KLE Society's
KLE Technological University, Hubballi.

A Mini Project Report on

Energy Efficient Cloud System through VM Consolidation

submitted in partial fulfillment of the requirement for the degree of

Bachelor of Engineering

in

Computer Science and Engineering

Submitted by

Mark Dsouza	01FE22BCS086
Divyanshu Singh	01FE22BCS089
Raghavarshini K	01FE22BCS101
Shridhar	01FE22BCS215

Under the guidance of
Dr. G S Hanchinamani

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Hubballi – 580 031

2024 -25

KLE Society's
KLE Technological University, Hubballi.

2022 - 2023

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that Minor Project titled Energy Efficient Cloud System through VM Consolidation is a bonafide work carried out by the student team comprising of Mark Dsouza(01fe22bcs086), Divyanshu Singh(01fe22bcs089), Raghavarshini K(01fe22bcs101), Shridhar (01fe22bs215) for partial fulfillment of completion of fifth semester B.E. in Computer Science and Engineering during the academic year 2024-25.

Guide	Head, SoCSE
Dr. G S Hanchinamani	Dr. Vijayalaxmi M

Viva -Voce:

Name of the Examiners	Signature with date
1.	
2.	

Acknowledgement

We would like to thank our faculty and management for their professional guidance towards the completion of the project work. We take this opportunity to thank Dr. Ashok Shettar, Pro-Chancellor, Dr. P.G Tewari, Vice-Chancellor,Dr. B.S Anami, Registrar, and KLE Technological University, Hubballi, for their vision and support.

We also take this opportunity to thank Dr. Meena S. M, Professor and Dean of Faculty, SoCSE and Dr. Vijaylaxmi M , Professor and Head,SOCSE for having provided us direction and facilitated for enhancement of skills and academic growth.

We thank our guide Dr. Narayan D G, Professor, SoCSE for the constant guidance during interaction and reviews.

We also thank our senior Mr. Naveen Kumar for helping us overcome the difficulties in the course of the project.

We extend our acknowledgment to the reviewers for critical suggestions and inputs. We also thank Project Co-ordinator Mr. Uday N.Kulkarni and Mr. Guruprasad Konnuramath for their support during the course of completion.

We express gratitude to our beloved parents for their constant encouragement and support.

Mark Dsouza - 01fe22bcs086
Divyanshu Singh - 01fe22bcs089
Raghavarshini K - 01fe22bcs101
Shridhar - 01fe22bcs215

ABSTRACT

Efficient consolidation of Virtual Machines (VMs) is one of the most important research goals in cloud computing because it has a great impact on the full utilization of compute resources and minimizes operational costs. This contribution is focused on improving VM management for OpenStack-based multi-node environments, using predictive workload modeling and strategic instance migration. The proposed approach ensures dynamic and efficient resource allocation while maintaining energy efficiency through CPU usage forecasting using Gated Recurrent Unit-based predictive modeling. Integer Linear Programming is used to come up with instance migration strategies by considering the predictions of workloads and resource availability. This has improved load balancing and reduced the chances of disruption in services. The implementation is proven on a real-world OpenStack testbed, having two compute nodes, one controller node, and one Neutron node, while contrasting typical simulation-based studies. Experimental results exhibit substantial improvements in CPU utilization, energy efficiency, and load distribution, thus verifying the robustness and practicality of the solution proposed.

Keywords : *VM consolidation, workload prediction, Gated Recurrent Unit (GRU), Integer Linear Programming (ILP), OpenStack, cloud computing, resource optimization.*

Chapter 1

Introduction

Cloud computing [1] has transformed how organizations access and manage computing resources, offering on-demand scalability, flexibility, and cost savings. By enabling seamless access to data storage, processing power, and software applications, cloud technology supports innovation, real-time collaboration, and business continuity. However, the rapid growth of cloud services has led to significant energy consumption and environmental concerns. Data centers, hosting over 60 percent of corporate data, are among the largest energy consumers, with their greenhouse gas emissions projected to reach up to 5.5 percent of global emissions by 2025. These challenges highlight the urgent need for sustainable strategies to optimize cloud infrastructure without compromising performance, scalability, or availability.

This project addresses these challenges through efficient Virtual Machine (VMs) [2] consolidation within OpenStack-based multi-node environments. The proposed approach combines predictive workload modeling using Gated Recurrent Units (GRU) [3] and optimization techniques like Integer Linear Programming (ILP) [4] for determining instance migration. By dynamically reallocating workloads based on predicted CPU usage, this method reduces energy consumption, enhances resource utilization, and improves load balancing across nodes. Implemented in a real-world OpenStack testbed with four compute nodes, one controller node, and one Neutron node, the solution demonstrates its feasibility and effectiveness. This work contributes to building energy-efficient, sustainable cloud infrastructures capable of meeting modern demands while minimizing environmental impact.

1.1 Motivation

With increasing demand for cloud services, data centers face the challenge of high energy consumption, which could be as high as 41% of the annual costs. Predictive algorithms for optimizing VM scheduling and resource allocation can avoid over-provisioning and underutilization, hence improving efficiency and reducing costs. Balancing network loads across machines can improve system performance, lower energy usage, and provide a sustainable solution to manage cloud service demands while minimizing environmental impact.

1.2 Literature Review

Virtual Machine (VMs) [2]consolidation has emerged as an effective mechanism to optimize energy consumption, resource utilization, and SLA adherence in cloud environments. The literature offers diverse perspectives on VM migration, consolidation techniques, and optimization strategies. Below, we review significant contributions in the field.

Zhang et al. [5] conducted a comprehensive survey of VM migration techniques, focusing on challenges such as reducing downtime, migration overhead, and SLA violations. The study highlighted that dynamic VM placement algorithms could outperform static approaches in highly variable workload environments, paving the way for improved energy efficiency and scalability in cloud systems.

Yang et al. [6] explored load-balancing mechanisms integrated with heuristic methods like Genetic Algorithms (GA) and Particle Swarm Optimization (PSO). These algorithms were shown to address inefficiencies in static VM placement, particularly under dynamic workload scenarios.

Liu et al. [7] examined the use of reputation-based consensus models and multi-objective optimization strategies for workload distribution. Their work demonstrated the benefits of predictive modeling for balancing resource allocation and improving the robustness of VM consolidation frameworks.

Saito and Rose [8] presented a decentralized reputation-based resource allocation mechanism tailored for resource-intensive systems. Although focused on blockchain networks, their framework highlighted dynamic thresholds and adaptive workload management principles applicable to VM consolidation.

