HETEROGENEITY-AWARE TASK ALLOCATION IN MOBILE AD HOC CLOUD

IBRAR YAQOOB

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2017

HETEROGENEITY-AWARE TASK ALLOCATION IN MOBILE AD HOC CLOUD

IBRAR YAQOOB

THESIS SUBMITTED IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2017

UNIVERSITY OF MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: **Ibrar Yaqoob**

Registration/Matric No: **WHA130040**

Name of Degree: **Doctor of Philosophy**

Title of Thesis: **Heterogeneity-aware task allocation in mobile ad hoc cloud**

Field of Study: **Mobile Cloud Computing (Computer Science)**

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;
(2) This Work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every rights in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature Date:

Subscribed and solemnly declared before,

Witness's Signature Date:

Name:

Designation:

ABSTRACT

Mobile Ad Hoc Cloud (MAC) enables the use of a multitude of proximate resource-rich mobile devices to provide computational services in the vicinity. MAC is a candidate blueprint for future compute-intensive applications with the aim of delivering high functionalities and a rich experience to mobile users. However, inattention to mobile device resources and operational heterogeneity-measuring parameters, such as CPU speed, number of cores, and workload, when allocating task in MAC, causes inefficient resource utilization that prolongs task execution time and consumes large amounts of energy. Task execution is remarkably degraded because the longer execution time and high energy consumption impede the optimum use of MAC. In this study, we minimize execution time and energy consumption by proposing heterogeneity-aware task allocation solutions for MAC-based compute-intensive tasks. Results reveal that incorporation of the heterogeneity-measuring parameters guarantees a shorter execution time and reduces the energy consumption of the compute-intensive tasks in MAC. We develop a mathematical model to validate the proposed solutions' empirical results. In comparison with random-based task allocation (RM), the proposed five solutions based on CPU speed (SO), number of cores (CO), workload (WO), CPU speed and workload (SW), and CPU speed, core, and workload (SCW) reduce execution time up to 56.72%, 53.12%, 56.97%, 61.23%, and 71.55%, respectively. In addition, these heterogeneity-aware task allocation solutions save energy up to 69.78%, 69.06%, 68.25%, 67.26%, and 57.33%, respectively. Furthermore, we apply Mann-Whitney U test and Vargha and Delaney's A_{12} statistics to find the significance of differences between the results. Our findings from both tests reveal that the proposed solutions have significant statistical and practical differences compared to RM-based solution. For this reason, the proposed solutions significantly improve tasks' execution performance, which can increase the optimum use of MAC.

ABSTRAK

Mobile Ad Hoc Cloud (MAC) membolehkan penggunaan pelbagai peranti yang kaya dengan sumber proksimat mudah alih yang menyediakan perkhidmatan pengkomputeran kepada pengguna mudah alih di persekitaran pelaksanaan tugas intensif pengiraan. MAC disifatkan sebagai calon cetakan biru untuk aplikasi intensif pengiraan masa depan yang bertujuan untuk menyampaikan fungsian tinggi dan pengalaman impresif beraneka untuk pengguna mudah alih. Walau bagaimanapun, kecuaian sumber peranti mudah alih dan kepelbagaian pengendalian, seperti kelajuan CPU, bilangan teras, dan beban kerja, semasa memperuntukkan tugas dalam MAC, menyebabkan penggunaan sumber yang tidak cekap yang memanjangkan masa pelaksanaan tugas dan menggunakan lebih tenaga. Prestasi pelaksanaan tugas ketara amat merosot kerana masa pelaksanaan yang lebih panjang dan penggunaan tenaga yang tinggi yang menghalang realisasi MAC. Dalam kajian ini, kami menyasarkan untuk meminimumkan masa pelaksanaan dan penggunaan tenaga dengan mencadangkan mekanisme peruntukan tugas sedar-keheterogenan untuk tugas-tugas intensif pengiraan berasaskan MAC. Analisis penyelesaian yang dicadangkan menunjukkan bahawa penggabungan keheterogenan yang mengukur parameter menjamin pengurangan dalam masa pelaksanaan dan penggunaan tenaga bagi tugas intensif pengiraan dalam MAC. Kami mengesahkan dan menentusahkan cadangan penyelesaian kami masing-masing, menggunakan pemodelan matematik dan perbandingan. Berbanding dengan peruntukan tugas secara rawak, cadangan lima penyelesaian yang berdasarkan kepada hanya kelajuan CPU, hanya teras, hanya beban kerja, kelajuan campur beban kerja, dan kelajuan CPU campur teras dan beban kerja, mengurangkan masa pelaksanaan sehingga masing-masing 56.72%, 53.12%, 56.97%, 61.23%, dan 71.55%. Di samping itu, penyelesaian peruntukan tugas sedar-keheterogenan membantu menjimatkan tenaga masing-masing sehingga 69.78%, 69.06%, 68.25%, 67.26%, dan 57.33%. Tambahan

pula, kami menggunakan dua ujian statistik yang terkenal, iaitu ujian statistik Mann-Whitney U dan Vargha & Delaney A_{12} untuk mengetahui kepentingan perbezaan di antara keputusan. Penemuan kami dari hasil kedua-dua ujian mendedahkan bahawa penyelesaian yang dicadangkan mempunyai perbezaan statistik dan praktikal ketara berbanding dengan penyelesaian berasaskan rawak. Oleh itu, keputusan penilaian ini, menyokong bagi menerima pakai cadangan penyelesaian kami boleh meningkatkan prestasi pelaksanaan tugas yang meningkatkan kebolehgunaan MAC.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisors Professor Dr. Abdullah Gani and Associate Professor Salimah Mokhtar, for their invaluable suggestions, support, and guidance throughout my doctoral study. In addition, I would like to thank Dr. Ejaz Ahmed for his continuous support and guidance. Moreover, I am in deeply indebted to my fellow friends, Ibrahim Abaker Targio Hashem, Abdullah Yousafzai, Dr. Syed Adeel Ali Shah, Abdelmuttlib Ibrahim Abdalla Ahmed, and Ali Abo-Hammad for thought-provoking discussions. I would also like to express my special appreciations to Dr. Muhammad Imran, Dr. Anjum Naveed, and Dr. Muhammad Zubair Khan for their continuous support. Their presence added a significant impact on my study and achievements.

I would also like to thank Bright Sparks Unit and Faculty of Computer Science and Information Technology, University of Malaya, for offering me a prestigious research scholarship throughout my doctoral study. Lastly, I would like to include a special note of appreciation to my parents and siblings. Without their precious spiritual support and prayers, it would never have been possible to reach this stage of life.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

3G	THIRD GENERATION
ACM	ASSOCIATION FOR COMPUTING MACHINERY
CCS	CONNECTED AD HOC CLOUDLET SERVICE
CO	CORE-BASED SOLUTION
DT	DATA TRACE
IEEE	INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS
MAC	MOBILE AD HOC CLOUD
MANET	MOBILE AD HOC NETWORK
MCC	MOBILE CLOUD COMPUTING
MIPS	MILLION INSTRUCTIONS PER SECOND
OCS	OPPORTUNISTIC AD HOC CLOUDLET SERVICE
QoS	QUALITY OF SERVICE
RCS	REMOTE CLOUD SERVICE
RM	RANDOM-BASED SOLUTION
SCW	CPU SPEED, CORE, AND WORKLOAD (BASED SOLUTION)
SO	CPU SPEED-BASED SOLUTION
SW	CPU SPEED AND WORKLOAD (BASED SOLUTION)
TMC	TRUST MANAGEMENT SYSTEM
US	UNITED STATES
VS	VERSUS
WO	WORKLOAD-BASED SOLUTION
WiFi	WIRELESS FIDELITY

CHAPTER 1: INTRODUCTION

This chapter presents an overview of the research carried out in this thesis. First, we provide the background information to familiarize the readers with Mobile Ad Hoc Cloud (MAC) paradigm. The motivation to undertake the research work is described. We state the research problem investigated and addressed. The aim and objectives of the research are outlined. The research methodology proposed to address the problem is discussed. Lastly, organization of the thesis is described.

The chapter is organized into six sections. In Section 1.1, we discuss the background of MAC. Section 1.2 presents the motivation for this research. Section 1.3 highlights the research gap, briefly explains the problem of task allocation and summarizes the statement of problem. Section 1.4 enlists the research objectives of the study conducted in this thesis. Section 1.5 summarizes the methodology followed in the research. Finally, Section 1.6 sketches the layout for the rest of the thesis.

1.1 Background

This section provides a brief discussion on cloud computing and Mobile Cloud Computing (MCC) that leads to the MAC. The purpose is to familiarize the readers with MAC paradigm.

1.1.1. Cloud Computing

Cloud computing is a paradigm for enabling ubiquitous, convenient and on-demand network access to a shared pool of configured computing resources (e.g., networks, server, storage, application, and services). These resources can be rapidly provisioned and released with minimal management effort or service provider interaction (Mell et al., 2009). Figure 1.1 depicts a typical environment of cloud computing (Yaqoob et al., 2016). Cloud computing provides users with different capabilities to store and process their data in third-party data centers. It focuses on optimizing the effectiveness of the dynamically shared resources in an on-demand

manner (Armbrust et al., 2010; Buyya et al., 2009). For instance, cloud computing resources allocated to North American users during their working hours with a specific application (e.g., a web server) can be reallocated to their European counterparts during respective job timings with a different application (e.g., email).

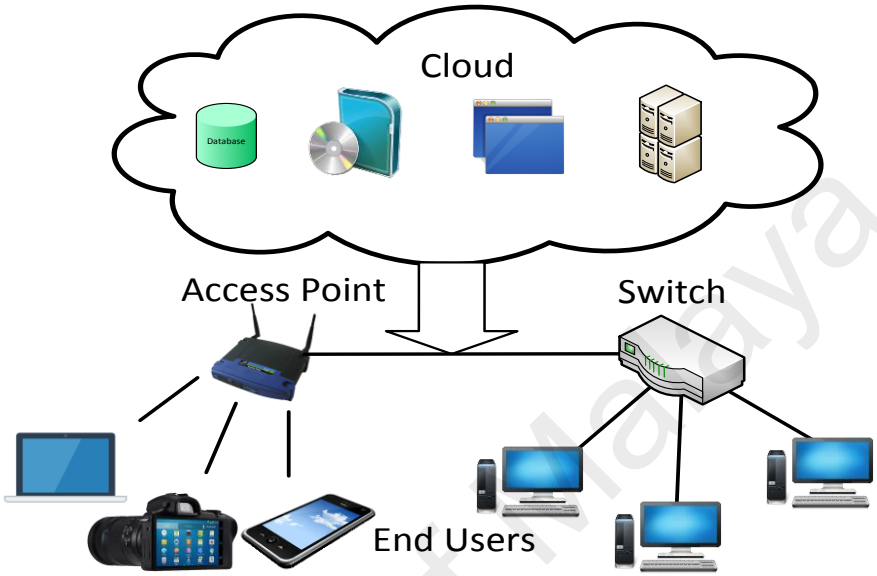

Figure 1.1: Illustration of cloud computing

1.1.2. Mobile Cloud Computing

MCC has emerged as a distributed computing paradigm that enables the execution of compute-intensive applications by augmenting the resources of constrained mobile devices. Figure 1.2 articulates a simplified environment of MCC (Ahmed *et al.*, 2015). MCC alleviates resource limitations of mobile devices by using various augmentation strategies, such as storage augmentation, energy augmentation, screen augmentation, and application processing augmentation (Bahl *et al.*, 2012). It has three types of computing models to augment the resources of mobile devices: (a) remote cloud, (b) server-based cloudlet, and (c) mobile ad hoc cloudlet (Pedersen *et al.*, 2012; Satyanarayanan *et al.*, 2009; Shaukat *et al.*, 2015). In the case of a remote cloud-based computing model, mobile devices act as a thin client while accessing the cloud through wireless technologies. This model can provide many benefits, such as low computation time, high computation power, and on-demand availability of resources. However, the application suffers from high latency, jitter, and packet losses (Abolfazli *et al.*, 2014). In

the case of server-based cloudlet, mobile devices offload their computations to the locally available resource-rich devices, such as servers. In the absence of any server-based cloudlet, the mobile devices share their resources to enable the execution of compute-intensive applications. This computing model is known as MAC (Guo *et al.*, 2016).

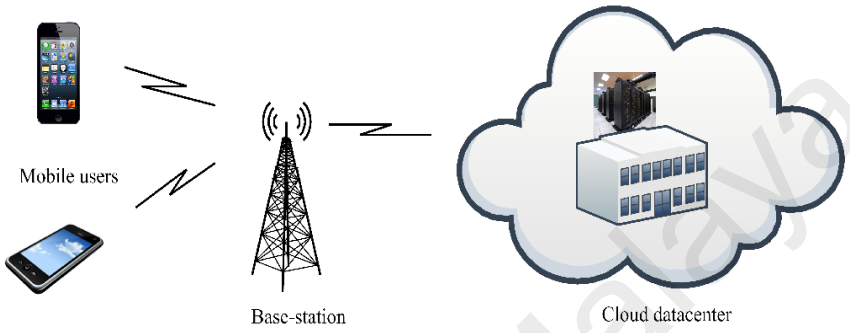

Figure 1.2: A simplified example of MCC

1.1.3. Mobile Ad Hoc Cloud

Noticeable advances in MCC have paved the way towards new computing paradigm called MAC. MAC is a group of mobile devices in the vicinity that share their resources with each other by taking some incentives, as shown in figure 1.3 (Yaqoob et al., 2016). MAC is a new type of MCC. It is usually deployed over Mobile Ad Hoc Networks (MANETs) which allows the execution of compute-intensive applications by leveraging the resources of other mobile devices (Zaghdoudi et al., 2015). As an alternative solution, MAC is an emerging paradigm that mitigates several bottlenecks of server-based cloudlets, such as longer delay and low throughput. Moreover, MAC offers a viable solution for a mobile device to execute an application when there is no or weak wireless Internet connection to the remote cloud or the nearby server-based cloudlet is not available (Loke et al., 2015). In MAC, mobile devices are expected to manage the cloud, authenticate the users, monitor the resources, and schedule the tasks besides executing the application. Such additional functionalities consume mobile device energy

and processor cycles. Finally, local stationary devices, such as personal computers, set-top boxes can also become members of MAC.

Figure 1.3: A typical MAC environment

1.2 Research Motivation

Recent studies reveal that mobile devices are a great source of idle resources. It is reported that per hour usage of the mobile phones is less than 25% (Falaki *et al.*, 2010). Another research indicates that mobile supercomputing is not always the solution because of the high cost of 3G networks (Miluzzo *et al.*, 2012). WiFi connectivity is always not present (e.g., less than 20% connectivity is not present in US cities) (Balasubramanian *et al.*, 2010). The study done by Golchay *et al.* (2016) reveals that most of the devices surrounding users in a nearby future will be mobile devices, and able to perform the processing of compute-intensive tasks smoothly. Hence, these statistics provide a strong motivation for carrying research in MAC paradigm as performing task execution using local mobile device resources will become a core component of the future computing landscape.

In MAC, mobile device resources are not free and some incentives need to be paid for lending the computing services from the nearby mobile devices (Miluzzo *et al.*, 2012). Therefore, performing task allocation in MAC without considering the mobile device resources and operational context can be very expensive in terms of incentives.

The task allocation must be done in such a way that it ensures efficient resource utilization. The efficient utilization of the mobile device resources not only minimize and stabilize the incentive cost, but also improves task execution experience of the user by minimizing the task execution time and energy consumption that leads to the success of MAC. Thus, these factors motivate to carry research in the MAC with respect to heterogeneity-aware task allocation.

MAC applications where heterogeneity-aware task allocation solutions can play an important role are numerous: gaming, unmanned vehicular surveillance, battlefields, disaster recovery, and vehicular safety. In gaming, players share the resources to run the game in the distributed manner. The devices connectivity is considered stable as the players tend to stay in the same place while playing the game. In the unmanned vehicular surveillance, a group of unmanned vehicles forms the MAC to monitor the area and run the information fusion algorithms. Similarly, the battlefields, disaster recovery, and vehicular safety applications can also be run on the group of cloud provider nodes to perform the compute-intensive tasks on the resource-constrained mobile devices.

1.3 Statement of Problem

The devices forming MAC usually have different specifications and operational contexts. These devices have a different level of workload running on their background. The higher workload on the device increases the execution time in the MAC. The CPU speed and number of cores of the mobile devices can also vary which affect the application performance and lifetime of the MAC.

The execution of larger size task on a device that has high specification can reduce the execution time of the task compared to low specification devices. The complexity of the task also affects its execution time. The task with high complexity takes more time in execution.

The existing random-based task allocation solution does not incorporate the mobile device resource and operational heterogeneity during task allocation process. However, task allocation is performed in a random manner. In addition, random-based task allocation does not consider the operational context of the mobile devices. Therefore, there is a need of heterogeneity-aware task allocation algorithms to minimize the execution time and energy consumption in MAC.

Based on this discussion, it can be argued that the problem of inefficient task allocation has not been addressed. Thus, the highlighted research gap leads to the following statement of problem.

MAC is a group of mobile devices in the vicinity that share their resources with each other to execute compute-intensive tasks. However, negligence of mobile device resources and operational heterogeneity-measuring parameters, such as CPU speed, number of cores, and workload, when allocating task in MAC, causes inefficient resource utilization that prolongs task execution time and consumes large amounts of energy. Task execution performance is remarkably degraded because of the longer execution time and high energy consumption that impede the realization of MAC.

1.4 Statement of Objectives

This research work aims to address the problem of inefficient task allocation that results in longer execution time and high energy consumption. The following objectives are defined to achieve the aim of this research.

1. To review the state-of-the-art on MAC for obtaining insights with respect to task allocation issue.
2. To investigate the impact of heterogeneity-measuring parameters and random-based task allocation on task execution performance.

3. To propose and develop five heterogeneity-aware task allocation solutions for minimizing the task execution time and energy consumption, and devise a mathematical model.

4. To evaluate the performance of the proposed heterogeneity-aware task allocation solutions with random-based task allocation in terms of execution time and energy consumption, and validate the developed mathematical model.

1.5 Proposed Research Methodology

The research work is divided into four phases, as shown in figure 1.4 to achieve the set of objectives defined in Section 1.4. Each research phase is targeted to achieve an objective. In the first phase, we review the state-of-the-art research carried out in the MAC domain to identify the research gap. We investigate several problems inhibiting the adoption of MAC and review the corresponding solutions by classifying the literature. The investigation reveals that the research in MAC is in its infancy and many issues associated with this domain are remain to be solved. Among these issues, we identify one task allocation issue because the random-based task allocation solution does not enable the controller to consider the resource and operational heterogeneity of mobile devices while allocating tasks in MAC.

The second phase of research involves investigating the research problem by conducting experiments on real mobile devices. In this context, a multithreaded matrix multiplication application is developed to use as compute-intensive tasks. The impact of mobile device resource and operational heterogeneity, such as CPU speed, number of cores, and workload is measured on task execution performance to establish the research problem.

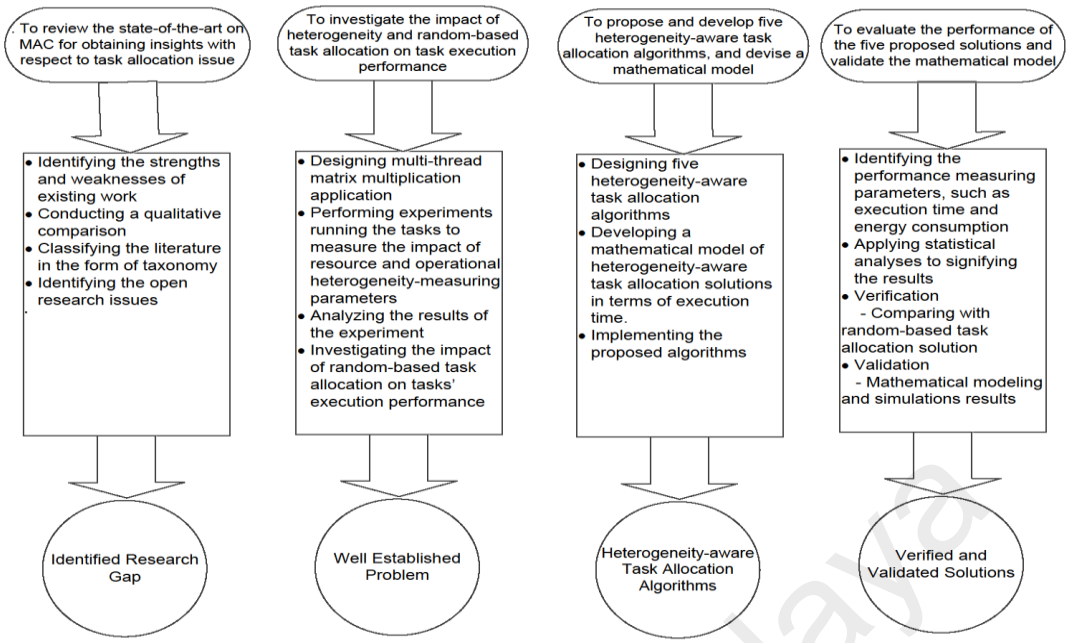

Figure 1.4: Proposed research methodology

Five heterogeneity-aware task allocation algorithms are proposed in the third phase of the research. Implementation of the solutions is also carried out in this phase. The proposed solutions aim to minimize the execution time of the compute-intensive tasks and save the energy consumption in MAC. The execution time and energy consumption are minimized by incorporating heterogeneity-measuring parameters. A mathematical model is developed to validate the execution time results obtained from the proposed solutions.

Evaluation of the implemented algorithms and validation of the developed mathematical model are performed in the fourth phase. The developed multi-threaded matrix multiplication application is tested with different specifications of the mobile devices. The mathematical model is validated against the empirical results obtained from five proposed heterogeneity-aware task allocation solutions. Furthermore, statistical analyses are applied to signifying the results. Lastly, comparison of the five proposed solutions is done with the random-based task allocation in MAC paradigm.

1.6 Thesis Layout

Table 1.1 presents organization of the thesis. This thesis is organized into seven chapters as follows:

Table 1.1: Thesis Layout

What?	Why?	How?
Introduction	(a) Highlighting the reason for the research (b) Stating the research problem and presenting the research objectives (c) Discussing the thesis organization	(a) Stating the rationale for undertaking the research (b) Formally writing the statement of problem and statement of objectives
Literature Review: Mobile Ad Hoc Cloud	(a) Investigating the state-of-the-art research in MAC for identifying the research problem	(a) Analyzing the critical aspects of the existing solutions (b) Classifying and categorizing the literature by devising two taxonomies (c) Performing comparison based on objectives, strengths, and weaknesses (d) Identifying the open research issues
Problem Analysis	(a) Establishing the identified research problem to understand the impact of the problem	(a) Conducting empirical study to analyze the impact of heterogeneity-measuring parameters on task execution performance in MAC paradigm (b) Analyzing the impact of the random-based task allocation on task execution performance
Heterogeneity-aware Task Allocation Algorithms	(a) Giving the clear understanding of the proposed heterogeneity-aware task allocation solutions to the readers (b) Measuring reliability of the proposed solutions	(a) Presenting the pseudo-codes of the proposed five algorithms (b) Discussing the mathematical model of the proposed five solutions (c) Highlighting the distinct features of the proposed five solutions to measure their effectiveness
Evaluation	(a) Presenting the collected data and discussing the statistical methods used to measure the accuracy of the collected data (b) Finding whether or not differences between the results obtained from proposed and random-based task allocation are significant	(a) Reporting the collected data (b) Explaining the tools used for evaluating the proposed solutions (c) Applying statistical methods on the collected data to find the statistical and practical differences between the results obtained from proposed solutions' and random-based task allocation (c) Analyzing the differences between the mathematical model and proposed solutions results through various statistical methods/tests.
Results and Discussion	(a) Highlighting the trustworthiness and effectiveness of the proposed heterogeneity-aware task allocation solutions by validating and analyzing the simulation results	(a) Discussing the insights obtained from the proposed solutions results (b) Comparing the performance of heterogeneity-aware task allocation solutions with random-based task allocation (c) Discussing the statistical significance of the proposed solutions' results (d) Validating the mathematical model by comparing it with the results obtained from the proposed solutions
Conclusion	(a) Summarizing the findings of the research work and highlighting the importance and deficiencies of the proposed solutions	(a) Reporting the re-examination of the research objectives b) Summarizing the findings of the research work and highlighting the significance of the proposed solutions (c) Discussing the limitations of the research work and suggesting future directions of the research

Chapter 2 presents a review of the state-of-the-art research carried out in the MAC domain. We analyze several obstacles to the adoption of MAC and review the solutions by devising a taxonomy. Moreover, MAC roots are analyzed and taxonomized as architectural components, applications, objectives, characteristics, execution model, scheduling type, formation technologies, and node types. The similarities and differences among existing proposed solutions by highlighting the advantages and disadvantages are also investigated. We also compare the literature based on objectives. Furthermore, the chapter discusses several new principles for the deployment of MAC. Lastly, several open research issues are presented. Among these issues, we identify one as a research problem.

Chapter 3 presents the experimental study to analyze the impact of mobile device resource and operational heterogeneity on task execution performance in MAC. This chapter aims at establishing the research problem. The effect of resource and operational heterogeneity is investigated with respect to different aspects as follows: (a) CPU speed, (b) number of cores, and (c) workload. Moreover, the impact of random-based task allocation on task execution performance is also investigated.

Chapter 4 presents five heterogeneity-aware algorithms that aim to solve the issue of longer execution time and high energy consumption in MAC. These algorithms are presented in form of pseudo-codes in the chapter. The distinctive features of the proposed algorithms are also discussed. Furthermore, a mathematical model of the solutions in terms of execution time is presented.

Chapter 5 presents the data collected for the evaluation of the proposed solutions. It explains the tools used for evaluating the proposed solutions, performance measuring parameters, and the statistical methods that help to analyze the accuracy of the collected data obtained from the mathematical model and proposed solutions. In

addition, statistical and practical significance of the results compared to random-based task allocation is also discussed in this chapter.

Chapter 6 discusses the effectiveness of the proposed solutions by analyzing the collected results reported in Chapter 5. It analyzes the different aspects of task allocation regarding execution time and energy consumption. Moreover, this chapter provides a discussion on the validation of the mathematical model with the simulation results. Furthermore, the performance of the proposed solutions is compared with the random-based task allocation in terms of execution time and energy consumption.

Chapter 7 concludes the thesis by reflecting on the sets of objectives. It summarizes the findings of the research work, highlights the significance of the proposed solutions, discusses the limitations of the study, and recommends future directions of the research.

CHAPTER 2: MOBILE AD HOC CLOUD

This chapter aims to identify the most significant of MAC's shortcomings. To achieve this, we investigate the recent research efforts directed at MAC. We analyze several problems hindering the adoption of MAC and review corresponding solutions by devising a taxonomy. MAC roots are analyzed and taxonomized as architectural components, applications, objectives, characteristics, execution model, scheduling type, communication technologies and nodes types. The similarities and differences among proposed solutions are analyzed in terms of their advantages and disadvantages. We also compare the literature based on objectives. Furthermore, the chapter advocates that the problems stem from the intrinsic characteristics of MAC by identifying several new principles. Finally, several open research issues are presented for further investigation.

The chapter is organized into five sections. In Section 2.1, we investigate the latest research conducted in the MAC domain. Section 2.2 discusses the taxonomy of MAC. In Section 2.3, we identify and discuss the key principles for successful deployment of MAC. Section 2.4 discusses open research issues in realizing the vision of MAC. Finally, we provide concluding remarks in Section 2.5.

2.1 State-of-the-art in MAC

MAC is in its infancy and a very limited literature is available on the subject. The purpose of this section is to discuss the research carried out in MAC domain. In this context, we investigate several problems inhibiting the adoption of MAC and review corresponding solutions by devising a taxonomy shown in figure 2.1 (Yaqoob *et al.*, 2016). Furthermore, we compare the existing solutions in the context of task offloading, task scheduling and allocation, MAC formation, security and privacy, mobility and incentives, and resource management in tables 2.1-2.6, respectively (Yaqoob *et al.*, 2016).

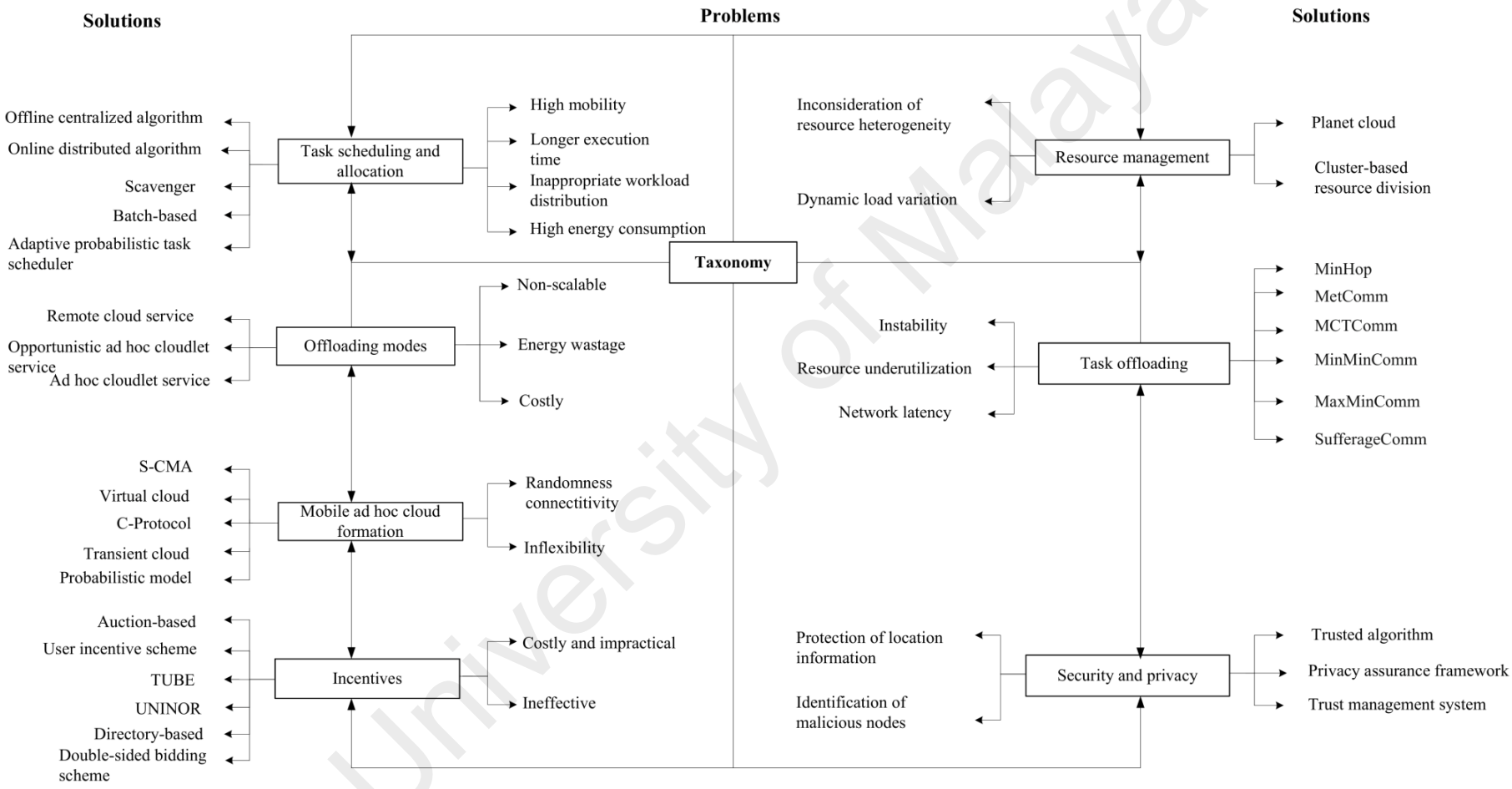

Figure 2.1: Context-based literature taxonomy

2.1.1. Task Offloading

The study done by B. Li *et al.* (2015) focused on the decision problem about how to offload computation-intensive applications in MAC. To address the problem, a set of online and batch scheduling heuristics, namely MinHop, MetComm, MCTComm, MinMinComm, MaxMinComm, and SufferageComm were proposed that offload the independent tasks among nodes in a dynamic manner. The MinHop heuristic assigns a task based on a minimum number of hops from the client node. The METComm heuristic assigns task to that device that can take minimum execution time to complete the task. The MCTComm heuristic assigns tasks based on the minimum expected completion time on a device. The remaining heuristics are used to assign a task by considering the communication cost. To investigate the performance of proposed heuristics different metrics, such as average makespan, the average waiting time, the average slowdown and the average utilization are used. The results suggested that the expected completion time must be taken into account while mapping the tasks. Moreover, the proposed heuristics are efficient in terms of performance, however, only the matter of problem is complexity.

A novel service mode called opportunistic ad hoc cloudlet service (OCS) was proposed in (Chen *et al.*, 2015). Moreover, a new architecture of cloudlet was presented. Classification of the offloading is categorized into three modes, namely remote cloud service (RCS), connected ad hoc cloudlet service (CCS), and (OCS). In addition, the OCS is further classified into three categories, namely OCS (back & forth), OCS (one way-3G/4G), and OCS (one way-WiFi). The OCS mode is treated as intermediate between RCS and CCS mode. The OCS mode can enable the energy-efficient and intelligent strategy to offload compute-intensive task using ad hoc cloudlet in cost-effective and flexible manner. Despite many advantages of the OCS, selecting reliable and secure nodes to form ad hoc cloudlet to offload the task is a major problem.

Table 2.1: Comparison of task offloading based proposed solutions

Proposed Solutions	Specified Focus	Advantages	Disadvantages
MinHop MetComm MCTComm MinMinComm MaxMinComm SufferageComm (B. Li *et al.*, 2015)	To focus on the decision problem about how to offload computation-intensive applications in MAC.	• High performance • Optimal task Offloading	• High complexity • Longer time in decision-making process
OCS RCS CCS (Chen *et al.*, 2015)	To enable the energy-efficient and intelligent strategy to offload compute-intensive task using ad hoc cloudlet.	• Cost-effective • Flexible	• Selection of reliable and secure nodes is difficult.

2.1.2. Task Scheduling and Allocation

To solve the problem of task allocation in heterogeneous wireless environment, algorithms were proposed in (Lu *et al.*, 2015). The objective of this study was to minimize average task response time for an entire set of tasks by determining whether they need to be distributed or not and on which device they should be executed. Moreover, the algorithm also considered the parameters, such as communication delay, processing delay, and queuing delay while allocating the task. Furthermore, the authors proved the task allocation problem as NP-hard and proposed two approaches named offline centralized and online distributed to solve the problem. The results were very promising in terms of response time in different scenarios. Despite many benefits of the proposed approaches, load imbalance problem will remain a challenging issue that needs to be solved in future.

A new cyber foraging-based system called Scavenger was proposed in (Kristensen *et al.*, 2010). It enables the task distribution and scheduling mechanism among the nodes taking part in communication. To perform the scheduling, scavenger considers multiple factors, such as data locality, network capability, device strength, and task complexity. Moreover, the scheduler helps to determine whether the task execution would be feasible on the local device or in a remote environment. The proposed system shows significant performance improvement in mobile applications execution and also

results in saving energy consumption. However, scheduling a small task in Scavenger leads to time wastage because it requires more time than the actual execution time.

Shi *et al.* (2016) formulated the energy efficient task scheduling problem in local mobile clouds. In this context, an adaptive probabilistic scheduler was proposed that helps to schedule different tasks by satisfying the task's time constraints while keeping the low energy consumption for compute-intensive real-time applications. The proposed scheduler provides many advantages such energy-efficient scheduling, scalability, and flexibility. However, high complexity is one of the disadvantages.

A new task allocation mechanism with the objective of reducing the energy consumption and the computational cost was proposed in (Guo *et al.*, 2016). Moreover, a two-stage Stackelberg game was formulated to determine the number of execution units that slave nodes are willing to offer, while master node sets the price strategies for the different slave nodes according to their shared resources. Although proposed solution helps to solve the task allocation problem in the ad hoc mobile cloud, however, negligence of resource and operational heterogeneity of mobile devices while allocating the tasks is one of the disadvantages.

Table 2.2: Comparison of task scheduling and allocation based proposed solutions

Proposed Solutions	Specified Focus	Advantages	Disadvantages
Scavenger (Kristensen *et al.*, 2010)	To enable the task distribution and scheduling mechanism among the nodes taking part in communication.	• Performance enhancement • Energy saving	• Wastage of time
Offline centralized and online distributed (Lu *et al.*, 2015)	To minimize average task response time for an entire set of tasks by determining whether tasks need to be distributed to a mobile device or not and on which mobile device it should be executed.	• Fast task execution • Energy efficient	• Imbalance load balancing
Adaptive probabilistic scheduler (Shi *et al.*, 2016)	To schedule the tasks while keeping low energy consumption.	• Energy-efficient scheduling • Scalable • Flexible	• High complexity
Task allocation mechanism (Guo *et al.*, 2016)	To enable task allocation for ad hoc mobile clouds.	• Optimal task allocation helps to reduce energy consumption and computational cost	• Negligence of mobile device resource and operational heterogeneity

2.1.3. MAC Formation

C-Protocol was proposed in (Zaghdoudi *et al.*, 2015). It is responsible for the management and deployment of P2P mobile cloud over MANET. To establish the MAC, c-protocol uses four types of messages, such as cloud setup, add provider, add customer, and cloud setup. The proposed protocol manages the mobile nodes in a dynamic manner. In the infrastructure-less environment, mobile nodes can easily divide their compute-intensive tasks to perform the execution by using proposed architecture of MAC platform. The establishment of MAC can provide several advantages, such as ubiquity, availability, affordability, opportunity, and spontaneity. However, challenges, such as how to convince users to contribute through their mobile devices as a provider nodes and lightweight formation require attention.

A collaborative platform named transient cloud was proposed in (Penner *et al.*, 2014). It allows nearby mobile devices to share their resources with each other. Moreover, modified version of Hungarian method to perform the task assignment within the ad hoc cloud is proposed that provides many advantages, such as load balancing, and collocating executions. The proposed platform allows users to create MAC using on-the-fly mobile devices available in the vicinity. The only limitation of the work is that the current technologies only allow partial implementation of transient cloud but still it can show the potential of MAC.

A sporadic cloud-based mobile augmentation (S-CMA) solution was proposed in (Ordonez-Morales *et al.*, 2015). It enables the users to lend the resources from ad hoc cluster of moving mobile devices. In the solution, a virtualization layer is used to tackle the complexity that is derived from the mobility of the cluster. S-CMA enables sharing and allocation of resources in mobile ad hoc cluster. Moreover, it provides a solution to existing approaches that can improve the experience of mobile users towards adapting the mobile ad hoc cluster platform. Furthermore, the proposed S-CMA helps to cope

with many challenges associated with traditional CMA, such as noticeable computation, communication cost of migrating compute-intensive tasks to remote servers, and network latency. Despite many merits of S-CMA, challenges, such as enabling autonomy and coping with mobility problem are yet to be investigated.

An ad hoc cloudlet-based gaming architecture was proposed in (Chi *et al.*, 2014). The architecture is comprised of two modules. The first module enables the mobile users to download the gaming resources from the cloud servers or nearby mobile users. The second module is based on cloudlet-based task allocation that enables the users to execute their tasks on local nearby available mobile devices in a dynamic manner. To formulate the problem for both of modules, several algorithms have been proposed that result in minimizing the energy consumption cost compared to cloud-based gaming architecture. The only problem in the proposed algorithms is ignoring heterogeneous resources of mobile devices forming mobile ad hoc cloudlet while allocating task that causes wastage of resources in terms of energy consumption and execution time.

A distributed platform (i.e., Hyrax) was proposed in (Hamza *et al.*, 2012). It allows mobile devices in the vicinity to execute compute-intensive tasks. It uses fault tolerance mechanism of Hadoop to minimize frequent disconnections with mobile servers. Mobile devices can access remote cloud if the nearby resources are not available. Hyrax server has two client-side MapReduce processes, called *NameNode* and *JobTracker*, to manage computation process among a group of mobile devices. These devices employ two Hadoop processes (i.e., *TaskTracker* and *DataNode*) to receive tasks from the *JobTracker*. These devices connect to the server and other devices via IEEE 802.11g technology. The Hyrax transparently uses distributed resources and provides interoperability across heterogeneous platforms. However, the Hyrax has high overhead because of the complexity of Hadoop algorithm.

A fine-grained cloudlet architecture was proposed in (Verbelen *et al.*, 2012) that helps to manage applications at the component level. The proposed architecture enables the users to dynamically form the cloudlet by finding mobile devices with available resources within a local area network. Moreover, the proposed cloudlet architecture also provides a framework that is responsible for managing and distributing component based applications. These applications usually have strict real-time requirements. Despite many benefits of the proposed architecture, such as fast execution of compute-intensive applications and rapid data analysis, several challenges with respect to deployment, calculation and scheduling are yet to be considered.

Considering the ad hoc nature of MAC, the authors in (Alnuem *et al.*, 2014; Imran *et al.*, 2013) proposed a localized and distributed algorithm for segregation of critical and non-critical nodes. Based on limited topology information (i.e., 1-hop, 2-hop), each node determines whether it is critical or not. A node is determined as critical if its removal (due to failure or movement) partitions the network into disjoint segments, non-critical otherwise. The proposed algorithm can help to avoid engaging critical nodes for compute-intensive task execution.

