for the 50-task workflow, HH-ECO reduces the Makespan by 38.2% and 36.0% when compared
to HHWS and DDVM, respectively. For the workflow with 180 tasks, HH-ECO attains a notable reduction of Makespan of 35.2% and 84.2% than HHWS and DDVM, respectively. These results show the superior efficiency and effectiveness of HH-ECO in improving workflow execution time, thereby achieving considerable reductions in Makespan across different workflow sizes. Further, the HH-ECO approach maintains the application workflow, reduces the waiting time for the succeeding task, and ensures optimal execution of the workflow tasks. The SD error value of HH-ECO is lower (44.04461) compared to HHWS and DDVM, emphasizing higher consistency in the performance of HH-ECO. In terms of p-values, HH-ECO (0.008842) shows statistically significant differences than HHWS (0.011381) and DDVM (0.85453), indicating more reliable performance results. HH-ECO has the lowest deviation (102.3255), followed by HHWS and DDVM, indicating the stable performance of HH-ECO across different scenarios. Even though HHWS constantly proves promising metrics, especially in makespan, HH-ECO shows more reliable and steady performance across different evaluation criteria, highlighting its suitability for different workflow scenarios.

Figure 6.2 Average energy consumption Vs. No. of workflow tasks

Figure 6.2 illustrates the average energy consumption across different workflows (30, 50, 100, 150, and 180) for HHWS, DDVM, and the proposed HH-ECO. In all these scenarios, HH-ECO reliably outperforms both HHWS and DDVM, showing significantly lesser energy consumption percentages. For example, with 30 workflow tasks, HH-ECO achieves a reduction of 43.85% compared to HHWS and 37.84% compared to DDVM. For the workflow with 180 tasks, HH-ECO attains a notable reduction of average energy consumption of 34.91% and 84.2% than HHWS and DDVM, respectively. This trend continues across other workflow sizes of (50, 100, 120), with HH-ECO consistently representing superior energy efficiency. Since these results emphasize the efficacy of the proposed HH-ECO approach in lessening energy
consumption, it becomes a compelling choice for cloud resource management. This is because HH-ECO initially allocates VMs to the appropriate hosts energy-efficiently and then optimally schedules the workflow tasks on the allocated VM resources using the C-PSO algorithm. Moreover, it applies mutation on the scheduled tasks to diminish the dominance in processing time, thus mitigating resource wastage and conserving considerable energy. On the other hand, the DDVM approach lacks in assigning the optimal maximum and minimum utilisation level of the resource, which makes the host become stressed. In the context of energy-efficient modelling, the stress resources are either underloaded or overloaded, referring to the imbalanced resources. Consequently, there is an indication that stress resources are to be migrated to the appropriate destination resources. The SD error value of HH-ECO is lower (1279789.909) compared to HHWS and DDVM, emphasizing higher consistency in the energy efficiency of HH-ECO. In terms of p-values, HH-ECO (0.297732041) shows statistically significant differences from HHWS and DDVM, indicating more reliable performance outcomes. HH-ECO has the lowest standard deviation (2272866.314), followed by HHWS and DDVM, indicating minimum variability in energy consumption data.

![Figure 6-3 SLA violations Vs. No. of workflow tasks](image)

Figure 6-3 SLA violations Vs. No. of workflow tasks

Figure 6-3 reveals the SLA violation ratio in the HH-ECO, HHWS, and DDVM approaches while varying the number of workflow tasks from 30 to 180. The SLA violations inversely expose the effectiveness of the system. Even when the number of workflow tasks with unique characteristics increases, the proposed approach constantly demonstrates minimum SLA violation rates that range between 0.127 and 0.194. On the other hand, both HHWS and DDVM show higher SLA violation rates, with HHWS usually performing slightly better than DDVM. For example, at 30 tasks, HH-ECO exhibits a violation rate of 0.194, while DDVM and HHWS have rates of 0.532 and 0.43, respectively. These differences continue across all task counts, representing the
superior SLA compliance of the HH-ECO method. The unique consideration of the mutation in HH-ECO fetches an accurate global solution throughout the application execution. However, HHWS fails to consider the dominance relationship. DDVM implements only the traditional scheduling of the first-fit heuristic method, which ensures minimum waiting time, whereas HH-ECO and HHWS can filter out the local optimal solutions. Hence, the performance of HHWS is weaker than HH-ECO. After that, statistical measures are evaluated to support the reliability and significance of the data additionally. The SD error is slightly higher than DDVM, implying lesser consistency. P values for HH-ECO (0.006358) are higher than HHWS and DDVM, indicating lower statistical significance in the measurements. HH-ECO has a slightly higher standard deviation (0.0241) than DDVM, indicating a little variability in SLA compliance data.