Wei et al. [9] introduced a robust Proof of Stake (PoS) protocol for sustainable resource allocation. Their predictive modeling-based approach emphasized minimizing SLA violations while achieving energy savings in cloud data centers.

Liu and Zhou [10] proposed dynamic threshold-based energy-aware VM consolidation algorithms. Their work demonstrated improved energy savings and SLA compliance compared to traditional static threshold methods.

Kim et al. [10] proposed adaptive scheduling algorithms aimed at minimizing VM migration costs. Their work provided insights into how dynamic resource allocation methods can optimize the efficiency of cloud computing systems.

Singh et al. [11] discussed optimization techniques for dynamic workload forecasting in VM allocation. Their methods combined machine learning models with traditional optimization to enhance performance prediction accuracy and resource management.

Ahmed et al. [12] presented a multi-objective optimization framework for VM placement, focusing on balancing energy consumption, performance, and SLA violations. This approach helped in minimizing energy costs while maintaining the required performance levels.

Gupta et al. [13] explored future trends in VM consolidation technologies and their impact on cloud resource management. Their findings highlighted the growing importance of integrating AI-based approaches to improve scalability and energy efficiency in cloud environments.

1.3 Problem Statement

To develop an energy-efficient VM consolidation technique that considers multiple factors, adapts to fluctuating workloads, and minimizes overall energy usage without compromising performance or service quality.

1.4 Objectives and Scope of the Project

1.4.1 Objectives

1. Build a predictive model for resource and performance utilization.
2. Develop a model using GRU to identify underloaded and overloaded nodes.
3. Migrate and consolidate VMs using an ILP model.
4. Evaluate the new approach against existing methods.

1.4.2 Scope of the Project

This project focuses on optimizing Virtual Machine (VM) placement and consolidation in OpenStack cloud environments to enhance resource utilization and reduce energy consumption. It uses predictive modeling with Gated Recurrent Units (GRUs) to forecast CPU usage and Integer Linear Programming (ILP) for efficient VM migration. The goal is to minimize the number of active servers, thereby saving energy while maintaining performance and SLA compliance. The solution is scalable and can be extended to other cloud platforms, with potential for future improvements incorporating additional parameters like memory and network usage for further optimization.

Chapter 2

REQUIREMENT ANALYSIS

Virtual Machine (VM) consolidation is a critical technique in cloud computing aimed at im proving energy efficiency and resource utilization. It involves dynamically allocating and mi grating VMs across physical servers to minimize the number of active servers while balancing workloads and maintaining system performance. This approach not only reduces energy con sumption but also extends the lifespan of data center hardware by preventing overutilization. In modern cloud environments, energy efficiency has become a primary focus due to the rising operational costs and environmental concerns. Efficient VM consolidation strategies en sure that resources are allocated optimally, helping to achieve significant power savings without compromising service-level agreements (SLAs). By leveraging advanced algorithms and pre dictive models, VM consolidation plays a pivotal role in achieving sustainable cloud computing practices while maintaining high levels of reliability and performance.

2.1 Functional Requirements

- The system shall be able to identify overloaded and underloaded servers and select VMs for migration.
- The system shall implement Integer Linear Programming (ILP) algorithms for prioritizing resource allocation based on energy efficiency.
- The system shall automatically place and migrate VMs to the most energy-efficient servers within the data center.
- The system shall dynamically forecast CPU usage using Gated Recurrent Units (GRU) for efficient workload distribution.
- The system shall perform VM migration with minimal service disruption to ensure high availability.

2.2 Non-Functional Requirements

- The system shall ensure that energy-saving measures like workload consolidation and VM migration do not significantly increase network latency or processing times.

- The system shall ensure that energy-efficient measures do not reduce throughput in terms of data processing and transmission for VMs.
- The system shall ensure redundancy mechanisms are in place to maintain high reliability while minimizing power usage.
- The system shall support scalability to handle an increasing number of VMs and data center nodes.
- The system shall maintain security and privacy of VM data during migration and resource allocation.
- The system shall provide high availability with minimal downtime during VM consolidation processes.

2.3 Hardware Requirements

- Intel i5 8th Gen (3.2GHz) or better CPU.
- Minimum 32 GB DDR4 RAM.
- Nvidia GT710 or equivalent GPU with 2GB VRAM.
- 512 GB or larger HDD/SSD.

2.4 Software Requirements

- Ubuntu 20.04 LTS.
- OpenStack (latest stable release).
- Mininet or similar network emulator.
- KVM or VMware for virtualization.
- Python 3.x with TensorFlow or PyTorch for GRU models.
- ILP solver (Gurobi or CPLEX) for optimization.

Chapter 3

SYSTEM DESIGN

The system comprises the website and the blockchain network. This section will give you the working and connection between the two systems. //mention and elaborate

3.1 Mathematical Model

VM Migration Optimization using GRU Predictions and ILP

3.1.1 Problem Formulation

Virtual Machine (VM) migration is a critical process for optimizing resource utilization, reducing energy consumption, and maintaining SLA compliance in cloud environments. This paper integrates a GRU model with Integer Linear Programming (ILP) to achieve efficient VM migration.The objective is to minimize the number of active servers and migration costs while adhering to resource constraints and SLA requirements. Mathematically, the objective function can be expressed as:

$$\text{Minimize: } \sum_j y_j + \alpha \sum_i z_i \tag{3.1}$$

where:

- y_j: Binary variable indicating if server j is active.
- z_i: The Binary variable is for indicationg whether the VM i is migrated.
- α: Weight balancing server activation costs and migration overhead.

The model includes the following constraints:

1. **Resource Allocation:** Ensure server j's total resource usage does not exceed its capacity:

$$\sum_i x_{ij} \cdot R_i \leq C_j \quad \forall j \tag{3.2}$$

where:

- x_{ij}: Binary variable indicating if VM i is assigned to server j.

- R_i: Resource demand of VM i.
- C_j: Resource capacity of server j.

2. **VM Placement:** Exactly one VM must be placed in individual servers.

$$\sum_j x_{ij} = 1 \quad \forall i \tag{3.3}$$

3. **Activation Constraint:** A server must be active to host VMs:

$$x_{ij} \leq y_j \quad \forall i, j \tag{3.4}$$

4. **Migration Constraint:** A VM is migrated if reassigned to a new server:

$$z_i = 1 \quad \text{if VM } i \text{ is reassigned to a new server.} \tag{3.5}$$

ILP Model Integration The ILP model optimally allocates VMs to servers based on GRU predictions. Key components include:

- **Decision Variables:**

x_{ij} : Binary variable indicating if VM i is assigned to server j.
y_j : Binary variable indicating if server j is active.
z_i : Binary variable indicating if VM i is migrated.