Table 2.3: Comparison of MAC formation based proposed solution

Proposed Solutions	Specified Focus	Advantages	Disadvantages
C-Protocol (Zaghdoudi *et al.*, 2015)	To manage and deploy P2P mobile cloud over MANET.	• Ubiquity • Availability • Affordability • Spontaneity	• Lack of incentive schemes
Transient Cloud (Penner *et al.*, 2014)	To enable the nearby mobile devices to share their resources as a cloud.	• Enable compute • intensive task • execution in a • distributed manner.	• Partial implementation • Lack of incentive schemes
S-CMA (Ordonez-Morales *et al.*, 2015)	To enable the users to lend the resources from ad hoc cluster of moving mobile devices.	• Noticeable computation • Minimized communication cost • Low network • Latency	• Negligence of mobility factor
Ad hoc cloudlet (Chi *et al.*, 2014)	To propose a gaming architecture that is based on ad hoc cloudlet.	• Alternative solution for infrastructure • less environment	• Costly due to incentives

Table2.3: continued,

Proposed Solutions	Specified Focus	Advantages	Disadvantages
Hyrax (Hamza *et al.*, 2012)	To provide a distributed platform of mobile devices in a local proximity to execute the compute-intensive tasks on available mobile devices.	• Interoperability • Parallel task • Processing	• High overhead • Complexity
A fine-grained cloudlet architecture (Verbelen *et al.*, 2012)	To enable the users to dynamically form the cloudlet by finding mobile devices with available resources within a local area network	• Fast execution • Rapid data analysis	• Inefficient scheduling • Complex calculation
Ad hoc mobile cloud (Maly *et al.*, 2015)	To enable the mobile devices to form ad hoc mobile cloud.	• Secure formation	• Partial implementation • Lack of rigorous evaluation

Ad hoc mobile cloud computing-based solution called m-cloud was proposed in (Maly *et al.*, 2015). Due to the openness of the Android platform's source code, the solution is implemented in it. Proof-of-concept application can dynamically download modules from a server and then it is possible to run them. The proposed m-cloud enables the mobile devices to use the mobile technologies in emergency situations. In addition, a security policy has also been introduced to avoid the downloading and running of malicious code. Despite many advantages of the work, lack of full implementation and rigorous evaluation are some of the limitations that would be investigated in the future as discussed in the study.

2.1.4. Privacy and Security

A trusted algorithm with the objective of securing spontaneous ad hoc mobile cloud network was proposed in (Lacuesta *et al.*, 2014). The algorithm ensures the reliability and security of the nodes responsible for transmission and communication in the MAC. Furthermore, the algorithm also helps to manage the joining and leaving nodes mechanism. The algorithm is based on AES encryption that employs simple key management feature. The algorithm can ensure the secure communication among nodes

forming MAC. The only problem of the algorithm is delay caused by encryption mechanism compared to without security procedure.

The study (Gong *et al.*, 2015) investigated the privacy issues and proposed a framework that ensures the location privacy while allocating the task to mobile devices in MAC. The framework is based on differential privacy and geo cast that enables the devices to share their resources in mobile ad hoc cloudlet by ensuring the privacy of location information. Moreover, analytical model and task allocation strategies have been developed. The proposed framework is not only ensuring privacy without affecting the quality of services (QoSs) but also minimizes the system overhead in MAC. Despite many benefits of the proposed framework, such as location privacy assurance, low system overhead, and high QoS, however, integrity, and confidentiality of user data are remaining concerns.

A trust management system (TMC) for ad hoc mobile clouds was proposed in (Hammam *et al.*, 2013). The goal of the proposed system is to prevent the malicious mobile nodes from participating in ad hoc mobile cloud. TMC has built over PlanetCloud that was introduced in terms of ubiquitous computing. It monitors the nodes once ad hoc cloud is formed and identifies good and bad nodes by looking at their behavior. After observing the behavior, TMC computes the trust value and store it in the cloud repository. The mobile devices validate the trust value from the stored values and then allow to any mobile node for participating in ad hoc mobile cloud. The node with larger trust value will be considered more reliable and secure. The only problem associated with TMC is the overhead of trusted value storage in the cloud.

The authors in (Mandal *et al.*, 2015) proposed a solution that helps in enabling pairwise key establishment and distribution for the devices participating in the MAC. The objective of this study was to enable secure communication among the mobile devices forming the MAC. The results demonstrated that the solution reduced up to

75% in the number of SekGens required to establish keys in the MAC compared to non-optimized naive schemes. However, the proposed scheme is based on centralized approach instead of distributed which results in a lack of parallel execution support.

The study done by Idowu *et al.* (2012) revealed how QoS can be improved for critical infrastructure systems using probabilistic model. The model can help in detecting vulnerabilities, synchronizing mobile sensors using ad hoc and secure Bayesian networks in cloud computing. The proposed model ensures high QoSs in critical infrastructure protection by deploying SaaS and PaaS in cloud computing. Moreover, the proposed model helps in monitoring and predicting wireless nodes behavior to mobile users. The proposed model can provide many advantages, such as reliable detection and recognition of a condition that can enable to mitigate the risk for the critical system protection. The only problem with the proposed model is its implementation and evaluation that seems very complex.

Table 2.4: Comparison of security and privacy based proposed solutions

Proposed Solutions	Specified Focus	Advantages	Disadvantages
TMC (Hammam *et al.*, 2013)	To prevent the malicious mobile nodes from participating in MAC.	• Trust assurance • Reliability • Security	• Overhead of trusted value storage
Location privacy (Gong *et al.*, 2015)	To ensure the location privacy while allocating the task to mobile devices in MAC.	• Location privacy assurance • Low system overhead • High quality of service	• Lack of confidentiality and integrity of data
Trust assurance (Lacuesta *et al.*, 2014)	To ensure the reliability and security of the nodes responsible for transmission and communication in MAC.	• Secure communication	• Delay in communication
A probabilistic model for vulnerability detection (Idowu *et al.*, 2012)	To detect vulnerabilities, and synchronizing mobile sensors using ad hoc and secure Bayesian networks in cloud computing.	• Reliable detection • Recognition of Condition	• Complex implementation and evaluation
A pairwise Key establishment scheme (Mandal *et al.*, 2015)	To enable pairwise key establishment for the devices participating in MAC.	• Minimization of number of SekGens executions • Secure communication	• Lack of parallel execution support

2.1.5. Incentives and Mobility

A vision and initial design considerations of MAC was provided in (Miluzzo *et al.*, 2012). The concept of the cloud provider and cloud customer nodes were also introduced. Moreover, the incentive scheme with mathematical examples was also proposed. The authors envisioned that mobile devices will be able to form an ad hoc cloud due to their high processing and memory capabilities. The goal of the study was to provide a vision to the researchers that in future MAC will be a new computing paradigm that can enable the users to execute their compute-intensive task in the dynamic and infrastructure-less environment.

A virtual cloud framework that enables MAC computing using the mobile devices in the local vicinity was proposed in (Huerta-Canepa *et al.*, 2010). The framework detects the nodes available in a specific area by checking its stability and mobility pattern and form a virtual cloud that can allow the mobile devices to execute compute-intensive tasks. The architecture of the proposed framework is comprised of five components, namely application manager, resource manager, context manager, p2p component and offload manager. The work was a preliminary and open door for future research in terms of task management (when the node executing the task and suddenly leaves the cloud) and selection of secure mobile node for job execution.

A workload distribution scheme was proposed in (Truong-Huu *et al.*, 2014) that considers the randomness of the connection time among the cooperating devices by adopting a multi-stage stochastic programming approach. The scheme enables the mobile devices to form MAC and distribute their workload with each other by considering the mobility factor. Once the ad hoc cloud is formed and the workload is distributed to the neighboring devices, difficulty arises when provider nodes may move out of the range before sending results back to the source node. To cope with this problem, stochastic programming approach is employed that enable optimal decision

making. Furthermore, to make the optimal workload distribution, parameters, such as computing capacity, network bandwidth, and energy constraints have been proposed. The evaluation results show that the stochastic approach not only enables optimal workload distribution but also deals with the randomness of connection time problem that occurs after the workload distribution. However, to motivate users to participate in the execution of compute intensive tasks, incentive mechanisms must be provided.

Tang *et al.* (2016) proposed a double-sided bidding mechanism in mobile cloud where each user who wants to execute his task and the supplier who is willing to share the device resources can submit a bid though demand resource-price function and supply resource-price function, respectively. Despite many advantages of the proposed mechanism, such as attractive and nominal, however, the bidding mechanism causes unnecessary energy consumption.

Table 2.5: Comparison of incentives and mobility based proposed solutions

Proposed Solutions	Specified Focus	Advantages	Disadvantages
Stochastic programming approach (Truong-Huu *et al.*, 2014)	To distribute the workload by considering the randomness of the connection time among the cooperating devices by adopting a multi-stage stochastic programming approach.	• Optimal workload distribution • Randomness of connection time	• Low-level QoS
Virtual cloud computing (Huerta-Canepa *et al.*, 2010)	To enable ad hoc cloud computing using the mobile devices in the local vicinity.	• Incorporation of • mobility pattern while forming MAC	• Significant delay while forming ad hoc cloud due to decisions involvement.
Incentive scheme (Miluzzo *et al.*, 2012)	To motivate the mobile user to opt-in MAC participation.	• Nominal • Truthful	• It is not designed by considering the rationality of • Individual mobile users.
Directory-based architecture (Yousafzai *et al.*, 2016)	To keep track of the retribution and reward valuations	• Provide motivation to the user to participate in MAC	• Third party involvement can raise management security and privacy concerns
Double-sided bidding mechanism (Tang *et al.*, 2016)	To facilitate the user and supplier by providing a double-sided bidding mechanism.	• Attractive • Nominal	• Unnecessary energy consumption

A directory-based framework was proposed in (Yousafzai *et al.*, 2016) to keep track of the retribution and reward valuations (in terms of energy saved and consumed) for devices even after they move from one ad hoc environment to another. The proposed framework can help to motivate the mobile users to share their devices in MAC environment. However, the involvement of the third party that keeps track of retribution and reward valuations can raise some management, privacy, and security concerns.

2.1.6. Resource Management

A multihop mobile ad hoc cloud (MMADC) framework was proposed in (Malhotra *et al.*, 2014). The framework improves the resource utilization and also copes with the scalability and connectivity issues in MAC. The MMADC is comprised of three types of nodes, namely, mobiles nodes (CN), consumer nodes (PN), and matchmakers node (MN). If CN wants to execute some task, it first sends a request to MN that keeps the list of available PN, in this way a cloud is formed. The MN is usually multiple hops away from the CN that can degrade the performance of the task in terms of overall systems performance. The proposed framework coped with this problem by dividing the ad hoc network into static and dynamic clusters. While the former divides the cluster into fixed size, the later divides into dynamic sizes. The proposed framework can help to solve many problems related to scalability, memory space, and processing capabilities. The only disadvantage of the work is the extra time that is required to select cluster head.

PlanetCloud was proposed in (Khalifa *et al.*, 2014b). It provides intrinsic support for highly mobile and heterogeneously-compostable MAC. Moreover, it enables the MAC to adapt the real-time dynamics variations in its underlying infrastructure by isolating the hardware and code management concerns. The PlanetCloud is powered by an application layer that is responsible for encapsulating cloud applications and enable safe and reliable execution in the resource heterogeneous MAC environment. The

evaluation results of the PlanetCloud platform show that it can perform very well in terms of execution time with very less number of VM migrations even in the case when a large number of nodes left the MAC. Despite many advantages of PlanetCloud, such as fast execution time and minimum delay overhead, however, complexity is a major concern.

Table 2.6: Comparison of resource management based proposed solutions

Proposed Solutions	Specified Focus	Advantages	Disadvantages
MMADC (Malhotra *et al.*, 2014)	To improve the resource utilization and also cope with the scalability and connectivity issues in MAC	• Scalability • Memory space • Processing capabilities	• Cluster head selection causes wastage of time
PlanetCloud (Khalifa *et al.*, 2014b)	To provide intrinsic support for highly mobile and heterogeneously-compostable MACs.	• Fast task execution • Minimum delay Overhead	• Complexity

2.2 Taxonomy of MAC

Figure 2.2 shows the taxonomy of MAC where the following parameters are considered for the classification of the research work. a) architectural components, b) applications, c) objectives, d) characteristics, e) execution model, f) scheduling type, g) formation technologies, and h) node types (Yaqoob *et al.*, 2016). Furthermore, in this section comparison of literature based on objectives is also presented in table 2.7 (Yaqoob *et al.*, 2016).

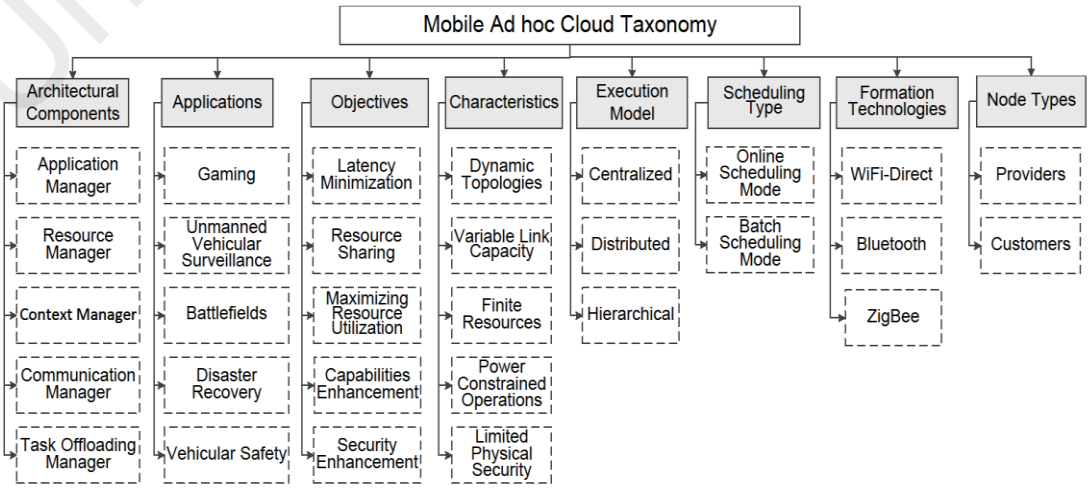

Figure 2.2: MAC taxonomy based on literature

Table 2.7: Literature comparison based on objectives

	MAC Formation	Mobility	Incentives	Quality of Service	Resource Management	Task offloading & Allocation	Energy-Efficient	Cost-effective	Security	Privacy
(B. Li *et al.*, 2015)						✓		✓		
(Kristensen *et al.*, 2010)						✓	✓			
(Zaghdoudi *et al.*, 2015)	✓					✓				
(Lacuesta *et al.*, 2014)									✓	
(Idowu *et al.*, 2012)				✓					✓	
(Chi *et al.*, 2014)	✓			✓						
(Verbelen *et al.*, 2012)				✓				✓		
(Lu *et al.*, 2015)						✓	✓			
(Miluzzo *et al.*, 2012)	✓		✓							
(Malhotra *et al.*, 2014)					✓			✓		
(Huerta-Canepa *et al.*, 2010)		✓								
(Khalifa *et al.*, 2014b)		✓			✓					
(Gong *et al.*, 2015)										✓
(Ordonez-Morales *et al.*, 2015)	✓							✓		
(Hammam *et al.*, 2013)									✓	
(Hamza *et al.*, 2012)						✓				
(Truong-Huu *et al.*, 2014)		✓								
(Chen *et al.*, 2015)						✓		✓		
(Penner *et al.*, 2014)	✓									
(Mandal *et al.*, 2015)									✓	
(Maly *et al.*, 2015)	✓									
(Tang *et al.*, 2016)			✓							
(Yousafzai *et al.*, 2016)			✓							
(Shi *et al.*, 2016)						✓	✓			
(Guo *et al.*, 2016)						✓		✓		

2.2.1. Architectural Components

MAC is comprised of five main components that are responsible for performing various functions to maintain the system. These components are application manager, resource manager, context manager communication manager, and task offloading manager. The application manager is in charge of launching and modifying an application to add the offloading support and proxy creation. The resource manager is responsible for application profiling and monitoring of resources on the mobile devices. For each application, the profile is defined in terms of a number of mobile devices required to form the MAC and amount of resources required for offloading. The context manager collects and synchronizes the contextual information from different widgets and provides it to other processes. The communication manager handles the communication between the consumer and mobile devices. The offloading manager is responsible for dispatching jobs from consumer to provider mobile devices, getting back the results, and creating protected space for the offloaded jobs coming from other devices.

2.2.2. Applications

MAC enables the execution of various applications on resource-constrained mobile devices by sharing their resources (N. Fernando *et al.*, 2016). Few example applications can be gaming, unmanned vehicular surveillance, battlefields, disaster recovery, and vehicular safety. In gaming, players share the resources of the MAC to run the game in the distributed manner. The devices connectivity is considered stable as the players tend to stay in the same place while playing the game. In the unmanned vehicular surveillance, a group of unmanned vehicles forms the MAC to monitor the area and run the information fusion algorithms. Similarly, the battlefields, disaster recovery, and vehicular safety applications can also be run on the group of cloud

provider nodes to perform the compute-intensive tasks on the resource-constrained mobile devices.

2.2.3. Objectives

The objectives attribute indicates the primary objective of the proposed work in MAC. Current MAC solutions aim to attain a number of objectives, such as latency minimization, resource sharing, maximizing resource utilization, capabilities enhancement, and security enhancement (Loke *et al.*, 2015).

2.2.4. Characteristics

The MAC has some special characteristics that make it unique from server-based cloudlet and the remote cloud. These characteristics are mainly inherited from the MANET on which the cloud will be deployed (Niroshinie Fernando *et al.*, 2011). These characteristics are dynamic topologies, variable link capacity, finite resources, power constrained operations, and limited physical security (Sciarrone *et al.*, 2015). The key factor that contributes in the dynamic topologies is the user mobility. The variable link capacity is because of the varying noise and interference level for each device as well as the data rate supported by the device. The mobile devices manufacturing companies keep the finite resources for making them more portable. The size of mobile devices increases if the resources increase. The operations performed in the MAC are power constrained due to battery powered nature of the devices. The frameworks and algorithms designed for MANETs should be lightweight. The physical security in MAC is limited due to more prone nature of wireless networks to physical security threats than that of fixed wired networks.

2.2.5. Execution Models

The execution models for the MAC can be categorized into two main categories, namely centralized and distributed. In centralized MAC, a server node is responsible for managing the execution and distribution of the application in ad hoc cloud of mobile

devices. On the other hand, a group of mobile devices is responsible for the management of the application execution in distributed MAC without centralized control.

2.2.6. Scheduling Types

In MAC, task scheduling can be categorized into online and offline. The online takes task scheduling decisions at the run-time considering the process characteristics and context of the MAC. However, in the batch scheduling, also known as offline scheduling, task scheduler makes task scheduling decisions before actual execution of the application in MAC (W. Zhang *et al.*, 2016). In batch/offline task scheduling, a table is created that contains possible scheduling decisions for use at runtime. The table generation completely depends on the prior knowledge of application execution behavior.

2.2.7. Formation Technologies

To establish MAC, three wireless communication technologies are prominent i.e., WiFi-direct, Blue-tooth, and ZigBee. WiFi-direct is a new addition to the android operating system that enables the mobile devices to connect with other over WiFi and exchanges data. In WiFi-direct, one device acts as a group owner and rest of the devices need to be connected with the owner to perform the specified task; whereas, Blue-tooth is a wireless technology standard comes in mobile devices for short distance communication. Blue-tooth enables the mobile devices to form piconet to communicate with each other. Piconet comprises of master nodes and slave nodes. The master node can talk with seven slave nodes in a piconet. Blue-tooth is better for mobile devices as it consumes less power. However, ZigBee is a high-level communication protocol that can help in forming the Personal area network (PAN) with low-power digital radios.

2.2.8. Node Types

MAC comprises of two types of mobile nodes, namely providers and consumers (Khalifa *et al.*, 2014a). The provider nodes share their resources to facilitate the other mobile devices in terms of running their applications. The consumer nodes leverage the resources of provider nodes to perform their compute-intensive tasks. A device can be consumer node at one time and provider at the other time.

2.3 Principles for Enabling MAC Computing

We have identified several principles from the literature as presented in Section 2.1. These principles provide straightforward guidelines for designing the frameworks that can satisfy the performance metrics for the mobile user's compute-intensive applications execution in MAC. The principles for enabling MAC are categorized into six main categories, namely attractive incentives, optimal task allocation, lightweight formation, agile security, stability, and autonomy. Here, we discuss each of the principles in detail. Figure 2.3 shows the principles for deployment of successful MAC (Yaqoob *et al.*, 2016).

2.3.1. Attractive Incentives

To enable the MAC computing by leveraging the resources of nearby available mobile devices requires some attractive incentive schemes. The designing of incentive schemes should be comprised of three steps, namely analysis, design, and evaluation. The term analysis means investigating about the mobile user's incentive choices. For example, what type of incentives mobile users want to participate in MAC. After conducting analysis, the incentive schemes should be designed according to the need of the mobile users. Finally, the incentive schemes should be evaluated by applying appropriate evaluation methods. Moreover, while proposing any incentive scheme in MAC, the factors, namely budget balance, individual rationality, and truthfulness must

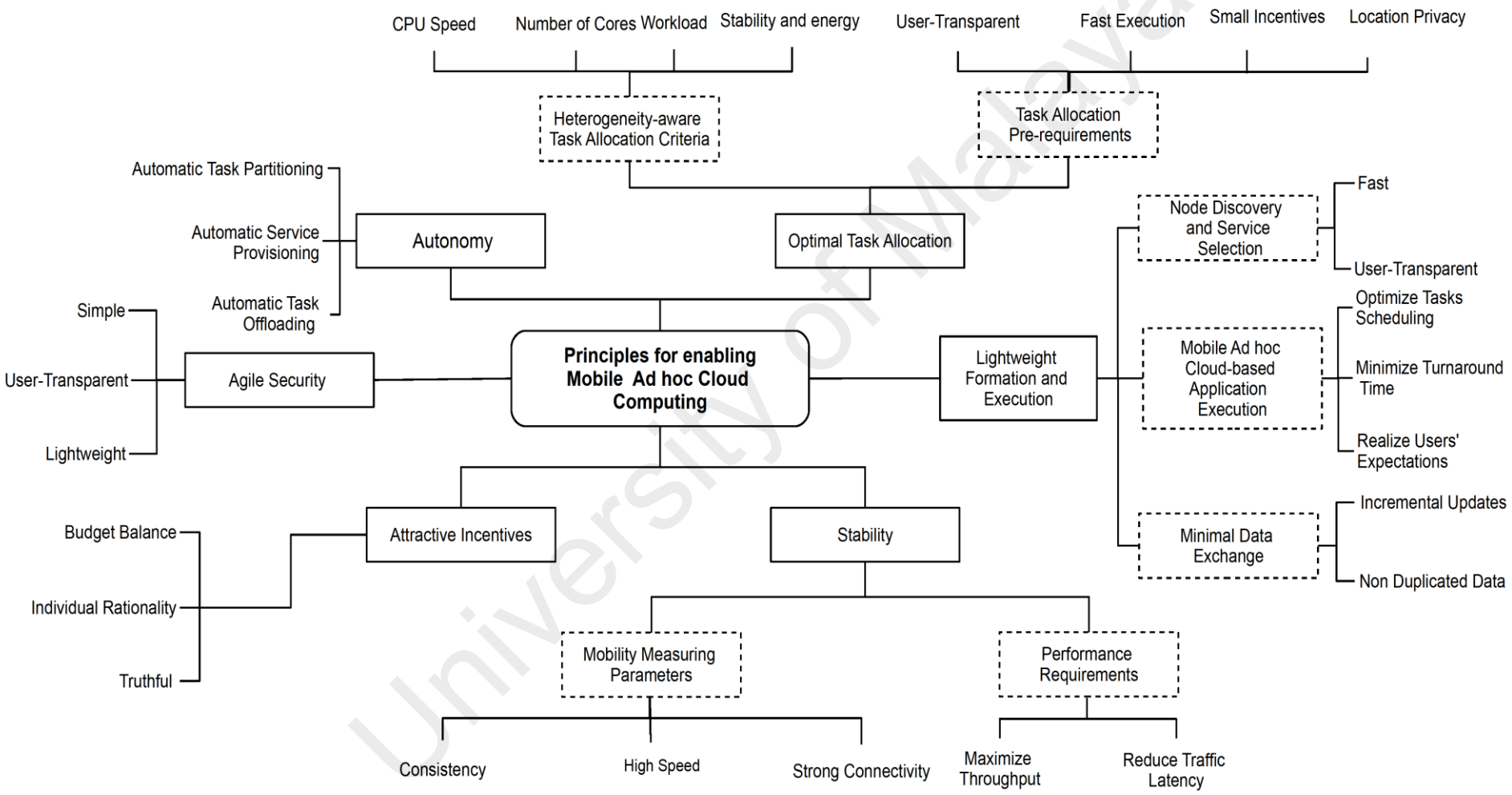

Figure 2.3: Identified key Principles for deployment of successful MAC

be taken into account. Without any attractive incentive scheme, mobile users may not agree to share their device in MAC because it will not give any benefit to them. Therefore, the framework designers should incorporate all possible incentives to enable the successful deployment of MAC.

2.3.2. Optimal Task Allocation

To facilitate the mobile users to execute compute-intensive tasks in an optimal manner requires new task allocation mechanisms (Rashidi *et al.*, 2016). The task allocation framework should have the ability of dynamic decision making regarding partitioning matters, such as individual or distributed processing. After partitioning phase, the framework must be able to allocate the task to other mobile devices by taking into account the heterogeneity of mobile device resources in terms CPU speed, number of cores, background workload, and stability and energy level. The consideration of these parameters while allocating the task can enable proper utilization of the resources that can result in fast task execution and energy saving.

2.3.3. Lightweight Formation

To make the MAC adoptable, the formation mechanism must be lightweight. Before designing any framework with regard to resource discovery, maintenance, and releasing information, the constraints of the mobile devices, such as limited processing capabilities and the battery should be incorporated within the framework (Egbe *et al.*, 2016; McGilvary *et al.*, 2015; Mtibaa *et al.*, 2013). The node discovery mechanism should be transparent and lightweight. Once the MAC is formed scheduling of tasks should be optimal enough to execute the given tasks in minimum turnaround by meeting the expectation of mobile users. The latency in forming MAC can cause wastage of resources that may not be free of cost because of incentive mechanisms. Moreover only minimal data should be exchanged while forming a cloud of mobile devices. In addition, non-duplicated and incremental updates based mechanism can ensure the

minimal data exchange that can result in saving energy and processing of mobile devices opted in MAC participation.

2.3.4. Agile Security

To facilitate the adoption of MAC proper agile security mechanisms are required (Shila *et al.*, 2016). The agility can be achieved by using the lightweight and user transparent techniques. The MAC framework designers should incorporate off-the-shelf authorization and authentication mechanisms for ensuring proper security in a lightweight manner. The authentication and authorization mechanisms should be designed in such a way that it requires minimal interaction and time from the mobile user to enable task execution. Apart from the above-presented principles, QoS of the communication channel is also mandatory to be monitored for enabling smooth collaboration among mobile devices. The incorporation of these presented principles can help the mobile users to enable secure MAC computing.

2.3.5. Stability

In MAC, only the nodes having higher stability value should be selected for task execution. The stability pattern of the nodes needs to be incorporated dynamically while developing any new framework for MAC. The nodes participating in MAC are of dynamic nature in terms of movement that can lead towards incomplete task execution (C. Li *et al.*, 2016). The stability of nodes can be measured by taken into account the mobility measuring parameters, namely consistency, high speed, and high connectivity. Once the node stability is measured then mobile users demand, such as maximum throughput and reduce traffic latency can be fulfilled. After classifying the nodes into lower and higher stability pattern categories then it does not mean that lower stability pattern devices are not useful, but these devices can be used to run a backup of the task by giving fewer incentives to overcome the failure chances.

2.3.6. Autonomy

The MAC platform requires minimal human interaction to access the devices for task execution purpose. The automated service should be based on automatic task partitioning and its offloading. Once the controller device receives some compute-intensive task, it's partitioning, and the offloading decision should be taken in an automatic manner. The MAC-based automated service provisioning can be implemented in three steps: (a) developing a model that can predict the resources and QoS required for the given compute-intensive task, (b) allocating task according to the prediction model, (c) Periodically monitoring of tasks in terms of defined QoS rules (Akinola *et al.*, 2015). Despite difficulties are involved in all the three steps of automated service provisioning, the automated service access process is essential for MAC.

2.4 Open Research Issues

This section discusses the open research issues related to the MAC. The purpose of discussing the open issues is to give research directions to new researchers in the domain.

2.4.1. Heterogeneity-aware Task Allocation

In MAC, where heterogeneous resource constrained devices participate to execute some compute-intensive tasks, inefficient task allocation has become a significant problem. The negligence of the heterogeneous mobile device resources, such as CPU speed, background workload, and number of cores while allocating task can cause inefficient resource utilization that results in longer execution time and high energy consumption. The incorporation of these parameters has become very challenging due to complexity and overhead. In the context of task allocation, several research efforts have been carried out in (Fang *et al.*, 2014; B. Li *et al.*, 2015; Lu *et al.*,

2015; Zhou *et al.*, 2015). These proposed research works are in their infancy and require further optimization and extension.

2.4.2. Incentives

To convince the mobile users to opt-in to MAC participation requires nominal incentives mechanisms. Without giving some advantage, it would be very difficult to convince a mobile user to share their available device resources with others. Therefore, to enable the MAC computing, some proper incentives mechanisms are required to be proposed that can motivate the people to share their mobile device resources. The finding of proper incentive mechanisms that can motivate the mobile users encountered to agree on the load offloading has become very difficult due to individual rationality and different demands of the mobile user. Although several researchers have proposed some incentives mechanisms, these solutions are in their infancy (Al Noor *et al.*, 2014; Miluzzo *et al.*, 2012; Tang *et al.*, 2016; Yousafzai *et al.*, 2016). To design appropriate and practical incentive mechanisms requires some future research.

2.4.3. Mobility

Once MAC is formed and sub-tasks of a task are distributed to the selected mobile devices, the mobility can affect the overall task execution time. As can be seen in a scenario where the mobile device leaves the MAC after taking some subtask to execute, which can cause all the sub-tasks to be rescheduled that results in wastage of resources of mobile devices in terms of energy and processing. Moreover, rescheduling of the tasks can be costly in terms of incentives. To cope with the mobility problem several research efforts have been carried out in MANETs that can be applied in the MAC after applying some modifications (Basarkod *et al.*, 2013; Gavalas *et al.*, 2010; Kaur, 2014; Khalifa *et al.*, 2014b; Ordonez-Morales *et al.*, 2015; Rahimi *et al.*, 2013; D. Zhang *et al.*, 2014). In future high attention needs to be paid to address the mobility-related challenges in MAC.

2.4.4. Minimal Data Exchange

Once the compute-intensive task is divided into sub-tasks and distributed to mobile devices, then devices usually share their task execution state in terms of processing after a specific interval of time that can affect the battery power consumption (Benkhelifa *et al.*, 2016). The mobile devices usually have limited resources, and no one wants to waste their device resources for unimportant purposes. Therefore, the amount of data exchange is required to be minimized to perform task execution in the MAC. In this context, several research efforts have been carried out in MCC, where to cope with the problem of minimal data exchange researchers have classified data into three categories: (a) configuration data, such as states information, (b) input data, and (c) OS image and application migration data (Ahmed *et al.*, 2015). The amount of the configuration data transferred can be reduced by only moving the essential configuration data. Although these research efforts cannot be applied directly in MAC, provide basic guidelines to the researchers for designing minimal data exchange based frameworks.

2.4.5. Security and Privacy

Due to a random selection of mobile nodes that usually participate in MAC, security and privacy have become a major concern. Most of the applications of MAC are very sensitive in nature . Therefore, security risks are needed to be measured at priority. The joining of any malicious node in MAC can increase the overall task execution time that can lead towards performance degradation. Moreover, the privacy of the location is also a serious concern for the users who share their devices. To cope with the security and privacy problems off-the-shelf authentication and authorization mechanisms are required that can prevent the malicious node to participate in the MAC by protecting the location privacy. To cope with security and location privacy in MAC, several research efforts have been carried out in (Gong *et al.*, 2015; Hammam *et al.*,

2013; Lacuesta *et al.*, 2014; Thapa *et al.*, 2016) but these efforts are in its infancy. The further extension is required within these proposed mechanisms that can ensure more reliable and secure communication.

2.5 Conclusion

Momentous advances in MCC have paved the way for a new computing paradigm called MAC. Although there are studies of MCC and ad hoc computing, the convergence of these two areas grants further academic efforts for the flourishing of MAC. In this chapter, we reviewed the state-of-the-art research carried out in the MAC domain. We analyzed several problems inhibiting the adoption of MAC and reviewed the corresponding solutions by devising a taxonomy. We compared the proposed solutions by highlighting their advantages and disadvantages. We then devised another taxonomy based on reviewed literature of MAC. In addition, we compared the literature based on objectives. We identified and discussed the key principles that can guide the framework designers to incorporate specific features for enabling successful MAC. We also presented open issues and selected the important one of inefficient task allocation for remediation. Finally, it is concluded that MAC is in its early stage of development and must pay close attention to the presented issues –especially inefficient task allocation –to facilitate the adoption of MAC, which will be a core component of the future computing landscape.

CHAPTER 3: PROBLEM ANALYSIS

This chapter establishes the problem of inefficient task allocation causes by ignorance of heterogeneity-measuring parameters, while making task allocation decisions in MAC. These heterogeneity-measuring parameters are CPU speed, number of cores, and workload. We perform an in-depth investigation of the problem by conducting an experimental study to show that ignorance of the heterogeneity-measuring parameters and random-based task allocation can considerably prolong the tasks' execution time and consumes large amounts of energy. The reason to analyze their impact on tasks' execution time and energy consumption is to show the gravity of the problem.

The rest of the chapter is organized as follows. Section 3.1 discusses the applications and mobile devices that are used to analyze the problem. Section 3.2 describes the performance-measuring parameters. Section 3.3 discusses the system variables used to conduct the experiment. Section 3.4 presents results and its discussions. Section 3.5 presents the analysis of random-based task allocation in terms of execution time. Discussions based on the empirical data analysis are summarized in Section 3.6. We reiterate the findings of the analysis conducted in Section 3.7.

3.1 Empirical Study: Experimental Setup

This section presents the empirical study conducted to establish the problem. We discuss the experimental setup including mobile devices and the compute-intensive tasks that are developed to perform the analysis.

3.1.1. Mobile Device

Samsung S II i9100g smartphone is used to conduct the experiment; specification details are provided in table 3.1. The effect of heterogeneity on tasks' execution time and energy consumption is investigated by changing sizes of tasks, CPU speed, workload, and number of cores of the mobile devices. To customize the CPU speed (e.g., 600MHZ, 800MHZ, 1008MHz, and 1200MHz) and number of cores (e.g., 1

and 2) two applications, Master CPU and Kernel Tuner are used, respectively (AnTuTu, 2011; Čokulov, 2014). In addition, Power Tutor application is used to measure the energy consumption (L. Zhang *et al.*, 2010).

Table 3.1: Specification of mobile device

Mobile components	Specification
CPU	Dual core 1.2GHz
RAM	1 GB
OS	Android Jellybeans 4.1
Processor Architecture	ARMv7 rev3(v71)

3.1.2. Multi-Threaded Matrix Multiplication

A multi-threaded matrix multiplication application is designed to study the impact of heterogeneity of mobile device resources and workload on execution time and energy consumption when allocating tasks in MAC. This application represents the class of compute-intensive applications. In the past, matrix multiplication has been used in image processing and MCC to perform analysis of specified problems (ShirazAhmed *et al.*, 2014; Shiraz & Gani, 2014). The application takes a set of the matrix as an input and gives a result after the multiplication. The application divides the matrix multiplication task and distributes it among the number of available mobile devices. The matrix multiplication is performed on the local device to compute the results. The applications and corresponding task sizes selected for the experiment are presented in table 3.2. In addition, we develop an infinite loop application to analyze the workload impact on task execution time and energy consumption.

3.2 Performance Measuring Parameters

Execution time and energy consumption are selected as performance-measuring parameters to evaluate the impact of heterogeneous resource availability and different background workloads in the MAC environment.

3.2.1. Execution Time

The execution time of the given task is defined as the number of seconds needed by the system to complete a task, including the time spent executing run-time or system services on its behalf. The task execution time depends on the task size, processor speed, and number of background tasks being executed on the mobile device. The execution of the compute-intensive task on a slow device can prolong the execution time. Therefore, task allocation based on the specification of the mobile devices can significantly improve task performance.

3.2.2. Energy Consumption

Energy consumption is the number of millijoules (mJ) used to execute a task. Similar to execution time, energy consumption is more for large tasks and less for smaller tasks. We have evaluated each performance-measuring parameter by running multi-threaded matrix multiplication and infinite loop applications in different scenarios.

3.3 System Variables

This section discusses different system variables used to investigate the impact of heterogeneity on tasks' performance in terms of execution time and energy consumption. These variables are named task size, workload, CPU speed, and number of cores. These variables are investigated by conducting experiments on Samsung mobile device. This performance evaluation study enables us to know the impact of diverse resources of mobile devices and workload on execution time and energy consumption.

3.3.1. Task Size

The execution time of different tasks depends on the processing capabilities of the mobile device being used. The size of the task is directly proportional to the execution time and energy consumption. As the task size grows, execution time

becomes longer. This parameter is chosen to show that each task requires a different processing time and consumes different level of energy. Therefore, the task should be assigned by considering the mobile device specification and workload in MAC. Consideration of the specification and workload while designing new algorithms can assist in ensuring the fast execution of tasks by consuming less amounts of energy.

3.3.2. Workload

An infinite loop application is developed and run to investigate this parameter. Workload is analyzed by running the infinite loop applications on a mobile device. This parameter is selected to measure its impact on execution time and energy consumption.

3.3.3. Processor Speed

Different CPU speeds as a parameter help evaluate the tasks' execution performance. In MAC, where each mobile device has a different processor, execution time of a specific task will vary. The measurement of the impact of different CPU speeds on specified task execution can help to analyze whether or not this parameter should be considered when allocating tasks in MAC.

3.3.4. Number of Cores

This parameter is selected to measure the impact of number of CPU cores on the performance of the specific tasks in terms of execution time and energy consumption. Consideration of this parameter can help making efficient task allocation in MAC, where different mobile devices have different numbers of cores for task execution.

3.4 Results and Discussions

In this section, we discuss the experimental results by running specified tasks. The impact of differences in task sizes, workload, CPU speed, and customizing the number of cores on execution time is investigated. The details of tasks used to evaluate against individual parameter are listed in table 3.2.

Table 3.2: Tasks for evaluations of various parameters

Parameters	Running Tasks	Mobile Settings using "CPU Master" Application
• Task Size	• 100 times 100×100 size matrices multiplication • 200 times 200×200 size matrices multiplication • 300 times 300×300 size matrices multiplication • 400 times 400×400 size matrices multiplication • 500 times 500×500 size matrices multiplication	• Scaling interactive 300MHz to 1200 MHz
• Workload (Running two, four, six, eight, and ten applications)	• 300 times 300×300 size matrices multiplication	• Scaling interactive 300MHz to 1200MHz
• Processor speed (300MHz, 600MHz, 800MHz, 1008MHz, and 1200MHz)	• 300 times 300×300 size matrices multiplication	• Scaling hotplug 300MHz to 1200 MHz
• Number of Cores (1 and 2)	• 300 times 300×300 size matrices multiplication	• Scaling interactive 300MHz to 1200MHz

3.4.1. Workload Impact on Execution Time

Figure 3.1 shows that execution time of the specified task is 235s with no applications running in the background, 535s with two applications running in the background, 710s with four applications running in the background, 896s with six applications running in the background, 1066s with eight applications running in the background, and 1237s with 10 applications running in the background. The size of background application is the same, but we run it multiple times to investigate its impact on the execution of specified task (table 3.2). When there are up to two applications running in the background, then task execution time is shorter. The task execution time increases when there are between four and six applications running in the background. However, the largest increase in execution time is measured when there are between eight and 10 applications running in the background. The investigation concludes that failure to consider background traffic load parameters when making the task allocation decision in MAC where devices have heterogeneous background load can significantly prolong the execution of delay-sensitive applications.

