Optimizing Task Allocation for Edge Compute Micro-Clusters

Yousef Alhaizaey

Submitted in fulfilment of the requirements for the degree of
Doctor of Philosophy

School of Computing Science

College of Science and Engineering
University of Glasgow

July 2023

Abstract

There are over 30 billion devices at the network edge. This is largely driven by the unprecedented growth of the Internet-of-Things (IoT) and 5G technologies. These devices are being used in various applications and technologies, including but not limited to smart city systems, innovative agriculture management systems, and intelligent home systems. Deployment issues like networking and privacy problems dictate that computing should occur close to the data source at or near the network edge.

Edge and fog computing are recent decentralised computing paradigms proposed to augment cloud services by extending computing and storage capabilities to the network's edge to enable executing computational workloads locally. The benefits can help to solve issues such as reducing the strain on networking backhaul, improving network latency and enhancing application responsiveness. Many edge and fog computing deployment solutions and infrastructures are being employed to deliver cloud resources and services at the edge of the network — for example, cloudless and mobile edge computing.

This thesis focuses on edge micro-cluster platforms for edge computing. Edge computing micro-cluster platforms are small, compact, and decentralised groups of interconnected computing resources located close to the edge of a network. These micro-clusters can typically comprise a variety of heterogeneous but resource-constrained computing resources, such as small compute nodes like Single Board Computers (SBCs), storage devices, and networking equipment deployed in local area networks such as smart home management. The goal of edge computing micro-clusters is to bring computation and data storage closer to IoT devices and sensors to improve the performance and reliability of distributed systems. Resource management and workload allocation represent a substantial challenge for such resource-limited and heterogeneous micro-clusters because of diversity in system architecture. Therefore, task allocation and workload management are complex problems in such micro-clusters.

This thesis investigates the feasibility of edge micro-cluster platforms for edge computation. Specifically, the thesis examines the performance of micro-clusters to execute IoT applications. Furthermore, the thesis involves the evaluation of various optimisation techniques for task allocation and workload management in edge compute micro-cluster platforms. This thesis involves the application of various optimisation techniques, including simple heuristics-based optimisations, mathematical-based optimisation and metaheuristic optimisation techniques, to optimise task allocation problems in reconfigurable edge computing micro-clusters.

The implementation and performance evaluations take place in a configured edge realistic environment using a constructed micro-cluster system comprised of a group of heterogeneous computing nodes and utilising a set of edge-relevant applications benchmark. The research overall characterises and demonstrates a feasible use case for micro-cluster platforms for edge computing environments and provides insight into the performance of various task allocation optimisation techniques for such micro-cluster systems.

Acknowledgements

The first and foremost acknowledgement goes to my PhD supervisor, Dr. Jeremy Singer. I would like to express my deepest gratitude for him for the invaluable guidance, unlimited support, and continuous encouragement I received throughout my research journey. The constant weekly meetings and discussions, in person and virtually over Zoom during the Covid-19 pandemic, have helped me to learn research and move forward. I am very grateful for the opportunity to study under his supervision and for believing in me to complete my PhD study. This work would not have been possible without his unlimited support and constant encouragement during tough times.

I would also like to extend my appreciation to my second supervisor, Dr. Anna Lito Michala, for her feedback, input, and suggestions on my research. The input and feedback are appreciated.

My deepest thanks go to my wife, Halimah, and my daughters, Danah and Wed, for their unaccountable support and patience and for accompanying me to the United Kingdom during my PhD study. I am forever grateful for your unwavering support and love.

I would also like to extend my grateful thanks to my family in Saudi Arabia, to my parents for their patience, to my sisters and my brothers for their support. A particular thank goes to my brother, Abdulrahim Alhaizaey, for the friendship and for keeping me motivated.

I would also like to thank my friends and officemates at the University of Glasgow. Without any particular order, I would like to thank Adrian Ramsingh, Dejice Jacob, Haruna Umar Adoga, Ibrahim Alghamdi, Kyle Simpson, Saad Alahmari, and Yosuef Alotaibi for their friendship, support, and feedback throughout this journey. All kinds of your support are appreciated.

Thanks also to Dr. Jose Cano for reviewing my work during the annual progress reviews, Dr. Blesson Varghese for the valuable suggestions and recommendation, and the anonymous reviewers for feedback on all published papers.

I acknowledge King Khalid University in Saudi Arabia and the Saudi Arabian Cultural Bureau in the UK for providing the financial and logistical support that made this research possible. I am grateful for their support and for the opportunities they have provided me. I also acknowledge the University of Glasgow's School of Computing Science for accommodating me and providing me with the research equipment and support during my research journey.

Finally, I acknowledge and extend my appreciation to my PhD examiners, Professor Huaglory Tianfield [Glasgow Caledonian University] and Dr. Lauritz Thamsen [University of Glasgow], for their time, effort, and invaluable feedback in evaluating my research work. Your constructive feedback and insightful comments have undoubtedly enriched the quality of my thesis. I would not forget to thank my PhD viva convener, Dr. Wagar Nabi [University of Glasgow], for the exceptional organisation and guidance before, throughout, and after the viva examination.

Thank you all for your invaluable contributions to this work.

Dedication

This work is dedicated to my wife Halimah Al Hiza and my daughters Danah & Wed.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

Edge computing is a distributed computing paradigm recently proposed to extend cloud capabilities to the edge of the network [1, 2, 3]. The underlying objective of edge computing is to deliver cloud services to a variety of Internet of Things (IoT) applications by utilising computation and storage resources distributed across the edge of the network to process workloads locally at or near the data sources. Edge computing demonstrates a compelling computing paradigm that overcomes limitations related to latency, networking bandwidth and jitters, energy consumption and data privacy [4].

Several edge-based architectures, systems, and solutions have been proposed to facilitate the deployment of edge computing [5]. For example, Cloudlet data centres represent a core example of edge computation, whereby resource-rich computers or clusters are deployed in various locations to provide computation and storage services for nearby mobile devices [1]. Furthermore, Content Delivery Networks (CDNs) are another form of edge computation proposed to distribute and cache content in different locations in order to minimise the response time of applications [3]. These edge-based solutions are generally equipped with sufficient computational resources to deliver cloud services at the edge of the network near data sources.

In this thesis, the focus is on edge micro-cluster platforms. Edge micro-clusters represent pragmatic components of edge computing infrastructure. Such micro-clusters demonstrate practical solutions that can enable edge computation for various IoT applications and use cases [6]. For example, such micro-clusters can be used in Road Side Units (RSUs) to deliver computation services to vehicles on roads [7]. They also provide practical solutions for smart cities applications to provide citizen-facing services such as smart parking management systems [8]. Furthermore, edge micro-clusters provide useful testbeds for research and

education in universities that can enable students and researchers to experiment in a physical edge computing environment [6, 9].

Edge micro-clusters may consist of multiple heterogeneous compute nodes with different hardware capabilities. These relate to the hardware specifications such as Central Processing Unit (CPU) speed and the Random Access Memory (RAM) size. For example, micro-clusters may include resource-limited nodes, such as Single Board Computers (SBCs), e.g., Raspberry Pi devices, Odroid devices, Google Coral Boards, or more powerful compute devices with server capabilities such as Mini-ITX. Edge micro-clusters can provide energy-efficient systems that can be powered using battery packs or renewable energy. Furthermore, edge micro-cluster systems represent compact solutions as they require a small physical footprint, that can enable such micro-clusters to be easily portable to deliver computation services to new locations where conventional computing solutions such as traditional servers would be challenging [7]. Because of differences in infrastructure, architecture, resource connectivity and capacity, and workload characteristics, task allocation and optimisation need to be reexamined for such edge computing systems [10].

Edge micro-cluster platforms will likely receive and process workloads in batch-arrival execution mode. The batch execution mode requires the micro-cluster systems to receive heterogeneous workloads from different sources and collect workloads in batches based on workload type or arrival time [11]. Batch execution consumes high resources and requires extensive parallel computations [12]. Therefore, batch execution for edge micro-clusters mandates effective and efficient task allocation optimisation techniques to meet the Quality of Service QoS requirements of the applications. For example, in a smart farming use case, IoT devices such as sensors, actuators, and drones generate and gather farming-related data, e.g., scan plants, capture images, or monitor animal movements. Those IoT devices are typically not equipped with sufficient computing resources, and they subsequently may require to offload workloads to nearby edge micro-clusters for processing and storage [13]. When utilising such edge micro-clusters, effective and efficient task allocation optimisation techniques are required to meet the technical specifications for such compact and resource constraint clusters.

1.2 Motivation for Micro-Clusters in Edge Computing

This section outlines the concept of edge compute micro-clusters and advocates that this is a compelling demonstrator for a typical compact edge computing platform capable of delivering and extending computing services to the edge of the network. In addition, it discusses workload characteristics for such micro-cluster platforms in further detail.

1.2.1 Edge Micro-Clusters

Edge micro-clusters are *heterogeneous* compact platforms. They will consist of multiple computing devices that have different computing capabilities. This might be in terms of CPU core count and clock speed or in terms of available RAM. Further, some devices might feature specialized accelerator nodes, such as the Movidius Neural Compute Stick. Since a micro-cluster is likely to be deployed for a long time, it is possible that upgraded nodes with higher specifications may be added at a later date, or some nodes might need to be replaced. This increases heterogeneity in micro-clusters.

A micro-cluster is located at the network *edge*. For this reason, it is end-user facing, deployed at one hop from data sources. This allows micro-clusters to provide computing resources and deliver content directly to consumers. Furthermore, this addresses the expectation of minimal latency. In terms of connectivity, nodes in the cluster will have a regular, non-exotic network topology, typically peers on a single Local Area Network (LAN).

A micro-cluster is *micro-scale*. This has implications for the power draw and energy consumption. Micro-cluster platforms will be likely to run off renewable energy and a battery pack. In terms of cost, electronic equipment must be commodity, cheap components since they are so widely deployed. Further, in terms of physical footprint the micro-cluster should be small enough to be compact and portable, perhaps occupying a road side unit cabinet or embedded in the fabric of a building.

A micro-cluster platform is a *compute cluster*, consisting of a small number of nodes (perhaps up to 10) with a small number of total cores (perhaps up to 100). The multiplicity of nodes is important for (a) redundancy, given the commodity nature of the devices; and (b) parallel throughput, given the nature of the workloads.

1.2.2 Workloads for Micro-Clusters

There is a wide range of use cases for edge computing. Generally, such applications are presented as high-level abstract scenarios including, but not limited to, self-driving cars, augmented reality tourism, and smart healthcare [2, 14, 15]. With respect to understanding concrete compute resource requirements, it is more practical to consider specific usage scenarios at a task processing level.

This research assumes that micro-clusters are likely to receive a set of related compute requests as a batch-of-jobs, with a low-latency requirement for completion of all jobs in the batch. This might be object detection in a set of still photo images or audio processing tasks.

In general, research literature in the field of edge computing describes high-level application use cases. Popular examples include, but are not limited to, surveillance cameras in smart

cities [16], smart home sensors [17], support for autonomous vehicles [18], and voice-based smart services [19, 20].

This trend of heterogeneity at the edge has been identified by the edge research community [11]. This research, therefore, aims to tackle the lack of a real testbed for micro-cluster platforms and characterise the deployment features and utility by modelling and experimenting with a physical configured testbed instead of using simulation solutions, as presented in this dissertation.

1.3 Research Focus

This thesis characterises and advocates the feasibility of heterogeneous edge micro-cluster platforms for enabling practical and effective edge computation for several IoT use cases. In addition, this research investigates and evaluates the performance of different optimisation techniques for batch execution optimisation in edge micro-cluster systems. The evaluation demonstrates a clear overlap in the performance of the metaheuristic Particle Swarm Optimisation (PSO) and the mathematical-based optimisation techniques for optimising batch execution, which indicates the efficiency of optimisation techniques for edge deployment. In particular, the metaheuristics-based task allocation optimisation is a viable and effective technique for optimising and executing large batches for large-scale clusters, while the mathematical-based optimisation technique is effective and efficient for small-scale edge micro-clusters. Effectiveness and efficiency relate to the performance metrics used to evaluate the applicability of the optimisation techniques for edge micro-clusters. Effectiveness refers to the ability of the techniques to provide optimal or near-optimal solutions for the optimisation problems, whereas efficiency refers to the scalability and overhead time of the optimisation techniques. Furthermore, this research evaluates the developed task allocation optimisation solutions in a physical edge environment using a realistic edge cluster testbed and representative real-world benchmark application software. The work further helps to characterise the performance of edge micro-cluster platforms and reflects on the performance of the optimisation technique for optimising batch execution for such compact edge computing platforms.

1.4 Thesis Statement

Edge micro-cluster platforms are a key enabler for practical edge computing, since they represent pragmatic instantiations of computing resources at the network edge. A fundamental problem in utilising micro-cluster systems for executing edge-related workloads is optimising task allocation for batch execution. This thesis argues that edge micro-cluster platforms require efficient and effective task allocation optimisation techniques that satisfy the resource-constraints of micro-clusters, dynamic workloads at the edge, and strict QoS requirements of IoT-based applications. In particular, this could be achieved by modifying the task allocation optimization techniques for the resource-constrained environment of edge computing micro-clusters.

1.5 Research Questions

This thesis aims to answer the following research questions related to optimising task allocation for edge compute micro-clusters.

RQ1. Which task allocation optimisation methods provide effective solutions to optimise makespan time for edge micro-cluster platforms for heterogeneous workloads in batch-mode?

RQ2. Which optimisation methods are efficient and appropriate (i.e., sufficiently lightweight) for edge micro-cluster platforms?

RQ3. Which task allocation optimisation methods provide effective solutions to optimise both makespan time and energy consumption (i.e., multi objective optimisation) for edge micro-clusters?

RQ4. Is it appropriate to use commodity Single Board Computers (SBCs) to model edge micro-clusters to empirically evaluate task allocation techniques for batch-mode execution?

RQ5. What kind of edge workloads provide real-world representative benchmarks for evaluating task allocation methods for batch-mode execution on micro-cluster platforms?

The research questions of this thesis are consistent with the research framework published in a recent systematic literature review [21]. The authors reviewed 138 papers on fog and edge computing and the related optimisation problems. Furthermore, they developed a framework capturing the trends in edge research community.

Figure 1.1 represents a visual mapping linking the thesis's research questions and framework's research questions. With respect to the framework's RQ1, this thesis examines and compares the performance of various optimisation techniques for optimising task allocation in edge micro-cluster platforms. Regarding the framework's RQ2, this thesis aims to enhance edge micro-clusters deployment in edge networks and optimise task allocation for such resource-constraints systems. This requires examining the performance characteristics of micro-clusters and evaluating the optimisation techniques accordingly. Furthermore, concerning the framework's RQ3, three edge-relevant performance metrics were targeted: the makespan time metric, the allocation overhead time of optimisation techniques, and the energy consumption. These performance metrics are significant for edge micro-cluster platforms. Finally, regarding the framework's RQ4, instead of using simulation-based tools, this thesis aims to empirically evaluate the performance of different optimisation techniques on a configured and realistic micro-cluster platform comprised of several single broad computers and representative applications benchmarks suite.

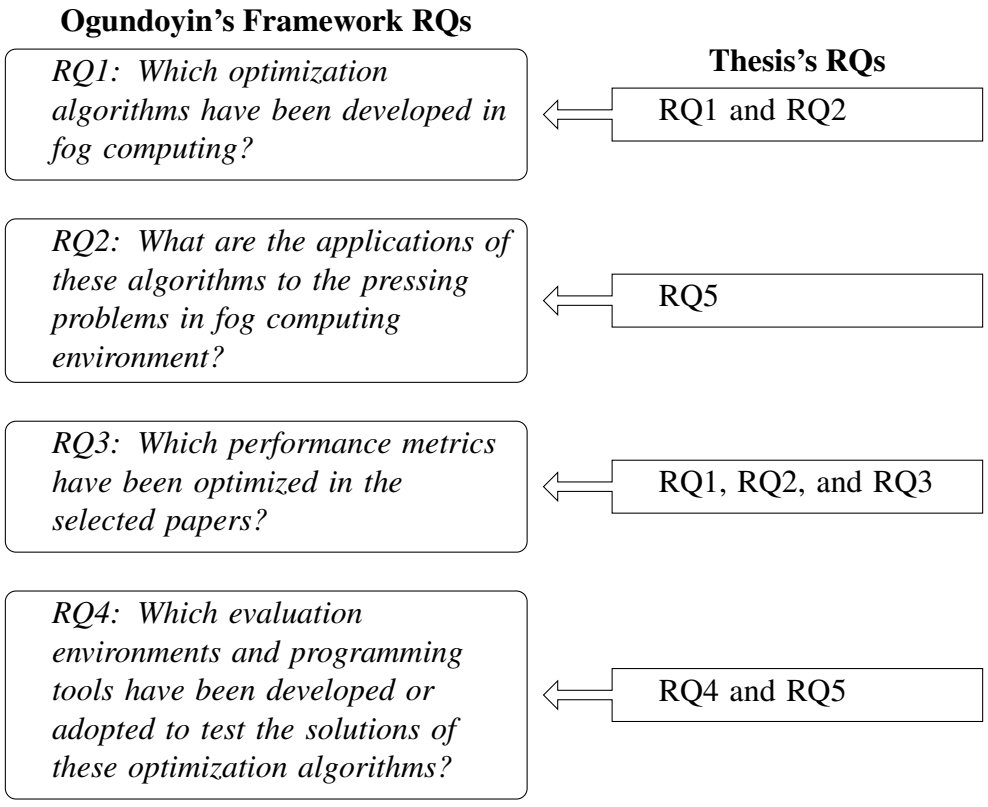


Figure 1.1: Mapping the Thesis's RQs with Ogundoyin's Framework RQs

1.6 Contributions

The research presented in this thesis makes the following key contributions to the field of edge computation:

1. The characterisation and the motivation of heterogeneous edge micro-cluster platforms for facilitating practical edge computation. This thesis describes the underlying hardware specifications and characteristics of edge micro-cluster systems and demonstrates typical edge-related workloads for edge computing platforms.

2. The utilisation and experimentation with a physical edge micro-cluster testbed. Rather than relying on simulation-based software solutions for reporting performance analysis of the proposed optimisation techniques, the experiments and the evaluations are conducted using a realistic configured edge micro-cluster testbed. This approach requires building a physical and representative edge micro-cluster testbed, using representative benchmark applications software and wall-clock timing metrics for performance evaluation.

3. The design and evaluation of analytical performance models for predicting makespan time and energy consumption required by edge micro-cluster platforms for executing heterogeneous edge-relevant applications in batch-execution mode.

4. The design and evaluation of a multi-objective optimisation model for edge micro-cluster platforms. The model can provide a Pareto-optimal set of feasible solutions representing the trades-off system's competing performance.

5. The provision of the performance of various optimisation techniques for optimising task allocation in edge micro-cluster platforms. The research demonstrates the related effectiveness and efficiency of different optimisation techniques for orchestrating and optimising workload allocation in edge micro-clusters.

These contributions help to enhance and extend the feasibility and usability of edge micro-cluster platforms to be more efficient and practical deployment infrastructures for edge computing. The aim is to push those edge micro-cluster platforms beyond their basic applications, use cases, and deployment scenarios by integrating efficient and effective optimisation techniques for such compact platforms. The experimental work and the analysis were conducted in a realistic heterogeneous edge micro-cluster testbed built utilising several heterogeneous Raspberry Pi devices drawn from different generations and using a representative benchmark software suite. The results and the findings from this research could be generalised to other comparable micro-cluster systems that deploy similar or equivalent resource-constrained devices. The source code and materials related to this work can be

accessed in the GitHub repository available at `https://github.com/yalhaizaey/edge-micro-clusters`.

1.7 Publications

The work described in this thesis has already been published in the following peer-reviewed articles. Meanwhile, the plan is to publish the outcomes from Chapter 5 in edge and fog computing-relevant venues in the near future.

- Y. Alhaizaey, J. Singer and A. L. Michala. Optimizing Task Allocation for Edge Micro-Clusters in Smart Cities. In 2021 IEEE 22nd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), Pisa, Italy, 2021, pp. 341-347, doi: 10.1109/WoWMoM51794.2021.00062 [10].
- Y. Alhaizaey, J. Singer and A. L. Michala. Optimizing Heterogeneous Task Allocation for Edge Compute Micro Clusters Using PSO Metaheuristic. In 2022 Seventh International Conference on Fog and Mobile Edge Computing (FMEC), Paris, France, 2022, pp. 1-8, doi: 10.1109/FMEC57183.2022.10062755 [22].

1.8 Thesis Outline

This section provides an outline of the thesis chapters. The chapters of the thesis and their main sections are depicted in Figure 1.2. The thesis overall is structured as follows:

- **Chapter 1 – Introduction:** This chapter presents an overview of the research by providing the research scope, presenting the research focus and thesis statement, and defining the research questions. In addition, it highlights the work contributions and outlines the overall structure of the thesis.
- **Chapter 2 – Literature Review:** This chapter reviews relevant literature in the fields related to edge, fog, and cloud computing technologies. It overviews the landscape of cloud computing and the need for decentralising the technologies toward the edge of the network. A particular emphasis is placed on exploring work related to edge micro-cluster platforms and the related resource management problems and optimisation techniques.
- **Chapter 3 – Experimental Evaluation of Feasibility and Task Allocation of Edge Micro-Clusters:** This chapter provides a comprehensive overview of edge compute

micro-cluster platforms and their potential deployment use cases. The chapter discusses edge micro-clusters and underneath technologies related to hardware specifications, configuration requirements, virtualisation technologies, and construction techniques. It develops and evaluates the performance of a set of heuristic-based and mathematical-based task allocation techniques for optimising the makespan time of batch execution mode. The chapter also highlights the need to empirically evaluate experiments using cluster testbed and practical benchmarking tools. Chapter 3 provides answers to RQ1, RQ4, and RQ5 and presents research contributions 1, 2, and 5.

- **Chapter 4 – A Linear Model for Task Allocation in Micro-Clusters:** This chapter extends Chapter 3 by presenting a linear-based system model designed to predict the makespan time of batch execution for edge micro-cluster platforms. It additionally extends the optimisation techniques implemented in the previous chapter by examining the efficiency and effectiveness of the metaheuristics PSO optimisation technique. Chapter 4 provides answers to RQ1 and RQ2 and presents research contributions 2, 3 and 5.

- **Chapter 5 – Multi-Objective Optimisation for Edge Micro-Clusters:** This chapter extends Chapters 3 and 4 by evaluating and optimising energy consumption performance metric in edge micro-cluster platforms. It proposes and evaluates an analytical energy consumption model that provide energy prediction for micro-clusters. Furthermore, the chapter designs and evaluates a multi-objective optimisation model for optimising multi-objective optimisation problems in edge micro-clusters. Chapter 5 provides answers to RQ3 and presents research contributions 3 and 4.

- **Chapter 6 – Conclusions and Future Directions:** This chapter concludes the research. It reviews the thesis statement in light of the conducted work. It provides reflections upon the research questions and highlights the research contributions. The chapter further discusses the limitations of the work and provides recommendations for potential directions and future work.

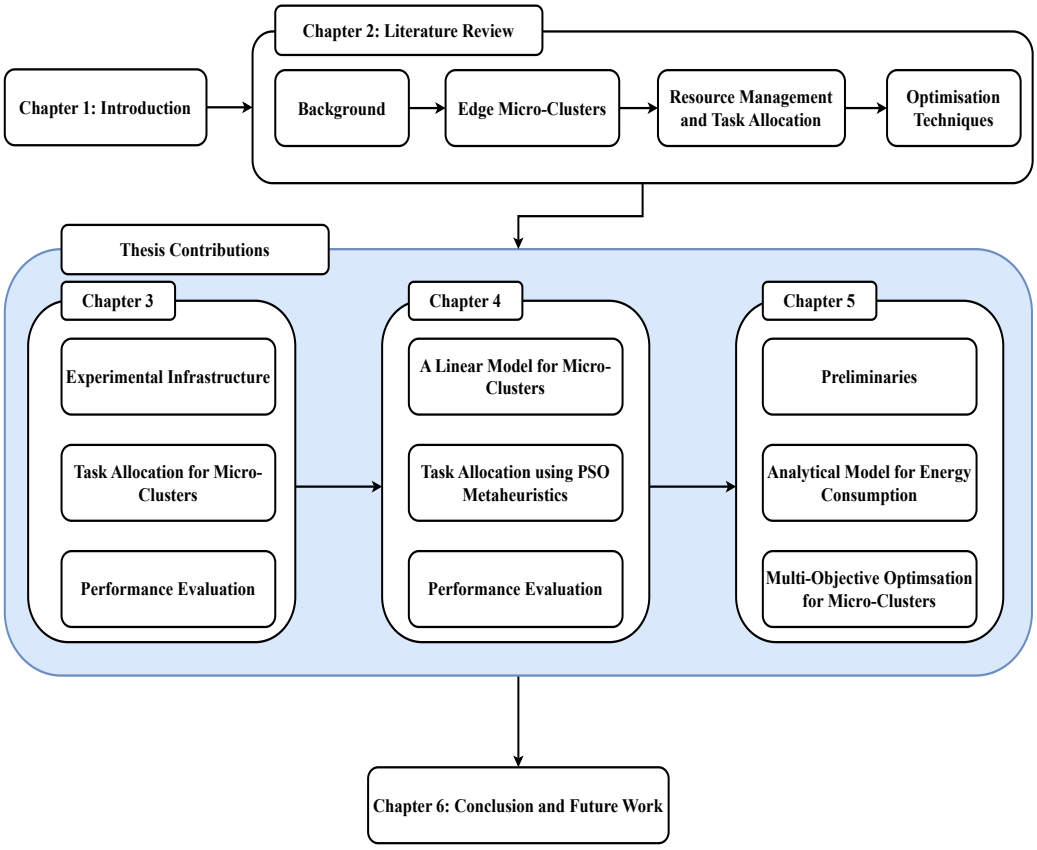


Figure 1.2: Diagram illustrating the structure of the thesis

Chapter 2

Literature Review

Nowadays, with the presence of the Internet-of-Things (IoT) and smart environments, such as smart cities and smart industries, there is a coexistence of various emerging computing technologies and paradigms that underpin the computing landscape. This chapter presents an overview of the computing landscape, defines the background of different edge computing-related technologies, and discusses work specifically related to edge micro-cluster platforms, task allocation and workload management, and the related optimisation techniques.

2.1 Background

The Internet-of-Things (IoT) revolution plays an essential role in changing the Internet from its conventional paradigm, where the client-server architecture is deployed, to becoming a heterogeneous network that enables everything to connect, communicate, generate data, and share resources. This new revolution sparks a massive surge of communication and data at the edge of the network as the number of connected devices is expected to be 50 billion by 2020 [17, 23] and reach over 80 billion by 2025 [3].

There are several interconnected technologies behind the development of IoT. Cloud computing is considered a key enabler or catalyst of this revolution. Cloud efficiently provides on-demand computational and storage resources for IoT applications and users. Cloud requires offloading data to centralised data centres located far from data sources for processing and storage purposes. However, this would not always be an efficient choice for IoT applications when billions of devices and sensors are connected and request services simultaneously. Significant concerns are related to latency, power consumption, and data privacy, which are already becoming apparent to end users [2, 3, 5, 24].

Considering the cloud deployment concerns mentioned above, extending cloud resources to the edge of the network becomes a recent practicable solution, introducing new emerging

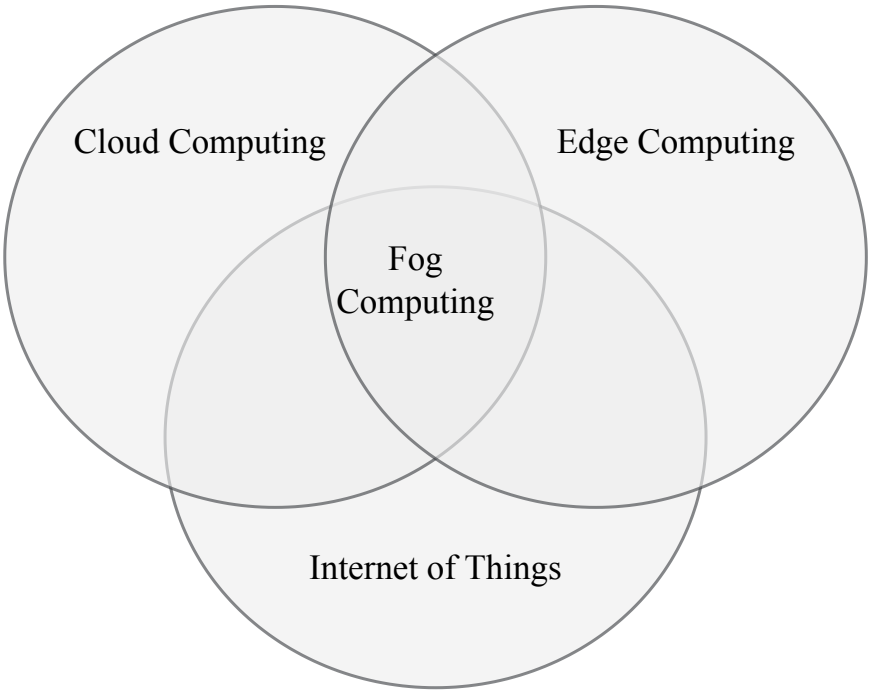


Figure 2.1: Venn diagram visualises the computing landscape.

distributed paradigms known as fog computing or edge computing. The overall objective is to leverage resources at the edge of the network to execute applications locally near or at data sources instead of offloading computational logic to cloud data centres [24, 25, 26, 27].

Overall, the computing model has become more complex and diverse, with many interconnected technologies and computing paradigms. This chapter will first discuss recent emerging technologies capturing the trend in decentralising the computing landscape before delving into discussing the work-related edge micro-cluster platforms, resource management and relevant optimisation. Figure 2.1 depicts the overall picture of the current computing models.

The remainder of this chapter is structured as follows. Sections 2.2, 2.3, and 2.4 provide high-level overviews edge-related computing paradigms. Section 2.5 discusses micro-cluster systems and the related deployment contexts in detail. Section 2.6 studies resource management in edge computing with a particular emphasis on task allocation in Section 2.6.2. Section 2.7 discusses the optimisation techniques employed in solving orchestration and management-related problems in edge computing. Section 2.8 and 2.9 provide discussion against the related work and the conclusion. Figure. 2.2 presents the outline of this chapter.

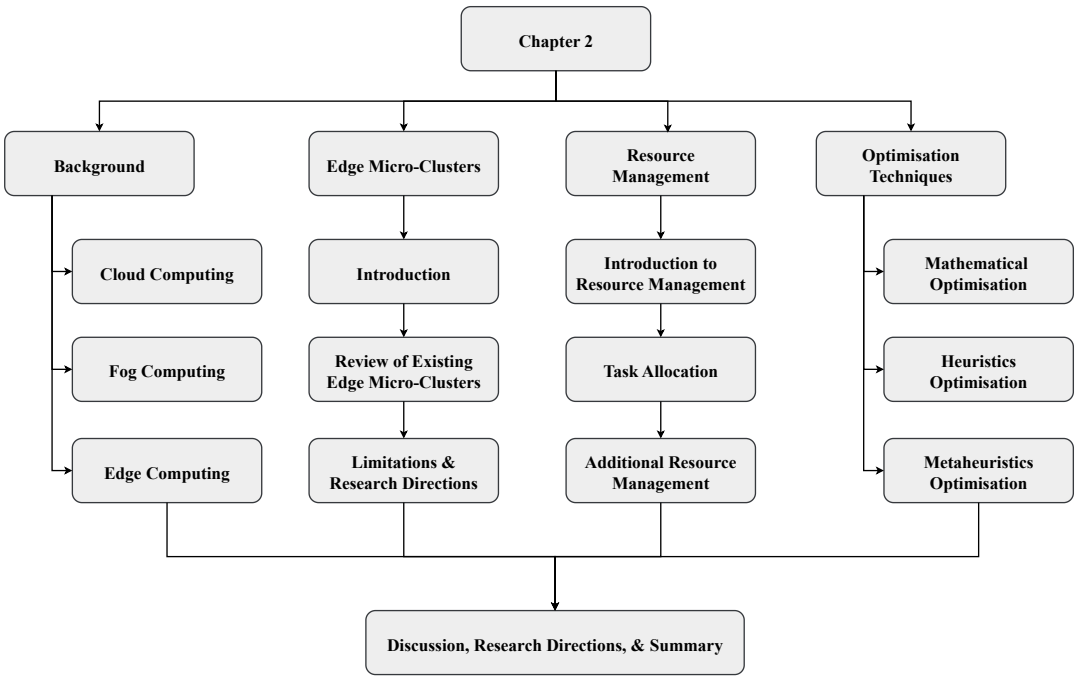


Figure 2.2: Chapter 2 outline.

2.2 Cloud Computing

Cloud computing is a computing paradigm relying on massive, complex, and centralised computing infrastructures that provide unlimited virtualised computing and storage resources over the Internet. It facilitates on-demand services such as Platforms as a Service (PaaS) and Software as a Service (SaaS) for end users. Cloud computing is provided by centralised data centres operated by providers such as Google Cloud Platform, Amazon Elastic Compute Cloud (EC2), and Microsoft Azure. Cloud is a key enabler for new applications, such as Smart Cities, Artificial Intelligence, and the IoT [28].