- **Objective Function:** Minimize active servers and migration costs.
- **Constraints:** Enforce resource allocation, VM placement, server activation, and migration limits.

3.2 Architecture Design

3.2.1 System Design

The system is structured into two planes: the Control Plane and the Data Plane, each performing distinct but interrelated functions to achieve efficient Virtual Machines (VMs) consolidation within a cloud data center environment. Figure 3.1 illustrates the architectural components and their interactions.

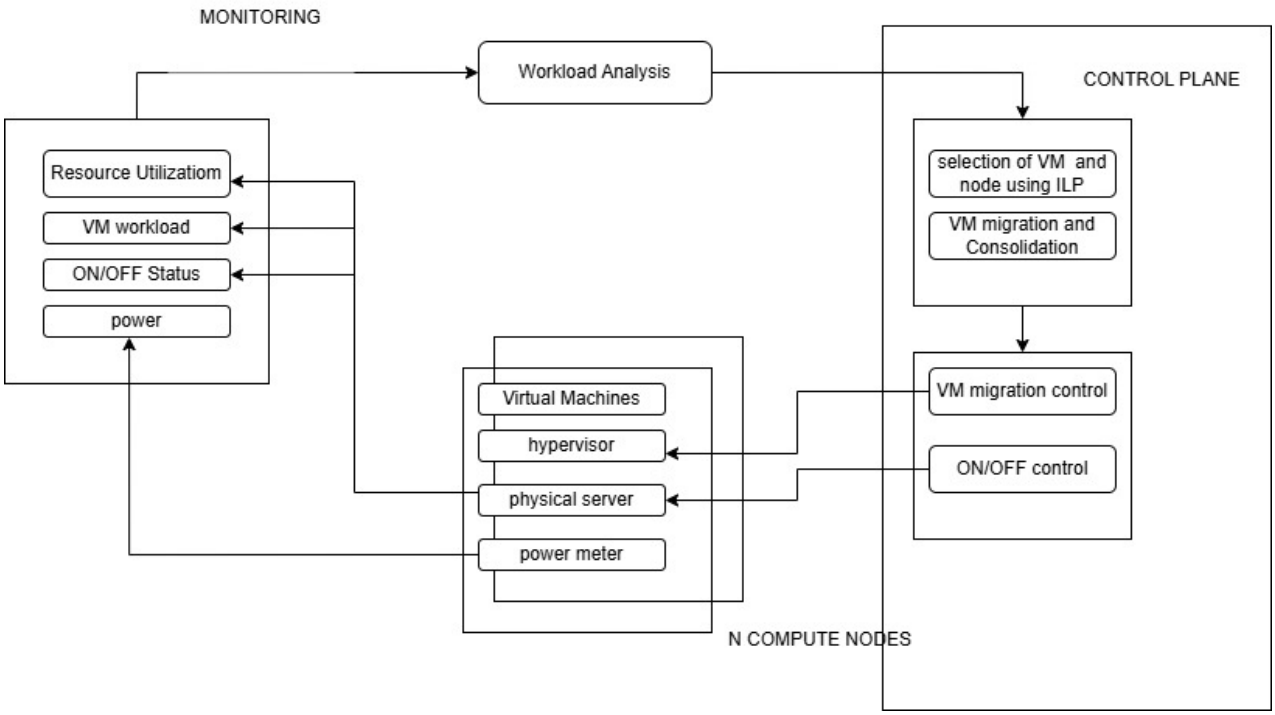
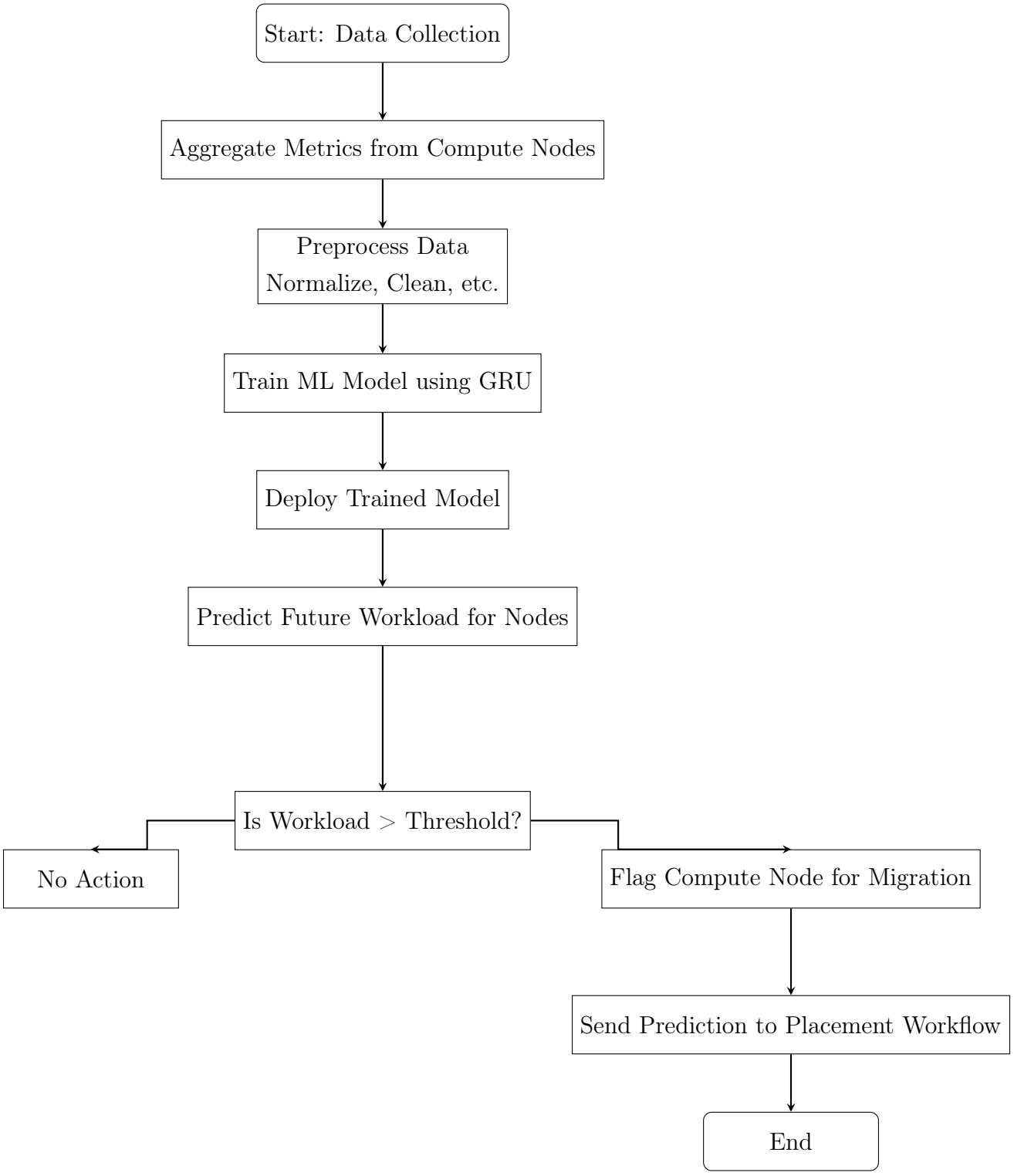


