Arab Academy for Science, Technology, and Maritime Transport

College of Computing and Information Technology

CS

B. Sc. Final Year Project

PROJECT II

Optimization of Cloud Resource Allocation

Submitted By:

Fathy Mohamed (CS) 20204068

Ahmed Adel (CS) 20204089

Youssef Tarek (CS) 20204050

Youssef Hassan (CS) 19108390

Moayad Mohamed (CS) 20204144

Supervised By:

Assoc Prof Fahima Maghraby

Dr Ehab Aboseif

October – 2024

Table of Contents

List of Figures

List of Tables

DECLARATION

I hereby certify that this material, which I now submit for assessment on the program of study leading to the award of Bachelor of Science in *Computer Science* is entirely my own work, that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Student Name: Fathy Mohamed

Registration No.: 20204068

Signed: _____

Date: 10/02/2025.

Student Name: Ahmed Adel

Registration No.: 20204089

Signed: _____

Date: 10/02/2025.

Student Name: Youssef Tarek

Registration No.: 20204050

Signed: _____

Date: 10/02/2025.

Student Name: Youssef Hassan

Registration No.: 19108390

Signed: _____

Date: 10/02/2025.

Student Name: Moayad Hamzeh

Registration No.: 20204144

Signed: _____

Date: 10/02/2025.

ACKNOWLEDGEMENT

I would like to express my deepest gratitude to Assoc. Prof. Fahima Maghraby and Dr. Ehab Aboseif for their invaluable guidance, unwavering support, and insightful feedback throughout this research. Their expertise, encouragement, and dedication have been instrumental in shaping the direction and quality of this work.

I am also grateful to my institution and faculty members for providing the necessary resources and a conducive learning environment. My heartfelt appreciation extends to my colleagues, friends, and family, whose support and motivation have been a constant source of strength during this journey.

This research would not have been possible without the collective efforts of all those who contributed, and I am truly thankful for their role in this endeavor.

ABSTRACT

The rapid worldwide adoption of cloud data centers (CDCs) has heightened the demand for hosting application services on the cloud, particularly in data-intensive industries. This surge has led to an increase in the number of cloud servers, resulting in higher energy consumption and sustainability concerns. Traditional resource management techniques, such as heuristics and reinforcement learning, struggle with scalability and adaptability, especially in dynamic environments with non-stationary resource demands. These methods often fail to capture the dependencies among the thermal characteristics of hosts, the resource consumption of tasks, and the corresponding scheduling decisions, leading to poor scalability and increased computational resource requirements. To address these limitations, we propose a holistic AI-based resource management approach for sustainable cloud computing. This approach formulates the optimization of energy efficiency in data centers as a multi-objective scheduling problem, integrating energy, thermal, and cooling models. We employ a Gated Graph Convolution Network (GGCN) as a surrogate model to approximate Quality of Service (QoS) for system states and generate optimal scheduling decisions. The GGCN surrogate model, which simulates the behavior of more complex models, provides faster predictions while maintaining the essential characteristics and dynamics of the system. Our approach aims to optimize energy consumption, cooling costs, SLA violations, resource utilization, and response times, thereby enhancing the overall energy efficiency of CDCs.

Chapter 1
Introduction

1. Introduction

1.1 Introduction

Cloud computing has proven to be a reliable, cost-effective, and scalable choice for hosting and delivering software solutions across diverse industrial applications. Many businesses have migrated to cloud data centers (CDCs) to benefit from on-demand, elastic, and scalable resource provisioning, saving on capital investments and maintenance of in-house infrastructure. Major cloud providers like Amazon, Microsoft, and Google have experienced a surge in the number of CDCs to meet the increasing demands driven by AI and Internet of Things (IoT) applications. This rapid expansion necessitates significant energy consumption, particularly for cooling, which can match the energy required by computing nodes themselves. Effective resource management to lower energy consumption, costs, and carbon footprints is critical. Traditional techniques, such as heuristics and reinforcement learning, struggle with scalability and adaptability in dynamic environments, leading to suboptimal resource allocation. Addressing these challenges requires innovative approaches that balance performance and energy efficiency. We propose a holistic AI-based resource management technique designed for energy-efficient cloud computing, utilizing a Gated Graph Convolution Network (GGCN) to model the complex energy, thermal, and cooling dynamics of CDCs. By integrating a GGCN surrogate model for Quality of Service (QoS) estimation with a multi-objective scheduling technique.

1.2 Study Scope

This study focuses on developing and evaluating an AI-based holistic resource management technique for sustainable cloud computing, aiming to optimize energy efficiency in cloud data centers (CDCs) by integrating energy, thermal, and cooling models. Utilizing a Gated Graph Convolution Network (GGCN) as a surrogate model to estimate Quality of Service (QoS) for various system states and generate optimal scheduling decisions, the project seeks to simplify cloud processes such as allocation and monitoring. This is achieved through automation and providing these processes via a user-friendly interface and a machine learning module, thereby saving valuable time and effort, particularly for small businesses. The project includes the development of the Cloud Task Scheduler software solution,

implementing advanced scheduling algorithms like novel CNN models for energy consumption prediction. Additionally, it integrates these components into a cohesive system, tests the solution using appropriate datasets, and evaluates its energy efficiency and job completion rates.

1.3 Motivation

The motivation for this study arises from the increasing energy consumption of cloud data centers due to the growing demand for cloud resources, driven by data-intensive industries and the proliferation of AI and IoT applications. Traditional resource management techniques are insufficient in addressing the complex interdependencies between thermal characteristics, resource consumption, and scheduling decisions, necessitating a more sophisticated approach. An integral part of any computing business model is the cloud sector, which is crucial for providing building, deployment, and serverless backend services for web applications and dynamic websites. Both governmental agencies and the private sector have been collaborating over the past century to enhance cloud computing by developing new technologies to improve deployment and sustainability. The availability of reliable server systems capable of efficiently monitoring the cloud has made smart cloud solutions more common. Integrating cloud computing with artificial intelligence could potentially revolutionize web services. This project aims to mitigate the adverse effects of escalating energy consumption in cloud computing environments by developing an intelligent task scheduler. The goal is to optimize resource allocation, reduce operational costs, enhance system reliability, and contribute to the sustainability goals of cloud computing.

1.4 Purpose

The purpose of this study is to design and implement a holistic resource management technique that enhances the sustainability of cloud computing by optimizing energy efficiency. This is achieved by developing a model that considers energy, thermal, and cooling costs in scheduling decisions, ultimately reducing the operational costs and carbon footprint of cloud data centers. As one of the main pillars of any computing business model, the cloud computing sector is crucial for providing efficient deployment and backend services. This project aims to develop a software solution that optimizes task scheduling in cloud computing environments to minimize energy consumption while maintaining job completion rates. By leveraging advanced algorithms and artificial intelligence techniques, the project

addresses the growing energy consumption challenges faced by cloud providers and organizations utilizing cloud computing.

1.5 Problem Statement

The main problem addressed in this study is the inefficiency of traditional resource management techniques in cloud data centers, leading to high energy consumption, poor scalability, and increased operational costs. Existing methods fail to capture the dependencies between the thermal characteristics of hosts, resource consumption of tasks, and scheduling decisions, resulting in suboptimal energy usage. Cloud providers and organizations deploying cloud computing solutions face significant challenges related to escalating energy consumption, operational costs, and reliability issues. The lack of efficient resource allocation strategies leads to inefficient utilization of hardware resources, contributing to higher energy consumption and decreased system reliability.

1.6 Intended Use

The proposed AI-based resource management technique is intended for use by cloud service providers, data center operators, and organizations deploying cloud computing solutions to improve energy efficiency and sustainability in their operations. By integrating this technique into existing cloud management systems, users can optimize resource allocation, minimize energy consumption, achieve significant energy savings, improve operational efficiency, and enhance system reliability.

1.7 Intended Audience

The intended audience for the Cloud Task Scheduler project includes software developers, cloud computing engineers, system administrators, and decision-makers in organizations utilizing cloud computing. Additionally, researchers and practitioners in the fields of cloud computing, artificial intelligence, and optimization will benefit from the project's findings and methodologies. Specifically, this study targets:

- Cloud service providers and data center operators seeking to enhance sustainability.

- Researchers and practitioners focused on cloud computing, artificial intelligence, and energy management.
- Policymakers and regulatory bodies interested in promoting sustainable practices in the technology sector.

1.8 System Scope

The system scope of this project includes the development of software components such as the GGCN-based energy consumption prediction model, and integration modules. It also encompasses the design and implementation of the user interface, as well as documentation and deployment guidelines for end-users.
The system scope includes:

- Cloud Users: Submit workloads through IoT devices.
- Cloud Workload Management Portal: Interface for workload submission.
- Workload Manager: Processes workloads and realizes them as container instances.
- Cloud Broker: Manages job allocation and resource scheduling.
    - Service Manager: Manages SLAs and QoS requirements.
    - CDC Manager: Monitors resource utilization and performs task allocation and migration.
    - Resource Manager: Implements the GGCN-based scheduling model.

1.8.1 Benefits:

- Energy Efficiency: The Cloud Task Scheduler optimizes resource allocation to minimize energy consumption, leading to significant cost savings and environmental benefits.
- Operational Cost Reduction: By improving resource utilization and reducing energy consumption, the project helps organizations lower their operational costs associated with cloud computing.
- Enhanced Reliability: The intelligent task scheduler improves system reliability by mitigating thermal hotspots and optimizing workload distribution and resource utilization, resulting in better performance stability and improved compliance with SLAs and QoS requirements.
- Scalability and Flexibility: The Scheduler can adapt to varying workload demands and organizational needs, providing scalability and flexibility in resource allocation.

- Sustainability: By promoting energy-efficient resource allocation, the project contributes to the overall sustainability goals of cloud computing environments, aligning with global environmental initiatives.

1.8.2 Objectives:

- Develop a holistic resource management technique using AI for sustainable cloud computing.
- Address challenges by integrating energy, thermal, and cooling models into the scheduling decision process.
- Utilize a GGCN surrogate model for efficient QoS estimation.
- Validate the proposed technique through extensive experiments on simulated testbeds.
- Provide a user-friendly solution for seamless integration into existing cloud computing environments.
- Contribute to the sustainability goals of cloud computing by promoting energy-efficient resource allocation.

1.8.3 Goals:

- Implement an advanced task scheduling algorithm to minimize energy consumption while maintaining job completion rates.
- Design and integrate a Graph Gated Convolutional Network (GGCN) energy prediction model to forecast energy consumption based on host and task metrics.
- Achieve significant energy savings in cloud data centers.
- Enhance the sustainability of cloud computing operations.
- Thoroughly evaluate the Cloud Task Scheduler system to assess its effectiveness in minimizing energy consumption and improving job completion rates.
- Ensure high QoS and SLA compliance for cloud users.
- Identify areas for optimization and fine-tuning to enhance the system's performance and reliability.
- Document the design, implementation, and usage of the Cloud Task Scheduler system for easy understanding and deployment.
- Provide a scalable and adaptable resource management solution.

- Provide comprehensive deployment guidelines to facilitate the integration of the scheduler into diverse cloud computing setups.
- Disseminate the findings and methodologies of the project to relevant stakeholders, including software developers, cloud computing engineers, and researchers.
- Explore potential future directions for further enhancing the energy efficiency and performance of cloud computing systems based on the project's findings.

1.9 SWOT Analysis

1.9.1 Strengths:

- Innovative Approach:

  The project proposes novel techniques, such as GGCN models, to optimize task scheduling in cloud computing, showcasing innovation in the field.

- Potential for Significant Improvement:

  This project demonstrates a substantial improvement in energy consumption and job completion rates compared to existing baselines, indicating the potential for significant benefits.

- Interdisciplinary Integration:

  The project integrates insights from various domains, including cloud computing, artificial intelligence, and optimization, allowing for a comprehensive approach to addressing energy efficiency challenges.

