A LIGHTWEIGHT HETEROGENEOUS HYBRID MOBILE CLOUD COMPUTING FRAMEWORK FOR COMPUTE-INTENSIVE MOBILE APPLICATIONS

ZOHREH SANAEI MOGHADDAM

Faculty of Computer Science and Information Technology
University of Malaya
Kuala Lumpur

2014

A LIGHTWEIGHT HETEROGENEOUS HYBRID MOBILE CLOUD COMPUTING FRAMEWORK FOR COMPUTE-INTENSIVE MOBILE APPLICATIONS

ZOHREH SANAEI MOGHADDAM

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

Faculty of Computer Science and Information Technology
University of Malaya
Kuala Lumpur

2014

UNIVERSITI MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: Zohreh Sanaei Moghaddam (Passport No.:H91956583)

Registration/Matrix No.: WHA100049

Name of Degree: Doctor of Philosophy

Title of Thesis: A Lightweight Heterogeneous Hybrid Mobile Cloud Computing Framework For Compute-Intensive Mobile Applications

Field of Study: Mobile Computing

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

The state-of-the-art Mobile Cloud Computing (MCC) paradigm has gained a momentous ground to mitigate mobile devices' shortcomings (i.e., computing and energy) by outsourcing resource-intensive mobile tasks to the cloud. Researchers have proposed solutions for compute-intensive mobile applications by leveraging varied types of cloud-based resources, particularly coarse, medium, and fine granular cloud resources. Coarse-grained cloud resources feature high scalability and low locality that originates communication latency, fine-grained resources offer low scalability and high locality that leads to computation latency, and medium-grained resources provide medium scalability and locality breeding communication and computing latency. Such communication and computation latencies negatively impact on energy efficiency and response time of compute-intensive mobile applications leading to mobile application performance degradation. As a result, leveraging vertical heterogeneous granular cloud resources creates a bottleneck of limited computing and communication capabilities which results in increased response time and energy consumption. Vertical heterogeneity rises within one type of granular resources, like coarse or fine. This research is undertaken with the aim to obtain efficient computation outsourcing for compute-intensive mobile applications using horizontally heterogeneous granular cloud-based resource. Horizontal heterogeneity happens across varied types of granular resources, like coarse and fine. Using a series of benchmarking experiments we investigate the impacts of computation and communication latencies of granular resources on round-trip time and energy consumption of compute-intensive mobile applications and establish the research problem. Moreover, we propose a lightweight heterogeneous hybrid MCC framework for compute-intensive mobile applications that aims to reduce response time and prevent energy dissipation on mobile devices. We analyse execution

of a compute-intensive mobile application considering two performance metrics, namely Round-Trip Time (RTT) and Energy Consumption (EC) in two execution models of local and hybrid. We evaluate performance of the proposed framework in real environment and validate the results through statistical modelling. The results of RTT analysis advocates average of 93.5% RTT saving in hybrid mode compared with local mode and the EC analysis results testify average of 94% energy saving in hybrid mode compared with local mode. The results express that utilizing heterogeneous hybrid cloud-based computing resources can significantly reduce RTT and EC of mobile device in hybrid mode compared with local mode execution.

ABSTRACT

Kecanggihan paradigma Pengkomputeran Awan kembara atau Mobile Cloud Computing (MCC) telah menerima satu masa yang sesuai untuk mengurangkan pemasalahan peralatan mobil (seperti pengkomputeran dan tenaga) dengan menggunakan sumber luar bagi tugas-tugas mobil secara intensif terhadap awan. Penyelidik-penyelidik telah mencadangkan pelbagai penyelesaian untuk aplikasi-aplikasi mobil berasaskan komputasi dengan mengeluarkan pelbagai jenis sumber-sumber berasaskan awan, terutamanya sumber awan yang kasar, sederhana, dan halus. Sumber awan yang kasar memberi ciri pengskalaan yang tinggi dan penglokaliti yang rendah yang menyebabkan latensi komunikasi, sumber awan yang halus menawarkan pengskalaan yang rendah dan lokaliti yang tinggi yang menyebabkan latensi pengkomputeran, dan sumber awan yang sederhana menyediakan pengskalaan yang sederhana dan komunikasi yang melahirkan lokaliti dan latensi pengkomputeran. Komunikasi berkenaan dan latensi pengkomputeran secara negatifnya memberi impak kepada keberkesanan tenaga dan masa maklumbalas bagi aplikasi mobil yang berasaskan komputasi di mana ia menyebabkan penurunan keupayaan aplikasi mobil. Sebagai keputusannya, mengeluarkan sumber awan yang pelbagai membina satu leher botol yang sangat terhad kepada pengkomputeran dan keupayaan komunikasi di mana menyebabkan penambahan masa maklumbalas dan penggunaan tenaga. Kepelbagaian secara vertikal meningkatkan sejenis sumber granular sahaja seperti kasar atau halus. Penyelidikan ini mengsasarkan untuk memperolehi “outsourcing” komputasi yang berkesan bagi aplikasi mobil yang berasaskan komputasi menggunakan sumber berasaskan awan yang heterogeneous granular. Kepelbagaian secara horizontal berlaku pada semua jenis sumber granular seperti kasar dan halus. Dengan menggunakan satu siri aras tanda untuk eksperimen, kami menyiasat kesan-kesan komputasi dan komunikasi latensi