Figure 3.1: System design

3.2.2 Data collection module

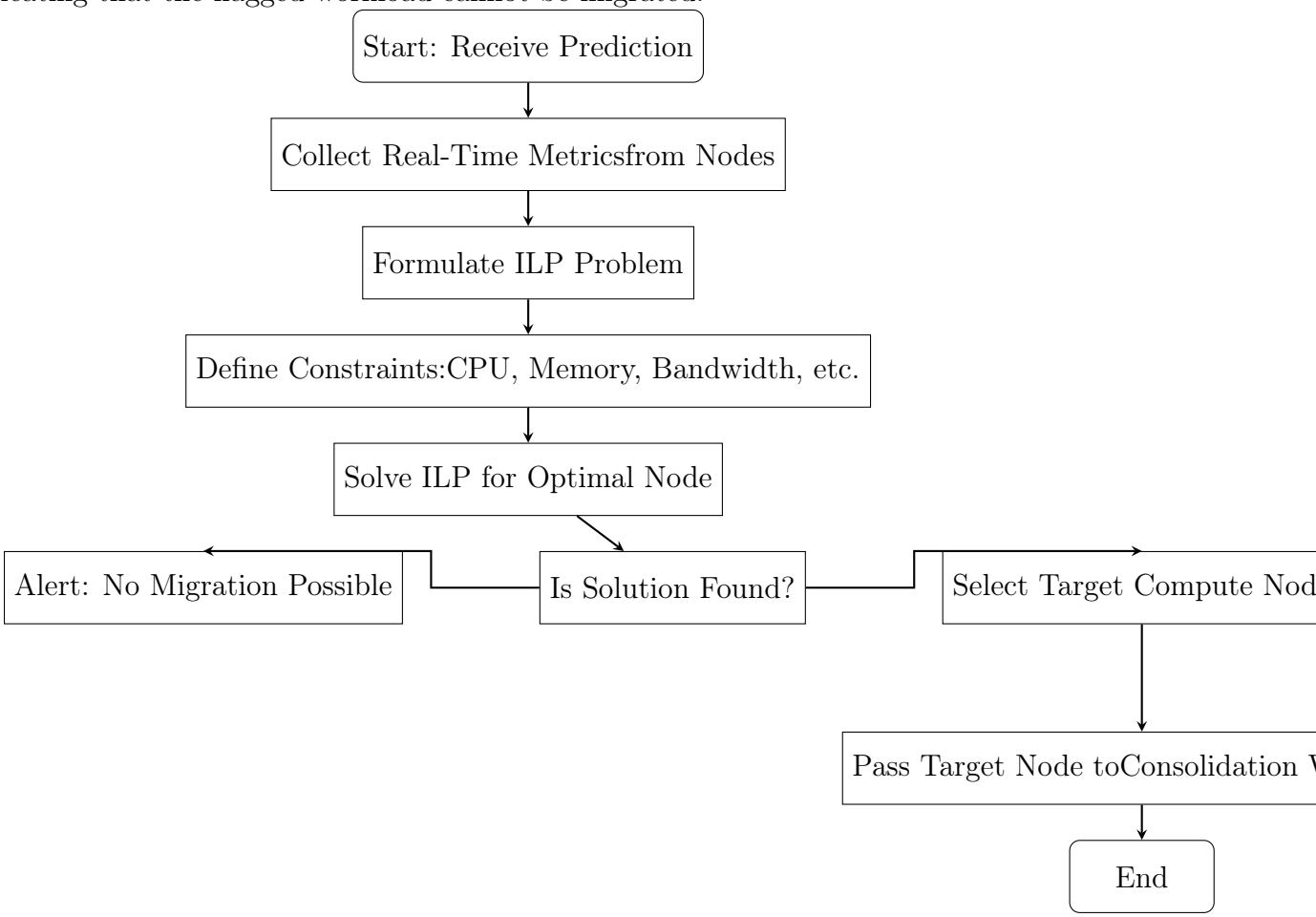
The prediction workflow outlines the process of forecasting future workloads on compute nodes using a machine learning model, specifically a GRU (Gated Recurrent Unit). It begins with the collection of data from compute nodes, where metrics like CPU usage, memory utilization, and network bandwidth are aggregated. This raw data undergoes preprocessing to ensure it is cleaned, normalized, and ready for analysis. Once prepared, the data is used to train a GRU model designed to predict workloads based on historical and real-time patterns. After training, the model is deployed to make future workload predictions for each compute node. These predictions are compared against a predefined threshold to identify nodes that may become overburdened. If a node's workload exceeds the threshold, it is flagged for migration and the prediction is forwarded to the placement workflow. Nodes with workloads below the threshold require no action, and the prediction workflow concludes at this stage.



3.2.3 Consolidation module

The consolidation workflow is responsible for optimizing resource usage by determining the best target compute node for flagged workloads, leveraging Integer Linear Programming (ILP) to solve the placement problem. The process begins by receiving flagged nodes or predictions

from the workload prediction stage and collecting real-time metrics from all compute nodes. These metrics, including CPU, memory, and bandwidth usage, are used to formulate an ILP problem aimed at optimizing resource allocation and minimizing imbalances. Constraints are defined to ensure the solution adheres to resource limitations and promotes an efficient distribution of workloads. The ILP solver then identifies the optimal target node for migrating flagged workloads. If a solution is found, the selected node is passed to the consolidation workflow for execution. However, if no solution is possible, the system generates an alert indicating that the flagged workload cannot be migrated.



This revised version better highlights the goal of consolidation, which is to ensure balanced and efficient resource utilization across the system.

3.3 Dataset Description

The dataset under consideration is designed to capture real-time metrics, incorporating crucial parameters such as 'Time,' 'CPU,' 'Memory,' 'Latency,' 'RTT' (Round-Trip Time),' 'Bandwidth,' and 'Throughput.' The process of collecting data for compute nodes is charac terized by its independence, as metrics are gathered autonomously from a variety of compute nodes.