- Practical Application:

  The proposed solution addresses real-world challenges faced by cloud providers and organizations, offering practical applications for enhancing energy efficiency and performance in cloud environments.

1.9.2 Weaknesses:

- Complexity of Implementation:

  Implementing advanced algorithms and neural network models may require significant computational resources and expertise, potentially posing challenges in deployment and maintenance.

- Data Dependency:

  The effectiveness of the proposed solution relies heavily on the availability and quality of data for training and evaluation, which may be limited or difficult to obtain in certain scenarios.

1.9.3 Opportunities:

- Market Demand:

  With the increasing adoption of cloud computing by businesses, there is a growing market demand for solutions that can optimize resource allocation and reduce energy consumption, presenting opportunities for the project's adoption and commercialization.

- Continued Research:

  The project opens avenues for further research and development in the optimization of cloud task scheduling, particularly in the integration of advanced artificial intelligence techniques and optimization algorithms.

1.9.4 Threats:

- Competitive Landscape:

  The field of cloud computing and optimization is highly competitive, with other researchers and organizations working on similar solutions. Competing

solutions may pose a threat to the adoption and market penetration of the Cloud Task Scheduler.

- Technological Obsolescence:

  Rapid advancements in technology may render certain components or approaches obsolete over time, necessitating continuous updates and adaptation to maintain relevance and effectiveness.

1.10 PESTEL Analysis

1.10.1 Political Factors:

- Data Protection Regulations:

  Government policies regarding data protection, privacy, and sovereignty can significantly impact cloud computing services. Compliance with regulations like GDPR (General Data Protection Regulation) in the EU and data localization laws in various countries influences how data is stored and managed.

- Government Surveillance Laws:

  Laws related to government surveillance and access to data can affect customer trust in cloud services. Companies need to navigate between compliance with legal requirements and maintaining customer privacy.

1.10.2 Economic Factors:

- Cost Pressures:

  Economic conditions such as inflation, currency fluctuations, and recession can impact the affordability and demand for cloud services. Organizations may prioritize cost-cutting measures, affecting their investment in cloud infrastructure and services.

- Pricing Models:

  Changes in pricing models for cloud services by providers can influence adoption rates and competitiveness within the industry. Pricing strategies need to align with market demands and cost structures.

1.10.3 Social Factors:

- Technological Adoption:

  Social acceptance and adoption of technology influence the demand for cloud services. Factors such as digital transformation initiatives, remote work trends, and reliance on mobile devices shape the market landscape for cloud computing.

- Environmental Awareness:

  Increasing awareness of environmental sustainability may drive demand for eco-friendly cloud solutions. Providers may face pressure to adopt green technologies and reduce carbon footprints in data centers.

1.10.4 Technological Factors:

- Advances in Technology:

  Rapid technological advancements, such as edge computing, artificial intelligence, and quantum computing, impact the capabilities and competitiveness of cloud services. Cloud providers need to innovate continuously to stay ahead in the market.

- Security and Privacy Technologies:

  Evolving cybersecurity threats and privacy concerns require constant updates to security measures and encryption technologies in cloud infrastructure. Trust in cloud services depends on robust security features.

1.10.5 Environmental Factors:

- Energy Consumption:

  The energy consumption of data centers and associated carbon emissions are significant environmental considerations for the cloud computing industry. Efforts to optimize energy efficiency and utilize renewable energy sources are crucial for sustainability.

- E-waste Management:

  The disposal of electronic waste generated by obsolete hardware and data center equipment poses environmental challenges. Cloud providers need to implement responsible e-waste management practices to minimize environmental impact.

1.10.6 Legal Factors:

- Intellectual Property Rights:

  Legal frameworks governing intellectual property rights, patents, and copyrights impact the development and use of cloud technologies. Compliance with intellectual property laws is essential for both providers and users of cloud services.

- Antitrust Regulations:

  Antitrust regulations and competition laws influence market dynamics and the behavior of dominant cloud providers. Regulatory scrutiny may affect mergers, acquisitions, and business practices within the industry.

1.11 Marketing Mix (4Ps)

1.11.1 Product:

Our cloud computing service offers IaaS, PaaS, and SaaS solutions tailored to diverse business needs, prioritizing security, reliability, and seamless integration with existing systems.

1.11.2 Price:

We provide flexible pricing plans including pay-as-you-go, subscription-based, and custom models, with discounts for long-term contracts and startups.

1.11.3 Place:

Access our services through our intuitive online platform, ensuring global availability with regional data residency options. Collaborate with strategic partners to extend market reach and offer bundled solutions.

1.11.4 Promotion:

Utilize digital marketing channels such as LinkedIn posts, thought leadership content, industry events (tech summits), and referral programs to raise brand awareness, drive conversions, and foster customer loyalty.

1.12 Acronyms and Abbreviations

Abbreviation	Full Form
CDC	Cloud Data Centers
AI	Artificial Intelligence
FSM	Finite State Machine
UML	Unified Modeling Language
QoS	Quality of Service
IoT	Internet of Things
RL	Reinforcement Learning
SLA	Service Level Agreement

CPU	Central Processing Unit
RAM	Random Access Memory
GGCN	Gated Graph Convolution Network
DQL	Deep Q Learning
ML	Machine Learning
ANN	Artificial Neural Network
NARX	Nonlinear Auto-Regressive Network with Exogenous Inputs
HDIC	Holistic Dynamic Inter-Cloud
LSTM	Long Short-Term Memory
CRUZE	Cooling and Resource Utilization in Zero Energy Data Centers
FECBench	Fault Emulation and Classification Benchmark
MALE	Memory Aware Load balancing and Energy Efficiency
TOPSIS	Technique for Order of Preference by Similarity to Ideal Solution
MITEC	Memetic Inspired Task Scheduler for Energy Efficient Computing
GRANITE	Green Adaptive Networking Integrated with Trusted Environment
PADQN	Priority Aware Deep Q-Network
SDAE-MMQ	Stacked Denoising Autoencoder based Multi-Model Q-Learning
LAN	Local Area Network
VLAN	Virtual LAN

Table 1

CHAPTER 2
OVERALL DESCRIPTION

2. Overall Description

2.1 System Perspective

The system aims to optimize the energy efficiency of cloud data centers (CDCs) by leveraging artificial intelligence to manage resources holistically. This approach integrates energy, thermal, and cooling models into a multi-objective scheduling framework. Unlike traditional methods that focus solely on computational resource management, the system considers the complex interdependencies between various system components. By using a Gated Graph Convolution Network (GGCN), the cloud task scheduler can accurately approximate Quality of Service (QoS) scores and generate optimal scheduling decisions in real-time, making it particularly effective in environments with dynamic and non-stationary resource demands. The project aims to develop a comprehensive software solution that optimizes task scheduling in cloud computing environments to minimize energy consumption while maintaining performance and reliability. The Cloud Task Scheduler will interact with various components, including cloud providers, networking infrastructure, computational resources, and user interfaces, leveraging advanced algorithms and AI models to intelligently allocate resources, manage workloads, and optimize energy efficiency.

2.2 System Functional requirements

- Workload Processing:
    - Accept Workloads: The system should accept workloads submitted by users through a graphical user interface (GUI).
    - Container Instances: Convert incoming workloads into container instances for better resource management and portability.
    - Workload Management: CTS should effectively manage the distribution of tasks across cloud nodes to optimize resource utilization and minimize energy consumption.
- Resource Monitoring:

    - Continuous Monitoring: The system must continuously monitor the resource utilization of all active tasks and hosts.
    - Track QoS Parameters: Maintain up-to-date information on QoS parameters, including energy consumption, thermal characteristics, and resource usage.
    - Resource Allocation: The system must efficiently allocate computing resources based on workload demands, considering factors such as energy consumption, performance, and reliability.

- Task Scheduling:

    - QoS Estimation: Utilize the GGCN surrogate model to estimate QoS parameters accurately for different scheduling decisions.
    - Multi-Objective Optimization: Optimize scheduling based on integrated energy, thermal, and cooling models to achieve holistic resource management.
    - Dynamic Adaptation: Adjust scheduling decisions dynamically to respond to changing resource demands and maintain optimal performance.

- Task Migration:

    - Container Checkpoints: Enable the transfer and restoration of container checkpoints to facilitate efficient task migration.
    - Hotspot Reduction: Actively work to minimize thermal hotspots and distribute workloads to avoid performance degradation.

- Performance Optimization:

    - Heuristic Search: Implement performance to power ratio heuristics to explore the scheduling search space efficiently, reducing the time to converge to optimal decisions.
    - Model Adaptation: Periodically adjust the weights of the GGCN model using backpropagation to adapt to new workloads and volatile conditions.

- Energy Efficiency Optimization:
    - Utilizes advanced algorithms and models to predict and optimize energy consumption.
    - Incorporates energy, thermal, and cooling models to holistically manage the data center.
    - Leverages a Gated Graph Convolution Network (GGCN) for Quality of Service (QoS) estimation and scheduling decisions.
- Fault Tolerance:
    - Implements redundancy and replication to prevent single points of failure.
    - Uses dynamic load balancing to redistribute workloads from failing nodes.
    - Incorporates checkpointing and rollback mechanisms for quick recovery.
    - Employs automated self-healing and machine learning to predict and mitigate potential failures.
- Analysis:
    - Provides real-time monitoring and visualization of resource usage and performance metrics.
    - Stores historical data for trend analysis and forecasting.
    - Generates regular reports and alerts to keep administrators informed.
    - Offers optimization insights and recommendations based on collected data.
- User Interface:
    - Features a centralized dashboard for key metrics and interactive elements for detailed information.
    - Includes a configuration management panel for easy system parameter adjustments.
    - Visualizes performance data with charts, graphs, and heatmaps.

    - Implements an alert system for critical events and customizable notification settings.

- Scalability:

    - Supports elastic resource provisioning to adjust computing nodes based on workload demands.
    - Implements advanced load balancing to distribute workloads evenly.
    - Manages large clusters of heterogeneous resources, including private and public cloud nodes.
    - Supports both horizontal and vertical scaling to handle varying demands efficiently.

These requirements are further explained in the system features section.

2.3 User Classes and Characteristics

Users of said service will vary from small companies and startups as well as regular businesses that aim to start optimizing their own cloud solution primarily divided into regular business users (most important user class) and small businesses/startups users.

Profiling the users:

- Cloud Users:

    - Diverse Backgrounds: Can be individuals or organizations requiring cloud services.
    - QoS and SLA Focused: Need reliable cloud services with specific quality of service and service level agreement requirements.
    - GUI Interaction: Use a graphical user interface to submit workloads and monitor their execution.

- System Administrators:

    - Operational Oversight: Manage the overall operations of the cloud data center.

- Performance Monitoring: Continuously monitor system performance and resource utilization.
- Model Configuration: Configure and maintain the GGCN model and other resource management parameters.

- Developers:

  - System Maintenance: Develop and maintain the system's software components.
  - Feature Implementation: Implement new features and optimize existing ones based on user feedback and system performance data.
  - Scalability Focused: Ensure the system scales efficiently to handle varying workloads and resource demands.

2.4 Methodology

- Requirement Analysis:

  - Conduct a thorough analysis of user requirements, system constraints, and environmental factors to define the functional and non-functional requirements of the cloud task scheduler system.
  - Engage with stakeholders to understand user needs and system constraints.

- Algorithm and Model Selection:

  - Evaluate existing algorithms, models, and techniques for task scheduling, resource allocation, and energy optimization in cloud computing.
  - Select appropriate methods based on their effectiveness and suitability for the project goals.

- System Design:

  - Design the overall architecture focusing on integrating the GGCN model with energy, thermal, and cooling management modules, and

components of the cloud task scheduler system, including modules for workload management, energy optimization, fault tolerance, monitoring, and user interface.
- Develop detailed models for each component to ensure seamless integration and optimal performance.