untuk sumber yang granular pada masa “round-trip” dan penggunaan masa untuk aplikasi mobil yang berasaskan komputasi dan megisytihar masalah penyelidikan yang berkenaan. Tambahan lagi, kami mencadangkan satu rangka kerja MCC secara hybrid heterogeneous ringan untuk aplikasi mobil yang berasaskan komputasi di mana ia mensasarkan masa tindak balas dan mengelak daripada pengselerakan tenaga pada alatan mobil. Kami menganalisa eksekusi aplikasi berasaskan komputasi berdasarkan dua pengukuran keupayaan, dinamakan sebagai “Round-Trip Time” (RTT) dan “Energy Consumption” (EC) dalam dua model pelaksanaan iaitu lokal dan hibrid. Kami menilai keupayaan rangka kerja yang dicadangkan dalam persekitaran sebenar dan mengesahkan keputusan melalui permodelan statistik. Keputusan untuk analisa RTT menunjukkan bahawa secara purata 93.5% RTT tersimpan dalam mod hybrid berbanding mod lokal dan analisa EC menunjukkan keputusan yang mengesahkan secara purata 94% tenaga tersimpan dalam mod hibrid berbanding mod lokal. Keputusan tersebut menunjukkan bahawa sumber komputasi berasaskan awan heterogeneous hybrid mampu secara signifikan mengurangkan RTT dan EC pada perkakasan mobil dalam pelaksanaan mod hibrid berbanding mod setempat.

ACKNOWLEDGEMENTS

Completing my PhD degree is probably the most challenging activity of my life. It has been a great privilege to spend three years in the department of Computer Science and IT at University of Malaya, and its members will always remain dear to me. However, It would have not been possible to write this doctoral thesis without the help and support of the kind people around me, to only some of whom I can give particular thanks here.

Above all, I would like to thank the world's creator for letting me through all the difficulties. I have experienced Your guidance day by day. I will keep on trusting You forever. Thank you.

My hearty thanks must go to my advisor, Prof. Dr. Abdullah Gani who has patiently provided the vision, encouragement and advise necessary for me to proceed through this doctoral program and complete my thesis. His continuous support and guidance helped me producing a valuable piece of research reported in this thesis.

I would also like to gratefully express my special appreciation and thanks to my beloved husband, Saeid Abolfazli, for his great support, encouragement and unwavering and unconditional love. He had always been a tremendous mentor for me. I would like to sincerely thank my dearest and loveliest parents for their faith in me and allowing me to be as ambitious as I wanted. Words can not express how grateful I am to my loveliest mother and dearest father for all of the sacrifices that they have made on my behalf. I owe them everything and I hope that this work makes them proud.

Finally, I would like to thank Ministry of Higher Education, Malaysia for the financial support and assistance of the entire period of my PhD.