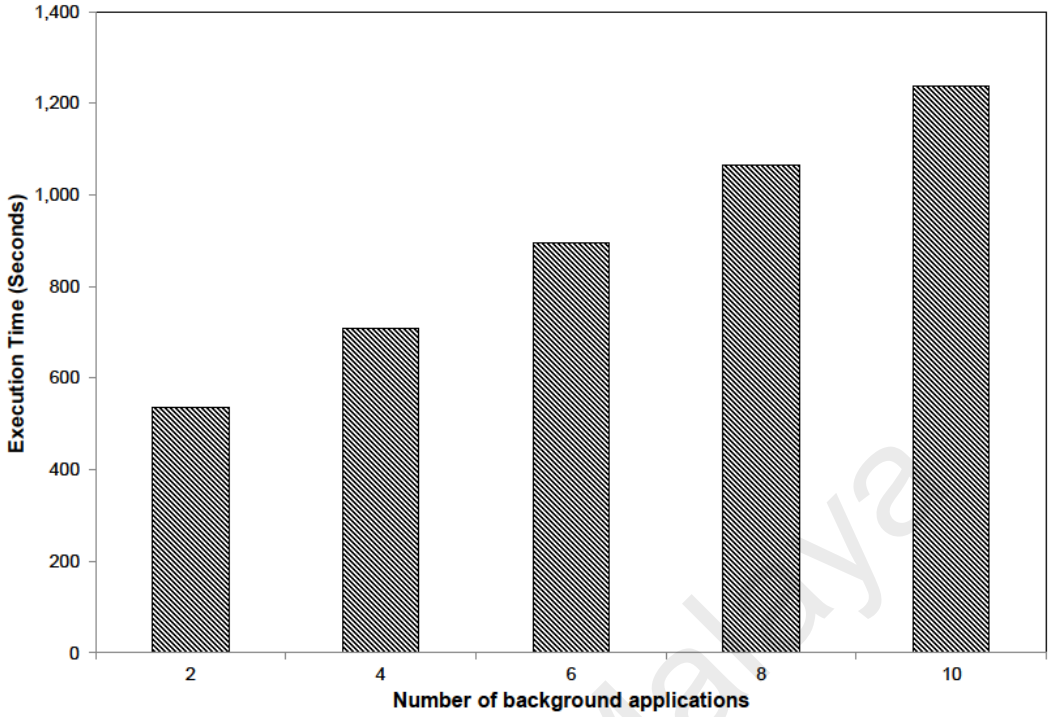

Figure 3.1: Impact of applications running in the background on execution time

3.4.2. Workload Impact on Energy Consumption

To measure the energy consumption of specified task, we use the Power Tutor application. The analysis reveals that changes in the background traffic can affect energy consumption. The energy consumption is 872mJ for the task execution with no background application running, 1300mJ with two, 1400mJ with six, 1600mJ with eight, and 1700mJ with 10. Figure 3.2 shows the effect of an increase in background traffic on energy consumption. This analysis concludes that when allocating tasks in MAC, background traffic or workload must be considered because when the task is assigned to a mobile device with less of a background traffic load, it consumes less energy.

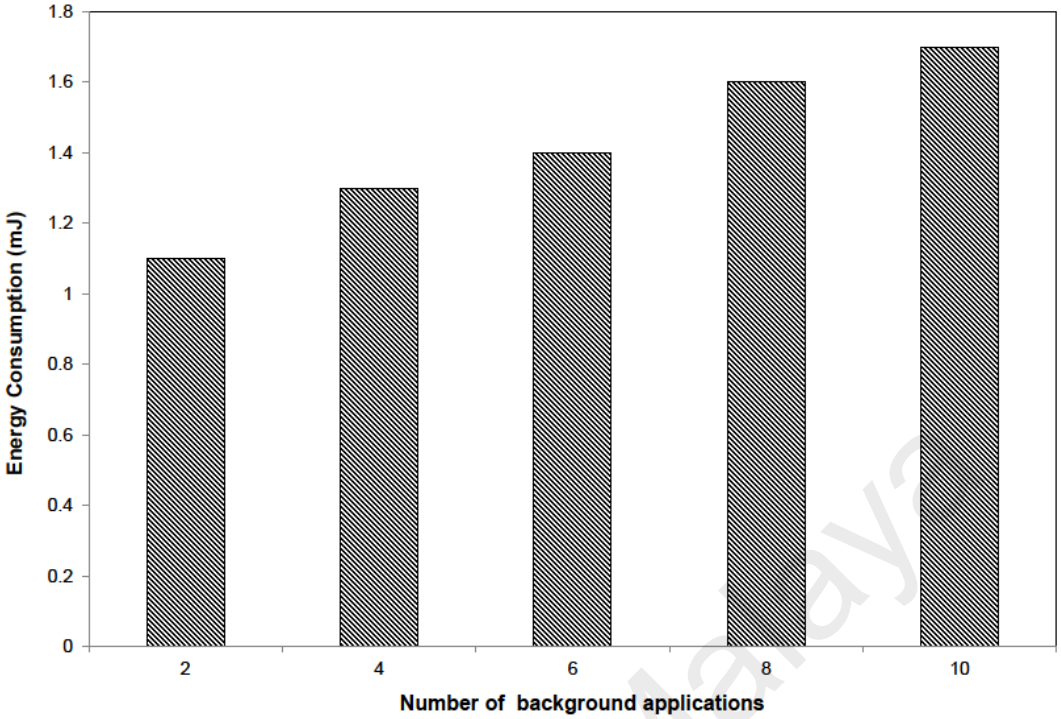

Figure 3.2: Impact of applications running in the background on energy consumption

3.4.3. Varying Number of Cores' Impact on Execution Time

To run the multi-threaded application on a single core and multi-cores CPUs, the Kernel Tuner application is used to customize the number of CPU cores. Processors with more cores can execute a task faster than a single-core processor. Figure 3.3 shows the execution time of running task on a single- and dual-core processor. Our analysis indicates that when the task runs on a single-core CPU its execution time is 364s and on a dual-core CPU its execution time is 231s. The analysis recommends that this parameter must be considered when allocating tasks in MAC because if a small task is assigned to a multi-core processor mobile device and a large task to a single core processor mobile device, performance is degraded. Moreover, task assignment can be performed more efficiently by assigning large tasks to a multi-core processor mobile device and small tasks to a single-core mobile device (assuming that both the single- and multi-core processor mobile devices have no other workload).

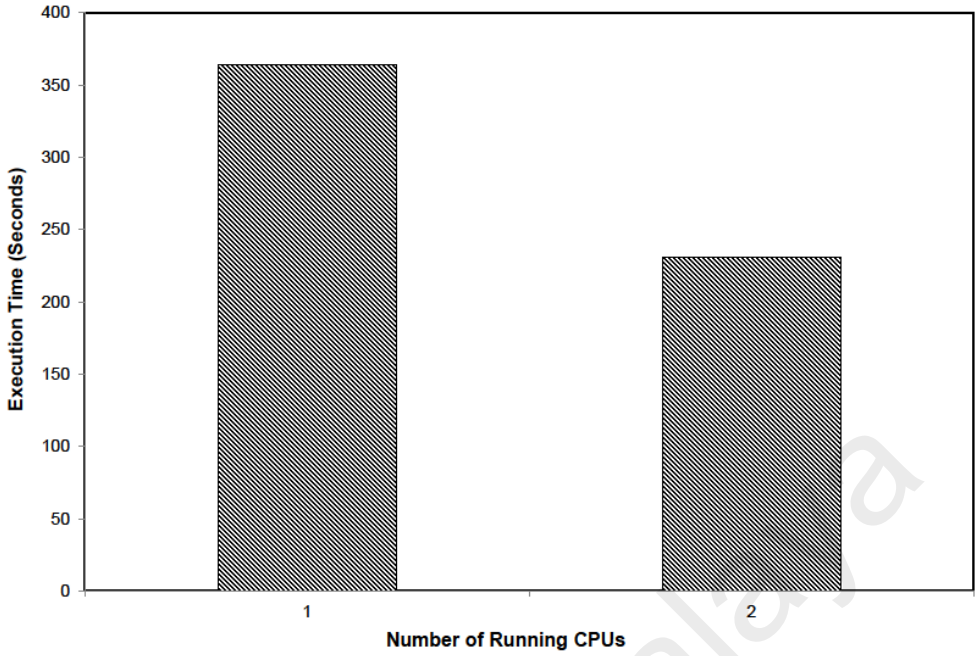

Figure 3.3: Impact of number of CPU cores on execution time

3.4.4. Varying Number of Cores' Impact on Energy Consumption

To measure the energy consumption of the mobile device, we used the Power Tutor application. Energy consumption increases when a single processor is used to execute a task. The energy consumption of task execution is 1006mJ by using 1CPU core and 514mJ by using both CPU cores. Figure 3.4 shows the effect of using a single CPU core and dual CPU cores on energy consumption. This analysis reveals that selection of mobile devices with a higher number of CPU cores when allocating tasks can conserve the MAC's energy resources.

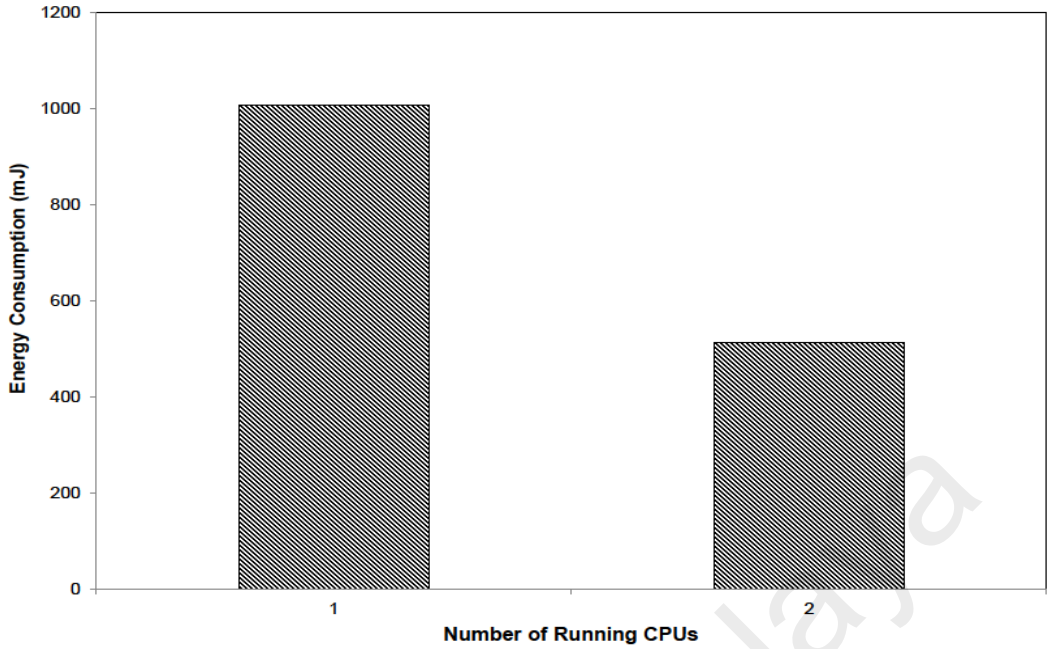

Figure 3.4: Impact of number of CPU cores on energy consumption

3.4.5. Varying Processor Speeds' Impact on Execution Time

Figure 3.5 shows the execution time of a specified task on heterogeneous CPU frequencies. The execution time is 1541s with 300 MHz CPU frequency, 349s with 600MHz CPU frequency, 260s with 800MHz CPU frequency, and 205s with 1008MHz CPU frequency, 185s with 1200MHz CPU frequency. The execution time of task increases slightly at the lower level of CPU frequencies and decreases at a higher level. To tune the CPU speed, we used the Master CPU android application. This analysis reveals that consideration of CPU speed as a parameter when allocating tasks in MAC where devices have different processing capabilities can significantly shorten the amount of time needed to carry out a task.

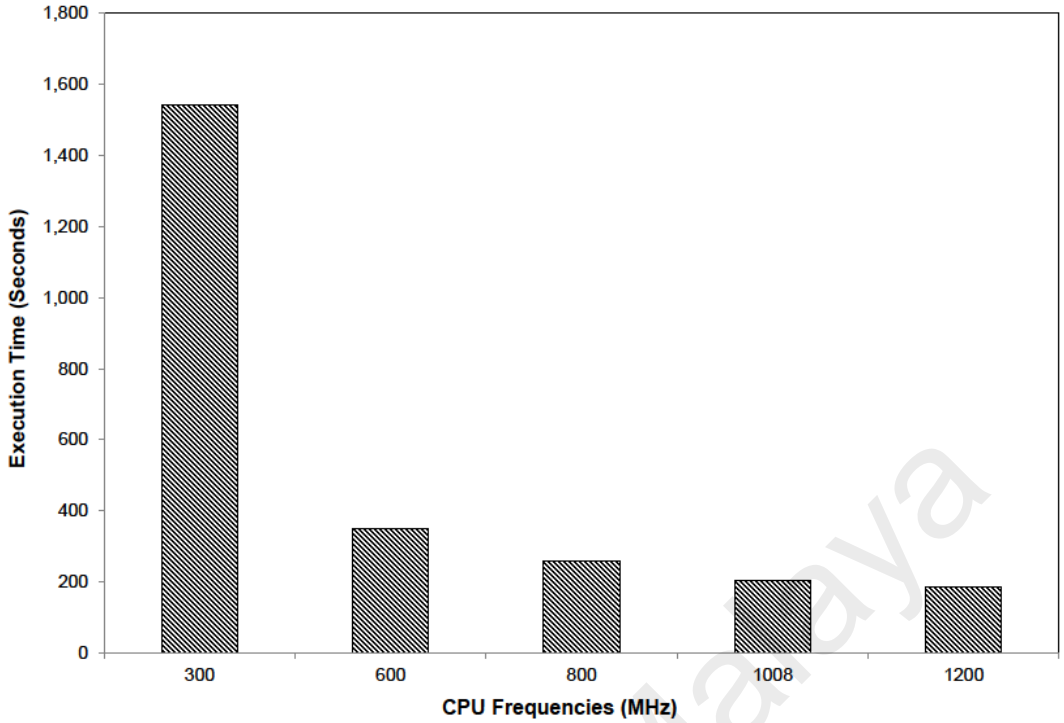

Figure 3.5: Impact of various processor speeds on execution time

3.4.6. Varying Processor Speeds' Impact on Energy Consumption

Figure 3.6 shows energy consumption of task is 503mJ with 300MHz CPU frequency, 641mJ with 600 MHz CPU frequency, 770mJ with 800 MHz CPU frequency, 880mJ with 1008 CPU frequency, and 993mJ with 1200 MHz CPU frequency. Energy consumption slightly increases with the mobile device's CPU frequencies. The reason for this low rate of energy consumption is that at lower CPU frequency task is executed very slowly and its does not heat up the whole board of mobile devices. This analysis concludes that consideration of this parameter when allocating tasks in MAC can be very useful. For example, in MAC, if certain mobile devices with high CPU speed have lower battery power then we can assign the task to a mobile device with a slower CPU speed (assuming that fast execution is not required).

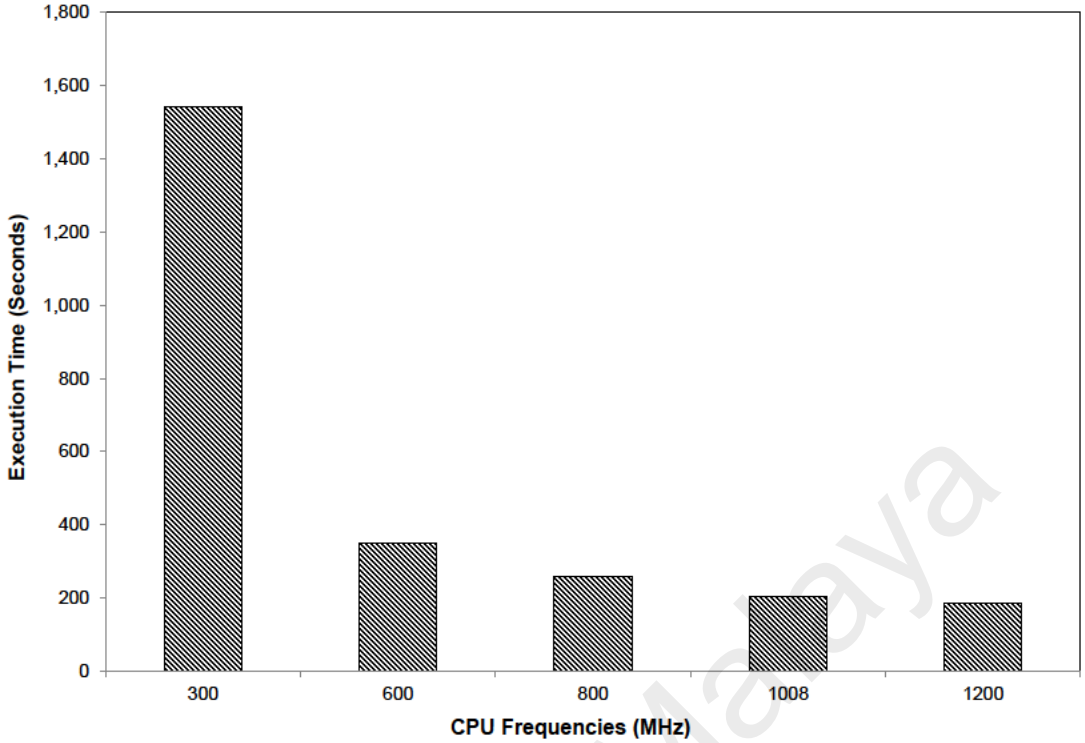

Figure 3.6: Impact of various processor speeds on energy consumption

3.4.7. Varying Task Sizes' Impact on Execution Time

Figure 3.7 shows the execution time of several tasks: 2s for task 1, 31s for task 2, 235s for task 3, 856s for task 4, and 1813s for task 5. Table 3.2 gives details about these tasks. The study of this parameter reveals that a slight increase in task size can prolong its execution time. Large tasks have a longer execution time. The reason for this analysis is to show that tasks of different sizes require different execution times. Keeping this parameter in mind when allocating tasks in MAC can minimize task execution time. Moreover, in MAC, different users have tasks of different sizes, so execution time can be minimized by assigning tasks according to the specification of mobile devices participating in MAC.

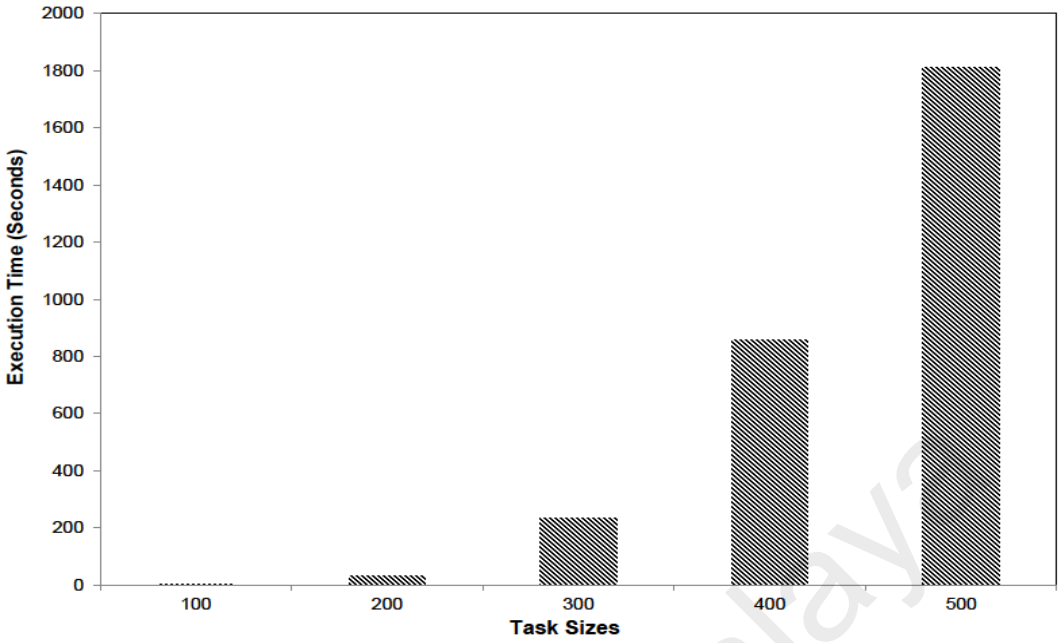

Figure 3.7: Impact of various task sizes on execution time

3.4.8. Varying Task Sizes' Impact on Energy Consumption

Figure 3.8 shows that energy consumption is 513mJ for task1, 725mJ for task 2, 842mJ for task3, 1080mJfor task4, and 1300mJ for task 5. Energy consumption increases slightly with task size. This analysis reveals that consideration of this parameter when allocating tasks in MAC where mobile devices have heterogeneous resources particularly in terms of energy consumptions can be useful. With this analysis, the tasks that require more energy can be assigned to mobile devices which have more battery power by proposing new task allocation solutions.

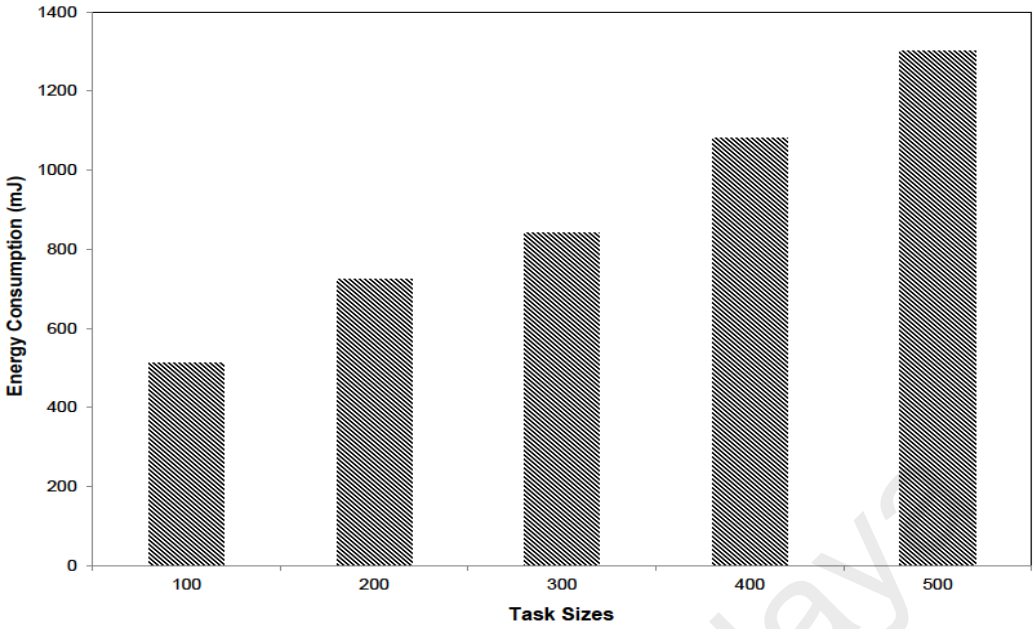

Figure 3.8: Impact of various task sizes on energy consumption

3.5 Analysis of Random-based Task Allocation Mechanism

Figure 3.9 shows the execution times of the various tasks which are determined by running them on different compute nodes. In figure 3.9, the X and Y axes represent the task ID and execution time, respectively. In addition, circles of different colors represent the available nodes for task execution. The large circles give an indication of the node selection for the specific task. The execution time of the task ID 1 is measured as 1500s, 1785s, 2250s, and 3000s while performing its execution on available compute nodes 1 to 4, respectively. The execution time of the task ID 2 is measured 620s, 738s, 930s, and 1239s while running it on nodes 1 to 4, respectively. The execution time of the task ID 3 is measured as 196s, 234s, 294s, and 392s while running it on available nodes 1 to 4, respectively. The execution time of task ID 4 is measured as 39s, 234s, 58s, and 78s while running it on available nodes 1 to 4. The execution time of task ID 5 is measured as 2s, 3s, 4s, and 5s while running it on available nodes 1 to 4. Our analysis demonstrates that the random-based task allocation mechanism can cause inappropriate selection of the compute nodes, significantly prolonging task execution time. As can be seen in the figure, task ID 1 requires minimum execution time at node 1; however, the random task allocation based mechanism selects node 3. Similarly,

tasks having IDs 2 to 5 can also be executed quickly if they were run on node 1 instead of nodes 2 and 4.

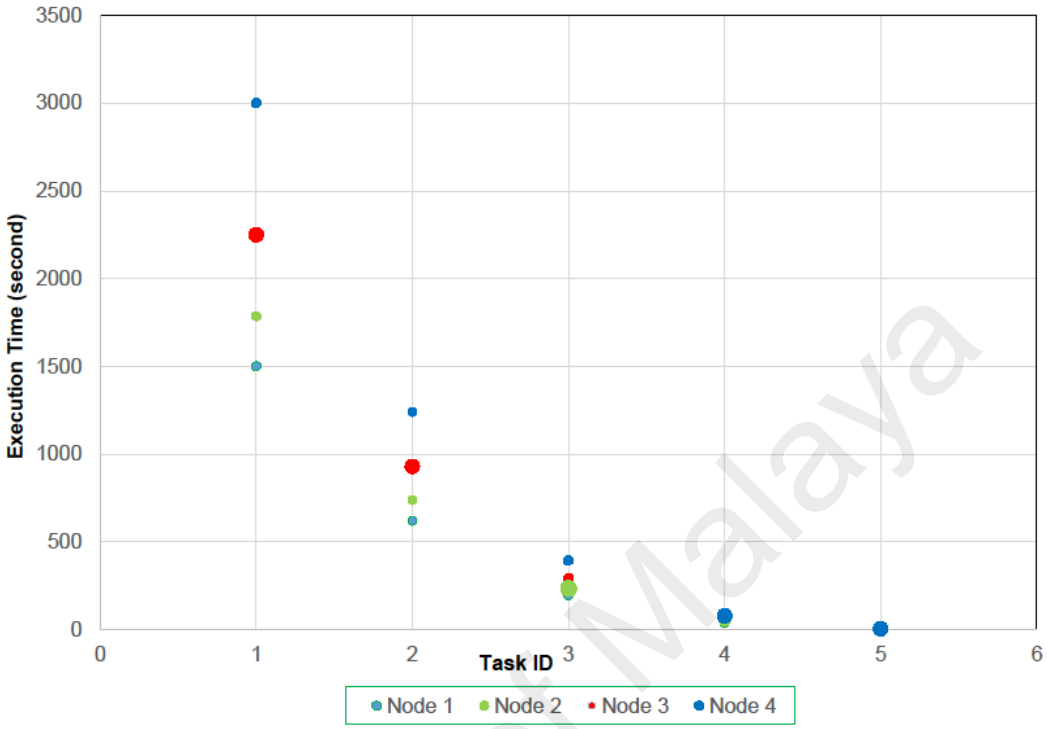

Figure 3.9: Impact of random-based task allocation on execution time

3.6 Discussions

The empirical results highlight the following:

- Performing task allocation without considering the heterogeneous resources of mobile devices forming the MAC can lead to inefficient task allocation.
- The inefficient task allocation results in longer execution time which remarkably degrades the performance of the task and impedes the realization of MAC.
- The existing random-based task allocation solution has merits; however, the significantly higher execution time and energy consumption for the compute-intensive tasks are challenges that need to be addressed. In addition, random-based task allocation usually wastes resources.

This section clearly shows the serious drawbacks of ignoring resource and operational heterogeneity-measuring parameters, such as CPU speed, number of cores, and workload on task execution time.

3.7 Conclusion

In this chapter, we conducted experiments to analyze the impact of heterogeneity and random-based task allocation on the execution time of a specified task in MAC. We evaluated the impact of heterogeneity-based parameters on the performance of task execution in terms of time and energy consumption by varying task size, workload, CPU speed, and number of CPU cores. We also analyzed the impact of a random-based task allocation mechanism on task execution.

Based on the analysis results, it is concluded that inattention to heterogeneity-based parameters and random-based task allocation mechanism can remarkably degrade the performance of task execution in MAC. In random-based task allocation, the tasks are assigned without looking into the specifications of the compute nodes that most of the times result in longer execution time and high energy consumption. Moreover, through random-based task allocation, sometimes a larger task is assigned to a slower device that uses resources inefficiently. Inefficient resource use prolongs the execution time. In the next chapter, we propose heterogeneity-aware task allocation algorithms. The adoption of our proposed solutions can shorten execution time and reduce energy consumption, thereby increasing the usability of MAC.

CHAPTER 4: HETEROGENEITY-AWARE TASK ALLOCATION ALGORITHMS

This chapter aims to propose five heterogeneity-aware task allocation algorithms for minimizing tasks' execution time and energy consumption in MAC. The proposed five algorithms incorporate heterogeneity-measuring parameters, such as CPU speed, number of cores, and workload while performing task allocation. These algorithms are presented in form of pseudo-codes in the chapter. The distinctive features of the proposed algorithms are also discussed. Furthermore, a mathematical model of the solutions in terms of execution time is presented.

The chapter is organized into seven sections. Section 4.1 presents the proposed algorithms for the incorporation of heterogeneity-measuring parameters when allocating task in MAC. Section 4.2 elaborates the MAC framework with respect to the proposed solutions. Section 4.3 depicts the proposed solutions in a sequence diagram. Section 4.4 explains the mathematical equations used for node selection and calculating the energy consumption. The mathematical model for execution time is presented in Section 4.5. The mathematical model of the execution time will be used in chapter 6 to validate the simulation results of the research. Section 4.6 highlights the distinctive features of the proposed algorithms. Section 4.7 summarizes and concludes the chapter.

4.1 Heterogeneity-aware Task Allocation

MAC is comprised of two types of nodes: controller and compute. The controller node manages all the compute-intensive tasks and is responsible for their execution; the compute node offers its own resources for computation through some incentive mechanisms. The random-based task allocation mechanism degrades task execution by extending the execution time and energy consumption. Moreover, lack of incorporation of the heterogeneity-measuring parameters, such as CPU speed, number of cores, and workload in the solution contributes to prolonging the task execution time.

In this context, we propose five algorithms that consider the heterogeneity-measuring parameters while allocating compute-intensive tasks. Figure 4.1 illustrates the task allocation handler module. The proposed task allocation algorithms are implemented in that module. The details of the proposed algorithms are provided in the following subsections.

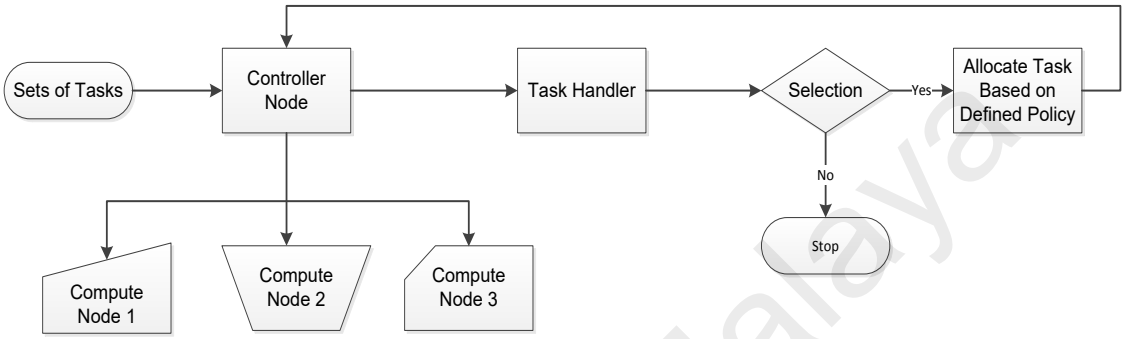

Figure 4.1: Task handler module

4.1.1. Proposed Algorithms

Algorithm 1 presents the pseudo-code of the task allocation based on mobile device CPU speed in MAC. In the algorithm, N, T, and S_N are used as input parameters. (See table 4.1 for the description of the symbols.)

Table 4.1: Description of the symbols used in the algorithms

Symbol	Description
N	Number of mobile devices
T	Number of given tasks or sub-tasks
S_N	CPU speed of N mobile device that is computed by multiplying it with number of cores
C_N	Number of cores of N mobile device
$\boldsymbol{C_N}$	Total capacity of N mobile device
R_T	Real execution time of the task
$\boldsymbol{R_N}$	Residual capacity of N mobile device in terms of workload
CPI	Cycle per instruction of specified CPU architecture of mobile device
E_{tL}^{n}	Expected workload on nth node (because of the task execution)
^	All symbols with ^ represent the maximum or minimum value returned by the function (arg)

To enable task allocation based on CPU speed, several steps are performed. First, tasks of different lengths are sorted in descending order (line 2 in Algorithm 1). Subsequently, as the controller node tracks the compute nodes in terms of specification

and task size, allocating the tasks at that time based on mobile device CPU speed results in fast execution and less energy consumption. One device that has a faster processor speed is selected, and all the tasks are assigned to that device (lines 4-9 in algorithm 1). Thus, the solution helps the controller node to assign the task in a way that minimizes task execution time. However, further reduction in execution time is possible if the task allocation decision is based on several parameters instead of just one.

Algorithm 1: CPU Speed-based Task allocation Algorithm

Input: N, T, S_N
1 $X \leftarrow T$
2 Sort(Desend, X)
3 **for** i=1:|X| **do**
4 $(\widehat{s_n}) = arg\ max_{\forall\, n \in N \,\forall\, s_n \in S_N} f(s_n)$
5 NodeID $\leftarrow getID(N, \widehat{s_n})$
6 map<NodeID> $\leftarrow x_{\{1\}} where\ x_{\{1\}} \in x$
7 X $\leftarrow X/x_1$
8 **end**
9 **OUTPUT**:map<N,X>

To show the procedure of task allocation based on cores, algorithm 2 presents the pseudo-code. In algorithm 2, N, T, and C_N are used as input parameters. (See table 4.1 for the description of the symbols.) Once MAC is formed, compute nodes share their specifications in terms of number of cores with the controller node. When the tasks need to be assigned to compute node, first they need to be sorted in descending order (line 2 of algorithm 2). The controller then selects the compute nodes to perform the task execution based on the higher number of cores that are associated with the available mobile devices (lines 4-7 of algorithm 2). Basing task allocation on the number of cores can minimize the execution time as multi-threaded tasks are executed on the selected devices. Although using a core-based solution minimizes execution time, further reductions are also possible if other parameters, such as workload and CPU speed are incorporated. In addition, task allocation based on multiple parameters can lead to an optimal reduction in execution time compared to core-based solution.

Algorithm 2: Core-based Task Allocation Algorithm

Input: N, T, C_N

1 X $\leftarrow T$
2 Sort(Desend, X)
3 **for** i=1:|X| **do**
4 $(\widehat{c_n}) = arg\ max_{\forall\ n\ \in\ N\ \forall\ c_n\ \in\ C_N}\ f(c_n)$
5 NodeID $\leftarrow getID(N, \widehat{c_n})$
6 map<NodeID> $\leftarrow x_{\{1\}} where\ x_{\{1\}} \in x$
7 X $\leftarrow X/x_1$
8 **end**
9 **OUTPUT**:map<N,X>

The pseudo-code of the task allocation procedure based on the workload parameter is presented in algorithm 3. In the algorithm, N, T, CPI, $\boldsymbol{C_N}$, R_T, and $\boldsymbol{R_N}$ are used as input parameters (See table 4.1 for the description of the symbols.) To perform the task allocation based on workload parameter, first load on the each device is determined (line 5 algorithm 3). Subsequently, the value of the estimated load is subtracted to the total capacity of the each node to find the residual capacity of the each compute node (line 7 in algorithm 3). Thus, the workload on the each device is determined in this way and device is selected based on the residual capacity of the device. In this way, the tasks are allocated to the selected compute nodes with respect to task sizes. Although this solution can help to minimize the execution time, further reduction in execution time is also possible. In certain scenarios, workload on the low speed devices is lighter than in high-speed mobile devices; however executing the task on a high-speed mobile device could be a better option. Therefore, allocating tasks based on two parameters, such as CPU speed and workload can further shorten task execution time.

Algorithm 3: Workload-based Task Allocation Algorithm

Input: $N, T, CPI, \mathbf{C_N}, R_T, \mathbf{R_N}$

1 $X \leftarrow T$
2 Sort(Desend, X)
3 $\forall_{n=1..|N|} E^n_{tL} = 0$
4 **for** i=1:|X| **do**
5 $\forall n \in N \; map < X_i, E^n_L > \leftarrow \dfrac{|X_i| \times \frac{1}{S_n} \times CPI_n}{R_{X^n_i}} \times 100$
6 **end**
7 $\forall n \in N, \; \mathbf{R_n} = \mathbf{C_n} - E^n_{tL}$
8 **for** i=1:|X| **do**
9 while(|X| != NULL)
10 $(\widehat{w_n}) = arg\ max_{\forall n \in N, w_n \in \mathbf{R_N}} f(w_n)$
11 f(w_n) $\leftarrow w_n$
12 NodeID $\leftarrow getID(N, \hat{w_n})$
13 map<NodeID> $\leftarrow x_{\{1\}} where\ x_{\{1\}} \in X$
14 $E^n_{tL} \leftarrow E^n_{tL} + E^n_L;$
15 X $\leftarrow X/x_1$
16 **end**
17 **OUTPUT**:map<N,X>

The incorporation of CPU speed and workload when allocating tasks is discussed in the form of pseudocode in algorithm 4. In the algorithm, N, S_N, T, $\boldsymbol{C_N}$, CPI, R_T, and $\boldsymbol{R_N}$ are used as an input parameters. (See table 4.1 for the description of the symbols.) To find the residual workload on the compute node, first expected workload of each task to be executed on the compute node is calculated (lines 5-7 of algorithm 4). Afterward, a weighted average graph formula is used to incorporate the two parameters, such as CPU speed and workload when allocating tasks to the compute nodes (lines 9-11 of algorithm 4). On the one hand, the task is allocated based on the maximum values derived from the weighted average formula and on the other hand, weight is assigned to each parameter, such as CPU speed and workload, based on their significance impact that is measured as 0.05 and 0.95, respectively. These weight values are identified through measuring the impact of many combinations. This combination shows the shortest execution time, so these weight values are selected (See figure 6.6 in chapter 6).

Algorithm 4: Two Parameters-based (CPU Speed and Workload) Task Allocation Algorithm

Input: $N, S_N, T, \mathbf{C_N}, CPI, R_T, \mathbf{R_N}$

1 $X \leftarrow T$
2 Sort(Desend, X)
3 $\forall_{n=1..|N|} E^n_{tL} = 0$
4 **for** i=1:|X| **do**
5 $\forall n \in N \; map < X_i, E^n_L > \leftarrow \dfrac{|X_i| \times \frac{1}{S_n} \times CPI_n}{R_{X^n_i}} \times 100$
6 **end**
7 $\forall n \in N, \; \mathbf{R_n} = \mathbf{C_n} - E^n_{tL}$
8 while(|X| != NULL)
9 $(\widehat{w_n}, \widehat{s_n}) = arg\ max_{\forall n \in N, w_n \in \mathbf{R_N}, s_n \in S_N} f(w_n, s_n)$
10 f(w_n, s_n) $\leftarrow \alpha \times w_n + \beta \times s_n$
11 NodeID $\leftarrow getID(N, \hat{w_n}, \hat{c_n})$
12 map<NodeID> $\leftarrow x_{\{1\}} where\ x_{\{1\}} \in X$
13 $E^n_{tL} \leftarrow E^n_{tL} + E^n_L$;
14 X $\leftarrow X/x_1$
15 **end**
16 **OUTPUT**:map<N,X>

The procedure of task allocation by considering the resources and operational heterogeneity, such as CPU speed, number of cores, and workload in MAC is presented as a pseudo-code in algorithm 5 where N, S_N, T, C_N, $\mathbf{C_N}$, CPI, R_T, and R_N are used as input parameters. (See table 4.1 for the description of the symbols.) Once the controller device has received a set of tasks, they are sorted in descending order (lines 1-3 of algorithm 5). After the tasks are sorted, the devices are selected to execute the tasks according to their resources and operational heterogeneity, such as CPU speed, number of cores, and workload that are measured through weighted average formula (lines 5-12 in algorithm 5). Moreover, in order to determine the load which each task puts on the compute node is calculated through an equation (line 5 of algorithm 5). In addition, the weights are assigned to each individual parameter, such as CPU speed, cores, and, workload according to their impact which is measured as 0.15, 0.20 and 0.65, respectively. This combination of weights is chosen because it executes the task the fastest (See figure 6.7 in chapter 6). Lastly, once the compute nodes have been selected according to the defined algorithm criteria in the algorithm, workload information is updated after task execution (lines 13-14 of algorithm 5).

Algorithm 5: Three Parameters-based (CPU Speed, Workload, and Core) Task Allocation Algorithm

Input: $N, S_N, T, C_N, \mathbf{C_N}, CPI, R_T, \mathbf{R_N}$

1 $X \leftarrow T$
2 Sort(Desend, X)
3 $\forall_{n=1..|N|}\ E^n_{tL} = 0$ **for** i=1:|X| **do**
4 $\forall n \in N\ map < X_i, E^n_L > \leftarrow \dfrac{|X_i| \times \dfrac{1}{S_n} \times CPI_n}{R_{X^n_i}} \times 100$
5 **end**
6 $\forall n \in N, \mathbf{R_n} = \mathbf{C_n} - E^n_{tL}$
7 while(|X| != NULL)
8 $(\widehat{w_n}, \widehat{s_n}, \widehat{c_n}) = arg\ max_{\forall\, n \in N,\, w_n \in \mathbf{R_N}}\ s_n \in S_N,\ c_n \in C_N,\ f(w_n, s_n, c_n)$
9 f(w_n, s_n, c_n) $\leftarrow \alpha \times w_n + \beta \times s_n + \gamma \times c_n$
10 NodeID $\leftarrow getID(N, \hat{w_n}, \hat{s_n}, \hat{c_n})$
11 map<NodeID> $\leftarrow x_{\{1\}}\ where\ x_{\{1\}} \in X$
12 $E^n_{tL} \leftarrow E^n_{tL} + E^n_L$;
13 $X \leftarrow X/x_1$
14 **end**
15 **OUTPUT**:map<N,X>

4.2 MAC Framework

Figure 4.2 depicts the key components of the MAC framework: context monitor, decisioner module, and task manager. The proposed heterogeneity-aware task allocation algorithms are implemented in the task handler module. This section provides the details of the proposed task handler module. A discussion of other modules that are linked to perform execution of compute-intensive tasks in MAC paradigm is also provided.