- Implementation:
    - Develop the software components of the system according to the design specifications, utilizing programming languages, frameworks, and libraries suitable for algorithm implementation, data processing, and user interface development.
    - Implement core system components including workload processing, resource monitoring, and task scheduling modules.
    - Develop and integrate the GGCN surrogate model for QoS estimation.

- Testing and Validation:
    - Conduct extensive testing and validation of the cloud task scheduler system to ensure its functionality, performance, reliability, and usability.
    - Use the CloudSim toolkit to simulate various scenarios and validate the system's performance.
    - Perform unit testing, integration testing, and system testing to identify and resolve any issues or discrepancies.
    - Compare the results against state-of-the-art schedulers to ensure superior performance in terms of energy efficiency and QoS.

- Deployment and Evaluation:
    - Deploy the cloud task scheduler system in a realistic environment, such as a cloud computing infrastructure or simulation environment.
    - Evaluate its performance, energy efficiency, and impact on workload management through empirical testing and analysis.
    - Monitor system performance continuously and make necessary adjustments to optimize efficiency.

- Maintenance and Upgrades:

    - Prepare comprehensive documentation covering system design, implementation details, user guidelines, and maintenance procedures.
    - Regularly update the GGCN model and other system components based on performance data and user feedback.
    - Ensure ongoing maintenance and support for the deployed system, including updates, bug fixes, and enhancements as needed.
    - Implement new features and optimizations to keep the system current with technological advancements and changing user needs.

2.5 Design and Implementation Constraints

Compatibility:

- The project must ensure compatibility with existing cloud computing infrastructures, including various hardware configurations, operating systems, and networking protocols.

Scalability:

- The solution should be scalable to accommodate varying workload demands and support the dynamic expansion or contraction of computing resources without sacrificing performance or energy efficiency.

Resource Limitations:

- Consideration must be given to resource limitations, such as memory, processing power, and storage capacity, especially when deploying the solution on resource-constrained devices or environments.

Data Privacy and Security:

- The project must adhere to data privacy and security regulations, ensuring that sensitive information is protected from unauthorized access, tampering, or disclosure.

Interoperability:

- The system should support interoperability with other software tools and platforms commonly used in cloud computing environments, enabling seamless integration and data exchange.

Real-time Processing:

- Depending on the application requirements, the project may need to support real-time data processing and decision-making, imposing constraints on processing speed and latency.

Training Data Availability:

- Availability and quality of training data for machine learning models, such as neural networks, may pose constraints on model development and performance.

Algorithm Complexity:

- Complex algorithms, such as deep reinforcement learning or graph neural networks, may require significant computational resources and expertise for implementation, training, and optimization.

Testing and Validation:

- Comprehensive testing and validation processes must be carried out to ensure the reliability, accuracy, and robustness of the solution under various scenarios and conditions.

Budget and Time Constraints:

- The project must adhere to budget and time constraints, balancing the need for thorough development and testing with project deadlines and resource limitations.

Regulatory Compliance:

- The solution should comply with relevant regulatory requirements and standards governing cloud computing, data protection, and energy efficiency.

Documentation and Training:

- Adequate documentation and training materials should be provided to users and administrators to facilitate system deployment, configuration, and maintenance.

2.6 Assumptions and Dependencies

Accurate Resource Metrics:

- The system assumes continuous and accurate monitoring of resource utilization and QoS parameters, relying on reliable sensors and monitoring tools.

Model Accuracy:

- The accuracy of the GGCN surrogate model is crucial for providing reliable QoS estimations, depending on proper training and regular updates.

Infrastructure Support:

- The system assumes robust cloud infrastructure support, including hardware and network capabilities to handle resource management operations efficiently.

Interoperability:

- The system must interoperate with existing cloud management tools and frameworks, depending on compatibility with industry standards and protocols.

2.7 Operating Environment

The system operates within a hybrid public-private cloud environment, consisting of:

Private Cloud:

- Resource-constrained nodes offering low latency services.
- Nodes are located within the same Local Area Network (LAN) as the cloud broker.

Public Cloud:

- Geographically distant nodes connected via a Virtual LAN (VLAN).
- Nodes provide abundant resources but have higher communication latency.

The system's components, such as the cloud broker, workload manager, and resource manager, must function seamlessly across both private and public cloud environments. This ensures efficient resource management, real-time processing, and dynamic adaptation to changing workloads and resource demands. The operating environment must support continuous monitoring, real-time scheduling, and robust task migration capabilities to maintain optimal performance and energy efficiency.

CHAPTER 3
REQUIREMENTS SPECIFICATION

3. Requirements Specification

Functional requirements define what a product must do and what its features and functions are. Nonfunctional requirements, also known as quality attributes, describe the general properties of a system. In this section we will thoroughly specify both our functional and nonfunctional requirements in detail.

3.1 Functional Requirements

3.1.1 Workload Submission

Description
This functionality allows cloud users to submit their workloads along with specific SLA and QoS requirements through the Cloud Workload Management Portal.

Requirement(s)
The system must provide a user interface for cloud users to input their workloads, including details such as computational requirements, deadlines, and performance expectations.

3.1.2 Admission Control

Description
Upon receiving incoming workloads, this functionality ensures that they are processed and realized as container instances within the system.

Requirement(s)
The system must have an admission controller that verifies incoming workloads, allocates appropriate resources, and initializes corresponding container instances.

3.1.3 Integration of AI Models

Description
Integrate artificial intelligence models such as the gated graph convolutional network (GGCN) for optimizing resource allocation and task scheduling.

Requirement(s)
The system must integrate AI models to enhance resource allocation and task scheduling capabilities, leveraging techniques like deep reinforcement learning, recurrent neural networks, and graph neural networks.

3.1.4 Task Allocation

Description
The Cloud Broker allocates submitted jobs to various computing resources based on their resource requirements and SLA/QoS constraints.

Requirement(s)
The system must employ a task allocation mechanism that considers the resource availability, workload characteristics, and SLA/QoS agreements to assign tasks to suitable cloud worker nodes.

3.1.5 Resource Monitoring

Description
The CDC Manager continuously monitors the resource utilization of all active tasks and hosts in the system, including energy and thermal characteristics.

Requirement(s)
The system must integrate resource monitoring capabilities to collect real-time data on CPU usage, memory utilization, temperature, and energy consumption for all cloud hosts and tasks.

3.1.6 Resource Scheduling

Description
The Resource Manager decides the schedule for each task in the system, considering energy, thermal, and cooling sustainability models.

Requirement(s)
The system must implement a resource scheduling algorithm that optimizes task placement based on energy efficiency, thermal management, and cooling requirements while meeting SLA/QoS objectives.

3.1.7 Exploration Strategy

Description
The Resource Manager employs an exploration strategy to evaluate multiple scheduling decisions and select the most optimal one based on QoS scores.

Requirement(s)
The system must include an exploration strategy that systematically explores the scheduling decision space, considering various factors such as task dependencies, resource availability, and system constraints.

3.1.8 Task Migration

Description
The system supports task migration, involving the transfer and restoration of container checkpoints, to optimize resource utilization dynamically.

Requirement(s)
The system must enable task migration to balance resource usage, mitigate hotspots, and adapt to changing workload conditions by seamlessly transferring tasks between cloud worker nodes.

3.2 Nonfunctional Requirements

3.2.1 Performance Requirements

Description
The system should efficiently handle many incoming workloads with minimal latency and high throughput.

Requirement(s)
The system must ensure that workload processing and task allocation are performed within acceptable time frames, maintaining low response times and maximizing system throughput.

3.2.2 Reliability Requirements

Description
The system should operate reliably, minimizing the risk of service disruptions or data loss.

Requirement(s)
The system must implement fault-tolerant mechanisms, redundancy, and backup strategies to ensure continuous operation and data integrity in the event of failures or disruptions.

3.2.3 Security Requirements

Description
The system must implement robust security measures to protect sensitive data and ensure the integrity and confidentiality of user information.

Requirement(s)
The system must enforce access control, encryption, authentication, and auditing mechanisms to safeguard data and prevent unauthorized access or malicious activities.

3.2.4 Usability Requirements

Description
The Cloud Workload Management Portal should have a user-friendly interface, making it easy for cloud users to submit and manage their workloads.

Requirement(s)
The user interface must be intuitive, well-designed, and accessible, providing clear instructions and feedback to users throughout the workload submission and management process.

3.2.5 Scalability Requirements

Description
The system should be scalable to accommodate increasing numbers of cloud users, workloads, and data center resources.

Requirement(s)
The system must be designed to scale horizontally and vertically to support the growth of cloud infrastructure and accommodate fluctuating demand from users.

3.2.6 Maintainability Requirements

Description
The system should be designed and implemented in a modular and maintainable manner, facilitating future updates, enhancements, and bug fixes.

Requirement(s)
The system must adhere to coding standards, documentation practices, and software engineering best practices to ensure readability, extensibility, and ease of maintenance by developers and administrators.

3.2.7 Interoperability Requirements

Description
The system should be compatible with different cloud infrastructures and technologies, allowing seamless integration with existing systems and services.

Requirement(s)
The system must support standard protocols, APIs, and data formats to facilitate interoperability with various cloud platforms, tools, and services, enabling easy integration and data exchange between different components and environments.

3.2.8 Performance Monitoring Requirements

Implement monitoring tools and logging mechanisms to track system performance, resource utilization, and potential bottlenecks for optimization and troubleshooting.

3.2.9 Compliance Requirements

Description

The system should comply with relevant regulations and standards regarding data privacy, security, and environmental sustainability.

Requirement(s)

The system must adhere to industry regulations, compliance frameworks, and environmental standards, ensuring that data handling practices, security measures, and resource management policies align with legal and ethical requirements.

3.2.10 Cost-effectiveness

Description

The system should optimize resource utilization to minimize energy consumption and operational costs for cloud providers and users.

Requirement(s)

The system must prioritize energy-efficient resource management strategies and cost-effective allocation decisions, aiming to reduce overall operational expenses and carbon footprints while maintaining performance and QoS standards.

3.2.11 Adaptability

Description

The system should adapt to changes in workload patterns, resource availability, and environmental conditions.

Requirement(s)

The system must be designed with flexible architectures and adaptive algorithms that can dynamically adjust resource allocation and scheduling strategies in response to changing conditions and requirements.

CHAPTER 4
SYSTEM ANALYSIS

4. System Analysis

Systems analysis is the process of studying a procedure or business to identify its goal and purposes and create systems and procedures that will efficiently achieve them. In this section we will discuss and analyze different Apps to illicit our system requirements.

4.1 Data Gathering

After thorough research we found that the previous work was missing some key features. None of the methods we have encountered offered process scheduling automation along with machine learning algorithms in conjunction with utilizing the variation of cooling, energy, and thermal output. In our research we found two methodologies that are relevant candidates for our case study, Meta-Heuristic Methods and Reinforcement Learning. The dataset decided upon was COSCO: Container Orchestration using Co-Simulation and Gradient Based Optimization for Computing Environments, it includes several virtual machine instances with their metrics such as inter-task dependencies, energy consumption and thermal characteristics.

4.2 Related Work

In this section, we will discuss the functionalities offered by Meta-Heuristic Methods and Reinforcement Learning. And provide examples from within each.

4.2.1 Meta-Heuristic Methods.

Previous work in the field of cloud resource management has focused on reducing energy consumption while maintaining system reliability. Notable contributions include:

- CRUZE: Utilizes efficient design models and a Cuckoo optimization approach to reduce energy consumption and enhance system reliability.
- FECBench: Develops performance interference prediction models using machine learning to minimize profiling costs by exploring multidimensional resource metrics.

- MALE: Aims to minimize energy consumption by reducing memory consumption and contention through optimal mapping of virtual machines to cloud hosts.
- TOPSIS: Uses heuristics based on thermal features to reduce energy consumption and prevent overheating via threshold-based load balancing.
- MITEC: Applies a genetic algorithm to optimize scheduling decisions, updating energy and thermal models to improve fitness scores.