![Figure 6-4 Waiting Time Ratio (WTR) Vs. No. of workflow tasks](image-url)

Figure 6-4 illustrates the WTR of HH-ECO, HHWS, and DDVM approaches while varying the number of workflow tasks (30 to 180). The HH-ECO approach effectively manages the waiting time of each task even when the number of tasks in the workflow is high by applying the C-PSO algorithm with an adaptive mutation on the scheduled tasks. At a workflow of 30 tasks, HH-ECO shows a remarkably lower waiting time ratio of 0.576 than HHWS and DDVM, which have ratios of 1.31 and 1.54, respectively. However, when the workflow size is increased to 180 tasks, HH-ECO has a considerable increase in waiting time ratio to 6.71, which is less than HHWS and DDVM. However, during the tasks (50, 100, 150), the waiting time ratio of HH-ECO is more than HHWS. This is because HH-ECO prioritizes metrics such as makespan, SLA violation, and energy efficiency rather than minimizing waiting times. While focusing on other performance metrics, HH-ECO’s scheduling decisions cause slightly extended waiting times for certain tasks compared to HHWS. Even though the WTR is higher in the HH-ECO compared to the HHWS approach, the proposed HH-ECO optimally mitigates the
energy cost by minimizing the makespan, thereby increasing the energy efficiency of workflow task execution in the cloud. This highlights the significance of considering workflow complexity in resource allocation and task scheduling strategies to reduce the waiting time ratio, which will be considered in the future. SD error and P values for HH-ECO are higher than HHWS, but lower than DDVM, indicating lesser consistency and statistical significance in the waiting time measurements. HH-ECO has a lower standard deviation (0.0241) than HHWS and DDVM, indicating stability in SLA compliance data in SLA compliance data.

6.6.2 Impact of an Average Number of Active Hosts

Table 6-3 shows the performance comparison and error analysis of the proposed and existing approaches in terms of total cost, average energy consumption, and Degree of Disparity based on the number of active hosts.

Table 6-3 Performance comparison of total cost, average energy consumption, and Degree of Disparity

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Sub-metrics</th>
<th>HH-ECO</th>
<th>HHWS</th>
<th>DDVM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Total Cost ($)</strong></td>
<td>Number of Active Hosts</td>
<td>2</td>
<td>1324.22</td>
<td>1785.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>1343.39</td>
<td>1812</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td>1382.1</td>
<td>1892.25</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
<td>1431.9</td>
<td>2105.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>1463.25</td>
<td>2140.12</td>
</tr>
<tr>
<td>SD error</td>
<td></td>
<td></td>
<td>8.952515</td>
<td>55.13399</td>
</tr>
<tr>
<td>P value</td>
<td></td>
<td></td>
<td>8.72E-07</td>
<td>9.52E-05</td>
</tr>
<tr>
<td>Standard deviation</td>
<td></td>
<td></td>
<td>58.47606</td>
<td>165.6717</td>
</tr>
<tr>
<td><strong>Degree of Disparity</strong></td>
<td>Number of Active Hosts</td>
<td>2</td>
<td>3.31</td>
<td>3.33</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>3.34</td>
<td>3.36</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td>3.35</td>
<td>3.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
<td>3.4</td>
<td>3.63</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>3.53</td>
<td>3.84</td>
</tr>
<tr>
<td>SD error</td>
<td></td>
<td></td>
<td>0.041312</td>
<td>0.056184</td>
</tr>
<tr>
<td>P value</td>
<td></td>
<td></td>
<td>5.29E-06</td>
<td>1.45E-05</td>
</tr>
<tr>
<td>Standard deviation</td>
<td></td>
<td></td>
<td>0.086776</td>
<td>0.20969</td>
</tr>
<tr>
<td><strong>Average Energy Consumption (Watt-Sec)</strong></td>
<td>Number of Active Hosts</td>
<td>2</td>
<td>82965</td>
<td>124212.51</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
<td>83291</td>
<td>138515.63</td>
</tr>
<tr>
<td></td>
<td></td>
<td>6</td>
<td>93405</td>
<td>156465.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>8</td>
<td>114195</td>
<td>173212.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>134926</td>
<td>189947.17</td>
</tr>
<tr>
<td>SD error</td>
<td></td>
<td></td>
<td>8184.735785</td>
<td>969.0258099</td>
</tr>
</tbody>
</table>
Performance with a varying average number of active hosts is represented in Figure 6-5, Figure 6-6 and Figure 6-7.