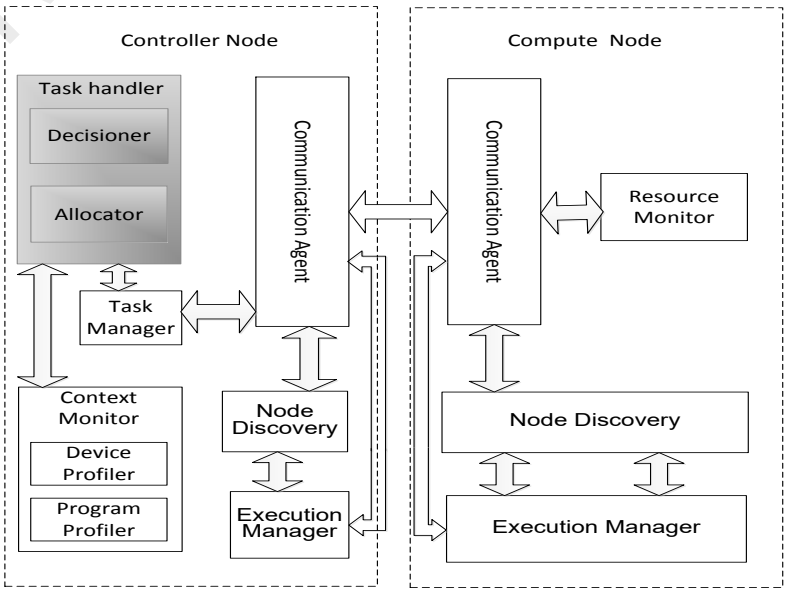

Figure 4.2: The task handler module in MAC framework

4.2.1. Context Monitor

It allows multiple parameters to be profiled at run time. The information of context-awareness gathered from the context monitor used as an input for task handler module that can lead to accurate decision making. The context monitor profiler tracks the number of instructions running on the compute nodes, data sizes of input, and workload running in the background. Moreover, the context monitor module is responsible for keeping track of the specification of the devices in terms of CPU speed and number of cores. Later, the context monitor passes this information to the task handler that is responsible for task partitioning, making a decision and allocating the task based on the criteria defined in proposed heterogeneity-aware task allocation algorithms.

4.2.2. Task Handler

This module decides where and how to send the compute-intensive tasks to compute nodes. In addition, it determines whether or not the task needs to be divided into sub-tasks. The task handler usually uses the information given by the context monitor module and passes it to the decisioner module, allocates tasks based on defined criteria as discussed in the five proposed algorithms. In the case of multi-parameters based task allocation, the weighted average formula is used. In this context, the weight of the each parameter is identified. Lastly, once the decisioner has decided where to send the task, it forwards the information to the allocator.

4.2.3. Task Manager

The task manager module is a middle layer between the allocator and communication manager. Once the allocator receives the information from the decisioner about sending the task based on the proposed solutions criteria, it transmits that information to the task manager. The task manager then sends this task to the communication manager. Later, the communication manager performs task allocation

based on the information. Once the task is allocated to the compute node, the context monitor module updates the context-awareness information into its databases. The task manager also sends the task result back to the mobile device that initiated the task execution request.

4.2.4. Communication Agent

The communication agent handles the connection and transfers the data between the controller and compute nodes according to the defined policy of task manager. Moreover, it is responsible for initiating the node discovery procedure. To form the MAC, the communication agent initiates and maintains communication between the compute and controller nodes.

4.3 Illustration of Task Handler using Sequence Diagram

Figure 4.3 shows a sequence of steps that are required for performing task allocation in the MAC environment. In the figure, CH, TAM, TH, CM, CN1, CN2, and CN3 are used to represent communication agent, task allocation manager, task handler, context monitor, compute node 1, compute node 2, and compute node 3, respectively.

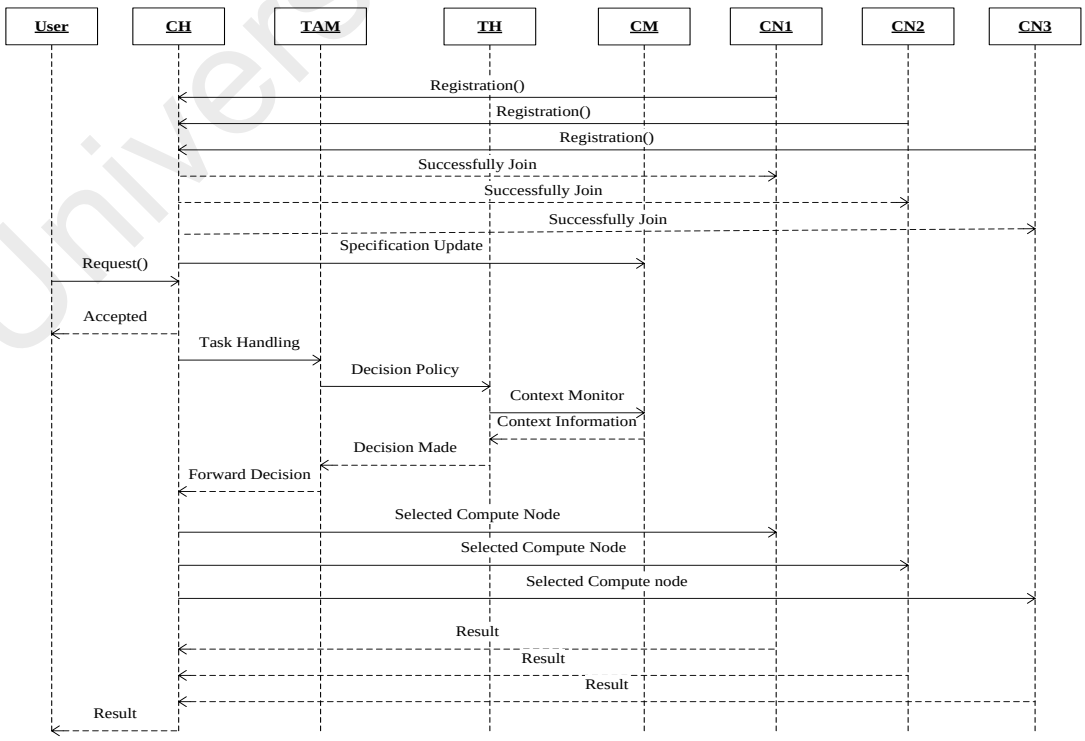

Figure 4.3: Sequence of steps for performing task allocation using task handler

First, compute nodes perform registration after receiving the node discovery request. Subsequently, MAC is formed through WiFi-direct or Bluetooth. Once MAC is formed, the nodes are divided into compute and controller nodes. One device is selected as the controller node and others are compute nodes. The responsibility of the controller node is to collect the set of tasks from the users and perform their computation on the available compute nodes. When CH receives tasks it sends them to the TAM which then forwards the tasks to the proposed TH module. The proposed module helps to select the appropriate nodes to execute the task. In this context, proposed task TH module first sends the request to CM to collect the information related to the compute nodes specifications in terms of CPU speed and number of cores. Moreover, CM keeps information about workload running in the background on the compute nodes. After receiving the request from TH, CM sends back this information to the TH module. The proposed TH module uses the information given by CM as an input to perform the selection decision. Once the TH decides where to submit the task, it passes the information to TAM. The TAM forwards this information to CH and it allocates the task to the compute nodes as defined by TH module. The selected compute node executes this task and sends it back to the CH. Later, CH sends the results back to the user who initiated the request for task execution.

4.4 Mathematical Equations for Node Selection and Calculating Energy Consumption

In these algorithms, task allocation is based on CPU speed, number of cores, and workload. In algorithms 1 and 2, task allocation is based only on high processor speed and number of cores. In these algorithms, the controller checks the specification of the available compute nodes and allocates the tasks to the compute nodes that have high CPU speed and number of cores. In Algorithm 3, task allocation is based on the workload-only parameter. According to this criterion, the device having the lightest

workload is selected for task execution. In this context, the workload is estimated on the compute nodes through equation 4.1.

$$E = On\_D_i = \left(\frac{nIS * \frac{1}{CPU\ Speed} * CPI}{real - time} \right) * 100 \tag{4.1}$$

Where e is the estimated load of the specified task and On_D_i is the workload on device i. In addition, nIS represents the number of instructions calculated through Valgrind. The value of E is updated and sent back to the controller node from time to time. In order to measure the residual capacity of the compute nodes to make the task allocation decision based on workload parameter, equation 4.2 is used. In algorithms 4 and 5, the same equation is used to calculate the workload.

$$R_j = T_i - E \tag{4.2}$$

Where R_j is the residual capacity on J node, T_i is total CPU capacity in terms of load, and E is the estimated load on the device. The value of E is 0 when there is no workload as in the start of the task allocation procedure. Once the load is estimated for the compute nodes, selection is performed using information from equation 4.3.

$$W = \sum_{1}^{n} R_j \tag{4.3}$$

In algorithm 4, task allocation is performed based on two parameters, such as CPU speed and workload. To make one metric from two parameters, the weighted average formula is used, as seen in equation 4.4. This equation helps to make one metric from two parameters. In addition, the workload value is calculated with the same equation used in the algorithm 3.

$$SD_i = \alpha \times w + \gamma \times s \tag{4.4}$$

In algorithm 5, the task allocation is performed based on three parameters: CPU speed, workload, and number of cores. To make one metric from three parameters, the weighted average formula is used, as seen in equation 4.5. This equation helps to make one metric from three parameters. In addition, the workload value is calculated with the same equation used in algorithm 3.

$$SD_i = \alpha \times w + \beta \times s + \gamma \times c \tag{4.5}$$

Algorithms 4 and 5 select the mobile device as compute nodes to perform the task allocation based on the maximum value calculated through equation 4.6. Where SD_i is the status of the device I, α is the weight of workload, β is the weight of CPU speed and γ is the weight of number of cores. To normalize the units of different variables, such as CPU speed, number of cores and workload, equation 4.7 is used, where V represents the value.

$$Selected\ Device = \max\{SD_1, SD_2, \ldots\ SD_n\} \tag{4.6}$$

$$\mathrm{f}: \mathrm{a} \rightarrow \mathrm{b} = [V_{Actual} - V_{Minimum}] * [\frac{1}{[V_{Maximum} - V_{Minimum}]}] \tag{4.7}$$

Equation 4.8 calculates the energy consumption.

$$P_c \times \frac{\mathbf{C}}{\mathrm{M}} \tag{4.8}$$

Where P_c is the power consumption of the processor when it is in an active state (i.e. when it is performing the computation, approximately 600 milliwatt (mW)), whereas $\mathbf{C}$ represents the number of instructions of the tasks and M represents the speed of mobile devices.

4.5 Mathematical model for Execution Time

The mathematical model to compute the task execution time is formally formulated in this section. The description of the symbols is provided in table 4.2.

Table 4.2: Description of the symbols used in the mathematical model

Symbol	Description
$\boldsymbol{T_i}$	ith task
I^{T_i}	Number of instructions of ith task executed in execution time slots
O^{T_i}	Number of instructions of background tasks executed in execution time slots
P	Processor speed
$\boldsymbol{T}$	Set of tasks
T_i	Execution time of ith task
C	Number of cores
S_i	Size of ith slot
x, y, a, b, l, m	Variables used in the Lambda expression

Execution time of a certain task in ith slot without background workload can be formally modeled as follows:

$$\lambda xy.\frac{x}{y}(I^{\boldsymbol{T_i}})(\lambda ab.ab)(P)(C) \tag{4.9}$$

The time of background tasks in the ith slot can be modeled using equation 4.10.

$$\left(\lambda lm.\frac{l}{m}\right)(O^{\boldsymbol{T_i}})(\lambda ab.ab)(P)(C) \tag{4.10}$$

The total task execution time can formally be modeled by merging equation 4.9 and 4.10 as can be seen in equation 4.11. The simplest form of the equation 4.11 is provided in equation 4.12.

$T_i \ \forall i=1\ldots|\boldsymbol{T}| and\ \boldsymbol{T_1}<\boldsymbol{T_2}\ldots<\boldsymbol{T}_{|\boldsymbol{T}|}$

$$= \forall\ sum(i,1)\left(\left(\lambda xy.\frac{x}{y}\right)(I^{\boldsymbol{T_i}})(\lambda ab.ab)(P)(C) + \left(\lambda lm.\frac{l}{m}\right)(O^{\boldsymbol{T_i}})(\lambda ab.ab)(P)(C)\right) \tag{4.11}$$

$$\text{Or} \qquad T_i = \sum_{i=1}^{n}\left(\frac{I^{T_i}}{P\times C} + \frac{O^i}{P\times C}\right) \tag{4.12}$$

For the calculation of the execution time of a task, it is necessary to divide the entire execution time of a task into slots. The slot size is based on the number of tasks executed in the time interval. The slot changes when either the running task execution is completed or a new task starts. Therefore, the time slot size varies with the workload on the device. Figure 4.4 shows task execution time in terms of slots.

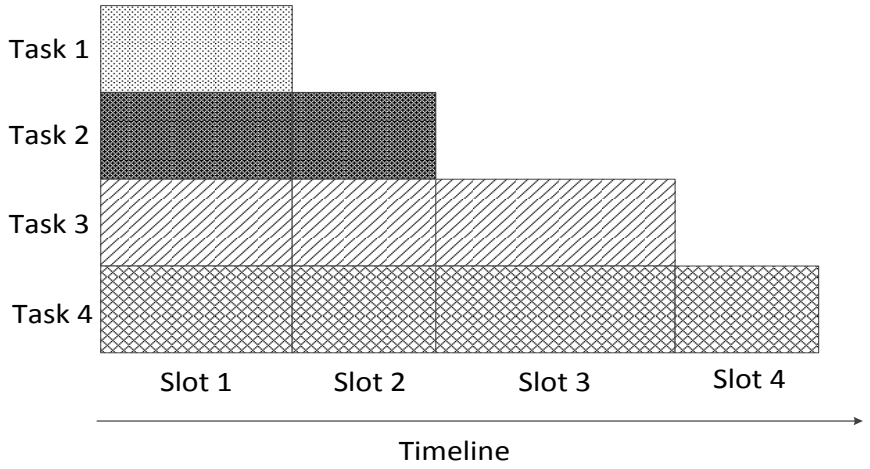

Figure 4.4: Task execution times

Where the execution time of an ith task is the sum of all the time slots taken by that task for the execution. The size of the time slot varies with the arrival of new tasks and completion of old ones. The size of the first slot depends on the number of tasks being executed in that slot and the size of the smallest task. The size of the remaining time slots depends on the difference between the sizes of the slot numbered task and the next-smaller task.

The size of the slot in terms of time can be modeled by using the number of instructions need to be executed, processor speed, P, and number of cores C. First slot size can be calculated as follows:

$$\left(\lambda\, lm.\frac{l}{m}\right)\left((\lambda xy.xy)(\min(\boldsymbol{T}))(|\boldsymbol{T}|)\right)\left((\lambda ab.ab)(P)(C)\right)\ if\ i=1 \tag{4.13}$$

The size of the rest of the slots can be measured through equation 4.14.

$$\left(\lambda\, lm.\frac{l}{m}\right)\left((\lambda vu.vu)\left((\lambda ab.a-b)(\boldsymbol{T_i})(\boldsymbol{T_{i-1}})\right)\left((\lambda xy.x-y)(|\boldsymbol{T}|)(i+1)\right)\right)\left((\lambda ab.ab)(P)(C)\right) \tag{4.14}$$

The complete form of the model used for calculating the slot size can be as follows:

$$S_i=\begin{cases}\left(\lambda\, lm.\frac{l}{m}\right)\left((\lambda xy.xy)(\min(\boldsymbol{T}))(|\boldsymbol{T}|)\right)\left((\lambda ab.ab)(P)(C)\right)\ if\ i=1\\ \left(\lambda\, lm.\frac{l}{m}\right)\left((\lambda vu.vu)\left((\lambda ab.a-b)(\boldsymbol{T_i})(\boldsymbol{T_{i-1}})\right)\left((\lambda xy.x-y)(|\boldsymbol{T}|)(i+1)\right)\right)\left((\lambda ab.ab)(P)(C)\right)\ if\ i>1\end{cases} \tag{4.15}$$

$$\text{Or}\qquad \mathrm{S_i}=\begin{cases}(\min(\boldsymbol{T})\ \times|\boldsymbol{T}|)/(P\times C)\\ (\boldsymbol{T}_i-\boldsymbol{T}_{i-1})\times\left(|\boldsymbol{T}|-(i+1)\right)/(P\times C)\end{cases}\ \text{if}\ i\geq 1 \tag{4.16}$$

4.6 Distinctive Features of the Proposed Algorithms

The proposed algorithms have five distinctive features. One of these features is that proposed algorithms incorporate the resource and operation heterogeneity in the MAC. The algorithms ensure efficient resource utilization. Moreover, the algorithms help to minimize the execution time and are very valuable when there is a deadline for tasks execution. Furthermore, the proposed algorithms consume less energy. Additional details of these features are provided in this section.

4.6.1. Resource and Operational Heterogeneity-awareness

The mobile devices forming MAC usually have different CPU speeds and numbers of cores. The proposed solutions consider the mobile device resources while performing any computation. In addition, the proposed solutions incorporate the workload running in the background as an operational heterogeneity-measuring parameter when allocating tasks to the compute nodes. The consideration of these parameters while making task allocation decision can ensure faster task execution than random-based task allocation.

4.6.2. Appropriate Resource Utilization

The proposed five solutions ensure the appropriate resource utilization in MAC environment. Once the controller node receives any task, the proposed solution helps the controller to select the device based on the defined policies in the algorithms. The proposed algorithms ensure that tasks of different lengths are executed according to the capacity of the available mobile devices. Thus, proposed solutions enable the controller to execute larger tasks on devices that have high specifications. Efficient resource utilization also helps in saving cost in terms of incentives. In the MAC environment, mobile device owners usually share their resources by taking incentives. Therefore, the resources in the MAC are needed to be considered very carefully and used efficiently.

4.6.3. Time Minimization

The proposed algorithms help to minimize tasks execution time by enabling appropriate compute node selection. The tasks of different computation lengths require different processing capabilities at the time of execution in MAC. The proposed algorithms enable the controller to assign larger tasks to devices that have high specifications and less workload running in the background. Thus, the proposed algorithms can minimize the task execution time better than random-based task allocation. In random-based task allocation, larger tasks are assigned to a device that has low processing capabilities that can prolong its execution time.

4.6.4. Deadline-based Task Execution

One of the features of proposed algorithms is that it can help in meeting the deadlines of the tasks defined by the users. The proposed algorithms enable the controller to assign tasks with short deadlines to devices that have high processing capabilities, according to the defined criteria in each algorithm. In this way, the proposed algorithms can help to meet the deadlines.

4.6.5. Energy Efficiency

One of the key features of the proposed algorithms is that they help to minimize the energy consumption. Most of the devices participating in the MAC have limited processing capabilities and battery resources. Therefore, the algorithms are designed to consume less energy. The incorporation of the heterogeneity-awareness leads to the appropriate resource utilization that results in saving energy.

4.7 Conclusion

In this chapter, we presented the proposed heterogeneity-aware task allocation solutions in the form of pseudo-codes that help to simplify the understanding of the solutions. The distinctive features of the proposed algorithms are discussed to prove their effectiveness. The proposed algorithms perform task allocation in light of the

specifications and workload running in the background on the available mobile devices that are acting as compute nodes. The first proposed solution is based on CPU speed which enables the controller node to select that compute node with respect to high CPU speed. The second algorithm bases task allocation on number of cores. The device with more cores is selected to execute the task. The third proposed algorithm helps to select that compute node that has less workload running in the background. In the fourth algorithm, task allocation is based on two parameters (CPU speed and workload) and in the fifth algorithm, it is based on three parameters (CPU speed, number of cores, and workload). The additional decrease in the execution time can be measured when task allocation is based on two and three parameters (Algorithm 4 and 5). Based on the pseudo-codes described in the proposed five algorithms and looking into their distinctive features, it is concluded that the proposed five heterogeneity-aware task allocation algorithms help to shorten the execution time and conserve energy that ensure the effectiveness of the algorithms. In the following chapters, details of the implementation of the proposed algorithms, validation of the results collected from mathematical model and simulation, and verification of five proposed solutions with random-based task allocation mechanism are discussed.

CHAPTER 5: EVALUATION

This chapter presents the data collected through proposed heterogeneity-aware task allocation solutions in MAC environment. The chapter discusses the experimental setup used to test the performance of the proposed algorithms, performance-measuring parameters and statistical methods that include the Mann-Whitney U test, Vargha and Delaney's A_{12} statistics, and Pearson's correlation coefficient. These methods (Mann-Whitney U test and A_{12} statistics) help to know whether or not the differences between the results are significant. Furthermore, descriptive statistics are applied to analyze the accuracy of the collected data.

The chapter is organized into eight sections. Section 5.2 explains the experimental setup, length of computational sub-tasks used in data traces, performance metrics, and data gathering and processing. Section 5.3 describes the evaluation methods used to access the reliability and validity of the collected data. Section 5.4 presents the data collected to validate the accuracy of the developed mathematical model by comparing the results obtained from the mathematical model with simulation results. Section 5.5 reports the differences between the execution time data obtained from each proposed and random-based task allocation solution. Section 5.6 reports the differences between the energy consumption data obtained from each proposed and random-based task allocation solution. Section 5.7 presents the data collected for the performance comparison of five heterogeneity-aware task allocation solutions with random-based task allocation. Section 5.8 concludes the chapter.

5.1 Performance Evaluation

This section presents the methodology used for the evaluation of five heterogeneity-aware task allocation solutions. In this context, details of experimental setup, length of computational sub-tasks used in data traces, and performance-measuring parameters are discussed.

5.1.1. Experimental Setup

We implemented heterogeneity-aware task allocation algorithms in distributed form of MAC environment that is simulated in MATLAB. All of the proposed algorithms are implemented on controller node in MAC group. To conduct the experiments, we have simulated four mobile devices with different specifications, discussed in table 5.1. Task allocation has been simulated by enabling controller node in MAC that distributes the tasks to compute nodes. The testing of these devices that were simulated in the MATLAB was performed by comparing the results obtained from problem analysis results on certain specifications. In this context, we ran the same tasks that were used in problem analysis, in the simulated environment and compared their execution time and energy consumption with the execution time on the real mobile devices. The execution times were nearly identical. Thus, we found that the devices were configured in the same way as real devices. Processing speed in terms of million instructions per second (MIPS) on the simulated mobile device has been matched with the speed on the real mobile devices. These speeds were calculated using equation 5.1.

$$\text{MIPS}= \frac{\text{Processor Clock Frequency}}{\text{Average Cycles Per instruction (CPI)}\times 1000000} = \frac{\text{Cycles/Second}}{\text{Cycles/Instruction}} = \frac{\text{Million instructions}}{\text{Second}} \quad (5.1)$$

Table 5.1: Specification of mobile device used in simulation

Mobile	Processor Speed	Number of Core
Mobile 1	1300MIPS	Quad-core
Mobile 2	1008MIPS	Single-core
Mobile 3	800MIPS	Dual-core
Mobile 4	600MIPS	Octa-core

Multi-threaded matrix multiplication application is used for compute-intensive tasks. The application takes a set of the matrix as an input and gives result after the multiplication. The application divides the matrix multiplication task and distributes it among the number of available mobile devices. In order to use the application in a simulated environment, we computed the number of instructions of the applications by using Valgrind's Lackey tool (Ejaz, 2016). Table 5.2 presents the computational lengths of the tasks that were used for the experiments (One task or data trace is comprised of

five sub-tasks). In addition, the mathematical model is implemented in MATLAB to generate the results.

5.2 Performance Measuring Parameters

We select execution time and energy consumption as performance-measuring parameters to evaluate the impact of proposed heterogeneity-aware task allocation solutions based on CPU speed, number of cores, workload, CPU speed plus workload (two parameters), and CPU speed plus workload and number of cores (three parameters) on task execution time and energy consumption.

The execution time of the given task is defined as the time (in seconds) spent by the system executing that task, including the time spent executing run-time or system services on its behalf. The task execution time depends on the task size, processor speed, and number of background tasks (workload) being executed on the mobile device. The execution of the compute-intensive task on the slow device can prolong the execution time and vice versa. Therefore, task allocation based on the specification of the mobile devices and workload can significantly improve task performance.

Energy consumption (in mJ) represents the amount of energy consumed to execute the task. To measure the performance of the proposed solutions, energy consumption is selected as a performance-measuring parameter.

Table 5.2: Data traces for evaluations of various parameters

Data Trace #	Computational Lengths of Five Sub-tasks (equal to one task)				
Data Trace-1	1.0e+12 × 0.0016	1.0e+12 × 0.0419	1.0e+12 × 0.2355	1.0e+12 × 0.7434	1.0e+12 × 1.9649
Data Trace-2	1.0e+12 × 0.0017	1.0e+12 × 0.0468	1.0e+12 × 0.2396	1.0e+12 × 0.7368	1.0e+12 × 1.8003
Data Trace-3	1.0e+12 × 0.0016	1.0e+12 × 0.0430	1.0e+12 × 0.2392	1.0e+12 × 0.7411	1.0e+12 × 1.9595
Data Trace-4	1.0e+12 × 0.0024	1.0e+12 × 0.0402	1.0e+12 × 0.2385	1.0e+12 × 0.7431	1.0e+12 × 1.6787
Data Trace-5	1.0e+12 × 0.0026	1.0e+12 × 0.0452	1.0e+12 × 0.2339	1.0e+12 × 0.7392	1.0e+12 × 1.1712
Data Trace-6	1.0e+12 × 0.0025	1.0e+12 × 0.0402	1.0e+12 × 0.2328	1.0e+12 × 0.7306	1.0e+12 × 1.0971
Data Trace-7	1.0e+12 × 0.0027	1.0e+12 × 0.0449	1.0e+12 × 0.2332	1.0e+12 × 0.7433	1.0e+12 × 1.0344
Data Trace-8	1.0e+12 × 0.0021	1.0e+12 × 0.0427	1.0e+12 × 0.2377	1.0e+12 × 0.7411	1.0e+12 × 1.1869
Data Trace-9	1.0e+12 × 0.0022	1.0e+12 × 0.0431	1.0e+12 × 0.2365	1.0e+12 × 0.7399	1.0e+12 × 1.7547
Data Trace-10	1.0e+12 × 0.0018	1.0e+12 × 0.0448	1.0e+12 × 0.2366	1.0e+12 × 0.7323	1.0e+12 × 1.1190
Data Trace-11	1.0e+12 × 0.0022	1.0e+12 × 0.0467	1.0e+12 × 0.2334	1.0e+12 × 0.7382	1.0e+12 × 1.2238
Data Trace-12	1.0e+12 × 0.0026	1.0e+12 × 0.0418	1.0e+12 × 0.2351	1.0e+12 × 0.7398	1.0e+12 × 1.8909
Data Trace-13	1.0e+12 × 0.0029	1.0e+12 × 0.0438	1.0e+12 × 0.2314	1.0e+12 × 0.7321	1.0e+12 × 1.2575
Data Trace-14	1.0e+12 × 0.0027	1.0e+12 × 0.0418	1.0e+12 × 0.2381	1.0e+12 × 0.7334	1.0e+12 × 1.9293
Data Trace-15	1.0e+12 × 0.0020	1.0e+12 × 0.0414	1.0e+12 × 0.2325	1.0e+12 × 0.7386	1.0e+12 × 1.4733
Data Trace-16	1.0e+12 × 0.0020	1.0e+12 × 0.0458	1.0e+12 × 0.2359	1.0e+12 × 0.7377	1.0e+12 × 1.9172
Data Trace-17	1.0e+12 × 0.0019	1.0e+12 × 0.0453	1.0e+12 × 0.2375	1.0e+12 × 0.7353	1.0e+12 × 1.5678
Data Trace-18	1.0e+12 × 0.0015	1.0e+12 × 0.0404	1.0e+12 × 0.2353	1.0e+12 × 0.7409	1.0e+12 × 1.9340
Data Trace-19	1.0e+12 × 0.0016	1.0e+12 × 0.0440	1.0e+12 × 0.2347	1.0e+12 × 0.7302	1.0e+12 × 1.3371
Data Trace-20	1.0e+12 × 0.0017	1.0e+12 × 0.0456	1.0e+12 × 0.2331	1.0e+12 × 0.7374	1.0e+12 × 1.1656
Data Trace-21	1.0e+12 × 0.0024	1.0e+12 × 0.0418	1.0e+12 × 0.2365	1.0e+12 × 0.7396	1.0e+12 × 1.7482
Data Trace-22	1.0e+12 × 0.0021	1.0e+12 × 0.0406	1.0e+12 × 0.2323	1.0e+12 × 0.7428	1.0e+12 × 1.1524
Data Trace-23	1.0e+12 × 0.0027	1.0e+12 × 0.0438	1.0e+12 × 0.2400	1.0e+12 × 0.7311	1.0e+12 × 1.4427
Data Trace-24	1.0e+12 × 0.0016	1.0e+12 × 0.0467	1.0e+12 × 0.2300	1.0e+12 × 0.7408	1.0e+12 × 1.8173
Data Trace-25	1.0e+12 × 0.0028	1.0e+12 × 0.0406	1.0e+12 × 0.2340	1.0e+12 × 0.7336	1.0e+12 × 1.8001
Data Trace-26	1.0e+12 × 0.0021	1.0e+12 × 0.0464	1.0e+12 × 0.2318	1.0e+12 × 0.7337	1.0e+12 × 1.1455
Data Trace-27	1.0e+12 × 0.0016	1.0e+12 × 0.0461	1.0e+12 × 0.2358	1.0e+12 × 0.7377	1.0e+12 × 1.1450
Data Trace-28	1.0e+12 × 0.0028	1.0e+12 × 0.0444	1.0e+12 × 0.2335	1.0e+12 × 0.7372	1.0e+12 × 1.4018
Data Trace-29	1.0e+12 × 0.0015	1.0e+12 × 0.0417	1.0e+12 × 0.2312	1.0e+12 × 0.7326	1.0e+12 × 1.2400
Data Trace-30	1.0e+12 × 0.0021	1.0e+12 × 0.0403	1.0e+12 × 0.2390	1.0e+12 × 0.7432	1.0e+12 × 1.4909

5.3 Evaluation Methods

We apply several statistical methods to access the reliability and validity of the data generated by running five heterogeneity-aware task allocation solutions and random-based task allocation. This section aims to describe the statistical methods applied on the collected data.

5.3.1. Descriptive Statistics

Descriptive statistics summarize the data. These statistics reveal the pattern among datasets, and thus allow us to reach some conclusion after analyzing the data (Ferreira, 2011). There are two types of descriptive statistics: measures of central tendency and measures of spread. In measures of central tendency, trends are captured and calculated from the data, and are expressed as mean, median, and mode. A mean shows the average of all the collected data. The median is the middle point of the data distribution. The mode value shows the common value presented in the given dataset. The measures of spread show the way in which data is distributed and relates to each other. This type of descriptive statistics usually includes, among others, a range of the values by expressing the minimum and maximum values, frequency distribution, quartiles, variance, and standard deviation. These statistics are very useful to identify trends within the data. In this study, some of the methods belong to both of the types of descriptive statistics are applied to measure the accuracy and summarizing purposes.

5.3.2. Confidence Interval

We present the execution time and energy consumption that is measured by running five proposed solutions. For reliability assurance, we iterate the data collection of each solution 30 times (30 data traces). Instead of presenting point estimate for the corresponding execution time of each data trace of proposed solutions, we use a 95% interval estimate. Therefore, we raise the confidence and reliability of results to 95% when reporting the results of execution time and energy consumption.

5.3.3. Inferential Statistics

Inferential statistics allows the identification of trends in the characteristics of a sample that is drawn from a large population (Sheskin, 2003). There are two types of inferential statistics: parametric and non-parametric. In the former, data must normally be distributed, whereas in the latter there is no assumption about such normal data distribution. In this study, non-parametric tests are conducted to determine whether or not the differences are significant.

5.3.3.1. Null Hypothesis

The null hypothesis is used in inferential statistics, a general statement that shows there is no relationship between two measured phenomena and denoted by H_0 (H-naught). It is the opposite of the alternative hypothesis. The rejection of null hypothesis ensures statistical significance in given set of values. In this context, P-value is used that helps to determine the statistical significance of the results. The value of P provides evidence to reject the null hypothesis when it is less than or equal to 0.05.

5.3.3.2. Mann-Whitney U Test

The Mann-Whitney U test is a non-parametric test, used to determine whether or not the differences between two samples are truly significant (Arcuri *et al.*, 2011). It is applied when data is not normally distributed (In the specific case of our collected data). The test helps to rank the data for each condition using equation 5.2.

$$U_x = N_x.N_y + \frac{N_x(N_x+1)}{2} - \sum r_x \quad (5.2)$$

Where U_x is the Mann-Whitney calculation for sample X, N is number of samples and Σr_x is the sum of the ranks of sample x.

5.3.3.3. Vargha and Delaney's A_{12} statistics

Vargha and Delaney's A_{12} statistics test is used to evaluate the effective size. The range of A_{12} (0-1) is divided into three categories, such as small, medium and large. "*Values of A_{12} such that $0.36 < A_{12} \leq 0.44$ or $0.56 \leq A12 < 0.64$ indicate small effect*

size. Values of A_{12} such that $0.29 < A_{12} \leq 0.36$ or $0.64 \leq A_{12} < 0.71$ indicate medium effect size. Values of b A_{12} such that $0 \leq A_{12} \leq 0.29$ or $0.71 \geq A_{12} \leq 1$ indicate large effect size. A value of 0.5 indicates no difference between the populations (Holt et al., 2014)." The equation that is used to estimate the value of A is provided below (Vargha *et al.*, 2000).

$$A_{12} = \frac{\left(\frac{R_k}{m} - \frac{(m+1)}{2}\right)}{n} \tag{5.3}$$

Where R is the rank of sample k, whereas m represents the number of samples and n represents the total entries.

5.3.3.4. Pearson's Correlation Coefficient

This statistics measure is used to find the strength of the association between two continuous variables (Ahlgren *et al.*, 2003). To investigate this relationship, scatter plot is used to check the linearity. When there is no linearity, the correlation coefficient cannot be calculated because it shows no relationship between two variables. Moreover, in terms of axis representation any variable can be plotted on the x- or y-axis. Once this statistics measure is applied, the value of Pearson's r indicates the strength of the relationship between two variables; an r that is close to 1 indicates a strong association. The scattered point plotted through the Pearson's correlation coefficient must be near the straight line, when there is a strong association between two variables. We used this measure to validate the model results.

5.4 Data Collected For Mathematical model Validation

The correctness of the developed mathematical model is validated by comparing the results obtained from the experimental results. The execution time is used as a parameter for validation of the mathematical model. To ascertain that there are no significant differences between the results obtained from simulation and mathematical model, three statistical methods – Pearson's correlation coefficient, Mann-Whitney U

test, and Vargha and Delaney's A_{12} statistics – are applied. The reason of applying three statistical methods is to perform rigorous verification of the differences and similarity between the mathematical model and simulation results.

Figure 5.1 presents the results obtained from the Pearson's correlation. The x and y axes represent the execution time results (in seconds) obtained through simulation and mathematical model of CPU speed-based task allocation solution, respectively. The statistical method reveals that the results obtained from simulation and mathematical model are very closely related as the value of Pearson's r is 0.99618. Figure 5.1 indicates that scattered points of two variables are very close to a straight line. Hence, the small difference validates the model results.

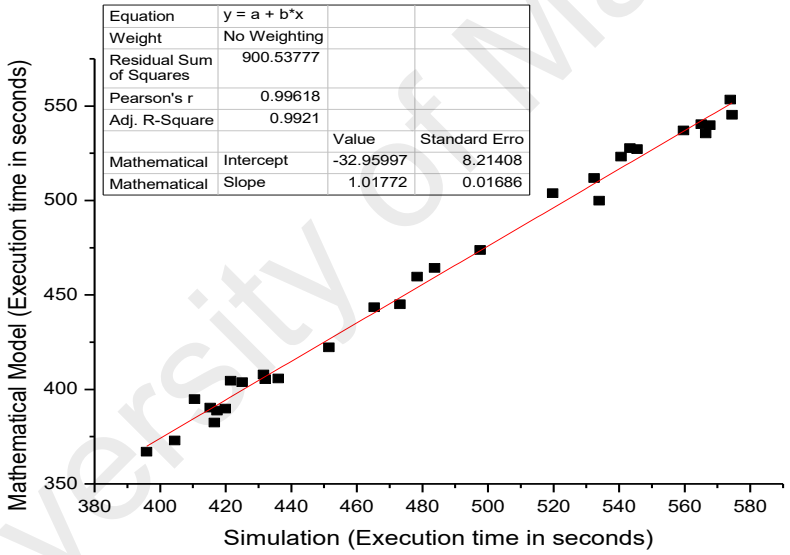

Figure 5.1: Pearson's correlation coefficient results (CPU speed-based task allocation)

Table 5.3 presents the results obtained through mathematical model and simulation of CPU speed-based task allocation to perform the validation. The first row represents the execution time and the first column represents the different data traces. To prove that there is no significant difference between the execution time obtained from model and simulation, Mann-Whitney U test and Vargha and Delaney's A_{12} statistics are applied. The results of the tests indicate that differences in execution time measured through the mathematical model and simulation results are not significant (as

the P value found greater than 0.05, and effect size is small). Hence, the small amount of difference validates the model results.

Table 5.3: Validation of mathematical model with simulation results (execution time) of CPU speed-based task allocation.

Data Trace #	Simulation Results	Mathematical model
Data Trace-1	574.4807692	545.3598475
Data Trace -2	543.3076923	527.6710354
Data Trace -3	573.9230769	553.3846172
Data Trace -4	519.7884615	503.8650337
Data Trace -5	421.5576923	404.6150567
Data Trace -6	404.4615385	372.9923819
Data Trace -7	395.8653846	366.9688122
Data Trace -8	425.0961538	403.7541642
Data Trace -9	533.9230769	499.9186359
Data Trace -10	410.4807692	394.7918476
Data Trace -11	431.5961538	407.8212667
Data Trace -12	559.6538462	537.022677
Data Trace -13	436.0961538	405.7858181
Data Trace -14	566.4038462	535.4998481
Data Trace -15	478.4230769	459.6856248
Data Trace -16	565.1153846	540.3200967
Data Trace -17	497.6538462	473.7421221
Data Trace -18	567.7115385	539.7852783
Data Trace -19	451.4615385	422.2742418
Data Trace -20	419.8846154	389.7908817
Data Trace -21	532.4038462	511.8833446
Data Trace -22	417.3461538	388.7521003
Data Trace -23	473.1346154	445.0326553
Data Trace -24	545.4615385	527.2093038
Data Trace -25	540.5961538	523.2162002
Data Trace -26	415.2884615	390.3211805
Data Trace -27	416.5769231	382.3820439
Data Trace -28	465.3269231	443.5192085
Data Trace -29	432.1153846	405.4100296
Data Trace -30	483.75	464.2737612
P-Value (Mann-Whitney U Test)	**0.06288**	
Z-Score	**1.85545**	
Vargha and Delaney A_{12}	**0.64**	
Effect Size	**Small**	
Significant Difference	**No**	

Figure 5.2 presents the results obtained from the Pearson's correlation. The x and y axes represent the execution time results (in seconds) obtained through simulation and mathematical model of core-based task allocation solution, respectively. The statistical method reveals that the results obtained from simulation and mathematical model are very closely related as the value of Pearson's r is 0.99781. Figure 5.2 indicates that scattered points of two variables are very close to a straight line. Hence, the small difference validates the model results.

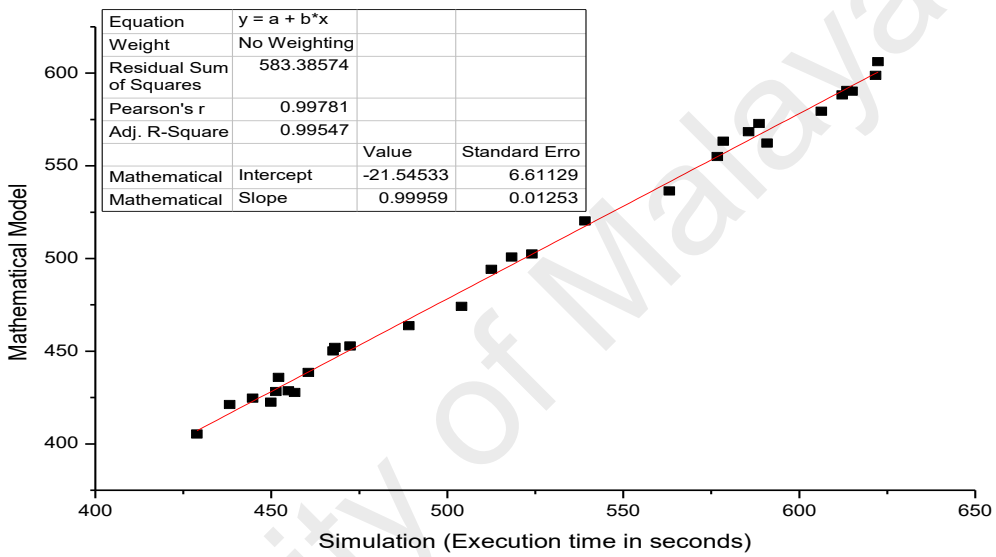

Figure 5.2: Pearson's correlation coefficient results (Core-based task allocation)

Table 5.4 presents the correctness of the developed mathematical model that is validated by comparing the simulation results obtained from core-based task allocation. The first row represents the execution time and the first column represents the different data traces. To prove that there is no significant difference between the execution time obtained from model and simulation, Mann-Whitney U test and Vargha and Delaney's A_{12} statistics are applied. The results of the tests indicate that differences in execution time measured through the mathematical model and simulation results are not significant (as the P value found greater than 0.05, and effect size is small). Hence, the small amount of difference validates the model results.

Table 5.4: Validation of mathematical model with simulation results (execution time) of core-based task allocation.