Other works in this category propose autonomic resource management mechanisms leveraging multiple resource layers and host heterogeneity. However, many methods, including CRUZE, Ella-W, and GRANITE, lack adaptability in volatile settings. CRUZE and MITEC are used as baselines for comparison in this study.

4.2.2 Reinforcement Learning.

Recent machine learning (ML) based schedulers aim to optimize energy consumption in cloud data centers (CDCs) using reinforcement learning (RL) techniques, particularly deep Q learning (DQL). Key approaches include:

- PADQN and SDAE-MMQ: Use DQL to make task placement decisions based on QoS scores. SDAE-MMQ enhances this with a stacked denoising autoencoder and MiniMax-Q learning.
- HDIC: Utilizes a nonlinear auto-regressive network (NARX) for value prediction in scheduling.
- ANN Approach: Applies an artificial neural network to produce task scheduling decisions, trained using QoS metrics like energy consumption and execution time.

These methods generally scale well but struggle with inter-task dependencies and adapting quickly to changes. They often focus on specific aspects of resource management rather than a comprehensive approach.

Our study compares these methods with a proposed holistic approach that integrates energy, thermal, and cooling considerations to optimize scheduling while reducing overheads.

4.3 Data Analysis

After gathering the data and analyzing both, we concluded that there isn’t an appropriate model that provides full task scheduling automation with machine

learning algorithms that takes into consideration thermal output, cooling, and energy efficiency. In our design we include said content as well as mobile application integration.

CHAPTER 5
SYSTEM DESIGN

5. System Design

System design is the process of designing the elements of a system such as the architecture, modules, and components, the different interfaces of those components, and the data that goes through that system. In this section we will review the Diagrams utilized in designing this system as well as the intended Technologies and Hardware for this system.

5.1 Diagrams & Illustrations

5.1.1 System Model

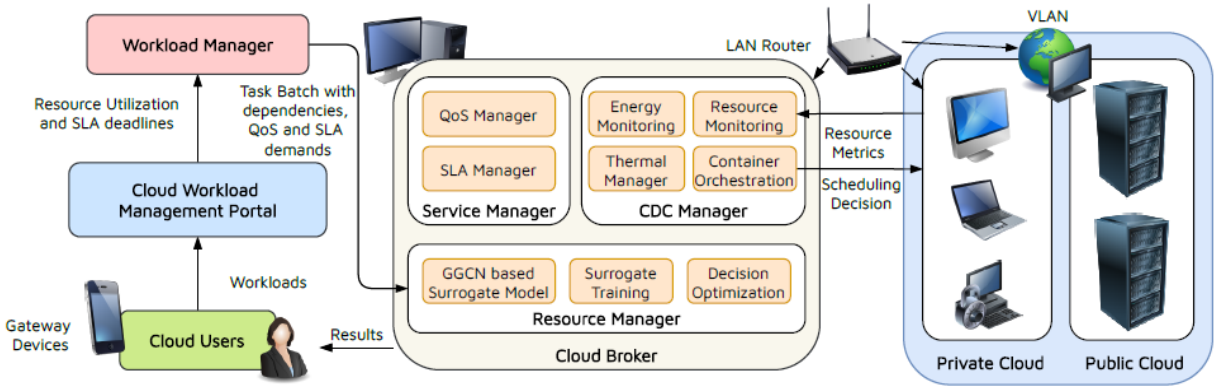

Fig. 1

- Cloud Users: The users share workloads as jobs to the CDC. The data is collected using IoT sensors and passed on to the CDC using gateway devices like smartphones and tablets.
- Cloud Workload Management Portal: A graphical user interface for cloud users to interact with the system for the submission of workloads along with their SLA and QoS details.
- Workload Manager: Initially, the workload manager processes all the incoming workloads. An admission controller realizes all workloads as container instances.
- Cloud Broker: The central cloud server that allocates incoming jobs to various computing resources (cloud worker nodes). It consists of the following components:

    - Service Manager: Contains two elements, SLA and QoS managers that manage the heterogeneous cloud services while processing workloads. The QoS manager contains the information about QoS requirements for different workloads, while the SLA manager contains the information about an agreement signed between a cloud user and a provider based on QoS requirements.
    - CDC Manager: Continually monitors the resource utilization of all active tasks and hosts in the system. It also monitors the QoS parameters (including the energy and thermal characteristics of cloud hosts) and performs the task allocation and migration. In this work, we assume tasks as container instances and task migration as the transfer and restoration of container checkpoints.
    - Resource Manager: Decides the schedule for each task in the system. The resource manager includes the sustainability models for energy, thermal and cooling parts of the CDC. For resource scheduling, the manager contains a GGCN based surrogate model that estimates QoS parameters. It performs training and on-the-fly tuning of the GGCN model to adapt in non-stationary settings. This manager also runs an exploration strategy that checks the QoS scores for a set of allocations and chooses the best one as the scheduling decision.
- Cloud Hosts: The cloud broker is connected to a heterogeneous set of worker nodes. Some nodes are present in the same Local Area Network (LAN) as the broker, called the private cloud. Others are present in a geographically distant location and connected using a virtual LAN (VLAN). As is common practice, we assume that private cloud nodes are resource constrained but offer low latency services, and public-cloud nodes have abundant resources but have high communication latency.

The scheduler resides as the Resource Manager in the Cloud Broker, taking tasks as inputs from the Workload Manager. It uses resource metrics from the Resource Monitor and executes scheduling decisions through Container Orchestration (as tasks are realized as containers in our system).

5.1.2 GGCN Model

A Gated Graph Convolutional Network (Gated GCN) is an advanced type of Graph Neural Network (GNN) designed to process graph-structured data. It improves traditional Graph Convolutional Networks (GCNs) by incorporating gating mechanisms to better control information flow between nodes.

Key Features of GGCN:

1. Edge-conditioned Convolutions
    - Unlike standard GCNs that aggregate neighboring node features using a fixed aggregation function, Gated GCNs learn edge-specific transformations, making them more flexible.
2. Gated Mechanisms (e.g., GRU or LSTM-like Units)
    - Gated GCNs use gating mechanisms (like in Recurrent Neural Networks) to regulate how much information is passed through each edge.
    - This prevents over smoothing, a common issue in deep GCNs.
3. Message Passing Framework
    - Each node updates its state by aggregating information from its neighbors while considering edge features through a gating function.
4. Higher Expressive Power
    - Gated GCNs can capture more complex dependencies in graphs compared to traditional GCNs, making them useful in applications like:
        - Social Networks
        - Molecular Graphs
        - Traffic Networks
        - Recommendation Systems

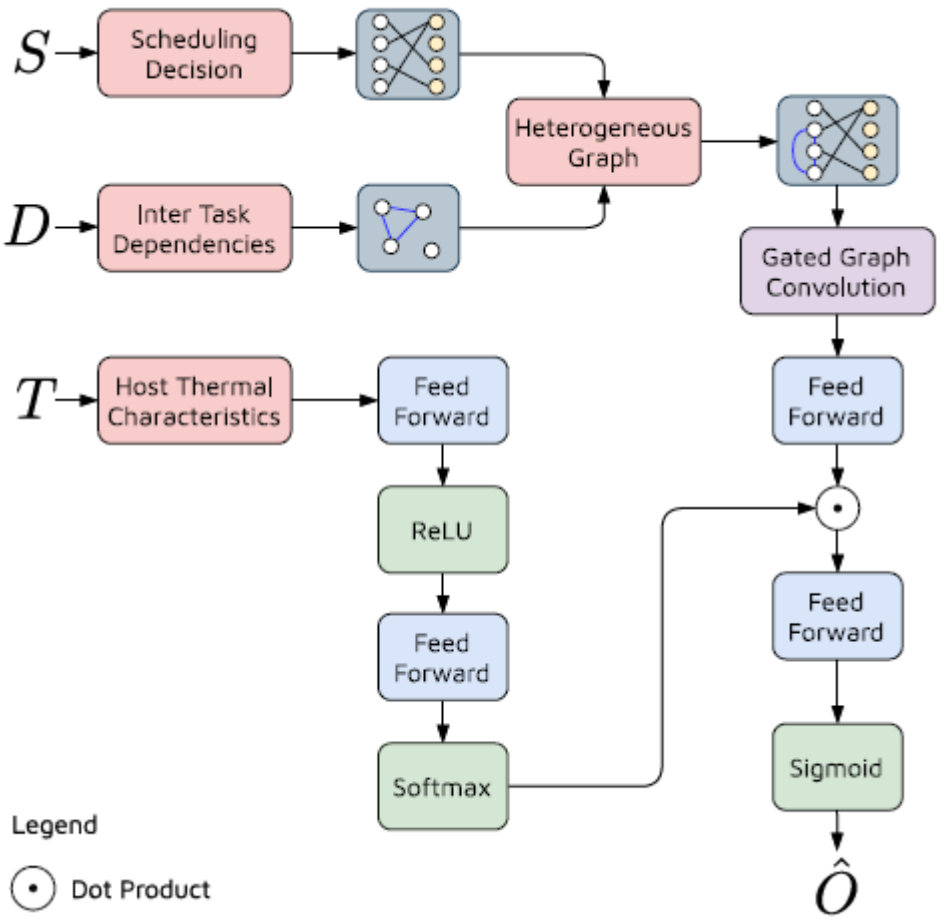

Fig. 2

The three inputs to the model and the graph structure are shown in red. Feed-forward and graph convolution operations are shown in blue and purple respectively. All activations are shown in green, and all data structures are shown in grey.

5.1.3 Graphical Representation of The Model

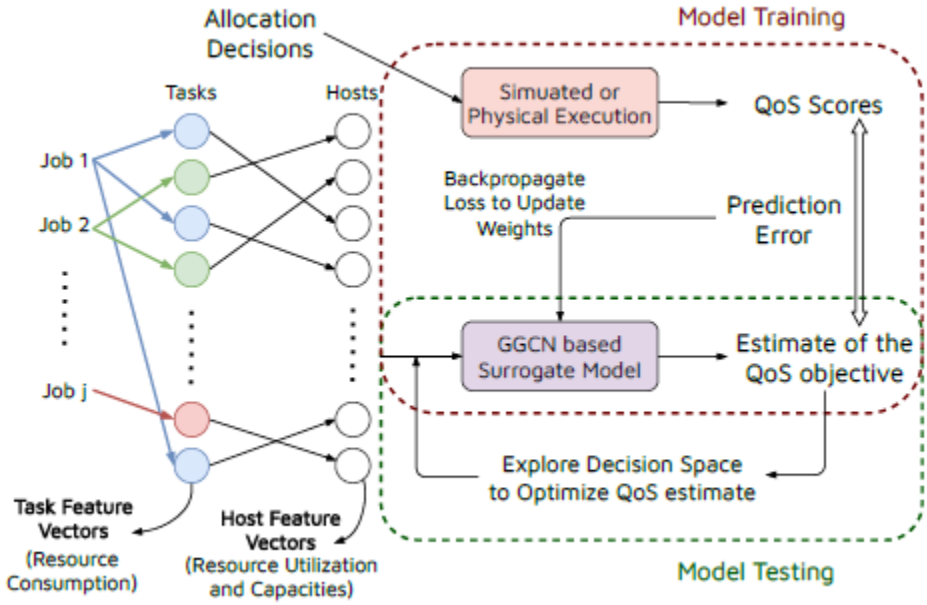

Fig. 3

The sigmoid operator allows us to generate an output within [0; 1] to enable training with normalized QoS scores. The GGCN model is agnostic to the QoS objective in general; however, in our work we use energy, temperature and SLA violation rates to train and fine-tune the model. To train the GGCN model, we use the Mean-Square-Error (MSE) loss between the predicted and ground-truth QoS scores.