As for Test 2 (Impact of an average number of active hosts), the total costs of the HH-ECO, HHWS, and DDVM approaches with 100 workflow tasks are shown in Figure 6-5, when the average number of active hosts varies from 2 to 10. While simulations allow for flexibility in scaling up the number of hosts, there may be computational or resource constraints that limit the feasibility of simulating larger numbers of hosts. Total cost is proportional to time spent and resource utilisation, including processing, memory, storage, and bandwidth. These costs include expenses associated with processing, memory usage, storage, and bandwidth utilization. The total cost represents the cumulative expenses incurred throughout the execution of the workflow applications on the remote server. Figure 6-5 displays the total cost for different numbers of active hosts (2, 4, 6, 8, and 10) in the HHWS, DDVM, and proposed HH-ECO approach. While varying the number of active hosts, the total cost across the approaches changes accordingly. In the HHWS approach, the total cost ranges from $1785.1 (2 active hosts) to $2140.12 (10 active hosts). DDVM shows a slightly higher total cost than HHWS, ranging from $2107.73 to $2315.25 (with active hosts 2 to 10).

On the other hand, the proposed HH-ECO approach proves the minimum total cost, starting from $1324.22 with 2 active hosts and increasing gradually to $1463.25 with 10 active hosts. This shows the efficiency of the proposed HH-ECO approach in lessening the total cost of execution across when varying numbers of active hosts, thus
showcasing potential cost savings of up to 31.6% than HHWS and 36.74% than DDVM with 10 active hosts. However, it is vital to consider the related errors and statistical significance. In HH-ECO, lower SD error (8.9525) and P values (8.72E-07) show greater consistency and significance in the cost measurements. HH-ECO has a low standard deviation (58.476) compared to HHWS and DDVM, representing the stability in total cost data.

Figure 6-6 shows the Degree of Disparity for different numbers of active hosts (2, 4, 6, 8, and 10) in the HHWS, DDVM, and proposed HH-ECO approaches. In the HHWS approach, the Degree of Disparity varies between 3.33 and 3.84 when the number of active hosts rises, representing moderate load balancing. DDVM shows relatively lower values ranging between 1.8 and 3.87, representing better load balancing than HHWS but still having some disparity. This is because the DDVM approach can balance the active hosts only while using a minimum number of resources. On the other hand, the proposed HH-ECO approach has a more consistent Degree of Disparity that ranges between 3.31 and 3.53, representing better load balancing with different numbers of active hosts. The SD error values are relatively small for HH-ECO (0.0413) and HHWS (0.0562), indicating low uncertainty in the total cost measurements. Additionally, the P values are very low for HH-ECO (5.29E-06) and HHWS (1.45E-05), representing high statistical significance in the observed differences in total estimated costs. Also, the standard deviation for HH-ECO (0.0868) is relatively smaller compared to HHWS (0.2097) and DDVM (0.8378), further supporting the reliability of cost data in HH-ECO.
Figure 6-7 Average energy consumption vs. No. of Hosts

Figure 6-7 illustrates the average energy consumption of HH-ECO, HHWS, and DDVM approaches. The advantage of exploiting the C-PSO algorithm with an adaptive mutation on the workflow task execution creates a significant impact on the application completion. For example, at 2 active hosts, HH-ECO utilizes 33.19% and 38.10% less energy than HHWS and DDVM, respectively. This trend continues when the number of active hosts rises, with HH-ECO showing minimum energy consumption percentages than the other existing methods. At 10 active hosts, HH-ECO takes 28.96% and 35.45% less energy than HHWS and DDVM, emphasizing its efficiency in resource utilization. The results highlight the effectiveness of the proposed HH-ECO method in reducing energy consumption across changing the numbers of active hosts, indicating its potential for energy-efficient cloud resource management. Notably, HH-ECO migrates the tasks when the allocated resources are either overloaded or underloaded conditions during execution. Moreover, the idle server utilisation policy also assists in minimising energy consumption. The relatively higher SD error (8184.74) and P values (0.00565) for HH-ECO compared to HHWS and DDVM show more uncertainty and more statistical significance in the total cost measurements. Detailed error analysis shows that HH-ECO maintains a smaller standard deviation (22465.39) than HHWS and DDVM, thereby indicating its reliability in cost estimation. It highlights the potential benefits of implementing HH-ECO over other techniques in terms of cost optimization and resource efficiency.