However, IoT allows billions of devices to connect, generate, and offload data to the cloud, which subsequently brings new challenges to cloud providers [23]. These challenges are generally characterised by high latency, networking jitters, non-responsiveness, security concerns, data privacy issues, and high power consumption and carbon emissions. These challenges have led to revisiting the cloud conventional computing model by pushing computing resources towards the edge of the network introducing new emerging computing paradigms known as edge computing, fog computing, and cloudlet computing [3, 28, 29].

2.3 Fog Computing

Fog computing is a hierarchical virtualised computing paradigm that deliver computing, storage, and networking resources across the networking path between cloud data centres and the edge of the networks. Fog computing hierarchically distributes resources across three or more tiers of architectures, including edge, fog, and cloud computing. Resources in fog computing are characterised to be rich and powerful to offer cloud services to mobile and end users. The overall objective of fog computing is to enable IoT applications and mobile users to utilise available computing resources outside centralised cloud data centres to improve networking latency, data privacy, and application responsiveness [2, 17, 21]. Figure 2.3 shows the network hierarchy description and the location of the emerging computing paradigms.

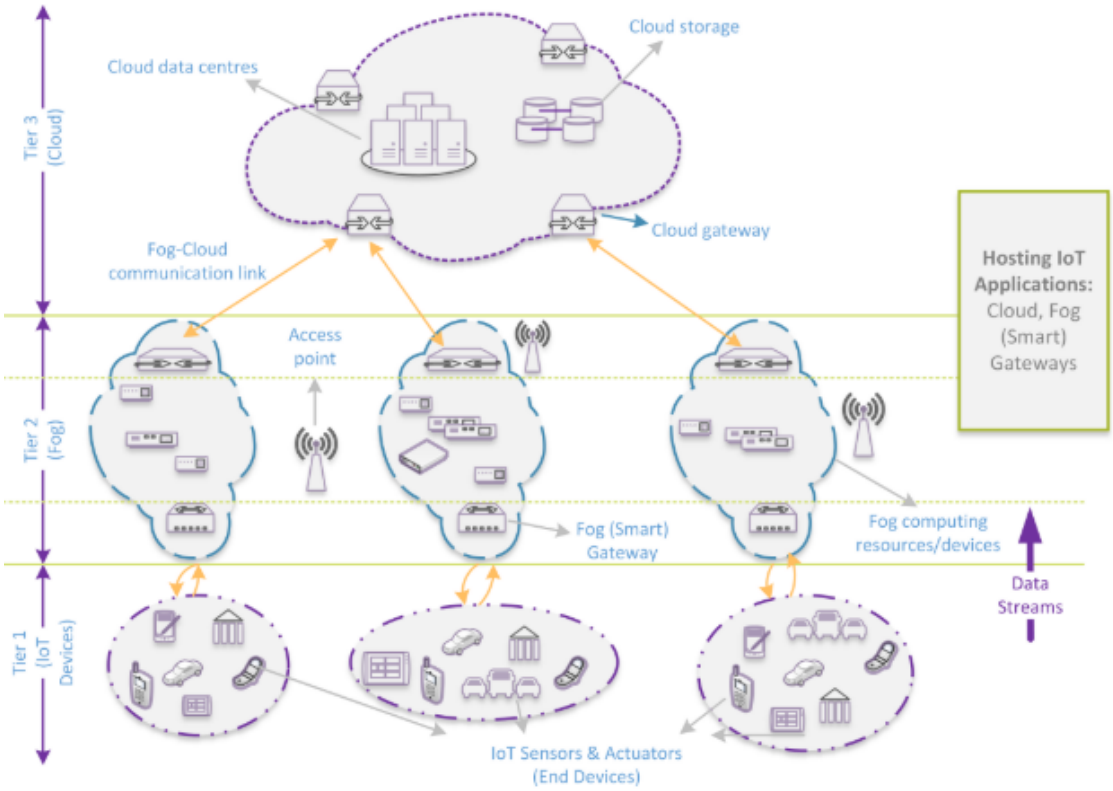


Figure 2.3: Hierarchical framework for edge, fog, cloud computing [30].

2.4 Edge Computing

Edge computing is a distributed computing model in which computation and storage processes are deployed at or near the edge of a network, where end devices like sensors and IoT devices are located. Edge computing allows for faster data computation and decision-making by reducing latency, energy consumption, and application responsiveness [17, 21, 31].

It has become a mainstream computing paradigm, driven by growth in the adoption of 5G and Internet-of-Things (IoT) technology. Many end-user applications require low-latency data processing and interactive responses, motivating the provision of highly available edge compute capability. Edge computing can bring several advantages to end users and applications, including low latency, high bandwidth, high security, improved privacy, low energy consumption and monetary cost [23, 32].

Edge computing and fog computing are used interchangeably in the research literature to generally refer to the utilisation of computing and storage resources distributed between cloud and edge. However, fog computing is a more hierarchical computing paradigm with rich resources and a wide coverage range. In contrast, edge computing is limited in computing resources and geographical coverage [17].

Overall, edge computing and fog computing are not competitors to cloud computing. Instead, these paradigms represent methods to augment cloud computing by decentralising resources across the edge of the network to support mobile and end users to reduce deployment problems related to latency, energy consumption, and privacy issues [4, 5].

2.4.1 Cloudlet Data Centres (Cloudlet)

One core technology behind the advancements in edge and fog computing is the Cloudlet technology. Cloudlet is a customised resource-rich cloud data centre typically located near the edge of the network to support mobile and stationary devices. The objective of Cloudlet data centres is to reduce the latency and energy consumption in cloud data centres by allowing appellations and end users to offload computation tasks and deliver application services near data sources. Cloudlets are limited to cover a wide range of end devices at the edge [1, 21]. To address the coverage challenge, end devices may be utilise either centralised cloud data centres or use more devices at the extreme edge such as edge micro-cluster platforms or edge computing.

2.4.2 Mobile Edge Computing (MEC)

Mobile Edge Computing (MEC) is a form of Edge Computing. MEC is based on a distributed computing paradigm that can bring computing and storage resources close to end users. MEC consists of virtualised servers deployed in various locations, typically in cellular networks, such as base stations, access points, and networking infrastructures, to support mobile users and devices and to enable real-time applications and low-latency communication. Overall, MEC extends edge computing and fog computing but is more limited to mobile and cellular network infrastructure. [14, 24, 33, 34, 35, 36].

2.4.3 Content Delivery Networks (CDNs)

Content Delivery Networks (CDNs) are distributed networks of servers that are utilised to cache and deliver application and web content to end users based on geographical locations. CDN servers are deployed in strategic locations, such as using Internet Service Providers' cellular networks and data centres, to enable fast and reliable content delivery to users. The main goal of CDNs is to improve the availability and responsiveness of applications and websites. However, the increment in the number of devices and users is a significant obstacle to CDN servers. In the context of edge computing, edge computing is an extension to CDN technology where computing and storage infrastructures that are located in close proximity to end users are utilised for various kinds of applications and not limited to web content as like in CDN technology [3, 11, 37, 38].

2.4.4 Edge Micro-Cluster Platforms

Edge micro-clusters are dedicated computing units deployed very close to IoT devices at the extreme edge of the network. A micro-cluster system can comprise various heterogeneous but resource-limited computing nodes mounted in a mini rack case and controlled by a head node. These kinds of edge systems feature several advantages, including low power consumption, small-physical footprint, low cost, and easy configuration. Micro-cluster platforms represent feasible edge systems to extend computing services to remote environments located at the extreme edge of the network. However, resource capacity and node limitations in such edge micro-clusters platforms require effective and efficient management to optimise their deployment in IoT and edge environments [6, 11, 21, 39]. Section 2.5 provides a systematic review of these micro-clusters and their related deployments and management in the context of edge computing. It further highlights the research gap related to task allocation and optimisation required for such edge systems.

2.4.5 Other Edge Computing Technologies

In addition to the systems mentioned above, there are many edge computing-related technologies, including but not limited to Mobile Cloud Computing (MCC), Mist Computing, Dew Computing, mobile Ad hoc Cloud (Ad hoc computing), and Volunteer Cloud. These distributed edge-related technologies are generally proposed to enable and facilitate performing computing and storage logic outside centralised cloud data centres and to extend cloud services near IoT devices and users. Overall, the standard objective of these systems is to facilitate IoT communications and computation and mitigate deployment concerns related to network latency, application responsiveness, and energy consumption. On the other hand, they are different in computing capacity, geographic coverage, connectivity, and deployment context [3, 21, 40]. Figure. 2.4 represents a high-level overview of different edge computing-related technologies.

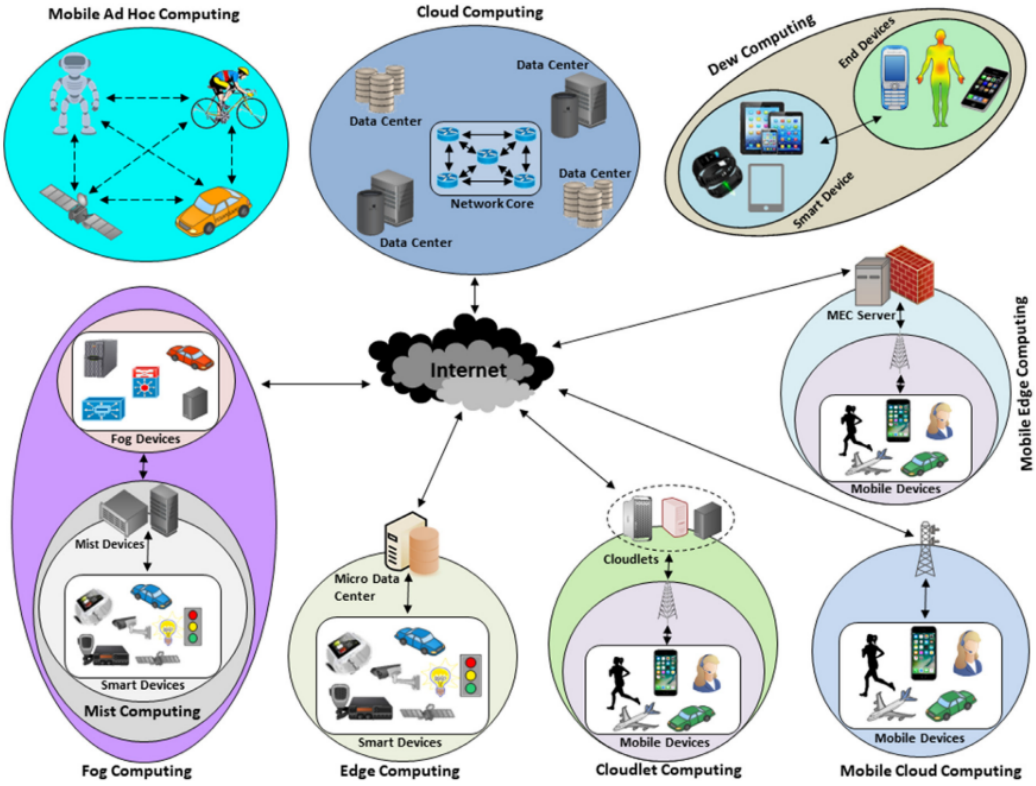


Figure 2.4: Edge computing architectures and related computing paradigms [21].

2.5 Edge Computing Micro-Clusters

2.5.1 Introduction

As mentioned above, micro-cluster systems represent promising computing infrastructure for practical edge computing. Such edge systems are being introduced by the edge and IoT research community for various research objectives. Recent studies establish the need for such compute clusters for edge applications and demonstrate the feasibility and suitability of using Single Board Computers (SBCs), such as Raspberry Pi devices, to build micro-cluster systems for performing edge-related workloads. Figure 2.5 presents exemplar edge micro-cluster platforms built by utilising SBCs for edge computing and IoT-related research.

This section systematically reviews existing work related to edge computing micro-clusters, with a particular focus on use cases, applications, and related performance evaluations of micro-cluster computing systems. Section 2.5.2 provides a high-level systematic review of the literature related to edge micro-clusters. Section 2.5.3 provides a discussion of the limitations in current studies and highlights the knowledge gap.

2.5.2 Review of Edge Micro-Cluster Systems

This section provides a high-level systematic review of research related to edge compute micro-clusters. The section presents a taxonomy focusing on various primary edge cluster projects, use cases and applications of micro-clusters, performance-related evaluation of micro-clusters, and studies that utilised alternative SBC-based edge systems. The review highlights initial research projects on micro-cluster systems, which aim to build experimental SBC cluster testbeds for students and researchers in cloud-related areas. It further explores studies that showcase various applications and use cases of micro-cluster systems. The section additionally provides performance analysis and evaluation studies that aim to investigate different capabilities of edge micro-clusters in terms of energy consumption, efficiency, and benchmarking of SBC clusters to evaluate their performance capabilities. The review finally examines alternative SBC-based edge micro-cluster solutions for potential edge computing applications.

2.5.2.1 Primary Research Projects and Initiatives

Recent research in the literature motivates and establishes building edge clusters by utilising SBCs such as Raspberry Pi devices. The Glasgow Pi Cloud [9], Iridis-Pi Cluster [41], and Bolzano Cloud Cluster [42] represent leading research projects for building edge SBC-based clusters. These clusters are proposed to motivate utilising small-factor computers, such as

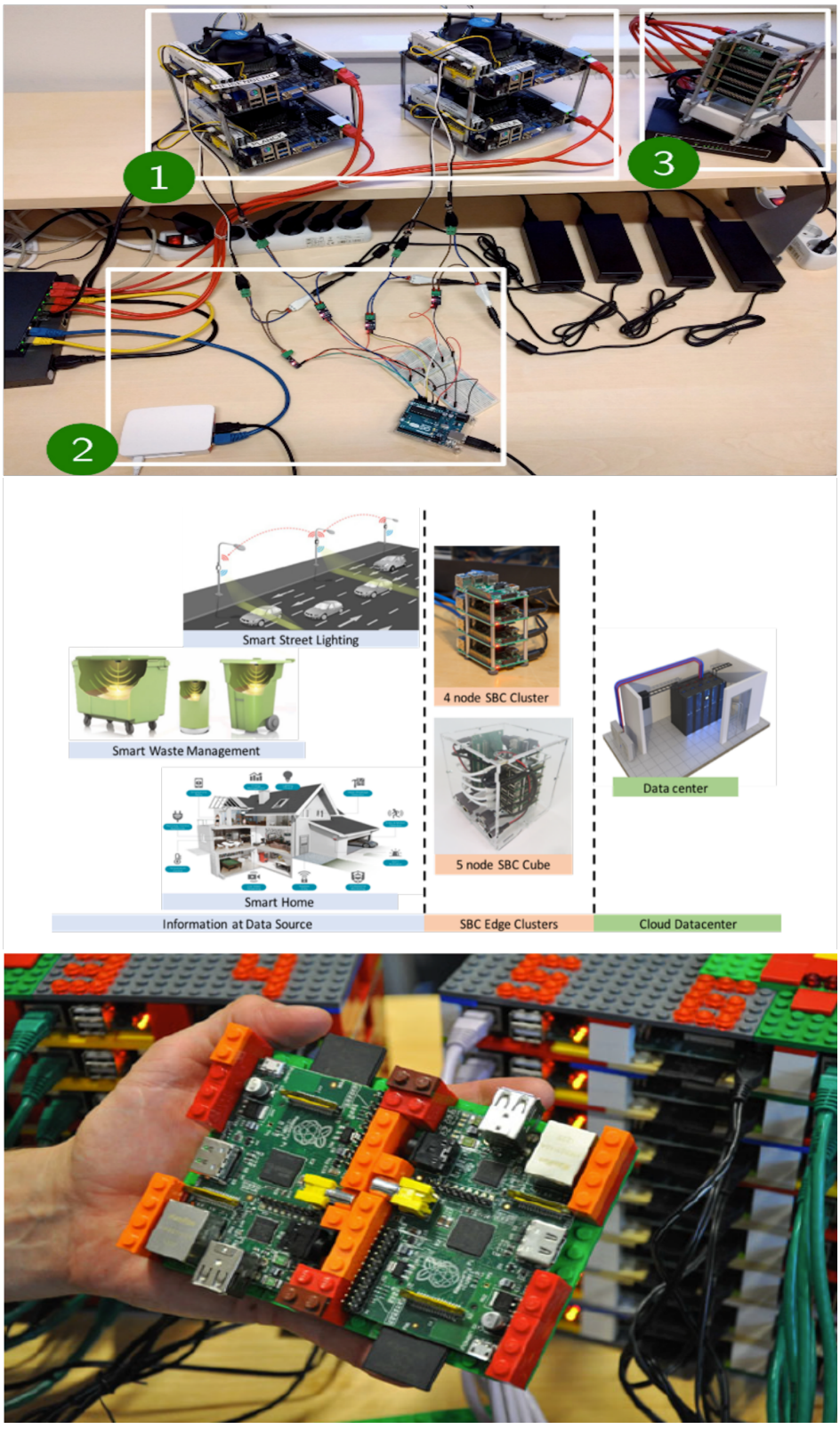


Figure 2.5: Examples of various edge compute micro-cluster systems for edge and IoT networks [7, 8, 41]

Raspberry Pi devices, to build physical edge data centres. The primary aim of these projects is to enable students and researchers to access physical clusters for education and research purposes, as the access to real cloud data centres is not affordable and represents a significant obstacle to many educational institutions.

The Glasgow Raspberry Pi Cloud (PiCloud) [9] is a scale model of cloud data centres composed of a cluster of 54 Raspberry Pi devices developed at the University of Glasgow. PiCloud is constructed to replicate the architecture of centralised data centres. PiCloud aims to provide students and researchers with hands-on experience and implement practical experiments related to cloud computing. The work represents an essential contribution that motivates the construction of physical micro data centres by utilising the hardware features and capabilities of Raspberry Pi devices.

Iridis-Pi cluster [41] is another project with SBC-based clusters constructed at the University of Southampton for educational purposes to enable students to learn high-performance computing. Iridis-Pi is a homogeneous micro-cluster consists of 64 Raspberry Pi Model B nodes mounted using a chassis built from Lego and interconnected via Ethernet. The work describes the overall architecture of micro-clusters in terms of both hardware and software and evalautes the performance of the cluster nodes using High Performance LINPACK (HPLINPACK) benchmarks.

Bolzano Cloud Cluster [42] is an early work that utilises Raspberry Pi devices to build an edge cloud cluster consisting of 300 Raspberry Pi 1 nodes. Similar to PiCloud and Iridis-Pi clusters, the primary purpose of the Bolzano Cloud Cluster is to provide experimental infrastructure for education and research related to cloud environments.

Furthermore, authors in [43] design a homogeneous SBC-based cluster consisting of 64 nodes using Raspberry Pi 3 Model B. The cluster was built for research and teaching purposes. The primary objectives are to enable students to understand high-performance clusters and to provide a testbed for researchers to implement and test algorithms.

In addition, Pi-Crust [44] is a medium-sized Raspberry Pi cluster composed of ten nodes of Raspberry Pi 2 Model B. The aim of the project is to discuss the equipment and configuration required for building SBC-based clusters. The work compares the performance of a new version of the Raspberry Pi cluster with old Raspberry Pi clusters and with a supercomputer machine available at Texas A& M university using a matrix multiplication problem.

Overall, these primary projects and initiatives motivate the feasibility and usability of constructing edge SBC-based cluster systems for experiments, education, and research into cloud and edge computing.

2.5.2.2 Applications and Use Cases of Edge Micro-Clusters

Several research works provide compelling use cases and applications of edge SBC-based clusters for edge and IoT environments. [6] discusses potential applications of SBC-based clusters. The article investigates different applications of SBC clusters that span various domains, including education, IoT, smart cities and edge computing. The authors conclude that the SBC clusters are 'enablers for edge computing' and consider such micro-clusters to be 'the game changers' in pushing computation intelligence toward the network edge. Furthermore, the work presents several examples of Raspberry Pi-based clusters developed for research and educational applications.

Qureshi *et al.* [8] provide the design and implementation of an SBC-based cluster for smart-parking management systems in smart cities. They constructed a medium-size SBC cluster using Raspberry Pi, Odroid Xu-4 and Latte Panda devices. The cluster communicates with parking sensors to monitor car parking systems and analyse daily data. The work presents a practical use case for SBC-based clusters and demonstrates the applicability of such micro-cluster for edge computing and smart cities applications.

Elkhatib *et al.* [45] provide Raspberry Pi-based micro-cloud (rPi-based micro-cloud) to deliver Web-related services. The work examines four evaluation metrics: service latency, hosting capacity, I/O overhead and system startup latency. Generally, the work demonstrates the usability of such micro-clusters to host Web services and notes several performance limitations related to responsiveness when the number of concurrent users is high. Therefore, more efforts are required to evaluate resource management and optimisation to cope with the noted performance limitations.

Further work demonstrating the applicability of SBC clusters in edge environments and software defined networks has been conducted in [46]. The work designed a small SBC-based cluster comprised of four homogeneous Raspberry Pi devices and developed an energy monitoring application for SBC clusters. The application basically profiles nodes' energy consumption for performing containerised services. The work generally presents a use case of SBC clusters as a component of edge and fog computing.

The work presented in [47] proposes tactical edge cloudlets that can be hosted on vehicles or other platforms to provide edge infrastructures for computation offload for military users. Tactical cloudlets offer general purpose, discoverable and forward-deployed cloudlets located in one single hop in near proximity of mobile devices. The work presents a use case for edge systems to extend computation to more critical edge scenarios, such as the military.

Authors in [48] present an architecture for cloudlet computing based on a Single Board Computer cluster. They deploy a small Raspberry Pi cluster to act as a cloudlet edge server to provide data to simulated drivers. They show that Raspberry Pi clusters can be suitable

deployment solutions for IoT applications that require low computation power. The work demonstrates a use case for an edge cloudlet server based on SBC to act as a component of edge computing. However, more complex resource management and task allocation are required for inter-cluster management.

Overall, the research mentioned above showcases how edge micro-clusters are being utilised in different fields and demonstrates their deployment applicability for edge and IoT networks. However, performance-related analysis and additional complex management techniques and optimisation are still required.

2.5.2.3 Performance Analysis and Evaluation of Edge Micro-Clusters

Performance-related evaluations of edge micro-clusters are considered in several related research works. This includes evaluation of edge-related performance metrics such as energy consumption, resource utilisation, and containerisation applicability for such edge clusters.

The performance of various SBC-based clusters was analysed in [39]. The study analysed and compared the performance, energy efficiency, memory usage, and scalability of three 16-node SBC clusters constructed using Raspberry Pi 3 Model B, Raspberry Pi 3 Model B+, and Odroid C2, respectively. The paper, furthermore, presents a new SBC clusters construction technique (i.e., Pi Stack) using Printed Circuit Boards (PCBs) developed to enable edge computing deployment by allowing efficient control and easy configuration. Generally, the results from this work reveal significant performance improvement in SBC clusters compared with 64-node SBC clusters from 2013, giving natural improvements in the SBC industry. The study concludes that SBC clusters can perform extensive tasks and open new computational opportunities. Although the study provides meaningful technical analysis of SBC clusters and their promising future, the evaluation of complex resource management techniques was overlooked and still missing.

Rausch *et al.* [7] investigated energy and resource consumption characteristics of portable edge micro-clusters. They provided a prototype of a micro-cluster composed of four Mini-ITX form-factor hardware as computing nodes and used a Raspberry Pi-based cluster to generate clients' workloads. The work examined nodes' resource consumption and provided an analysis of related power characteristics. Overall, the work highlighted the applicability of edge micro-clusters for several edge computing use cases; however, further experiments are required to involve and examine complex management techniques related to workload allocations for such micro-cluster systems.

Morabito [49] conducted extensive experiments to evaluate the performance of different SBCs to run container technologies. The work benchmarks different SBCs, including Raspberry Pi 2 Model B, Raspberry Pi 3 Model B, Odroid C1, Odroid C2, and Odroid XU4 using

micro-benchmark tools, such as *Sysbench* and *Linpack*. The results from this work reflect that container technologies have minimal impact on the performance of the SBCs. However, clustering and workload management are not provided. The paper used micro-benchmark tools, which might not provide realistic performance. However, computer science benchmarks need to be realistic, representative and reflective of real-world scenarios to generate reliable, accurate results [50, 51]. Therefore, further evaluation for node clustering using realistic benchmark applications that represent real-world scenarios is recommended.

Furthermore, Pahl *et al.* [52] investigate the suitability of recent lightweight containerisation technologies and container cluster management solutions, such as Docker [53] and Kubernetes [54], to meet edge clouds based on SBC clusters. The study concludes that edge SBC-based clusters represent affordable Platform as a Service (PaaS) and can connect IoT and cloud data centres. In addition, lightweight virtualisation technologies demonstrate feasible solutions for such small clusters, giving lightweightness and interoperability features.

Authors in [55] constructed an edge compute cluster of eight Raspberry Pi devices for developing a lightweight auto-scaling for edge architecture. A fuzzy logic-based solution is proposed for auto-scale systems. The study overall shows a use case for SBC cluster platforms for edge computing. However, task allocation management and comparison with other techniques are not included in the work.

Miori *et al.* [56] examine deploying OpenStack Swift software platform on a Raspberry Pi-based cluster to increase the cluster performance for more potential edge applications. However, the study reveals some deployment limitations related to nodes performance and load balancing and suggests using lightweight container solutions such as Docker to distribute workloads.

Overall, the above mentioned performance-related research substantially contributes to evaluate various capabilities features of edge micro-cluster systems. However, further research for complex scenarios of workloads allocation and optimisation is required.

2.5.2.4 Other SBC-based Edge Micro-Clusters

Alternative to Raspberry Pi-based cluster systems, which are mostly used in SBC-based related studies, a few further research studies have utilised different forms of SBC-based edge clusters [7, 39, 57, 58]. As mentioned earlier, authors in [7] provide a prototype of a micro-cluster edge computer composed of four Mini-ITX form-factor hardware boards, investigate energy consumption and evaluate its applicability for edge environment.

Basford *et al* [39] build a SBC-based cluster constructed using 16 nodes of Odroid C2 devices. The cluster's performance was evaluated and compared with the performance of comparable SBC cluster systems configured using Raspberry Pi 3 (Model B) and Raspberry Pi 3

(Model B+). The results show that the Odroid C2 SBC-based cluster outperforms the Raspberry Pi SBC-based clusters in terms of performance, value for money, and energy efficiency.

Furthermore, in [57], the NVIDIA Jetson Nano Developer Kit is used and evaluated. The study explores the deployment of the NVIDIA Jetson Nano Developer Kit, a powerful Single Board Computer (SBC) capable of running artificial intelligence algorithms, in the construction of a small SBC-based cluster for smart grid applications. The performance of the cluster is compared with a single-node platform and a remote cloud server. The study presents alternative SBC-based clusters utilising various commodity SBCs for edge computing. However, necessary resource management is not addressed.

In [58], the authors integrate the Raspberry Pi 3 Model B with the Intel Movidius Neural Compute Stick, an embedded deep learning device, to enable deep learning and real-time image processing in vehicular edge computing systems. The study demonstrates that the combined system of Raspberry Pi 3 Model B and Intel Movidius Neural Compute Stick is capable of processing image applications in real time. This highlights the promising potential of Single Board Computers for edge computing applications. Yet, the evaluation of the concurrent workloads is required to identify the system's limitations.

These studies overall showcase various micro-cluster systems that utilise alternative forms of SBC devices to build and experiment with edge-clusters for edge computing applications. While these research demonstrate the feasibility and performance of other SBC-based micro-cluster solutions, further research work is necessary to investigate the complexity of workload allocation and optimisation in these systems.

2.5.3 Limitations and Research Directions

The existing research demonstrates that edge micro-clusters platforms, such as Raspberry Pi stacks and SBC-based micro data centres, represent practical 'cloud-in-box' infrastructures for edge computing and IoT environments. Such edge clusters are recognised as a 'game changer' for decentralising computing resources and a 'key enabler' for edge and fog computing [6]. These edge systems can provide deployment features such as low-cost, low-energy consumption, small physical footprints, and sufficient collective computing resources. Ongoing research aims to further improve feasibility of micro-cluster platforms for edge computing and IoT networks [59].

Table 2.1 summarises research related to edge micro-clusters and their use cases. Overall, edge micro-cluster systems have been utilised in the literature for education, research, IoT, smart cities, and edge computing applications. Generally, they are employed in simple edge application use cases without considering the complexity of edge scenarios for micro-clusters such as multitasking execution and the resource-constrained nature of micro-clusters.

There is an absence of deployment of such micro-clusters to address complex compute-intensive edge and IoT workloads. The majority of studies in this field discuss hardware specifications, software and virtualisation technologies, and construction methods for such edge systems. However, there is a lack of integration and evaluation for complex multitasking edge scenarios, which require efficient and effective workload management and optimisation techniques that are necessary for these micro-clusters platforms.

Optimisations and resource management represent core components of distributed infrastructures. They encompass different management aspects to address problems and enable efficient task allocation and resource utilisation to meet different application QoS requirements. Therefore, further research is still required in this direction to investigate different workload management aspects and re-examine the performance of optimisation techniques for such micro-cluster platforms. By integrating appropriate optimisation techniques for task allocation in micro-clusters, we could enhance the functionally and capabilities of such micro-cluster systems for edge and IoT environments.

This thesis, therefore, aims to address this knowledge gap by investigating task allocation and evaluating optimisation techniques necessary for edge micro-cluster systems. This can be achieved by utilising appropriate optimisation techniques that can provide effective and efficient optimised task allocation for edge micro-cluster deployment. The following sections discuss resource management aspects and optimisation techniques in the context of edge computing.

Table 2.1: Comparison of hardware, software, and deployment context of various edge cluster platforms.

Papers	**Nodes Qty**	**Node Type**	**Deployment Context**	**Resource Management**
Abrahamsson *et al.* [42]	300 node	Raspberry Pi Model B	Education and Research	NA
Alhaizaey *et al.* [10]	8 node	Raspberry Pi Heterogeneous	Edge Computing	Task Allocation
Basford *et al.* [39]	3 clusters 16 nodes each	Raspberry P Model B, Raspberry Pi Model B+, and Odroid C2	Edge Cluster Construction using Pi Stack	NA
Bourhnane *et al.* [57]	3 node	NVIDIA Jetson Developer Kit	Machine Learning	NA
Cox *et al.* [41]	64 node	Raspberry Pi Model B	Education	NA
Fabian *et al. et al.* [55]	8 node	Raspberry Pi 2 Model B	Edge Computing	Auto-Scaling
Damián *et al.* [48]	4 node	Raspberry Pi 3 Model B	Edge Computing	NA
Johnston *et al.* [6]	NA	Raspberry Pi	Edge, Education, IoT	NA
Miori *et al.* [56]	NA	Raspberry Pi1 model B	Edge Computing	NA
Mikhail *et al.* [43]	64	Raspberry Pi 3 Model B	Multipurpose Clusters	NA
Qureshi *et al.* [8]	8 node	Raspberry Pi 3B+, Odroid Xu-4, and LattePanda 4G	Smart Cities Management Systems	Kubernetes environment
Rausch *et al.* [7]	4 node	Mini-ITX	Edge Computing	Simple Load Balancing
Sagkriotis *et al.* [46]	4 node	Raspberry Pi Model B	Power consumption	NA
Tso *et al.* [9]	56 node	Raspberry Pi 1 Model B	Education	NA
Wilcox *et al.* [44]	10 node	Raspberry Pi 3 Model B	Education	NA

2.6 Resource Management in Edge Computing

This section discusses resource management in edge and fog computing paradigms. It first highlights several systematic literature review articles that comprehensively review edge resource management from different perspectives. After that, the section discusses various resource management-related studies, with a particular focus on task allocation in edge computing. Finally, the section provides a high-level overview of different optimisation techniques that are being utilised to solve orchestration-related problems in in distributed environments.

2.6.1 Introduction to Resource Management

Resource management in distributed environments can be defined as an 'umbrella' term that covers different resources and workload management aspects. This includes hardware components, systems software, and orchestration techniques that manage resources and workload. The overall objective of resource management is to run systems effectively in order to meet various applications' Qualitty of Service (QoS) requirements and systems' Service Level Agreements (SLA) [60].

Resource management is a critical concern in edge and fog computing. Research states the complexity and difficulty of managing resources in edge computing. This complexity might be attributed to several co-dependent factors, including 1) the dynamicity of the workloads at the network edge, 2) the heterogeneous nature of devices deployed at the edge, and 3) the resource constraints at the network edge [11].

With the ongoing trends in edge and fog computing as promising distributed computing paradigms for IoT applications, several literature survey articles and review papers have been published to discuss and review edge computing-related concepts from different points of view, including preliminaries, such as definitions, architectures and frameworks to advance management and orchestration-related issues, such as resource management, virtualisation solutions, and optimisations techniques.

Hong and Varghese [11] provide a comprehensive survey on resource management and challenges in fog and edge computing. The survey investigated resource management from three perspectives: architecture, infrastructure, and algorithmic perspectives. (1) The architecture perspective described the overall fog and edge systems and were classified into data flow architectures, which deal with data flow directions. It defines the data flow direction, whether systems offload data from the edge to the cloud. Controlling architectures describe mechanisms that manage the overall environment and are defined as: centralised and distributed control. Tenancy architectures describe the type of tenancy, whether systems host a

single tenancy or be shared by multiple tenancies. (2) The infrastructure perspectives identify hardware and software facilities used to deploy and manage resources in Fog and Edge computing. (3) The algorithm perspective identifies management techniques that underpin fog and edge and is classified into discovery, benchmarks, load balancing, and placement algorithms.