I dedicate this thesis to my beloved husband for their constant support and unconditional love.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ACRONYMS

ACT	Application Computation Time.
API	Application Programming Interface.
ART	Application Response Time.
AT	Arbitrator Time.
AWS	Amazon web Service.
CDMA	Code Division Multiple Access.
CE	Consumed Energy.
CGR	Coarse-Grained Resource.
CL	Communication Latency.
CMA	Cloud-based Mobile Application.
CMH	Cloud-Mobile Hybrid.
COE	Computing Outsourcing Engine.
CPU	Central Processing Unit.
EC	Energy Consumption.
EC2	Elastic Compute Cloud.
ECU	Elastic Compute Units.
F	Frequency.
FGR	Fine-Grained Resource.
GAE	Google App Engine.
GPU	Graphical Processing Unit.
GQL	Google Query Language.
H2MCO	Horizontally Heterogeneous Mobile Computation Outsourcing.
HCR	Hybrid Cloud Resource.
HEC	Hybrid Energy Consumption.
HRTT	Hybrid Round-Trip Time.
HTTP	Hypertext Transfer Protocol.
I/O	Input/Output.
IP	Internet Protocol.
LRTT	Local Round-Trip Time.
MCC	Mobile Cloud Computing.
MCO	Mobile Computation Outsourcing.
MGR	Medium-Grained Resource.
MNO	Mobile Network Operators.
OBR	Object Request Broker.
OS	Operating System.
OVF	Open Virtualization Format.
PDA	Personal Digital Assistants.
QoS	Quality of Service.
RAM	Random Access Memory.
RAT	Radio Access Technology.
RET	Remote Execution Time.
RMA	Resource-intensive Mobile Application.
ROA	Resource-Oriented Architecture.
RSS	Received Signal Strength.
RTT	Round-Trip Time.

S3	Simple Storage Service.
SLA	Service-Level Agreement.
SOA	Service-Oriented Architecture.
SOAP	Simple Object Access Protocol.
SQL	Structured Query Language.
TEC	Total Energy Consumed.
UDDI	Universal Description Discovery and Integrity.
URI	Unified Resource Identifier.
VHMCO	Vertically Heterogeneous Mobile Computation Outsourcing.
VM	Virtual Machine.
VPU	Virtual Processing Unit.
WAN	Wide Area Network.
WCDMA	Wideband Code Division Multiple Access.
WCI	Wired Communication Interface.
WiFi	Wireless Fidelity.
WLAN	Wireless Local Area Network.
WLCI	Wireless Communication Interface.

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

This chapter introduces a holistic view of the research undertaken in this thesis. We present motivation for the research on Mobile Computation Outsourcing (MCO) in Mobile Cloud Computing (MCC) and state the research problem. Moreover, the chapter specifies the aim and objectives of this study and describes the methodology proposed to achieve the aim and objectives.

The remainder of this chapter is as follows. In Section 1.1, we present motivation for undertaking this research and highlight the significance of the work. Section 1.2 introduces the identified research problem to be addressed in this thesis. Section 1.3 presents the aim and objectives of this study following with proposed methodology in section 1.4. Finally, we present the thesis layout in section 1.5.

1.1 Motivation

In the recent years, explosive growth of heterogeneous mobile and distributed computing, wireless networking, and web technologies have significantly advanced ubiquitous computing for mobile users. Mobile devices such as smartphones, tablets, and Personal Digital Assistants (PDA) have become the essential part of modern life. Among them, smartphones possess a likelihood of one-upmanship due to their miniature's nature and remarkable features, particularly, telephony, perception, multimedia, and geolocation services (Albanesius, 2011). According to the Gartner, smartphones have been leading the mobile device market share by more than 55% overall sale in last quarter of 2013 (Gartner, 2013). Smartphones have become enabler technology to serve mankind in several critical

areas, particularly healthcare, education, tele-monitoring, urban management, and disaster recovery. These smart handheld computers are expected to generate 150.6 billion dollars benefits by the end of 2014, while devices like PDAs will bring in only 2.7 billion dollars comparatively (*"Global Markets For Smartphones and PDAs"*, 2009).

However, smartphones capabilities are encumbered by their intrinsic limitations, particularly constraint battery, Central Processing Unit (CPU), and memory resources. Hence, they fail to fulfil insatiable computing requirements of mobile users. Though, mobile hardware technology leaders have gained remarkable achievements in developing high-end, long-lasting CPU, Random Access Memory (RAM), and battery, but they fail to meet ever-increasing user computing demands. Therefore, software approaches to outsourcing resource-intensive computation and augmenting mobile devices' capabilities becomes inevitable.

M. Satyanarayanan (Satyanarayanan, 2001) introduces the *cyber foraging* approach to empower computation capabilities of mobile devices by offloading the entire or part of application to the remote computing resources. In this approach, mobile applications are migrated, partially or entirely, to a nearby resource-rich wall-connected non-mobile free device, called surrogate. Surrogates as a normal desktops or computing devices perform intensive computation on behalf of the resource-constraint mobile devices and return the results back to the mobile. However, several Quality of Service (QoS) issues, especially data safety, user security, reliability, availability, and scalability of surrogates inhibit their adoption (Sharifi, Kafaie, & Kashefi, 2011).