6.6.3 Impact of The Percentage of Resource Utilization

Table 6-4 shows the performance comparison and error analysis of the proposed and existing approaches in terms of makespan, Degree of Disparity, Migration Efficiency, and SLA Violation based on resource utilization.

Table 6-4 Performance comparison of total cost, average energy consumption, and
### Degree of Disparity

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Sub-Metrics</th>
<th>HH-ECO</th>
<th>HHWS</th>
<th>DDVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Makespan (s)</td>
<td>Percentage of Resource Utilization</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>274.03</td>
<td>358.21</td>
<td>648.53</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>295.13</td>
<td>375.04</td>
<td>583.96</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>372.5</td>
<td>603.51</td>
<td>672.1</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>413.34</td>
<td>658.5</td>
<td>715</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>454.18</td>
<td>713.49</td>
<td>757.9</td>
</tr>
<tr>
<td></td>
<td>SD error</td>
<td>13.61616</td>
<td>56.60114</td>
<td>41.76888</td>
</tr>
<tr>
<td></td>
<td>P value</td>
<td>0.000608</td>
<td>0.026207</td>
<td>0.000977</td>
</tr>
<tr>
<td></td>
<td>Standard deviation</td>
<td>76.57248</td>
<td>164.6349</td>
<td>66.08427</td>
</tr>
<tr>
<td>Degree of Disparity</td>
<td>Percentage of Resource Utilization</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>1.79</td>
<td>2.32</td>
<td>1.81</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>2.54</td>
<td>3.85</td>
<td>2.61</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>3.3</td>
<td>5.21</td>
<td>3.85</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>4.3</td>
<td>5.32</td>
<td>4.4</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>4.9</td>
<td>5.47</td>
<td>5.21</td>
</tr>
<tr>
<td></td>
<td>SD error</td>
<td>0.096401936</td>
<td>0.641152608</td>
<td>0.179916647</td>
</tr>
<tr>
<td></td>
<td>P value</td>
<td>0.002388664</td>
<td>0.052171281</td>
<td>0.013150112</td>
</tr>
<tr>
<td></td>
<td>Standard deviation</td>
<td>1.264507809</td>
<td>1.34819509</td>
<td>1.367106433</td>
</tr>
<tr>
<td>Migration Efficiency</td>
<td>Percentage of Resource Utilization</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.95</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>1</td>
<td>-</td>
<td>0.5</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>0.81</td>
<td>-</td>
<td>0.66</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>0.82</td>
<td>-</td>
<td>0.73</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>0.89</td>
<td>-</td>
<td>0.78</td>
</tr>
<tr>
<td></td>
<td>SD error</td>
<td>0.077287</td>
<td>-</td>
<td>0.207131</td>
</tr>
<tr>
<td></td>
<td>P value</td>
<td>0.001203</td>
<td>-</td>
<td>0.035031</td>
</tr>
<tr>
<td></td>
<td>Standard deviation</td>
<td>0.082037</td>
<td>-</td>
<td>0.182428</td>
</tr>
<tr>
<td>SLA Violation</td>
<td>Percentage of Resource Utilization</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>0.185</td>
<td>0.39</td>
<td>0.532</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>0.194</td>
<td>0.43</td>
<td>0.548</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>0.125</td>
<td>0.34</td>
<td>0.544</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>0.161</td>
<td>0.34</td>
<td>0.625</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>0.165</td>
<td>0.41</td>
<td>0.658</td>
</tr>
<tr>
<td></td>
<td>SD error</td>
<td>0.027803477</td>
<td>0.046296148</td>
<td>0.025160154</td>
</tr>
<tr>
<td></td>
<td>P value</td>
<td>0.007579617</td>
<td>0.003827507</td>
<td>0.000356463</td>
</tr>
<tr>
<td></td>
<td>Standard deviation</td>
<td>0.02670206</td>
<td>0.040865633</td>
<td>0.056398582</td>
</tr>
</tbody>
</table>

Performance with varying percentages of resource utilisation is represented in Figure 6-8, Figure 6-9, Figure 6-10 and Figure 6-11.
Figure 6-8 Makespan vs. Percentage of Resource Utilization