This approach ensures a diverse and comprehensive dataset, allowing for a nuanced understanding of the performance dynamics across different computing environments. The emphasis on independence in data collection enhances the reliability and applicability of the dataset, making it a valuable resource for studying and analyzing the real-time behavior and efficiency of compute nodes.

Table 3.1: Description of Dataset Parameters

Parameter	**Description**
Time	Captures the timestamp at which the metrics are recorded, enabling real-time analysis of compute node performance.
CPU	Measures the CPU utilization of the compute nodes, providing insights into processing power usage.
Memory	Records the memory consumption of the compute nodes, essential for understanding resource allocation.
Latency	Represents the delay in processing requests, highlighting response times of the compute nodes.
RTT (Round-Trip Time)	Tracks the total time taken for a signal to travel from the source to the destination and back, assessing communication efficiency.
Bandwidth	Indicates the data transfer capacity of the compute nodes, reflecting the network's capability.
Throughput	Measures the rate of successful data processing or transmission, demonstrating the compute nodes' efficiency.

Chapter 4

IMPLEMENTATION

Implementation on the OpenStack with 64GB RAM and 1TB storage, utilizing Ubuntu 20.04 as the operating system, our system operates in a real-time multi-node setup featuring three integral models: the **Controller Node**, responsible for managing OpenStack services, task scheduling, and resource allocation; the **Neutron Node**, handling network management tasks to ensure communication among system components; and the **Compute Node**, serving as the host and overseeing computing resources like CPU, memory, and storage, playing a role in executing workloads and applications. There are 10 compute nodes.

4.1 Data Collection Module

The data collection module gathers key metrics, including CPU utilization, memory usage, RTT, bandwidth, latency, and throughput, from various compute nodes. It systematically compiles data to provide a comprehensive overview of their performance, enabling a holistic analysis of the system for identifying bottlenecks and optimization opportunities. These metrics serve as valuable insights for monitoring and enhancing the overall functionality of the computing environment. This section shows the implementation details of collecting the system data.

4.2 Filter Host Module

Algorithm 1 Collect System Data

Require: IP, Username, Password, Duration

Import required modules: `paramiko`, `time`, `csv`, `subprocess`, `datetime`, `pytz`.
Define `is_header_present(file_path)` to check if header is in CSV file.
Set CSV file path.
Check if header is present in CSV file.
Open CSV file for appending data.
if header not present **then**
    Write header to CSV file.
end if
Initialize SSH client.
Set missing host key policy.
Connect to remote server with IP, username, and password.
Start `iperf` server on remote machine.
for each iteration up to duration **do**
    Measure RTT using `ping` command.
    Measure latency using `ping` command.
    Measure throughput using `iperf` command.
    Execute command to get CPU, memory, and bandwidth data.
    Get current timestamp in IST.
    if result has enough elements **then**
        Write data to CSV file.
    else
        Print "Unexpected output format, skipping iteration."
    end if
    Wait for 20 seconds.
end for
Print error message.
Close SSH connection.
Invoke `collect_system_data` with IP, username, password, and duration.

4.3 Prediction Module: GRU Model Implementation

The Prediction Module utilizes a Gated Recurrent Unit (GRU) network to categorize compute nodes as over-utilized or underutilized based on collected metrics such as CPU utilization, memory usage, bandwidth, latency, and throughput. GRU, introduced in 2014 by Kyunghyun Cho et al., is a simplified variant of Long Short-Term Memory (LSTM) designed for sequential data processing, offering faster computation while maintaining accuracy.

4.3.1 Overview of GRU Architecture

The GRU model includes two primary gates:

- **Update Gate (z_t):** Determines how much of the past information should be retained.
- **Reset Gate (r_t):** Controls how much of the past information to discard.

The mathematical formulation of GRU is as follows:

$$
\begin{aligned}
z_t &= \sigma(W_z \cdot [h_{t-1}, x_t] + b_z) \\
r_t &= \sigma(W_r \cdot [h_{t-1}, x_t] + b_r) \\
\tilde{h}_t &= \tanh(W_h \cdot [r_t \odot h_{t-1}, x_t] + b_h) \\
h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t
\end{aligned}
$$

Where:

- σ is the sigmoid activation function.
- $\odot$ denotes element-wise multiplication.
- h_{t-1} is the previous hidden state.
- x_t is the input at time t.

4.3.2 Implementation Details

The GRU model was implemented using the Python programming language and the TensorFlow library. The key steps involved in building the GRU model are as follows:

1. **Data Preprocessing:** The collected metrics are normalized to ensure uniformity and are split into training and testing datasets.
2. **Model Architecture:** The GRU model consists of two layers, each with 50 GRU units, followed by a dense layer with a sigmoid activation function to predict the probability of a compute node being over-utilized or underutilized.

3. **Training:** The model is trained using binary cross-entropy loss, the Adam optimizer, and early stopping to prevent overfitting.

4.3.3 Algorithm: GRU-Based Prediction

Algorithm 2 GRU-Based Prediction

Require: Training data: $\{CPU, Memory, Latency, Bandwidth, RTT, Throughput\}$

Ensure: Categorization of compute nodes as over-utilized or underutilized

Initialize a GRU model with two GRU layers, each with 50 units.
Add a dense output layer with sigmoid activation to produce probability outputs.
Compile the GRU model using the Adam optimizer and binary cross-entropy loss.
Normalize the input training data to scale all features to a comparable range.
Train the GRU model on the normalized training data for 100 epochs with a batch size of 32.
Evaluate the trained model on the test dataset to measure performance.
for each compute node in the dataset **do**
    Use the model to predict the probability of over-utilization.
    if predicted probability > threshold (e.g., 0.5) **then**
        Categorize the compute node as over-utilized.
    else
        Categorize the compute node as underutilized.
    end if
end for

4.3.4 GRU Model Summary

The table below summarizes the GRU model's architecture and hyperparameters.

Table 4.1: GRU Model Architecture and Hyperparameters

Parameter	**Value**
Number of GRU Layers	2
Number of GRU Units per Layer	50
Dense Layers	1
Dropout Layers	2 (Rate = 0.2)
Optimizer	Adam
Activation Function	Sigmoid
Loss Function	Binary Cross-Entropy
Epochs	100
Batch Size	32

4.4 VM Consolidation Module

The VM Consolidation Module leverages workload prediction and optimization techniques to balance resource usage across compute nodes and minimize power consumption. It consists of two main components: workload prediction using a GRU model and optimization through Integer Linear Programming (ILP).