Advancements of distributed and high performance computing have bred a novel utility-based computing technology, called cloud computing that is embraced by the academic and industrial communities. The cloud is "a type of parallel and distributed system consisting of a collection of interconnected and virtualized computers dynamically provi-

sioned and presented as one or more unified computing resources based on service-level agreements established through negotiation between the service provider and consumers" (Buyya, Yeo, Venugopal, Broberg, & Brandic, 2009). These cloud resources can provide elastic, scalable, and rich computation and storage platform for resource-intensive processing (Mell & Grance, 2011; Armbrust et al., 2010) to reduce operational and maintenance costs while increase operational safety.

Exploitation of cloud resources for augmenting mobile devices has emerged a new research area called Mobile Cloud Computing (MCC). MCC inherits mobility of mobile computing and rich resources of cloud computing and is capable of providing resource-intensive computation and huge information at user fingertips anywhere, anytime via on-demand, elastic, scalable computing infrastructure (e.g., Amazon and Google). The MCC vision is realized by performing resource-intensive components of mobile applications outside the mobile device inside the remote cloud-based resource (Abolfazli, Sanaei, Gani, & Buyya, 2014) that is referred to as computation outsourcing or outsourcing in brief.

However, despite rich cloud-based computing resources, outsourcing performance is affected by several factors, particularly resource heterogeneity and granularity. Cloud computing incorporates highly heterogeneous computing infrastructures -as resources- with dissimilar capabilities scattered around the globe. These heterogeneous resources have three main granularity levels of coarse, fine, and medium. Coarse-Grained Resources (CGRs) are highly scalable and elastic to performing computation (i.e., high scalability), but are located in distance (i.e., low proximity) and exist in very few geographical regions only (i.e., low multiplicity) breeding communication latency.

In contrast, Fine-Grained Resources (FGRs) feature limited computing capabilities (i.e., low scalability), are located near the mobile end-users (i.e., high locality), and are very large in number (i.e., high multiplicity) leading to computation latency. Medium–

Grained Resources (MGRs) have medium capabilities. They have medium computing power (i.e., medium scalability), are located in less proximity to users (i.e., locality), and are moderately distributed worldwide (i.e., medium multiplicity) compared to coarse- and fine-grained resources ending to both communication and computing latencies. Hence, these heterogeneous granular resources feature varied computing and communication latencies.

Although performance gain of utilizing heterogeneous computing resources instead of homogeneous resources is verified (Rosenberg & Chiang, 2010; M. Guevara, 2013), leveraging heterogeneous cloud-based resources has not become the dominant approach in MCC yet. Such performance gain can remarkably improve user perceived interaction experience from compute-intensive mobile applications (Huang et al., 2010) due to high impact on energy consumption and response time of resource-intensive MCC applications. However, majority of MCC solutions such as (Satyanarayanan, Bahl, Caceres, & Davies, 2009; Cuervo et al., 2010; B. Chun, Ihm, Maniatis, Naik, & Patti, 2011) leverage homogeneous type of resources which are dominantly coarse-grained resources. The results of utilizing single type of cloud resource (coarse-, fine-, or medium-grained) is that they either originate computing latency due to low computing power or communication latency because of long distance between mobile and cloud resources, or in the third case both computing and communication latencies.

Therefore, it is essential to study the impact of utilizing heterogeneous cloud-based resources for computation outsourcing in MCC, leverage hybrid resources, and propose a lightweight computation outsourcing solution with convergence of heterogeneous resources. This lightweight solution deemed could improve application execution time and energy efficiency by benefiting from low computation and communication latency of converged heterogeneous granular resources.

1.2 Statement of Problem

The problem to address in this research is stemmed from varied scalability, locality, and multiplicity characteristics of three classes (i.e., coarse, fine, and medium-grained) of heterogeneous granular cloud-based resources in computation outsourcing. Leveraging individual class of heterogeneous granular resources as remote computational resources for mobile computation outsourcing with varied impact on response time and energy consumption of compute-intensive mobile applications degrades efficiency of computation outsourcing performance. Therefore, gaining an insight into the cause of the problem demands investigation on heterogeneous granular cloud resources that used in mobile computation outsourcing. It also demands analysis of heterogeneity genealogy in MCC. Before stating our research problem, we outline the impacts of three classes of heterogeneous granular cloud resources on energy efficiency and response time of compute-intensive mobile applications as following.