Ghobaei-Arani *et al.* [61] review resource management approaches in edge and fog computing. They conducted a comprehensive systematic review of resource management approaches in edge and fog and classified resource management into six main categories: application placement, resource scheduling, task offloading, load balancing, resource allocation, and resource provisioning. The survey categorises the techniques utilised to solve management problems into heuristic-based algorithms, metaheuristic-based and mathematical-based methods. Furthermore, the survey finds that the majority of works used simulation-based tools for solution implementations, including iFogSim, CloudSim, and Matlab environments.

Authors in [21] provide a comprehensive survey of optimisation techniques and their applications in the context of edge computing. They first discuss various deployment infrastructures related to edge and fog computing, followed by a taxonomy of the optimisation algorithms employed to solve optimisation problems in edge paradigms. The paper furthermore presents a classification of orchestration techniques and their attributed evaluation metrics. The survey classifies optimisation algorithms into integer programming, heuristic, and heuristics optimisation. Similar to [61], optimisation problems were categorised into scheduling, allocation, placement, offloading, load balancing, and provisioning, whereas the performance metrics were grouped according to the orchestrations techniques, including but not limited to makespan time, latency, cost, and energy consumption. Finally, the authors discuss evaluation environments employed to evaluate optimisation techniques, including simulation, analytical and testbeds tools. The survey designs a research guideline framework, which helps in designing and solving optimisation problems in the fog and edge computing research community.

Aslanpour *et al.* [62] shed light on performance evaluation metrics in cloud, fog, and edge computing. The authors re-classify evaluation metrics as the computing model becomes more complex with the proliferation of the Internet of Things and emerging distributed paradigms. They analysed various performance metrics used to evaluate cloud, fog, and edge computing orchestration techniques. They provided a comprehensive taxonomy and detailed definitions of relevant metrics and their applications. They categorise metrics according to the MAPE-K orchestration process standard provided by IBM, in which orchestration techniques are defined according to four deployment phases: Monitoring phase, Analysing phase, Planning, and Execution phases. This classification aims to support developers and researchers in selecting and evaluating the most relevant metrics for optimisation techniques.

From another perspective, Mansouri *et al.* [36] review cloud and edge computing with a particular focus on resource virtualisation techniques, which are fundamental technologies in both paradigms. They investigate different virtualisation solutions deployed in cloud and edge and state that cloud classical virtualisation, such as hypervisors-based and Xen, cannot be effectively deployed in edge environments as edge naturally utilise more heterogeneous and resource-constrained devices that are not capable enough to run heavyweight virtualisation and consequently require lightweight-virtualisation techniques like containerisation and docker. The article thus recommends that combining classical and lightweight virtualisation techniques, for example, Xen and containerisation, might yield efficient deployment plans for different IoT applications. However, selecting appropriate virtualisation solutions still depends on three main factors: virtualisation techniques features, devices capabilities, and IoT application requirements.

Overall, this section provides a high-level overview on resource management concepts and highlights several systematic literature review articles related to resource management in edge, fog, and cloud computing paradigms. The survey articles addressed the topic from different viewpoints covering management aspects, optimisation techniques, and evaluation environments in edge computing-related paradigms. The following sections discuss those resource management aspects, focusing on task allocation concepts.

2.6.2 Task Allocation in Edge Computing

In the context of edge computing, task allocation techniques are applied to produce optimal or near-optimal allocation decisions for mapping IoT-related workloads to edge devices. The overall objective is to generate effective allocation solutions that meet different QoS requirements of IoT applications. This typically involves minimising edge-related performance metrics, such as makespan time, energy consumption, and latency. Task allocation is a complex optimisation problem in edge computing due to resource heterogeneity and the distribution of edge infrastructures. Researchers have proposed task allocation solutions by utilising various optimisation techniques. Solutions generally can be classified into categories, including heuristic-based solutions, metaheuristics-based solutions, mathematical-based solutions, and machine-learning solutions [11, 21, 61].

Heuristic-based solution are utilised in [30, 63, 64]. Taneja *et al.* [30] develop a heuristic-based algorithm to deploy applications for the fog-cloud paradigm. The proposed technique is composed of three-integrated algorithms to find allocation solutions. The algorithms iterate from fog nodes toward cloud nodes, placing application modules on the most eligible fog nodes. If no fog nodes are available or eligible for processing appellations, contact is made with cloud nodes. The evaluation is conducted using iFogSim considering different networking typologies with varied workloads.

The authors in [63] develop a heuristic-based solution to manage users and cloudlet servers association in fog computing and mobile cloud computing. The authors formulated the problem as an integer linear programming and developed a heuristic-based solution to optimise cost metrics. The developed solution is solved using the Gurobi optimiser, examined using simulation, and compared with various techniques, including best-first, random and optimal solutions. Overall, the work addressed how mobile users can select an appropriate cloudlet server to deploy services. Empirical evaluations are recommended to examine the solution' precision.

Rausch *et al.* [64] provide scheduling workloads in edge systems. The work presents a container scheduling system based on a greedy heuristic multi-criteria decision-making algorithm, i.e., Skippy. Skippy interacts with container management systems, like Kubernetes, and schedules containers to system nodes according to the edge system state. The system presents a method to enhance container management systems to allow such container management systems to support edge systems. However, the paper does not consider edge workload dynamic, multi-tasking and multi-tenancy scenarios in edge nodes, which might require involving and evaluating relevant optimisation techniques.

A few research works employed metaheuristics-based optimisations [65, 66, 67, 68, 69, 70]. The authors in [65] developed a genetic algorithm-based solution to optimise service allocation in fog computing. They compared the proposed solutions with the greedy first fit, exact-based solution, and cloud solutions using iFogSim simulation. The study shows that their genetic-based solution outperforms other techniques by not violating application deadlines. However, the complexity of the proposed solutions requires further investigation .

Azimi *et al.* [66] compared two metaheuristic-based optimisation techniques, Particle Swarm Optimisation (PSO) and BAT algorithm, to optimise application allocations in edge environments. They developed a Particle Swarm Optimisation-based solution to optimise applications execution time in edge architecture that is comprised of three layers: the things layer, the edge layer, and the cloud layer, where the edge layer contains multiple edge clusters. The proposed PSO-based solution is compared with the BAT algorithm, another swarm-based optimisation technique. The experimental simulation results show that PSO outperforms BAT optimisation in finding the best fitness value across the proposed scenarios. An extended version of the work with technical implementation detail is presented by the same authors in [67]. However, the works require further evaluation in realistic edge environments.

Gill *et al.* [68] proposed a Particle Swarm Optimisation solution for scheduling task in smart home environments to optimise various QoS metrics including energy consumption, latency, bandwidth, and response times. They implemented the proposed solution using iFogSim and compared it with two algorithms, namely: round robin and first come first serve scheduling.

A Genetic Algorithm-based solution is proposed for scheduling applications in a cloud-fog

computing environment in [69]. The developed solution is compared with Bee Life Algorithm, Modified Particle Swarm Optimization, and Round Robin using iFogSim simulation for a cloud environment and a fog environment. The work provides an overall performance comparison of various evolutionary algorithms for a cloud-fog environment. However, the study does not consider other performance metrics and evaluate realistic cloud-fog systems.

Cano *et al.* [70] consider the problem of variant task allocation for distributed robotics systems using constraint programming, greedy heuristic, and local search metaheuristics showing that constraint programming outperforms other techniques. They evaluated the proposed solution in a multi-agent navigation environment. The system's complexity requires further evaluation.

Mathematical-based solutions are utilised in [71, 72, 73, 74]. Skarlat *et al.* [71] developed a mathematical-based optimisation model to optimise application mapping and resource utilisation for a fog landscape. The developed mathematical-based model is implemented using IBM CPLEX solver. The Fog computing environments and the involved entities, such as applications and fog nodes, are implemented using iFogSim simulation software. The developed mathematical-based solution is compared with first-fit heuristic-based and cloud-based solutions for different application scenarios. The work requires further evaluations of the complexity of the developed optimisation model and an evaluation in a realistic fog environment.

Yin *et al.* [72] present task scheduling based on containers for fog computing manufacturing systems. They proposed optimise scheduling tasks in smart manufacturing systems by a mathematical-based solution. The solution optimises the task execution time and reallocates tasks to reduce delay. The system is evaluated using simulation, and the complexity of the system is not provided and requires further evaluation.

Authors in [73] extended the framework presented in [65] to address the fog service placement problem (FSPP) in a fog environment, which aims to find an optimal mapping between IoT applications and fog computational resources. The objective is to optimise fog resource utilisation while satisfying the QoS requirements of IoT applications. The FSPP is mathematically modelled and evaluated using benchmarking different target platforms representing fog and cloud implementation. However, the solution's complexity is provided. In addition, evaluation in a realistic fog environment is required to validate the estimated performance.

Wang *et al.* [74] consider the network dynamicity, such as users' mobility and load changing, to dynamically decide which mobile micro-cloud should perform the incoming computations offload from mobile users in mobile edge computing environments. They proposed two algorithms, offline and online algorithms, to find the allocation decisions. The proposed solution is evaluated by simulation using a real-world data set of user mobility traces of taxis.

Machine learning is utilised in the literature to address several resource management problems in edge computing [13, 75, 76, 77, 78, 79]. Liu *et al.* [13] proposed a Machine learning-based solution to optimise resource allocation in IoT networks for smart farming as a use case. The proposed to optimise resource allocation by classifying the IoT devices into different clusters according to defined priorities using a centralised K-means clustering algorithm running at edge gateways. After performing clustering, the cluster with the highest priority is allocated to the edge server, and the cluster with the lowest priority is allocated to compute tasks locally at IoT devices. The work is analysed using simulations and compared with local computing, edge computing server, and greedy scheme. Further work is required to evaluate multiple concurrent users in a realistic edge environment.

Authors in [75] developed a reinforcement learning mechanism to address application distribution in multi-layer fog-cloud computing systems. The work addresses the questions of which and how many services are to be deployed in fog resources instead of cloud resources. A deep reinforcement learning-based solution is developed to dynamically learn the optimal deployment solution without prior knowledge of fog resource states. The model is validated using face detection and mobile game use cases. The evaluation is conducted using an Odroid-XU4 board, a laptop machine, and an Amazon t2.micro cloud machine.

Xiaolan Liu *et al.* [76] addressed resource allocation in edge computing and IoT networks using reinforcement learning. The proposed solution decides whether to execute tasks locally or offload tasks to edge devices for processing. The system objective is to minimise a weighted sum of the task executing latency and power consumption. The solution is evaluated using simulations and compared with local computing, where tasks are executed at local end devices and in edge computing, where tasks are offloaded to the edge server. The study shows the effectiveness of RL in addressing resource management. The work only evaluates for one edge server to offload tasks, while edge computing typically has several local edge servers. Considering the availability of several edge servers and the comparison with other techniques to examine the solution's complexity and the overhead on the system in realistic edge environments is recommended.

Furthermore, Mao *et al.* [77] utilised deep reinforcement learning to build a system to manage resources from experience for large-scale systems. They simulated a system environment comprised of various resources where tasks arrive online. Overall the study shows that RL performs comparably to ad-hoc heuristics to optimise the task's average slowdown time, defined as the task completion time divided by the task's ideal duration time, in large-scale systems. The work requires more experiments to validate in realistic environments.

An edge cloud architecture (ECO) based on Machine Learning is presented in [78]. The authors developed the ECO to train machine learning models and evaluated it on several use cases like federated learning, transfer learning and stage model deployment. Overall, the

study shows applicability of the developed model to train ML at the edge.

Bian *et al.* [79] develop online task scheduling for fog computing environment using deep reinforcement learning to achieve multi-resource fairness among tasks and reduce task latency. The proposed technique is compared with random-based and shortest execution time. Similar to the RL-related papers, the work demonstrates the viability of RL techniques for resource management. However, the complexity and practical evaluation require further evaluation.

Table 2.2 provides a summary of task allocation research in edge and fog computing systems. Overall, task allocation concept covers techniques for producing optimal or near-optimal decisions of mapping IoT tasks to edge or fog resources in order to optimise various QoS deployment requirements. The above section examined studies in task allocation in edge computing with a particular emphasis on the utilised optimisation techniques, evaluation environments, and deployment contexts. As aforementioned, the techniques utilised in the literature largely fall under categories including heuristics, metaheuristics, mathematical-based, and machine-learning techniques. Furthermore, it is noted that simulations and analytical tools are mostly utilised for experimental evaluations, and there is an absence of using realistic edge environments for empirical evaluation, which is necessary for generating meaningful results. In this research, the experimental evaluations will be conducted using a configured edge micro-cluster testbed.

Table 2.2: A summary of task allocation in edge and fog computing.

Papers	Optimisation Technique	Experimental Environment	Deployment Context
Azimi [66]	Metaheuristic-based	MATLAB Software	Fog Computing
Bian [79]	Reinforcement Learning	Simulation	Fog Computing
Cano [70]	Constraint Programming and Metaheuristics	Robotics Systems	Distributed Systems
Gill [68]	Metaheuristic-based	iFogSim Simulation	Smart Home Environment
Hongzi Mao [77]	Reinforcement Learning	Simulation and Data Set	Cloud Computing
Hong Yao [63]	Heuristic-based	Simulation	Fog Computing and Mobile Cloud Computing
Liu [13]	Machine Learning	Simulation	Edge Computing for Smart Farming
Nan Wang [75]	Reinforcement Learning	A computer machine and a cloud instance	Fog and Cloud computing
Nguyen [69]	Evolutionary Algorithms	iFogSim Simulation	Cloud-Fog Environment
Nisha Talagala [78]	Machine Learning	Edge Server	Edge-Cloud Computing
Rausch [64]	Heuristic-based	Edge Testbed	IoT, Edge, and Cloud Environment
Skarlat [71]	Mathematical-based	iFogSim Simulation	Fog Computing
Skarlat [65]	Metaheuristic-based	iFogSim Simulation	Fog Computing
Taneja [30]	Heuristic-based	iFogSim Simulation	Fog-Cloud Computing
Venticinque [73]	NA	A workstation machine and a Raspberry Pi 3	Fog Computing
Wang [74]	Mathematical-based	Simulation	Mobile Edge Computing
Xiaolan Liu [76]	Reinforcement Learning	Simulation	Edge Computing and IoT Networks
Yin [72]	Mathematical-based	Simulation	Fog Computing and Smart Manufacturing

2.6.3 Other Resource Management Aspects

This section provides a high-level overview of relevant resource management concepts in the context of edge, fog and cloud computing. It briefly reviews studies related to load balancing, offloading and migration, resource provisioning, and energy consumption, with a focus on the employed optimisation techniques and evaluation environments.

2.6.3.1 Load balancing

Load balancing is a significant concept in edge and fog computing. Load balancing techniques are used to balance workloads over system nodes and monitor the system to avoid overloading certain nodes while other nodes are idle. [11, 61, 80, 81, 82, 83].

Song *et al.* [80] construct a fog computing system model using graph theory and proposed a dynamic load balancing techniques based on graph repartitioning to address load balancing in fog computing. The proposed system model and solution were evaluated using a distributed system Hadoop to simulate a fog environment.

The work in [81] proposes a mathematical-based model to address load balancing between two edge data centres by a cooperation-based solution. The proposed cooperation-based technique allows two edge data centres to cooperate and balance the load in case of overloading to reduce the execution time of tasks. The work was mathematically optimised. Further additional experiments are required to consider more than two edge servers and evaluate the solution complexity.

Authors in [82] propose a load balancing solution for Edge Data Centers (EDCs) in fog computing paradigms that enhances security and load balancing in EDCs. The solution first authenticates EDCs prior to offloading workloads in order to avoid unauthenticated edge servers. EDCs broadcast their overloaded workloads along with their ID number to other EDCs. The proposed technique obtained the fastest response time compared with other techniques, such as random, proportional, and static allocation. However, authenticating EDC could prolong the response time since this could add an extra job. This could affect QoS and QoE as well. The work is conducted and evaluated using MATLAB Simulation.

Authors in [83] optimise the problem of load balancing in distributed Internet of Vehicles (IoV). The authors presented a centralised modified constrained particle swarm optimization (MPSO-CO) to improve the latency performance and load balancing in fog and cloud environments. The experiments results show that the MPSO-CO outperforms other load balancing algorithms such as PSO-CO, greedy load balancing algorithm, and Max-Min Load balancing algorithm. The proposed algorithm was implemented using simulation.

2.6.3.2 Offloading and Migration

Offloading and migration concepts cover techniques that deal with how and where to offload and migrate services in edge and fog computing. This generally can be either from end devices to edge nodes, between cloudlet and edge servers in edge network, or between edge nodes and remote cloud data centres. Offloading and migration allow the resource-constrained end devices to offload their intensive-computational tasks to more powerful resources located in the near edge or centralized cloud centres. It also provides migration of services between edge nodes to maintain proximity to the mobile or stationary end devices [11, 61, 84, 85, 86, 87].

Barbalace *et al.* [84] study service migration between heterogeneous edge nodes to support mobile users in keep physical proximity to edge servers. The paper proposes a solution, i.e., H-Container, allowing migration of containerised applications between compute nodes with different CPU architectures. The system is evaluated using a variety of ARM 64-bit and x86 64-bit computers.

Authors in [85] investigate different Machine Learning approaches to predict offloading time required to offload stateless and stateful docker containers in cloud and fog computing. Stateless means offloading container images without preserving the application state, while stateful deals with live migration of application states with the container images. They compare four different Machine Learning prediction approaches, namely Multivariate Linear Regression (MLR), Polynomial Multivariate Regression (PMR), Random Forest Regression (RFR), and Ridge regression (RR). The results from the study show that RFR outperforms MLR, RMR, and RR. They evaluated the solution using lab-based systems. However, further work is to consider and address resource limitations in fog nodes.

A latency-aware workload offloading to allow mobile users to offload workloads in distributed edge cloudlets is proposed in [86]. The technique is based on a mathematical model to minimise workload response times. The work is evaluated using simulation and compared with two other offloading strategies. The first one considers offloading to the nearest cloudlet to the mobile user measured by the round trip time between users and the cloudlet. The second strategy offloads to a remote cloud data centre with sufficient resources. The solution complexity and evaluation in realistic environments are required.

The work in [87] proposed a seamless offloading mechanism to support the migration of web applications in edge cloud environments. The authors proposed a serialization algorithm, which captures, saves and restores the web worker states after the mobile user is migrated to another edge or cloud server. The evaluation experiments are conducted using three web applications and evaluated in one mobile worker (Odroid-XU4), two edge servers (source and destination servers) and one cloud server. The solution complexity when several edge servers are available requires investigation.

Generally, offloading and migration techniques support mobile users in decision-making related to how, when and where to offload their intensive-computational workloads in edge computing. In addition, it supports edge servers to collaborate and migrate services between them.

2.6.3.3 Resource Provisioning

Resource provisioning covers management approaches that determine the required resources in edge computing. It dynamically manages resources to enable effective and dynamic resource auto-scaling features to cope with workload characteristics and network dynamics at the edge. [11, 55, 61, 88].

Authors in [55] proposed an algorithm using fuzzy logic to enable edge cluster systems to dynamically auto-scale resources to deal with dynamic changes in the edge environment. The algorithm was evaluated using an edge compute cluster composed of several Raspberry Pi devices.

Le Tan *et al.* [88] propose a load prediction technique based on edge data centres locations to enable effective auto-scaling in mobile edge computing. They propose a solution based on Vector Auto Regression (VAR), a statistical model for calculating changes in data over time, to predict load in edge data centres. The solution is evaluated using simulation and real mobility traces of the taxis data set and compared with unknown-location load prediction.

Overall, resource provisioning provides techniques that enable auto-scaling in edge systems. Resource provisioning techniques can support edge systems to automatically cope with network dynamics and uncertainties in edge environments. It can help to avoid resource over-provisioning, which leads to increased cost and energy consumption, or resource under-provisioning, which might negatively affect the QoS requirement of applications.

2.6.3.4 Energy and Power Management in Edge Computing

Energy consumption and power management are critical for edge computing and IoT networks due to factors such as resource constrained of edge devices, hardware heterogeneity, limited power sources, and increasing demands of workload execution at the edge. However, little work has been conducted to model, manage, and optimise energy consumption for edge micro-cluster systems [7, 46, 49, 89, 90].

As mentioned and discussed in Section 2.5.2, Rausch *et al.* [7] investigate power consumption characteristics of edge-based computer clusters. The work employs a simple load balancing technique to investigate the relation and between resource utilisation and energy consumption in edge-based computer clusters. Sagkriotis *et al.* [46] develop an energy monitoring application for SBC-based edge clusters. The application profiles energy consumption of

nodes. In addition, [49] investigates power consumption for various SBCs that are utilised in edge-cluster systems. The work utilised systems benchmarks to examine the impact of deploying docker containers on energy consumption on several SBC devices, including Raspberry Pi and Odroid. Furthermore, Wiesner and Thamsen [89] develop an edge simulator tool, i.e., LEAF simulator, for modeling energy consumption and determining power usage of nodes in fog computing environments. A survey on energy aware edge computing is provided [90]. The survey provides a systematic review that reviews existing efforts on energy efficiency in edge computing.

As mentioned above, not much work addresses and optimises energy consumption for edge micro-clusters. In addition, as stated in [90], most of the existing research focuses on a single objective, such as latency, privacy, power, or energy efficiency. Therefore, there is a need to model and optimise multi-objective optimisation of energy efficiency and execution times in edge systems.

2.7 Optimisation Techniques

Optimisation techniques are algorithms designed to solve complex optimisations problems in several domains like cloud computing, edge and fog computing. Optimisations are a core component in cloud computing and becomes more essential in fog and edge computing [62]. They play an essential role in solving complex orchestration and optimisation problems in these distributed and heterogeneous environments. This section briefly presents different optimisation techniques that are being employed to solve orchestration-related problems in edge, fog and cloud, specifically focusing on their reasoning, advantages, and disadvantages. These techniques generally fall under three main categories: mathematical optimisation, metaheuristic-based optimisation, and heuristic-based optimisation [21, 61, 59].

2.7.1 Mathematical Optimisation

Mathematical optimisations are deterministic techniques that can provide optimal solutions for complex optimisation problems. Mathematical-based optimisations systematically solve the optimisation problems by minimising or maximising the system's objective functions. Overall, mathematical optimisation can be classified into 1) integer programming, 2) mixed-integer programming, 3) non-linear integer programming, and 4) non-linear mixed-integer programming. Mathematical optimisation generally features advantages, such as providing the best optimal solutions and allowing for specifically modelling optimisation problems by defining systems variables and constraints. However, several disadvantages make mathematical optimisations not suitable for complex management problems. Mathematical optimisa-

tions can be computationally expensive, and the system complexity can grow exponentially for large and complex optimisation problems with many systems variables and constraints. Furthermore, they might require intensive computations and consume the systm computing resources. In addition, mathematical optimisations require expert system engineers to model optimisation problems and define system constraints mathematically [21].

Generally, it seems that mathematical-based optimisations are inappropriate optimisation techniques for resource management in large distributed environments, such as in huge centralised data centres. This is because of the complexity and computation time for large and complex distributed systems [91, 92, 93, 94]. Therefore, heuristics-based and metaheuristics-based solutions are more effective and efficient optimisation techniques to cope with such complexity [36].

2.7.2 Metaheuristics Optimisation

Metaheuristics optimisations are stochastic-based techniques in which a degree of randomness is employed to find optimal or near-optimal solutions for complex optimisation problems. Metaheuristics Optimisation find an optimal or near-optimal solution by iteratively optimising solutions for several iterations. Metaheuristic-based techniques are successfully used in software engineering and cloud environments where optimal solutions are challenging to obtain and require a trade-off between several conflicting objectives [21, 95, 96, 97].

Authors in [98] and [99] review metaheuristic optimisations resource management in cloud settings. In [98], Comparative analysis of six metaheuristic techniques has been conducted using the CloudSim simulator to evaluate workload scheduling in a cloud environment. The authors analysed the performance of six different metaheuristic techniques: Ant Colony Optimisation (ACO), Particle Swarm Optimisation (PSO), Genetic Algorithm (GA), Artificial Bee Colony (ABC), Crow Search Algorithm (CSA), and Penguin Swarm Optimization (PeSOA), to optimise makespan time and cost metrics. They implemented experiments using the CloudSim simulation tool and provided knowledge on the performance of metaheuristics for scheduling tasks in cloud environments. In [99], a review of various metaheuristics-based optimisation techniques used for optimising Virtual Machine Placement (VMP) in cloud data centres is presented. The review described the logic of different metaheuristics techniques and discussed their implementation for VMP in cloud environments. The paper, specifically, reviewed Simulated Annealing (SA), Genetic Algorithm (GA), Ant Colony Optimisation (ACO), Particle Swarm Optimisation (PSO) and Biogeography-Based Optimisation (BBO).

Metaheuristics optimisation features several advantages, including 1) easy implementation, 2) fast convergence time compared with exponentially complex mathematical optimisations, 3) the ability to find optimal or near-optimal solutions for complex optimisation problems,

and 4) the ability to avoid local-optimum solutions. However, optimal solutions are not always guaranteed and premature convergence might occur due to the difficulty in adjusting the variables.

2.7.3 Heuristics Optimisation

In contrast to metaheuristics-based optimisations, where techniques follow nature or swarm dynamics to search spaces to find optimal or near optimal solutions, heuristics-based optimisation techniques are stochastic algorithms that follow specific instructions to solve complex optimisation problems in feasible computational time. Because many edge computing-related orchestration problems, such as task allocations and resource scheduling, are complex optimisation problems, the exact optimisation might not be applicable to solving the problems in a feasible time. Heuristics-based optimisation features easy implementation and fast computation time. However, optimal solutions are not guaranteed. They generally can be classified into 1) construction heuristics, in which the algorithm follows specific rules for searching for the optimal solution, and 2) improvement heuristic, in which the solution is iterative and slightly improve or change the candidate solutions [21]. For example, greedy-based techniques are examples of heuristic optimisations that finds best local choice at each time.

2.8 Discussion against Related Work

This section provides a critical analysis of the related work, identifying gaps in the field which will be addressed through this research contribution.

Micro-clusters have been introduced in the research areas related to edge computing and IoT networks. These edge computing systems facilitate practical deployment solutions for edge and IoT applications by leveraging their deployment features such as low latency, low energy consumption, and high networking bandwidth. However, existing research efforts are largely limited to high-level management aspects, such as construction techniques, use cases and applications, without considering the complexity of task allocation and optimisation techniques necessary for operating such resource-constrained edge systems.

Despite the progress made in researching micro-cluster systems for edge computing, it is noted that there is an absence of work involving complex management problems related to task allocation and optimisation for edge micro-cluster systems. In addition, the literature review reveals that most studies investigating edge-related resource management rely on simulation-based solutions and analytical tools, such as CloudSim, iFogSim, or Matlab; in contrast, there is an absence of utilising realistic edge environments for empirical evaluation, which is necessary to generate meaningful results.

Task allocation techniques aim to provide optimal or near-optimal decisions of mapping IoT tasks to edge or fog resources in order to optimise various QoS deployment requirements. Task allocation solutions in the context of cloud and edge computing are based on optimisation techniques including heuristics, metaheuristics, mathematical-based, and machine-learning techniques. These optimisation techniques differ in 1) their effectiveness in finding optimal or near-optimal solutions for complex optimisation problems, and in 2) their efficiency in terms of computation time and the number of iterations required to converge or find feasible solutions. The No Free Lunch Theorem (NFLT) states that no single optimisation technique can be most applicable to all optimisation problems [21, 100]. This mandates further examination for micro-cluster systems. Because of the differences of edge micro-cluster systems in terms of (for example) resource availability, node capabilities, networking connectivity, and power management, it is imperative to reexamine the performance of relevant optimisation techniques to capture their effectiveness and efficiency for task allocation optimisation in edge micro-cluster settings.

In comparison to existing research efforts, this research aims to address and optimise task allocation for edge micro-cluster systems. The experimental evaluations in this research will be conducted by utilising a configured physical edge micro-cluster testbed to address the lack of empirical performance evaluation observed in existing studies. By addressing these research gaps and leveraging a realistic testbed, this research contributes to enhance

the applicability of micro-clusters in edge computing and IoT environments. The empirical evaluations will provide valuable insights into the performance of different optimisation techniques in real-world edge micro-cluster scenarios.

2.9 Summary

This chapter has laid down the background and theoretical framework for this thesis. It comprehensively reviewed literature related to emerging edge computing technologies, task allocation and resource management at the edge of the network, and the relevant optimisation techniques.

The chapter provided a high-level overview of edge-related systems and an in-depth review of existing micro-cluster systems for edge computing and IoT networks. Specifically, it first presented a general overview of the computing landscape and the emerging distributed paradigms that have been proposed to facilitate edge computing. It highlighted several edge computing systems constructed for enabling edge computing with technical-related configuration and use cases. Furthermore, the chapter presented an overview of resource management in edge computing with a particular focus on task allocation and optimisation techniques for edge computing. Figure. 2.2 demonstrates the overall picture of the related work.

As described in Section 2.5, edge micro-cluster systems have been introduced in the context of edge computing as practical solutions for decentralising resources toward the edge of the network. However, the literature review revealed that the related work around such micro-clusters are limited to high-level management issues, such as construction techniques, hardware and software specifications, high-level scenarios and use cases. There is an absence of evaluating significant management issues related to complex task allocation and optimisation techniques necessary for such edge micro-cluster systems.

To address these research directions, this thesis presents and evaluates the applications of various task allocation optimisation techniques to allocate and optimise task allocation in edge micro-cluster platforms. Chapter 3 provides and characterises a realistic edge micro-cluster system for edge and IoT environments and empirically evaluates various task allocation optimisation techniques for solving task allocation. Chapter 4 develops an analytical linear model for predicting the execution time for processing workloads in edge micro-clusters and provide an evaluation of PSO-based metaheuristic optimisation technique for micro-clusters. Chapter 5 extends the work by modelling and evaluating energy consumption for micro-clusters and develops a multi-objective optimisation framework that facilitates optimising other edge-relevant performance metrics in such edge systems.

Chapter 3

Experimental Evaluation of Feasibility and Task Allocation of Edge Micro-Clusters

This chapter characterises and advocates the feasibility of edge micro-cluster systems as practical deployment solutions for edge computation. It presents and discusses the technical requirements of micro-clusters including device specifications, software systems and virtualisation features, and workload characteristics. Furthermore, various task allocation optimisation techniques were explored and evaluated for optimising the execution of heterogeneous workloads for batch-arrival execution in micro-clusters. This chapter is based on the work that has been published and presented in the IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks [10].

3.1 Introduction

Edge micro-clusters are used to extend and provide computing and storage services locally or at edge of the network. Such micro-cluster systems are composed of a small number of resource-constrained computing nodes that are interconnected to execute complex computations in a decentralised mode. The design of edge micro-clusters depends on various factors, including node specifications, resource availability, and workload characteristics.

The focus of this chapter is on the design of the experimental framework and the evaluation of the feasibility and task allocation techniques for micro-cluster systems. We first design and describe the experimental framework that we use to evaluate the feasibility and optimisation performance for edge micro-clusters. This framework includes the design of the

experimental micro-cluster testbed, the selection of computing nodes, the development of edge-representative workloads, and the evaluation of task allocation optimisation techniques.

In detail, this chapter characterises and motivates the concept of micro-cluster systems for edge computing and IoT environments. It describes and presents a useful micro-cluster testbed that represents a concrete instantiation of micro-clusters and demonstrates the characteristics of edge-relevant workloads for such compact platforms. Additionally, it provides lightweight resource management that executes at the edge, implementing task allocation among cluster nodes. Furthermore, several task allocation optimisation techniques are explored and evaluated, including randomised-based allocation, heuristic-based approaches, and mathematical optimisation by integer programming.

The implications of findings for the design and deployment of edge micro-clusters are identified. Results highlight the feasibility of micro-cluster systems for executing edge and IoT applications. We empirically demonstrate the feasibility and benefits of utilising micro-cluster systems for edge computing and IoT environments. Specifically, we show that edge micro-clusters using heterogeneous and resource-limited devices can effectively execute edge-relevant workloads in batch execution. In addition, we investigate and evaluate various task allocation techniques and demonstrate their impact on the overall performance of edge micro-clusters. Findings demonstrate a need for further research to advance micro-cluster systems, particularly in terms of task allocation and optimisation techniques. Overall, this chapter provides the structure of the experimental evaluation of feasibility and optimisation techniques for micro-clusters and contributes to the feasibility and advancement of such systems for edge computing.

The remainder of this chapter is structured as follows. Section 3.2 describes the experimental framework, including micro-cluster system setup, testbed configuration, benchmark software, and the networking structure. Sections 3.3 and 3.4 present the research assumptions and formulates task allocation problem for micro-cluster systems. Section 3.5 develops and evaluates various task allocation techniques for optimising task allocation. Section 3.6 presents results analysis and evaluation. Finally, the conclusion and future directions are discussed in Section 3.7.

3.2 Experimental Infrastructure

This section advocates the need to empirically evaluate experiments and optimisation techniques in a physical representative edge environment instead of using simulation software. It discusses the experimental infrastructures, including the system setup, device specifications and the used benchmark applications.