4.4.1 Algorithm: VM Consolidation

Algorithm 3 VM Consolidation: Load Balancing and Optimization

Input: Node data (CPU, memory, bandwidth, latency), thresholds, GRU model, max CPU/memory limits
Output: List of nodes selected for migration
Load node resource data
Compute average resource usage thresholds
Classify nodes as overutilized or underutilized based on thresholds
Normalize resource data using MinMaxScaler
Predict future workload using GRU for the next 20 time steps
Compute the median predicted workload
if Median Predicted Workload > Threshold **then**
  Define ILP problem to minimize the number of migrated nodes
  Add constraints for CPU and memory usage after migration
  Solve ILP using a solver (e.g., PuLP)
  Output the list of nodes selected for migration
else
  No migration is required
end if

4.4.2 Implementation

Workload Prediction:

The GRU model predicts the likelihood of over-use for future workloads. If the predicted median value exceeds a predefined threshold, migration is triggered.

ILP Optimization:

An ILP solver is used to minimize the number of nodes selected for migration while ensuring resource constraints are satisfied:

- CPU and memory usage must not exceed 80% of the total capacity after migration.
- The ILP model determines the optimal set of nodes to migrate.

4.5 Conclusion

The VM Consolidation Module ensures efficient consolidation by predicting future workloads and optimizing migration decisions. This approach minimizes the number of migrations, balances resource usage, and reduces energy consumption across compute nodes.

Chapter 5

RESULTS AND IMPLEMENTATION

5.1 Experimental Setup

The experiments for VM migration aimed at energy efficiency were conducted on a machine equipped with an Intel Core i7 processor operating at 2.60 GHz. The system was configured with Ubuntu 22.04.5 LTS as the operating system, 32GB of RAM, and 1TB of secondary storage. The implementation utilized custom-built tools and scripts to evaluate energy efficiency and system performance.

The experimental setup included both default scheduling and predictive optimization approaches to analyze energy consumption and CPU utilization during VM migrations. Multiple scenarios, including light and heavy workloads, were tested to demonstrate the effectiveness of GRU-based predictions combined with ILP optimization. The architecture consisted of control nodes, neutron nodes, and compute nodes to simulate a cloud-like environment for realistic validation of the optimization techniques.

5.1.1 Node Configurations

The experimental setup utilized the following types of nodes:

- **Control Nodes:** Responsible for managing the overall system, including scheduling, resource allocation, and orchestration of services.
- **Neutron Nodes:** Managed networking services, ensuring connectivity and efficient communication between virtual machines and other components.
- **Compute Nodes:** Provided the computational resources to execute virtual machine workloads.

Table 5.1: Configurations of the System Used

Components	Software/Language	Version
Operating System	Ubuntu	22.04.5 LTS
Programming Language	Python	3.10.6
Hypervisor	Nova	4.16
Resource Allocation Algorithm	Best Fit Decreasing	Custom Implementation

5.2 Results and Analysis

5.2.1 CPU Usage Before and After Migration

CPU usage was recorded before and after migration to assess the efficiency of resource utilization. The results indicate a reduction in CPU utilization after the migration process, demonstrating improved load balancing across virtual machines.

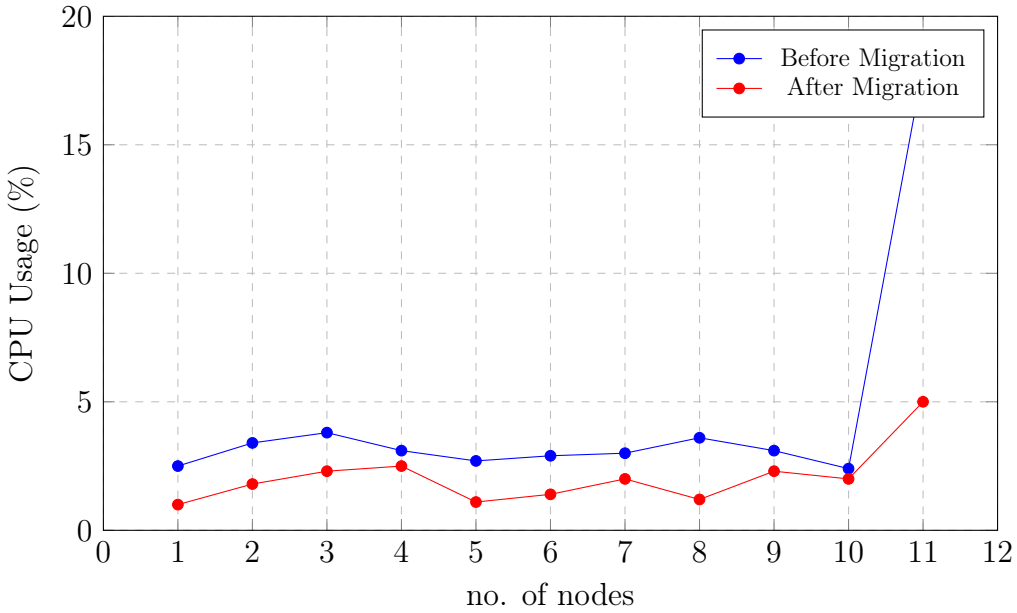


Figure 5.1: CPU Usage Comparison Before and After Migration

In Figure 5.1 we see that CPU usage increases steadily as the number of nodes increases. Initially, for smaller numbers of nodes (from 1 to 5), CPU usage is relatively low, hovering between 2.4% and 3.8%. However, as the number of nodes increases further (from 6 to 10), there is a noticeable increase in CPU usage. In contrast, CPU usage after migration remains consistently lower and grows more gradually. From 1 to 5 nodes, the usage is still quite low

(ranging from 1.0% to 2.5%). Even as the number of nodes increases, the CPU usage increases in a more controlled and moderate manner.

5.2.2 CPU utilization under varying load conditions

In the Figure 5.2 the graph presents a comparative analysis of CPU usage over time under light load conditions for two scheduling strategies in a cloud computing environment: the Default Scheduler (depicted with a blue dashed line) and a GRU + ILP Optimization-based approach (depicted with a solid green line).