Data Trace #	Simulation Results	Mathematical model
Data Trace-1	622.3542	606.2164
Data Trace -2	588.5833	572.7741
Data Trace -3	621.75	598.788
Data Trace -4	563.1042	536.4167
Data Trace -5	456.6875	427.6773
Data Trace -6	438.1667	421.2181
Data Trace -7	428.8542	405.3218
Data Trace -8	460.5208	438.48
Data Trace -9	578.4167	563.2381
Data Trace -10	444.6875	424.6307
Data Trace -11	467.5625	450.1298
Data Trace -12	606.2917	579.3774
Data Trace -13	472.4375	452.7693
Data Trace -14	613.6042	590.6762
Data Trace -15	518.2917	500.8069
Data Trace -16	612.2083	588.1786
Data Trace -17	539.125	520.1804
Data Trace -18	615.0208	590.2096
Data Trace -19	489.0833	463.7451
Data Trace -20	454.875	428.6527
Data Trace -21	576.7708	555.0127
Data Trace -22	452.125	435.8677
Data Trace -23	512.5625	494.1278
Data Trace -24	590.9167	562.2166
Data Trace -25	585.6458	568.3602
Data Trace -26	449.8958	422.5086
Data Trace -27	451.2917	428.2165
Data Trace -28	504.1042	474.1621
Data Trace -29	468.125	451.9524
Data Trace -30	524.0625	502.4223
P-Value (Mann-Whitney U Test)	**0.1096**	
Z-Score	**1.60411**	
Vargha and Delaney A_{12}	**0.621111**	
Effect Size	**Small**	
Significant Difference	**No**	

Figure 5.3 presents the results obtained from the Pearson's correlation. The x and y axes represent the execution time (in seconds) results obtained through simulation and mathematical model of workload-based task allocation solution, respectively. The statistical method reveals that the results obtained from simulation and mathematical

model are very closely related as the value of Pearson's r is 0.99606. Figure 5.3 indicates that scattered points of two variables are very close to a straight line. Hence, the small difference validates the model results.

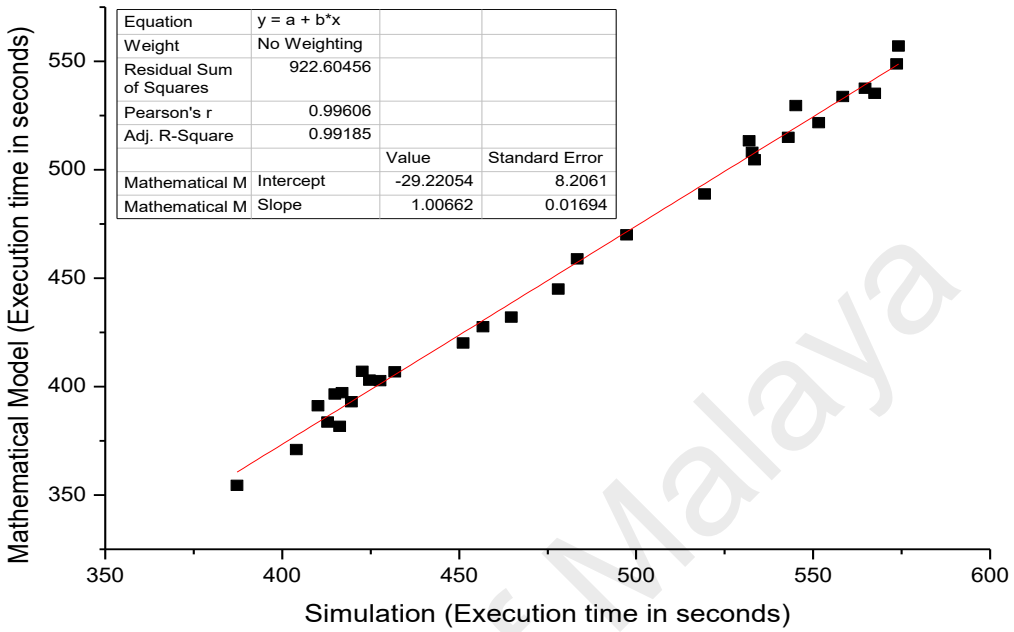

Figure 5.3: Pearson's correlation coefficient results (Workload-based task allocation)

Table 5.5 presents the results obtained from mathematical model and simulation of workload-based task allocation to perform the validation. The first row represents the execution time and the first column represents the different data traces. To prove that there is no significant difference between the execution time obtained from model and simulation, Mann-Whitney U test and Vargha and Delaney's A_{12} statistics are applied. The results of the tests indicate that differences in execution time measured through the mathematical model and simulation results are not significant (as the P value found greater than 0.05, and effect size is small). Hence, the small amount of difference validates the model results.

Table 5.5: Validation of mathematical model with simulation results (execution time) of workload-based task allocation

Data Trace #	Simulation Results	Mathematical model
Data Trace-1	574.1731	557.0378
Data Trace -2	542.9808	514.9056
Data Trace -3	573.6154	548.7319
Data Trace -4	519.3269	488.7459
Data Trace -5	412.8654	383.5646
Data Trace -6	403.9808	370.9064
Data Trace -7	387.2308	354.4123
Data Trace -8	424.6923	403.009
Data Trace -9	533.5	504.5251
Data Trace -10	410.1346	391.1784
Data Trace -11	422.6154	407.0046
Data Trace -12	551.6154	521.7339
Data Trace -13	427.6731	402.6726
Data Trace -14	558.3654	533.7669
Data Trace -15	478.0385	444.944
Data Trace -16	564.7308	537.5334
Data Trace -17	497.2885	469.9351
Data Trace -18	567.4231	535.2342
Data Trace -19	451.1538	420.0441
Data Trace -20	419.5577	393.0233
Data Trace -21	531.9423	513.2839
Data Trace -22	416.9423	397.1437
Data Trace -23	464.7115	431.9813
Data Trace -24	545.1538	529.5804
Data Trace -25	532.7885	507.9904
Data Trace -26	414.8846	396.5261
Data Trace -27	416.2692	381.6956
Data Trace -28	456.7885	427.5346
Data Trace -29	431.8269	406.8175
Data Trace -30	483.3462	458.9244
P-Value (Mann-Whitney U Test)	**0.06148**	
Z-Score	**1.87023**	
Vargha and Delaney A_{12}	**0.641111**	
Effect Size	**Small**	
Significant Difference	**No**	

Figure 5.4 presents the results obtained from the Pearson's correlation. The x and y axes represent execution time (in seconds) results obtained through simulation and mathematical model of two parameters (CPU speed and workload) based task allocation solution, respectively. The statistical method reveals that the results obtained from simulation and mathematical model are very closely related as the value of Pearson's r is 0.99249. Figure 5.4 indicates that scattered points of two variables are very close to a straight line. Hence, the small difference validates the model results.

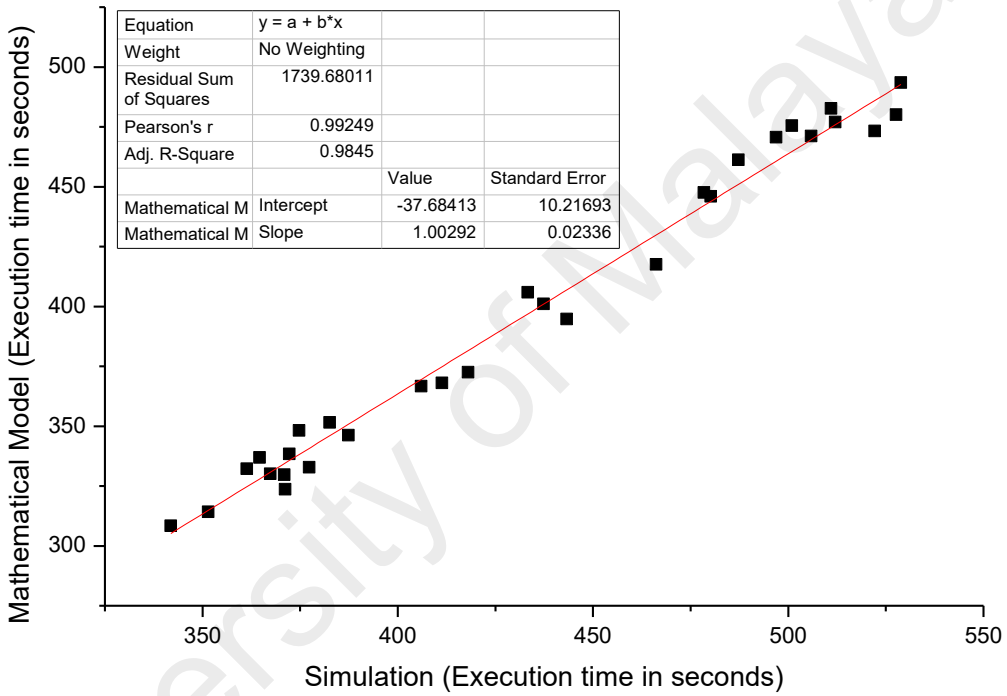

Figure 5.4: Pearson's correlation coefficient results (CPU Speed and workload-based task allocation)

Table 5.6 presents the results obtained through mathematical model and simulation of two parameters (CPU speed and workload) based task allocation to perform the validation. The first row represents the execution time and the first column represents the different data traces. To prove that there is no significant difference between the execution time obtained from model and simulation, Mann-Whitney U test and Vargha and Delaney's A_{12} statistics are applied. The results of the tests indicate that differences in execution time measured through the mathematical model and experimental are not significant (as the P value found greater than 0.05, and effect size is medium). Hence, the small amount of difference validates the model results.

Table 5.6: Validation of mathematical model with simulation results (execution time) of two parameters (CPU speed and workload) based task allocation

Data Trace #	Simulation Results	Mathematical model
Data Trace-1	528.8846	493.4529
Data Trace -2	496.9038	470.6625
Data Trace -3	527.6154	480.0475
Data Trace -4	466.1923	417.5726
Data Trace -5	367.3846	330.113
Data Trace -6	351.4808	314.2495
Data Trace -7	341.8654	308.4224
Data Trace -8	371.1731	323.6717
Data Trace -9	480.1538	445.9227
Data Trace -10	364.6346	336.8545
Data Trace -11	377.3077	332.8014
Data Trace -12	505.9038	471.1604
Data Trace -13	382.6154	351.5731
Data Trace -14	512.0577	476.9599
Data Trace -15	433.3269	405.9156
Data Trace -16	510.9423	482.643
Data Trace -17	443.2692	394.718
Data Trace -18	522.1731	473.2697
Data Trace -19	406.0192	366.639
Data Trace -20	374.7308	348.2363
Data Trace -21	478.4231	447.5536
Data Trace -22	372.2692	338.4403
Data Trace -23	418.0385	372.5086
Data Trace -24	500.9231	475.538
Data Trace -25	487.25	461.1744
Data Trace -26	361.3846	332.1599
Data Trace -27	370.9231	329.6952
Data Trace -28	411.3462	368.0531
Data Trace -29	387.3654	346.1717
Data Trace -30	437.3846	401.1115
P-Value (Mann-Whitney U Test)	**0.0139**	
Z-Score	**2.46161**	
Vargha and Delaney A_{12}	**0.685556**	
Effect Size	**Medium**	
Significant Difference	**No**	

Figure 5.5 presents the results obtained from the Pearson's correlation. The x and y axes represent the execution time results obtained through simulation and mathematical model of three parameters (CPU speed, core, and workload) based task allocation solution, respectively. The statistical method reveals that the results obtained from simulation and mathematical model are very closely related as the value of Pearson's r is 0.99781. Figure 5.5 indicates that scattered points of two variables are very close to a straight line. Hence, the small difference validates the model results.

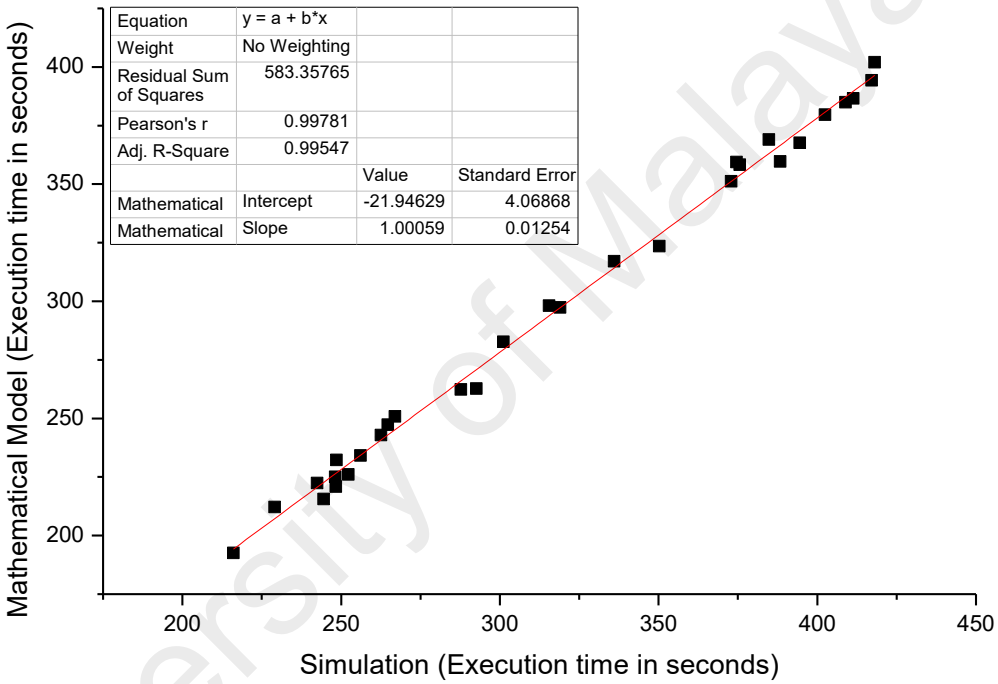

Figure 5.5: Pearson's correlation coefficient results (CPU speed, core, and workload-based task allocation).

Table 5.7 presents the results obtained through mathematical model and simulation of three parameters (CPU speed, core, and workload) based task allocation to perform the validation. The first row represents the execution time and the first column represents the different data traces. To prove that there is no significant difference between the execution time obtained from model and simulation, Mann-Whitney U test and Vargha and Delaney's A_{12} statistics are applied. The results of the tests indicate that differences in execution time measured through the mathematical model and experimental are not significant (as the P value found greater than 0.05, and effect size is small). Hence, the small amount of difference validates the model results.

Table 5.7: Validation of mathematical model with simulation results (execution time) of three parameters (Speed, core, and workload) based task allocation

Data Trace #	Simulation Results	Mathematical model
Data Trace-1	418.0833	401.9455
Data Trace -2	384.8125	369.0032
Data Trace -3	417.1875	394.2255
Data Trace -4	350.2292	323.5417
Data Trace -5	244.5417	215.5315
Data Trace -6	229.0833	212.1347
Data Trace -7	216.0625	192.5301
Data Trace -8	256.1667	234.1258
Data Trace -9	374.5417	359.3631
Data Trace -10	242.4583	222.4015
Data Trace -11	264.6875	247.2548
Data Trace -12	394.4792	367.5649
Data Trace -13	262.5833	242.9151
Data Trace -14	402.5	379.572
Data Trace -15	315.5625	298.0778
Data Trace -16	408.9583	384.9286
Data Trace -17	336.0625	317.1179
Data Trace -18	411.3333	386.5221
Data Trace -19	287.7292	262.3909
Data Trace -20	252.3333	226.1111
Data Trace -21	372.9167	351.1585
Data Trace -22	248.5417	232.2843
Data Trace -23	301.125	282.6903
Data Trace -24	388.3333	359.6333
Data Trace -25	375.6042	358.3185
Data Trace -26	248.3125	220.9252
Data Trace -27	248.1458	225.0707
Data Trace -28	292.625	262.683
Data Trace -29	267.0208	250.8482
Data Trace -30	319	297.3598
P-Value (Mann-Whitney U Test)	**0.12356**	
Z-Score	**1.54497**	
Vargha and Delaney A_{12}	**0.616667**	
Effect Size	**Small**	
Significant Difference	**No**	

5.5 Data Collected for Analyzing the Impact of Heterogeneity-aware Task Allocation on Execution Time

This section presents the execution time data, collected by running the proposed solutions and random-based task allocation. The Mann-Whitney U test and Vargha and Delaney statistics are used to analyze the differences between the collected data. In the collected data tables, column attributes represent the execution time and row attributes represent the data traces. Moreover, symbols such as SO, CO, WO, SW, and SCW are used to represent the proposed solutions that are based on CPU speed, number of cores, workload, CPU speed and workload, and CPU speed, number of cores and workload, respectively. RM represents a random-based task allocation.

5.5.1. SO vs. RM

Table 5.8 presents execution time data that is collected by running 30 data traces using SO- and RM-based task allocation. The Mann-Whitney U test and Vargha and Delaney statistics A_{12} are applied to measure whether the differences are significant or not. These tests help to find the significant differences and effectiveness in 30 data traces of SO, in comparison with RM. The objective of these tests was to fail the following null hypothesis:

$H_{0\text{-execution time}}$: There are no differences in execution time measured through SO when compared with RM.

The rejection of the null hypothesis ensures that the proposed solution outperforms the RM. The P-value measured through Mann-Whitney U test that was found less than 0.05. This value indicates that SO has significant differences from RM in execution time results. At the same time, the A_{12} statistics also reveals a large effect on execution time results. These two tests conclude that the proposed solution has significant statistical and practical differences from RM that leads to the rejection of the null hypothesis.

Table 5.8: Data collected through SO- and RM-based task allocation

Data Trace #	RM	SO
Data Trace-1	1949.305556	574.4807692
Data Trace -2	1125.1875	543.3076923
Data Trace -3	2916.468254	573.9230769
Data Trace -4	465.9375	519.7884615
Data Trace -5	398	421.5576923
Data Trace -6	1813.194444	404.4615385
Data Trace -7	341.8653846	395.8653846
Data Trace -8	228.6538462	425.0961538
Data Trace -9	1975.396825	533.9230769
Data Trace -10	1005.654762	410.4807692
Data Trace -11	1214.087302	431.5961538
Data Trace -12	1183.4375	559.6538462
Data Trace -13	729.1666667	436.0961538
Data Trace -14	460.8125	566.4038462
Data Trace -15	2194.345238	478.4230769
Data Trace -16	1226.875	565.1153846
Data Trace -17	1555.357143	497.6538462
Data Trace -18	1918.650794	567.7115385
Data Trace -19	835.6875	451.4615385
Data Trace -20	224.1538462	419.8846154
Data Trace -21	1118.75	532.4038462
Data Trace -22	490.9375	417.3461538
Data Trace -23	901.6875	473.1346154
Data Trace -24	491.9423077	545.4615385
Data Trace -25	2017.956349	540.5961538
Data Trace -26	1136.40873	415.2884615
Data Trace -27	965.7738095	416.5769231
Data Trace -28	962.9960317	465.3269231
Data Trace -29	726.7857143	432.1153846
Data Trace -30	931.8125	483.75
P-Value (Mann-Whitney U Test)	**0.000058**	
Vargha and Delaney A_{12}	**0.802222222**	
Effect Size	**Large**	
Result	**SO > RM**	
Significance	**Yes**	

5.5.2. CO vs. RM

Table 5.9 presents the execution time data that is collected by running thirty data traces using CO- and RM-based task allocation. The Mann-Whitney U test and Vargha and Delaney statistics A_{12} are applied to measure whether the differences are significant

or not. These tests help to find the significant differences and effectiveness in 30 data traces of CO, in comparison with RM. The objective of these tests was to fail the following null hypothesis:

$H_{0\text{-execution time}}$: There are no differences in execution time measured through CO when compared with RM.

Table 5.9: Data collected through CO- and RM-based task allocation

Data Trace #	RM	CO
Data Trace-1	1949.305556	622.3541667
Data Trace -2	1125.1875	588.5833333
Data Trace -3	2916.468254	621.75
Data Trace -4	465.9375	563.1041667
Data Trace -5	398	456.6875
Data Trace -6	1813.194444	438.1666667
Data Trace -7	341.8653846	428.8541667
Data Trace -8	228.6538462	460.5208333
Data Trace -9	1975.396825	578.4166667
Data Trace -10	1005.654762	444.6875
Data Trace -11	1214.087302	467.5625
Data Trace -12	1183.4375	606.2916667
Data Trace -13	729.1666667	472.4375
Data Trace -14	460.8125	613.6041667
Data Trace -15	2194.345238	518.2916667
Data Trace -16	1226.875	612.2083333
Data Trace -17	1555.357143	539.125
Data Trace -18	1918.650794	615.0208333
Data Trace -19	835.6875	489.0833333
Data Trace -20	224.1538462	454.875
Data Trace -21	1118.75	576.7708333
Data Trace -22	490.9375	452.125
Data Trace -23	901.6875	512.5625
Data Trace -24	491.9423077	590.9166667
Data Trace -25	2017.956349	585.6458333
Data Trace -26	1136.40873	449.8958333
Data Trace -27	965.7738095	451.2916667
Data Trace -28	962.9960317	504.1041667
Data Trace -29	726.7857143	468.125
Data Trace -30	931.8125	524.0625
P-Value (Mann-Whitney U Test)	**0.000173**	
Vargha and Delaney A_{12}	**0.782222222**	
Effect Size	**Large**	
Result	**CO > RM**	
Significance	**Yes**	

The rejection of the null hypothesis ensures that the proposed solution outperforms the RM. The P-value derived through Mann-Whitney U test is measured as 0.000173. This value indicates that CO has significant differences from RM in execution time results. On the other hand, the A_{12} statistics also reveals a large effect on execution time results. These two tests conclude that the proposed solution has significant statistical and practical differences from RM that leads to the rejection of the null hypothesis.

5.5.3. WO vs. RM

Table 5.10 presents execution time data that is collected by running thirty data traces using WO- and RM-based task allocation. The Mann-Whitney U test and Vargha and Delaney statistics A_{12} are applied to measure whether the differences are significant or not. These tests help to find the significant differences and effectiveness in 30 data traces of CO, in comparison with RM. The objective of these tests was to fail the following null hypothesis:

$H_{0\text{-execution time}}$: There are no differences in execution time measured through WO when compared with RM.

The rejection of the null hypothesis ensures proposed that the solution outperforms the RM. The P-value derived through Mann-Whitney U test is measured as 0.000051. This value indicates that WO has significant differences from RM in execution time results. On the other hand, A_{12} statistics also reveals a large size effect on execution time results. These two tests conclude that the proposed solution have significant statistical and practical differences from RM that leads to the rejection of the null hypothesis.

Table 5.10: Data Collected through WO- and RM-based task allocation

Data Trace #	RM	WO
Data Trace-1	1949.305556	574.1730769
Data Trace -2	1125.1875	542.9807692
Data Trace -3	2916.468254	573.6153846
Data Trace -4	465.9375	519.3269231
Data Trace -5	398	412.8653846
Data Trace -6	1813.194444	403.9807692
Data Trace -7	341.8653846	387.2307692
Data Trace -8	228.6538462	424.6923077
Data Trace -9	1975.396825	533.5
Data Trace -10	1005.654762	410.1346154
Data Trace -11	1214.087302	422.6153846
Data Trace -12	1183.4375	551.6153846
Data Trace -13	729.1666667	427.6730769
Data Trace -14	460.8125	558.3653846
Data Trace -15	2194.345238	478.0384615
Data Trace -16	1226.875	564.7307692
Data Trace -17	1555.357143	497.2884615
Data Trace -18	1918.650794	567.4230769
Data Trace -19	835.6875	451.1538462
Data Trace -20	224.1538462	419.5576923
Data Trace -21	1118.75	531.9423077
Data Trace -22	490.9375	416.9423077
Data Trace -23	901.6875	464.7115385
Data Trace -24	491.9423077	545.1538462
Data Trace -25	2017.956349	532.7884615
Data Trace -26	1136.40873	414.8846154
Data Trace -27	965.7738095	416.2692308
Data Trace -28	962.9960317	456.7884615
Data Trace -29	726.7857143	431.8269231
Data Trace -30	931.8125	483.3461538
P-Value (Mann-Whitney U Test):	**0.000051**	
Vargha and Delaney A_{12}	**0.804444444**	
Effect Size:	**Large**	
Result	**WO > RM**	
Significance	**Yes**	

5.5.4. SW vs. RM

Table 5.11 presents execution time data that is collected by running thirty data traces using SW- and RM-based task allocation. The Mann-Whitney U test and Vargha and Delaney statistics A_{12} are applied to measure whether the differences are significant or not. These tests help to find the significant differences and effectiveness in 30 data

traces of SW, in comparison with RM. The objective of these tests was to fail the following null hypothesis:

$H_{0\text{-execution time}}$: There are no differences in execution time measured through SW when compared with RM.

Table 5.11: Data collected through SW- and RM-task allocation

Data Trace #	RM	SW
Data Trace-1	1949.305556	528.8846154
Data Trace -2	1125.1875	496.9038462
Data Trace -3	2916.468254	527.6153846
Data Trace -4	465.9375	466.1923077
Data Trace -5	398	367.3846154
Data Trace -6	1813.194444	351.4807692
Data Trace -7	341.8653846	341.8653846
Data Trace -8	228.6538462	371.1730769
Data Trace -9	1975.396825	480.1538462
Data Trace -10	1005.654762	364.6346154
Data Trace -11	1214.087302	377.3076923
Data Trace -12	1183.4375	505.9038462
Data Trace -13	729.1666667	382.6153846
Data Trace -14	460.8125	512.0576923
Data Trace -15	2194.345238	433.3269231
Data Trace -16	1226.875	510.9423077
Data Trace -17	1555.357143	443.2692308
Data Trace -18	1918.650794	522.1730769
Data Trace -19	835.6875	406.0192308
Data Trace -20	224.1538462	374.7307692
Data Trace -21	1118.75	478.4230769
Data Trace -22	490.9375	372.2692308
Data Trace -23	901.6875	418.0384615
Data Trace -24	491.9423077	500.9230769
Data Trace -25	2017.956349	487.25
Data Trace -26	1136.40873	361.3846154
Data Trace -27	965.7738095	370.9230769
Data Trace -28	962.9960317	411.3461538
Data Trace -29	726.7857143	387.3653846
Data Trace -30	931.8125	437.3846154
P-Value (Mann-Whitney U Test):	**0.000008**	
Vargha and Delaney A_{12}	**0.836111111**	
Effect Size:	**Large**	
Result	**SW > RM**	
Significance	**Yes**	

The rejection of the null hypothesis ensures that the proposed solution outperforms the RM. The P-value derived through Mann-Whitney U test is measured as 0.000008. This value shows that SW has significant differences from RM in execution time results. At the same time, A_{12} statistics also reveals a large effect on execution time results. These two tests conclude that the proposed solution has significant statistical and practical differences from RM that leads to the rejection of the null hypothesis.

5.5.5. SCW vs. RM

Table 5.12 presents execution time data that is collected by running thirty data traces using SCW- and RM-based task allocation. The Mann-Whitney U test and Vargha and Delaney statistics A_{12} are applied to measure whether the differences are significant or not. These tests help to find the significant differences and effectiveness in 30 data traces of SCW, in comparison with RM. The objective of these tests was to fail the following null hypothesis:

$H_{0\text{-execution time}}$: There are no differences in execution time measured through SCW when compared with RM.

The rejection of the null hypothesis ensures that the proposed solution outperforms the RM. The P-value derived through Mann-Whitney U test that is measured as 0.000000029537. This value indicates SCW has significant differences from RM in execution time results. On the other hand, A_{12} statistics also reveals a large effect size on execution time results. These two tests conclude that the proposed solution has significant statistical and practical differences from RM that leads to the rejection of the null hypothesis.

Table 5.12: Data collected through SCW- and RM-based task allocation

Data Trace #	RM	SCW
Data Trace-1	1949.305556	418.0833
Data Trace -2	1125.1875	384.8125
Data Trace -3	2916.468254	417.1875
Data Trace -4	465.9375	350.2292
Data Trace -5	398	244.5417
Data Trace -6	1813.194444	229.0833
Data Trace -7	341.8653846	216.0625
Data Trace -8	228.6538462	256.1667
Data Trace -9	1975.396825	374.5417
Data Trace -10	1005.654762	242.4583
Data Trace -11	1214.087302	264.6875
Data Trace -12	1183.4375	394.4792
Data Trace -13	729.1666667	262.5833
Data Trace -14	460.8125	402.5
Data Trace -15	2194.345238	315.5625
Data Trace -16	1226.875	408.9583
Data Trace -17	1555.357143	336.0625
Data Trace -18	1918.650794	411.3333
Data Trace -19	835.6875	287.7292
Data Trace -20	224.1538462	252.3333
Data Trace -21	1118.75	372.9167
Data Trace -22	490.9375	248.5417
Data Trace -23	901.6875	301.125
Data Trace -24	491.9423077	388.3333
Data Trace -25	2017.956349	375.6042
Data Trace -26	1136.40873	248.3125
Data Trace -27	965.7738095	248.1458
Data Trace -28	962.9960317	292.625
Data Trace -29	726.7857143	267.0208
Data Trace -30	931.8125	319
P-Value (Mann-Whitney U Test):	**0.000000029537**	
Vargha and Delaney A_{12}	**0.916666667**	
Effect Size:	**Large**	
Result	**SCW > RM**	
Significance	**Yes**	

5.6 Data Collected for Analyzing the Impact of Heterogeneity-aware Task Allocation on Energy Consumption

This section presents the energy consumption data, collected from running five proposed solutions. The Mann-Whitney U test and Vargha and Delaney statistics are

used to analyze the differences between the collected data. In the data collection tables, columns represent energy consumption and row attributes represent data traces.

5.6.1. SO vs. RM

Table 5.13 presents energy consumption data that is collected by running 30 data traces using SO- and RM-based task allocation. The Mann-Whitney U test and Vargha and Delaney statistics A_{12} are applied to measure whether the differences are significant or not. These tests help to identify the significant differences and effectiveness in 30 data traces of SO, in comparison with RM. The objective of these tests was to reject the following null hypothesis:

$H_{0\text{-energyconsumption}}$: There are no differences in energy consumption data obtained through SO when compared with RM.

The rejection of the null hypothesis ensures that the proposed solutions outperform the RM. The P-value derived through Mann-Whitney U test that is measured as 0.000012. This value indicates that SO has significant differences from RM in energy consumption results. The A_{12} statistics also reveals a large effect on energy consumption results. These two tests conclude that the proposed solution has significant statistical and practical differences from RM that results in the rejection of the null hypothesis.

Table 5.13: Energy consumption data collected through CPU SO- and RM-based task allocation

Data Trace #	RM	SO
Data Trace-1	2259093	344688.5
Data Trace -2	1655913	325984.6
Data Trace -3	1054491	344353.8
Data Trace -4	2013722	311873.1
Data Trace -5	1500307	252934.6
Data Trace -6	190264.5	242676.9
Data Trace -7	917765.8	237519.2
Data Trace -8	1272169	255057.7
Data Trace -9	1969781	320353.8
Data Trace -10	950930	246288.5
Data Trace -11	595363.1	258957.7
Data Trace -12	228943.2	335792.3

Table 5.13: continued,

Data Trace #	RM	SO
Data Trace -13	842041.1	261657.7
Data Trace -14	651022.6	339842.3
Data Trace -15	1197252	287053.8
Data Trace -16	669060.6	339069.2
Data Trace -17	315580.4	298592.3
Data Trace -18	862806.1	340626.9
Data Trace -19	627752.1	270876.9
Data Trace -20	1678109	251930.8
Data Trace -21	2071569	319442.3
Data Trace -22	522465.5	250407.7
Data Trace -23	315525	283880.8
Data Trace -24	939443.6	327276.9
Data Trace -25	162178.8	324357.7
Data Trace -26	1038804	249173.1
Data Trace -27	1314719	249946.2
Data Trace -28	311803.1	279196.2
Data Trace -29	189995.1	259269.2
Data Trace -30	469763.5	290250
P-Value (Mann-Whitney U Test)	**0.000012**	
Vargha and Delaney A_{12}	**0.828889**	
Effect Size	**Large**	
Result	**SO > RM**	
Significance	**Yes**	

5.6.2. CO vs. RM

Table 5.14 presents energy consumption data that is collected by running 30 data traces using CO- and RM-based task allocation. The Mann-Whitney U test and Vargha and Delaney statistics A_{12} are applied to measure whether the differences are significant or not. These tests help to identify the significant differences and effectiveness in 30 data traces of CO, in comparison with RM. The objective of these tests was to reject the following null hypothesis:

$H_{0\text{-energyconsumption}}$: There are no differences in energy consumption data obtained from CO when compared with RM.

Table 5.14: Energy consumption data collected through CO- and RM-based task allocation

Data Trace #	RM	CO
Data Trace-1	2259093	373412.5
Data Trace -2	1655913	353150
Data Trace -3	1054491	373050
Data Trace -4	2013722	337862.5
Data Trace -5	1500307	274012.5
Data Trace -6	190264.5	262900
Data Trace -7	917765.8	257312.5
Data Trace -8	1272169	276312.5
Data Trace -9	1969781	347050
Data Trace -10	950930	266812.5
Data Trace -11	595363.1	280537.5
Data Trace -12	228943.2	363775
Data Trace -13	842041.1	283462.5
Data Trace -14	651022.6	368162.5
Data Trace -15	1197252	310975
Data Trace -16	669060.6	367325
Data Trace -17	315580.4	323475
Data Trace -18	862806.1	369012.5
Data Trace -19	627752.1	293450
Data Trace -20	1678109	272925
Data Trace -21	2071569	346062.5
Data Trace -22	522465.5	271275
Data Trace -23	315525	307537.5
Data Trace -24	939443.6	354550
Data Trace -25	162178.8	351387.5
Data Trace -26	1038804	269937.5
Data Trace -27	1314719	270775
Data Trace -28	311803.1	302462.5
Data Trace -29	189995.1	280875
Data Trace -30	469763.5	314437.5
P-Value (Mann-Whitney U Test)	**0.000018**	
Vargha and Delaney A_{12}	**0.822222**	
Effect Size	**Large**	
Result	**CO > RM**	
Significance	**Yes**	

The rejection of the null hypothesis ensures that the proposed solution outperforms the RM. The P-value derived through Mann-Whitney U test that is measured as 0.000018. This value indicates that CO has significant differences with RM in energy consumption results. The A_{12} statistics also reveals a large effect size on energy consumption results. These two tests conclude that proposed solution has

significant statistical and practical differences from RM that results in the rejection of the null hypothesis.

5.6.3. WO vs. RM

Table 5.15 presents energy consumption data that is collected by running 30 data traces using WO- and RM-based task allocation. The Mann-Whitney U test and Vargha and Delaney statistics A_{12} are applied to measure whether the differences are significant or not. These tests help to identify the significant differences and effectiveness in 30 data traces of WO, in comparison with RM. The objective of these tests was to reject the following null hypothesis:

$H_{0\text{-energyconsumption}}$: There are no differences in energy consumption data obtained through WO when compared with RM.

The rejection of the null hypothesis ensures proposed that the solution outperforms the RM. The P-value derived through Mann-Whitney U test that is measured as 0.000012. This value indicates that WO has significant differences from RM in energy consumption results. The A_{12} statistics also reveals a large effect size on energy consumption results. These two tests conclude that proposed solution has significant statistical and practical differences from RM that results in the rejection of the null hypothesis.

Table 5.15: Energy consumption data collected through WO- and RM-based task allocation

Data Trace #	RM	WO
Data Trace-1	2259093	345456.2
Data Trace -2	1655913	326800.4
Data Trace -3	1054491	345121.6
Data Trace -4	2013722	313024.7
Data Trace -5	1500307	274624
Data Trace -6	190264.5	243876.6
Data Trace -7	917765.8	259064.7
Data Trace -8	1272169	256065.4
Data Trace -9	1969781	321409.5
Data Trace -10	950930	247152.2
Data Trace -11	595363.1	281366.8
Data Trace -12	228943.2	355850.2

Table 5.15: Continued,

Data Trace #	RM	WO
Data Trace -13	842041.1	282675.3
Data Trace -14	651022.6	359900.2
Data Trace -15	1197252	288013.6
Data Trace -16	669060.6	340028.9
Data Trace -17	315580.4	299504
Data Trace -18	862806.1	341346.7
Data Trace -19	627752.1	271644.7
Data Trace -20	1678109	252746.5
Data Trace -21	2071569	320594
Data Trace -22	522465.5	251415.4
Data Trace -23	315525	304898.4
Data Trace -24	939443.6	328044.7
Data Trace -25	162178.8	343839.7
Data Trace -26	1038804	250180.8
Data Trace -27	1314719	250713.9
Data Trace -28	311803.1	300501.6
Data Trace -29	189995.1	259989
Data Trace -30	469763.5	291257.7
P-Value (Mann-Whitney U Test):	**0.000012**	
Vargha and Delaney A_{12}	**0.828889**	
Effect Size:	**Large**	
Result	**WO > RM**	
Significance	**Yes**	

5.6.4. SW vs. RM

Table 5.16 presents energy consumption data that is collected by running thirty data traces using SW- and RM-based task allocation. The Mann-Whitney U test and Vargha and Delaney statistics A_{12} are applied to measure whether the differences are significant or not. These tests help to identify the significant differences and effectiveness in 30 data traces of SW, in comparison with RM. The objective of these tests was to reject the following null hypothesis:

$H_{0\text{-energyconsumption}}$: There are no differences in energy consumption data obtained through SW when compared with RM.

The rejection of the null hypothesis ensures that the proposed solution outperforms the RM. The P-value derived through Mann-Whitney U test that is measured as 0.000388. This value indicates that SW has significant differences from

RM in energy consumption results. The A_{12} statistics also reveals a large effect size on energy consumption results. These two tests conclude that proposed solution has significant statistical and practical differences from RM that results in the rejection of the null hypothesis.

Table 5.16: Energy consumption data collected through SW- and RM-based task allocation

Data Trace #	RM	SW
Data Trace-1	2259093	458109.3
Data Trace -2	1655913	441398.9
Data Trace -3	1054491	459550.2
Data Trace -4	2013722	436754.7
Data Trace -5	1500307	376932
Data Trace -6	190264.5	364847.4
Data Trace -7	917765.8	361103.8
Data Trace -8	1272169	380204.4
Data Trace -9	1969781	445028.6
Data Trace -10	950930	360289.1
Data Trace -11	595363.1	383100.7
Data Trace -12	228943.2	459482.8
Data Trace -13	842041.1	384094.8
Data Trace -14	651022.6	464973.3
Data Trace -15	1197252	399139
Data Trace -16	669060.6	464157.1
Data Trace -17	315580.4	424318.1
Data Trace -18	862806.1	453925.9
Data Trace -19	627752.1	383913.9
Data Trace -20	1678109	364226
Data Trace -21	2071569	443802.7
Data Trace -22	522465.5	362422.8
Data Trace -23	315525	410442.7
Data Trace -24	939443.6	438058.6
Data Trace -25	162178.8	447210.7
Data Trace -26	1038804	372469.5
Data Trace -27	1314719	363511
Data Trace -28	311803.1	402795.8
Data Trace -29	189995.1	370600.8
Data Trace -30	469763.5	405480.2
P-Value (Mann-Whitney U Test):	**0.000388**	
Vargha and Delaney A_{12}	**0.766667**	
Effect Size:	**Large**	
Result	**SW > RM**	
Significance	**Yes**	

5.6.5. SCW vs. RM

Table 5.17 presents energy consumption data that is collected by running thirty data traces using SCW- and RM-based task allocation. The Mann-Whitney U test and Vargha and Delaney statistics A_{12} are applied to measure whether the differences are significant or not. These tests help to identify the significant differences and effectiveness in 30 data traces of SCW, in comparison with RM. The objective of these tests was to reject the following null hypothesis:

$H_{0\text{-energyconsumption}}$: There are no differences in energy consumption data obtained through SCW when compared with RM.

The rejection of the null hypothesis ensures that the proposed solution outperforms the RM. The P-value derived through Mann-Whitney U test that is measured as 0.000015. This value indicates that SCW has significant differences from RM in energy consumption results. The A_{12} statistics also reveals a large effect size on energy consumption results. These two tests conclude that proposed solution has significant statistical and practical differences from RM that results in the rejection of the null hypothesis.

Table 5.17: Energy consumption data collected through CPU SCW- and RM-based task allocation

Data Trace #	RM	SCW
Data Trace-1	2259093	363984.6
Data Trace -2	1655913	343745.2
Data Trace -3	1054491	363608.7
Data Trace -4	2013722	328037.5
Data Trace -5	1500307	264221.2
Data Trace -6	190264.5	253250
Data Trace -7	917765.8	247491.3
Data Trace -8	1272169	266880.8
Data Trace -9	1969781	337640.4
Data Trace -10	950930	257478.8
Data Trace -11	595363.1	271174
Data Trace -12	228943.2	353999
Data Trace -13	842041.1	273776.9
Data Trace -14	651022.6	358419.2
Data Trace -15	1197252	301618.3
Data Trace -16	669060.6	357944.2

Table 5.17: Continued,

Data Trace #	RM	SCW
Data Trace -17	315580.4	314102.9
Data Trace -18	862806.1	359611.5
Data Trace -19	627752.1	284156.7
Data Trace -20	1678109	263576.9
Data Trace -21	2071569	336653.8
Data Trace -22	522465.5	261878.8
Data Trace -23	315525	297778.8
Data Trace -24	939443.6	345200
Data Trace -25	162178.8	341693.3
Data Trace -26	1038804	260633.7
Data Trace -27	1314719	261399
Data Trace -28	311803.1	292701.9
Data Trace -29	189995.1	271593.3
Data Trace -30	469763.5	304973.1
P-Value (Mann-Whitney U Test):	**0.000015**	
Vargha and Delaney A_{12}	**0.825556**	
Effect Size:	**Large**	
Result	**SCW > RM**	
Significance	**Yes**	

5.7 Data Collected for Comparison of Five Heterogeneity-aware Task Allocation Solutions with Random-based Task Allocation

This section presents the data collected to verify five proposed heterogeneity-aware task allocation solutions with RM in terms of execution time and energy consumption. After collecting the data, descriptive statistics, Mann-Whitney U test, and Vargha and Delaney statistics are applied. These statistics provide summaries and evidence of significant differences that help to draw some conclusions about performance evaluation.