3.2.1 Rationale

The majority of studies in edge resource management utilise simulation-based software, such as CloudSim [101], iFogSim [102], MATLAB, LEAF simulator [89], or use frameworks to synthesize or emulate edge infrastructure [18, 103] to conduct and evaluate experiments. Simulation tools are affordable tools for researchers to evaluate their work as conducting experiments in real-world infrastructures can be expensive and might not be affordable for many researchers and institutions. However, simulations provide an high-level abstraction of the edge computing environment, which may not be representative of the full scope of uncertainties [62, 36].

Furthermore, sole dependence on simulation software might not be sufficient for evaluating experiments and techniques. As stated in [50], methodology in computing science research is crucial and plays a significant role in establishing a baseline for evaluating ideas and techniques. Methodology needs to be solid enough to generate trustworthy results and draw a meaningful conclusion. It requires relevant workloads, solid experimental design, and rigorous analysis.

Therefore, this research considers an empirico-realist approach for evaluation, which requires the use of a physical micro-cluster testbed, representative benchmark applications software, and wall-clock timing metrics for performance reporting. The scale of the micro-cluster testbed is limited; however, the findings will be more tangible and translatable to near-term pragmatic edge deployment scenarios. Section 3.2.2 describes the micro-cluster testbed configuration, while Section 3.2.3 presents the workload and applications benchmarks.

3.2.2 Micro-Cluster Testbed

To experiment with a physical representative edge micro-cluster, a micro-cluster testbed is constructed by utilising Raspberry Pi devices to represent a minimal instantiating of a micro-cluster, featuring heterogeneous resources that are capable of processing typical edge computation and storing relevant data.

As this framework is a distributed system, there is an intelligent resource management component that monitors the micro-cluster performance and maintains a profile of cluster nodes related metrics, such as CPU load, memory, network, and bandwidth. This functionality helps in creating appropriate task allocation plans that meet QoS requirements in each application scenario.

The configured edge micro-cluster comprises eight single-board computers (SBCs) that represent heterogeneous edge nodes, with 32 available compute cores and total of 11 GB mem-

ory. SBCs are drawn from different generations of the Raspberry Pi device. The micro-cluster testbeds are limited to Raspberry Pi devices; however, in the principle, such edge micro-clusters include comparable Single Board Computers, such as Odroid or BeagleBone. Table 3.1 presents the specific Raspberry Pi devices used in this paper.

While the devices share the same Arm architecture, they are heterogeneous in that they have different chipsets with differing cache sizes and clock speeds. The nodes are connected in a flat topology with a gigabit Ethernet switch. Each node is running the latest instance of the Raspberry Pi OS. Figure 3.1 presents the developed micro-cluster testbed setup.

In addition to the *performance*, *reliability*, *portability*, and *energy efficiency* requirements for edge systems recognised in [7], the micro-cluster framework is characteristically re-configurable and additionally facilitates the following key features:

1. *Expandability*, micro-cluster platforms could be easily expandable to allow for accommodating a variety of nodes. This is specifically relevant for provisioning requirements like scaling micro-cluster sizes.

2. *Distributability*, micro-cluster platforms can be distributable to allow splitting platforms across several locations, allowing minimises communication latency and extending computation services to new environments.

3. *Portability*, similar to [7], micro-cluster platforms can be compact enough to be easily portable to enable easy movement of the testbed to another location.

Table 3.1: Raspberry Pi devices specifications in micro-cluster prototype. The 4-node micro-cluster comprises one of each model. The 8-node cluster comprises three RPi2B, three RPi3B and one of each other model.

Cluster Node	**max CHz**	**Core Count**	**Memory**	**OS**	**Qts**
Raspberry Pi 2B	0.9	4	1 GB	Raspberry Pi OS	3
Raspberry Pi 3B	1.2	4	1 GB	Raspberry Pi OS	3
Raspberry Pi 3B+	1.4	4	1 GB	Raspberry Pi OS	1
Raspberry Pi 4B	1.5	4	4 GB	Raspberry Pi OS	1

3.2.3 Software Benchmarks

Benchmarks should be realistic, representative and reflective of real-world scenarios in computing science experiments. This is to enable researchers to reach valid conclusions [50].

Figure 3.1: Prototype of heterogeneous edge micro-cluster system with Raspberry Pi nodes.

Therefore, for a pragmatic empirical characterisation approach, this research identifies realistic benchmark-based compute workloads that can execute directly on micro-cluster nodes and report meaningful performance metrics. [104] reports and reviews several benchmarks developed for edge computing. Two types of benchmarks are reported, namely: micro-benchmarks and macro-benchmarks. Micro benchmarks are specialised to test system-level performance metrics like CPU, memory, network, and storage. However, these types of benchmarks might not be sufficient to fully understand the related system's performance for applications. In contrast, macro benchmarks are developed to understanding application system performance. The DeFog [19] and EdgeBench [20] suites are macro benchmarks that are generic, i.e., not dedicated for specific application types, and consist of idealized example tasks relating to edge applications that may be executed by edge devices.

DeFog [19] is a benchmark software suite developed to capture and understand the performance of different deployment platforms in Edge, Fog, and Cloud. The DeFog aims to compare applications deployments in three different deployment modes, namely a cloud deployment mode using Amazon Elastic Compute Cloud instance (Amazon EC2), an edge deployment mode using two different Single Board Computers, i.e., Raspberry Pi 3 model B and Odroid XU4, and a fog deployment mode that enables hierarchical deployment across cloud and edge, by using relevant performance metrics such as communication metrics and computational metrics.

Furthermore, the DeFog suite incorporates six applications benchmarks that might benefit

from edge and fog computing deployment, including image processing, audio processing, and game-based applications. DeFog is developed in the bash scripting language and utilises Docker containerisation technology which facilities instant application initialisation and task offloading.

DeFog benchmark suite is ideally suited for evaluating edge micro-cluster platforms, as it provides the following key features:

1. a representative set of pre-configured, containerised workloads with a straightforward script-based deployment model.
2. benchmarks that take common file formats (e.g. JPEGs, WAVs) as input.
3. realistic small-scale tasks that could be aggregated to produce large-scale use cases such as the edge applications outlined above.

In addition, there are key differences and inputs in how the DeFog benchmark suite is utilised to evaluate the performance of edge micro-cluster platforms. The reasoning of the DeFog benchmark had to be modified to suit edge micro-clusters deployment and enable running multiple concurrent workloads in nodes. Therefore, the benchmark codes are modified to enable task-level parallelism for benchmark applications across a cluster of heterogeneous nodes.

First, the original developers developed the DeFog benchmark suite to run and compare individual task execution in three deployment platforms, i.e., a cloud instance, an edge instance, and a fog instance. Instead, this research focuses on edge deployment mode, and the benchmark applications were re-developed to enable running multiple containerised tasks to study batch execution in edge micro-cluster systems.

In addition, DeFog compared the performance of cloud instances and edge platforms by executing independent application runs. The benchmark functionality is extended to model workloads heterogeneity characteristics to enable running multiple heterogeneous and concurrent workloads originating from different applications in an edge micro-cluster.

Furthermore, the DeFog framework only supplied limited input data for each benchmark. Supplementary input data were provided to allow the observation of batch execution of multiple concurrent instances of an application with different inputs.

Three DeFog benchmark applications were re-targeted and adapted to generate workloads for micro-cluster systems.

1. Object detection application (YOLO): This application deploys deep learning to provide a real time object classification for image processing. The application receives

image files in .jpg format, runs a pre-trained neural network model to provide an estimation of the objects inside the image, and then returns the result files as image files with overlaid object classifications and corresponding confidence level.

2. Speech to text conversion application (PocketSphinx): This application converts audio files to text files. The application receives audio files in .wav format and uses a pre-trained model to generate .txt files that contain the recognized text.

3. Text-audio synchronization application (Aeneas): This is a forced alignment application works to automatically generate a synchronization file that maps text fragments and audio clips. This application takes paired audio files and text files as input (.wav and.txt) and generates textual output files with embedded timing metadata.

These particular workload types are likely to become increasingly popular for edge end-users given the growing need for environment awareness and digital accessibility.

3.2.4 Networking Structure in Micro-Cluster Setup

The management and task allocation techniques are implemented using a MacBook machine directly connected to the micro-cluster testbed via a flat LAN. The MacBook machine is a 2.4 GHz Dual-Core Intel Core i5 processor and 4 GB of memory. The MacBook's hardware capabilities are limited and relatively compared to a head node of the micro-cluster, that is a Quad-Core A72 1.5GHz Arm processor and 4 GB of memory in Raspberry Pi 4 model B in the micro-cluster testbed. The development is implemented on the MacBook machine instead of using a cluster head node because the Google OR framework for optimisation, which we used for in task allocation techniques, does not work on Raspberry Pi devices due to dependency issues.

The communication structure between the MacBook machine and the micro-cluster is facilitated using a flat LAN with Secure Shell (SSH) and Secure Copy (SCP) protocols for secure and efficient communication and task transferring. The MacBook acts as a controller for the micro-cluster and communicates with nodes using SSH to remotely access and control the micro-cluster nodes. Task data are transferred between the MacBook and the micro-cluster using SCP.

The connection structure between the MacBook and the micro-cluster is a flat LAN topology, as shown in Figure 3.2. As illustrated in the diagram, all nodes are connected to the same local IP block. The MacBook communicates with each node in the micro-cluster directly over LAN, rather than through a cluster head node. This communication topology simplifies network management and reduces the complexity and overhead of routing and switching, making it an effective approach for smaller-scale networks like micro-clusters.

The network communication overhead for the micro-cluster is minimal, as the connection is done using the LAN. This means that the communication latency and bandwidth are appropriate to support the required data transfer. This enables efficient task allocation for the micro-cluster without significant communication overhead affecting the overall performance. Figure 3.2 illustrates the network topology used for the micro-cluster setup.

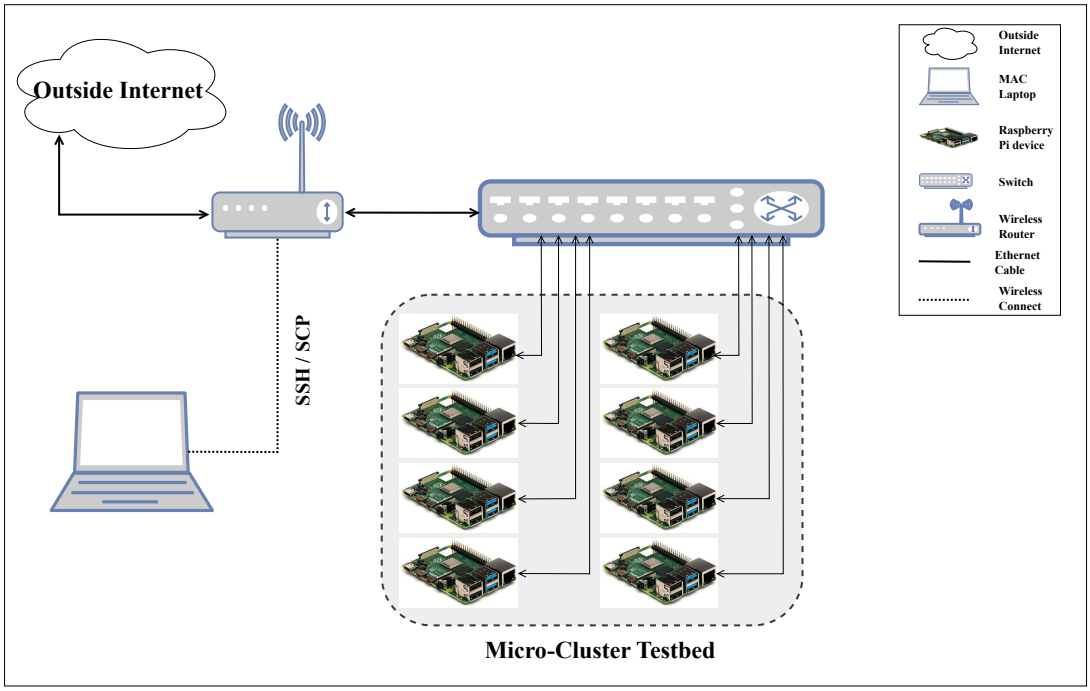


Figure 3.2: Network topology diagram for micro-cluster system.

3.2.5 Tasks Launching and Execution in Micro-Cluster System

The system setup is facilitated using several interconnected technologies for task dispatching, distribution and execution. The MacBook machine establishes connections with the micro-cluster system through a LAN and security mechanisms to ensure secure communication and access to the micro-cluster system. In particular, the MacBook machine communicates with the micro-cluster testbed using LAN network and offload tasks into the micro-cluster using SSH and SCP protocols. The system topology is described in Section 3.2.4.

Tasks are distributed to the micro-cluster's nodes according to the allocation decisions provided by the allocation techniques. The allocation decisions determine which node within the micro-cluster is responsible for executing each task. The decisions are provided by allocation techniques using metrics such as resource capacity in the allocation solution. The

allocation techniques are explained in Section 3.5.

Upon task arrival to the micro-cluster system, each task is allocated to one node according to the allocation decision. Tasks are processed in parallel execution across multiple nodes, leveraging the collective computational power of the micro-cluster system in order to enable efficient task execution. Tasks are packaged up into different Docker containers inside nodes to ensure execution isolation and avoid interference. A fresh container for each task is created to avoid overlapping or inadvertent data sharing. When a node finishes a task execution, the corresponding Docker container is terminated. This ensures that each task is completed independently without any interference from concurrent tasks. The overhead related to Docker setup does not affect the execution time as applications' Docker images have already been installed on the node storage media (SD card) in the experimental setup phase.

Overall, this system setup allows the MacBook machine to efficiently offload tasks to the micro-cluster system through a LAN network, leveraging the collective processing through parallel execution while ensuring secure and isolated task execution using container technology.

3.3 Assumptions and Observations for the System Models

This section outlines the assumptions and observations underpinning the edge micro-clusters, the system models, and the optimisation techniques implemented in Chapters 3, 4 and 5.

1. Edge micro-clusters are built using heterogeneous and resource-constrained devices, such as SBCs. We observe that such nodes have very limited resources which can be fully consumed for concurrent task execution. Therefore, we assume that each node has resource capacity limits, determining the number of tasks that can be executed concurrently on each node type. These resource capacity limits are determined using the system constraints for task allocation.

2. The assumption is that edge micro-clusters receive and execute tasks in batch execution mode. This means that tasks arrive simultaneously at the micro-cluster and are executed in parallel on multiple nodes. The execution of tasks in parallel execution across all nodes allows for concurrent execution leveraging the available resources efficiently.

3. System objective functions are modelled differently in Chapter 3 and 4, based on observations from the experiments, where the system objective function is an abstraction function that estimates the task execution time, while in Chapters 4 and 5, the system objective function follows a linear model.

- For Chapter 3, the system model provides an abstract system model that assumes the makespan time for executing a set of tasks is the sum of task execution times. We assume the task execution time on nodes based on ahead-of-time task profiling in which task-node execution time is defined by empirical measurement, coupled with some controlled statistical noise.
- For Chapters 4 and 5, we observe that task execution time linearly increases in relation to the number of concurrent tasks being executed in the node. Therefore, a new linear model is developed to estimate the required makespan time more effectively. This observation involves task profiling that determines the interpolation values required for different tasks in the linear model.

3.4 Task Allocation Formulation

3.4.1 Objective Function

In the context of edge micro-clusters, the system model is developed for executing a batch of tasks in a parallel execution mode across multiple nodes. The task allocation is formulated as an allocation problem of a set of tasks T to a set of heterogeneous nodes N within a micro-cluster. The objective is to minimise the makespan time required for executing all tasks.

The system model considers task allocation of a set of tasks T, in which each task has an estimated execution time that represents the execution time task t requires on node n; and a set of cluster's processing nodes N with different capabilities, in which each node has a capacity limit, defined in constraint 1. The objective is to effectively allocate tasks to nodes within the micro-cluster such that the makespan time is minimised.

Equation 3.1 defines the system objective function. The objective function is to minimise the makespan time of task allocation for micro-clusters. Thus, the decision is how to allocate task *t* to node *n* such that the makespan time of a set of tasks is minimised and nodes capacities are maintained.

$$\text{Minimize} \sum_{n \in N} \sum_{t \in T} cost[n][t] * x_{nt} \tag{3.1}$$

Where, $cost[n][t]$ is a matrix element represents the cost, i.e., the estimated execution time, associated with allocating task *t* to node *n*. The matrix rows and columns represent nodes and tasks, respectively. Data in the cost matrix represents the estimated task execution time required for allocating a particular task t on a node n. The matrix is populated in this model by systematically varying task execution time values. We assume a normal distribution with coefficient of variation $c_v = 0.5$, where mean value μ is based on observed execution times that are profiled ahead-of-time in the experiment setup phase.

The variable x_{nt} is a decision variable defining a binary integer variable 0 or 1, which denotes whether a cluster node n is assigned to a task t or not. Note that this is an abstract system model, and an alternative objective function based on linear model is developed in Chapter 4.

$$x_{nt} = \begin{cases} 1, \text{if node } n \text{ allocated to task } t \\ 0, \text{otherwise} \end{cases} \tag{3.2}$$

The overall objective is to find an allocation plan that effectively utilises cluster's nodes such that node capacities are maintained (*constraint-1*) and ensuring that each task t is assigned to at least one cluster node n (*constraint-2*). Meanwhile, the makespan time to complete all tasks is minimised. Table 3.2 outlines the notations used in the task allocation formulation for micro-clusters. Section 3.4.2 defines the system constraints.

Table 3.2: List of notations and related descriptions

Notation	Description
T	set of tasks to be allocated
N	set of cluster nodes
t	individual task
n	individual node
x_{nt}	decision variable (*indicating whether node n executes task t)*
cap_n	processing capacity of node n
$cost[n][t]$	cost matrix (*indicating cost of node n executing task t)*

3.4.2 Systems Constraints

Two systems constraints are defined to represent the heterogeneity and resource-constrained expected in the edge micro-clusters:

Constraint-1: This constraint defines a node capacity constraint cap_n, in which each cluster node has a maximum capacity. The total number of tasks allocated to each node

should not exceed the upper bound capacity of the node. Otherwise the node's compute resources will become saturated, and the task execution time will be prolonged.

$$\left(\sum_{t \in T} x_{nt}\right) <= cap_n \ , \ \forall \ n \in N \tag{3.3}$$

Constraint-2: each task t must be allocated to exactly one node. n This is to ensure that all tasks are allocated.

$$\sum_{n \in N} x_{nt} == 1 \ , \ \forall \ t \in T \tag{3.4}$$

3.5 Task Allocation Optimisation Techniques for Micro-Clusters

This section explores various task allocation techniques developed for optimising workload execution in batch for micro-cluster platforms. The objective is to allocate compute tasks on the most eligible node, by generating an effective and efficient allocation decision based on the current condition of the cluster. The main objective is to minimise the makespan of workloads in batch arrival, that is the total execution time of tasks, by efficiently using resources of micro-clusters.

The computing resources in micro-clusters are heterogeneous and limited. Therefore, they need to be utilised effectively and efficiently. Efficient allocation of applications workloads on the most suitable edge node is critical. Failure to efficiently manage workloads on cluster nodes could impact the overall micro-clusters performance and consequently affects Quality of Service requirements of applications.

Furthermore, such micro-clusters might be deployed in remote fields, making human management not available. Thus, micro-clusters should be configured to be 'self-managed'. In addition, workloads are expected to be heterogeneous and highly dynamic, and edge clusters should be able to self-adapt to deal with workload variations.

As mentioned earlier, management techniques could be executed on an elected head node of the micro-cluster system. Task allocation techniques are developed and implemented on a MacBook machine that directly connects and communicates with the micro-cluster over a flat LAN. The networking and communication between the MacBook machine and the micro-cluster testbed is illustrated in Section 3.2.4. Sections 3.5.1 and 3.5.2 present the developed task allocation techniques. Section 3.6 evaluates their performance in terms of efficiency and effectiveness.

3.5.1 Heuristic-based Techniques

3.5.1.1 Cluster Election Technique

The cluster election technique is a greedy-based heuristic that allocates tasks to nodes according to current CPU loads average of nodes. The cluster-election technique monitors CPU loads on nodes by polling the Linux OS in each node using a command like /proc/loadavg and allocate tasks to the node that has the minimum CPU load. This technique considers nodes CPU load average as nomination criteria. This technique can be enhanced by integrating other metrics like node power level or memory. Algorithm 1 presents pseudocode of cluster-election task allocation technique.

Algorithm 1 Heuristic Cluster Election Technique

Input: T (set of offloaded tasks); N (set of nodes in micro-cluster)

for *task* t *in* T **do**
    for n *in* N **do**
        measure current CPU load of n;
        n' be the node in N with lowest CPU load;
    end
    Assign t to n';
    Commence immediate execution of t;
end

3.5.1.2 Best Node Selection Technique

The best-node selection technique is another greedy-based heuristic technique that prioritises nodes based on their memory resources. This technique considers memory capacity of nodes as a a primary selection criteria. Specifically, tasks are always allocated to the node with higher memory capacity, which is node Raspberry Pi 4 Model B in the micro-cluster testbed. The devices in the micro-cluster testbed are heterogeneous in terms of CPU clock speeds, however, Raspberry Pi 4 model B is more powerful than other nodes in term of the memory capacity. Algorithm 2 presents pseudocode of best node selection allocation technique.

Algorithm 2 Heuristic Best Node Selection Technique

Input: T (set of offloaded tasks); N (set of nodes in micro-cluster)

for *task* t *in* T **do**

  for *node* n *in* N **do**

    Check memory of n;

  end

  let n' be the node in N with the maximum memory;

  Assign t to n';

  Commence immediate execution of t;

end

3.5.1.3 Random Allocation Technique

The random-based allocation is a heuristic technique that randomly allocates tasks to nodes. Specifically, tasks are forwarded to any arbitrary node without any resource or capacity consideration. All nodes are equally targeted in this technique. This technique is used as a baseline for comparison purposes. Algorithm 3 presents pseudocode of random-based technique.

Algorithm 3 Random-based Allocation Technique

Input: T (set of offloaded tasks); N (set of nodes in micro-cluster)

for *task* t *in* T **do**

  select n from N randomly;

  Assign t to n;

  Commence immediate execution of t;

end

3.5.2 Mixed Integer Programming Allocation Technique

The heuristic-based techniques mentioned above show that when nodes become overloaded, processing capabilities decrease and tasks' execution time increase. In this technique, balancing workloads between the cluster nodes is considered, taking into account both nodes capabilities and the workload requirements. The problem is, therefore, formulated as a combinatorial assignment problem and solved using mixed-integer programming technique (MIP). MIP allocates tasks based on a pre-estimated cost matrix that estimates the task execution time task on different nodes within the micro-cluster and considers nodes processing capacities, that is, how many tasks each node can process simultaneously without consuming its

resources. This technique, therefore, forwards tasks based on allocation solutions produced by the MIP allocation solver. This technique is implemented using the Google open software suite for optimisation, OR-Tools [105]. Algorithm 4 presents pseudocode for mixed integer-based allocation technique.

3.6 Performance Evaluation

This section analyses the performance of allocation techniques developed in Section 3.5 and discusses on the overall functionality of micro-cluster platforms for handling edge-relevant applications in the context of edge computing and IoT environment.

3.6.1 Minimising Makespan Time

The empirical evaluation considers the makespan time metric to evaluate and compare the effectiveness of different task allocation techniques to complete a batch of benchmark invocations.

The makespan is defined as the wall-clock time for the edge micro-cluster to complete processing a set of tasks, defined from the time when the first request is generated to the time that all cluster nodes complete processing their assigned tasks. In particular, the makespan time includes, (1) the communication time C, which is the time to offload tasks to the cluster nodes and to receive final results back through a communication medium, (2) the allocation time D, which is the time to required by allocation techniques to generate solutions, and (3) the execution time E, which is the actual time each node takes to process the assigned tasks.

The experiments were conducted on two edge micro-cluster configurations. Each cluster comprises a set of heterogeneous compute nodes. The first micro-cluster comprises four nodes, whilst the second micro-cluster comprises eight nodes. Table 3.1 shows the devices specifications and the quantity deployed in each cluster.

As mentioned above, the overall objective is to minimise the batch execution makespan time, measured as wallclock-time, while efficiently utilising the resources in micro-clusters. The performance of the task allocation techniques explained in Section 3.5 were evaluated on three benchmarks software developed from DeFog benchmark suite 3.2.3, that is the image-detection (Yolo), audio-to-text converting (PocketSphinx), and audio-text synchronisation (Aeneas), respectively. Tasks distribution, launching and execution in the micro-cluster are explained in Section 3.2.5.

Results show the makespan time of executing 32 concurrent tasks for each benchmark. Figures 3.3 and 3.4 represent the makespan time required by 4-nodes and 8-nodes micro-cluster configurations, respectively. The graphs show results based on the mean of 15 runs.

Algorithm 4 Mixed-Integer Programming Allocation

Input: T (set of tasks); N (set of nodes in micro-cluster);
$cost$ (cost matrix); cap_n (node capacity)
Output: $Allocation\ Decision$

```
solver = solver.MIP
 x = {}
 for n in N do
 |  for t in T do
 |  |  x [n, t] = solver.IntVar(0, 1)
 |  end
end
// Constraint 1: Total tasks for each node is less than node capacity
 for n in N do
 |  for t in T do
 |  |  solver.Add(solver.Sum(x[n, t]) <= cap[n])
 |  end
end
// Constraint 2: Each task is assigned to one node
 for t in T do
 |  for n in N do
 |  |  solver.Add(solver.Sum ([x[n, t]) == 1)
 |  end
end
objective = [ ]
 for n in N do
 |  for t in T do
 |  |  objective.append(cost[n, t] * x[n, t])
 |  end
end
solver.Minimize(solver.Sum(objective))
 status = solver.Solve()
 for n in N do
 |  for t in T do
 |  |  if (status == Optimal) then
 |  |  |  allocate n to t
 |  |  end
 |  end
end
Return Allocation Decision;
```

Analysis demonstrates that the MIP-based allocation technique efficiently utilises the cluster's resources and optimises the batch execution for computation-intensive applications (i.e., image-detection and audio-text converting benchmarks), while it achieves a comparable performance to other greedy approaches for the audio-text synchronisation benchmark.

The two heuristic-based techniques, i.e., cluster-election and the best-node selection, achieve comparable performance for all benchmarks. The cluster-election technique successfully reflects the nodes CPU load averages and efficiently nominates nodes. The best-node selection optimises for memory resources of nodes. This technique allocates tasks to the node type Raspberry Pi 4 model B because of the high memory capacity of the node in comparison with other nodes. Although the best-node selection technique performs better than the random-based technique, the relative performance of the best-node could vary depending on the workload characteristics and systems resources. Therefore, it is not a typical technique as results should generalise for other clusters configurations in which allocating everything to the one node will not be appropriate as the node resource will be consumed leading to poor performance. Finally, results indicate that the random-based allocation is an inappropriate technique because it consume nodes resources leading high high execution times for all benchmarks.

Overall, the MIP-based technique effectively represents task computation times and node capabilities. It optimises the makespan execution time and outperforms other techniques for computation-intensive applications, i.e., image- detection and audio-to-text converting. Furthermore, the MIP-based technique shows a stable performance when cluster nodes increase. The cluster-election technique demonstrates a good performance to reflect the clusters conditions and nominates capable nodes. The cluster-election achieves a better performance than the best-node selection for computation-intensive benchmarks. However, this technique could be further enhanced by integrating other metrics like node connections or power requirements. The selection of the best node is not always an intelligent choice for the edge-micro clusters, as nodes execution time increase when resources overloaded. Finally, for light-computation applications (i.e., audio-text synchronisation), the MIP-allocation technique performs as well as the best node selection.

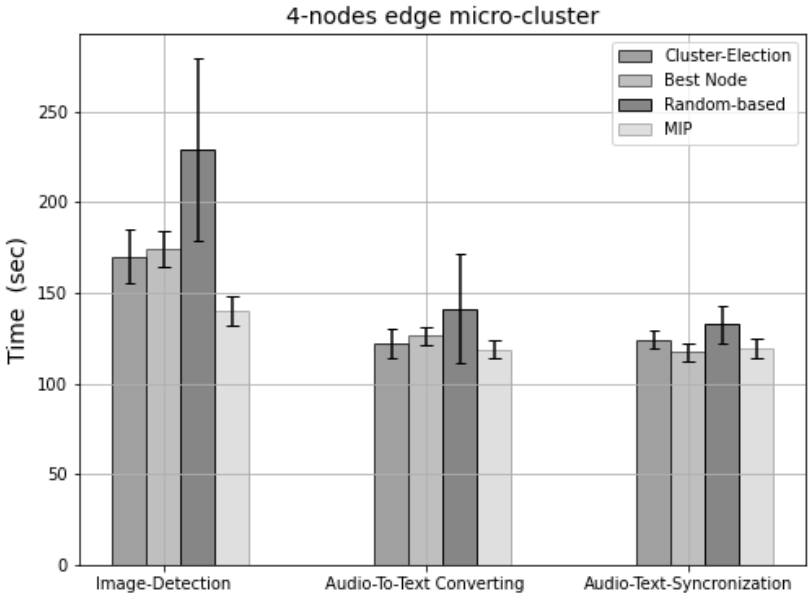


Figure 3.3: Makespan Time required to process of 32 concurrent tasks based on different allocation techniques running on 4-nodes micro-cluster (confidence intervals indicate one standard deviation).

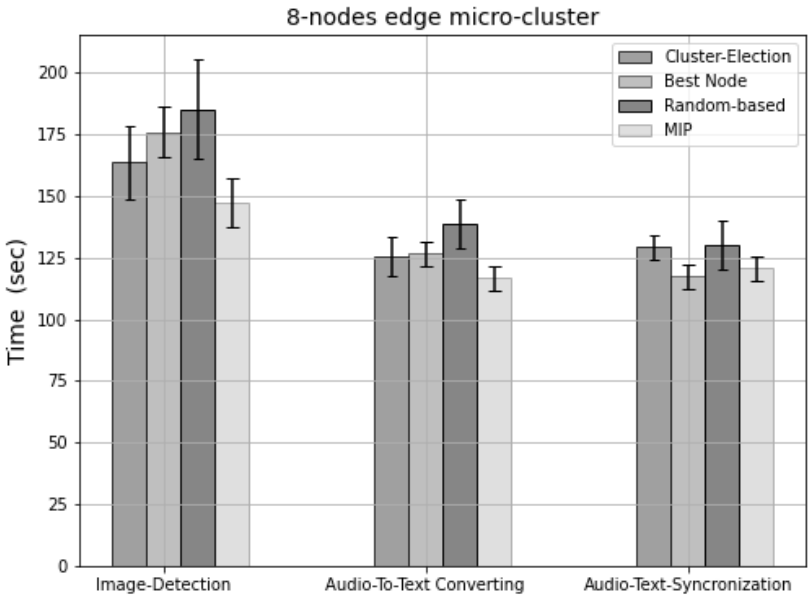


Figure 3.4: Makespan Time required to process of 32 batched concurrent tasks based on different allocation techniques running on 8-nodes micro-cluster (confidence intervals indicate one standard deviation) approaches running on 8-nodes edge micro-cluster (confidence intervals indicate one standard deviation).

3.6.2 Allocation Overhead

Another relevant evaluation metric is the allocation overhead time, which is defined to be the computation time an allocation technique requires to generate a solution. Allocation overhead is a critical metric in edge computing. Allocation techniques are required to be lightweight, not adding expensive overhead, as most IoT-based applications and management techniques are being deployed in resource-constrained devices and require lightweight management techniques. If the allocation techniques require expensive resources in terms of CPU, memory or energy, they would not be applicable in the context of edge computing.

Therefore, it is imperative to consider the lightness and the overhead of the technique when designing task allocation for edge context [62]. All makespan times reported above include the time for task allocation calculations, i.e. solution time for the MIP approach, live metric query time for the greedy heuristics, and pseudo-random function evaluation time for the random technique.

There is limited complexity for the MIP constraints meaning solution times are very short for the micro-cluster setup with these workloads. For the MIP-based technique, the allocation overhead never exceeds 1 second, which in the worst case is still below 2% of overall makespan time. The limits study experiment shows that MIP allocation for typical edge workloads and clusters remains below 1 second, for up to 1000 tasks (see Figure 3.5). However, overhead time grows exponentially for larger-scale micro-clusters, which requires further investigation.

3.6.3 Discussion

Overall, the experimental work discussed in this chapter demonstrates that micro-cluster platforms are capable of handling heterogeneous workloads in real-world edge application scenarios.

This requires lightweight and efficient task allocation techniques to effectively manage workloads, orchestrate nodes, and improve micro-cluster functionality. The deployment of lightweight but effective task allocation techniques for batched task execution is straightforward and can yield mathematically optimal solutions for tractable cluster sizes.

The work further presents a significant comparison between various lightweight task allocation techniques that are capable of improving workload allocations in micro-clusters. The evaluations reveal that the mixed-integer programming task allocation with respect to nodes capacities outperforms other heuristic-based task allocation and efficiently utilise the cluster resources.