As shown in Figure 6-8, the Makespan increases while increasing the percentage of resource utilization (CPU performance, memory, storage, and bandwidth) by a specific application. This result is taken for 100 tasks and 10 hosts. Among the three comparative approaches, the HH-ECO approach significantly reduces the Makespan compared to HHWS and DDVM because it focuses on maintaining the optimal utilisation of each server at a specific level throughout the execution of the workflow application. Thus, the HH-ECO approach is efficient enough to complete the execution of workflow applications in a reasonable time. At a load utilization of 5%, HH-ECO achieves a makespan of 274.03 s, whereas HHWS and DDVM have significantly higher makespan values of 358.21 s and 648.53 s, respectively. This trend continues across increasing load utilizations (10%, 15%, 20%, 25%), with HH-ECO constantly outstanding the other approaches by significant margins. The results highlight the effectiveness of HH-ECO in optimizing load utilization and lessening makespan, highlighting its potential to improve workflow scheduling efficiency in cloud computing environments. Additionally, by considering the error analysis, HH-ECO shows significantly lower SD errors (13.616) than HHWS and DDVM, suggesting greater consistency in makespan measurements. The lower p-values for HH-ECO (0.000608) compared to HHWS and DDVM indicate higher statistical significance, thus showing the reliability of the makespan measurements. However, the standard deviation is slightly higher for HH-ECO than DDVM, representing less stability in makespan measurements.
Figure 6-9 illustrates the value of DD for the HH-ECO approach compared to the HHWS and DDVM techniques. HH-ECO shows a lower DD than DDVM and HHWS approach across all load utilization levels (5%, 10%, 15%, 20%, 25%). For instance, at a load utilization of 5%, the proposed HH-ECO exhibits a degree of disparity of 1.79, which is less than DDVM and HHWS. Similarly, as the load utilization increases to 25%, HH-ECO still maintains the lowest degree of disparity at 4.9, followed by DDVM and HHWS. This indicates that the proposed HH-ECO approach effectively balances the workload across hosts, resulting in a lower degree of disparity compared to both HHWS and DDVM, ensuring more equitable resource allocation and utilization. Overall, HH-ECO demonstrates superior performance in managing load distribution, leading to more efficient resource utilization and improved system stability. HH-ECO exhibits significantly lower SD errors (0.0964) compared to HHWS and DDVM, suggesting greater consistency in DD measurements. The lower p-values for HH-ECO (0.0024) compared to HHWS and DDVM indicate higher statistical significance, further indicating the reliability of the DD values. However, the standard deviation is lower for HH-ECO (1.2645) than DDVM, representing more stability in DD data.
Figure 6-10 illustrates the comparison of migration efficiency between the existing DDVM and the proposed HH-ECO approach across different load utilization levels (5%, 10%, 15%, 20%, and 25%). Figure 6-10 shows the improvement in the Migration efficiency of HH-ECO compared with the DDVM approach when varying the resource utilisation percentage. The drop in DDVM efficiency to 0.5 at 10% resource utilization happens since the DDVM approach is not effective at managing low resource utilization levels. Although DDVM has a fair migration efficiency of 5%, it drops to 50% at the percentage of resource utilization (10%). After that, migration efficiency gradually increases. DDVM faces challenges in accurately managing the workload, causing a sudden decrease in efficiency. This is because the DDVM applies shutdown and freeze, which leads to providing an inaccurate solution when the load percentage is high. As HHWS did not consider the migration, this figure does not take it for comparison. At a load utilization of 5%, DDVM shows a migration efficiency of 100%, whereas the proposed HH-ECO method slightly decreases to 95%. However, when the load utilization increases, the proposed HH-ECO method performs well, showcasing its effectiveness in managing migrations efficiently. HH-ECO exhibits significantly very low SD errors (0.077287) compared to DDVM, suggesting greater consistency in migration efficiency measurements. The considerably lower p-values for HH-ECO (0.001203) than DDVM indicate higher statistical significance, further indicating the reliability of the migration efficiency values. However, the standard deviation is lower for HH-ECO (0.082037) than DDVM, representing more stability in the data.
Figure 6-11 illustrates the SLA violation percentages across varying load utilizations for HHWS, DDVM, and HH-ECO approaches. The HHWS approach applies the PSO and GSA algorithms on two sets of tasks of one application. When the percentage of resource utilisation is high, the HH-ECO approach reschedules the tasks within a reasonable time to complete the execution of workflow application based on the C-PSO algorithm. Thus, the independent identification of the optimal solution is not useful in all cases because it is likely to misclassify while identifying the optimal solution within a set of tasks and resources, which is likely to extend the overall processing time, resulting in many violations in response time and energy. DDVM lacks in scheduling the tasks on the optimal resources as the allocation is based on the first-fit heuristic algorithm, which dynamically allocates the resources based on the current availability of resources in which the local solution appears as the optimal global solution. At a load utilization of 5%, HH-ECO shows the lowest SLA violation rate at 0.185, compared to HHWS (0.39) and DDVM (0.532). It continues across increasing load utilisations, with HH-ECO consistently performing better than HHWS and DDVM. Particularly, at a load utilization of 25%, HH-ECO achieves a low SLA violation rate of 0.165, while HHWS and DDVM register 0.41 and 0.658, respectively. These results highlight the efficiency of HH-ECO in lessening SLA violations across varying load utilizations, thus providing significant improvement compared to the existing methods. HH-ECO exhibits significantly low SD errors (0.027803477) but slightly higher than DDVM (0.025160154), suggesting greater consistency in SLA violation data. P-values for HH-ECO (0.007579617) are higher than HHWS, and DDVM indicates less statistical significance in the SLA violation values. However, the standard deviation is lower for HH-ECO (0.02670206) than DDVM and HHWS, representing more stability in the data.
### 6.6.4 Impact of Server Behavior