5.1.4 Use-Case Diagram

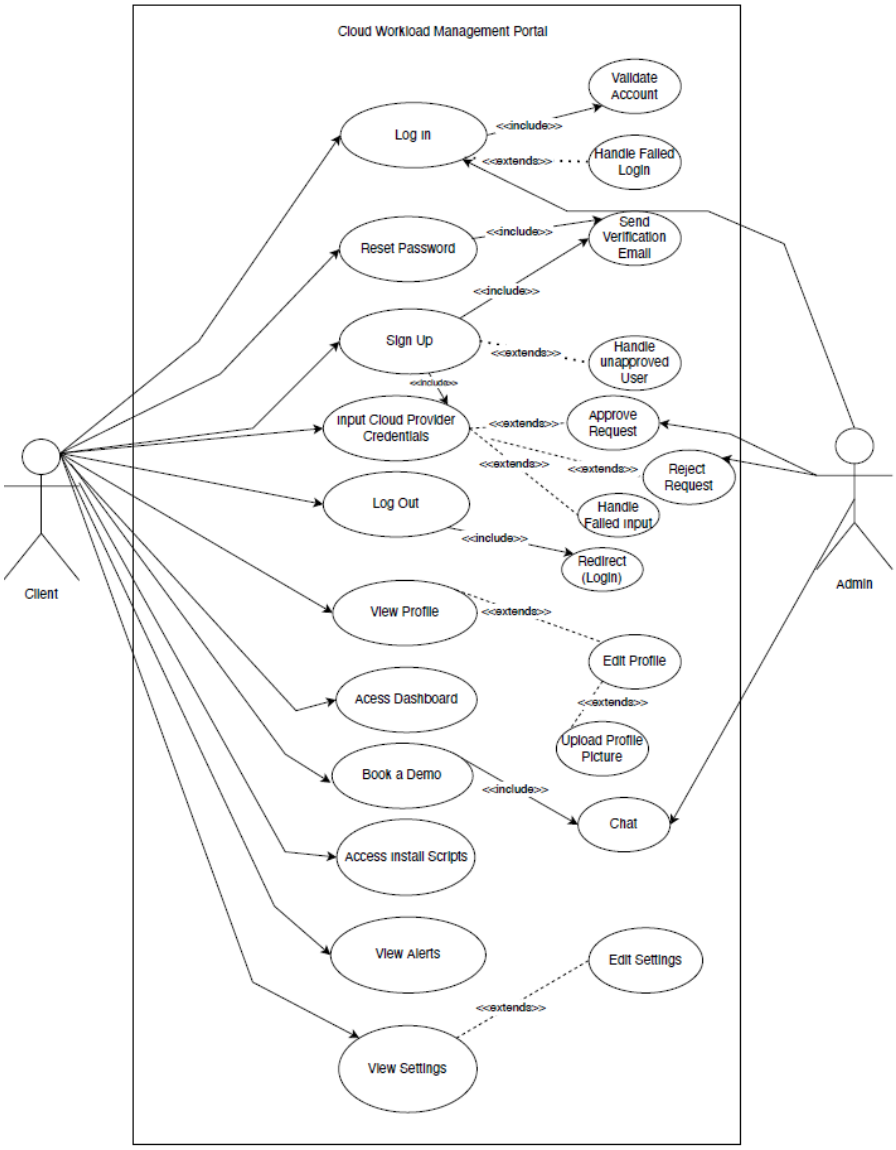

Fig. 4

The use case diagram represents the primary form of system requirements for our system. Use cases specify the expected behavior (what), and not the exact method of making it happen (how). A key concept of use case modeling is that it helps us design the system from the end user's perspective. It is an effective technique for communicating system behavior in the user's terms by specifying all externally visible system behavior.

5.1.5 Sequence Diagram 1

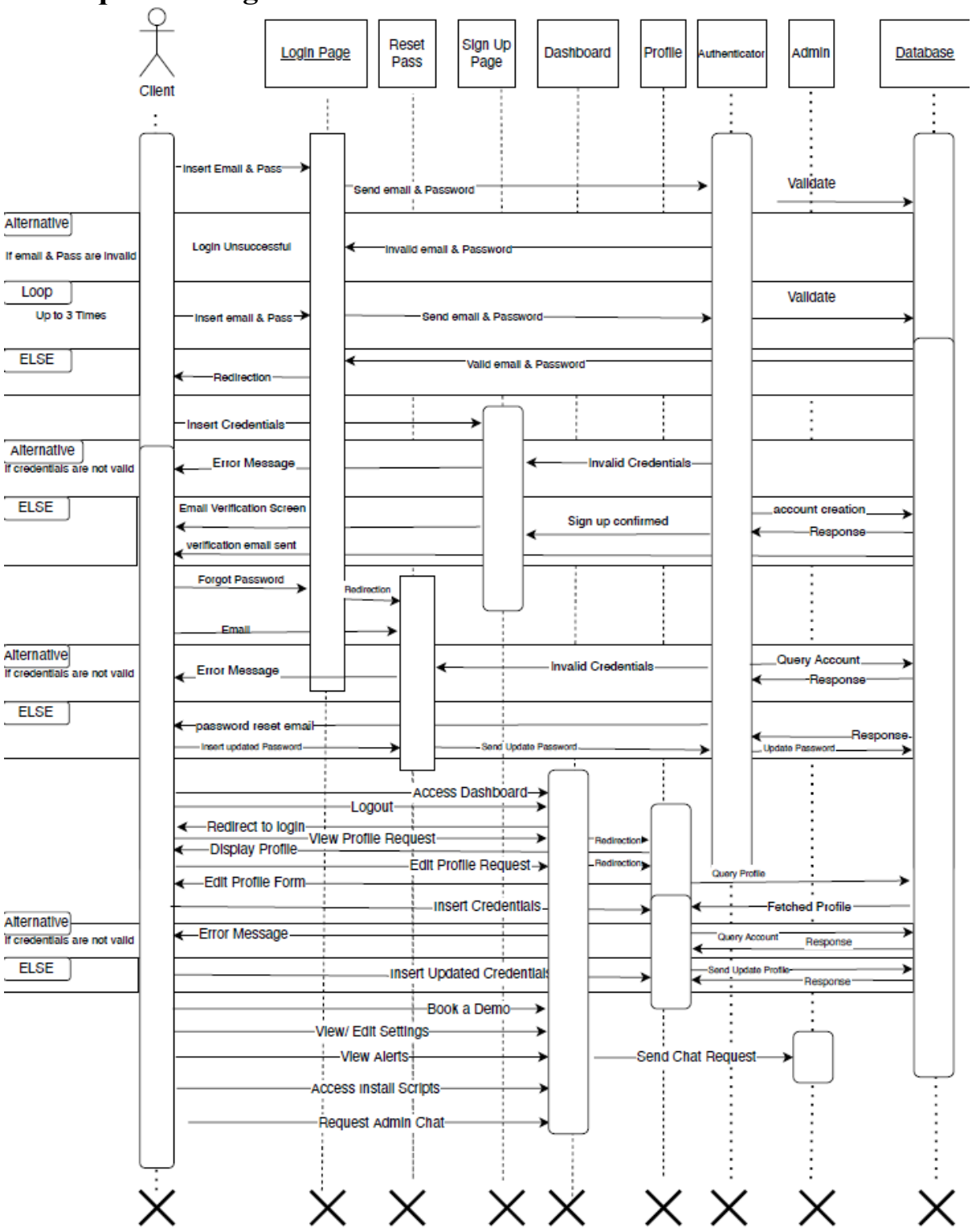

Fig. 5

5.1.6 Sequence Diagram 2

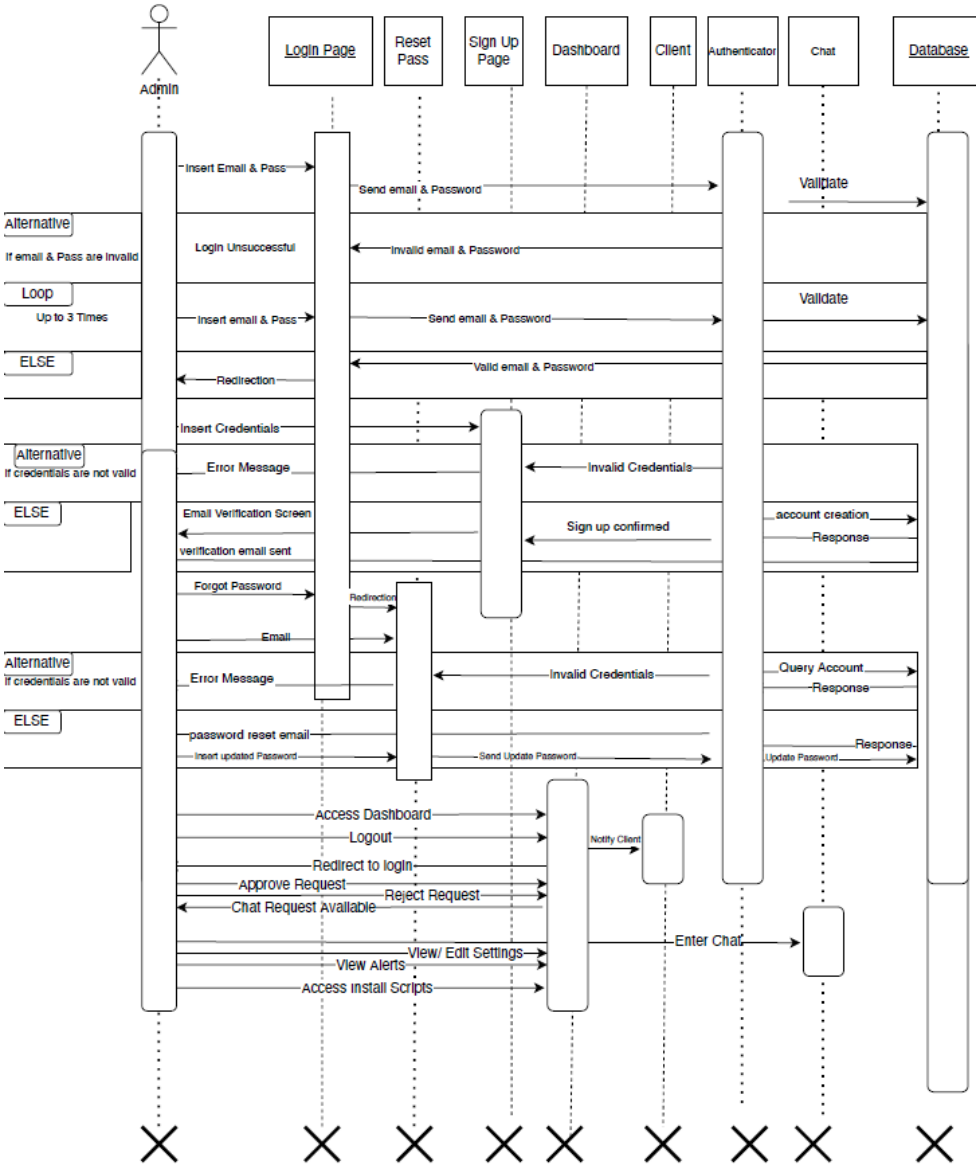

Fig. 6

A sequence diagram depicts the interaction between objects over time. It shows objects as vertical lines and messages exchanged as horizontal arrows. Events are arranged chronologically from top to bottom, illustrating the flow of control and data. Activation boxes indicate the duration an object is active during the interaction.