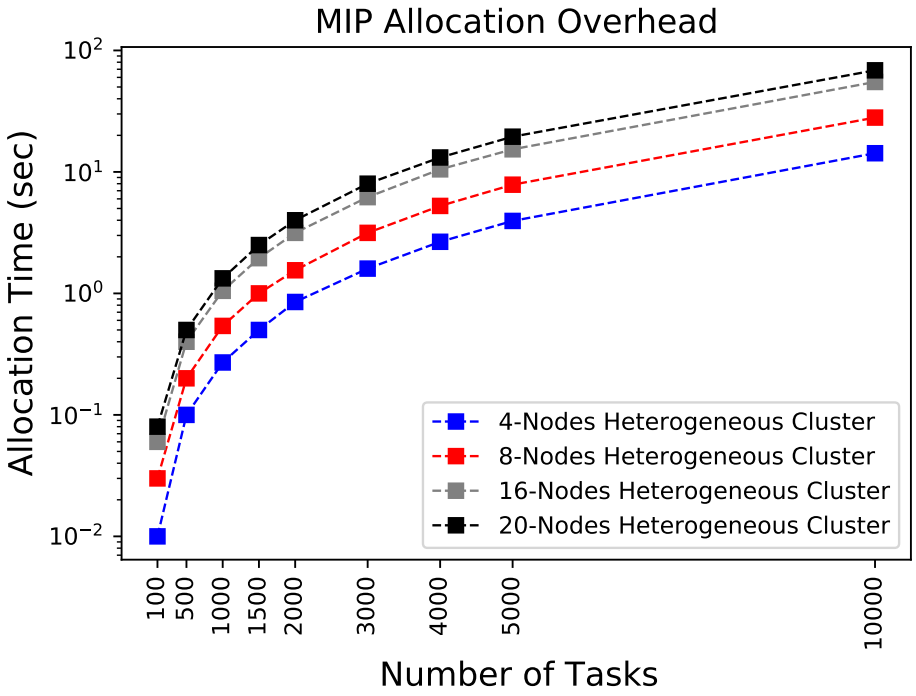


Figure 3.5: A limit study shows the allocation overhead time required by MIP-based allocation technique for different batch sizes and different cluster sizes.

Furthermore, the experimental infrastructure, including the micro-cluster testbed and the benchmarks software, could be used to extend this work and to implement further research. The need for investigating other complex task allocation techniques for edge micro-clusters will be considered in Chapter 4.

3.7 Summary

This chapter characterises the configuration and setup of edge micro-cluster platforms for in context of edge computation. This includes devices and hardware, systems software, and benchmarks software representing real-world workloads. In addition, it presents and characterises an approach to workloads management and task allocation for micro-clusters based on batch arrival of containerised edge workloads. Specifically, various task allocation optimisation techniques were evaluated for batch execution for edge micro-clusters. These optimisation techniques include common optimisation techniques in edge and cloud computing deployment, including heuristic-based, mathematical-based, and metaheuristic-based techniques.

Overall, the micro-cluster design concept has the potential to make edge computing more affordable and more accessible to everyday end-users and edge computation. Such micro-clusters will likely comprise commodity single board computer devices, including but not

limited to Raspberry Pi nodes. Results demonstrate empirically that such an execution paradigm is highly amenable to mathematically optimised workload management using integer programming, with this approach giving optimal or near-optimal makespan performance even when the overhead of integer programming is included.

The evaluation demonstrates that mathematical-based optimisation is both effective and efficient for task allocation optimisation for batch execution for micro-clusters, proving both lightweight overhead computation time and optimal makespan time. However, for large clusters and large batch Sizes, mathematical-based overhead time grows exponentially, indicating that this approach might not be fully appropriate for large micro-clusters. This indicates that alternative optimisation techniques, such as metaheuristic-based, would be more appropriate for large-scale micro-cluster and large batch sizes.

The experiments conducted in this chapter were based on an abstract objective function that estimates task execution times and models the makespan time as the total execution time. However, micro-clusters execute batch workloads in parallel execution mode as task execution is parallelised across cluster nodes. Chapter 4 will extend Chapter 3 by presenting a alternative system objective function based on a linear model that was developed for edge micro-clusters; and will further evaluate more task allocation optimisation techniques.

Chapter 4

A Linear Model for Task Allocation in Edge Micro-Clusters

The resource constraints of micro-cluster platforms mandate a customised task allocation system model. This chapter examines and characterises the performance of nodes in micro-cluster platforms and thereby develops an analytical-based linear model for optimising task allocation for such micro-cluster systems. In addition, a metaheuristic Particle Swarm Optimisation (PSO) technique is proposed and evaluated to minimise the makespan time and the allocation overhead time of heterogeneous workloads in batch execution. The chapter further provides a detailed comparative performance evaluation of the metaheuristic PSO, mixed-integer programming and randomised-based allocation for optimising task allocation in micro-clusters. This chapter is based on the work published and presented at the IEEE Seventh International Conference on Fog and Mobile Edge Computing (FMEC 2022) [22].

4.1 Introduction

The work presented in Chapter 3 characterises micro-cluster functionality in terms of hardware devices and workload management in the context of edge computing. The chapter provides a physical micro-cluster testbed by deploying several Raspberry Pi devices to form a prototype for micro-cluster platforms and explores various optimisation techniques that permit effective workload allocation for such micro-cluster systems based on an abstract system model.

This chapter examines node performance characteristics in micro-cluster platforms and develops a linear-based system model for optimising task allocation. In addition, it incorporates a metaheuristic-based optimisation technique to overcome allocation overhead complexity and generate feasible allocation decisions.

The remainder of this chapter is structured as follows. Section 4.2 outlines micro-cluster specifications and presents a linear-based system model for task allocation in edge micro-cluster systems. Section 4.3 describes the logic behind PSO and demonstrates the adoption of this metaheuristic-based technique for task allocation. Section 4.4 evaluates the performance of the PSO-based technique in comparison to other techniques. Finally, Section 4.5 provides the conclusion.

4.2 Linear Model for Task Allocation in Micro-Clusters

This section characterises node performance in micro-cluster platforms for executing concurrent representative benchmarks and presents a linear-based model for task allocation specifically for such compact platforms. The individual node performance is examined and evaluated in Section 4.2.1, and the linear-based system model is thereafter presented in Section 4.2.2.

4.2.1 Individual Node Performance Characterisation

The computing resources of nodes in edge micro-cluster systems differ from those deployed in centralised cloud data centres. Nodes in micro-clusters are typically integrated with tiny to moderate computing resources in terms of processors, memory, and storage, built-in small-size boards. For example, Raspberry Pi devices are integrated with ARM-based processors with clock speed ranging from 0.9 GHz to 1.8 GHz and memory size ranging from 1GB to 8GB. In contrast, cloud data centre providers deploy high performance server machines capable of delivering unlimited resources upon users requirements. Table 4.1 presents exemplar hardware specifications of nodes in edge micro-clusters and nodes in cloud data centres.

Therefore, in order to develop effective task allocation techniques for micro-cluster platforms, it is necessary to examine nodes performance characteristics for executing concurrent tasks by using representative benchmarks. Existing work benchmarked similar edge devices using micro-benchmark tools like *Sysbench* and *Linpack* [49], [104]. However, micro-benchmarks might not be sufficient to capture meaningful performance of micro-cluster nodes. We therefore use real-world holistic benchmarks to represent a balanced mixture of micro-behaviours for edge applications and capture relevant node performance metrics.

Below are experiments to characterise the performance of typical nodes used in micro-clusters when executing edge representative workloads. The main objective of these experiments is to capture performance characteristics of nodes when executing concurrent realistic workloads. This is necessary to model a realistic system objective function for micro-cluster platforms.

Table 4.1: Exemplar hardware specification of typical nodes deployed in edge-micro clusters and in centralised cloud data centres

Machine	CPU	Cores	Memory	OS	Virtualisation Technology
RPi 3B	ARM Cortex-A72 with 1.2GHz	4	1GB	RaspberryPi OS	Containers (e.g.,Docker)
RPi 4B	ARM Cortex-A72 with 1.5GHz	4	4GB	RaspberryPi OS	Containers (e.g.,Docker)
Amazon EC2	Intel core i7 with 3.2 GHz	2	8GB	Linux	VMs (e.g.,Xen)

The performance of different nodes are examined for executing concurrent workloads. Different generations of Raspberry Pi nodes that represent typical nodes heterogeneity in edge micro-clusters are benchmarked using concurrent and heterogeneous workloads from three different applications, image-detection (Yolo), audio-text converting (PocketSphinx), and audio-text synchronisation (Aeneas). These applications are drawn from extensions of the DeFog benchmark suite [19].

Figure 4.1 shows the performance trend of micro-cluster nodes for concurrent task execution. The graphs illustrate that node performance is correlated with the number of concurrent tasks, demonstrating linear increases in task execution times for all benchmarks in all nodes in the micro-cluster testbed. The observed linear increases in the execution time of concurrent tasks might be attributed to task contention, such as for the limited computational resource, memory bandwidth, or file-system contention. As mentioned above, nodes in micro-clusters are inherently resource-constrained. These limitations in nodes' hardware capabilities slow the execution of concurrent tasks, leading to increased execution time as concurrent tasks compete for limited resources. When a node executes multiple tasks concurrently, the resources like CPU and memory are shared among tasks, decreasing the available resources for each task.

The experiments, in addition, reveal node capacity limits for handling concurrent tasks. For example, Raspberry Pi devices RPi 2 model B, RPi 3 model B, and RPi 3 model B+ were not able to accommodate multiple concurrent tasks that require intensive-processing (i.e., image detection and audio-text converting) where some tasks fail when sending more than four concurrent tasks for image-processing workloads. Furthermore, those nodes were not responsive for more than eight concurrent tasks. The node performance limits are defined by node capacity constraints in system model as explain in the following section. Analysing task failure data is beyond the scope of this research and is subject to future work.

Based on results from the pre-experimental work mentioned above, a linear-based task allocation for edge micro-cluster platforms is developed, as presented in the following subsection 4.2.2.

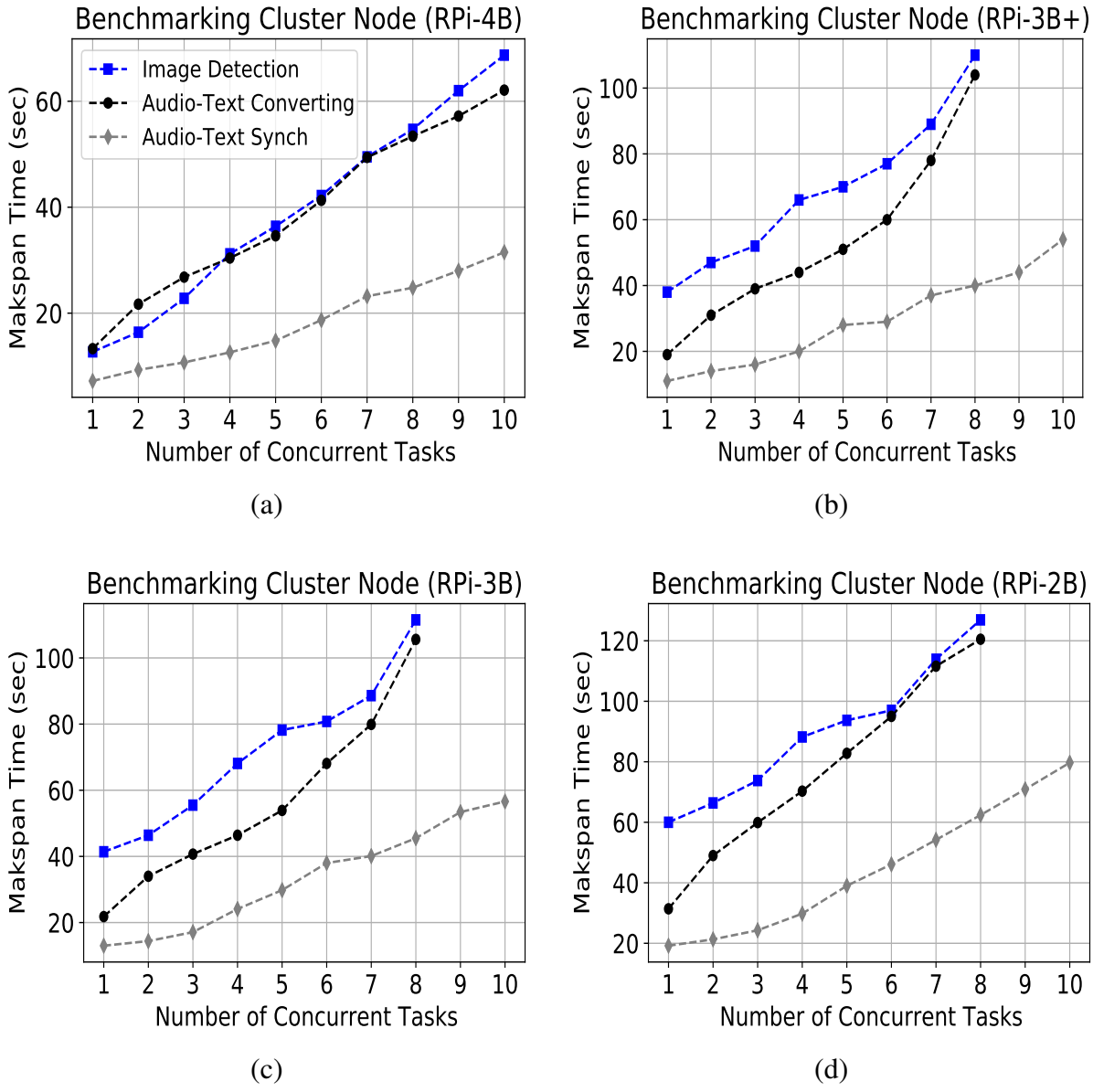


Figure 4.1: Line graphs showing the performance trends of micro-cluster nodes for processing concurrent workloads from different applications workloads.

4.2.2 Formulation of the Linear Model for Task Allocation

Edge micro-clusters represent heterogeneous edge platforms. A micro-cluster comprises several physical nodes equipped with different resource capabilities in terms of processors and memory. As mentioned earlier, edge micro-clusters are assumed to receive and process workloads in a batch-execution mode, in which different tasks are offloaded from various applications and aggregated into batches. The challenge is to effectively allocate tasks to micro-cluster nodes such that the makespan time of the entire batch is minimised and nodes resources are effectively utilised.

The objective is to effectively allocate tasks to the micro-cluster nodes in order to minimise the total execution time for a full batch of tasks (i.e. the *makespan* time) by effectively allocating tasks to nodes. Equation 4.1 describes the system objective function developed based on observations from the nodes performance experiments presented in Section 4.2.1. The system model is therefore formulated as a linear model using the standard form (y = mx + c).

Equation 4.1 evaluates candidate solutions provided by allocation techniques, where T is the makespan time, m_n is a constant represents the scaling reference of node n, $numTask_n$ represents the total number of tasks assigned to node n, and c_n is a constant represents the y-intercept of node n. As the cluster processes tasks in parallel execution, the makespan time is effectively the maximum execution time across all nodes.

Table 4.2 presents the values of the gradients (m) and the y-intercepts (c) for different nodes and different workloads. These data are derived based on analysis from node individual performance experiments discussed in Section 4.2.1. The values of the gradients (m) and y-intercepts (c) are calculated using the $slope$ and $intercept$ functions in Microsoft Excel to compute the linear regression from the data obtained from Figures 4.1. One node from each generation is evaluated and the obtained values are then generalised for the remaining nodes from the same generation. Note that a more sophisticated model could be deployed for complex use cases, but this was not necessary in these experiments.

$$T = \max_{n \in N} \left(m_n * numTask_n + c_n \right) \tag{4.1}$$

where:

T: denotes Makespan Time

m_n: is a constant value for the gradient of node n

$numTask_n$: is number of tasks assigned to node n

c_n: is a constant value for y-intercept of node n

As a micro-cluster is a heterogeneous platform and nodes have different capabilities, two system constraints are introduced to model these features. First, the node capacity constraint is represented in Equation 4.2, in which node n has a defined capacity and should not receive more than its defined capacity. cap_n is a non-negative integer value that represents node n capacity, x_{nt} is a binary matrix of $0, 1$ values indicating whether task t is allocated to node n. N and T represent a set of cluster nodes and a set of tasks, respectively. In addition, constraint 2 is represented in Equation 4.3 to make sure that each task t is assigned to only one node n and avoid assigning the same task to multiple nodes.

$$\left(\sum_{t \in T} x_{nt}\right) <= cap_n \ , \ \forall \ n \in N \tag{4.2}$$

$$\left(\sum_{n \in N} x_{nt}\right) == 1 \ , \ \forall \ t \in T \tag{4.3}$$

Table 4.2: Values of the gradients (m) and y-intercepts (c)

Workload	Image Detection		Audio-Text Converting		Audio-Text Synch	
	m-value	c-value	m-value	c-value	m-value	c-value
Node RPi 2B	8.6	50.21	13.55	18.10	7.07	5.66
Node RPi 3B	8.73	33.25	10.79	7.78	5.27	4.2
Node RPi 3B+	8.94	27.64	11.21	3.53	4.66	3.66
Node RPi 4B	6.54	2.4	5.06	10.06	2.49	5.4

4.3 Task Allocation Using Particle Swarm Optimisation Metaheuristic

This section describes the logic behind the Particle Swarm Optimisation (PSO) metaheuristic and provides the representation process demonstrating the mapping of PSO particles to micro-cluster nodes and tasks.

4.3.1 PSO Logic

Particle Swarm Optimisation (PSO) is a well-known metaheuristic-based optimisation technique inspired by the collaboration of bird swarms. PSO is utilised in substantial cloud data centres to optimise various resource management problems, such as issues related to optimising virtual machine placement or minimising energy and monetary cost [98] [99] [106].

In PSO, various particles are randomly initialised to discover the search space. The PSO system dynamic iteratively controls particles to optimise candidate solutions and gradually

converges towards near-optimal solutions for several iterations or upon meeting a pre-defined termination condition [106], [107].

As illustrated below, velocity, position, and hyperparameters control PSO particles during the search process. Velocity V_i^t and particle position X_i^t are initialised randomly at the start of the search process. Thereafter, velocity factor (Equation 4.4) determines the distance to next position of a particle. Specifically, velocity V_i^{t+1} computes the distance from the current position X_i^t and the new position X_i^{t+1} using information from current iteration and with the help of hyperparameter values, defined in next paragraph. This prompts particles to re-locate in direction of the best position. Furthermore, particle best positions $PBest_i$ and global best positions of the swarm $GBest$ are frequently updated and compared during the search process to maintain the best found position.

PSO hyperparameters direct particles in the search for a global optimum. The inertia weight parameter W is used to balance between exploration and exploitation, where a high W value increases exploration, and a small value increases exploitation. Equation 4.6 balances exploration and exploitation by assigning a large W value at the beginning of the search process and gradually decreasing it throughout the search process. *WMax* and *WMin* are constants representing upper and lower bounds, respectively, t is the current iteration, and $MaxIter$ is the total number of iterations. Equation 4.6 allows particles to discover the search space and then converge towards the best-found solution. Typically W value ranges from 0.4 to 0.9. In experiments, $W = 0.79$ is used because a high value led to premature convergence, whereas a small value directed the particles to locally optimal solutions.

Furthermore, the inertia weight W, the acceleration coefficients $c1$ and $c2$, and the random variables $r1$ and $r2$ help direct the velocity of the particles. Coefficient components $c1$ and $c2$ also impact exploitation and exploration in PSO. The individual component $c1$ is primarily responsible for increasing exploration, while the coefficient social component $c2$ is used to increase exploitation. Balancing between $c1$ and $c2$ helps PSO particles to converge toward near-optimal solutions. The random variables $r1$ and $r2$ are used to add randomness to particles in each iteration. Algorithm 5 represents the PSO-based task allocation. Table 4.3 summarises the hyperparameters used in PSO and the assigned values employed in this implementation.

$$V_i^{t+1} = W * V_i^t + c1 * r1 * (PBest_i - X_i^t) \\ +c2 * r2 * (GBest - X_i^t) \tag{4.4}$$

$$X_i^{t+1} = X_i^t + V_i^{t+1} \tag{4.5}$$

$$W = WMax - t((WMax - WMin)/MaxIter) \tag{4.6}$$

4.3.2 Representation Process

This section demonstrates the representation process by which PSO particles are converted to represent tasks and nodes. In PSO-based task allocation representation, PSO particles represent candidate allocation solutions. A PSO particle is effectively a function that maps tasks to micro-cluster nodes. PSO Particles are represented by $length$ and $range$. The $length$ of particles represents the total number of tasks to be allocated, and the $range$ of particles represents the total number of nodes in a micro-cluster.

For example, to allocate a batch of 10 tasks on an eight node micro-cluster. In PSO representation, the length of particles represents the total number of tasks, i.e., 10. The range of particles equals 8 (i.e., the number of nodes in a micro-cluster) and is randomly set between 0 to 7 as the edge micro-cluster contains eight different nodes.

Table 4.4 presents illustrative examples of the representation process demonstrating the mapping of PSO particles to tasks and nodes. The examples showcase sample allocation solutions for 10 tasks on an edge micro-cluster consisting of eight nodes. For example, in solution 1, the PSO particle is represented as array [7, 7, 7, 4, 7, 5, 7, 3, 7, 6]. In representation process, tasks $t1$, $t2$, $t3$, $t5$, $t7$, $t9$ are allocated to node $n7$; task $t4$ is allocated to node $n4$, task $t6$ is allocated to node $n5$. For heterogeneous workloads, tasks are aggregated from various workload types into one batch. In this example, the image-processing represent 4 tasks, audio-text synchronization represent 3 tasks, and audio-text converting represents 3 tasks. The estimated makespan is calculated using Equation 4.1.

Overall, the PSO system iteratively discovers the solution space during the search process. Particles are randomly initialised and gradually directed toward the best-found solution by updating the position using PSO velocity, position, and hyperparameters. PSO demonstrates fast convergence and has the ability to avoid being trapped in local optima. Furthermore, PSO is easy to implement without requiring the complexity of mathematical optimisation, which yields PSO as a promising lightweight optimisation technique for edge computing deployment.

The following section 4.4 evaluates and compares the performance of PSO-based optimisation technique with a mathematical-based technique and a randomised-based technique for optimising task allocation in edge micro-cluster systems.

Algorithm 5 PSO-based Task Allocation

Initialise Particles X_i^t Randomly;
Initialise Velocity V_i^t Randomly ;
Initialise hyperparamters using table 4.3;
for *t in Iteration* **do**
  for *i in Particles* **do**
    Compute $fitness$ using (Equation 4.1);
    Compute Velocity V_i^{t+1} using (Equation 4.4);
    Compute Position X_i^{t+1} using (Equation 4.5);
    #Update Best Position (PBest)
    if *fitness* $< PBest_i$ **then**
      $PBest_i$=$fitness$
      X_i=$PBest_i$
    end
    # Update Global Position (GBest)
    if *fitness* $<$ *GBest* **then**
      $GBest$=$fitness$
      X_i=$GBest$
    end
  end
end
return Best Allocation Decision

Table 4.3: PSO hyperparameters description and related values.

Parameters	**Description**	**Assigned Value**
W	inertia weight	0.79
c1	coefficient for individual component	1.49
c2	coefficient for social component	1.49
r1	random coefficient	0-1
r2	random coefficient	0-1
Particles	Number of particles in each iteration	50
Iteration	Number of generations	100

Table 4.4: Exemplar workloads mapping on an eight-node micro-cluster with estimated makespan time calculated by the linear model. Heterogeneous workload is represented as: Image Detection (40%), Audio-Text Synch (30%), and Audio-Text Converting (30%).

Workloads	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	**Est Makespan**
Image-Detection	7	7	7	4	7	5	7	3	7	6	40
Audio-Text Synch	6	7	7	3	3	7	6	5	4	4	15
Audio-Text Converting	6	7	7	4	5	7	3	7	4	3	31
Heterogeneous	7	0	4	7	3	5	7	3	6	7	25

4.4 Evaluation

This section evaluates the performance of the proposed allocation techniques for optimising task allocation for micro-cluster systems based on a linear model system.

The experiments were carried out utilising both modelling and empirical evaluations. Specifically, the optimisation techniques were developed and executed via modelling on a MacBook Pro machine (2.4 GHz dual-core i5 and 4GB RAM), while the performance evaluations were empirically analysed using a configured physical micro-cluster testbed and representative workload benchmarks. The allocation techniques effectively should be run on a cluster head node capable of performing the orchestration and management aspects; however, due to dependency issues in Raspberry Pi devices, the optimisation techniques are implemented via modelling on a MacBook machine connected to the same LAN network of micro-cluster using a networking switch. The tasks are then allocated to cluster nodes according to the obtained allocation decisions.

As mentioned earlier, the majority of work in the literature follows simulation-based experiments using simulation tools, such as CloudSim or MATLAB. An empirical evaluation approach using a physical micro-cluster testbed is implemented to capture realistic performance.

4.4.1 Optimisation Techniques

To evaluate the related performance, the performance of the PSO metaheuristic allocation technique for optimising task allocation in edge micro-cluster platforms is compared with the performance of the Mixed-Integer Programming and a random-based technique. The PSO-based optimisation and the random-based techniques were implemented in Python, while the mixed-integer programming-based method was implemented using Google open source software suite for optimisation, OR-Tool [105].

Similar to the work in Chapter 3, the Mixed-Integer Programming technique allocates tasks to cluster nodes according to the optimal found allocation. It estimates batches makespan time using values derived from Table 4.2 and takes into account the system's constraints defined in Equation 4.2 and Equation 4.3. This technique is implemented using Google open source software suite for optimisation, OR-Tool [105]. The random-based allocation method is a straightforward baseline allocation technique where workloads are randomly allocated to cluster nodes without knowledge about nodes' capabilities or workloads types.

4.4.2 Performance Evaluation

Similar to performance evaluation in Chapter 3, two edge-relevant performance metrics were considered to capture the related performance. First is the *allocation overhead time*, which is the runtime required by an allocation technique to complete the required computation and produce an allocation decision. The second metric is *makespan time*, which reflects the effectiveness of the optimisation techniques and quality of the solutions. Section 4.4.2.2 and Section 4.4.2.3 provide further details on these metrics.

4.4.2.1 Accuracy of the Linear Model

This section evaluates the linear model. The accuracy of the proposed linear model is empirically validated. The estimated makespan time generated by the linear model is compared to the actual makespan time on an eight node heterogeneous micro-cluster testbed using different application benchmarks.

Figure 4.2 presents results for different applications. The x-axis represents different batch sizes, and the y-axis represents makespan time in seconds. Overall, results demonstrate that the linear model is useful to estimate the required makespan time for batch execution in edge micro-cluster platforms.

There are minor underestimations in the estimated makespan times. This might be attributed to the networking contention when transferring multiple task inputs. In addition, the parameters of the linear model, i.e., the gradients (m) and y-intercepts (c), are obtained by evaluating one device from each generation and the values are then generalised for the other devices from the same generation in the testbed. Even though devices are from the same generation, their performance might vary slightly.

4.4.2.2 Allocation Overhead Time

The allocation overhead time represents runtimes of allocation techniques to complete computations and converge to produce a task allocation decision. Allocation overhead time is a critical metric in edge computing because orchestration techniques run on resource-constrained devices like single board computers [62]. In the context of edge micro-cluster platforms, allocation techniques are anticipated to be deployed on one cluster node representing a cluster head node.

The allocation overhead times of the optimisation techniques are evaluated using two scenarios, homogeneous workloads and heterogeneous workloads. For the first scenario, tasks are all the same types, representing homogeneous batches to be effectively allocated to the

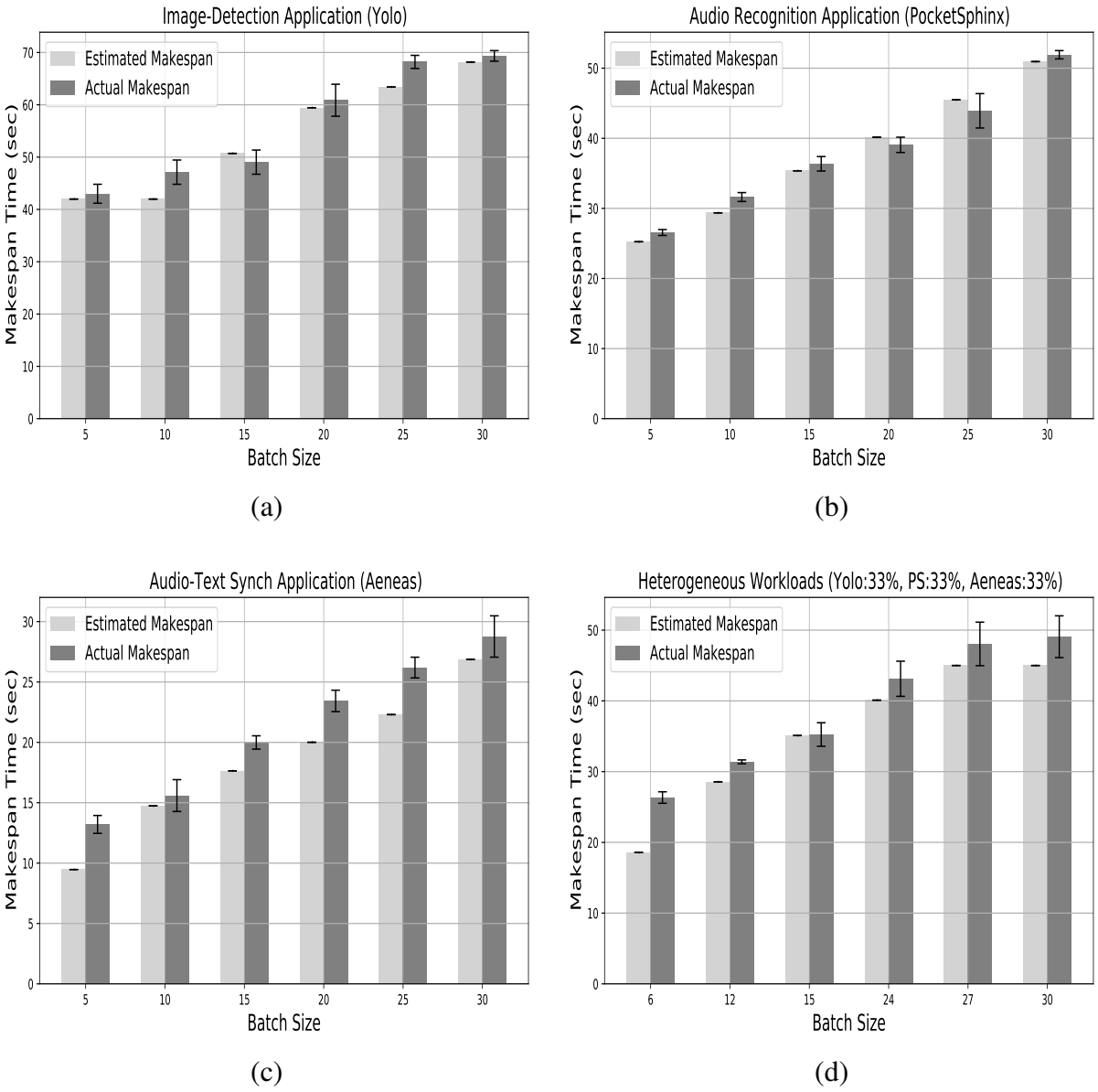


Figure 4.2: Comparison between the estimated makespan time calculated by the linear-based model and the actual makespan time (mean of 15 runs) on a physical edge micro-cluster for different workloads and batch sizes.

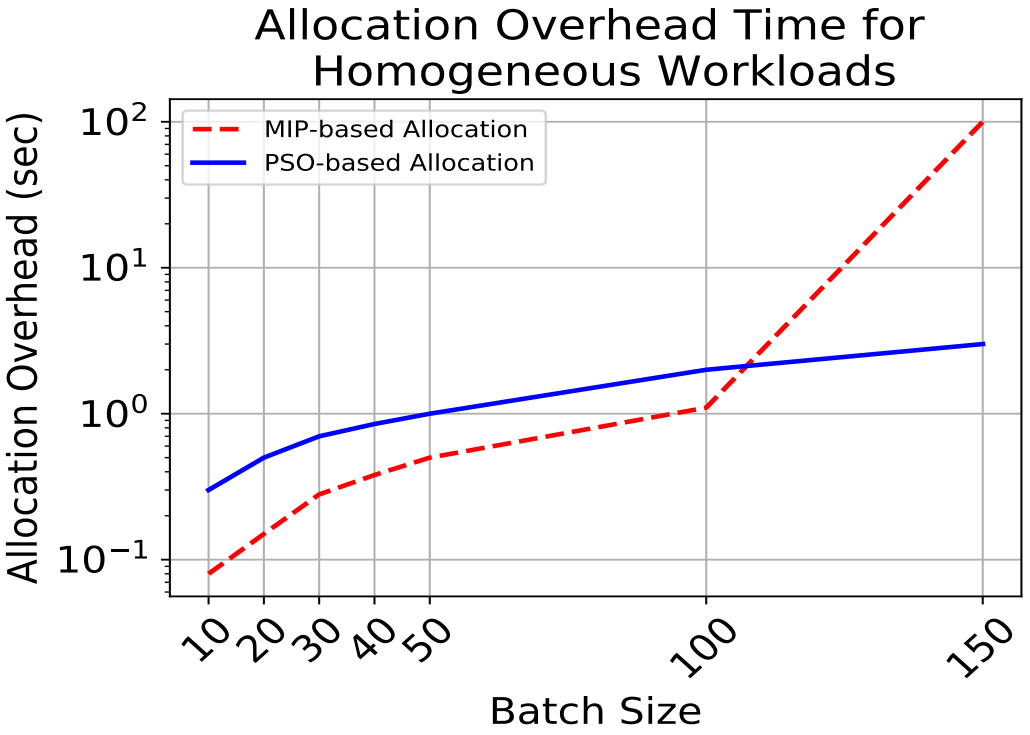
micro-cluster. In this scenario, tasks are generated from different applications and then classified into separate batches based on their task types. For the second scenario, batches comprise heterogeneous workloads, where tasks from different applications are aggregated in heterogeneous batches and thus each batch contains a mix of tasks.