Table 6.5 shows the results obtained from the proposed HH-ECO approach in terms of Average energy consumption and Average CPU Utilization based on the Server Behavior.

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Sub-metrics</th>
<th>Proposed HH-ECO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average CPU Utilization(%)</td>
<td>Categorization of server behavior</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>10.4</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>287.45</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>358.97</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>417.26</td>
</tr>
<tr>
<td>SD error</td>
<td></td>
<td>5.511261</td>
</tr>
<tr>
<td>P value</td>
<td></td>
<td>0.126837</td>
</tr>
<tr>
<td>Standard deviation</td>
<td></td>
<td>30.51704</td>
</tr>
<tr>
<td>Average Energy Consumption</td>
<td>Categorization of server behavior</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>7.16</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>25.82</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>50.38</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>79.64</td>
</tr>
<tr>
<td>SD error</td>
<td></td>
<td>83.1052</td>
</tr>
<tr>
<td>P value</td>
<td></td>
<td>0.645879</td>
</tr>
<tr>
<td>Standard deviation</td>
<td></td>
<td>180.0825</td>
</tr>
</tbody>
</table>

![Figure 6-12 Average CPU utilization vs. Categorization of server behavior](image)

The evaluation results in terms of Average CPU utilisation and Average energy consumption for different server behaviours are shown in Figure 6-12 and Figure 6-
the server behaviours such as idle, under load, balanced, and overload are graphically represented as 1, 2, 3, and 4, respectively. The average CPU utilisation of the allocated resources under various server behaviours is illustrated in Figure 6-12. The HH-ECO approach optimally utilises the CPU resources by applying an adaptive mutation method when one solution dominates. This process averts the local convergence of the solutions as it determines the global solution in both the workflow application and cloud resources. When there are sudden changes in the allocated resources, such as the state of underload or overload during the execution of workflow application, the proposed approach efficiently handles the tasks without violating the SLAs through adaptive mutation and task migration among the available resources. By effectively exploiting the optimal level of resource utilisation, the HH-ECO approach obtains average CPU utilisation for idle, underload, balanced, and overloaded servers as 10.4%, 287.45%, 358.97%, and 417.26%, respectively, during application execution. Existing methods are not shown here because the purpose is to prove the performance of HH-ECO in terms of idle, underloaded, balanced, and overloaded servers rather than making comparative analyses with the existing methods. Error analysis further explains the measurements’ reliability. The SD error of 5.511261 indicates some uncertainty in the average CPU utilization evaluations, although it is within a relatively narrow range. Moreover, the P value of 0.126837 indicates a moderate statistical significance, suggesting a reasonable level of confidence in the differences in CPU utilization across different server behaviour categories. However, the standard deviation is higher for HH-ECO (30.51704), representing less stability in the average CPU utilization data.

![Figure 6-13 Average energy consumption Categorization of server behavior](image)

Figure 6-13 illustrates the increase in average energy consumption of the HH-ECO approach while varying the server categorisation from the idle state to overload.
conditions. When the server is in an idle state, the energy consumption of the server is minimal regardless of the workload. The HH-ECO approach consumes reasonable energy as it considers the limited access level of the server resources rather than utilising the entire available resources. The consideration of idle server restarts during the execution of workflow applications facilitates less consumption of energy. The average energy consumption of the HH-ECO approach under the various server states of idle, under load, balanced, and overload is 7.16%, 25.82%, 50.38%, and 79.64%, respectively. The SD error of 83.1052 indicates some uncertainty in the average energy consumption evaluations, although it is within a relatively narrow range. Moreover, the P value of 0.645879 indicates a more statistical significance, suggesting better confidence in the differences in average energy consumption across different server behaviour categories. However, the standard deviation is higher for HH-ECO (180.0825), representing less stability in the average energy consumption data.