5.7.1. Execution Time

Table 5.18 presents the execution time data collected to compare five proposed heterogeneity-aware solutions – SO, CO, WO, SW, and SCW – with RM. The row attributes of the table represent the type of solution and the column attributes represent the execution time for 30 data traces. The average execution time of five proposed solutions is less than RM for all data traces.

Table 5.18: Comparison of the execution time results obtained from proposed solutions with random-based task allocation

Data Trace #	RM	SO	CO	WO	SW	SCW
Data Trace-1	1949.305556	574.4807692	622.3541667	574.1730769	528.8846154	418.0833
Data Trace -2	1125.1875	543.3076923	588.5833333	542.9807692	496.9038462	384.8125
Data Trace -3	2916.468254	573.9230769	621.75	573.6153846	527.6153846	417.1875
Data Trace -4	465.9375	519.7884615	563.1041667	519.3269231	466.1923077	350.2292
Data Trace -5	398	421.5576923	456.6875	412.8653846	367.3846154	244.5417
Data Trace -6	1813.194444	404.4615385	438.1666667	403.9807692	351.4807692	229.0833
Data Trace -7	341.8653846	395.8653846	428.8541667	387.2307692	341.8653846	216.0625
Data Trace -8	228.6538462	425.0961538	460.5208333	424.6923077	371.1730769	256.1667
Data Trace -9	1975.396825	533.9230769	578.4166667	533.5	480.1538462	374.5417
Data Trace - 10	1005.654762	410.4807692	444.6875	410.1346154	364.6346154	242.4583
Data Trace - 11	1214.087302	431.5961538	467.5625	422.6153846	377.3076923	264.6875
Data Trace - 12	1183.4375	559.6538462	606.2916667	551.6153846	505.9038462	394.4792
Data Trace - 13	729.1666667	436.0961538	472.4375	427.6730769	382.6153846	262.5833
Data Trace - 14	460.8125	566.4038462	613.6041667	558.3653846	512.0576923	402.5
Data Trace - 15	2194.345238	478.4230769	518.2916667	478.0384615	433.3269231	315.5625
Data Trace - 16	1226.875	565.1153846	612.2083333	564.7307692	510.9423077	408.9583
Data Trace - 17	1555.357143	497.6538462	539.125	497.2884615	443.2692308	336.0625
Data Trace - 18	1918.650794	567.7115385	615.0208333	567.4230769	522.1730769	411.3333
Data Trace - 19	835.6875	451.4615385	489.0833333	451.1538462	406.0192308	287.7292
Data Trace - 20	224.1538462	419.8846154	454.875	419.5576923	374.7307692	252.3333
Data Trace - 21	1118.75	532.4038462	576.7708333	531.9423077	478.4230769	372.9167
Data Trace - 22	490.9375	417.3461538	452.125	416.9423077	372.2692308	248.5417
Data Trace - 23	901.6875	473.1346154	512.5625	464.7115385	418.0384615	301.125
Data Trace - 24	491.9423077	545.4615385	590.9166667	545.1538462	500.9230769	388.3333
Data Trace - 25	2017.956349	540.5961538	585.6458333	532.7884615	487.25	375.6042
Data Trace - 26	1136.40873	415.2884615	449.8958333	414.8846154	361.3846154	248.3125
Data Trace - 27	965.7738095	416.5769231	451.2916667	416.2692308	370.9230769	248.1458
Data Trace - 28	962.9960317	465.3269231	504.1041667	456.7884615	411.3461538	292.625
Data Trace - 29	726.7857143	432.1153846	468.125	431.8269231	387.3653846	267.0208
Data Trace - 30	931.8125	483.75	524.0625	483.3461538	437.3846154	319
Mean	1116.91	483.2961538	523.5708333	480.5205	432.9980769	317.7006944
Standard Error	121.5258	11.40366766	12.3539733	11.48988	11.43966688	12.34525821
Median	985.7143	475.7788462	515.4270833	471.375	425.6826923	308.34375
Standard Deviation	665.624	62.46046017	67.66549852	62.93265	62.65763601	67.61776402

Table 5.18: continued,

Data Trace #	RM	SO	CO	WO	SW	SCW
Sample Variance	443055.4	3901.309085	4578.61969	3960.519	3925.97935	4572.162012
Confidence Level (95.0%)	248.5481	23.32311913	25.26671239	23.49944	23.3967458	25.24888804

Table 5.19 and 5.20 present the results of the Mann-Whitney U test and Vargha and Delaney statistics (Holt *et al.*, 2014). These tests help to find the significant differences and effectiveness in 30 data traces of SO, CO, WO, SW, and SCW, in comparison with RM and each other (finding the best solution among proposed solutions). The objective of these tests was to fail the following two null hypotheses:

$H_{0\text{-execution time}}$: There are no differences in execution time measured through SO, CO, WO, SW, and SCW when comparing with RM.

$H_{0\text{-execution time}}$: There are no differences in execution time measured through SO, CO, WO, SW, and SCW when comparing with each other.

The rejection of first null hypothesis ensures proposed solutions outperform the RM. Whereas, the rejection of the second hypothesis helps to know which proposed solution is best among five. In the context of the first hypothesis, table 5.19 presents the P-value derived from the Mann-Whitney U test that is measured as 0.000058, 0.000173, 0.000051, 0.000008, and 0.000000029537. These values show that SO, CO, WO, SW, and SCW have significant differences with RM in execution time results. On the other hand, A_{12} statistics also reveals large effect size for all tests regarding execution time results. These two tests conclude that proposed solutions have significant statistical and practical differences from RM that fail the null hypothesis. Hence, it is concluded that the proposed five solutions are preferable to RM in terms of execution time.

Table 5.19: Verification of execution time results obtained from proposed solutions using Mann-Whitney U test and Vargha and Delaney statistics

Null Hypothesis (H_0)	P-Value	A_{12}	Effect Size	Results	Significance
SO = RM	0.000058	0.802222222	Large	SO > RM	Yes
CO = RM	0.000173	0.782222222	Large	CO > RM	Yes
WO = RM	0.000051	0.804444444	Large	WO > RM	Yes
SW = RM	0.000008	0.836111111	Large	SW > RM	Yes
SCW = RM	0.000000029537	0.916666667	Large	SCW > RM	Yes

In the second hypothesis, table 5.20 shows the P-values derived through Mann-Whitney U test that are measured as 0.013549, 0.700686, 0.003848, 0.0000000001537, 0.009267, 0.000029, 0.000000000028719, 0.004745, 0.00000000024879, and 0.000002. These values show that SO, CO, WO, SW, and SCW have significant differences with each other in execution time results. On the other hand, A_{12} statistics also reveals large effect size for seven tests, such as SW=SO, SCW=SO, SW=CO, SCW=CO, SW=WO, SCW=WO, and SCW=SW regarding execution time results. A medium effect is noted for CO=SO, and WO=CO. However, no effect size was found for SO=WO. Finally, our findings from both tests revealed that proposed solutions have significant statistical and practical differences from each other that fail the null hypothesis. Hence, it is concluded that SCW is the best solution and preferable to SW, CO, WO, and SO in terms of execution time.

Table 5.20: Comparison of the execution time results obtained from the proposed solutions with each other

Null Hypothesis (H_0)	P-Value	A_{12}	Effect Size	Results
CO = SO	0.013549	0.314444444	Medium	SO > CO
WO = SO	0.700686	0.528 888889	No Effect	SO = WO
SW = SO	0.003848	0.717222222	Large	SW > SO
SCW = SO	0.0000000001537	0.981111111	Large	SCW > SO
WO = CO	0.009267	0.695555556	Medium	CO > WO
SW = CO	0.000029	0.814444444	Large	SW> CO
SCW = CO	0.000000000028719	1	Large	SCW> CO
SW = WO	0.004745	0.712222222	Large	SW > WO
SCW = WO	0.00000000024879	0.975555556	Large	SCW > WO
SCW = SW	0.000002	0.857777778	Large	SCW > SW

5.7.2. Energy Consumption

Table 5.21 presents the energy consumption data collected to compare five proposed heterogeneity-aware solutions – SO, CO, WO, SW, and SCW – with RM. The row attributes of the table represent the type of solution, whereas column attributes represent the execution time for 30 data traces. The average energy consumption of five proposed solutions is less than RM for all data traces.

Table 5.21: Comparison of the energy consumption results obtained from proposed solutions with random-based task allocation

Data Trace #	RM	SO	CO	WO	SW	SCW
Data Trace-1	2259093	344688.5	373412.5	345456.2	458109.3	363984.6
Data Trace - 2	1655913	325984.6	353150	326800.4	441398.9	343745.2
Data Trace - 3	1054491	344353.8	373050	345121.6	459550.2	363608.7
Data Trace - 4	2013722	311873.1	337862.5	313024.7	436754.7	328037.5
Data Trace - 5	1500307	252934.6	274012.5	274624	376932	264221.2
Data Trace - 6	190264.5	242676.9	262900	243876.6	364847.4	253250
Data Trace - 7	917765.8	237519.2	257312.5	259064.7	361103.8	247491.3
Data Trace - 8	1272169	255057.7	276312.5	256065.4	380204.4	266880.8
Data Trace - 9	1969781	320353.8	347050	321409.5	445028.6	337640.4
Data Trace - 10	950930	246288.5	266812.5	247152.2	360289.1	257478.8
Data Trace - 11	595363.1	258957.7	280537.5	281366.8	383100.7	271174
Data Trace - 12	228943.2	335792.3	363775	355850.2	459482.8	353999
Data Trace - 13	842041.1	261657.7	283462.5	282675.3	384094.8	273776.9
Data Trace - 14	651022.6	339842.3	368162.5	359900.2	464973.3	358419.2
Data Trace - 15	1197252	287053.8	310975	288013.6	399139	301618.3
Data Trace - 16	669060.6	339069.2	367325	340028.9	464157.1	357944.2
Data Trace - 17	315580.4	298592.3	323475	299504	424318.1	314102.9
Data Trace - 18	862806.1	340626.9	369012.5	341346.7	453925.9	359611.5
Data Trace - 19	627752.1	270876.9	293450	271644.7	383913.9	284156.7
Data Trace - 20	1678109	251930.8	272925	252746.5	364226	263576.9
Data Trace - 21	2071569	319442.3	346062.5	320594	443802.7	336653.8
Data Trace - 22	522465.5	250407.7	271275	251415.4	362422.8	261878.8
Data Trace - 23	315525	283880.8	307537.5	304898.4	410442.7	297778.8
Data Trace - 24	939443.6	327276.9	354550	328044.7	438058.6	345200

Table 5.21: continued,

Data Trace #	RM	SO	CO	WO	SW	SCW
Data Trace - 25	162178.8	324357.7	351387.5	343839.7	447210.7	341693.3
Data Trace - 26	1038804	249173.1	269937.5	250180.8	372469.5	260633.7
Data Trace - 27	1314719	249946.2	270775	250713.9	363511	261399
Data Trace - 28	311803.1	279196.2	302462.5	300501.6	402795.8	292701.9
Data Trace - 29	189995.1	259269.2	280875	259989	370600.8	271593.3
Data Trace - 30	469763.5	290250	314437.5	291257.7	405480.2	304973.1
Mean	959621.0821	289977.692	314142.5	296903.578	409411.485	304640.801
Standard Error	113095.6624	6842.2006	7412.38398	6890.47626	6962.06917	7411.04547
Median	890285.9432	285467.308	309256.25	295380.861	404137.981	299698.558
Standard Deviation	619450.4548	37476.2761	40599.2991	37740.6928	38132.8233	40591.9678
Sample Variance	3.83719E+11	1404471271	1648303088	1424359894	1454112213	1647707848
Confidence Level (95.0%)	231306.6012	13993.8715	15160.0274	14092.6063	14239.0302	15157.2899

Table 5.22 presents the results obtained by conducting the Mann-Whitney U test and Vargha and Delaney statistics (Holt *et al.*, 2014). These tests help to measure the significant differences and effectiveness in 30 data traces of SO, CO, WO, SW, and SCW, in comparison of RM. The objective of these tests was to fail the following null hypothesis:

$H_{0\text{-energyconsumption}}$: The proposed solutions, SO, CO, WO, SW, and SCW consume the same amount of energy as RM.

The rejection of the null hypothesis ensures that the proposed solutions consume less energy than RM. The P-values derived through Mann-Whitney U test that are measured as 0.000012, 0.000018, 0.000012, 0.000388, and 0.000015. These values indicate that SO, CO, WO, SW, and SCW have significant differences from RM in energy consumption results. At the same time, A_{12} statistics reveals large effect size for all tests regarding energy consumption results. These two tests conclude that proposed

solutions have significant statistical and practical differences from RM that fail the null hypothesis.

Table 5.22: Verification of energy consumption results obtained from five proposed solutions using Mann-Whitney U test and Vargha and Delaney statistics

Null Hypothesis (H_0)	P-Value	A_{12}	Effect Size	Results	Significance
SO = RM	0.000012	0.828889	Large	SO > RM	Yes
CO = RM	0.000018	0.822222	Large	CO > RM	Yes
WO = RM	0.000012	0.828889	Large	WO > RM	Yes
SW = RM	0.000388	0.766667	Large	SW > RM	Yes
SCW = RM	0.000015	0.825556	Large	SCW > RM	Yes

5.8 Conclusion

In this chapter, the performance of the proposed heterogeneity-aware solutions is evaluated by implementing them in the simulated MAC environment. The benchmarking is done by evaluating the designed multi-threaded tasks on the simulated controller and compute nodes. Data are collected by sampling the evaluation parameters for 30 data traces for each proposed solution. The point estimator for each experiment is measured by applying descriptive statistics. The validation of the mathematical model with simulation results is performed through Pearson's correlation coefficient, the Mann-Whitney U test, and Vargha and Delaney's A_{12} statistics. Moreover, verification of the proposed solutions by comparing them with random-based task allocation is done through the Mann-Whitney U test and Vargha and Delaney' A_{12} statistics.

Based on the obtained results, it is concluded that proposed solutions outperform the random-based task allocation mechanism. The statistical tests validated and verified that there are significant statistical and practical differences between the results obtained from each proposed and random-based solution. Hence, the statistical tests ensure that the proposed heterogeneity-aware task allocation solutions can help significantly to improve the task performance in terms of execution time and energy consumption.

CHAPTER 6: RESULTS AND DISCUSSION

This chapter evaluates the performance of five heterogeneity-aware task allocation solutions and offers insights into the performance of the proposed algorithms by varying the parameter values. The chapter discusses the analysis of the collected results presented in chapter 5 for signifying the usefulness of the proposed five task allocation solutions. The chapter discusses the validation of the mathematical model with the simulation results. The chapter also verifies the proposed solution by comparing five proposed solutions' execution time and energy consumption results with random-based task allocation.

The rest of the chapter consists of six sections. Section 6.1 discusses the validation of the mathematical model by comparing the results of five proposed solutions execution time obtained from the mathematical model with simulation results. Section 6.2 discusses the weights identified for multiple parameters based task allocation solutions. Section 6.3 compares the individual proposed solution with random-based task allocation in terms of execution time. Section 6.4 compares the proposed solutions with random-based task allocation in terms of energy consumption. Section 6.5 presents the comparison of the proposed heterogeneity-aware task allocation solutions with random-based task allocation using the mean values. Section 6.6 concludes the chapter by highlighting the significance of the proposed solutions. The usefulness of the proposed solutions is signified by analyzing the simulation results collected in different scenarios.

6.1 Mathematical Model Validation

The correctness of the developed mathematical model is validated by comparing the results obtained from a mathematical model with that of simulation. The execution time is used as a parameter to validate the mathematical model. The statistical analyses are also applied to verify the results.

6.1.1. Execution Time of CPU Speed-based Task Allocation

Figure 6.1 shows the comparison of execution time measured through the simulation of CPU speed-based solution with the mathematical model results. The X and Y axes show five data traces and execution times (in seconds), respectively. The graph shows that the experimental results of five data traces are closer to the results obtained from the mathematical model. The differences in execution time through simulation results and mathematical model are measured at 5.06% of data trace 1, 2.87% of data trace 2, 3.57% of data trace 3, 3.06% of data trace 4, and 4.01% of data trace 5. To prove that the differences are not significant, three statistical methods, namely Pearson's correlation coefficient, Mann-Whitney U test, and Vargha and Delaney's A_{12} were applied (in table 5.3). The results of these statistical methods indicated that differences in execution time were not significant. Hence, the small differences validate the model results with those collected from the simulation.

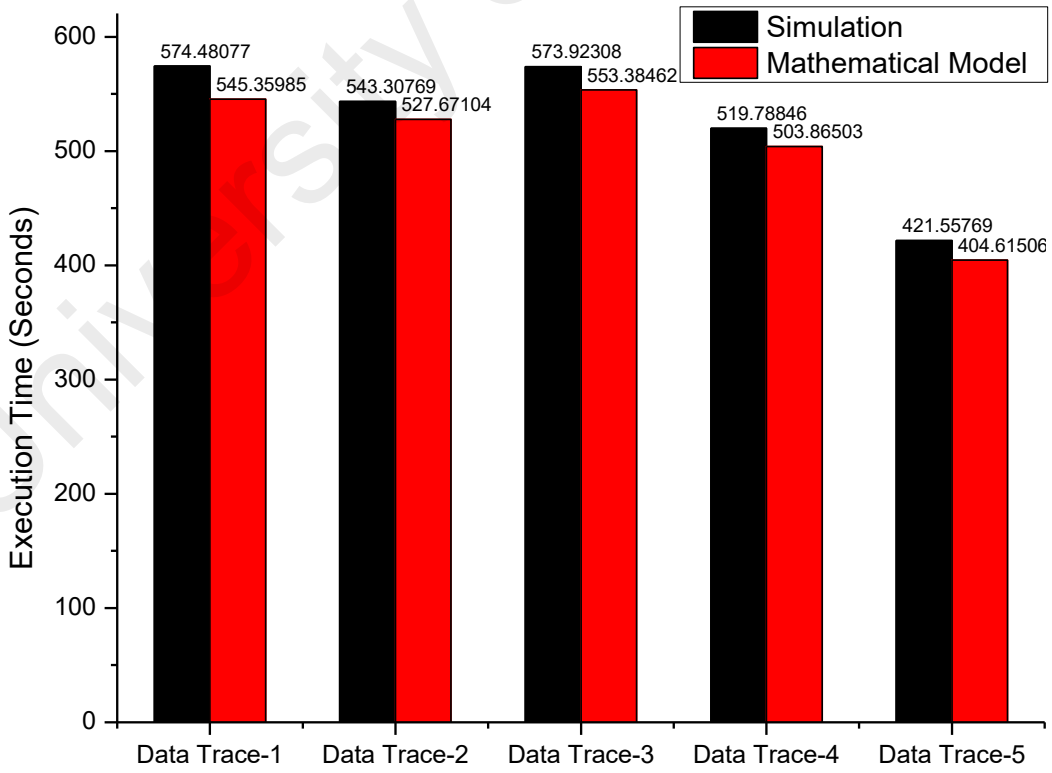

Figure 6.1: Comparison of execution time (SO) empirical results with mathematical model execution time

6.1.2. Execution Time of Core-based Task Allocation

Figure 6.2 shows the comparison of execution time measured through simulation of core-based solution with the mathematical model results. The X and Y axes show five data traces and execution times (in seconds), respectively. The graph shows that the experimental results of five data traces are closer to the results obtained from the mathematical model. The differences in execution time through simulation results and mathematical model are measured at 2.59% of data trace 1, 2.68% of data trace 2, 3.69% of data trace 3, 4.73% of data trace 4, and 6.35% of data trace 5. To prove that the differences are not significant, three statistical methods, namely Pearson's correlation coefficient, Mann-Whitney U test, and Vargha and Delaney's A_{12} were applied (in table 5.4). The results of these statistical methods indicated that differences in execution time were not significant. Hence, the small differences validate the model results with those collected from the simulation.

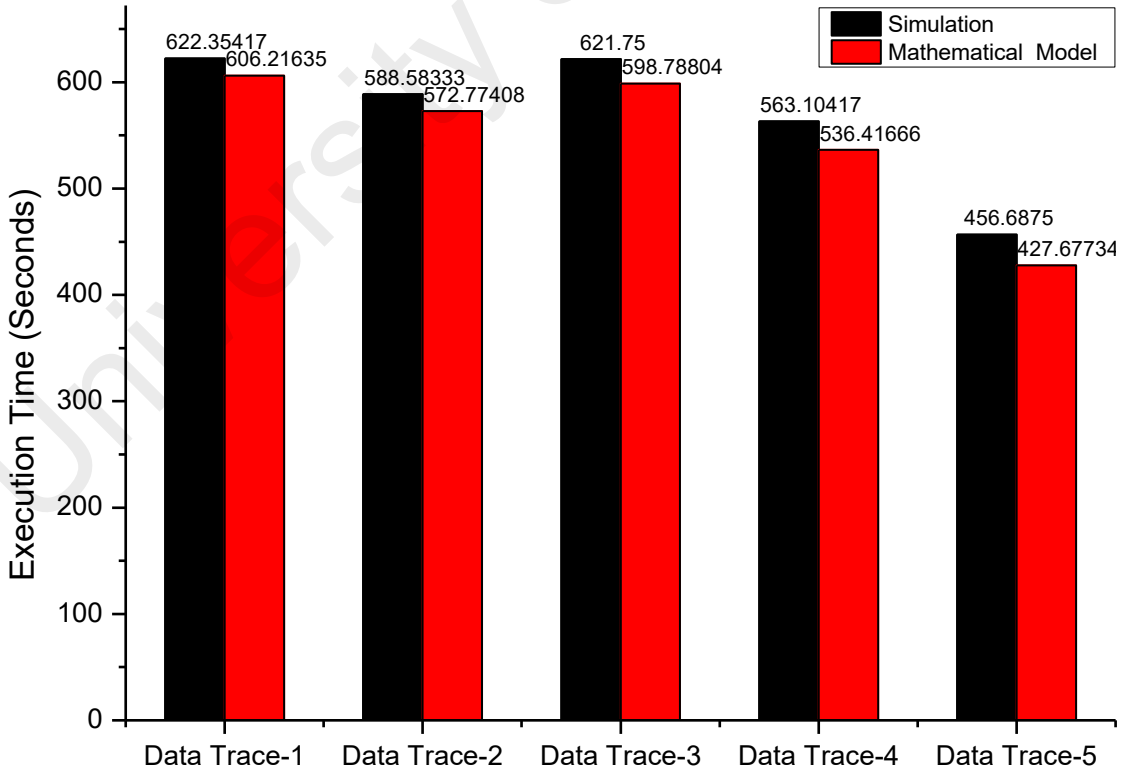

Figure 6.2: Comparison of execution time (CO) empirical results with mathematical model execution time

6.1.3. Execution Time of Workload-based Task Allocation

Figure 6.3 shows the comparison of execution time measured through simulation of workload-based solution with the mathematical model results. The X and Y axes show five data traces and execution times (in seconds), respectively. The graph shows that the experimental results of five data traces are closer to the results obtained from the mathematical model. The differences in execution time through simulation results and mathematical model are measured at 2.98% of data trace 1, 5.17% of data trace 2, 4.33% of data trace 3, 5.88% of data trace 4, and 7.09% of data trace 5. To prove that the differences are not significant, three statistical methods, namely Pearson's correlation coefficient, Mann-Whitney U test, and Vargha and Delaney's A_{12} were applied (in table 5.5). The results of these statistical methods indicated that differences in execution time were not significant. Hence, the small differences validate the model results with those collected from the simulation.

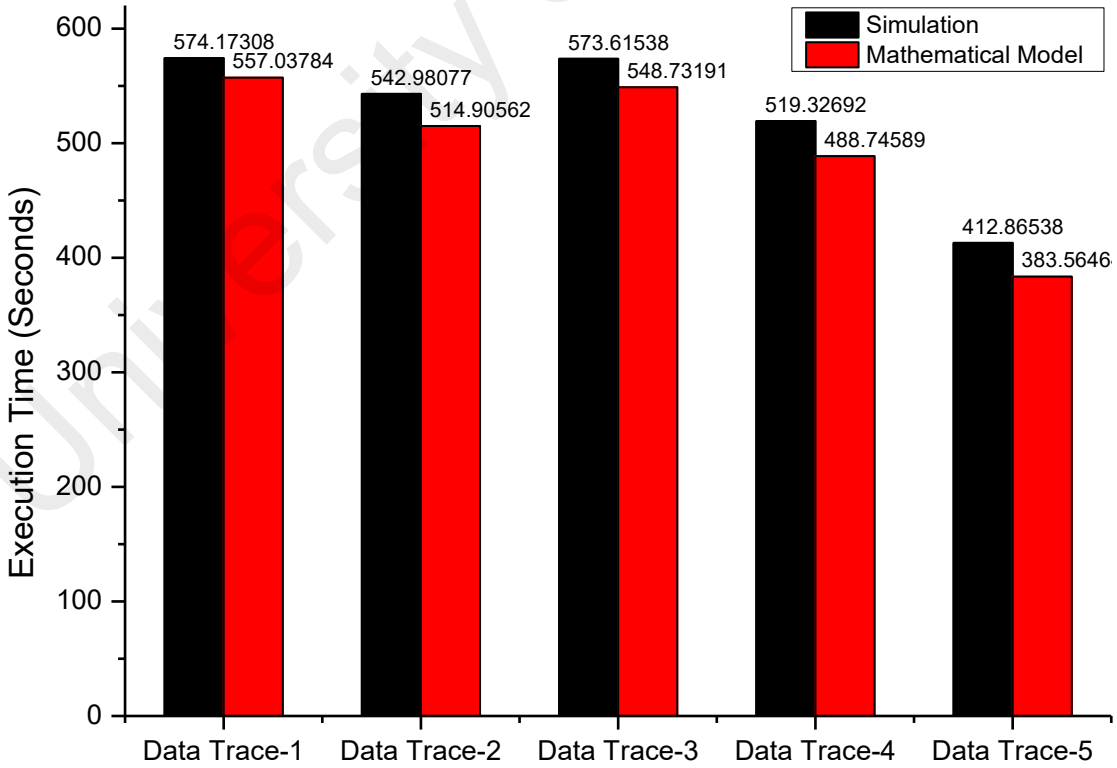

Figure 6.3: Comparison of execution time (WO) empirical results with mathematical model execution time

6.1.4. Execution Time of Two parameters-based (CPU Speed and Workload) Task Allocation

Figure 6.4 shows the comparison of execution time measured through simulation of two parameters (CPU speed and workload) based solution with the mathematical model results. The X and Y axes show five data traces and execution times (in seconds), respectively. The graph shows that the experimental results of five data traces are closer to the results obtained from the mathematical model. The differences in execution time through simulation results and mathematical model are measured at 6.69% of data trace 1, 5.28% of data trace 2, 9.01% of data trace 3, 10.42% of data trace 4, and 10.14% of data trace 5. To prove that the differences are not significant, three statistical methods, namely Pearson's correlation coefficient, Mann-Whitney U test, and Vargha and Delaney's A_{12} were applied (in table 5.6). The results of these statistical methods indicated that differences in execution time were not significant. Hence, the small differences validate the model results with those collected from the simulation.

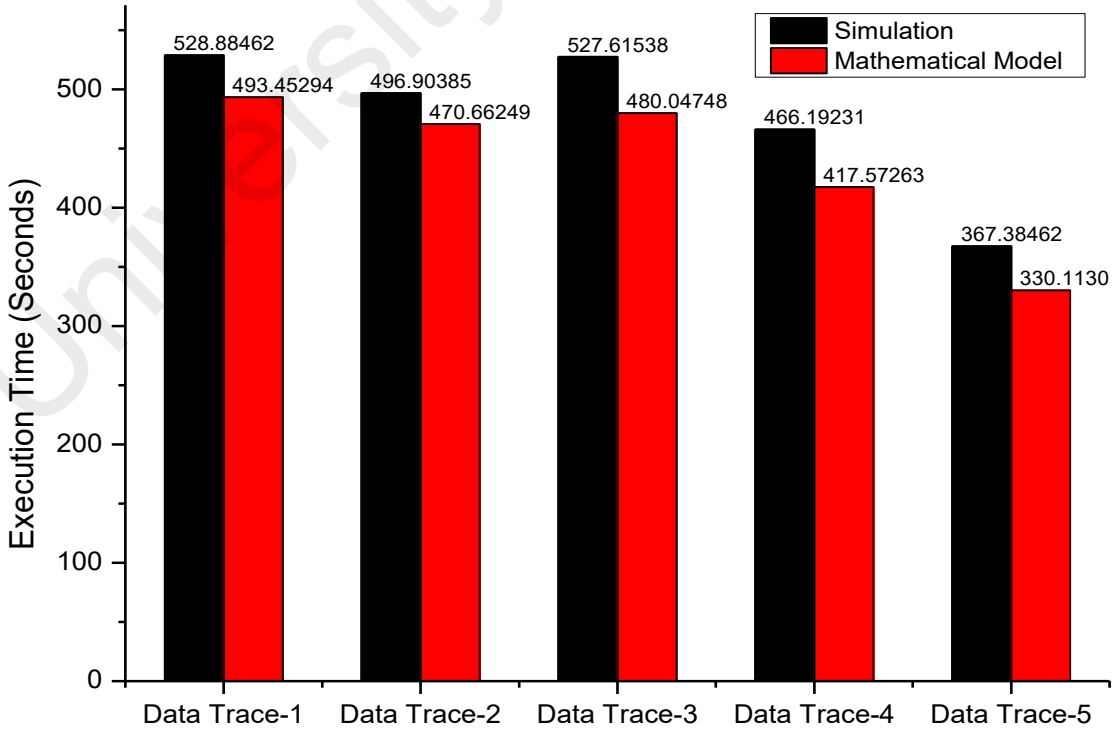

Figure 6.4: Comparison of execution time (SW) empirical results with mathematical model execution time

6.1.5. Execution Time of Three Parameters-based (CPU Speed, Core, and Workload) Task Allocation

Figure 6.5 shows the comparison of execution time measured through simulation of three parameters (CPU speed, core, and workload) based solution with the mathematical model results. The X and Y axes show five data traces and execution times (in seconds), respectively. The graph shows that the experimental results of five data traces are closer to the results obtained from the mathematical model. The differences in execution time through simulation results and mathematical model are measured at 3.85% of data trace 1, 4.10% of data trace 2, 5.50% of data trace 3, 7.62% of data trace 4, and 11.86% of data trace 5. To prove that the differences are not significant, three statistical methods, namely Pearson's correlation coefficient, Mann-Whitney U test, and Vargha and Delaney's A_{12} were applied (in table 5.7). The results of these statistical methods indicated that differences in execution time were not significant. Hence, the small differences validate the model results with those collected from the simulation.

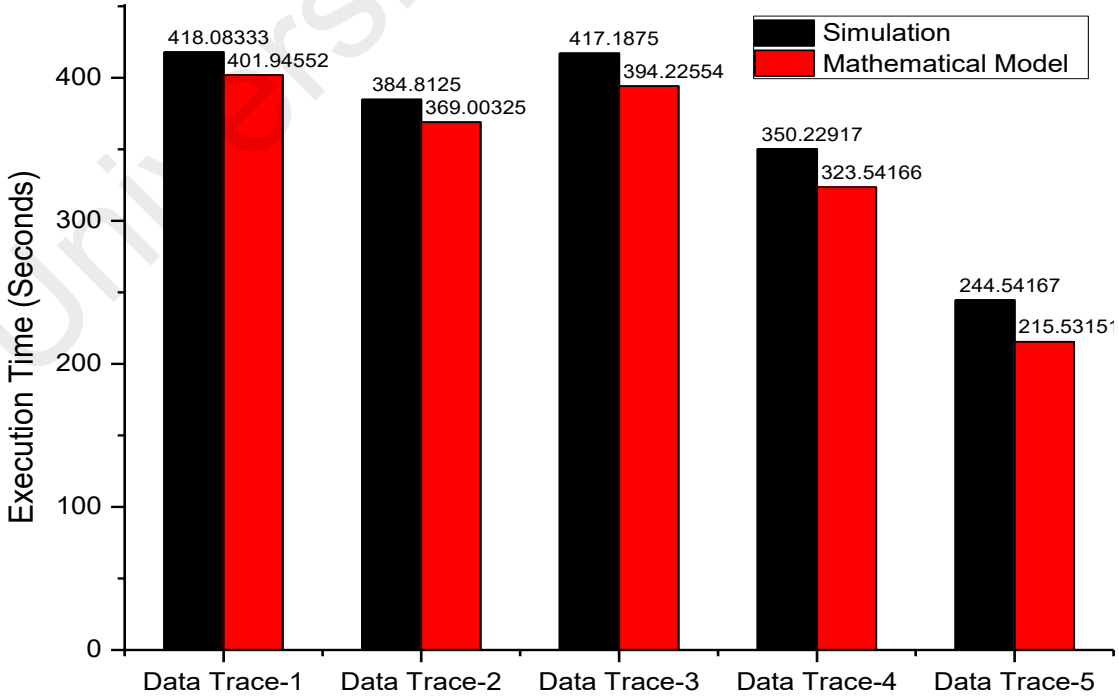

Figure 6.5: Comparison of execution time (SCW) empirical results with mathematical model execution time

6.2 Impact of Various Weights on Execution Time

Figures 6.6 and 6.7 indicate execution time against numerous combinations of weights that are identified to know the significance of each parameter used in two and three parameters-based task allocation solutions. The rationale for identifying the weights is that it can help to minimize the execution time of the compute-intensive tasks through knowing the significance of each parameter on which task allocation decision can be made in MAC. In this study, the proposed solutions presented in algorithms 4 and 5, perform task allocation based on two and three parameters, such as CPU speed and workload, and CPU speed, core, and workload, respectively. The significance of each parameter is measured by assigning different weights to each parameter and then analyzing their impact on task execution.

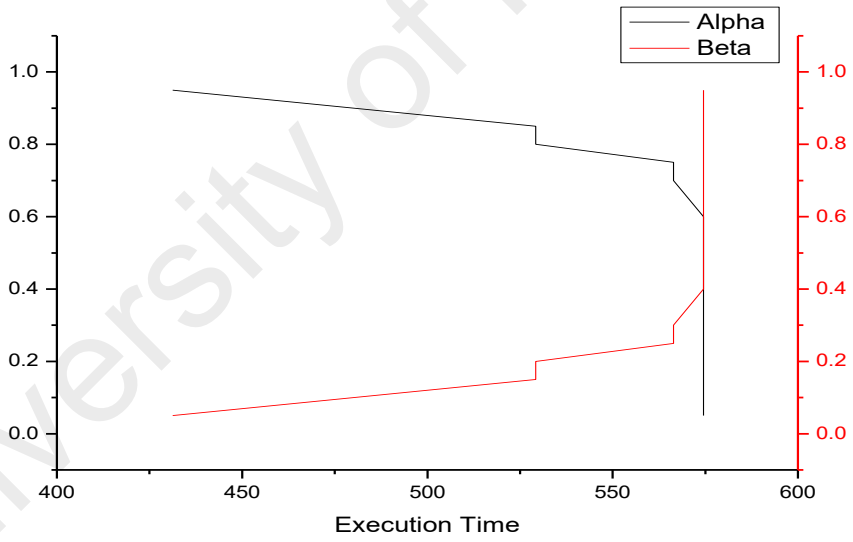

Figure 6.6: Impact of combinations of two parameters' weights on execution time

The best weight combination is selected based on the shortest execution time. In the proposed solution that is based on two parameters, a shorter execution time was found on this combination (Alpha 0.95 and Beta 0.05), where Alpha is weight of workload and Beta is CPU speed. Whereas, in three parameters-based solution, a shorter execution time was found on this combination (Alpha 0.65, Beta 0.15 and Gamma 0.20), where Alpha is weight of workload, Beta is CPU speed, and Gamma is core.

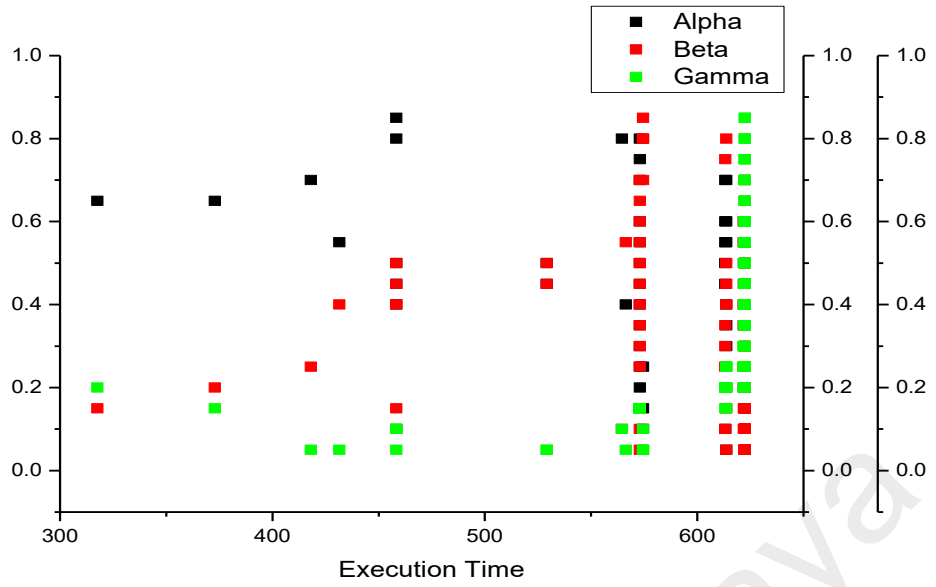

Figure 6.7: Impact of combinations of three parameters' weights on execution time

6.3 Comparison of Proposed Heterogeneity-aware Task Allocation Solutions based on Execution Time

In this section, we compare the execution time results of 30 data traces obtained from proposed heterogeneity-aware task allocation solutions with random-based task allocation. The x and y axes used in the graphs represent the data traces and execution time, respectively. The execution time is measured in seconds. Moreover, SO, CO, WO, SW, and SCW are used to represent the proposed solutions that are based on CPU speed, number of cores, CPU speed and workload, and CPU speed, number of cores and workload, respectively. RM and DT are used to represent a random-based task allocation solution and data trace, respectively. Furthermore, we verify the execution time results obtained from the proposed solutions and random-based task allocation solution using two statistical methods Mann-Whitney U test and Vargha and Delaney's A12 statistics.

6.3.1. SO vs. RM

Figure 6.8 shows the execution time of 30 data traces, measured by running the SO- and RM-based task allocation. The results indicate a significant reduction in execution time compared to the random-based solution in most of the cases. The reason

for the reduction in execution time is that through proposed solution the task assignment decision is made based on CPU speed. In this context, the tasks are sorted in ascending order and then assigned to the compute nodes on the basis of the CPU speed. Thus, the allocation of the larger task on high-speed mobile device helps to minimize the execution time; however, in random-based solution, most of the time this task is assigned to a device that has low processing capabilities. In the simulation scenario, this was found to cause inefficient resource utilization. The inefficient resource utilization can degrade tasks performance by delaying the execution time. In addition, inefficient resource utilization can increase energy consumption in the MAC environment.

6.3.1.1. Statistical Analyses (SO vs. RM)

We applied Mann-Whitney U test and Vargha and Delaney's A12 statistics to measure whether or not the differences between the execution time results obtained from SO and RM were significant (in table 5.8). The P-value derived through Mann-Whitney U test was measured 0.000058. This value indicated that SO has significant statistical differences with RM in execution time results (as the value of P found less than 0.05). At the same time, the A12 statistics also revealed a large effect size on execution time results. These two tests concluded that SO has significant statistical and practical differences from RM.

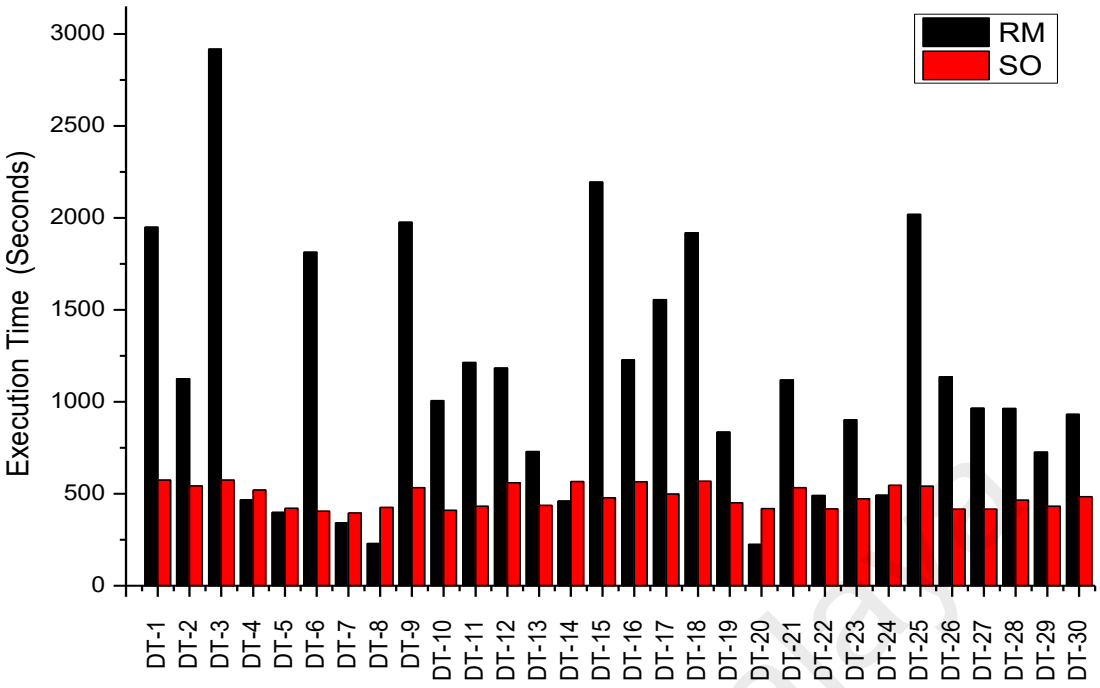

Figure 6.8: Execution time empirical results measured using SO-based task allocation

6.3.2. CO vs. RM

Figure 6.9 depicts the execution time of 30 data traces obtained from CO- and RM- based task allocation. The results indicate that the proposed solution helps to execute most of the data traces in a fast manner as their execution time was found much lower compared to the random-based mechanism. In random-based task allocation, the controller node assigns tasks to the compute nodes without considering the specification of the compute nodes as noticed in the simulation scenario. Therefore, random-based task allocation results in delaying the execution time in most of the data traces execution. Although using core-based solution execution time can be minimized, further reduction in execution time is possible if other parameters, such as workload and CPU speed are incorporated. In addition, multiple parameters-based task allocation solutions can help to obtain an optimal reduction in execution time.