Figure 4.3 presents a comparison of allocation overhead times required for homogeneous workloads and heterogeneous workloads, respectively. The figures overall show practical crossover graphs indicating conventional overlapping between different task allocation optimisation techniques for edge micro-cluster platforms.

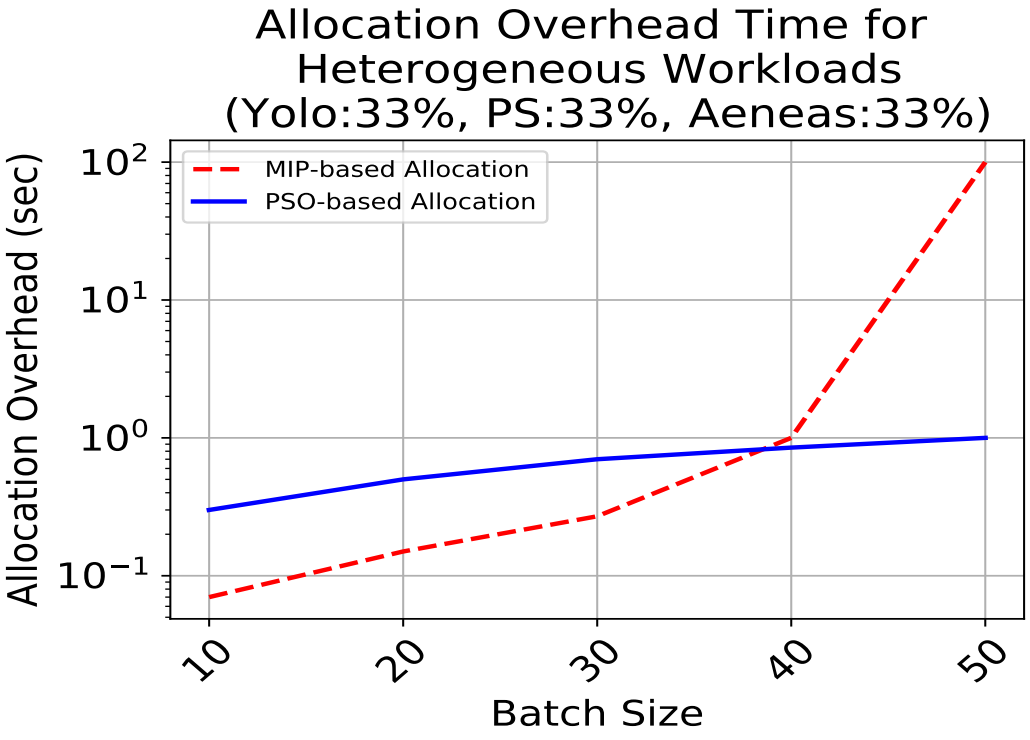
As seen in the graphs in Figure 4.3, the mixed-integer allocation technique demonstrate low overhead time for homogeneous batches. It converges very quickly and obtains the optimal allocation solution in less than 1 sec for up to 100 homogeneous tasks. The overhead time grows exponentially and dominates the total execution time for large homogeneous batches. Furthermore, the problem becomes more complex for the heterogeneous batches given the increased number of constraints, which makes exponentially complex MIP-based allocation an unfeasible technique for heterogeneous workloads.

On the other hand, the PSO-based technique demonstrates linear growth for both scenarios and achieves near-optimal solution quality for homogeneous and heterogeneous batches. The PSO-based technique shows faster computation time than the mixed-integer programming technique and effectively produces near-optimal solutions for large-scale batches.

The allocation overhead time for the random-based technique never exceeds a few msec and therefore was excluded from the above comparison for the poor performance results in terms of solution quality.



(a)



(b)

Figure 4.3: Crossover graphs representing the allocation overhead time required by the exponential complexity MIP-based and linear complexity PSO-based for different batch sizes and workload types. The Random-based allocation is excluded from this comparison as its overhead time is always minimal and never exceeds a few msec.

4.4.2.3 Solution Quality

The proposed task allocation techniques optimise task allocation in edge micro-cluster platforms. This could be achieved by identifying appropriate allocation solutions that can effectively distribute tasks to cluster nodes to minimise the makespan time of tasks in batch execution.

Similar to the definition of makespan time provided in Chapter 3, the makespan time is the total execution time required for micro-cluster systems to complete workloads in batch execution. As micro-clusters process workloads in parallel execution across all nodes, the makespan time effectively is the execution time required by the cluster node that requires the maximum execution time to complete its allocated workloads. The makespan time includes the time required by the allocation techniques to generate an allocation decision (i.e., allocation overhead time), the time required for offloading tasks to the micro-cluster, the time required for the nodes to process allocated workloads, and the time required to receive results back from micro-cluster platforms.

Figures 4.4, 4.5, 4.6, and 4.7 present the quality of solutions generated by different allocation techniques and therefore demonstrate their effectiveness for edge micro-cluster systems. These results show the makespan time required for the micro-cluster testbed to process different workloads. The experiments evaluate the effectiveness of different task allocation techniques, namely: PSO-based allocation, mixed integer-based allocation, and random-based allocation. The evaluation examined different batches of homogeneous and heterogeneous workloads to be allocated on a micro-cluster.

As mentioned in Section 4.4.2.2, two scenarios were considered, i,e., homogeneous batches and heterogeneous batches. For homogeneous workloads, tasks are offloaded to a micro-cluster platform from different applications and IoT appliances and classified into different batches according to task types. Results indicate that mixed-integer programming-based and PSO-based techniques both can achieve comparable solution quality, and the allocation overhead times do not significantly impact the total makespan time as shown in Figure 4.3. These findings overall indicate that the PSO-based technique is effective and performs as well as mixed-integer programming in producing optimal or near-optimal allocation solutions.

For the heterogeneous workloads, three heterogeneous batches were examined. Workloads were aggregated from different applications into batches. Batches descriptions are as follows: The image-detection workloads constitute (33%), audio-text synchronisation workloads constitute (33%), and audio-text converting workloads constitute (33%). Figure 4.7 presents the results. The analysis indicates the performance of mixed integer programming outperforms the performance of both PSO-based and random-based techniques for small-scale heterogeneous batches. However, the makespan time for large batch size (i.e., batch size 45 tasks) is influenced by the allocation overhead time for MIP-based allocation tech-

nique as the experiments reveal that MIP-based technique overhead time for heterogeneous workloads is growing exponentially and significantly influences the makespan time. It requires around 18 seconds to produce an allocation decision for large batches.

On the other hand, the overhead time of the PSO-based does not significantly impact the makespan time as it grows linearly in relation to the batch sizes. These findings indicate that the PSO-based technique is viable and efficient for edge micro-cluster platforms and maintain both near-optimal allocation decisions and minimal overhead time.

The results further illustrate that the random-based allocation is impractical and ineffective for edge micro-cluster systems because the makespan times are high for both homogeneous and heterogeneous batches compared to other techniques; even the allocation overhead time remains constant and never exceeds a few msec. The experiments reveal task failure, specifically when nodes exceeded their capacities. The analysis and measurements for task failures are not considered in this research and might involve techniques for load balancing or task migration between the cluster nodes.

For large problem sizes (100 tasks of homogeneous workloads and 50 tasks of heterogeneous workloads), we found that the PSO-based optimisation yields near-optimal solutions in comparison with the optimal solutions generated by MIP-based allocations (in terms of makespan). In a limits study, we found that the quality of solutions generated by the PSO-based allocation technique are within 4.84% for image-detection, 11.68% for audio-text converting, 2% for audio-text synchronisation, and 20% of heterogeneous workloads of the optimal solutions generated by MIP-based allocation. These results indicate that PSO-based allocation is able to generate effective and near-optimal solutions for large and complex problem sizes.

Overall, analysis indicates that a necessary balance between obtaining high-quality solutions and securing much faster ones is essential. It might be necessary to compromise between solution quality and computational complexity in order to obtain faster allocation decisions. This applies when the allocation overhead to find high-quality allocation solutions exceeds the expected total computation time of the batch, particularly for large-scale clusters and large batch sizes.

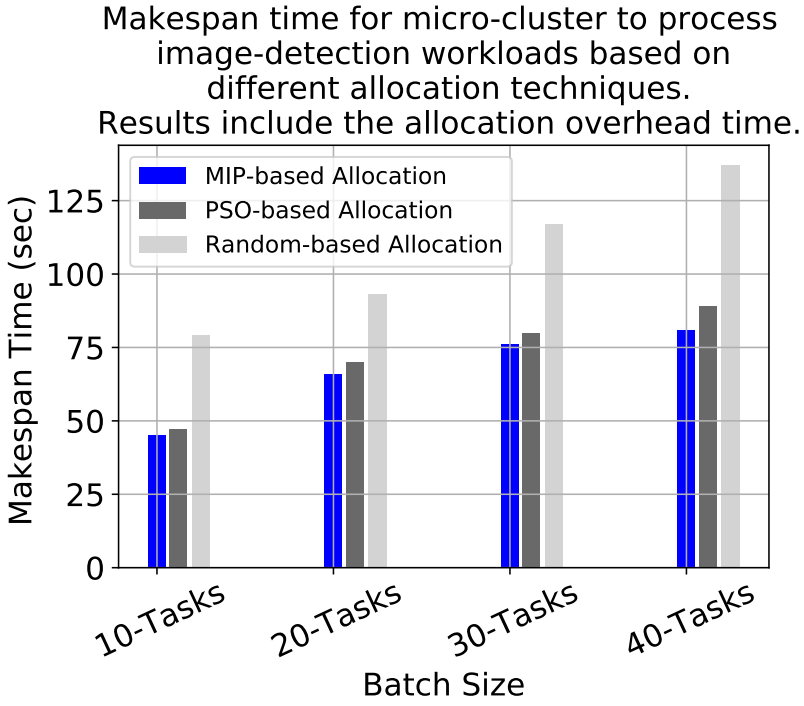


Figure 4.4: Quality of solutions obtained by different allocation techniques for image-detection workloads.

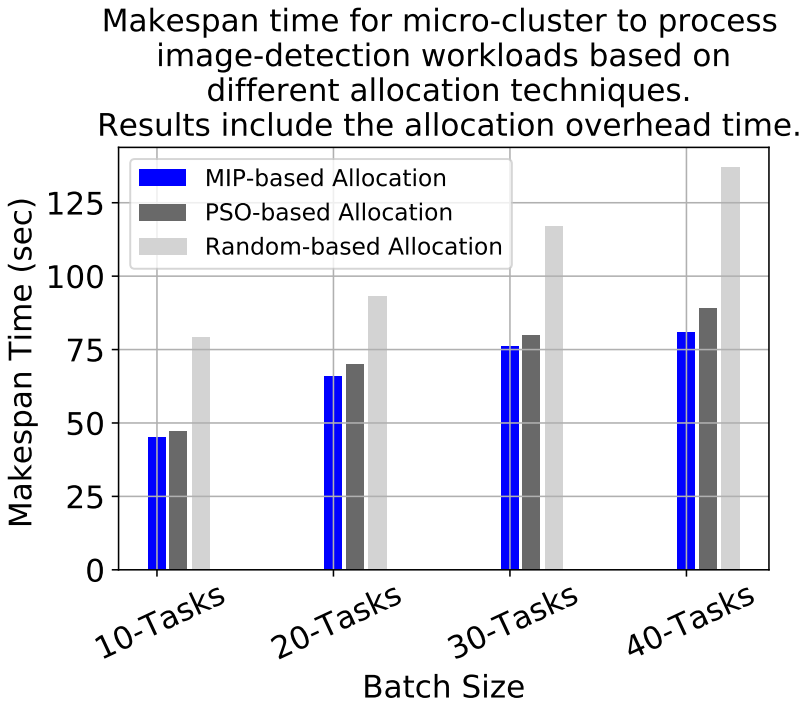


Figure 4.5: Quality of solutions obtained by different allocation techniques for audio-recognition workloads

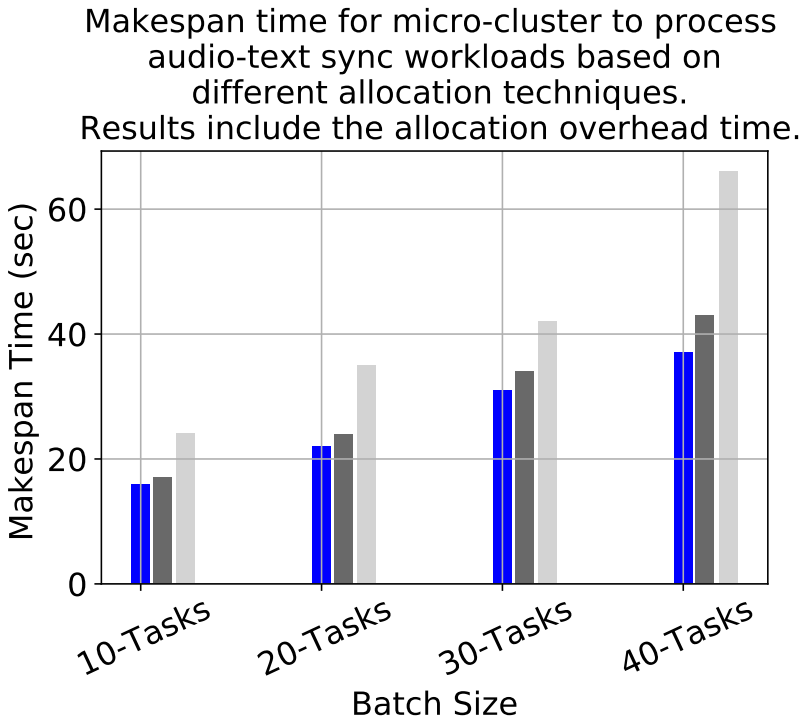


Figure 4.6: Quality of solutions obtained by different allocation techniques for audio-text synchronisation workloads

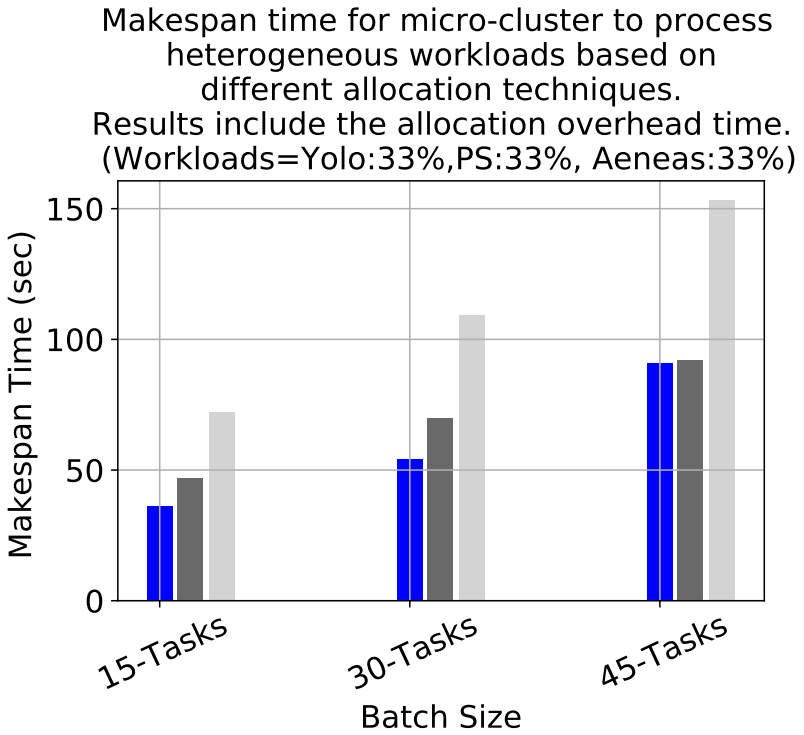


Figure 4.7: Quality of solutions obtained by different allocation techniques for heterogeneous workloads

4.5 Summary

This chapter extends the work presented in Chapter 3 with two fundamental contributions. First, it develops an analytical performance model for optimising task allocation in edge micro-cluster platforms. Secondly, it proposes metaheuristic optimisation based on the Particle Swarm Optimisation technique for optimising task allocation for micro-clusters.

The chapter investigates micro-cluster systems by specifically providing the following fundamental contributions. First, the performance characteristics of micro-cluster nodes are empirically examined and evaluated for executing concurrent representative workloads. The evaluation reveals the performance trends of nodes for processing various workload types. Second, a linear-based model for batch execution is developed for micro-cluster systems. Third, a task allocation optimisation based on metaheuristic optimisation is proposed and evaluated to optimise workload allocation for micro-clusters. Finally, a comparative evaluation of different optimisation techniques for optimising heterogeneous workloads in batch execution for edge micro-clusters in terms of effectiveness and efficiency is reported.

Evaluations reflect the performance of metaheuristics-based optimisation and compare it with mathematical-based allocation and randomised-based allocation. The evaluation provides an insightful overlapping indicating typical indicative limits for each optimisation technique for edge micro-clusters.

Results indicate that PSO-based and MIP-based techniques both achieve comparable solutions for small-scale micro-clusters platforms. Moreover, the PSO-based optimisation technique scales better and finds effective solutions for large-scale micro-clusters, with linear overhead time compared to the exponential overhead of the MIP-based technique for large-scale clusters. Accordingly, we argue that mathematical-based optimisation is most suitable for task allocation for limited-scale micro-clusters, while PSO-based optimisation demonstrates lightweight and effectiveness for large-scale edge micro-cluster platforms.

The following chapter will explore developing and evaluating a multi-objective optimisation framework to optimise other edge-relevant performance metrics for edge micro-clusters.

Chapter 5

Multi-Objective Optimisation for Edge Micro-Clusters

The previous technical chapters focus on optimising task execution in edge micro-cluster platforms by effectively allocating tasks and minimising the makespan time of applications in batch execution mode. This chapter extends earlier work by investigating and optimising energy consumption in micro-clusters. It first characterises per-node power consumption and presents an analytical model that can effectively predict the energy consumption required for performing edge-relevant workloads. Following on from this, a multi-objective optimisation model is developed to optimise two edge-relevant performance metrics of energy consumption and makespan time for workloads.

5.1 Introduction

Edge micro-clusters can be effective deployment solutions to extend computing resources and deliver IoT applications to remote environments located at the edge of the network. For example, micro-cluster platforms can be used in agricultural settings to deploy smart farming management systems that provide computation services like monitoring and executing application workloads locally near IoT environmental sensors.

However, conventional power supply networks might not be always available in many remote environments, which may represent a substantial barrier for IoT-based services. Thus, micro-clusters might utilise alternative power systems, such as batteries and renewable energy. Therefore, efficient power usage is essential in such deployment settings to enable delivering applications and, in turn, minimise energy consumption.

In light of the aforementioned deployment conditions, micro-cluster resources need to be efficiently utilised to reduce energy consumption and ultimately extend their alternative power

systems. In order to preserve power and cope with application requirements, an edge resource management mechanism is assumed to perform provisioning and scaling requirements in micro-clusters, where the system manager can turn nodes on and off to cope with workload dynamics. Specifically, inactive nodes can be turned off to save power. Nodes can be dynamically scaled up or down by turning them on or off to meet application requirements. Nodes like Single Board Computers (e.g., Raspberry Pi devices) can quickly boot in ~ 20 seconds which might not substantially affect the overall system response time in batch execution scenarios.

Work on power consumption for Single Board Computers has been presented in [49], where the author conducted experiments to examine power characteristics for several devices, including Raspberry Pi and Odroid. In addition, the power consumption characteristics of edge computer clusters were investigated in [7] using a simple load balancing technique. Furthermore, authors in [46] developed energy monitoring power consumption for SBC clusters to support edge cluster deployment. However, with the ongoing research efforts on edge computing, no much work considers multi-objective optimisation for edge micro-cluster systems for the edge-relevant performance metrics of energy consumption and execution time.

In this chapter, the energy consumption for edge micro-cluster systems is investigated. Section 5.2 provides preliminary experiments to investigate and identify energy consumption characteristics for micro-cluster platforms. Section 5.3 develops an energy consumption analytical model that can effectively predict the energy consumption required for executing workloads in micro-cluster platforms. Section 5.4 develops a multi-objective optimisation model that can optimise makespan time and energy consumption for micro-clusters, where a Pareto-Front optimisation is identified to balance between two competing objectives. Finally, Section 5.5 provides conclusions and outlook for potential future directions.

5.2 Preliminaries

This section introduces preliminary experiments that were conducted to identify and characterise the power consumption of micro-cluster platforms. The experiments were designed to examine nodes' power consumption characteristics under two fundamental operation states, idle state and active state. The active state represents when nodes are in full operation where node CPU usage is 100%. The idle state represents when micro-cluster nodes are on but not being used by any application to execute computational tasks.

Motivated by [49], the active state is represented by stressing micro-cluster's nodes CPU resources. The $Sysbench$ system benchmark was used to model micro-cluster active states by stressing nodes CPU to 100%. $Sysbench$ is basically a system software tool used to test CPU to calculate prime numbers. On the other hand, Linux $sleep$ command was used to model

micro-cluster idle state. Linux *sleep* command was used to set nodes to idle state for 600 sec. Power consumption was empirically measured for each node individually and for the entire micro-cluster by using a physical power meter monitoring device directly connected to the micro-cluster testbed. The cluster testbed is isolated from other equipment, such as a networking switch.

Figure 5.1 presents the preliminary power consumption of the micro-cluster nodes in active and idle states. Specifically, analysis shows that node type Raspberry Pi 2B require 3.2 Watts, Raspberry Pi 3B requires 4.8 Watts, Raspberry Pi 3B+ require 5.8 Watts, and Raspberry Pi 4B requires 6.2 Watts in full active states. The micro-cluster testbed overall consumes $\sim$ 36 Watts in full active state.

Furthermore, the preliminary experiments reveal that nodes in idle state consume a considerable amount of energy. Specifically, nodes type Raspberry Pi 2B and Raspberry Pi 3B consume 2.2 Watts in idle states, while Raspberry Pi 3B+ and Raspberry Pi 4B require 2.5 Watts and 3.3 Watts, respectively. The micro-cluster overall consumes $\sim$ 18 Watts in a full idle state. Numbers indicate that the power consumption of the micro-cluster testbed in full operation is increased by 100%. Table 5.1 presents raw data identifying micro-cluster nodes' hardware specifications and power consumption data in detail.

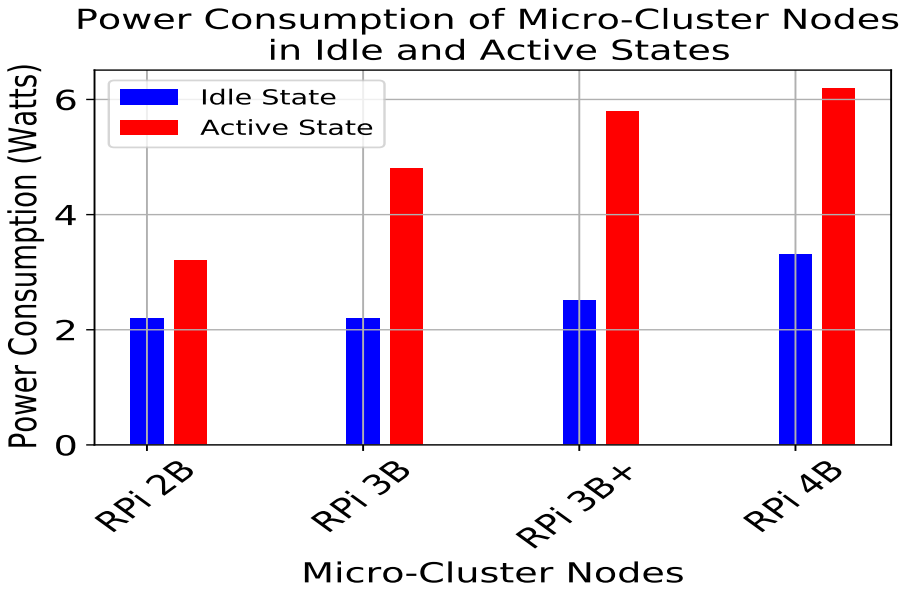


Figure 5.1: Characteristics of power consumption of micro-cluster individual nodes in idle and active states. The active state represents nodes' power consumption for 100% CPU.

Table 5.1: Micro-cluster nodes hardware specifications and power consumption characteristics in idle and active states.

Compute Node	**CPU Speed**	**Memory**	**Power Consumption in Idle state(Watts)**	**Power Consumption in Active State (Watts)**
Raspberry Pi 2 model B (3 nodes)	900 MHz	1 GB	2.2	3.2
Raspberry Pi 3 model B (3 nodes)	1200 MHz	1 GB	2.2	4.8
Raspberry Pi 3 model B+ (1 node)	1400 MHz	1 GB	2.5	5.8
Raspberry Pi 4 model B (1 nodes)	1500 MHz	4 GB	3.3	6.2
Micro-Cluster	–	–	**18**	**36**

5.3 Analytical Model for Energy Consumption

This section formulates an energy consumption analytical model to characterise energy consumption for edge micro-cluster platforms for executing edge-related applications. Section 5.3.1 develops an analytical model used to predict how much energy an edge micro-cluster system would require for performing edge-related workloads, and the performance is analysed in Section 5.3.2.

5.3.1 Energy Consumption Analytical Model

Energy Consumption is defined as the amount of power, measured in watts, consumed by micro-clusters over a period of time, measured in seconds. Equation. 5.1 formulates the Energy Consumption for micro-clusters.

$$Energy\ Consumption = Makespan\ Time * Cluster\ Power \tag{5.1}$$

Similar to the definition in Chapter 4, Makespan Time is defined as the execution time for a micro-cluster system to complete processing a set of tasks in batch execution. Equation 5.2 defines the Makespan Time, where: m_n is a constant of a gradient of node n, $numTask_n$ is the number of tasks assigned to node n, and c_n is a constant value of y-intercept of node n. The interpolation values for m_n and c_n are derived from Table 4.2 in presented Chapter 4.

$$Makespan\ Time = \max_{n \in N} \left(m_n * numTask_n + c_n \right) \tag{5.2}$$

Cluster Power is defined as the total amount of power required by a micro-cluster to execute workloads. As the micro-cluster might not use all its nodes for executing applications, Cluster Power effectively refers the power required by active nodes only, while inactive nodes are turned off to reduce power consumption and preserve energy. The data related to power consumption is obtained through empirical measurements using an energy monitor device, as described in Section 5.2.

Cluster Power is the sum of power consumption of active nodes in their active states. Active nodes are those within the cluster that are executing workloads based on the allocation solutions. Inactive nodes within the cluster are turned off and do not consume energy. Equation 5.3 represents the sum of power consumption for active nodes, where $power_n$ is the power consumption of node n in an active state, and N is the set of relevant active nodes.

$$Cluster\ Power = \sum_{n \in N} power_n \tag{5.3}$$

In our energy consumption model, we assume that active nodes are fully utilised during workload execution. More complex linear power models that estimate power consumption based on resource utilisation may not be applicable for edge systems [7]. Section 5.3.2 presents the performance evaluation of the energy consumption analytical model.

5.3.2 Performance Analysis

This section evaluates the energy consumption analytical model developed in Section 5.3.1. The developed model is empirically evaluated on a physical micro-cluster prototype testbed and by utilising the DeFog benchmark suite extended version presented in Chapters 3 and 4.

More specifically, the micro-cluster testbed comprises several heterogeneous and resource-constrained physical nodes compacted into a mini-rack case and connected using a network switch. Figure 3.1 presents the configured micro-cluster prototype. The DeFog benchmark suite [19] is extended to develop edge-related workloads by utilising four different applications scenarios, the image-detection application (Yolo), the audio-text recognition application (PocketSphinx), the audio-text synchronisation application (Aeneas), and a heterogeneous workload scenario where computing tasks are offloaded from various applications and thereby aggregated into heterogeneous batches. The Energy Consumption of the whole micro-cluster testbed is monitored using an energy monitoring device. For further reference, the experimental micro-cluster testbed configuration and the applications benchmarks were explained in detail in Chapter 3.

Figures 5.2 and 5.3 show the results. Figure 5.2 presents a comparison of the predicted Makespan Time with the actual Makespan Time for executing various workloads in batch execution in edge micro-cluster testbed. Figure 5.3 compares the predicted Energy Consumption estimated by the analytical model with the actual energy consumed by the micro-cluster for handling the application workloads. All figures report the mean of 10 runs for each experiment.

The results overall illustrate that the analytical energy consumption model is quite accurate and can effectively predict the required computation time for workloads and the Energy Consumption for edge micro-clusters. Both the estimated Makespan Time and Energy Consumption calculated by the analytical model are within 5% of the actual Makespan Time for all benchmark applications and the actual Energy Consumption consumed by the micro-cluster testbed.

There are minor overestimations in Energy Consumption calculations, which might be attributed to the power consumption input data used in the analytical model. The per-node power consumption data used in the energy consumption analytical model are based on 100% CPU utilisation using the $sysbench$ system benchmark, as explained in Section 5.2. However, the allocation decisions in the experiments effectively distribute workloads over the nodes resulting in not fully stressing the node CPU and therefore reducing the actual power consumption. Furthermore, as nodes in the micro-cluster system are heterogeneous, meaning that the powerful nodes can complete executing their allocated workloads and return to their idle states before other nodes. This will result in reducing the actual power consumption required by the micro-cluster. For example, node type Raspberry Pi 4B might finish processing its workloads early and return to its idle state before other nodes. This will reduce the micro-cluster power consumption for executing the remaining workloads. The analytical model predicts that the power consumption remains constant over the estimated makespan time.

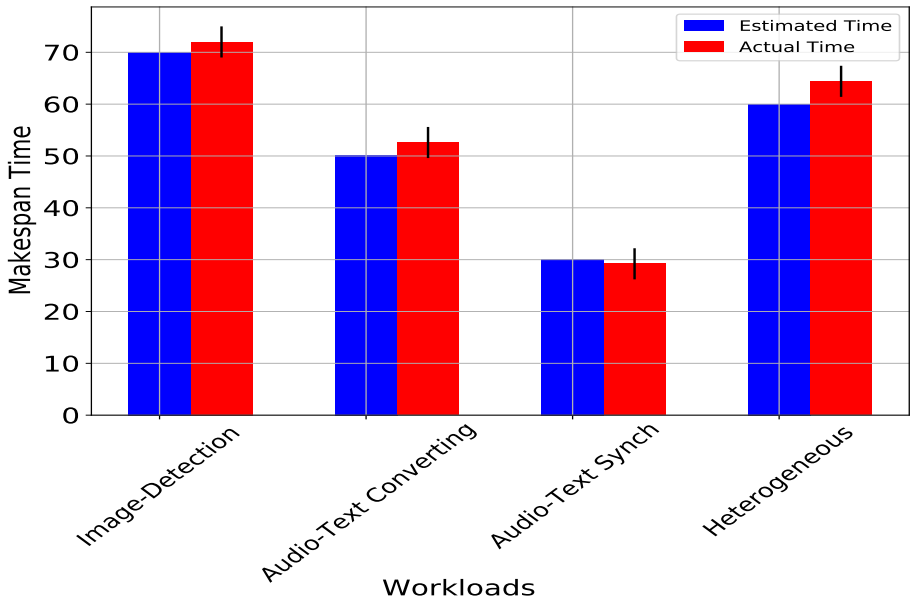


Figure 5.2: Analytical model for the predicted and the actual *Makespan Time* for micro-cluster platforms. *Makespan Time* is in (sec). All figures report the mean of 10 runs for each experiment.

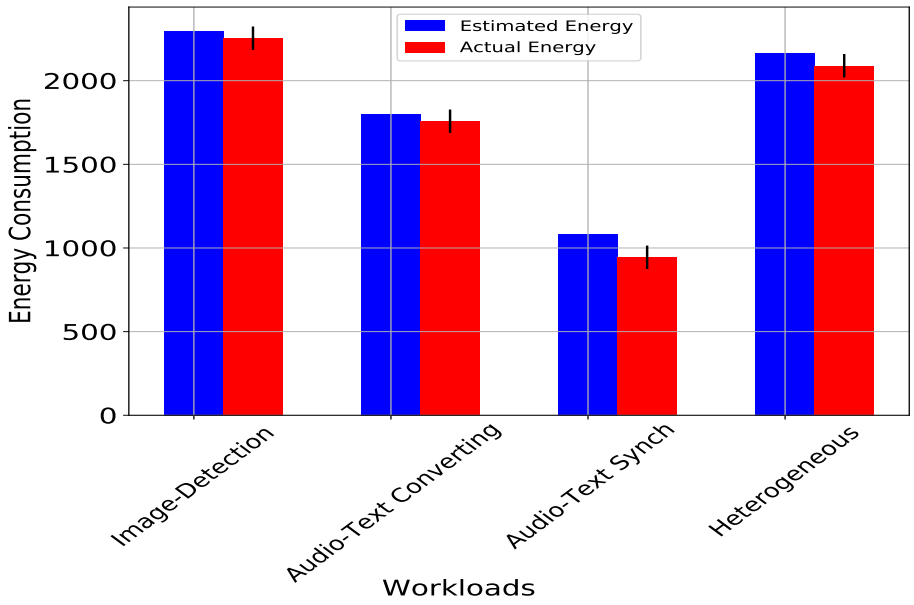


Figure 5.3: Analytical model for the predicted and the actual *Energy Consumption* for micro-cluster platforms. *Energy Consumption* is in (Joules/sec). All figures report the mean of 10 runs for each experiment.