Coarse-grained cloud resources are any resource-rich computing units that are characterized with high scalability and low multiplicity that are located in long distance with mobile end-users. Figure 1.1 illustrates the schematic view of the typical coarse-grained resources. Distant giant clouds (e.g., Amazon[1] and Google Cloud Platform [2]) that feature rich resources and high scalability are standing in this class. Utilizing these resources in computing outsourcing originates long Wide Area Network (WAN) latency due to their long distance and hence, degrades mobile application performance. It is known that "WAN delays in the critical path of user interaction can hurt usability by degrading the crispness of system response" (Satyanarayanan et al., 2009). Response time is considered crisp when it is less than 150 ms and is unacceptable when more than 2 seconds (Tolia, An-

[1] http://aws.amazon.com/

[2] https://cloud.google.com

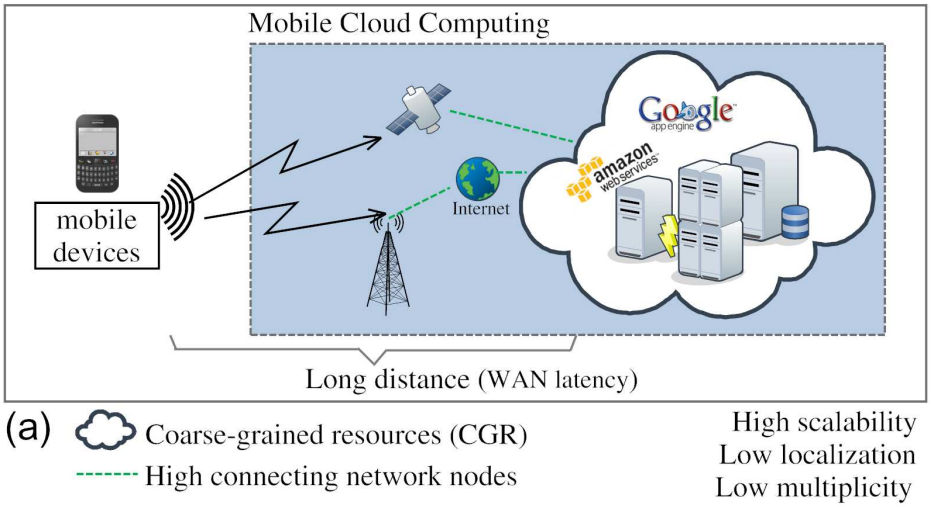


Figure 1.1: Graphical illustration of the coarse-grained resources

dersen, & Satyanarayanan, 2006). This is noticeable that acceptable latency is highly depending on the types of applications and users' demand in the network. For instance, for the 3D graphical applications, the network WAN latency should be less than 100 ms (*VMware View 5 with PCoIP Network Optimization Guide*, n.d.), while it is not acceptable for the users using photo editor applications (Satyanarayanan et al., 2009). Therefore, the ultimate goal in this domain is to lower WAN latency and reach crisp response time.

Moreover, accessing these coarse-grained resources often entertains passing through very large number of intermediate hops and is usually associated with varies cellular communications. Large number of intermediate hops originates noticeable delay in round-trip communication between mobile and cloud resources. Also, though cellular wireless connection supports a wide area connectivity and ubiquitous computing, but slow data transfer and long delay increase response time of application and impose negative impact on mobile user experience. Cellular radio consumes more battery compared with Wireless Fidelity (WiFi) radio (Perrucci, Fitzek, & Widmer, 2011). Therefore, leveraging coarse-grained resources causes communication latency that leads to increase in round-trip time and energy consumption of Cloud-based Mobile Application (CMA).