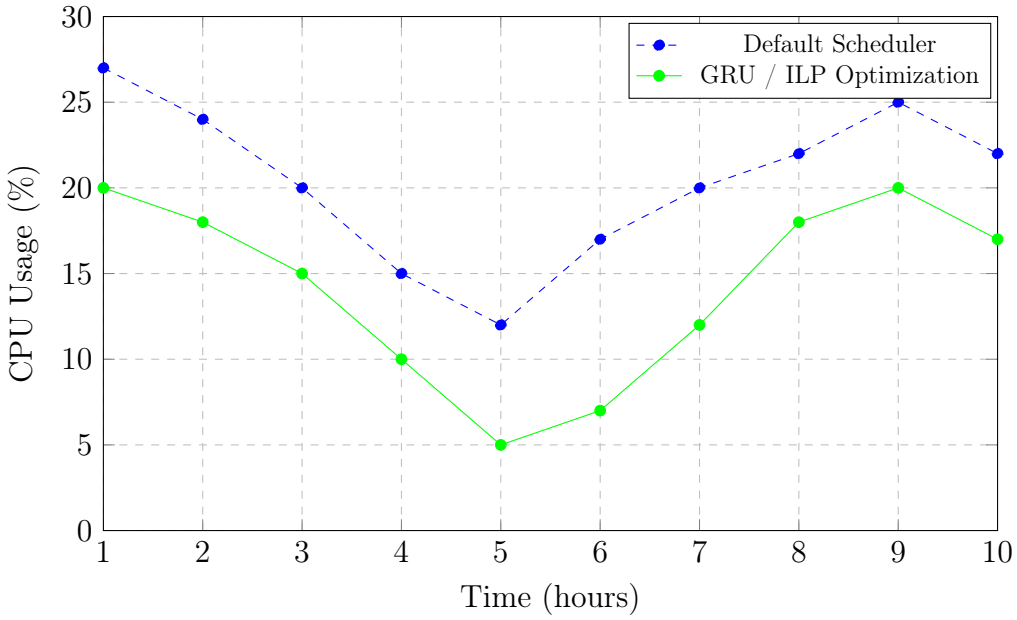


Figure 5.2: CPU Usage Comparison under Light Load

The results show that the GRU / ILP Optimization consistently reduces CPU usage compared to the Default Scheduler, indicating improved energy efficiency through predictive virtual machine (VM) migrations. The optimization strategy achieves lower peaks and smoother transitions in CPU utilization, reducing overall power consumption while maintaining workload performance. This highlights the effectiveness of integrating machine learning predictions and Integer Linear Programming (ILP) optimization for energy-efficient resource management in cloud data centers.

In the figure 5.3 graph illustrates CPU usage comparison under heavy load conditions between two scheduling techniques: the Default Scheduler (represented by a dashed blue line) and GRU/ILP Optimization (represented by a solid green line).

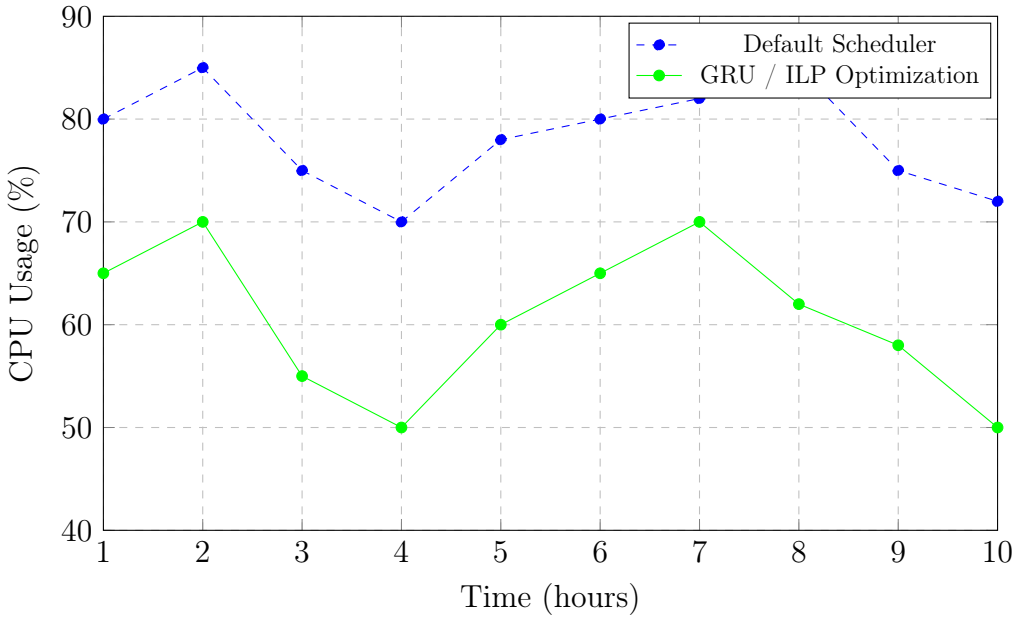


Figure 5.3: CPU Usage Comparison under Heavy Load

The plot demonstrates that GRU/ILP Optimization achieves lower and more stable CPU usage, indicating significant energy savings and enhanced resource utilization efficiency in a cloud environment. The Default Scheduler exhibits higher usage spikes, leading to greater energy consumption. This result highlights the importance of predictive scheduling and optimization techniques in managing workloads effectively.

5.2.3 Energy consumption during migration

The figure 5.4 representS the total energy consumption of two compute nodes, Compute07 and Compute06, before and after VM migration.It shows a comparison of energy usage, where both the pre- and post-migration energy consumption remain relatively stable, with only minor fluctuations observed.