5.4 Multi-Objective Optimisation for Micro-Clusters

This section presents a multi-objective optimisation model developed for optimising two essential performance metrics for edge micro-cluster systems, the Makespan Time and Energy Consumption. In Section 5.4.1, a multi-objective optimisation model is formulated using a weighted sum approach. Section 5.4.2 evaluates the performance of the model by defining Pareto Front representation of the set of optimal solutions representing the best trade-off between the conflicting objectives of the system.

5.4.1 Multi-Objective Optimisation Model

In the context of edge micro-cluster systems, the multi-objective optimisation aims to optimise two critical edge-relevant metrics of the Makespan Time and Energy Consumption for edge micro-cluster platforms. As defined earlier in Section 5.3, Makespan Time is the execution time required for a micro-cluster platform to execute application workloads in batch execution mode, and Energy Consumption is the total amount of energy a micro-cluster consumes for executing the required computation logic. These edge metrics are critical performance metrics to optimise in deployment conditions where, for example, the power supply resources are not available.

The multi-objective optimisation is modelled and normalised using a weighted sum approach. The weighted sum approach involves forming a composite objective function that combines the individual objective functions and assigns weighting factors to each objective function to determine its relative priority [108].

Because the objective functions in the system have different magnitudes, normalisation is applied to scale the values of the objective functions between 0 and 1, regardless of their original magnitudes. The normalisation facilitates fair comparison between the competing objectives of the systems. The objective function values are normalised using minimum-maximum normalisation, which subtracts the minimum value from the original value and then divides it by the range value of the objective function.

The minimum-maximum normalisation is implemented as follows: $NormalisedValue = (Value - minValue)/(rangeValue)$, where, $NormalisedValue$ is the new scaled value, $Value$ is the value obtained from the objective function, $minValue$ is the minimum value of the objective function, $maxValue$ is the maximum value of the objective function, and $rangeValue$ is the range calculated as $(maxValue - minValue)$. Equation 5.4 models the multi-objective function of the system.

$$Objective = (\alpha * Makespan\ Time) + ((1 - \alpha) * Energy\ Consumption) \tag{5.4}$$

Alpha, α, is a weighting factor $0 \leq \alpha \leq 1$. Alpha is used to determine the balance between the conflicting system objectives, which are the Makespan Time and Energy Consumption in our system model. Determining the appropriate α value is challenging and depends on various deployment settings and application requirements. The general principle of determining α depends on the system deployment settings, where it allows the system operator to adjust the weights based on preferences and priorities. For example, in the case of a micro-cluster that operates using limited power systems such as batteries, energy consumption in such deployment settings is critical to optimise in order save power. On the other side, if IoT applications have strict QoS expectations and require, for example, fast execution time for critical application workloads like real-time image processing, the execution time is more important to optimise.

As defined earlier, Makespan Time is total execution time required for a micro-cluster system to execute application workloads in batch execution. Makespan Time is calculated based on the linear model presented in Equation (5.2), where m_n is the gradient of cluster node n, $numTask_n$ is the total number of tasks allocated to node n, and c_n is the y-intercept value for node n. The linear interpolation values for the gradient m_n and the y-intercept c_n are defined in Table 4.2.

Energy Consumption is the amount of energy consumed by a micro-cluster platform over a period of time required for executing application workloads. Energy Consumption is calculated according to the Energy Consumption equation presented in Equation 5.1.

As mentioned above, note that the Makespan Time and Energy Consumption are normalised using min-max normalisation to provide standardised comparisons.

5.4.2 Performance Analysis

5.4.2.1 Experimental Setup

The experimental setup used for evaluating the designed multi-objective optimisation model for micro-clusters is modelled according to the micro-cluster system models developed in Chapters 3 and 4. Specifically, the evaluation involves examining the performance of the developed multi-objective optimisation model for an edge micro-cluster platform comprised of various heterogeneous worker nodes. In terms of the edge workloads, the interpolation data related to edge applications and the number of tasks in batches are similar to edge applications scenarios drawn from the DeFog benchmark suite [19]. Table 4.2 demonstrates data for application scenarios in detail.

The multi-objective optimisation is solved by utilising the metaheuristic Particle Swarm Optimisation technique (PSO) developed in Section 4.3.2 with changing the system objective

function. PSO technique is selected for solving the developed optimisation for its features for 1) providing fast allocation overhead time and 2) generating near-optimal effective allocation decisions, as demonstrated in Chapter 4. The experiments were run for 100 runs for each application scenario.

5.4.2.2 Results Analysis

To analyse the performance of the developed multi-objective optimisation model for micro-cluster platforms, the Pareto front (PF) concept is employed to capture the model's relative performance and draw meaningful results. The Pareto front can help identify the set of best optimal allocation solutions that represent potential trade-offs between the systems performance metrics, the Makespan Time and Energy Consumption, in the current problem settings. The PF can provide a comprehensive view of the trade-offs and potential solutions for multi-objective optimisations, which, therefore, can enable effective decision-making and optimisation.

Figures 5.4 depict the PF representing the relationship between Energy Consumption required by the micro-cluster system and the Makespan Time of workloads. The competing objective functions are normalised here, where 0 and 1 on x-axis indicate the near-optimal possible Makespan Time and a maximum Makespan Time respectively and numbers on y-axis represent the Energy Consumption and are translated similarly.

Alpha α represents different priority weights for adjusting the trade-off. Increasing α value is attributed to prioritising the Makespan Time, which yields minimising the execution time of workloads over the Energy consumption. Decreasing α value prioritises the Energy Consumption over the Makespan Time. Furthermore, Figure 5.5 shows the PF illustrating the static Power Consumption an edge micro-cluster system would require for improving the Makespan Time of various workloads.

Overall, the PF clearly demonstrates that improving the Makespan Time cannot be achieved without increasing the Power Consumption of the micro-cluster system. Specifically, if the primary system objective is the Makespan Time, the micro-cluster system needs to operate at its maximum capacity, which requires high Energy Consumption.

Ultimately, in the context of edge micro-cluster deployment, Pareto front performance evaluation demonstrates a set of solutions that provide the best trade-off between the Makespan Time and Energy Consumption. The solutions generally will depend on specific deployment objectives and constraints for different use cases. For example, it may be more important to prioritize minimizing Makespan Time over Energy Consumption in order to meet application deadlines in some critical deployment scenarios. On the other side, prioritizing and minimizing Energy Consumption might be more important if micro-clusters operate on a secondary

power supply system to reduce costs, save power, or reduce environmental impact. Overall, we argue that the developed multi-objective optimisation model is useful and applicable for edge micro-cluster deployment.

The developed multi-objective optimisation model can benefit edge micro-cluster platforms for edge deployment scenarios where several system objectives require optimisation. The multi-objective optimisation model can support decision-making by generating a set of optimalities that represent trades-off for the competing objectives.

5.5 Summary

With the current global concerns related to energy consumption issues, ongoing research aims to develop solutions for maintaining and optimising energy usage or developing alternative solutions like renewable ones. This chapter, therefore, investigated energy consumption in edge micro-cluster platforms. The work extended the previous technical chapters by developing an energy consumption analytical model for characterising, predicting and optimising energy consumption in micro-cluster systems.

The preliminary experiments investigated per-node power consumption characteristics for micro-clusters under two fundamental operation states, the idle and active state. Overall, the data reveals that idle nodes consume a considerable amount of energy, which requires effective management to save the overall system energy.

In addition, an energy consumption analytical model is developed for predicting energy consumption in micro-clusters. The model can provide an accurate estimation of the execution time and the relevant energy consumption required for executing application workloads. The model can feasibly be integrated into the deployment of energy consumption predictions to enable effective decision-making.

Furthermore, a generic multi-objective optimisation framework is designed to optimise two essential performance metrics in edge platforms, makespan time and energy consumption. A Pareto optimisation is introduced to balance and prioritise execution time and energy consumption for micro-clusters. The multi-objective optimisation framework is limited to two edge-relevant performance metrics in micro-cluster platforms. However, the framework can be generalised to integrate and optimise additional metrics such as networking metrics and monetary cost.

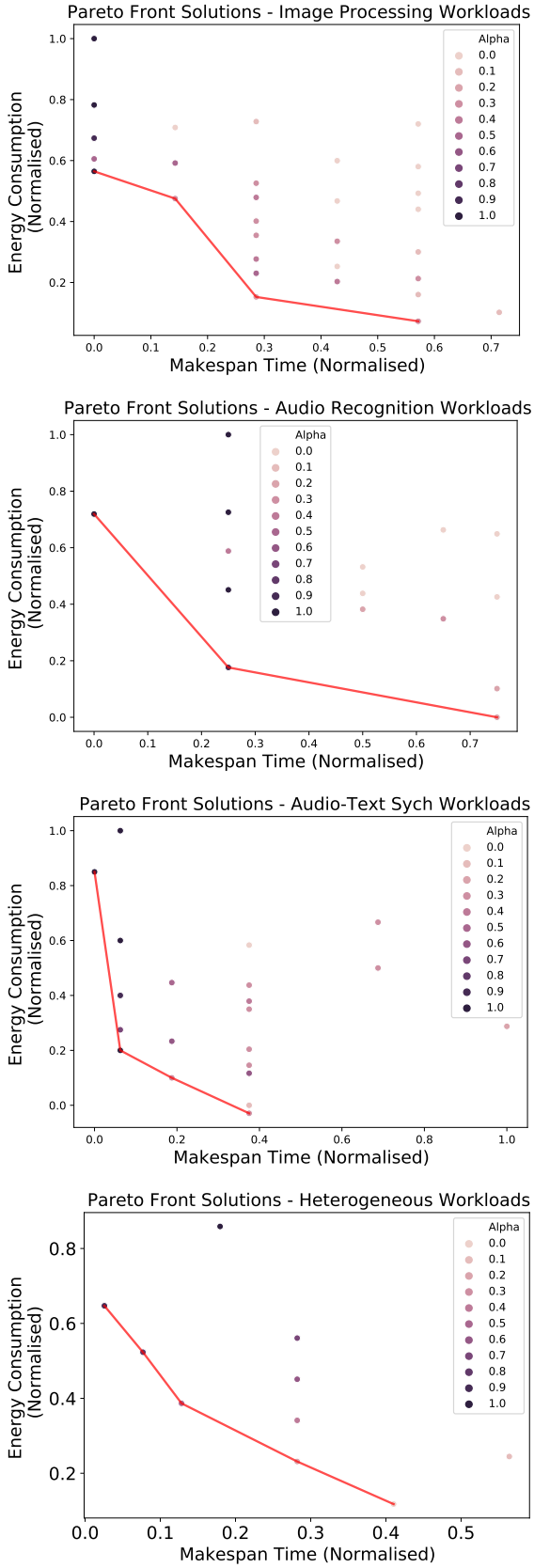


Figure 5.4: Pareto front representing solutions and trade-offs between *Makespan Time* and *Energy Consumption* for different workloads.

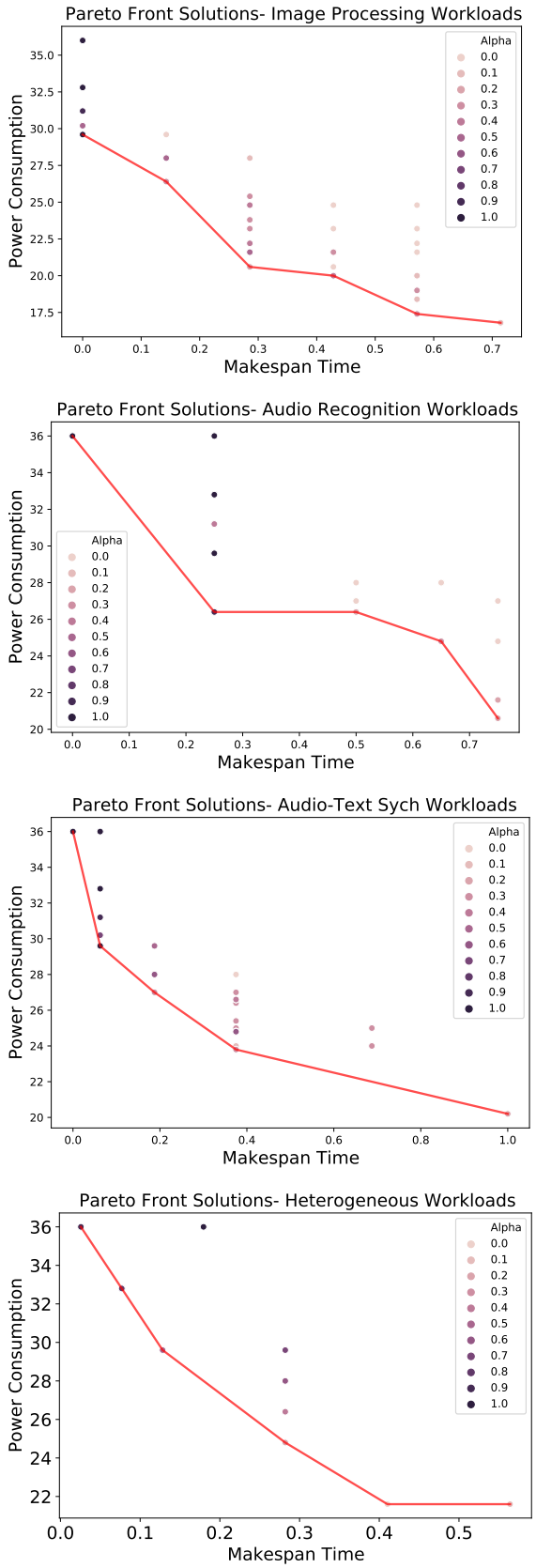


Figure 5.5: Pareto front representing solutions and trade-offs between *Makespan Time* and *Power Consumption* for different workloads.

Chapter 6

Conclusions and Future Directions

6.1 Overview

Edge computing is fundamentally a method to utilise resources at the edge of the network to locally perform computation and storage logic. The ultimate aim is to avoid or mitigate deployment concerns related to networking latency, data privacy, and energy consumption. Many solutions and computing paradigms have been proposed to accelerate edge computation, including MEC, Cloudlet, and Fog Computing. This thesis sheds light on edge micro-cluster platforms in order to enhance their feasibility and usability for edge and IoT environments by improving and optimising task allocation for these edge computing systems.

This thesis presented the development of edge micro-cluster platforms; pragmatic, compact, and low-cost edge infrastructures based on Single Board Computers for edge computing and IoT environments. The research developed applicable linear analytical models based on workload features, node performance trends and per-node power consumption. The models can effectively predict the makespan time and energy consumption required by micro-cluster systems to execute edge workloads in batch execution. Furthermore, a comparative performance evaluation of various optimisation techniques is conducted to quantify their relative effectiveness in finding optimal or near-optimal solutions and efficiency for providing allocation decisions in a reasonable time for solving task allocation and workload management in batch execution in such edge micro-clusters. Finally, the thesis developed a practical optimisation framework for optimising and addressing multi-objective trade-off scenarios for edge micro-clusters.

This chapter concludes this research and outlines potential future directions identified in this thesis. Section 6.2 revisits the thesis statement in light of the experiments and findings from the research. Section 6.3 presents a comprehensive answers to the thesis research questions. Section 6.4 summarises the main contributions and findings of this thesis. Section 6.5

presents the potential directions for future work derived from the limitations and the possible extensions to the current work. Finally, Section 6.6 provides concluding remarks on this thesis.

6.2 Review of the Thesis

In light of the conducted research, experimental work, and the findings from this thesis, the thesis statement of this work is now revisited to reflect the contributions and findings. The following section provides answers and reflections on research questions accordingly.

This thesis asserts that edge micro-cluster platforms require effective and efficient task allocation optimisation techniques that can satisfy resource constraints, dynamic workloads, and the QoS requirements of IoT-based applications. To achieve this statement, this thesis presents the design of a micro-cluster prototype for edge and IoT environments, develops analytical system models for task allocation for micro-cluster systems, and presents empirical evaluations of various optimisation techniques for optimising task allocation in edge micro-cluster systems. The thesis demonstrates that mathematically-based optimisation techniques can provide optimal or near-optimal decisions for small-scale micro-cluster systems, while metaheuristics-based optimisation techniques are efficient and effective for optimising task allocation for large and complex micro-clusters.

6.3 Reflection on Research Questions

6.3.1 Reflection on Research Question RQ1

RQ1. *Which task allocation optimisation methods provide effective solutions to optimise makespan time for edge micro-cluster platforms for heterogeneous workloads in batch execution mode?*

To answer this research question, the experiments conducted in Chapters 3 and 4 presented comparative evaluations of various task allocation optimisation techniques for micro-cluster platforms to optimise task allocation in batch execution. The makespan time performance metric was used to measure the effectiveness of optimisation techniques. The makespan time is effectively the maximum execution time micro-cluster systems require to complete processing application workloads in batch execution. The optimisation techniques that successfully minimise the required makespan time are considered optimal. The performance analysis demonstrated that effective task allocation is essential in micro-clusters and can

cause significant improvements in the overall performance of edge micro-clusters. The evaluations revealed that mathematical-based optimisation, i.e., using mixed-integer programming, and metaheuristic-based optimisation, i.e., using PSO-based optimisation, with respect to the nodes' capacities constraints outperform other heuristic-based allocation by effectively utilising the cluster resources and minimising the makespan time of edge applications. The experiments show that mathematical-based optimisation and metaheuristic-based optimisation can achieve comparable solution quality, proving the viability of metaheuristic-based optimisation to produce optimal or near-optimal solutions.

6.3.2 Reflection on Research Question RQ2

RQ2. *Which optimisation methods are efficient and appropriate (i.e., sufficiently lightweight) for edge micro-cluster platforms?*

To answer this research question, Chapters 3 and 4 presented efficiency-related evaluations of task allocation optimisation techniques for edge micro-clusters. The allocation overhead time is the performance evaluation metric utilised to measure the efficiency of the optimisation techniques. Allocation overhead time basically is the computation time the employed optimisation techniques required to solve the given optimisation problems. Allocation overhead is a critical performance metric in edge micro-clusters as devices in such systems are typically resource-limited, and complex optimisation may require powerful devices and yield expensive computation time.

The performance analysis revealed that the mathematical-based optimisation technique provides light overhead time for small-scale micro-clusters, i.e., for micro-clusters with less than ten nodes for up to 100 homogeneous tasks and 50 heterogeneous workloads. However, the allocation overhead time of this technique increases exponentially, affecting the overall makespan time of applications.

Furthermore, results demonstrate that metaheuristic-based optimisation overhead time keeps linear overhead time for edge micro-clusters and therefore outperforms the mathematical-based optimisations for larger problem sizes in micro-clusters, i.e., for workloads with more than 100 homogeneous tasks and 50 tasks heterogeneous workloads (see Figures 4.3).

Overall, conclusions that can be drawn from the efficiency performance analysis are that mathematical-based optimisation is only recommended for small-scale micro-cluster platforms, while metaheuristic-based optimisation techniques are recommended for complex and large-scale micro-cluster systems.

6.3.3 Reflection on Research Question RQ3

RQ3. *Which task allocation optimisation methods provide effective solutions to optimise multi-objective optimisation for edge micro-clusters?*

To answer this research question, Chapter 5 developed and evaluated a multi-objective optimisation framework for edge micro-cluster platforms. The work developed a multi-objective optimisation framework by employing the weighted sum approach to optimise two edge-relevant performance metrics of the makespan time and energy consumption required by micro-clusters to execute application workloads. The experiments utilised metaheuristic-based optimisation to solve the modelled multi-objective optimisation problem. Specifically, the PSO-based optimisation is selected to address the developed multi-objective optimisation problem for its functional features related to efficiency and effectiveness, which were demonstrated in Chapter 4. The evaluation furthermore presented a Pareto front set of possible solutions, providing a comprehensive overview of possible trade-offs between the competing objectives of the system.

6.3.4 Reflection on Research Question RQ4

RQ4. *Is it appropriate to use commodity Single Board Computers (SBCs) to model edge micro-clusters in order to empirically evaluate task allocation techniques for batch-mode execution?*

To answer this question, the experiments performed in Chapter 3 developed a realistic edge micro-cluster testbed configured using different Single Board Computers, i.e., Raspberry Pi devices, connected by using a networking switch commodity and mounted in a mini racks cluster case. The experiments empirically demonstrated that edge micro-cluster platforms configuration are straightforward using Single Board Computers and can effectively handle heterogeneous edge and IoT workloads by utilising appropriate and effective task allocation optimisation techniques. The experiment contributed to empirically evaluating optimisation techniques in the physical environment instead of using simulation and analytical tools, which merely provide an abstraction of the environments and might not reflect comprehensive performance. Generally, micro-clusters will likely comprise several Single Board Computers similar to Raspberry Pi nodes mounted into the mini-rack case to facilitate effective deployment and easy movement in edge and IoT environments. The overall design concept proves the possibility of micro-clusters to make edge computing more affordable and available for various smart IoT and edge computing use cases.

6.3.5 Reflection on Research Question RQ5

RQ5. *What kind of edge workloads provide real-world representative benchmarks for evaluating task allocation methods for batch-mode execution on micro-cluster platforms?*

To answer this research question, the experiments conducted in Chapters 3 and 4 developed edge representative workloads for edge micro-clusters based on the DeFog benchmark suite [19]. The DeFog benchmark was initially developed to fill the gap in edge benchmark by developing edge benchmarks to compare the performance of different edge and cloud platforms for single task execution. The DeFog benchmark is recognised in the edge research community because it provides representative edge applications by utilising containerisation technology that may help researchers to perform the required evaluations. This research further developed and extended the DeFog benchmark by: 1) modelling multitasking execution, 2) expanding the sets of workloads for selected benchmarks, and 3) modelling heterogeneous workload scenarios to model more representative edge environments. Therefore, the extended version of the DeFog provided real-world representative edge applications scenarios that can be employed to empirically evaluate various management aspects in edge micro-cluster systems.

6.4 Research Contributions

This thesis has addressed the problem of task allocation and workload management in edge micro-cluster platforms for edge and IoT environments. The research focuses on task allocation decisions relating to batch execution in edge micro-cluster settings for optimising edge-relevant performance metrics of makespan time and energy consumption. The work starts by constructing a physical and realistic micro-cluster prototype configured using heterogeneous SBC computers to model edge micro-cluster platforms. The research further adopts and develops edge representative applications based on DeFog edge benchmark suites for evaluating the relevant performance of edge environments. The problem of task allocation in micro-clusters was initially formulated utilising the general assignment problem before developing a linear-based system model customised for edge micro-clusters. The node performance trends for handling edge-relevant workloads in concurrent execution is characterised. The work further develops a multi-objective optimisation framework for optimising edge-relevant performance metrics. The research conducted comparative performance evaluation demonstrating the benefits of utilising the metaheuristic-based optimisation techniques over mathematical-based and heuristic-based optimisation for large-scale micro-clusters. The contributions of this thesis can be summarised as below.

6.4.1 Micro-Cluster Prototype for Edge and IoT Environments

Chapter 3 presented a compelling prototype for edge micro-cluster platforms configured using a realistic micro-cluster testbed, representative benchmark applications and wall-clock timing metrics. The work overall demonstrates a practical use case for micro-cluster platforms for edge computation and IoT environments. The underlying hardware and software specifications, edge workloads, features, and characteristics were described in detail. The work further highlighted the importance of experimenting with a configured system tested for reporting informative performance analysis of the proposed optimisation techniques instead of utilising simulation-based software solutions. In addition, various optimisation techniques were developed for optimising task allocation for such micro-clusters.

6.4.2 Analytical Models for Edge Micro-Clusters

Chapter 4 investigated micro-cluster nodes' performance and thereby characterised the performance trends of micro-cluster nodes for processing edge-relevant applications. The performance evaluations demonstrate that micro-clusters nodes' performance linearly increases in relation to the number of tasks. Thereby, the chapter presented the design and evaluation of an analytical performance model of micro-cluster platforms that can effectively predict the makespan time required to process heterogeneous workloads in batch-execution mode. By utilising a linear-based model, the model can accurately calculate the required makespan time for workloads.

6.4.3 Multi-Objective Optimisation Model for Micro-Clusters

Chapter 5 contributes with the design and evaluation of 1) the energy consumption analytical model for edge micro-clusters and 2) the multi-objective optimisation model for edge micro-clusters. The chapter first characterised per-node power consumption under two fundamental operational states, i.e., the idle and active states, and designed the energy consumption analytical model accordingly. The model can effectively predict the energy consumption required for executing different edge applications. Secondly, a multi-objective optimisation model is designed to optimise multi-objective optimisation for micro-cluster platforms. The model can generate a Pareto-optimal set of feasible solutions representing the trades-off system's competing performance objectives. The model was evaluated by employing two essential edge performance metrics of the makespan time and energy consumption required for executing edge applications. Moreover, the developed multi-objective model can be easily extended to a generic multi-objective optimisation framework to involve and optimise other performance metrics, such as operation cost and networking metrics.

6.4.4 Comparative Evaluation of Optimisation Techniques for Edge Micro-Clusters

This thesis overall provides comparative evaluations of various optimisation techniques for optimising task allocation management problems in edge micro-cluster platforms. The optimisation techniques employed in this thesis involve standard optimisation categories utilised in edge, fog, and cloud computing paradigms for solving related optimisation problems. Specifically, the techniques involve mathematical, metaheuristic, and heuristic-based optimisations. By conducting various empirical evaluations utilising a physical edge micro-cluster testbed and representative-edge applications, the analysis provides relative performance of the optimisation techniques for edge micro-cluster platforms and reveals informative overlapping attributed to the efficiency and effectiveness of different optimisation techniques for orchestrating and optimising workload allocation in edge micro-clusters.

6.5 Limitations and Future Directions

This thesis has highlighted the potential of micro-clusters as lightweight and compact infrastructural platforms to deliver computing and storage services to various IoT applications in edge computing and IoT environments. The work emphasised the importance of addressing and optimising task allocation for such resource-limited platforms to make them more viable for edge environments. This section outlines potential future research directions and discuss limitations based on this thesis.

6.5.1 Generalising System Heterogeneity

The work in this thesis constructed and evaluated an edge micro-cluster platform comprised of several heterogeneous Raspberry Pi devices clustered using a mini-rack cabinet and interconnected using a networking switch. The configured system testbed overall can model the system heterogeneity anticipated in such edge micro-cluster platforms. This can be in terms of nodes' hardware specifications and the modelled application workloads. The system heterogeneity in this thesis is limited to a single device, that is, various generations of Raspberry Pi devices. For future directions and to generalise the system settings, we can suggest incorporating and involving other Single Board Computers, such as Odroid devices, Arduino, and microcontrollers, to model a more heterogeneous edge system. This would be more relevant to generalising edge micro-clusters concepts for IoT and edge settings.

6.5.2 Expanding System Scale

The size of the edge micro-cluster system configured for this research may be arguably undersized relative to the cluster size envisioned in real-world edge environments. Nevertheless, the system testbed can be feasible and useful to perform empirical experiments, handle edge-relevant applications and generate meaningful results for performance evaluation. In addition, the linear-based and multi-objective optimisation models developed in this thesis can be efficiently generalised and adjustable to accommodate evaluating larger-scale micro-clusters. In short, expanding the system scale and involving other SBC types can be potential directions for future work.

6.5.3 Evaluating Complex Metaheuristics

The performance evaluation overall demonstrates that metaheuristic-based optimisations are viable techniques for solving task allocation in micro-cluster settings. Specifically, this thesis considers the metaheuristic Particle Swarm Optimisation (PSO) for solving the optimisation settings in edge micro-cluster. However, the work in this thesis was limited to swarm-based metaheuristic optimisation. For future directions, we can envision the applicability of evaluating other metaheuristic optimisations, including various metaheuristics themes such as evolutionary algorithms, genetic algorithms, other swarm-based optimisation, or simulated annealing. This is to evaluate the performance of different metaheuristic-based optimisations for edge micro-cluster settings.

6.5.4 Optimising Other Resource Management and Performance Metrics

This thesis focused on optimising task allocation in edge micro-cluster platforms, which is fundamental resource management in such edge systems. Edge computing, however, comes with several complex resource management aspects, including load balancing, task migration, resource provisioning, and more. We can recommend exploring and optimising other resource management aspects for such edge compact systems for future direction. Furthermore, this thesis optimised the makespan time and energy consumption while other edge-relevant performance metrics, such as networking metrics and monetary costs, can be an area for future directions.

6.5.5 Evaluating More Deployment Settings

This thesis experimented with representative edge-relevant applications including image-processing workloads, audio-recognition workloads, audio-text synchronisation, and heterogeneous workloads in a controlled micro-cluster experimental testbed. The workloads are recognised in the edge research community as representative edge benchmarks [19, 20, 104]. For future work, we recommend extending the deployment setting of edge micro-cluster platforms in more realistic edge, and IoT environments [59]. For example, we can envision deploying the edge micro-cluster to extend computing services to edge environments such as agriculture or smart home environments.

6.5.6 Developing Edge Benchmarks for Micro-Clusters and IoT Environments

This thesis adopted and enhanced the DeFog benchmark suite [19] to generate edge representative workloads and IoT applications by the following key extensions.

- Modify the DeFog logic. The DeFog benchmark suite is developed to execute a single task per time slot on each benchmark platform. This research extended the DeFog logic to allow multitasking and parallel execution across a cluster of nodes.
- Harness the DeFog Payloads. DeFog comes with limited workloads and tasks for each application. This research further harnesses the DeFog payloads with multiple and heterogeneous workloads to enable parallel executions.
- Evaluate More SBC Devices and Edge Platforms. The DeFog benchmark evaluates two single board platforms representing edge nodes and a an cloud platform. This research extended the DeFog benchmark to evaluate an edge micro-cluster platform.

Further research is recommended in the area of edge benchmarks to develop customised edge benchmarks [104, 109, 110] for micro-clusters and IoT systems. This can be executed by building on the DeFog original version and the extended version developed in this research.

6.6 Final Summary

Edge computing has been drawing the attention of the distributed systems research community and industry developers for several years given its benefits in augmenting cloud computing and IoT applications. Edge micro-cluster platforms are being utilised for edge computing and IoT environments for their interesting deployment features like low-cost, small physical footprints, and sufficient collective computing resources for edge and IoT applications.

This thesis aimed to enhance the feasibility and usability of such micro-cluster systems for edge deployments. The thesis has presented a compelling design concept of edge micro-cluster platforms for edge computing and IoT environments by characterising edge-relevant applications, identifying node performance trends, and optimising task allocation and workload management for such edge computing platforms.

The findings show that incorporating and optimising task allocation for edge micro-clusters can significantly improve the performance of edge applications. The thesis empirically proved that metaheuristics optimisation tools using PSO-based techniques demonstrate viable techniques for complex and large-scale micro-cluster systems by providing effective and efficient decisions. In contrast, the mathematically-based techniques can only provide optimal or near-optimal decisions for small-scale micro-cluster systems. The work can help in future deployments of micro-clusters in edge and IoT environments.

Appendix A

Glossary

Batch-execution

Batch-execution is an execution technique by which a set of tasks are scheduled to be processed in parallel. Each batch might contain homogeneous or heterogeneous workloads offloaded from various IoT devices and applications.

Benchmarks

Benchmarks are software tools used to measure, characterise, and evaluate relative performance of computer systems. Benchmarks that are used to test systems utilities are considered systems-related benchmarks. Benchmarks representing applications and workloads characteristics are designed to understand and capture system performance for real-world applications and workloads.

Cloud Computing

Cloud computing a system paradigm that provides on-demand and shared computing, storage and networking services to the end user. Cloud Computing paradigm faces significant challenges when support IoT-based applications, such as high latency, jitters, high response time, energy consumption. This led to decentralizing the Cloud Computing introducing a new recent computing paradigm known as, Edge Computing, Fog Computing, or Mobile Edge Computing.

Cloudlet

Cloudlet is a resource-rich computer or cluster of powerful computers deployed near the edge of network to provide computing services to mobile and stationary devices.

Cluster

A cluster is a computing system consists of a collection of inter-connected physical computer machines working together as a single computer machine.

Compute Node

A compute node is a single independent compute machine in the cluster performs jobs assigned to it. In micro-cluster, nodes can be any Single Broad Computer like Raspberry Pi or Odroid devices. In traditional cluster, compute node can be more powerful computer server.

Container Technology

Container technology is a lightweight virtualisation technology that enables lightweight isolation by packaging all applications dependencies in containers.

Data Centre

A Data centre is large physical computing infrastructure comprised of ten of thousands of powerful computing, storage, and network resources deployed far from end users and typically used for Cloud Computing.

Edge Computing

Edge computing is a recently distributed computing paradigm that utilises various resources deployed near or at the network's edge to provide computing and storage services to tiny end devices like IoT sensors. Edge computing typically leverages resources located at the network edge, such as networking routers, networking switches, and public/private micro-data centres.

Edge Data Centre

Edge data centre is a small-scale data centre deployed at the edge of the network close to the end-users (for example in shopping mall, stadium) to provide processing and storage services instead of using traditional centralized data centres. Edge Data Centres are being used interchangeably with Edge Node, Edge Cloud, and Cloudlet in the literature. Edge Data Centres is expected to support limited number of applications due to the resource limitations.