6.6.5 Cost-Benefit Analysis
The initial investment for setup, such as hardware, software licenses, and configuration costs, will be almost similar for HHWS, DDVM, and the proposed HH-ECO approach. Operational costs may vary depending on usage and service levels. HH-ECO has lower operational expenses because of its energy-efficient approach. When migrating from existing infrastructure, there may be additional costs associated with data migration. Because of efficient migration, the proposed HH-ECO has lower migration costs. HH-ECO shows the lowest total cost of $1324.22 compared to HHWS and DDVM ($1785.1 and $2107.73, respectively) when the number of active hosts is 2. Even when the number of active hosts is 10, HH-ECO shows the lowest total cost of $1463.25 compared to HHWS and DDVM ($2140.12 and $2315.25, respectively). Hence, the total cost required by the system is very less even when it achieves better performance compared to other existing methods. The proposed HH-ECO approach provides scalability, allowing the providers to adjust the resources based on demand, leading to further cost savings by optimizing resource utilization.

6.7 Summary
A hybrid workflow scheduling algorithm, HH-ECO, for energy-efficient CC has been proposed in this thesis. The approach focuses on accomplishing multi-objectives via resource allocation, task scheduling, and resource migration phases underpinned by applying the C-PSO algorithm with an adaptive mutation. It builds the system as an energy-efficient green cloud system through optimal allocation. It avoids the dominant solution among other workflow tasks that ensure the optimal global solution rather than local convergence solutions. Moreover, the scheduling of the workflow tasks is performed dynamically based on the scheduling plan, workflow model, and the optimal global solution. Also, the HH-ECO migrates the overloaded or underloaded resources during the execution of workflow applications, facilitating an
improved QoS and an energy-efficient cloud environment. The key achievements of the proposed methodology are:

- The Hybrid Heuristic Workflow algorithm provides an energy-efficient virtualised environment with high reliability and efficiency in resource allocation.
- Hybrid Heuristic algorithms efficiently allocate heterogeneous VMs to PM based on their capacity.
- The C-PSO plans the scheduling procedure of workflow application to utilise the advantages of a green cloud environment; the mutation process improves the potential profit without violating SLAs.
- The Chaotic iterations in PSO avoid earlier standstill of particles before reaching global optima to provide a near-global optimal solution to promise user demands and achieve optimal energy consumption.
- C-PSO with adaptive mutation characterises the resource utilisation of VMs and generates the globally best migration plan to optimise the Makespan and cost.

The experiment shows the benefit of combining the chaotic and PSO algorithm with an adaptive mutation while implementing VM allocation, task scheduling, and migration. The experimental results of the HH-ECO approach demonstrate improved performance, accomplishing an optimal Makespan of 38.27% and significant energy conservation of 38.06% when compared with existing CC approaches such as HHWS and DDVM. Moreover, the result validates that the proposed technique reduces the cost consumption and performance degradation than the HHWS and DDVM. Moreover, the HH-ECO approach considers the power consumption of the CPU and RAM along with the cooling system and network racks. Hence, gathering details of various power consumers in the HH-ECO approach greatly assists in reducing energy consumption. In future work, it is essential to handle the resource provisioning for the dynamic incoming of many users while maintaining optimal energy consumption. In addition, VM migration among various data centres needs to be investigated along with the network latency.
CHAPTER 7 CONCLUSIONS

With the growing reliance on computers and computing power, implementing CC has become vital in today's global environment. CC involves consolidating several VMs within physical servers, providing significant benefits to users and data centres. However, it is challenging in terms of energy consumption and QoS, requiring the development of new optimisation strategies for GCC. Developing a novel research methodology for energy-efficient practices and technologies is vital for overcoming these challenges for maintaining high performance. Scientific workflows are particularly significant in CC, which requires effective energy management in scientific workflows for minimising operational costs and enhancing the sustainability of computational resources. It ensures that the demand for high computational power must be fulfilled without excessive energy utilisation. For large-scale applications running on heterogeneous PMs, efficient resource allocation is vital for task interdependencies in scientific workflows.

To address these problems, this thesis proposed two methodologies, including the analysis of energy models in the cloud using a practical server with various load categories and hybrid heuristics-based energy-efficient optimisation of scientific workflow validated using cloudsim. Initially, a practical experiment is conducted utilising a high-end server and several standard servers equipped with numerous VMs. This setup aims to analyse and address issues related to consolidation, load balancing, and energy consumption within VMs and physical server configurations. Based on the load categories analysed, a novel HH-ECO approach is introduced to enhance energy efficiency and performance.