5.1.7 Finite-State Machine Diagram

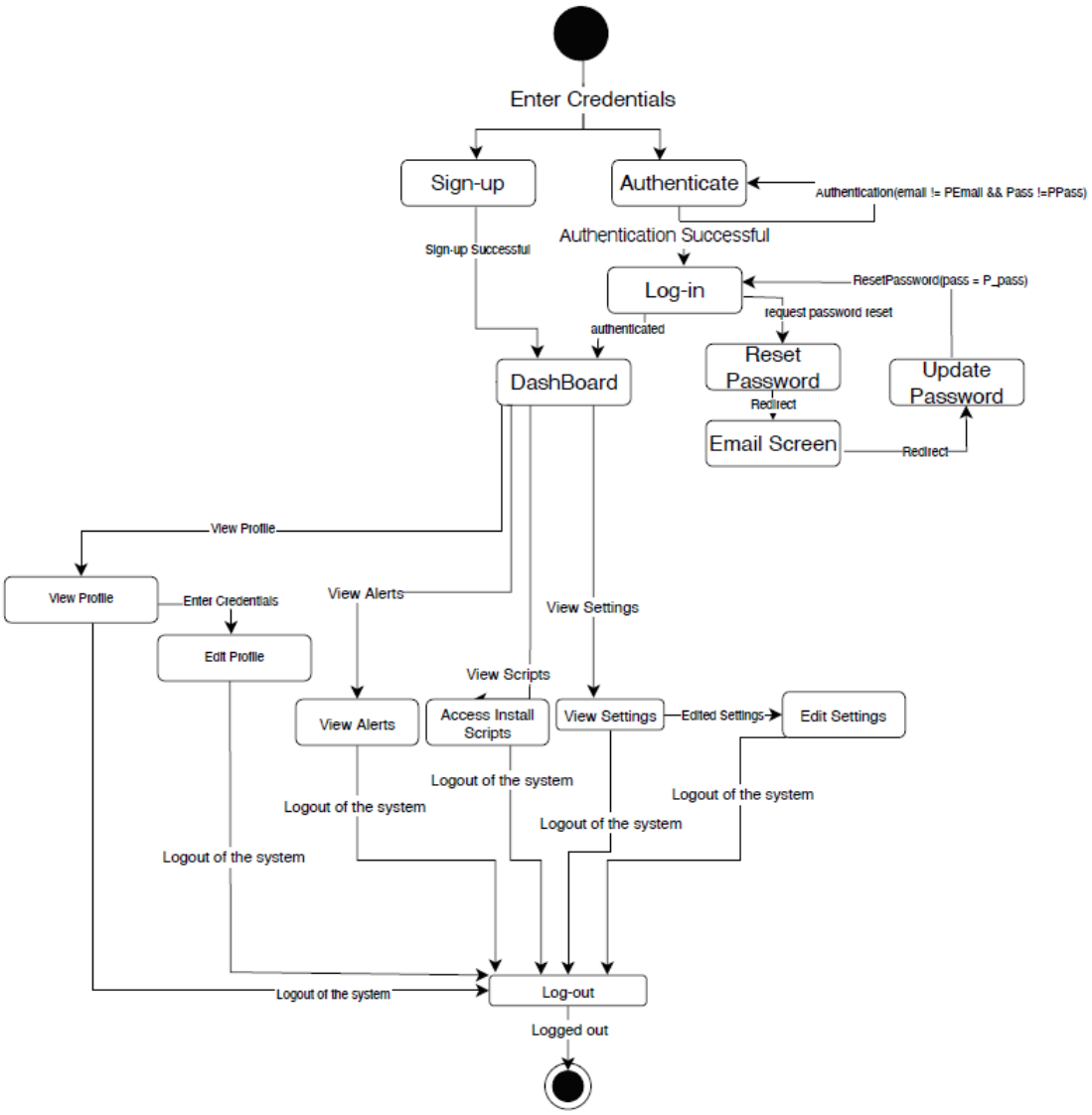

Fig. 7

A finite state machine (FSM) diagram illustrates the behavior of a system through states, transitions, events, and actions. States are represented by circles, indicating different conditions or situations of the system. Transitions, shown as arrows, connect these states and are triggered by events. Each transition may include actions, depicted as labels on the arrows, which occur when the transition is taken. FSM diagrams are used to model the dynamic behavior and state-dependent logic of systems.

5.1.8 Class Diagram

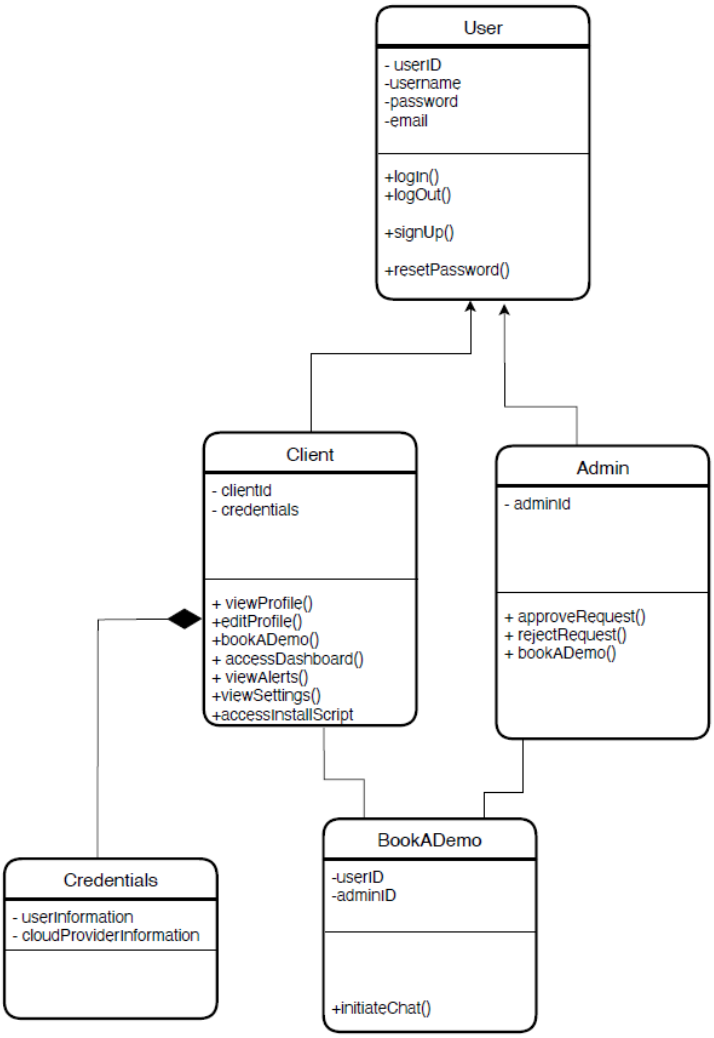

Fig. 8

The class diagram is a type of static structure diagram that describes the structure of our system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.
The purpose of the class diagram is:

- To show a static structure of classifiers in a system
- The diagram provides a basic notation for other structure diagrams prescribed by UML
- It is helpful for developers and other team members too
- Business Analysts can use class diagrams to model systems from a business perspective

CHAPTER 6
PROTOTYPE

6. Website Prototype

Prototype

Please note, website interface prototypes are not actual working pages. This prototype is to showcase the website interface structure and how the website is expected to behave and display data. It should be noted that this prototype is intended for computers, this might slightly affect the design when displayed on other devices and is not indicative of the final product.

Login

Fig. 9

DashBoard

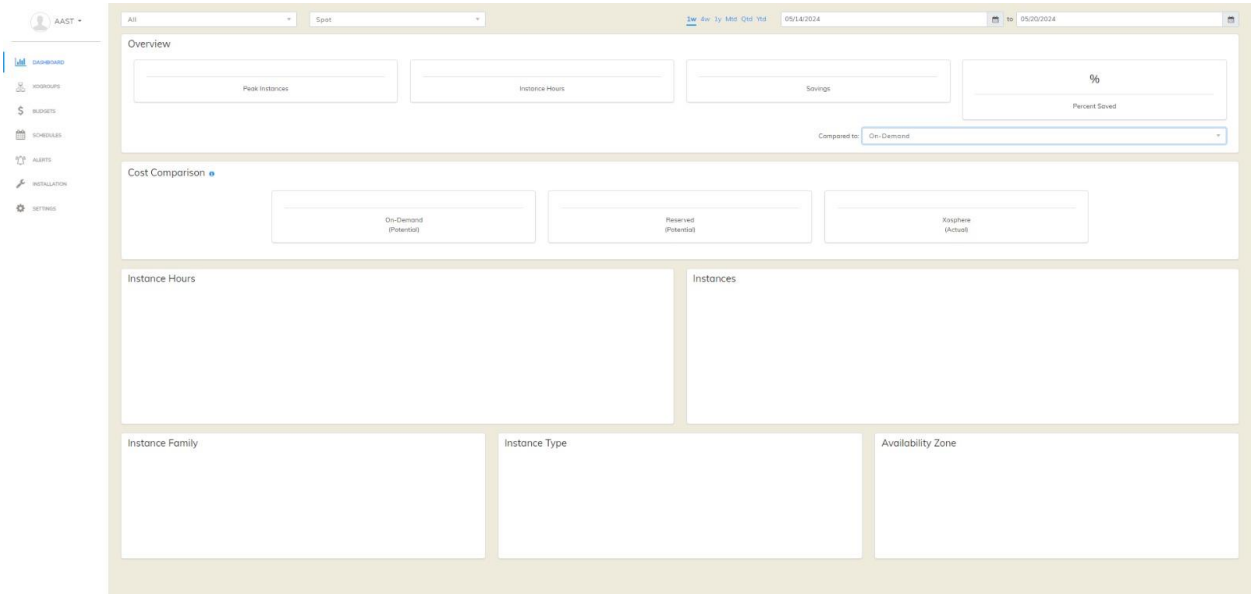

Fig. 10

CHAPTER 7
IMPLEMENTATION AND TESTING

7. Implementation and Testing

7.1 Implementation

The IntelliCloud system integrates advanced AI-driven methodologies to address sustainable resource management challenges in cloud computing environments. The implementation is broken down into the following phases:

7.1.1 System Components and Architecture

The system includes:

- Workload Manager: Handles incoming workloads and converts them into container instances.
- GGCN-based Surrogate Model: Approximates Quality of Service (QoS) metrics using geometric representations of inter-task and resource dependencies.
- Thermal and Energy Models: Integrates cooling, computational, and thermal overheads to optimize energy consumption and avoid thermal hotspots.
- Resource Scheduler: Leverages performance-to-power ratio heuristics to dynamically allocate tasks to optimal nodes.
- Monitoring and Feedback: Continuously gathers resource utilization and fine-tunes the GGCN model for real-time adaptability.

7.1.2 Development Environment

The implementation was carried out using:

- Programming Languages: Python (for AI and backend logic) and DART (for front-end interfaces).
- Frameworks and Tools:
    - TensorFlow for training the GGCN model.
    - Docker for container orchestration.
    - CloudSim and COSCO for simulation and validation.
    - Git for version control and Azure for deployment.
    - PHPmyAdmin for database

7.1.3 Deployment Workflow

1. Initial Configuration:
    - Set up hybrid cloud nodes (e.g., Azure B-series VMs for private and public clouds).
    - Initialize resource monitoring and task management systems.
2. Model Training:
    - Train the GGCN model with initial data from random scheduling decisions.
    - Fine-tune the model using real-time data during execution.
3. Task Allocation:
    - Tasks are allocated based on QoS predictions using the surrogate model.
    - Migration decisions are performed during runtime to address SLA violations and resource hotspots.
4. Continuous Improvement:
    - Adaptive learning updates the model every scheduling interval based on ground-truth QoS metrics.re allocated based on QoS predictions using the surrogate model.