6.3.2.1. Statistical Analyses (CO vs. RM)

We applied Mann-Whitney U test and Vargha and Delaney's A12 statistics to measure whether or not the differences between the execution time results obtained from CO and RM were significant (in table 5.9). The P-value derived through Mann-

Whitney U test was measured 0.000173. This value indicated that CO has significant statistical differences with RM in execution time results (as the value of P found less than 0.05). At the same time, the A12 statistics also revealed a large effect size on execution time results. These two tests concluded that CO has significant statistical and practical differences from RM.

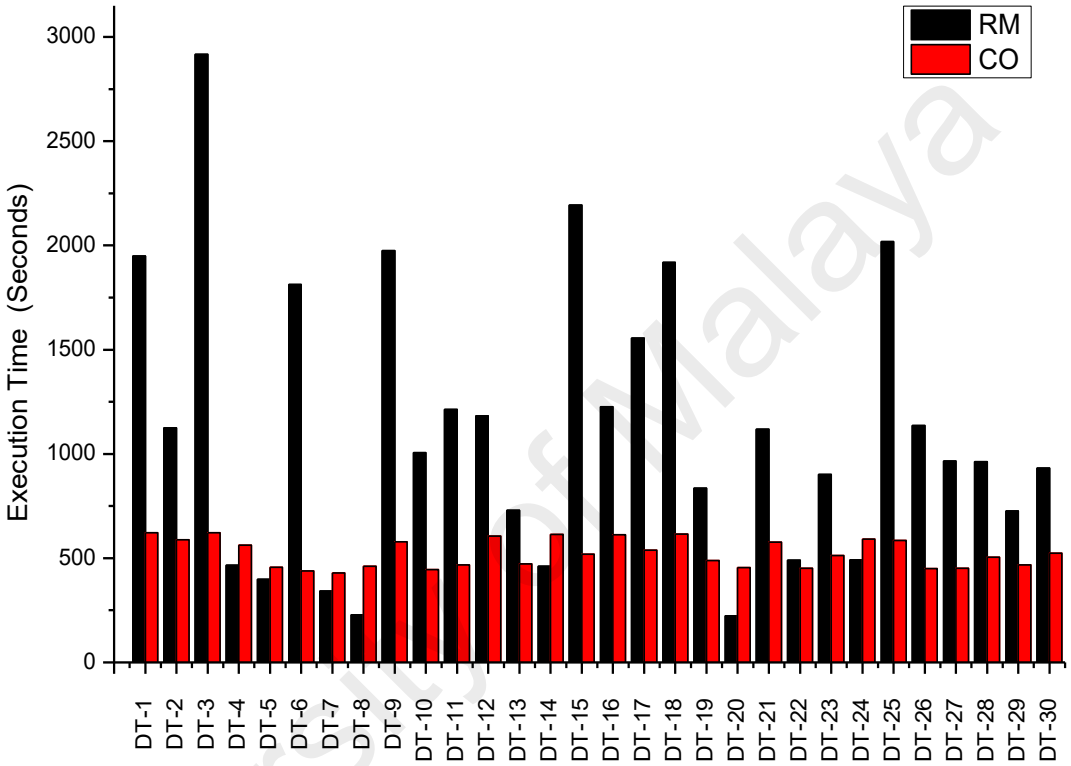

Figure 6.9: Execution time empirical results measured using CO-based task allocation

6.3.3. WO vs. RM

Figure 6.10 presents the execution time of 30 data traces that is measured through WO- and RM-based task allocation. The results reveal that workload-based task allocation helps to execute most of the data traces in a fast manner compared to the random-based mechanism. In our simulation scenario, it has been noticed that in random-based task allocation, the controller node assigns tasks to such compute nodes that have lower processing capabilities. Therefore, random-based task allocation prolongs the execution time. However, through the workload-based task allocation such compute nodes were being selected in the simulation scenario that have high specification as they have a lighter workload running in the background. Although the

use of workload-based solution can minimize execution time, further reduction in execution time is also possible if task allocation decision is based on two or three parameters instead of single.

6.3.3.1. Statistical Analyses (WO vs. RM)

We applied Mann-Whitney U test and Vargha and Delaney's A12 statistics to measure whether or not the differences between the execution time results obtained from WO and RM were significant (in table 5.10). The P-value derived through Mann-Whitney U test was measured 0.000051. This value indicated that WO has significant statistical differences with RM in execution time results (as the value of P found less than 0.05). At the same time, the A12 statistics also revealed a large effect size on execution time results. These two tests concluded that WO has significant statistical and practical differences from RM.

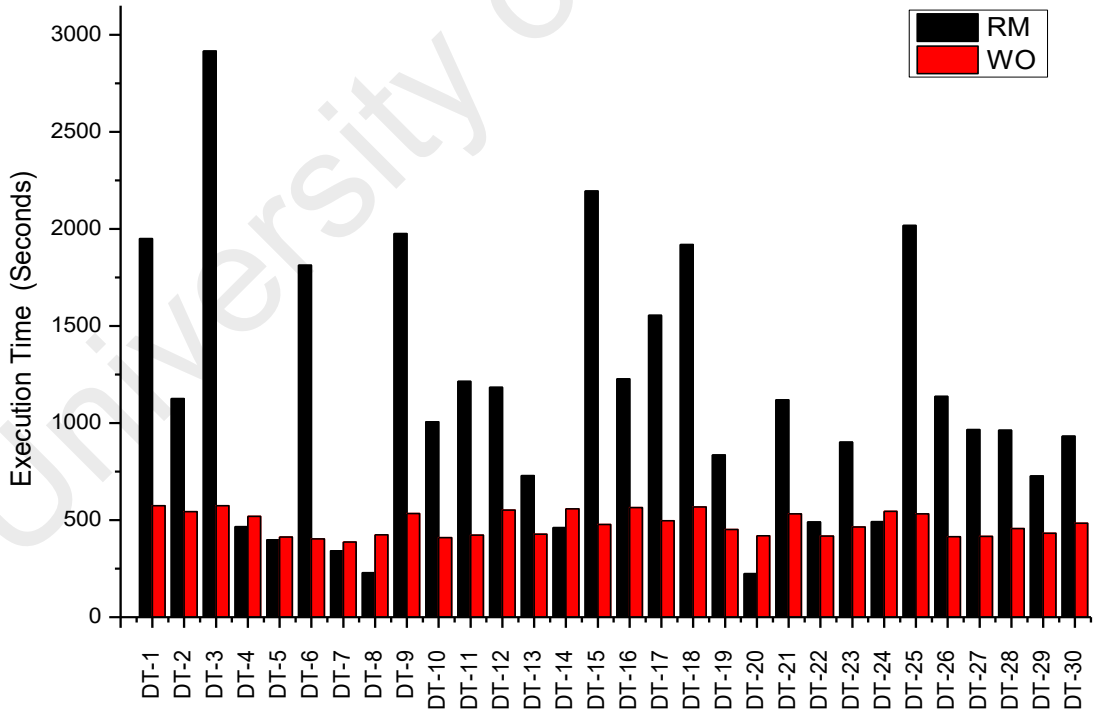

Figure 6.10: Execution time empirical results measured using WO-based task allocation

6.3.4. SW vs. RM

Figure 6.11 shows the execution time of the 30 data traces that is obtained through SW- and RM-based task allocation. The results reveal that the execution time

measured through two parameters-based task allocation is much lower compared to a random-based solution in most of the cases. The simulation scenario reveals that through random-based task allocation, most of the time the tasks were allocated to slower devices for the execution. Thus, the tasks took longer to complete. However, the proposed solution ensures minimization in task execution time due to the involvement two parameters (CPU speed and workload) when the task allocation decision was made. In our simulation scenario, through the proposed solution devices that have high specifications which ensure minimization in execution time are selected. Although performing task allocation based on two parameters helps to minimize the execution time compared to single parameter-based task allocation, further reduction in execution time is also possible if task allocation decision is based on three parameters.

6.3.4.1. Statistical Analyses (SW vs. RM)

We applied Mann-Whitney U test and Vargha and Delaney's A12 statistics to measure whether or not the differences between the execution time results obtained from SW and RM were significant (in table 5.11). The P-value derived through Mann-Whitney U test was measured 0.000008. This value indicated that SW has significant statistical differences with RM in execution time results (as the value of P found less than 0.05). At the same time, the A12 statistics also revealed a large effect size on execution time results. These two tests concluded that SW has significant statistical and practical differences from RM.

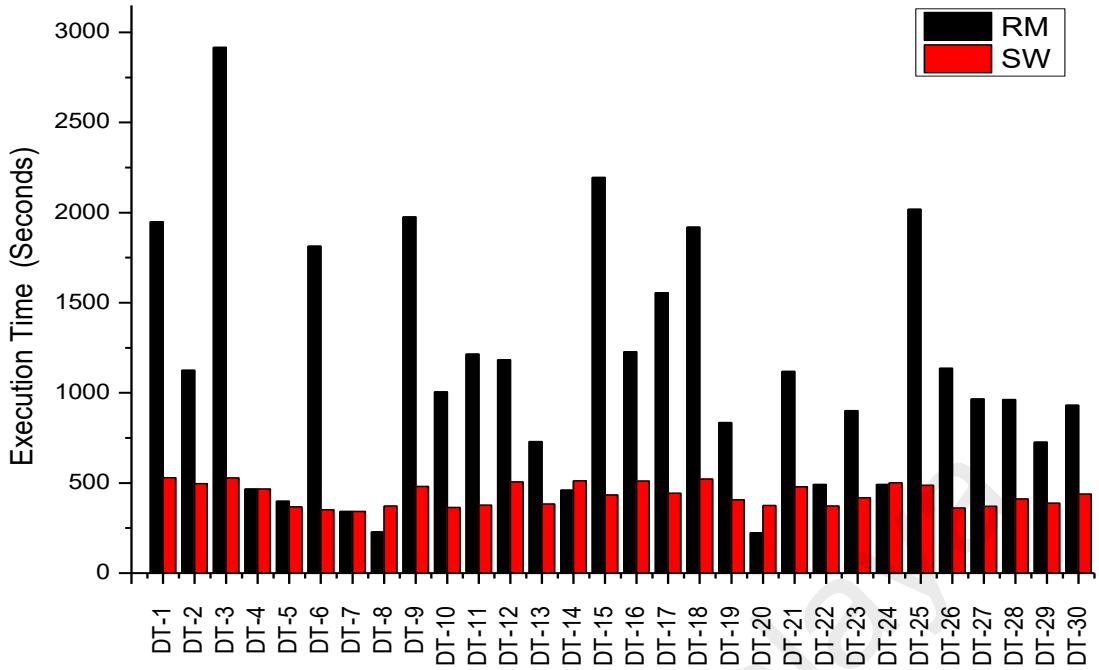

Figure 6.11: Execution time empirical results measured using SW- and RM-based task allocation

6.3.5. SCW vs. RM

Figure 6.12 illustrates the execution time of 30 data traces obtained through SCW- and RM-based task allocation. The results show that the proposed solution executes 30 data traces, in most of the time in less time compared to random-based task allocation. The reason is that in the simulation scenario, most of the time larger tasks were almost always allocated to devices that have lower processing capabilities. However, through three parameters-based task allocation, compute nodes that have high processing capabilities were selected in our simulation scenario. Therefore, the proposed solution outperforms the random-based task allocation in most of the data traces execution in terms of execution time. Thus, it is concluded that performing task allocation based on CPU speed, core, and workload running in the background can lead to efficient resource utilization in MAC. This efficient resource utilization minimizes the execution time of the tasks more than the random and other four proposed solutions.

6.3.5.1. Statistical Analyses (SCW vs. RM)

We applied Mann-Whitney U test and Vargha and Delaney's A12 statistics to measure whether or not the differences between the execution time results obtained

from SCW and RM were significant (in table 5.12). The P-value derived through Mann-Whitney U test was measured 0.000000029537. This value indicated that SCW has significant statistical differences with RM in execution time results (as the value of P found less than 0.05). At the same time, the A12 statistics also revealed a large effect size on execution time results. These two tests concluded that SCW has significant statistical and practical differences from RM.

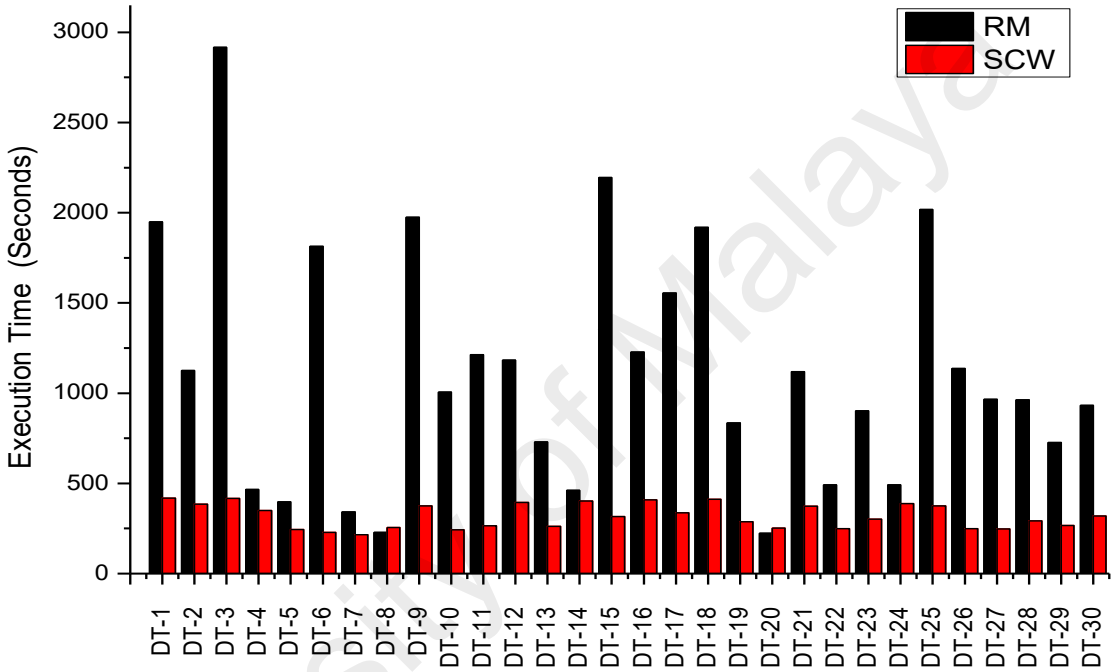

Figure 6.12: Execution time results measured using SCW- and RM-based task allocation

6.4 Comparison of Proposed Heterogeneity-aware Task Allocation Solutions based on Energy Consumption

This section discusses the proposed heterogeneity-aware task allocation solutions in terms of energy consumption, in comparison with random-based task allocation. The x and y axes used in the graphs represent the data traces and energy consumption, respectively. The energy consumption is measured in mJ. Furthermore, we verify the energy consumption results obtained from the proposed solutions and random-based task allocation solution using two statistical methods Mann-Whitney U test and Vargha and Delaney's A12 statistics.

6.4.1. SO vs. RM

Figure 6.13 presents the results of 30 data traces obtained from SO- and RM-based task allocation. The results reveal that random-based task allocation consumes more energy compared to CPU speed-based task allocation in most of the cases. The reason is that the random-based task allocation performs task assignment to slower devices in most of the cases as it has been noticed in the simulation scenario. The allocation of tasks to slower devices leads to energy wastage. However, CPU speed-based task allocation enables the controller node to assign all the tasks only to such devices that have high CPU speed which helps to execute a task by consuming less energy.

6.4.1.1. Statistical Analyses (SO vs. RM)

We applied Mann-Whitney U test and Vargha and Delaney's A12 statistics to measure whether or not the differences between the energy consumption results obtained from SO and RM were significant (in table 5.13). The P-value derived through Mann-Whitney U test was measured 0.000012. This value indicated that SO has significant statistical differences with RM in energy consumption results (as the value of P found less than 0.05). At the same time, the A12 statistics also revealed a large effect size on energy consumption. These two tests concluded that SO has significant statistical and practical differences from RM.

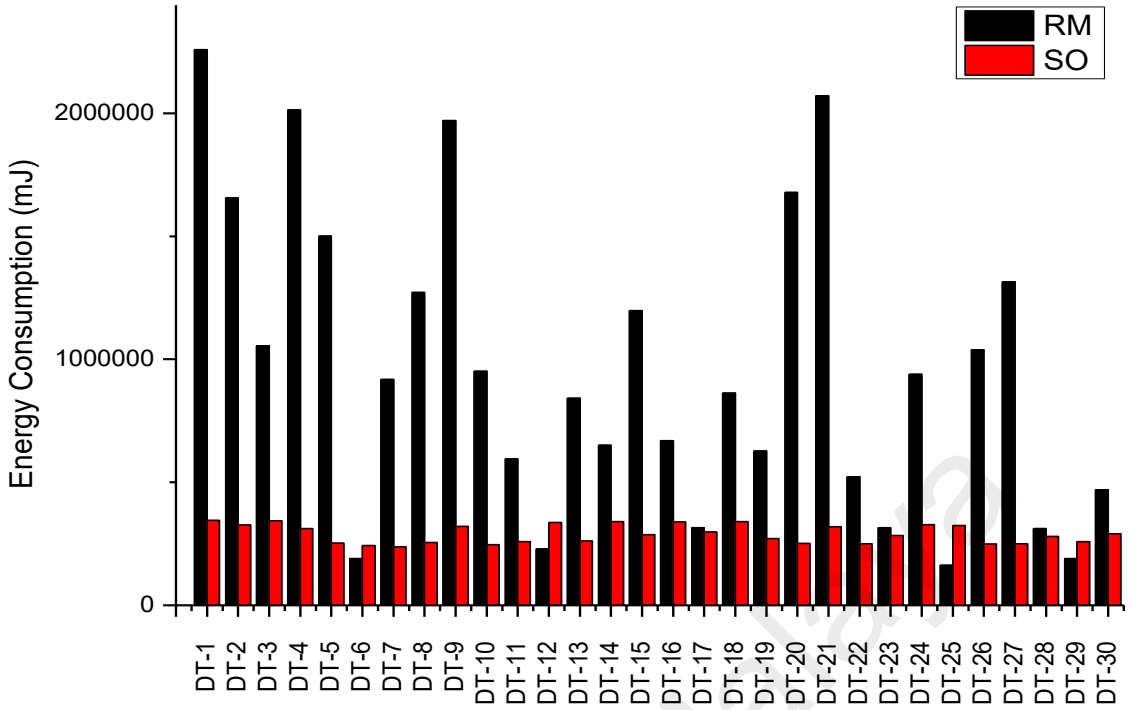

Figure 6.13: Energy consumption results measured using SO- and RM-based task allocation

6.4.2. CO vs. RM

Figure 6.14 presents the results of 30 data traces that are obtained from the CO- and RM- based task allocation. The results reveal that random-based task allocation most of the time consumes more energy due to performing allocation of tasks to slower devices, in most of the cases as it has been noticed in the simulation scenario. However, core-based task allocation enables the controller node to select only the devices for task execution that have the most cores. Thus, core-based task allocation helps to select such devices for computation that have high processing capabilities. In this way, this solution not only minimizes the execution time but also saves energy compared to random-based task allocation.

6.4.2.1. Statistical Analyses (CO vs. RM)

We applied Mann-Whitney U test and Vargha and Delaney's A12 statistics to measure whether or not the differences between the energy consumption results obtained from CO and RM were significant (in table 5.14). The P-value derived through Mann-Whitney U test was measured 0.000018. This value indicated that CO has significant statistical differences with RM in energy consumption results (as the value of

P found less than 0.05). At the same time, the A12 statistics also revealed a large effect size on energy consumption. These two tests concluded that CO has significant statistical and practical differences from RM.

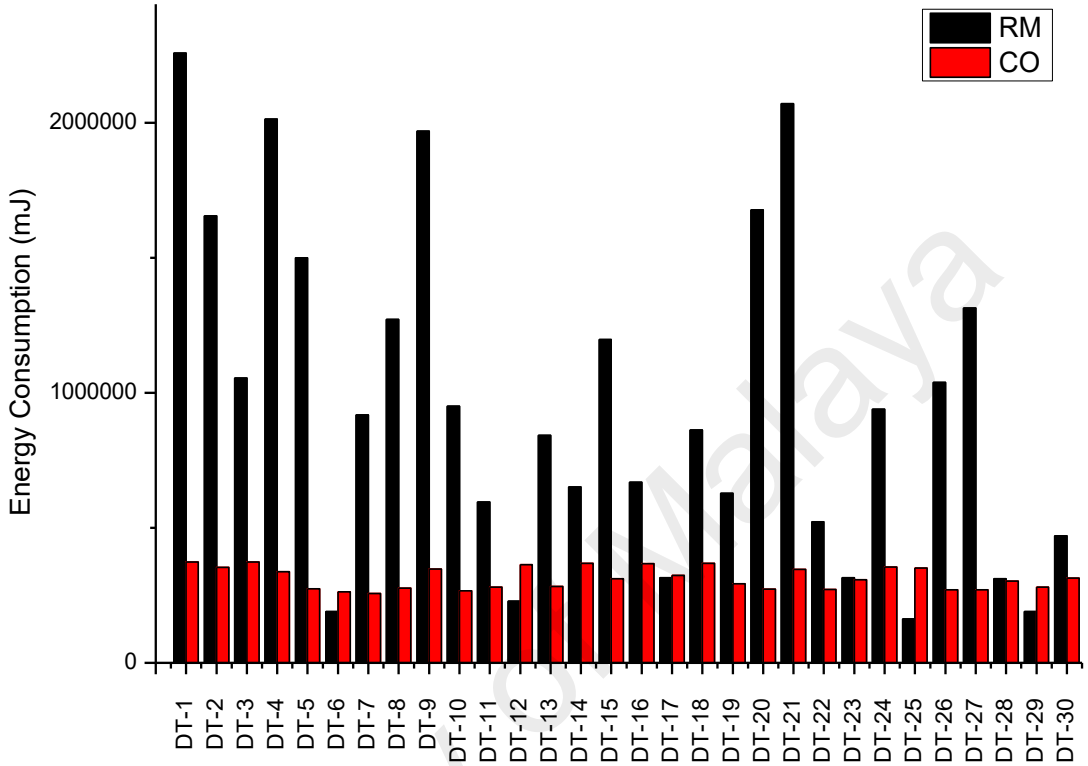

Figure 6.14: Energy consumption results measured using CO- and RM-based task allocation

6.4.3. WO vs. RM

Figure 6.15 illustrates the results of energy consumption obtained from WO- and RM- based task allocation. The results clearly show that random-based task allocation most of the time usually consumes more energy for executing different data traces. However, workload-based task allocation consumes less. The reason is that, particularly in our simulation scenario, the devices selected based on workload criteria have high processing capabilities. Therefore, this solution consumes less energy than random-based task allocation.

6.4.3.1. Statistical Analyses (WO vs. RM)

We applied Mann-Whitney U test and Vargha and Delaney's A12 statistics to measure whether or not the differences between the energy consumption results obtained from WO and RM were significant (in table 5.15). The P-value derived

through Mann-Whitney U test was measured 0.000012. This value indicated that WO has significant statistical differences with RM in energy consumption results (as the value of P found less than 0.05). At the same time, the A12 statistics also revealed a large effect size on energy consumption. These two tests concluded that WO has significant statistical and practical differences from RM.

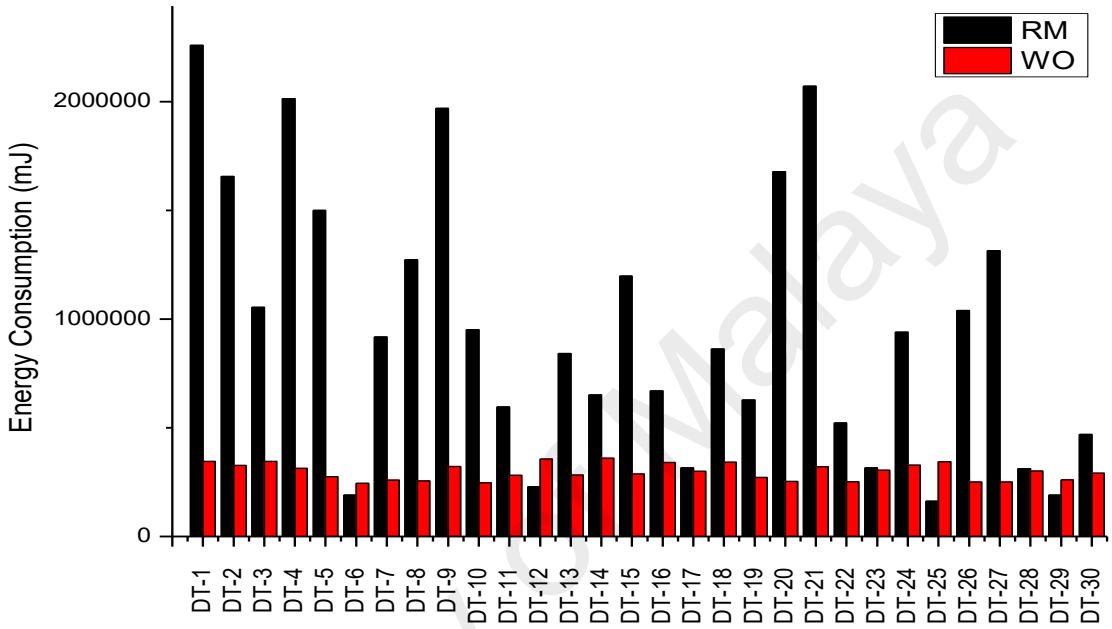

Figure 6.15: Energy consumption results measured using WO- and RM-based task allocation

6.4.4. SW vs. RM

Figure 6.16 presents the results obtained from SW- and RM-based task allocation. The results show that the proposed solution outperforms the random-based task allocation. The graph shows that most of the data traces are consuming more energy through random-based task allocation in most of the cases. The reason is that in our simulation scenario, it was observed that through random-based task allocation, controller node most of the time assigns larger tasks in terms of computational lengths to slower devices that lead to the consumption of extra energy. However, the compute nodes selected through the proposed solution had high processing capabilities. Therefore, the proposed solution consumes less energy than random-based task allocation.

6.4.4.1. Statistical Analyses (SW vs. RM)

We applied Mann-Whitney U test and Vargha and Delaney's A12 statistics to measure whether or not the differences between the energy consumption results obtained from SW and RM were significant (in table 5.16). The P-value derived through Mann-Whitney U test was measured 0.000388. This value indicated that SW has significant statistical differences with RM in energy consumption results (as the value of P found less than 0.05). At the same time, the A12 statistics also revealed a large effect size on energy consumption. These two tests concluded that SW has significant statistical and practical differences from RM.

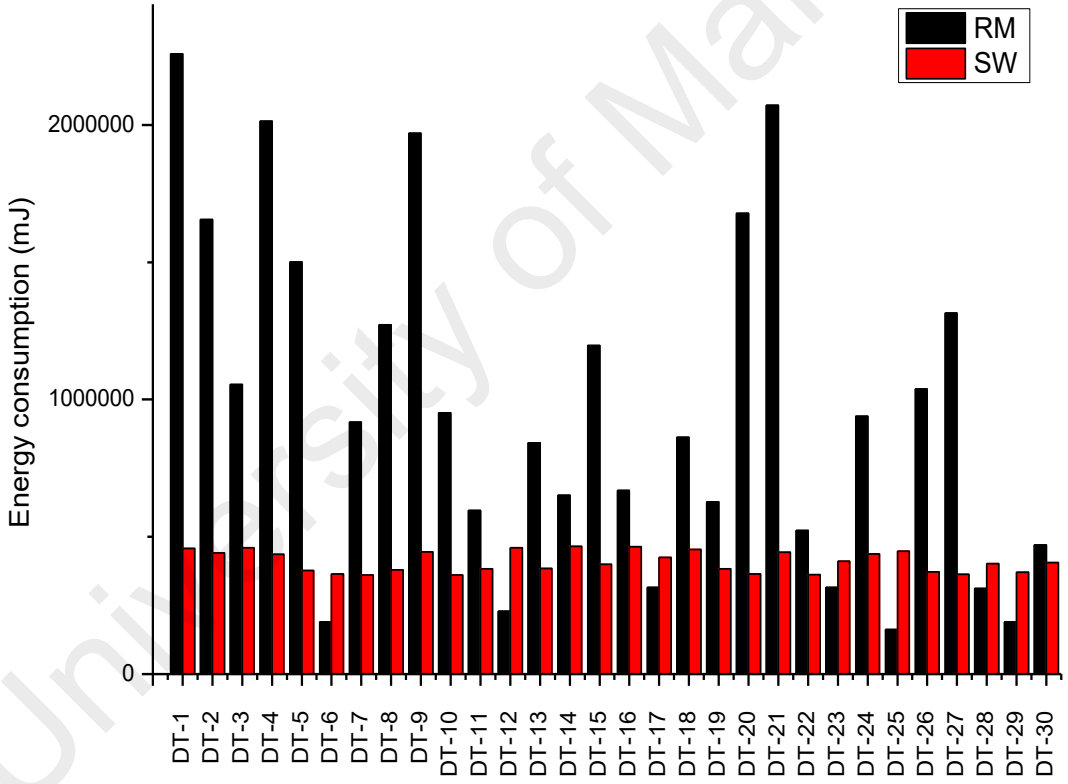

Figure 6.16: Energy consumption results measured using SW- and RM-based task allocation

6.4.5. SCW vs. RM

Figure 6.17 presents the results obtained from SCW- and RM-based task allocation. The results reveal that random-based task allocation results in consuming more energy than the proposed solution, in most of the cases. The reason is that through random-based task allocation tasks are being assigned to the slower devices as noted in

the simulation scenario. However, the proposed solution ensures that tasks are allocated to devices that have high CPU speed, core, and less workload. Based on this defined task allocation criteria, the devices that have high specifications were being selected for task execution in our simulation scenario. Thus, such selection helps to ensure minimum energy consumption than random-based task allocation.

6.4.5.1. Statistical Analyses (SCW vs. RM)

We applied Mann-Whitney U test and Vargha and Delaney's A_{12} statistics to measure whether or not the differences between the energy consumption results obtained from SCW and RM were significant (in table 5.17). The P-value derived through Mann-Whitney U test was measured as 0.000015. This value indicated that SCW has significant statistical differences with RM in energy consumption results (as the value of P found less than 0.05). At the same time, the A_{12} statistics also revealed a large effect size on energy consumption. These two tests concluded that SCW has significant statistical and practical differences from RM.

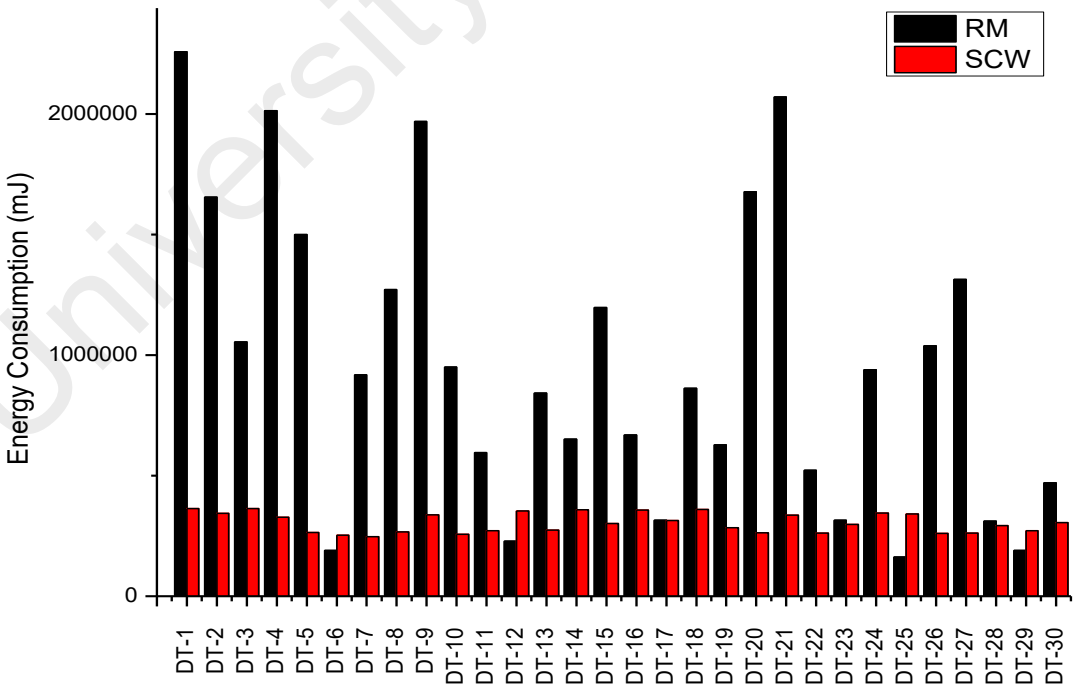

Figure 6.17: Energy consumption results measured using SCW- and RM-based task allocation

6.5 Overall Comparison of Proposed Heterogeneity-aware Task Allocation Solutions with Random-based Task Allocation

In this section, we provide the comparison of the proposed solutions with ransom based task allocation in terms of execution time and energy consumption. The comparison is based on means values presented in chapter 5 (tables 5.18 and 5.21). Moreover, the Mann-Whitney U Test and Vargha and Delaney's A_{12} statistics are applied to verify the proposed solutions' performance with random-based task allocation.

6.5.1. Execution Time

Figure 6.18 presents the execution time of five tasks. The x and y axes represent different types of proposed algorithms and execution time, respectively. The execution time is measured in seconds. The comparison results reveal that proposed solutions outperform the RM based task allocation.

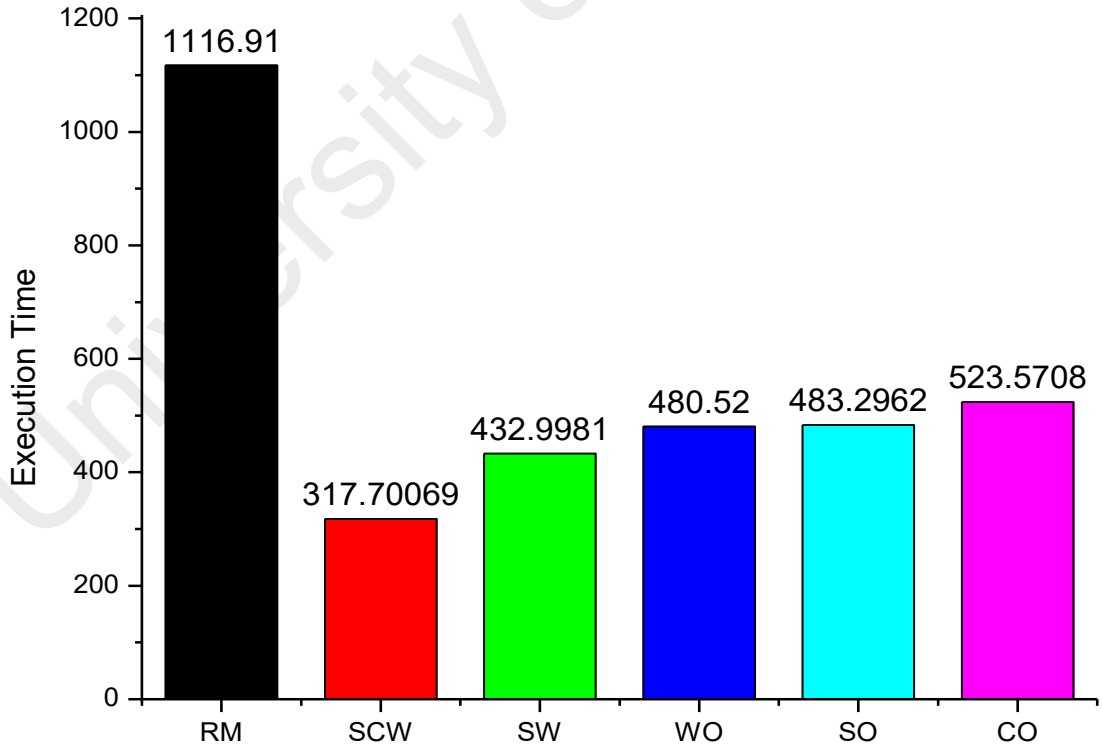

Figure 6.18: Comparison of execution time empirical results obtained from five proposed solutions with random-based task allocation

In RM-based task allocation, tasks of various sizes are assigned to the available mobile devices without considering resource and operational heterogeneity in MAC paradigm. This causes inefficient resource utilization that can prolong the execution time of the task. In this context, five task allocation solutions are proposed. One of the solutions proposed to reduce the task execution time is to base the decision on CPU speed while performing task allocation. The execution of the tasks that are performed based on high CPU speed mobile devices helps to shorten the task execution time than low CPU speed mobile devices that can be selected by random-based task allocation. Thus, results of the solution based on high CPU speed show that it can shorten the task execution time by as much as 56.72% compared to random-based task allocation.

The second possible solution is based on the number of cores. First, it needs to be clear that core-based solution is different than CPU speed-based task allocation. In the experiment, the devices were selected so that high-speed CPU devices do not have a higher number of cores, otherwise the result of the high CPU speed and the core-based proposed solution could remain same. Thus, the results of core-based proposed solution indicate that performing task allocation based on the higher number of cores help to minimize the execution time up to 53.12% compared to random-based task allocation. The reason for the shorter execution time is that the tasks that required more computing power were being executed on resource-rich mobile devices with a higher number of cores. The device with a higher number of cores ensures parallel processing of multi-threaded tasks.

The third proposed solution is based on workload parameter. In MAC, the devices may have a different workload running in the background. Therefore, basing task allocation only on high CPU speed and a larger number of cores may not be useful when the workload on high-speed mobile devices is very high. In such cases, basing task allocation on high CPU speed or core can degrade the performance of the tasks

compared to devices that have low CPU speed and fewer cores; however, the background workload on the devices is very low. In this context, workload-based task allocation can minimize the task execution time. Thus, the workload-based solution can improve the task execution up to 56.97% compared to random-based task allocation.

Although performing task allocation based on a single parameter helps to minimize the execution time, a greater reduction in execution time is possible by selecting compute nodes based on two and three parameters. Moreover, the solution of task allocation by considering only single parameter does not guarantee optimal reduction in the task execution time. To see a shorter execution time, the nodes for performing task execution must be selected based on high CPU speed and a lighter running workload. This criterion ensures improvement in task execution time up to 61.23% compared to random-based solution. Although much reduction in execution time is seen in these four solutions, a further reduction in execution time is also possible by basing task allocation on three parameters together, such as CPU speed, number of cores, and workload. The consideration of these parameters together leads to appropriate resource utilization than random-based task allocation solution. Efficient resource utilization enables the mobile devices to execute the task quickly. Thus, the results show a substantial increase in the performance by reducing the execution time up to 71.55% through final proposed task allocation solution (SCW) that is based on three parameters, such as CPU speed, workload, and number of cores. In addition, the three parameters-based solution is analyzed as one of the best of the five in terms of execution time in our scenario.

6.5.1.1. Statistical Significance of the Proposed Solutions' Execution Time Results Compared to RM-based Task Allocation

The Mann-Whitney U test and Vargha and Delaney statistics were applied (in table 5.19 and 5.20) to find the significant differences and effectiveness in 30 data traces

of SO, CO, WO, SW, and SCW, in comparison with RM and each other (finding the best solution among five proposed solutions).

The P-values derived from the Mann-Whitney U test were measured 0.000058, 0.000173, 0.000051, 0.000008, and 0.000000029537. These values indicated that SO, CO, WO, SW, and SCW have significant statistical differences with RM in execution time (as the P values found less than 0.05). On the other hand, A_{12} statistics also revealed large effect size for all tests regarding execution time results. These two tests concluded that proposed solutions have significant statistical and practical differences from RM.