The analysis of energy models is discussed in CC using a practical server with various load categories. It uses high-end and standard servers, each hosting multiple VMs, to analyse load balancing and energy consumption. By observing server behavior under several load conditions, key parameters are identified that affect performance and energy usage. The experimental setup uses VMware ESXi hypervisors to manage several VMs across three standard servers and one high-end server, using VMware's vSphere and vCenter for configuration and monitoring. The practical testbed evaluates server performance and energy consumption under different load categories for determining optimal configurations. This analysis presents detailed
results on memory, CPU, and power consumption for every server category. An idle server exhibited low CPU and memory utilisation, utilising 60%-75% of its total power capacity. An underloaded server exhibited a maximum utilisation of around 40% with more energy consumption. A balanced server showed optimal performance with CPU utilisation of about 63% and memory utilisation of about 58%, demonstrating that it could handle additional VMs. On the other hand, an overloaded server showed high CPU utilisation, memory utilisation, and peak power consumption, emphasising the requirement for VM migration to prevent performance degradation.

Based on the analysed load categories, a hybrid workflow scheduling algorithm, HH-ECO, for energy-efficient CC was proposed. The approach focused on accomplishing multi-objectives via resource allocation, task scheduling, and resource migration phases underpinned by applying the C-PSO algorithm with an adaptive mutation. It builds the system as an energy-efficient green cloud system through optimal allocation. It avoids the dominant solution among other workflow tasks that ensure the optimal global solution rather than local convergence solutions. Moreover, the scheduling of the workflow tasks was performed dynamically based on the scheduling plan, workflow model, and the optimal global solution. Also, the HH-ECO migrated the overloaded or underloaded resources during the execution of workflow applications, facilitating an improved QoS and an energy-efficient cloud environment.

The key achievements of the proposed methodology were:

- The analysis of energy models is performed in the cloud using a practical server with various load categories.
- HH-ECO approach provides an energy-efficient virtualised environment with high reliability and efficiency in resource allocation.
- HH-ECO approach allocates heterogeneous VMs based on the host CPU and memory capability and assists in successfully satisfying the user demands by minimising the cost of the schedulers.
- The C-PSO planned the scheduling procedure of the workflow application to utilise the advantages of a green cloud environment; the mutation process improved the potential profit without violating SLAs.
- The chaotic iterations in PSO avoid earlier standstill of particles before reaching global optima to provide a near-global optimal solution to promise user demands and achieve optimal energy consumption.
- C-PSO with adaptive mutation characterises the resource utilisation of VMs and generates the globally best migration plan to optimise the Makespan and cost.

Real-time experiments demand significant computational resources and extensive hardware, which are impractical at the academic level. Hence, the proposed HH-ECO approach is implemented using the CloudSim simulator. The experimental results of the HH-ECO approach demonstrated an adequate performance, accomplishing an
optimal Makespan of 36.27% and significant energy conservation of 43.85% when compared with existing CC approaches such as HHWS and DDVM. Based on different server behaviour, the results are obtained for the proposed HH-ECO approach.
CHAPTER 8 FUTURE WORK

In future work, we are proposing the following improvement.

VM migration among various data centres needs to be focused on along with the network latency. Additionally, instead of considering CPU and memory as the source of performance and energy consumption alone, other resources such as cooling systems and network racks such as routers and switches can be considered. Multiple tests can be done to check the mentioned resources regarding energy consumption, performance and QoS. Suitable algorithms will be developed to assist in improving performance and energy consumption. Incorporating advanced machine learning for dynamically optimising resource allocation by training these models to recognise patterns in scientific workflows, thus improving the energy efficiency and resource utilisation approaches.

As the final experiment was based on simulations, our intended future research will consider implementing the new algorithm, which will include CPU, memory, cooling system and network resources on real servers. Artificial intelligence (AI) and machine learning techniques will be programmed and used to check the behaviours of server components and cloud consumers. The AI algorithm will provide a comprehensive filtering mechanism that will check all the resources at different intervals. It will provide different filter sizes with capabilities to evaluate performance and power consumption, ensuring minimum energy consumption, minimum carbon emission, optimum performance and maximum quality of service to data centres and cloud users. By relying on our existing framework, AI will be used to put forward an AI control engine that will monitor and perform an action on the SLA violations, performance degradation, energy consumption and QoS.
LIST OF REFERENCES


[85] Shirvani, M. H. “A hybrid meta-heuristic algorithm for scientific workflow scheduling in heterogeneous distributed computing systems”, Engineering Applications of Artificial Intelligence, 90, 103501, 2020