Edge Node

Edge node is a compute node such as an individual server or sets of computing resources deployed as a part of edge computing data centres and typically located in a close proximity to the end users.

Fog Computing

Fog computing refers to a hierarchical computing paradigm proposed to utlise computing and storage resources using resources scattered across the network between the IoT devices and the remote Cloud servers. Fog Computing and Edge Computing are being used interchangeably in by the research community. However, Fog Computing makes use of resources in Edge and Cloud, while Edge Computing makes use of resources deployed at the network edge.

IoT device

An IoT device refers to a small-factor device that is not equipped with sufficient processing and storage resources, such as sensors, gadgets, and wearable devices. An IoT device is typically a resource-limited device that is not capable of executing intensive-computational tasks. Various applications utilise IoT devices for generating data. For example, sensor devices are used in smart farming applications to generate and offload environmental data like soil humidity and temperatures.

Load Balancing

Load balancing is the process of distributing or re-scheduling load across resources to avoid resources overwhelming or task failure to efficiently utilise resources.

Metaheuristics Optimisation

Metaheuristics optimisation is a category of optimisation techniques designed to search and find near-optimal solutions for complex optimisation problems in linear time. Metaheuristics optimisation is distinguished by its logic that simulates various natural features by which it has the ability to find near-optimal solutions.

Micro-Cluster

A micro-cluster platform is a small, potable, and compact computing cluster composed of heterogeneous resource-constrained compute nodes like Raspberry Pi or Odriod, mounted in mini racks or housed in a cluster case to construct a standalone computing platform. A

micro-cluster is typically controlled by a head node. In contract to typical cluster deployed in traditional cloud data centers, a micro-cluster might be equipped with limited computing resources.

Resource Allocation

Resource allocation refers to the process of distribution of resources, such as CPU, memory, and networking bandwidth, over tasks and users. Resource allocation can also involve determining the required number of resources for executing computational tasks and managing resources considering the network dynamics and workload characteristics.

Resource Management

Resource management is a comprehensive term that covers various management aspects and techniques in cloud and edge environments, including hardware and devices, workloads orchestration techniques such as task allocation, load balancing, load migration and resource provision, and optimisation techniques to handle complex management issues.

Resource Provisioning

Resource provisioning refers to the process of dynamically scaling up and scaling down systems' resources to adapt to workload characteristics and networking dynamics to optimise various QoS metrics and improve system performance metrics such as resource utilisation, energy consumption, execution time and cost.

Single Board Computer

A Single Board Computer is a type of resource-constrained computer device, in which all computing resources including processors, memory, storage, and input and output ports are all build on a small size circuit board. There is a wide range of Single Board Computers including, but not limited to, Raspberry Pi, and Odroid.

Task Allocation

Task allocation refers to the process of effectively assigning computational tasks to appropriate computing nodes, considering factors such as QoS metrics, resource utilisation, execution time, and energy consumption. The objective is to distribute tasks efficiently over computing nodes to meet diverse QoS requirements.

Task Launching

Task launching refer to the process of the initiation and execution of tasks inside nodes within the micro-cluster system. This process involves receiving task's input data, executing the task, and sending the results to the client node.

Task Scheduling

Task scheduling refers to the process of scheduling and controlling the execution of tasks on predetermined nodes. Task scheduling considers factors such as task dependencies, time sequence, task sequence, and resource availability. The goal is to optimise task ordering and timing to meet QoS requirements such as resource utilisation, execution time, and energy. Task scheduling is used interchangeably with task placement, task allocation, and resource allocation in the literature. However, task allocation determines which node is responsible for executing a task, while task scheduling determines the timing of task execution on the allocated node.

Task Offloading

Task offloading refers to the process of enabling intensive-computation IoT-applications, such as augmented reality, virtual reality, face recognition and multimedia delivery applications, to offload tasks to resource-rich edge or cloud nodes. IoT-devices are typically equipped with limited resources that are not capable enough to execute intensive-computations. Task offloading is used to support IoT-resource constrained devices to transfer their intensive-computational tasks to more resource-rich edge nodes or cloud nodes.

Bibliography

[1] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, 2009. [Online]. Available: https://doi.org/10.1109/MPRV.2009.82

[2] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing (MCC 2012)*. New York, NY, USA: Association for Computing Machinery, 2012, p. 13–16. [Online]. Available: https://doi.org/10.1145/2342509.2342513

[3] B. Varghese, P. Leitner, S. Ray, K. Chard, A. Barker, Y. Elkhatib, H. Herry, C.-H. Hong, J. Singer, F. P. Tso, E. Yoneki, and M.-F. Zhani, "Cloud futurology," *Computer*, vol. 52, no. 9, pp. 68–77, 2019. [Online]. Available: https://doi.org/10.1109/MC.2019.2895307

[4] B. Varghese, E. de Lara, A. Y. Ding, C.-H. Hong, F. Bonomi, S. Dustdar, P. Harvey, P. Hewkin, W. Shi, M. Thiele, and P. Willis, "Revisiting the arguments for edge computing research," *IEEE Internet Computing*, vol. 25, no. 5, pp. 36–42, 2021. [Online]. Available: https://doi.org/10.1109/MIC.2021.3093924

[5] A. J. Ferrer, J. M. Marquès, and J. Jorba, "Towards the decentralised cloud: Survey on approaches and challenges for mobile, ad hoc, and edge computing," *ACM Comput. Surv.*, vol. 51, no. 6, jan 2019. [Online]. Available: https://doi.org/10.1145/3243929

[6] S. J. Johnston, P. J. Basford, C. S. Perkins, H. Herry, F. P. Tso, D. Pezaros, R. D. Mullins, E. Yoneki, S. J. Cox, and J. Singer, "Commodity single board computer clusters and their applications," *Future Generation Computer Systems*, vol. 89, pp. 201–212, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X18301833

[7] T. Rausch, C. Avasalcai, and S. Dustdar, "Portable energy-aware cluster-based edge computers," in *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, 2018, pp. 260–272. [Online]. Available: https://doi.org/10.1109/SEC.2018.00026

[8] B. Qureshi, K. Kawlaq, A. Koubaa, B. Saeed, and M. Younis, "A commodity sbc-edge cluster for smart cities," in *2019 2nd International Conference on Computer Applications & Information Security (ICCAIS)*, 2019, pp. 1–6. [Online]. Available: https://doi.org/10.1109/CAIS.2019.8769500

[9] F. P. Tso, D. R. White, S. Jouet, J. Singer, and D. P. Pezaros, "The glasgow raspberry pi cloud: A scale model for cloud computing infrastructures," in *2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops*, 2013, pp. 108–112. [Online]. Available: https://doi.org/10.1109/ICDCSW.2013.25

[10] Y. Alhaizaey, J. Singer, and A. L. Michala, "Optimizing task allocation for edge micro-clusters in smart cities," in *2021 IEEE 22nd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2021, pp. 341–347. [Online]. Available: https://doi.org/10.1109/WoWMoM51794.2021.00062

[11] C.-H. Hong and B. Varghese, "Resource management in fog/edge computing: A survey on architectures, infrastructure, and algorithms," *ACM Comput. Surv.*, vol. 52, no. 5, sep 2019. [Online]. Available: https://doi.org/10.1145/3326066

[12] K. D. Kang, "Towards efficient real-time decision support at the edge," in *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, ser. SEC '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 419–424. [Online]. Available: https://doi.org/10.1145/3318216.3363380

[13] X. Liu, J. Yu, J. Wang, and Y. Gao, "Resource allocation with edge computing in iot networks via machine learning," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3415–3426, 2020. [Online]. Available: https://doi.org/10.1109/JIOT.2020.2970110

[14] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017. [Online]. Available: https://doi.org/10.1109/COMST.2017.2745201

[15] J. Winkowska, D. Szpilko, and S. Pejić, "Smart city concept in the light of the literature review," *Engineering Management in Production and Services*, vol. 11, no. 2, pp. 70–86, 2019. [Online]. Available: https://doi.org/10.2478/emj-2019-0012

[16] N. Chen, Y. Chen, E. Blasch, H. Ling, Y. You, and X. Ye, "Enabling smart urban surveillance at the edge," in *2017 IEEE International Conference on Smart Cloud (SmartCloud)*, 2017, pp. 109–119. [Online]. Available: https://doi.org/10.1109/SmartCloud.2017.24

[17] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016. [Online]. Available: https://doi.org/10.1109/JIOT.2016.2579198

[18] T. Rausch, C. Lachner, P. A. Frangoudis, P. Raith, and S. Dustdar, "Synthesizing plausible infrastructure configurations for evaluating edge computing systems," in *3rd USENIX Workshop on Hot Topics in Edge Computing (HotEdge 20)*. USENIX Association, 2020. [Online]. Available: https://www.usenix.org/conference/hotedge20/presentation/rausch

[19] J. McChesney, N. Wang, A. Tanwer, E. de Lara, and B. Varghese, "Defog: Fog computing benchmarks," in *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, ser. SEC '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 47–58. [Online]. Available: https://doi.org/10.1145/3318216.3363299

[20] A. Das, S. Patterson, and M. Wittie, "Edgebench: Benchmarking edge computing platforms," in *2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion)*, 2018, pp. 175–180. [Online]. Available: https://doi.org/10.1109/UCC-Companion.2018.00053

[21] S. O. Ogundoyin and I. A. Kamil, "Optimization techniques and applications in fog computing: An exhaustive survey," *Swarm and Evolutionary Computation*, vol. 66, p. 100937, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2210650221000985

[22] Y. Alhaizaey, J. Singer, and A. L. Michala, "Optimizing heterogeneous task allocation for edge compute micro clusters using pso metaheuristic," in *2022 Seventh International Conference on Fog and Mobile Edge Computing (FMEC)*, 2022, pp. 1–8. [Online]. Available: https://doi.org/10.1109/FMEC57183.2022.10062755

[23] D. Evans, "The internet of things. how the next evolution of the internet is changing everything," *Cisco White Paper*, 2011. [Online]. Available: https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf

[24] B. Varghese and R. Buyya, "Next generation cloud computing: New trends and research directions," *Future Generation Computer Systems*, vol. 79, pp. 849–861, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X17302224

[25] M. Satyanarayanan, W. Gao, and B. Lucia, "The computing landscape of the 21st century," in *Proceedings of the 20th International Workshop on Mobile*

Computing Systems and Applications, ser. HotMobile '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 45–50. [Online]. Available: https://doi.org/10.1145/3301293.3302357

[26] B. Varghese, "A History of the Cloud," *ITNOW*, vol. 61, no. 2, pp. 46–48, 05 2019. [Online]. Available: https://doi.org/10.1093/itnow/bwz049

[27] D. Kimovski, R. Mathá, J. Hammer, N. Mehran, H. Hellwagner, and R. Prodan, "Cloud, fog, or edge: Where to compute?" *IEEE Internet Computing*, vol. 25, no. 4, pp. 30–36, 2021. [Online]. Available: https://doi.org/10.1109/MIC.2021.3050613

[28] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599–616, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X08001957

[29] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A berkeley view of cloud computing," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28, Feb 2009. [Online]. Available: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html

[30] M. Taneja and A. Davy, "Resource aware placement of IoT application modules in fog-cloud computing paradigm," in *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 2017, pp. 1222–1228. [Online]. Available: https://doi.org/10.23919/INM.2017.7987464

[31] B. Varghese, N. Wang, S. Barbhuiya, P. Kilpatrick, and D. S. Nikolopoulos, "Challenges and opportunities in edge computing," in *2016 IEEE International Conference on Smart Cloud (SmartCloud)*, 2016, pp. 20–26. [Online]. Available: https://doi.org/10.1109/SmartCloud.2016.18

[32] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young, "Mobile edge computing—a key technology towards 5g," *ETSI white paper*, vol. 11, no. 11, pp. 1–16, 2015. [Online]. Available: https://infotech.report/Resources/Whitepapers/f205849d-0109-4de3-8c47-be52f4e4fb27_etsi_wp11_mec_a_key_technology_towards_5g.pdf

[33] K. Habak, M. Ammar, K. A. Harras, and E. Zegura, "Femto clouds: Leveraging mobile devices to provide cloud service at the edge," in *2015 IEEE 8th International Conference on Cloud Computing*, 2015, pp. 9–16. [Online]. Available: https://doi.org/10.1109/CLOUD.2015.12

[34] S. Wang, R. Urgaonkar, M. Zafer, T. He, K. Chan, and K. K. Leung, "Dynamic service migration in mobile edge-clouds," in *2015 IFIP Networking Conference (IFIP Networking)*, 2015, pp. 1–9. [Online]. Available: https://doi.org/10.1109/IFIPNetworking.2015.7145316

[35] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang, "A survey on the edge computing for the internet of things," *IEEE Access*, vol. 6, pp. 6900–6919, 2018. [Online]. Available: https://doi.org/10.1109/ACCESS.2017.2778504

[36] Y. Mansouri and M. A. Babar, "A review of edge computing: Features and resource virtualization," *Journal of Parallel and Distributed Computing*, vol. 150, pp. 155–183, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0743731520304317

[37] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, "A comprehensive survey on fog computing: State-of-the-art and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 416–464, 2018. [Online]. Available: https://doi.org/10.1109/COMST.2017.2771153

[38] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017. [Online]. Available: https://doi.org/10.1109/MC.2017.9

[39] P. J. Basford, S. J. Johnston, C. S. Perkins, T. Garnock-Jones, F. P. Tso, D. Pezaros, R. D. Mullins, E. Yoneki, J. Singer, and S. J. Cox, "Performance analysis of single board computer clusters," *Future Generation Computer Systems*, vol. 102, pp. 278–291, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X1833142X

[40] F. Liu, G. Tang, Y. Li, Z. Cai, X. Zhang, and T. Zhou, "A survey on edge computing systems and tools," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1537–1562, 2019. [Online]. Available: https://doi.org/10.1109/JPROC.2019.2920341

[41] S. J. Cox, J. T. Cox, R. P. Boardman, S. J. Johnston, M. Scott, and N. S. O'Brien, "Iridis-pi: a low-cost, compact demonstration cluster," *Cluster Computing*, vol. 17, no. 2, pp. 349–358, 2014. [Online]. Available: https://doi.org/10.1007/s10586-013-0282-7

[42] P. Abrahamsson, S. Helmer, N. Phaphoom, L. Nicolodi, N. Preda, L. Miori, M. Angriman, J. Rikkilä, X. Wang, K. Hamily, and S. Bugoloni, "Affordable and energy-efficient cloud computing clusters: The bolzano raspberry pi cloud cluster experiment," in *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*, vol. 2, 2013, pp. 170–175. [Online]. Available: https://doi.org/10.1109/CloudCom.2013.121

[43] M. E. Kryuchkov, A. V. Orlov, G. A. Mazurenko, A. B. Vavrenyuk, and Y. V. Timofeev, "Design of multipurpose computational cluster based on arm single-board computers," in *2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, 2018, pp. 322–324. [Online]. Available: https://doi.org/10.1109/EIConRus.2018.8317097

[44] E. Wilcox, P. Jhunjhunwala, K. Gopavaram, and J. Herrera, "Pi-crust: a raspberry pi cluster implementation," Texas A&M University: College Station, TX, USA, Tech. Rep., 2015. [Online]. Available: http://jorgehc.com/files/pi_crust_paper.pdf

[45] Y. Elkhatib, B. Porter, H. B. Ribeiro, M. F. Zhani, J. Qadir, and E. Rivière, "On using micro-clouds to deliver the fog," *IEEE Internet Computing*, vol. 21, no. 2, pp. 8–15, 2017. [Online]. Available: https://doi.org/10.1109/MIC.2017.35

[46] S. Sagkriotis, C. Anagnostopoulos, and D. P. Pezaros, "Energy usage profiling for virtualized single board computer clusters," in *2019 IEEE Symposium on Computers and Communications (ISCC)*, 2019, pp. 1–6. [Online]. Available: https://doi.org/10.1109/ISCC47284.2019.8969611

[47] G. Lewis, S. Echeverría, S. Simanta, B. Bradshaw, and J. Root, "Tactical cloudlets: Moving cloud computing to the edge," in *2014 IEEE Military Communications Conference*, 2014, pp. 1440–1446. [Online]. Available: https://doi.org/10.1109/MILCOM.2014.238

[48] D. Fernández-Cerero, J. Y. Fernández-Rodríguez, J. A. Álvarez García, L. M. Soria-Morillo, and A. Fernández-Montes, "Single-board-computer clusters for cloudlet computing in internet of things," *Sensors*, vol. 19, no. 13, p. 3026, Jul 2019. [Online]. Available: http://dx.doi.org/10.3390/s19133026

[49] R. Morabito, "Virtualization on internet of things edge devices with container technologies: A performance evaluation," *IEEE Access*, vol. 5, pp. 8835–8850, 2017. [Online]. Available: https://doi.org/10.1109/ACCESS.2017.2704444

[50] S. M. Blackburn, K. S. McKinley, R. Garner, C. Hoffmann, A. M. Khan, R. Bentzur, A. Diwan, D. Feinberg, D. Frampton, S. Z. Guyer, M. Hirzel, A. Hosking, M. Jump, H. Lee, J. E. B. Moss, A. Phansalkar, D. Stefanovik, T. VanDrunen, D. von Dincklage, and B. Wiedermann, "Wake up and smell the coffee: Evaluation methodology for the 21st century," *Commun. ACM*, vol. 51, no. 8, p. 83–89, aug 2008. [Online]. Available: https://doi.org/10.1145/1378704.1378723

[51] J. v. Kistowski, J. A. Arnold, K. Huppler, K.-D. Lange, J. L. Henning, and P. Cao, "How to build a benchmark," in *Proceedings of the 6th ACM/SPEC International*

Conference on Performance Engineering, ser. ICPE '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 333–336. [Online]. Available: https://doi.org/10.1145/2668930.2688819

[52] C. Pahl, S. Helmer, L. Miori, J. Sanin, and B. Lee, "A container-based edge cloud paas architecture based on raspberry pi clusters," in *2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*, 2016, pp. 117–124. [Online]. Available: https://doi.org/10.1109/W-FiCloud.2016.36

[53] D. Merkel, "Docker: Lightweight linux containers for consistent development and deployment," *Linux J.*, vol. 2014, no. 239, mar 2014. [Online]. Available: https://dl.acm.org/doi/10.5555/2600239.2600241

[54] E. A. Brewer, "Kubernetes and the path to cloud native," in *Proceedings of the Sixth ACM Symposium on Cloud Computing*, ser. SoCC '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 167. [Online]. Available: https://doi.org/10.1145/2806777.2809955

[55] F. Gand., I. Fronza., N. El Ioini., H. R. Barzegar., S. Azimi., and C. Pahl., "A fuzzy controller for self-adaptive lightweight edge container orchestration," in *Proceedings of the 10th International Conference on Cloud Computing and Services Science - CLOSER*, INSTICC. SciTePress, 2020, pp. 79–90. [Online]. Available: https://doi.org/10.5220/0009379600790090

[56] L. Miori, J. Sanin, and S. Helmer, "A platform for edge computing based on raspberry pi clusters," in *Data Analytics*. Springer International Publishing, 2017, pp. 153–159. [Online]. Available: https://doi.org/10.1007/978-3-319-60795-5_16

[57] S. Bourhnane, M. R. Abid, K. Zine-dine, N. Elkamoun, and D. Benhaddou, "Cluster of single-board computers at the edge for smart grids applications," *Applied Sciences*, vol. 11, no. 22, p. 10981, Nov 2021. [Online]. Available: http://dx.doi.org/10.3390/app112210981

[58] J. Hochstetler, R. Padidela, Q. Chen, Q. Yang, and S. Fu, "Embedded deep learning for vehicular edge computing," in *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, 2018, pp. 341–343. [Online]. Available: https://doi.org/10.1109/SEC.2018.00038

[59] D. Weikert, C. Steup, and S. Mostaghim, "Survey on multi-objective task allocation algorithms for IoT networks," *Sensors*, vol. 23, no. 1, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/1/142

[60] S. Singh and I. Chana, "A Survey on Resource Scheduling in Cloud Computing: Issues and Challenges," *Journal of Grid Computing*, vol. 14, no. 2, pp. 217–264, 2016. [Online]. Available: https://doi.org/10.1007/s10723-015-9359-2

[61] M. Ghobaei-Arani, A. Souri, and A. A. Rahmanian, "Resource Management Approaches in Fog Computing: a Comprehensive Review," *Journal of Grid Computing*, vol. 18, no. 1, pp. 1–42, 2020. [Online]. Available: https://doi.org/10.1007/s10723-019-09491-1

[62] M. S. Aslanpour, S. S. Gill, and A. N. Toosi, "Performance evaluation metrics for cloud, fog and edge computing: A review, taxonomy, benchmarks and standards for future research," *Internet of Things*, vol. 12, p. 100273, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2542660520301062

[63] H. Yao, C. Bai, M. Xiong, D. Zeng, and Z. Fu, "Heterogeneous cloudlet deployment and user-cloudlet association toward cost effective fog computing," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 16, p. e3975, 2017, e3975 cpe.3975. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.3975

[64] T. Rausch, A. Rashed, and S. Dustdar, "Optimized container scheduling for data-intensive serverless edge computing," *Future Generation Computer Systems*, vol. 114, pp. 259–271, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X2030399X

[65] O. Skarlat, M. Nardelli, S. Schulte, M. Borkowski, and P. Leitner, "Optimized IoT service placement in the fog," *Service Oriented Computing and Applications*, vol. 11, no. 4, pp. 427–443, 2017. [Online]. Available: https://doi.org/10.1007/s11761-017-0219-8

[66] S. Azimi., C. Pahl., and M. H. Shirvani., "Particle swarm optimization for performance management in multi-cluster IoT edge architectures," in *Proceedings of the 10th International Conference on Cloud Computing and Services Science - CLOSER*, INSTICC. SciTePress, 2020, pp. 328–337. [Online]. Available: https://doi.org/10.5220/0009391203280337

[67] S. Azimi, C. Pahl, and M. H. Shirvani, "Performance management in clustered edge architectures using particle swarm optimization," in *Cloud Computing and Services Science*, D. Ferguson, C. Pahl, and M. Helfert, Eds. Springer International Publishing, 2021, pp. 233–257. [Online]. Available: https://doi.org/10.1007/978-3-030-72369-9_10

[68] S. S. Gill, P. Garraghan, and R. Buyya, "Router: Fog enabled cloud based intelligent resource management approach for smart home IoT devices," *Journal of Systems and Software*, vol. 154, pp. 125–138, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121219300986

[69] B. M. Nguyen, H. Thi Thanh Binh, T. The Anh, and D. Bao Son, "Evolutionary algorithms to optimize task scheduling problem for the IoT based bag-of-tasks application in cloud–fog computing environment," *Applied Sciences*, vol. 9, no. 9, p. 1730, Apr 2019. [Online]. Available: http://dx.doi.org/10.3390/app9091730

[70] J. Cano, D. R. White, A. Bordallo, C. McCreesh, A. L. Michala, J. Singer, and V. Nagarajan, "Solving the task variant allocation problem in distributed robotics," *Autonomous Robots*, vol. 42, no. 7, pp. 1477–1495, 2018. [Online]. Available: https://doi.org/10.1007/s10514-018-9742-5

[71] O. Skarlat, M. Nardelli, S. Schulte, and S. Dustdar, "Towards QoS-aware fog service placement," in *2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC)*, 2017, pp. 89–96. [Online]. Available: https://doi.org/10.1109/ICFEC.2017.12

[72] L. Yin, J. Luo, and H. Luo, "Tasks scheduling and resource allocation in fog computing based on containers for smart manufacturing," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 10, pp. 4712–4721, 2018. [Online]. Available: https://doi.org/10.1109/TII.2018.2851241

[73] S. Venticinque and A. Amato, "A methodology for deployment of IoT application in fog," *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 5, pp. 1955–1976, 2019. [Online]. Available: http://dx.doi.org/10.1007/s12652-018-0785-4

[74] S. Wang, R. Urgaonkar, T. He, K. Chan, M. Zafer, and K. K. Leung, "Dynamic service placement for mobile micro-clouds with predicted future costs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 4, pp. 1002–1016, 2017. [Online]. Available: https://doi.org/10.1109/TPDS.2016.2604814

[75] N. Wang and B. Varghese, "Context-aware distribution of fog applications using deep reinforcement learning," *Journal of Network and Computer Applications*, vol. 203, p. 103354, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804522000236

[76] X. Liu, Z. Qin, and Y. Gao, "Resource allocation for edge computing in IoT networks via reinforcement learning," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–6. [Online]. Available: https://doi.org/10.1109/ICC.2019.8761385

[77] H. Mao, M. Alizadeh, I. Menache, and S. Kandula, "Resource management with deep reinforcement learning," in *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, ser. HotNets '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 50–56. [Online]. Available: https://doi.org/10.1145/3005745.3005750

[78] N. Talagala, S. Sundararaman, V. Sridhar, D. Arteaga, Q. Luo, S. Subramanian, S. Ghanta, L. Khermosh, and D. Roselli, "ECO: Harmonizing edge and cloud with ML/DL orchestration," in *USENIX Workshop on Hot Topics in Edge Computing (HotEdge 18)*. Boston, MA: USENIX Association, Jul. 2018. [Online]. Available: https://www.usenix.org/conference/hotedge18/presentation/talagala

[79] S. Bian, X. Huang, and Z. Shao, "Online task scheduling for fog computing with multi-resource fairness," in *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*, 2019, pp. 1–5. [Online]. Available: https://doi.org/10.1109/VTCFall.2019.8891573

[80] S. Ningning, G. Chao, A. Xingshuo, and Z. Qiang, "Fog computing dynamic load balancing mechanism based on graph repartitioning," *China Communications*, vol. 13, no. 3, pp. 156–164, 2016. [Online]. Available: https://doi.org/10.1109/CC.2016.7445510

[81] R. Beraldi, A. Mtibaa, and H. Alnuweiri, "Cooperative load balancing scheme for edge computing resources," in *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*, 2017, pp. 94–100. [Online]. Available: https://doi.org/10.1109/FMEC.2017.7946414

[82] D. Puthal, M. S. Obaidat, P. Nanda, M. Prasad, S. P. Mohanty, and A. Y. Zomaya, "Secure and sustainable load balancing of edge data centers in fog computing," *IEEE Communications Magazine*, vol. 56, no. 5, pp. 60–65, 2018. [Online]. Available: https://doi.org/10.1109/MCOM.2018.1700795

[83] X. He, Z. Ren, C. Shi, and J. Fang, "A novel load balancing strategy of software-defined cloud/fog networking in the Internet of Vehicles," *China Communications*, vol. 13, no. Supplement2, pp. 140–149, 2016. [Online]. Available: https://doi.org/10.1109/CC.2016.7833468

[84] A. Barbalace, M. L. Karaoui, W. Wang, T. Xing, P. Olivier, and B. Ravindran, "Edge computing: The case for heterogeneous-ISA container migration," in *Proceedings of the 16th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments*, ser. VEE '20. New York, NY, USA:

Association for Computing Machinery, 2020, p. 73–87. [Online]. Available: https://doi.org/10.1145/3381052.3381321

[85] A. A. Majeed, P. Kilpatrick, I. Spence, and B. Varghese, "Modelling fog offloading performance," in *2020 IEEE 4th International Conference on Fog and Edge Computing (ICFEC)*, 2020, pp. 29–38. [Online]. Available: https://doi.org/10.1109/ICFEC50348.2020.00011

[86] X. Sun and N. Ansari, "Latency aware workload offloading in the cloudlet network," *IEEE Communications Letters*, vol. 21, no. 7, pp. 1481–1484, 2017. [Online]. Available: https://doi.org/10.1109/LCOMM.2017.2690678

[87] H.-J. Jeong, C. H. Shin, K. Y. Shin, H.-J. Lee, and S.-M. Moon, "Seamless offloading of web app computations from mobile device to edge clouds via html5 web worker migration," in *Proceedings of the ACM Symposium on Cloud Computing (SoCC 2019)*. New York, NY, USA: Association for Computing Machinery, 2019, p. 38–49. [Online]. Available: https://doi.org/10.1145/3357223.3362735

[88] C. N. L. Tan, C. Klein, and E. Elmroth, "Location-aware load prediction in edge data centers," in *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*, 2017, pp. 25–31. [Online]. Available: https://doi.org/10.1109/FMEC.2017.7946403

[89] P. Wiesner and L. Thamsen, "LEAF: Simulating large energy-aware fog computing environments," in *2021 IEEE 5th International Conference on Fog and Edge Computing (ICFEC)*, 2021, pp. 29–36. [Online]. Available: https://doi.org/10.1109/ICFEC51620.2021.00012

[90] C. Jiang, T. Fan, H. Gao, W. Shi, L. Liu, C. Cérin, and J. Wan, "Energy aware edge computing: A survey," *Computer Communications*, vol. 151, pp. 556–580, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S014036641930831X

[91] Q. Li and Y. Guo, "Optimization of resource scheduling in cloud computing," in *2010 12th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, 2010, pp. 315–320. [Online]. Available: https://doi.org/10.1109/SYNASC.2010.8

[92] Z. Ye, X. Zhou, and A. Bouguettaya, "Genetic algorithm based QoS-aware service compositions in cloud computing," in *International Conference on Database Systems for Advanced Applications*, J. X. Yu, M. H. Kim, and R. Unland, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 321–334. [Online]. Available: https://doi.org/10.1007/978-3-642-20152-3_24

[93] Z.-H. Zhan, X.-F. Liu, Y.-J. Gong, J. Zhang, H. S.-H. Chung, and Y. Li, "Cloud computing resource scheduling and a survey of its evolutionary approaches," *ACM Comput. Surv.*, vol. 47, no. 4, jul 2015. [Online]. Available: https://doi.org/10.1145/2788397

[94] Y. Wang, Y. Xia, and S. Chen, "Using integer programming for workflow scheduling in the cloud," in *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, 2017, pp. 138–146. [Online]. Available: https://doi.org/10.1109/CLOUD.2017.26

[95] M. Harman, K. Lakhotia, J. Singer, D. R. White, and S. Yoo, "Cloud engineering is search based software engineering too," *Journal of Systems and Software*, vol. 86, no. 9, pp. 2225–2241, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121212002853

[96] S. Luke, *Essentials of Metaheuristics*, 2009. [Online]. Available: https://cs.gmu.edu/~sean/book/metaheuristics/Essentials.pdf

[97] M. Harman and B. F. Jones, "Search-based software engineering," *Information and Software Technology*, vol. 43, no. 14, pp. 833–839, 2001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584901001896

[98] H. Singh, S. Tyagi, P. Kumar, S. S. Gill, and R. Buyya, "Metaheuristics for scheduling of heterogeneous tasks in cloud computing environments: Analysis, performance evaluation, and future directions," *Simulation Modelling Practice and Theory*, vol. 111, p. 102353, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1569190X21000678

[99] D. A. Alboaneen, H. Tianfield, and Y. Zhang, "Metaheuristic approaches to virtual machine placement in cloud computing: A review," in *2016 15th International Symposium on Parallel and Distributed Computing (ISPDC)*, 2016, pp. 214–221. [Online]. Available: https://doi.org/10.1109/ISPDC.2016.37

[100] D. Wolpert and W. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997. [Online]. Available: https://doi.org/10.1109/4235.585893

[101] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and experience*, vol. 41, no. 1, pp. 23–50, 2011. [Online]. Available: https://doi.org/10.1002/spe.995

[102] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments," *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017. [Online]. Available: https://doi.org/10.1002/spe.2509

[103] A. Luckow, K. Rattan, and S. Jha, "Exploring task placement for edge-to-cloud applications using emulation," in *2021 IEEE 5th International Conference on Fog and Edge Computing (ICFEC)*, 2021, pp. 79–83. [Online]. Available: https://doi.org/10.1109/ICFEC51620.2021.00019

[104] B. Varghese, N. Wang, D. Bermbach, C.-H. Hong, E. D. Lara, W. Shi, and C. Stewart, "A survey on edge performance benchmarking," *ACM Comput. Surv.*, vol. 54, no. 3, apr 2021. [Online]. Available: https://doi.org/10.1145/3444692

[105] L. Perron and V. Furnon, "Or-tools," Google. [Online]. Available: https://developers.google.com/optimization/

[106] A. Salman, I. Ahmad, and S. Al-Madani, "Particle swarm optimization for task assignment problem," *Microprocessors and Microsystems*, vol. 26, no. 8, pp. 363–371, 2002. [Online]. Available: https://doi.org/10.1016/S0141-9331(02)00053-4

[107] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-international conference on neural networks*, vol. 4. IEEE, 1995, pp. 1942–1948. [Online]. Available: https://doi.org/10.1109/ICNN.1995.488968

[108] K. Deb and K. Deb, *Multi-objective Optimization*. Boston, MA: Springer US, 2014, pp. 403–449. [Online]. Available: https://doi.org/10.1007/978-1-4614-6940-7_15

[109] K. Toczé, J. Lindqvist, and S. Nadjm-Tehrani, "Characterization and modeling of an edge computing mixed reality workload," *Journal of Cloud Computing*, vol. 9, no. 1, p. 46, 2020. [Online]. Available: https://doi.org/10.1186/s13677-020-00190-x

[110] Q. Yang, R. Jin, N. Gandhi, X. Ge, H. A. Khouzani, and M. Zhao, "EdgeBench: A Workflow-based Benchmark for Edge Computing," 2020. [Online]. Available: http://arxiv.org/abs/2010.14027