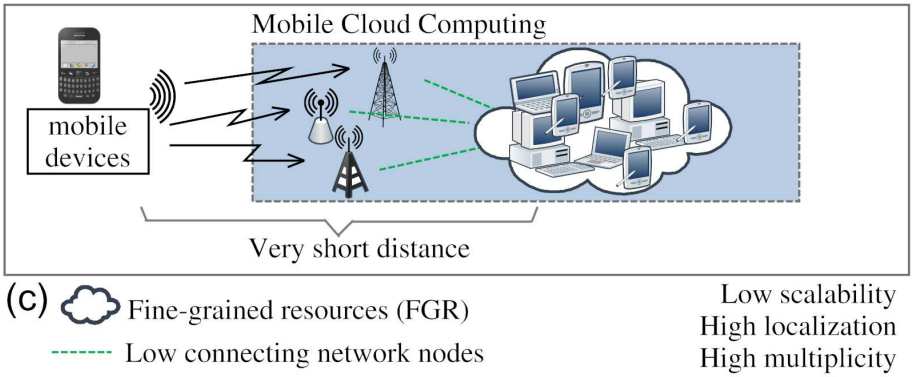


Figure 1.2: Graphical illustration of the fine-grained resources

Fine-grained cloud resources enfold cloud-based resources in close proximity providing cloud computing in mobile users' locality with low computing capabilities. Illustrative view of the fine-grained resources such as smartphones, tablets, notebooks, and personal computers is presented in Figure 1.2. The fine-grained resources are highly numerous which indicates their high multiplicity. Although these type of computing devices are placing close to the users, their resource-scarcity inhibit elastic scalability and availability of computing resources. This leads to the resource-intensive application performance degradation due to their high computing latency. Hence, they cannot individually play the role of a high performance computing server like coarse-grained resources to perform resource-intensive (e.g., CPU-, graphic-, and memory- intensive) tasks. However, fine-grained resources can be accessed via short-hop connection instead of many-hop connection that can noticeably conserve energy and enhance the performance by decreasing battery power consumption.

Medium-grained cloud resources are located in medium proximity to cloud-consumers, feature medium computing power, and are more numerous than coarse-grained resources. Figure 1.3 shows graphical representation of the medium-grained resources in MCC. The main advantage of medium-grained resources compare to coarse-grained resources is their better proximity to the users, and the major advantage of them to fine-grained resources

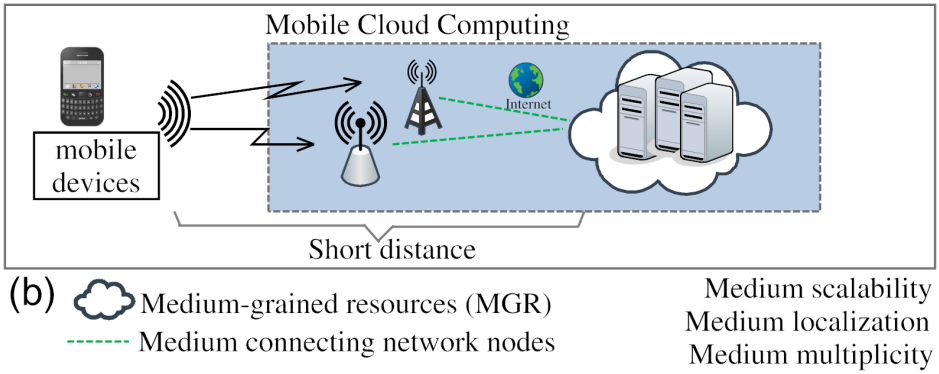


Figure 1.3: Graphical illustration of the medium-grained resources

is their higher computing power. Medium-grained resource can be any computing device with medium computing power and proximity level to mobile users. For example, a medium instance of Amazon Elastic Compute Cloud (EC2) can be called medium-grained resource when is located in medium proximity level (city level) to users. Performing resource-intensive tasks in proximate cloud-based resources enhances application performance, but fails to deliver on-demand scalability and service availability. This is intensified by growth in the number of cloud consumers that leads to the system performance degradation due to lack of scalable, rich computing resources like giant clouds. Thus, this kind of heterogeneous cloud resources fails to provide ever increasing scalability requirements of CMAs.

Deficiencies and shortcomings of each class of heterogeneous granular resources inhibit efficient leveraging of vertical heterogeneous granular cloud resources in mobile outsourcing. Vertical heterogeneity raises within one type of granular resources, like coarse or fine and creates a bottleneck of limited computing and communication capabilities which results in increased response time and energy consumption. In contrast, horizontal heterogeneity happens across varied types of granular resources, like coarse and fine. Horizontal heterogeneity can increase performance gain of mobile computation outsourcing by allowing computation-communication trade-off.

Therefore, considering the limitations of vertically heterogeneous resources, we can present the identified research problem as follows.