7.1.4 Website Showcase

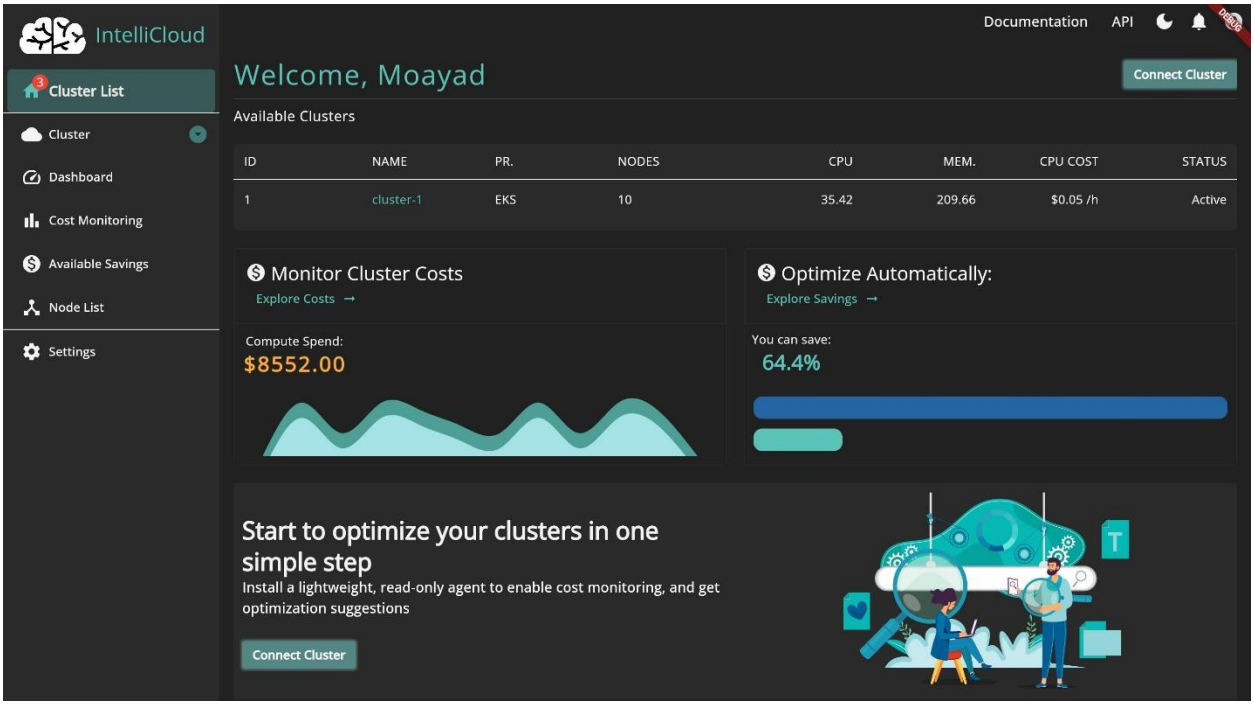

Fig. 11

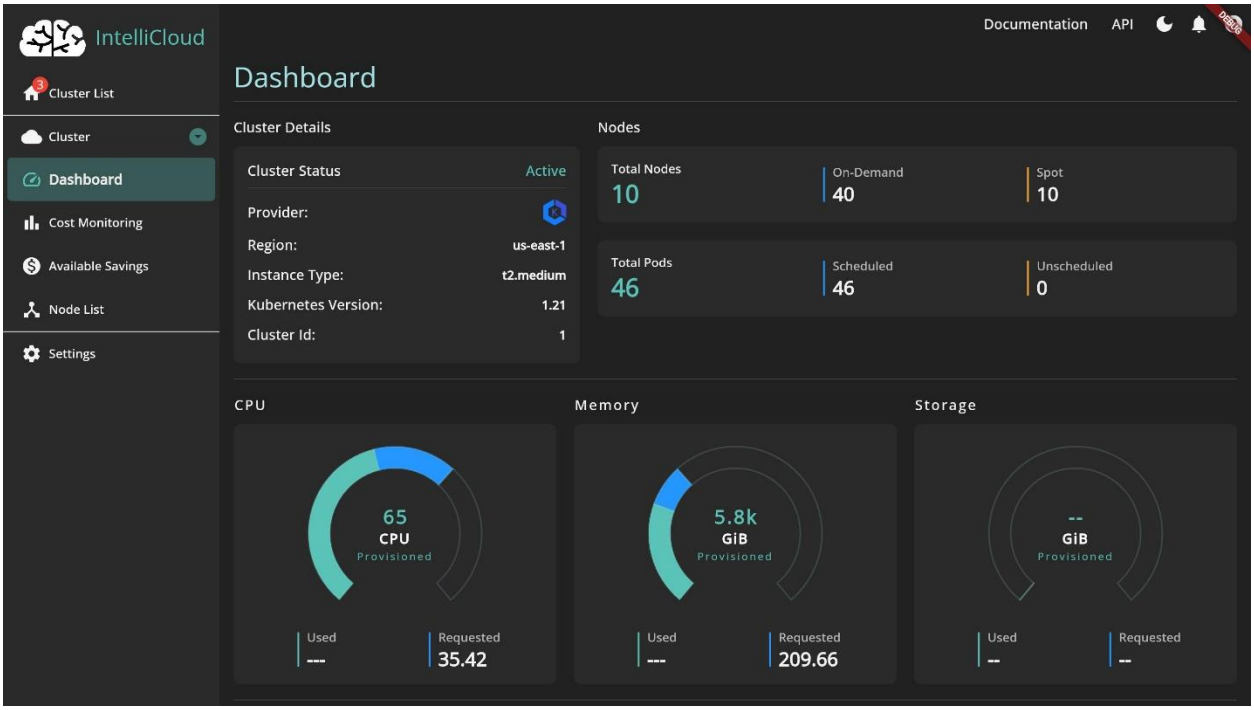

Fig. 12


IntelliCloud
Documentation
API
Cluster List
Cluster
Dashboard
Cost Monitoring
Available Savings
Node List
Settings
CPU Utilization
AVG. Provisioned
2345 CPU/h
AVG. Requested
2343 CPU/h
AVG. Used
223 CPU/h
60
40
20
0
3:00 AM
5:00 AM
7:00 AM
9:00 AM
11:00 AM
1:00 PM
3:00 PM
5:00 PM
7:00 PM
9:00 PM
11:00 PM
1:00 AM
Memory Utilization
AVG. Provisioned
2345 CPU/h
AVG. Requested
1223 CPU/h
AVG. Used
223 CPU/h
Storage Utilization
AVG. Provisioned
2345 CPU/h

Fig. 13

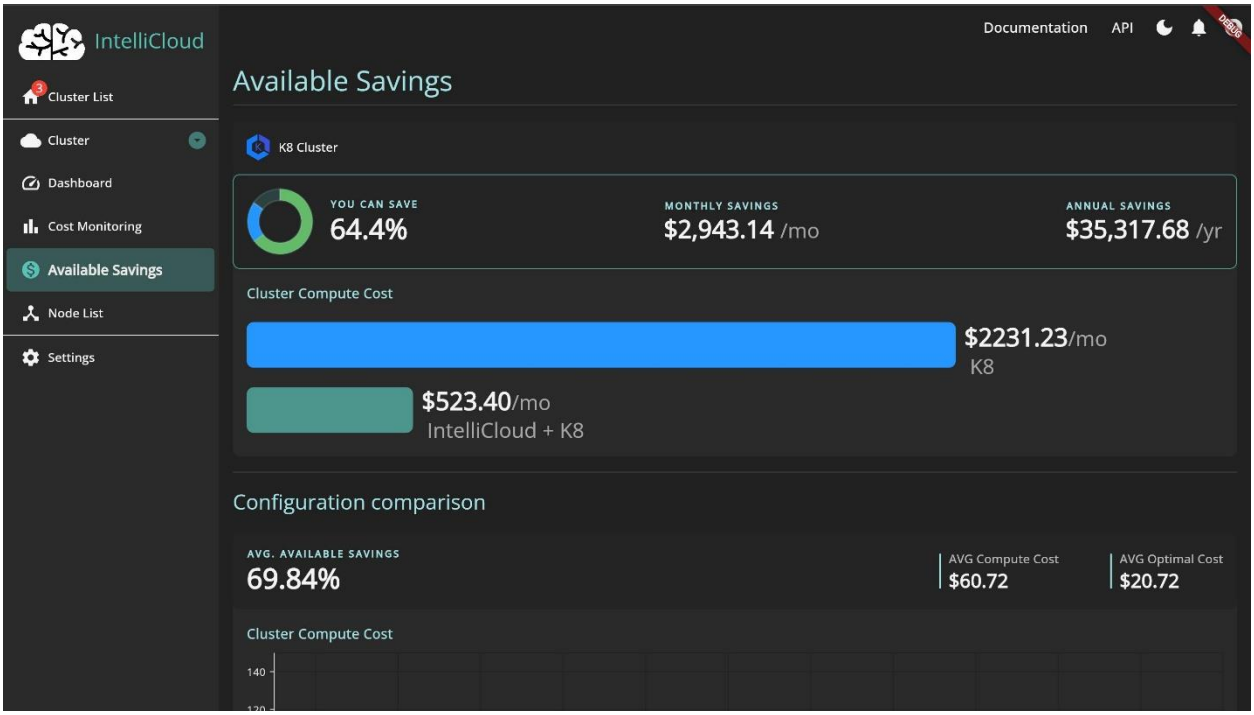
IntelliCloud
Documentation
API
Cluster List
Cluster
Dashboard
Cost Monitoring
Available Savings
Node List
Settings
Available Savings
K8 Cluster
YOU CAN SAVE
64.4%
MONTHLY SAVINGS
$2,943.14 /mo
ANNUAL SAVINGS
$35,317.68 /yr
Cluster Compute Cost
$2231.23/mo
K8
$523.40/mo
IntelliCloud + K8
Configuration comparison
AVG. AVAILABLE SAVINGS
69.84%
AVG Compute Cost
$60.72
AVG Optimal Cost
$20.72
Cluster Compute Cost
140

Fig. 14

7.2 Testing

7.2.1 Testing Methodology

Testing ensured that the IntelliCloud system met performance and sustainability objectives through several stages:

1. Unit Testing:
    - Verified individual modules like the GGCN model, energy model, and thermal model.
    - Example: Ensured that the GGCN surrogate model produced QoS predictions with an MSE below 0.02.

2. Integration Testing:
    - Evaluated interactions between components such as workload management, scheduling, and thermal monitoring.
    - Example: Tested end-to-end flow from workload submission to resource allocation.

3. Performance Testing:
    - Assessed under simulated and real-world workloads.
    - Tools: CloudSim for scalability tests and COSCO for real-time performance validation.

4. Scalability Testing:
    - Conducted with increasing workloads to evaluate how the system handles non-stationary environments.

5. User Acceptance Testing:
    - Engaged administrators and developers to validate usability and system outputs.

7.2.2 Testing Results

The results from the tests demonstrate IntelliCloud's superiority over baseline methods in terms of key performance metrics:

- Energy Efficiency:
    - Achieved up to 11.90% lower energy consumption compared to the best baseline (SDAE-MMQ):
    - Successfully maintained optimal performance-to-power ratios across hosts.
- Temperature Control:
    - Reduced the average temperature by 3.47%, preventing thermal hotspots.
- SLA Compliance:
    - Reduced SLA violation rates by 35.41%, ensuring high service reliability.
- Cost Efficiency:
    - Lowered execution costs by 53.86%, outperforming state-of-the-art methods.
- Scheduling Overheads:
    - Reduced scheduling time by 42.78%, maintaining efficiency in real-time scenarios.

7.2.3 Error Handling and Adaptation

- Bug Tracking:
    - Jira was utilized to log and resolve issues, achieving an average resolution time of 24 hours for critical bugs.
- Adaptive Feedback:
    - Real-time feedback loops ensured continuous improvements to QoS predictions and task placements.

7.2.4 Comparison with Baseline System

IntelliCloud consistently outperformed baseline methods (e.g., PADQN, CRUZE, MITEC) in all test environments, highlighting its robust design for sustainable cloud computing.