Table 5.20 presented the P-values derived through Mann-Whitney U test that were measured as 0.013549, 0.700686, 0.003848, 0.0000000001537, 0.009267, 0.000029, 0.000000000028719, 0.004745, 0.00000000024879, and 0.000002. These values indicated that SO, CO, WO, SW, and SCW have significant statistical differences with each other in execution time results (as the P values found less than 0.05). On the other hand, A_{12} statistics also revealed large effect sizes for seven tests, such as SW=SO, SCW=SO, SW=CO, SCW=CO, SW=WO, SCW=WO, and SCW=SW regarding execution time results. A medium effect was noted for CO=SO, and WO=CO. However, no effect size was found for SO=WO. Finally, our findings from both tests revealed that proposed solutions have significant statistical and practical differences from each other. Hence, we concluded that SCW was the best solution and preferable to SW, CO, WO, and SO in terms of execution time.

6.5.2. Energy Consumption

Figure 6.19 presents the results of energy consumption that are obtained through proposed heterogeneity-aware task allocation solutions. The purpose of this section is to discuss the performance of proposed solutions in terms of energy consumption.

The results of the CPU speed-based task allocation show that it consumes less energy than the random-based solution. The reason for less energy consumption is that the proposed solution enables the controller node in a way that always assigns the task to a high-speed mobile device that executes it quickly. Because of the fast execution, energy consumption is also minimized. However, in random-based task allocation, controller node allocates the task to the slower device as it does not consider the compute node resources that prolong its execution time and also consumes more energy. Thus, the CPU speed-based task allocation solution helps to save energy up to 69.78%.

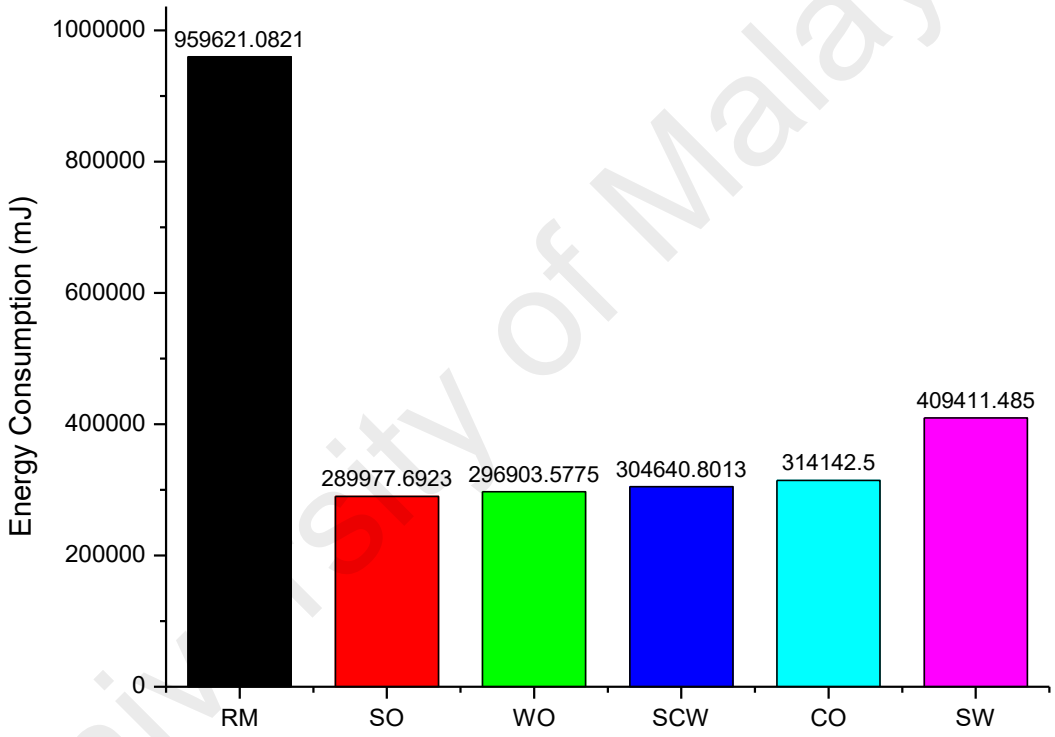

Figure 6.19: Comparison of energy consumption empirical results obtained from five proposed solutions with random-based task allocation

The results of core-based task allocation are also promising than random-based task allocation in terms of energy consumption. In random-based task allocation, most of the time, resources are not utilized in an efficient way. Inefficient resource utilization wastes energy. However, core-based task allocation ensures efficient resource utilization which leads to minimizing the energy consumption. The reason of minimizing the energy consumption is that proposed solution enables the controller to select such

compute nodes that have more number of cores. Thus, the selection of such device helps to execute the task in a quick manner than random-based task allocation because of its high processing capabilities. In this way, the core-based solution helps to reduce the energy consumptions up to 68.25% compared to random-based task allocation.

The workload-based task allocation solution enables the controller node to select only that device which has a lighter workload running in the background. Based on the workload criteria, much energy can be saved than random-based task allocation. In random-based task allocation, the larger tasks can be allocated to such devices that have already much workload running in the background which causes higher energy consumption. However, the workload-based selection of compute node helps to minimize the energy consumption. Thus, the results show that workload-based task allocation helps to save energy up to 69.06%.

The results of the two parameters-based (CPU speed and workload) task allocation reveal that it helps to reduce energy consumption compared to random-based task allocation. Although performing task allocation based on two parameters helps to reduce energy consumption up to 67.26% compared to random-based task allocation, single parameter-based task allocation helps to save more energy. The reason is that through single parameter-based task allocation, the tasks are being executed only on one device that usually has high specifications; whereas, through multiple parameters-based task allocation, tasks are usually allocated to more than one device as noted in our simulation, where specifications of the compute nodes were quite diverse. In that simulation scenario, although execution time could be minimized due to time slots factor compared to single parameter-based task allocation, energy consumption could not be minimized, in comparison with single parameter-based task allocation. Such is also the case with two and three parameters-based task allocation solutions (SW and SCW). Although the results of SW and SCW show that the solutions save energy

consumption up to 67.26 % and 57.33%, respectively, these results were not better than two single parameter-based task allocation solutions, such as SO and WO.

Hence, the discussion concludes that proposed heterogeneity-aware task allocation solutions use less energy than the random-based task allocation. In addition, the CPU speed-based task allocation is one of the most energy-efficient solutions in our simulation scenario.

6.5.2.1. Statistical Significance of the Proposed Solutions' Energy Consumption Results Compared to RM-based Task Allocation

The Mann-Whitney U test and Vargha and Delaney's A_{12} statistics were applied (in table 5.22) to measure the significant differences and effectiveness in 30 data traces of SO, CO, WO, SW, and SCW, in comparison with RM.

The P-values derived through Mann-Whitney U test were measured 0.000012, 0.000018, 0.000012, 0.000388, and 0.000015. These values indicated that SO, CO, WO, SW, and SCW have significant differences from RM in energy consumption results (as the P values found less than 0.05). At the same time, A_{12} statistics revealed large effect size for all tests regarding energy consumption results. These two tests concluded that the proposed solutions have significant statistical and practical differences from RM.

6.6 Conclusion

Heterogeneity-aware task allocation helps to address the issue of longer execution time and greater energy consumption by enabling the controller node to make an efficient task allocation decision. The proposed solutions help to minimize the task execution time and energy consumption for MAC-based task execution. The proposed solutions are implemented in controller node that is responsible for allocating the task in MAC environment. We validated the mathematical model presented in chapter 4 using simulation results. The differences between the execution time results obtained from the mathematical model and the results of each proposed solution were not significant as we

have investigated this through the Mann-Whitney U test, Pearson's correlation model, and Vargha and Delaney statistics. The verification of the proposed solutions performance with random-based task allocation in terms of execution time and energy consumption has also been performed. The performance evaluation results reveal that the heterogeneity-aware task allocation solutions, such as SO, CO, WO, SW, and SCW, outperform the random-based solution by reducing the execution time up to 56.72%, 53.12%, 56.97%, 61.23%, and 71.55%, respectively. In addition, these heterogeneity-aware task allocation solutions help to save energy up to 69.78%, 69.06%, 68.25%, 67.26%, and 57.33%, respectively. Based on the obtained results, we concluded that the proposed solutions help to improve task performance in terms of execution time and energy consumption, in comparison with random-based solution. These results have been verified by applying different statistical methods. The results of statistical analyses revealed that the differences were significant between the each proposed and random-based task allocation results. Hence, these statistical methods verified that our proposed solutions were statistically and practically superior to the random-based solution.

CHAPTER 7: CONCLUSION

We conclude the thesis by reporting on the re-examination of the research objectives defined in the first chapter. The purpose of this chapter is to summarize the research contributions and recommend future directions.

The chapter is organized into four sections. Section 7.1 discusses the reassessment of the objectives of this research. Section 7.2 highlights the contribution of the study. Section 7.3 examines the scope and limitation of the research work. Section 7.4 suggests future research directions of the study.

7.1 Reappraisal of the Research Objectives

The problem of inefficient task allocation causes by ignorance of heterogeneity-measuring parameters and its adverse impacts on task execution time and energy consumption have been investigated and addressed in this thesis. We revisit the four objectives and describe how this research study meets the objectives that were defined in Section 1.4.

The first objective was to review the state-of-the-art on MAC for obtaining insights with respect to task allocation issue. This has been achieved by analyzing the several problems inhibiting the adoption of MAC and reviewing corresponding solutions by devising a taxonomy. To study the state-of-the-art research carried out in MAC, online digital libraries including IEEE, ACM, Springer, and Elsevier have been used. Thus, numerous papers in the broader domain of MAC have been studied. Qualitative exploration was being done to analyze the strengths and weaknesses of the state-of-the-art literature and to identify the research issues that remain to be addressed. Among these issues, we have identified inefficient task allocation as a research problem.

The second objective was to investigate the impact of heterogeneity-measuring parameters and random-based task allocation on task execution performance. To achieve this objective, we have run a designed multi-threaded matrix multiplication

application and analyzed its impact on task execution time by varying the specifications and background workload on the mobile device. The empirical analysis revealed that performing random-based task allocation by ignoring the resource and operational heterogeneity in MAC prolongs the execution time and consumes high energy. Moreover, the analysis revealed that existing random-based task allocation mechanism was unable to adequately mitigate the impact of resource and operational heterogeneity on MAC-based task execution. Hence, this established the problem.

The third objective was to propose and develop five heterogeneity-aware task allocation solutions for minimizing the task execution time and energy consumption, and devise a mathematical model. The developed heterogeneity-aware task allocation algorithms helped to tackle the issue of longer execution time and high energy consumptions during the task execution in MAC paradigm. The proposed solutions enable the controller node to make task allocation decisions by considering the processing capabilities and operational context of compute nodes. Thus, proposed task allocation solutions help to reduce a significant amount of execution time and energy consumption by assigning compute-intensive tasks to such devices that can execute them in a fast manner.

The final objective was to evaluate the performance of the proposed heterogeneity-aware task allocation solutions with random-based task allocation in terms of execution time and energy consumption, and validate the developed mathematical model. The proposed solutions have been implemented in a simulated MAC environment. The performance evaluation results revealed that the proposed heterogeneity-aware task allocation solutions outperform the random-based task allocation. In comparison with the random-based task allocation, the proposed five solutions SO, CO, WO, SW, and SCW, reduce the execution time up to 56.72%, 53.12%, 56.97%, 61.23%, and 71.55%, respectively. In addition, these heterogeneity-

aware task allocation solutions save energy up to 69.78%, 69.06%, 68.25%, 67.26%, and 57.33%, respectively. We have also applied two statistical tests to find whether or not the differences between the results obtained from the heterogeneity-aware task allocation solutions and random-based task allocation were significant. Our findings from both of the tests revealed that proposed solutions have significant statistical and practical differences compared to random-based solution. We have validated the developed model by comparing the execution time results obtained from the model with the five proposed solutions. To prove that the differences were not significant between the model and simulation results, three statistical methods Pearson's correlation coefficient, Mann-Whitney U test, and Vargha and Delaney's A_{12} have been applied. The results of these statistical methods indicated that differences in execution time were not significant. Hence, the small differences validated the model results with those collected from the simulation.

7.2 Contributions of the Research

The contributions of this work to the body of knowledge are as follows:

- **Taxonomies:** We analyzed several problems inhibiting the adoption of MAC and reviewed the critical aspects of the corresponding solutions by devising the taxonomy. Moreover, MAC roots were also analyzed and taxonomized by classifying the literature. The literature review contributed to identifying open issues that remain to be addressed in MAC.
- **Empirical Analyses of Ignoring Heterogeneity-Measuring Parameters and Random-based Task Allocation Impact on Tasks' Execution Performance:** We contributed to the body of knowledge by investigating and demonstrating the adverse impacts of inefficient task allocation problem causes by ignorance of heterogeneity-measuring parameters, while making task allocation decisions in MAC. We perform an in-depth investigation of the problem by conducting an

experimental study to show that ignorance of the heterogeneity-measuring parameters and random-based task allocation can considerably prolong the tasks' execution time and consumes large amounts of energy in chapter 3.

- **Heterogeneity-Aware Task Allocation Algorithms:** We proposed five heterogeneity-aware task allocation algorithms that enable the controller node in MAC to address the issue of longer execution time and high energy consumption. Unlike the contemporary approach for handling the task execution in MAC, proposed algorithms reduce the execution time with minimal energy consumption. The existing random-based task allocation solution fails to perform task execution in such a way that it can help to minimize the execution time. On the contrary, heterogeneity-aware task allocation algorithms reduce the energy consumption by executing the task in a fast manner.
- **Mathematical Model for Validation:** We developed a mathematical model with respect to execution time. The model captures the key features of the heterogeneity-aware task allocation solutions and represents in mathematical form. The developed model is validated by comparing the results of the model with the execution time results obtained from the proposed solutions in a simulated environment.
- **Performance Evaluation of Heterogeneity-Aware Task Allocation Algorithms:** We performed evaluation by comparing the execution time and energy consumption results of five proposed heterogeneity-aware task allocation solutions with random-based task allocation solution. The evaluation results revealed that, in comparison with the random-based task allocation (RM), the proposed five solutions based on CPU speed (SO), number of core (CO), workload (WO), CPU speed and workload (SW), and CPU speed, core, and workload (SCW) reduce execution time up to 56.72%, 53.12%, 56.97%,

61.23%, and 71.55%, respectively. In addition, these heterogeneity-aware task allocation solutions save energy up to 69.78%, 69.06%, 68.25%, 67.26%, and 57.33%, respectively.

- **Statistical Analyses:** We contributed to the body of knowledge by identifying and discussing the statistical and practical differences between the results obtained from random and proposed heterogeneity-aware task allocation solutions. We applied Mann-Whitney U test and Vargha and Delaney's A_{12} statistics to find the significance of differences between the results. Our findings from both tests revealed that proposed solutions have significant statistical and practical differences compared to random-based solution. In addition, we also applied Mann-Whitney U test, Vargha and Delaney's A_{12} statistics, and Pearson's correlation coefficient model to validate one of the developed mathematical model results with empirical results obtained from the five proposed solutions. The statistical analyses results revealed that there are no practical and statistical differences between the mathematical model and proposed solutions' results. Hence, it validated the mathematical model.

  We have successfully published our work in well-reputed journals.

- **Accepted Article on Research Topic**

  Ibrar Yaqoob, Ejaz Ahmed, Abdullah Gani, Salimah Mokhtar, Muhammad Imran, Sghaier Guizani, Mobile ad hoc cloud: A survey, *Wireless Communications and Mobile Computing, 16(2016), 2572-2589 (Impact Factor 0.92).*

  Ibrar Yaqoob, Ejaz Ahmed, Abdullah Gani, Salimah Mokhtar, Muhammad Imran, Heterogeneity-Aware Task Allocation in Mobile Ad Hoc Cloud, *IEEE Access, 5 (2017): 1779-179 (Impact Factor 1.27).*

- **Accepted Article on Other Research Topic**

  Ibrar Yaqoob, Ibrahim Abaker Targio Hashem, Yasir Mehmood, Abdullah Gani, Salimah Mokhtar, Sghaier Guizani, Enabling Communication Technologies for Smart Cities, *IEEE Communications Magazine, 55.1 (2017): 112-120 (Impact Factor 5.12).*

  Ibrar Yaqoob, Ibrahim Abaker Targio Hashem, Abdullah Gani, Salimah Mokhtar, Ejaz Ahmed, Nor Badrul Anuar, Anthanasios Vasilakos, Big Data: From Beginning to Future, *International Journal of Information Management, 36(6), 1231-1247 (Impact Factor 2.69).*

  Ibrar Yaqoob, Ejaz Ahmed, Ibrahim Abaker Targio Hashem, A. I. A. Ahmed, Abdullah Gani, Muhammad Imran, Mohsen Guizani, Internet of Things Architecture: Recent Advances, Taxonomy, Requirements, and Open Challenges, *IEEE Wireless Communications, 2017(Impact Factor 4.14).*

  Ibrar Yaqoob, Iftikhar Ahmad, Ejaz Ahmed, Abdullah Gani, Muhammad Imran, Nadra Guizani, Overcoming the key challenges of establishing vehicular communication: Is SDN the answer?, *IEEE Communications Magazine, 2017 (Impact Factor 5.12).*

- **Accepted Article on Other Research Topic with Group Collaboration**

  Ejaz Ahmed, Ibrar Yaqoob, Abdullah Gani, Muhammad Imran, Mohsen Guizani, Social-Aware Resource Allocation and Optimization for D2D Communication, *IEEE Wireless Communications, vol.PP,no.99, pp.2-9, 2017 (Impact Factor 4.14).*

  Ejaz Ahmed, Ibrar Yaqoob, Arif Ahmed, Abdullah Gani, Muhammad Imran, Sghaier Guizani, Green Industrial Networking: Recent Advances, Taxonomy, and Open Research Challenges, *IEEE Communications Magazine, 54(10), 38-45 (Impact Factor 5.12).*

Ejaz Ahmed, Ibrar Yaqoob, Abdullah Gani, Muhammad Imran, Sghaier Guizani, Internet of Things based Smart Environments: State-of-the-art, Taxonomy, and Open Research Challenges, *IEEE Wireless Communications, 23(5), 10-16 (Impact Factor 5.12).*

Yasir Mehmood, Farhan Ahmad, Ibrar Yaqoob, Asma Adnane, Muhammad Imran, Sghaier Guizani, Internet-of-Things Based Smart Cities: Recent Advances and Challenges, *IEEE Communications Magazine, 2017(Impact Factor 5.12).*

Ibrahim Abaker Targio Hashem, Victor Chang, Nor Badrul Anuar, Kayode Adewole, Ibrar Yaqoob, Abdullah Gani, Ejaz Ahmed, and Haruna Chiroma, The role of big data in smart city, *International Journal of Information Management 36, Volume no. 5 (2016): 748-758 (Impact Factor 5.12).*

Aisha Siddiqa, Ibrahim Abaker Targio Hashem, Ibrar Yaqoob, Mohsen Marjani, Shahabuddin Shamshirband, Abdullah Gani, and Fariza Nasaruddin, A Survey of Big Data Management: Taxonomy and State-of-the-Art, *Journal of Network and Computer Applications, Volume no. 71(2016): 151-166 (Impact Factor 2.22)*

Ibrahim Abaker Targio Hashem, Nor Badrul Anuar, Abdullah Gani, Ibrar Yaqoob, Feng Xia, and Samee Ullah Khan, MapReduce: Review and open challenges, *Scientometrics, Volume no. 109 (2016):389-422 (Impact Factor 2.03).*

Ibrahim Abaker Targio Hashem, Ibrar Yaqoob, Nor Badrul Anuar, Salimah Mokhtar, Abdullah Gani, and Samee Ullah Khan, The rise of "big data" on cloud computing: Review and open research issuesm, *Information Systems, Volume no. 47(2015): 98-115 (Impact Factor 1.76).*

Nawsher Khan, Ibrar Yaqoob, Ibrahim Abaker Targio Hashem, Zakira Inayat, Waleed Kamaleldin Mahmoud Ali, Muhammad Alam, Muhammad Shiraz, and Abdullah Gani, Big data: survey, technologies, opportunities, and challenges, *The Scientific World Journal, Vol. 2014, Article ID 712826, 18 pages, 2014 (Impact Factor 1.73).*

7.3 Research Scope and Limitations

The proposed heterogeneity-aware task allocation algorithms are effective for all MAC-based compute-intensive tasks execution. In MAC, the controller node can adopt this solution to make any task allocation decision that can help to minimize the execution time and energy consumption. In addition, cloudlet based applications can also take advantage of these algorithms in a particular case when execution is performed in a distributed manner.

Despite many advantages of the heterogeneity-aware task allocation solutions, there are some limitations. In the proposed single parameter-based task allocation solutions, particularly in CPU speed (SO) and number of core (CO) based task allocation, the tasks are only being executed on a single compute node because of the defined selection criteria in the algorithms. Thus, this causes wastage of the resources as other mobile devices are not selected and utilized for performing computation in those cases. Similarly, the proposed solutions usually require the complete specification and workload information of the compute nodes to make the task allocation decision that poses significantly higher overhead on the controller node. Moreover, one of the limitations is that once controller collects all the information of the specifications and workload from the compute nodes, it makes task allocation decision based on the defined policy in the proposed algorithms; however, the workload information may be changed in between task assignment process on the compute nodes. Thus, in such situation, the proposed heterogeneity-aware task allocation solutions do not incorporate

any feature that enables the controller to change the task allocation policy at the run time.

7.4 Future Work

This study does not investigate all the aspects of task allocation problem in MAC. However, numerous research efforts have been carried out into this study. To explore the open research issues, several future research directions are suggested in which carrying further research can extend this study. This research work is only focused on the incorporation of the heterogeneity-awareness while making task allocation decision in MAC. However, after performing task allocation -if certain compute node leaves the MAC or task is allocated to such device that has more resources, but low battery power- then managing these situations are the key concerns that remain to be addressed.

Future directions of this research are as follows:

- The issue of identifying that which compute nodes have more stability patterns is aiming to be addressed in the future research. The compute nodes mobility information must be shared in the initial phases of MAC formation. In this context, there should be some mechanism that can enable the mobile devices to find their mobility pattern information and share it during the MAC formation. Thus, this requires extensive research work to be done.
- Performing task allocation by considering the battery power information of the compute nodes is remained to be addressed as a future research direction. The proposed solutions are not considering mobile battery power information while making task allocation decision that can cause real problems in certain cases if its battery dies after receiving the task for the execution. Thus, this can be done in the future.

Currently, we are only aware of these possible future research directions. However, there might be multiple other research areas which we are not aware of at this time. No matter how complete the research study is, the fascinating minds are always ready to come up with new ways to improve the state-of-the-art work and contribute to the body of knowledge.

REFERENCES

Abolfazli, Saeid, Sanaei, Zohreh, Ahmed, Ejaz, Gani, Abdullah, & Buyya, Rajkumar. (2014). Cloud-based augmentation for mobile devices: motivation, taxonomies, and open challenges. *IEEE Communications Surveys & Tutorials, 16*(1), 337-368.

Ahlgren, Per, Jarneving, Bo, & Rousseau, Ronald. (2003). Requirements for a cocitation similarity measure, with special reference to Pearson's correlation coefficient. *Journal of the American Society for Information Science and Technology, 54*(6), 550-560.

Ahmed, E., Gani, A., Khan, M. K., Buyya, R., & Khan, S. U. (2015). Seamless application execution in mobile cloud computing: Motivation, taxonomy, and open challenges. *Journal of Network and Computer Applications, 52*, 154-172. doi: 10.1016/j.jnca.2015.03.001

Akinola, Ayotuyi T, Adigun, Matthew O, & Akingbesote, Alaba O. (2015). *QoS-aware single service selection mechanism for ad-hoc mobile cloud computing.* Paper presented at the International Conference on Computing, Communication and Security (ICCCS), 2015, pp. 1-6.

Al Noor, Shahid, Hasan, Ragib, & Haque, Md Munirul. (2014). *Cellcloud: A novel cost effective formation of mobile cloud based on bidding incentives.* Paper presented at the 2014 IEEE 7th International Conference on Cloud Computing, pp.200-207.

Alnuem, Mohammed, Zafar, Nazir Ahmad, Imran, Muhammad, Ullah, Sana, & Fayed, Mahmoud. (2014). Formal specification and validation of a localized algorithm for segregation of critical/noncritical nodes in MAHSNs. *International Journal of Distributed Sensor Networks, 2014*, 10(6).

AnTuTu. (2011). Master CPU. Retrieved 15 August, 2015, from https://play.google.com/store/apps/details?id=com.antutu.CpuMaster

Arcuri, Andrea, & Briand, Lionel. (2011). *A practical guide for using statistical tests to assess randomized algorithms in software engineering.* Paper presented at the 2011 33rd International Conference on Software Engineering (ICSE), pp.1-10.

Armbrust, Michael, Fox, Armando, Griffith, Rean, Joseph, Anthony D, Katz, Randy, Konwinski, Andy, Stoica, Ion. (2010). A view of cloud computing. *Communications of the ACM, 53*(4), 50-58.

Bahl, Paramvir, Han, Richard Y, Li, Li Erran, & Satyanarayanan, Mahadev. (2012). *Advancing the state of mobile cloud computing.* Paper presented at the Proceedings of the third ACM workshop on Mobile cloud computing and services, pp.21-28.

Balasubramanian, Aruna, Mahajan, Ratul, & Venkataramani, Arun. (2010). *Augmenting mobile 3G using WiFi.* Paper presented at the Proceedings of the 8th international conference on Mobile systems, applications, and services, pp. 209-222.

Basarkod, PI, Manvi, Sunilkumar S, & Albur, DS. (2013). *Mobility based estimation of node stability in MANETs.* Paper presented at the Emerging Trends in International Conference on Computing, Communication and Nanotechnology (ICE-CCN), 2013, pp. 126-130.

Benkhelifa, Elhadj, Welsh, Thomas, Tawalbeh, Loai, Khreishah, Abdallah, Jararweh, Yaser, & Al-Ayyoub, Mahmoud. (2016). *GA-Based Resource Augmentation Negotation for Energy-Optimised Mobile Ad-hoc Cloud.* Paper presented at the 2016 4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud), pp.110-116.

Buyya, Rajkumar, Yeo, Chee Shin, Venugopal, Srikumar, Broberg, James, & Brandic, Ivona. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation computer systems, 25*(6), 599-616.

Chen, Min, Hao, Yixue, Li, Yong, Lai, Chin-Feng, & Wu, Di. (2015). On the computation offloading at ad hoc cloudlet: architecture and service modes. *Communications Magazine, IEEE, 53*(6), 18-24.

Chi, Fangyuan, Wang, Xiaofei, Cai, Wei, & Leung, Victor. (2014). *Ad Hoc Cloudlet Based Cooperative Cloud Gaming.* Paper presented at the IEEE 6th International Conference on Cloud Computing Technology and Science (CloudCom), 2014, pp. 190-197.

Čokulov, Predrag. (2014). Kernel Tuner. Retrieved 15 August, 2015, from https://play.google.com/store/apps/details?id=rs.pedjaapps.KernelTuner&hl=en

Egbe, Dominic Afuro, Akingbesote, AO, Adigun, MO, & Mutanga, MB. (2016). *Context based service discovery algorithm for ad hoc mobile cloud.* Paper presented at the 2016 International Conference on Industrial Informatics and Computer Systems (CIICS), pp. 1-6.

Ejaz, Ahmed. (2016). *Process state synchronization for mobility support in mobile cloud computing/Ejaz Ahmed.* University of Malaya.

Falaki, Hossein, Mahajan, Ratul, Kandula, Srikanth, Lymberopoulos, Dimitrios, Govindan, Ramesh, & Estrin, Deborah. (2010). *Diversity in smartphone usage.* Paper presented at the Proceedings of the 8th international conference on Mobile systems, applications, and services, pp. 179-194.

Fang, Weiwei, Li, Yangchun, Zhang, Huijing, Xiong, Naixue, Lai, Junyu, & Vasilakos, Athanasios V. (2014). On the throughput-energy tradeoff for data transmission between cloud and mobile devices. *Information Sciences, 283*, 79-93.

Fernando, N., Loke, S. W., & Rahayu, W. (2016). Computing with Nearby Mobile Devices: a Work Sharing Algorithm for Mobile Edge-Clouds. *IEEE Transactions on Cloud Computing, PP*(99), 1-1. doi: 10.1109/TCC.2016.2560163

Fernando, Niroshinie, Loke, Seng W, & Rahayu, Wenny. (2011). *Dynamic mobile cloud computing: Ad hoc and opportunistic job sharing.* Paper presented at 2011 Fourth IEEE International Conference on Utility and Cloud Computing (UCC), pp.281-286.

Ferreira, Daniel Furtado. (2011). Sisvar: a computer statistical analysis system. *Ciência e agrotecnologia*, *35*(6), 1039-1042.

Gavalas, Damianos, Konstantopoulos, Charalampos, & Pantziou, G. (2010). Mobility Prediction in Mobile Ad Hoc Networks. *Next Generation Mobile Networks and Ubiquitous Computing*, 226-240.

Golchay, Roya, Le Mouël, Frédéric, Ponge, Julien, & Stouls, Nicolas. (2016). *Spontaneous Proximity Clouds: Making Mobile Devices to Collaborate for*

Resource and Data Sharing. Paper presented at the Proceedings of the 12th EAI International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom'2016), Nov 2016, Beijing, China.

Gong, Y., Zhang, C., Fang, Y., & Sun, J. (2015). Protecting Location Privacy for Task Allocation in Ad Hoc Mobile Cloud Computing. *Emerging Topics in Computing, IEEE Transactions on, PP*(99), 1-1. doi: 10.1109/TETC.2015.2490021

Guo, Xijuan, Liu, Liqing, Chang, Zheng, & Ristaniemi, Tapani. (2016). Data offloading and task allocation for cloudlet-assisted ad hoc mobile clouds. *Wireless Networks*, 1-10. doi: 10.1007/s11276-016-1322-z

Hammam, Ahmed, & Senbel, Samah. (2013). *A trust management system for ad-hoc mobile clouds.* Paper presented at the 2013 8th International Conference on Computer Engineering & Systems (ICCES), pp. 31-38.

Hamza, S., Okba, K., Aicha-Nabila, B., & Youssef, A. (2012). A Cloud computing approach based on mobile agents for web services discovery. *2012 Second International Conference on Innovative Computing Technology (Intech)*, 297-304.

Holt, Nina Elisabeth, Briand, Lionel C, & Torkar, Richard. (2014). Empirical evaluations on the cost-effectiveness of state-based testing: An industrial case study. *Information and Software Technology, 56*(8), 890-910.

Huerta-Canepa, Gonzalo, & Lee, Dongman. (2010). *A virtual cloud computing provider for mobile devices.* Paper presented at the Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond, p.6.

Idowu, Ibrahim Olatunji, Shi, Qi, Merabti, Madjid, & Kifayat, Kashif. (2012). Ad-Hoc Cloud Networks: A Probabilistic Model for Vulnerability Detection in Critical Infrastructure Using Bayesian Networks.

Imran, Muhammad, Alnuem, Mohamed A, Fayed, Mahmoud S, & Alamri, Atif. (2013). Localized algorithm for segregation of critical/non-critical nodes in mobile ad hoc and sensor networks. *Procedia Computer Science, 19*, 1167-1172.

Kaur, Amanpreet Kaur Amandeep. (2014). Improving Node Stability Using Hotspot Algorithm in Mobile Ad-hoc Network. *International Journal of Research in Advent Technology*, 87-93.

Khalifa, Ahmed, Azab, Mohamed, & Eltoweissy, Mohamed. (2014a). *Resilient hybrid mobile ad-hoc cloud over collaborating heterogeneous nodes.* Paper presented at the 2014 International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), pp. 134-143.

Khalifa, Ahmed, Azab, Mohamed, & Eltoweissy, Mohamed. (2014b). *Towards a Mobile Ad-Hoc Cloud Management Platform.* Paper presented at the Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing, pp. 427-434.

Kristensen, Mads Darø, & Bouvin, Niels Olof. (2010). Scheduling and development support in the scavenger cyber foraging system. *Pervasive and Mobile Computing, 6*(6), 677-692.

Lacuesta, Raquel, Lloret, Jaime, Sendra, Sandra, & Peñalver, Lourdes. (2014). Spontaneous Ad Hoc Mobile Cloud Computing Network. *The Scientific World Journa*l, vol. 2014, Article ID 232419, 19 pages, 2014. doi:10.1155/2014/232419.

Li, Bo, Pei, Yijian, Wu, Hao, & Shen, Bin. (2015). Heuristics to allocate high-performance cloudlets for computation offloading in mobile ad hoc clouds. *The Journal of Supercomputing, 71(8), pp. 3009-3036.*

Li, Chunlin, Yanpei, Liu, & Youlong, Luo. (2016). Efficient service selection approach for mobile devices in mobile cloud. *The Journal of Supercomputing, 72*(6), 2197-2220.

Loke, Seng W, Napier, Keegan, Alali, Abdulaziz, Fernando, Niroshinie, & Rahayu, Wenny. (2015). Mobile computations with surrounding devices: Proximity

sensing and multilayered work stealing. *ACM Transactions on Embedded Computing Systems (TECS), 14*(2), 22.

Lu, Zongqing, Zhao, Jing, Wu, Yibo, & Cao, Guohong. (2015). *Task Allocation for Mobile Cloud Computing in Heterogeneous Wireless Networks.* Paper presented at the 2015 24th International Conference on Computer Communication and Networks (ICCCN), pp. 1-9.

Malhotra, Ahana, Dhurandher, Sanjay Kumar, & Kumar, Bijendra. (2014). *Resource allocation in multi-hop Mobile Ad hoc cloud.* Paper presented at the 2014 Recent Advances in Engineering and Computational Sciences (RAECS), pp. 1-6.

Maly, Filip, & Kriz, Pavel. (2015). An Ad Hoc Mobile Cloud and Its Dynamic Loading of Modules into a Mobile Device Running Google Android *New Trends in Intelligent Information and Database Systems* (pp. 191-198): Springer.

Mandal, Subhajit, Yang, Chen, Altaweel, Ala, & Stoleru, Radu. (2015). *An efficient pairwise key establishment scheme for Ad-Hoc Mobile Clouds.* Paper presented at the 2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), pp. 627-634.

McGilvary, G. A., Barker, A., & Atkinson, M. (2015, June 27 2015-July 2 2015). *Ad Hoc Cloud Computing.* Paper presented at the 2015 IEEE 8th International Conference on Cloud Computing, pp. 1063-1068.

Mell, Peter, & Grance, Tim. (2009). The NIST definition of cloud computing. *National Institute of Standards and Technology, 53*(6), 50.

Miluzzo, Emiliano, Cáceres, Ramón, & Chen, Yih-Farn. (2012). *Vision: mClouds-computing on clouds of mobile devices.* Paper presented at the Proceedings of the third ACM workshop on Mobile cloud computing and services, pp. 9-14.

Mtibaa, Abderrahmen, Fahim, Afnan, Harras, Khaled A, & Ammar, Mostafa H. (2013). *Towards resource sharing in mobile device clouds: Power balancing across mobile devices.* Paper presented at the ACM SIGCOMM Computer Communication Review, Vol. 43, No. 4, pp. 51-56.

Ordonez-Morales, Esteban F, Blanco-Fernandez, Yolanda, Bravo-Torres, Jack F, Lopez-Nores, Mart, Saiáns-Vázquez, Vıctor, & Pazos-Arias, José J. (2015). *S-*

CMA: sporadic cloud-based mobile augmentation supported by an ad-hoc cluster of moving handheld devices and a virtualization layer. Paper presented at the 2015 Fifth International Conference on Innovative Computing Technology (INTECH), pp. 152-157.

Pedersen, Morten V, & Fitzek, Frank HP. (2012). Mobile clouds: the new content distribution platform. *Proceedings of the IEEE, 100* (Special Centennial Issue), 1400-1403.

Penner, Terry, Johnson, Alison, Van Slyke, Brandon, Guirguis, Mina, & Gu, Qijun. (2014). *Transient clouds: Assignment and collaborative execution of tasks on mobile devices.* Paper presented at the 2014 IEEE Global Communications Conference (GLOBECOM), pp. 2801-2806.

Rahimi, M Reza, Venkatasubramanian, Nalini, & Vasilakos, Athanasios V. (2013). *Music: Mobility-aware optimal service allocation in mobile cloud computing.* Paper presented at 2013 IEEE Sixth International Conference on the Cloud Computing (CLOUD), pp. 75-82.

Rashidi, Shima, & Sharifian, Saeed. (2017). A hybrid heuristic queue based algorithm for task assignment in mobile cloud. *Future Generation Computer Systems, 68, 331-345.*

Satyanarayanan, Mahadev, Bahl, Paramvir, Caceres, Ramón, & Davies, Nigel. (2009). The case for vm-based cloudlets in mobile computing. *IEEE pervasive Computing, 8*(4), 14-23.

Sciarrone, A., Bisio, I., Lavagetto, F., Penner, T., & Guirguis, M. (2015, 6-10 Dec. 2015). *Context Awareness over Transient Clouds.* Paper presented at the 2015 IEEE Global Communications Conference (GLOBECOM), pp. 1-5.

Shaukat, Usman, Ahmed, Ejaz, Anwar, Zahid, & Xia, Feng. (2016). Cloudlet Deployment in Local Wireless Networks: Motivation, Architectures, Applications, and Open Challenges. *Journal of Network and Computer Applications, 62, pp.18-40.*

Sheskin, David J. (2003). *Handbook of parametric and nonparametric statistical procedures*: crc Press.

Shi, Ting, Yang, Mei, Li, Xiang, Lei, Qing, & Jiang, Yingtao. (2016). An energy-efficient scheduling scheme for time-constrained tasks in local mobile clouds. *Pervasive and Mobile Computing, 27*, 90-105. doi: http://dx.doi.org/10.1016/j.pmcj.2015.07.005

Shila, Devu Manikantan, Shen, Wenlong, Cheng, Yu, Tian, Xiaohua, & Shen, Xuemin Sherman. (2016). AMCloud: Towards a Secure Autonomic Mobile Ad Hoc Cloud Computing System. *IEEE Wireless Communications, vol. PP, no. 99, pp. 1–8.*

Shiraz, Muhammad, Ahmed, Ejaz, Gani, Abdullah, & Han, Qi. (2014). Investigation on runtime partitioning of elastic mobile applications for mobile cloud computing. *The Journal of Supercomputing, 67*(1), 84-103.

Shiraz, Muhammad, & Gani, Abdullah. (2014). A lightweight active service migration framework for computational offloading in mobile cloud computing. *The Journal of Supercomputing, 68*(2), 978-995.

Tang, L., He, S., & Li, Q. (2016). Double-sided Bidding Mechanism for Resource Sharing in Mobile Cloud. *IEEE Transactions on Vehicular Technology, PP*(99), 1-1. doi: 10.1109/TVT.2016.2565505

Thapa, Sobit Bahadur, & Gu, Qijun. (2016). Collaborative task execution with originator data security for weak devices. *International Journal of Sensor Networks, 21*(2), 67-81.

Truong-Huu, Tram, Tham, Chen-Khong, & Niyato, Dusit. (2014). *A Stochastic Workload Distribution Approach for an Ad Hoc Mobile Cloud.* Paper presented at the 2014 IEEE 6th International Conference on Cloud Computing Technology and Science (CloudCom), pp. 174-181.

Vargha, András, & Delaney, Harold D. (2000). A critique and improvement of the CL common language effect size statistics of McGraw and Wong. *Journal of Educational and Behavioral Statistics, 25*(2), 101-132.

Verbelen, Tim, Simoens, Pieter, De Turck, Filip, & Dhoedt, Bart. (2012). *Cloudlets: bringing the cloud to the mobile user.* Paper presented at the Proceedings of the third ACM workshop on Mobile cloud computing and services, pp.29-36.

Yaqoob, Ibrar, Ahmed, Ejaz, Gani, Abdullah, Mokhtar, Salimah, Imran, Muhammad, & Guizani, Sghaier. (2016). Mobile ad hoc cloud: A survey. *Wireless Communications and Mobile Computing*, *16*(16), 2572-2589. doi: 10.1002/wcm.2709

Yousafzai, Abdullah, Chang, Victor, Gani, Abdullah, & Noor, Rafidah Md. (2016). Directory-based incentive management services for ad-hoc mobile clouds. *International Journal of Information Management*, *36*(6, Part A), 900-906. doi: http://dx.doi.org/10.1016/j.ijinfomgt.2016.05.019

Zaghdoudi, Bilel, Ayed, Hella Kaffel-Ben, & Riabi, Imen. (2015). Ad Hoc Cloud as a Service: A Protocol for Setting up an Ad hoc Cloud over MANETs. *Procedia Computer Science*, *56*, 573-579.

Zhang, Daqiang, Zhang, Daqing, Xiong, Haoyi, Hsu, Ching-Hsien, & Vasilakos, Athanasios V. (2014). BASA: Building mobile Ad-Hoc social networks on top of android. *Network, IEEE*, *28*(1), 4-9.

Zhang, Lide, Tiwana, Birjodh, Qian, Zhiyun, Wang, Zhaoguang, Dick, Robert P, Mao, Zhuoqing Morley, & Yang, Lei. (2010). *Accurate online power estimation and automatic battery behavior based power model generation for smartphones.* Paper presented at the Proceedings of the eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis, pp. 105-114.

Zhang, Weishan, Tan, Shouchao, Xia, Feng, Chen, Xiufeng, Li, Zhongwei, Lu, Qinghua, & Yang, Su. (2016). A survey on decision making for task migration in mobile cloud environments. *Personal and Ubiquitous Computing*, *20*(3), 295-309.

Zhou, B., Vahid Dastjerdi, A., Calheiros, R. N., Srirama, S., & Buyya, R. (2016). mCloud: A Context-aware Offloading Framework for Heterogeneous Mobile Cloud. *IEEE Transactions on Services Computing, vol. 9, no. 5*, pp. 757-770.