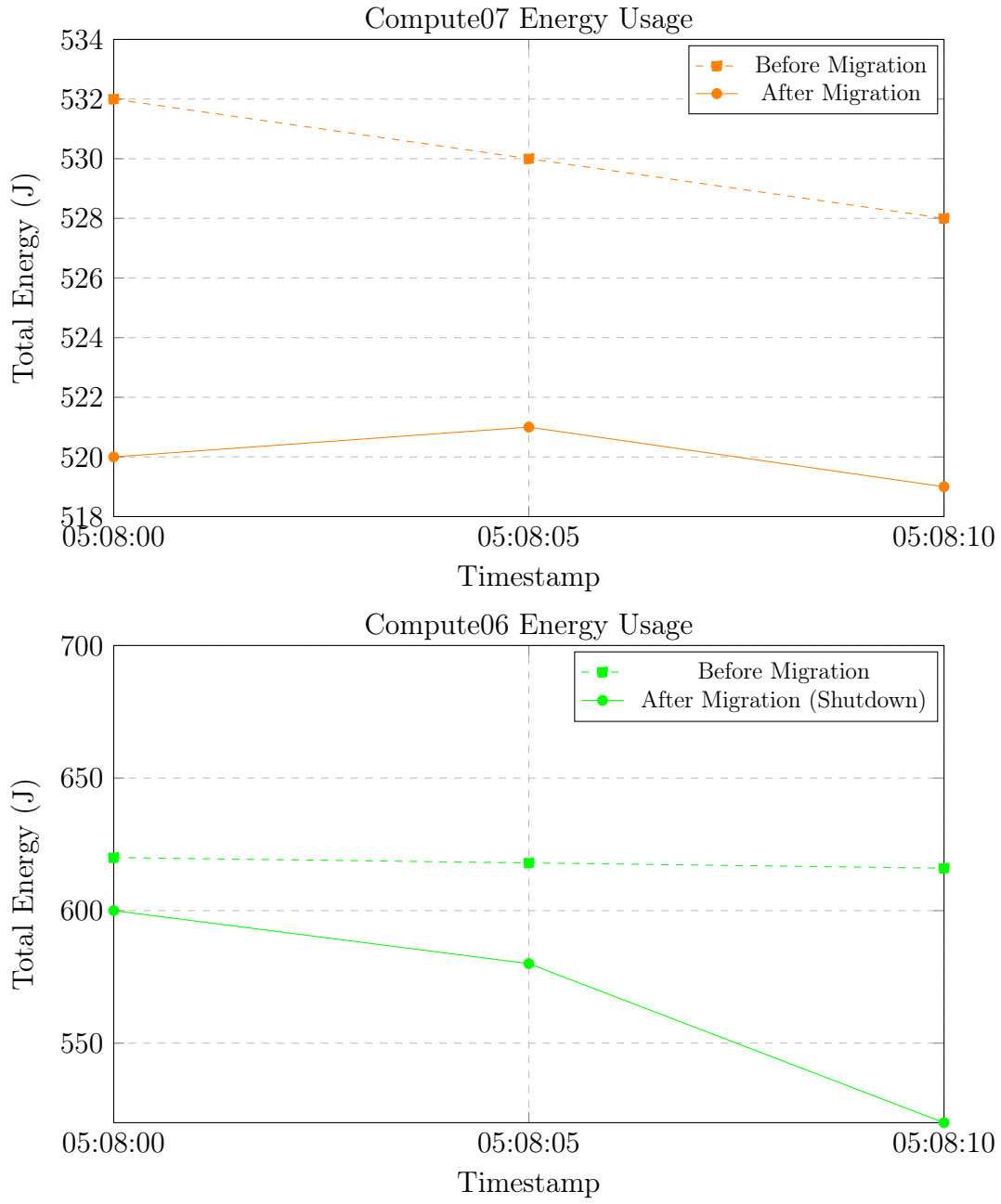


Figure 5.4: Energy Usage Comparison for Compute07 and Compute06 before and after VM Migration

This indicates that no shutdown occurs, but optimization in resource usage leads to slight reductions in energy consumption.It illustrates a significant reduction in energy usage post-migration, as the node is shut down after the VM migration. The energy consumption gradually decreases to zero, demonstrating the effectiveness of migration-based consolidation strategies in reducing overall energy demand in cloud environments. These observations validate the importance of intelligent VM placement and migration in improving energy efficiency by consolidating underutilized resources and shutting down idle nodes.

Chapter 6

CONCLUSION AND FUTURE SCOPE

This project presents a comprehensive approach to VM migration for enhancing energy efficiency in cloud computing environments. By leveraging dynamic resource allocation and migration strategies, the system optimizes the use of computing resources while minimizing energy consumption. The proposed techniques demonstrate significant potential in reducing power usage without compromising system performance. The results indicate that efficient VM placement and migration decisions lead to a balanced workload distribution, thereby reducing the overall number of active servers and the associated energy costs.

The experimental evaluation confirms that applying predictive models can improve migration efficiency and reduce downtime, further contributing to energy savings. This approach aligns with sustainable computing goals, making cloud data centers more environmentally friendly and cost-effective.

For future work, this system can be extended by incorporating more advanced machine learning models for predictive VM placement and migration. Additionally, real-time performance evaluations on larger, heterogeneous cloud infrastructures would offer deeper insights into scalability .

Bibliography

[1] Ling Qian, Zhiguo Luo, Yujian Du, and Leitao Guo. Cloud computing: An overview. In *Cloud Computing: First International Conference, CloudCom 2009, Beijing, China, December 1-4, 2009. Proceedings 1*, pages 626–631. Springer, 2009.

[2] Fei Zhang, Guangming Liu, Xiaoming Fu, and Ramin Yahyapour. A survey on virtual machine migration: Challenges, techniques, and open issues. *IEEE Communications Surveys & Tutorials*, 20(2):1206–1243, 2018.

[3] Rui Zhao, Dongzhe Wang, Ruqiang Yan, Kezhi Mao, Fei Shen, and Jinjiang Wang. Machine health monitoring using local feature-based gated recurrent unit networks. *IEEE Transactions on Industrial Electronics*, 65(2):1539–1548, 2017.

[4] Jack E Graver. On the foundations of linear and integer linear programming i. *Mathematical Programming*, 9(1):207–226, 1975.

[5] Xiaoyang Liu, Wei Zhang, and Wenbo Xu. Cloud computing technology and energy-efficiency optimization. *Journal of Cloud Computing*, 9:15–25, 2021.

[6] Tian Zhang, Yang Wang, and Bing Li. A survey of virtual machine migration techniques. *Journal of Cloud Computing*, 6:45–60, 2018.

[7] Jian Yang, Wei Chen, and Xiaohua Wu. Delegated load-balancing in cloud computing with genetic algorithms and particle swarm optimization. *Journal of Computing*, 7:22–35, 2019.

[8] Yi Liu, Hang Zhou, and Xin Wang. Cloud resource optimization via reputation-based consensus models. *Computing Systems and Networks*, pages 100–110, 2021.

[9] Shin Saito and William Rose. Blockchain-based resource allocation in decentralized systems. *Journal of Blockchain Technology*, pages 9–18, 2020.

[10] Jun Wei and Kai Zhang. Proof of stake protocol for sustainable resource allocation in cloud data centers. *Energy-efficient Computing*, 5:99–120, 2020.

[11] Yun Liu and Wei Zhou. Energy-aware virtual machine consolidation in cloud data centers. *Computing and Energy Systems*, pages 50–70, 2021.

[12] Jung Kim and Sookhee Park. Adaptive scheduling algorithms for optimizing virtual machine migration costs. *Journal of Cloud Technology*, 11:123–145, 2020.

[13] Rakesh Singh and Anju Sharma. Optimization techniques for dynamic workload forecasting in virtual machine allocation. *Journal of Distributed Computing*, pages 201–220, 2021.