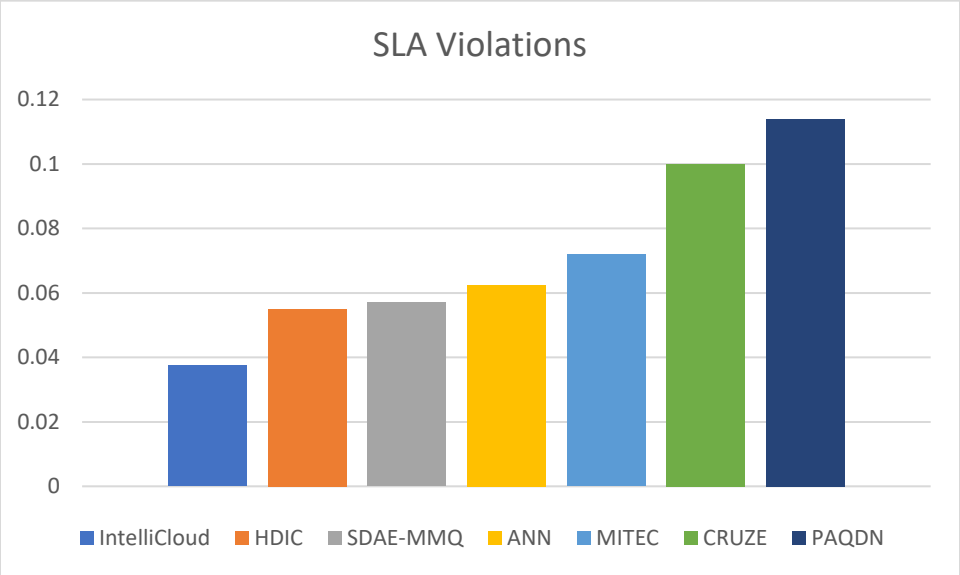

Table 2 (lower is better)

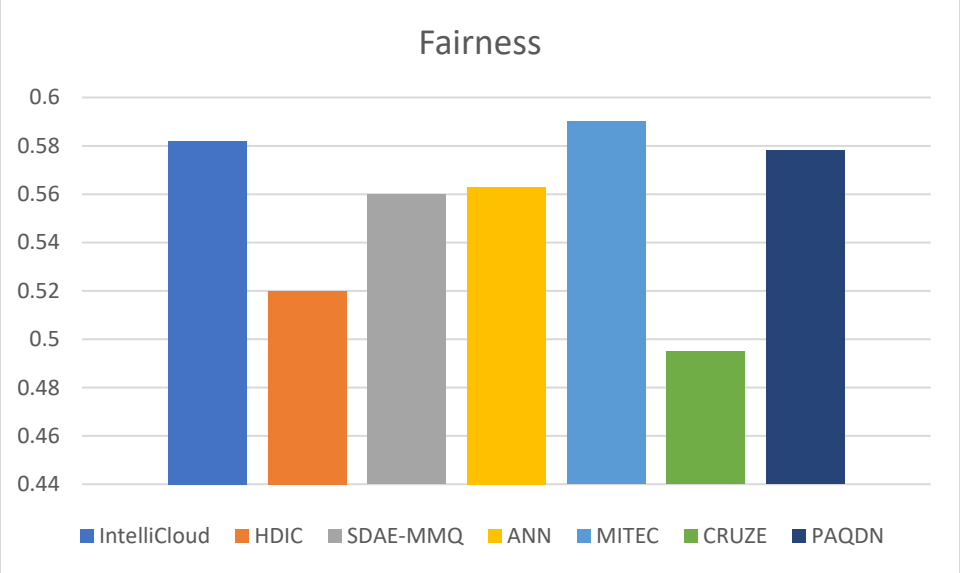

Table 3 (higher is better)

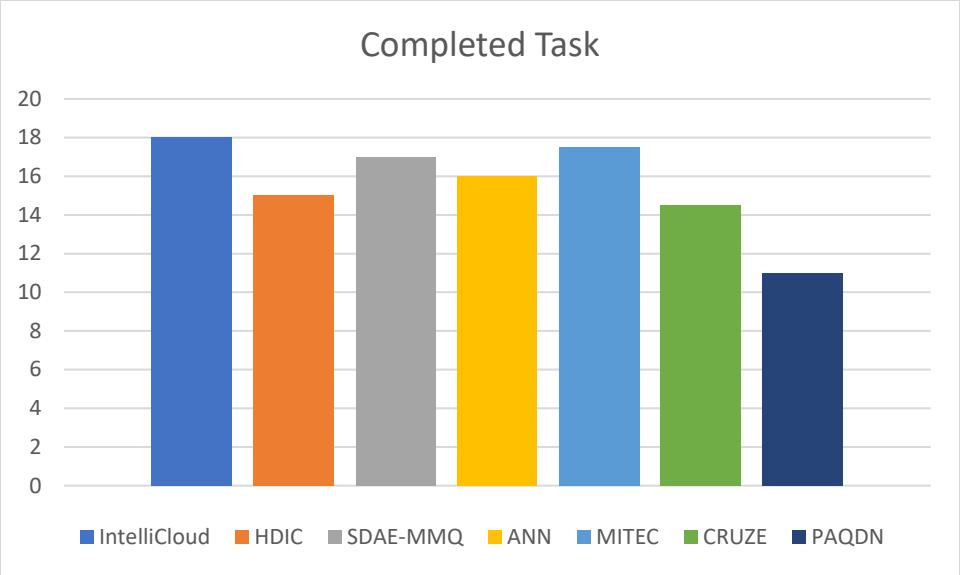

Table 4 (higher is better)

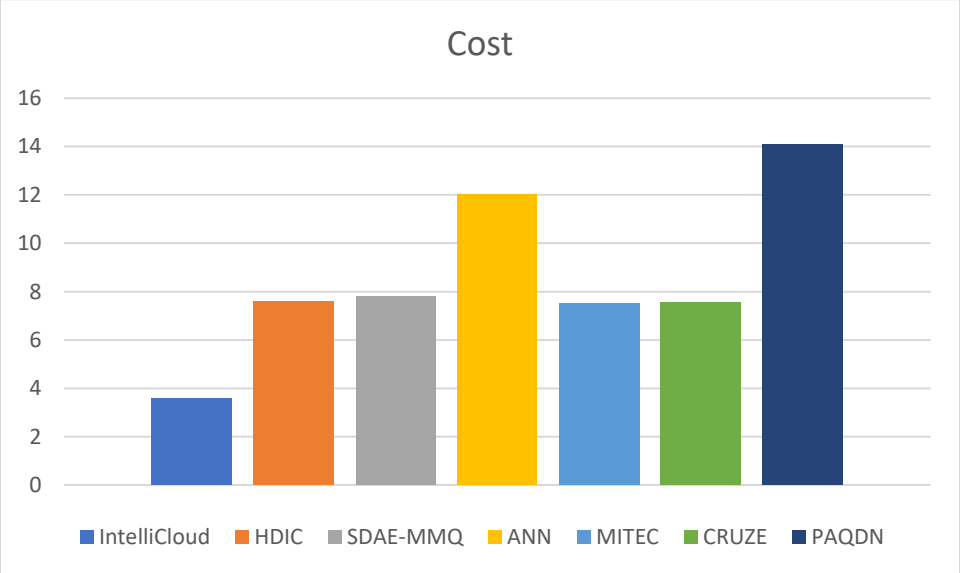

Table 4 (lower is better)

CHAPTER 8
CONCLUSION AND FUTURE WORK

8. Conclusion and Future Work

8.1 Conclusion

In this project, we proposed IntellicCloud, a scheduling approach based on a Gated Graph Convolution Network (GGCN). IntellicCloud enables energy-efficient utilization of cloud servers while reducing thermal hotspots. This is achieved through the integration of cooling-specific energy and temperature models, which are absent in previous approaches.

By leveraging a GGCN-based deep surrogate model, we can rapidly generate Quality of Service (QoS) estimates, thereby avoiding the high costs associated with testing various scheduling decisions. Additionally, IntelliCloud employs a performance-to-power ratio heuristic to balance the workload effectively across cloud hosts, maximizing compute power while minimizing energy consumption. This heuristic also facilitates efficient exploration of the scheduling search space, enabling rapid convergence to optimal decisions.

Extensive testing conducted on both physical and simulated testbeds demonstrate that IntelliCloud outperforms baseline approaches across most QoS metrics. Key performance parameters optimized by IntelliCloud include:

- Temperature: Reduces thermal hotspots through targeted workload allocation.
- Energy consumption: Achieves significant reductions compared to state-of-the-art models.
- Cost: Minimizes operational costs via optimized resource utilization.
- SLA violations: Ensures better compliance with Service Level Agreements.
- Scheduling time: Reduces decision-making overheads, ensuring real-time adaptability.

The low coefficient of variation for energy and temperature metrics indicates IntelliCloud's efficiency and reliability in handling dynamic workloads. Compared to existing AI-based schedulers (e.g., HDIC, SDAE-MMQ, ANN, and PADQN) and heuristic algorithms (e.g., CRUZE and MITEC), HUNTER consistently delivers superior results.

8.2 Future work

IntelliCloud can be extended to address additional parameters such as scalability, security, reliability, and their associated energy impacts. Future research could focus on enhancing cooling management by incorporating domain-specific strategies, particularly in IoT and Fog/Edge computing applications within sectors like agriculture, healthcare, and smart homes.

Currently, IntelliCloud focuses on task placement decisions. However, it can be extended to include air conditioning (AC) or fan settings, especially for environments with deadline-constrained or bursty workloads. Furthermore, the concept of serverless edge computing could be integrated to effectively scale applications.

References

[1] A. Berl, E. Gelenbe, M. Di Girolamo, G. Giuliani, H. De Meer, M. Q. Dang, K. Pentikousis, Energy-efficient cloud computing, The computer journal 53 (7) (2010) 1045–1051.

[2] J. Shuja, A. Gani, S. Shamshirband, R. W. Ahmad, K. Bilal, Sustainable cloud data centers: a survey of enabling techniques and technologies, Renewable and Sustainable Energy Reviews 62 (2016) 195–214.

[3] K. W. Chun, H. Kim, K. Lee, A study on research trends of technologies for Industry 4.0; 3D printing, artificial intelligence, big data, cloud computing, and internet of things, in: Advanced multimedia and ubiquitous engineering, Springer, 2018, pp. 397–403.

[4] P. Wankhede, M. Talati, R. Chinchamalatpure, Comparative study of cloud platforms-Microsoft Azure, Google Cloud Platform and Amazon EC2, Journal of Research in Engineering and Applied Sciences.

[5] E. Pakbaznia, M. Pedram, Minimizing data center cooling and server power costs, in: Proceedings of the 2009 ACM/IEEE international symposium on Low power electronics and design, 2009, pp. 145–150.

[6] S. S. Gill, P. Garraghan, V. Stankovski, G. Casale, R. K. Thulasiram, S. K. Ghosh, K. Ramamohanarao, R. Buyya, Holistic resource management for sustainable and reliable cloud computing: An innovative solution to global challenge, Journal of Systems and Software 155 (2019) 104–129.

[7] M. T. Chaudhry, T. C. Ling, A. Manzoor, S. A. Hussain, J. Kim, Thermal-aware scheduling in green data centers, ACM Computing Surveys (CSUR) 47 (3) (2015) 1–48.

[8] A. Akbari, A. Khonsari, S. M. Ghoreyshi, Thermal-aware virtual machine allocation for heterogeneous cloud data centers, Energies 13 (11) (2020) 2880.

[9] S. MirhoseiniNejad, G. Badawy, D. G. Down, Holistic thermal-aware workload management and infrastructure control for heterogeneous data centers using machine learning, Future Generation Computer Systems 118 (2021) 208–218.

[10] S. Tuli, S. Poojara, S. N. Srirama, G.Casale and N. R. Jennings "COSCO: Container Orchestration using Co-Simulation and Gradient Based Optimization for Fog Computing Environments", accepted in IEEE Trasnsactions on Parallel and Distributed Systems (TPDS), 2021.