Coarse-grained cloud resources feature high scalability and low locality that originates communication latency, fine-grained resources offer low scalability and high locality that leads to computation latency, and medium-grained resources provide medium scalability and locality breeding communication and computing latency. Such communication and computation latencies negatively impact on energy efficiency and response time of compute-intensive mobile applications leading to mobile application performance degradation.

Leveraging horizontally heterogeneous granular resources creates opportunity to perform computing-communication trade-off and improve CMA execution performance. Therefore, to advance mobile computation outsourcing in MCC there is a crucial necessity to a lightweight computation outsourcing framework built on a horizontally heterogeneous granular resource layer that can fulfill insatiable computing resources for optimal CMAs execution. Such horizontally heterogeneous hybrid resource aims to accumulate strengths and benefits of each type of granular resources and develop a multi-layered cloud platform for mobile devices which still is lacking.

1.3 Statement of Objectives

This research is undertaken with the aim to achieve efficient computation outsourcing for compute-intensive mobile applications using horizontally heterogeneous granular cloud-based resource. The aim is achieved by fulfilling the following objectives:

- To study the recent cloud-based mobile computation outsourcing approaches and to gain an insightful understanding of heterogeneity and granularity in MCC that helps us to identify the current problems in computation outsourcing domain of MCC.

- To analyse the identified problem caused by computation and communication latencies and unveiling the impact of leveraging heterogeneous granular resources on Round-Trip Time (RTT) and Energy Consumption (EC) of compute-intensive mobile applications.

- To propose a lightweight horizontally heterogeneous hybrid MCC framework for compute-intensive mobile applications to achieve efficient computation outsourcing by performing trade-off between computation and communication latency that leads to efficient RTT and EC in CMAs.

- To evaluate and validate performance of our proposed lightweight horizontally heterogeneous hybrid MCC framework considering two performance metrics of round-trip time (RTT) and energy consumption (EC) of mobile applications.

1.4 Proposed Methodology

We use the following steps in order to achieve the aim and objectives of this study.

- Comprehensive review and synthesis of the recent mobile computation outsourcing efforts in MCC are undertaken to identify the impact of exploiting heterogeneous granular cloud-based resources on CMAs' performance referring to scholarly digital libraries, particularly IEEE, ScienceDirect, and Web of Science. We also examine the impacts of heterogeneity in MCC from two different dimensions and identify several research issues through literature. We identify the most significant research problems to address in this research.

- We investigate the identified problem and verify its significance through experimental analysis in real MCC environment using android-based smartphone and Amazon EC2 cloud Virtual Machines. Using series of benchmarking experiments on local

mobile device and real cloud computing testbeds, we evaluate the performance of executing compute-intensive applications in MCC to verify the severity of the identified research problem.

- To alleviate the identified problem, we implement and design a lightweight horizontally heterogeneous hybrid MCC framework for efficient outsourcing of compute-intensive applications. The proposed framework is composed of three layers of heterogeneous granular resources each with different scalability, locality, and multiplicity degree. To achieve efficient execution of CMAs, we design our horizontally heterogeneous hybrid cloud in convergence of fine-grained (i.e., Mobile Network Operators (MNO) dealers), medium-grained (i.e., MNOs), and coarse-grained (i.e., distant giant clouds like Amazon EC2) resources to perform computing-communication trade-off for efficient outsourcing. Beside, we use Resource-Oriented Architecture (ROA) design philosophy in design and development of the framework to mitigate the system complexity and management overhead. ROA as a well-known design philosophy independent from specific technology, vendor, and business policy is deployed to incorporate different prefabricated services towards conveniently generating complex applications and services. Thus, lightweight feature of the framework is achievable in the presence of ROA design philosophy and horizontally heterogeneous hybrid MCC resources.

- We evaluate performance of our proposed framework via benchmarking experiments. A complex prototype application, including three heavy operations, namely power, prime, and factorial is used in this benchmarking experiment. The performance evaluation testbed is built using real android-based smartphones and cloud-based resources in wireless environment. Application round-trip time and consumed

energy are opted as two performance metrics in this experiment. We used 30 different workloads with three intensity levels of low, medium, and high to carry out our performance evaluation in the real MCC environment. The results of performance evaluation are validated using statistical modelling. The statistical model is devised using regression analysis as a predominant observation-based analysis and modelling method. The dataset is created using independent replication method on the real time environment to be used for training the regression model. We validate the devised models based on the split-sample validation approach. The valid statistical models are used to validate the results of our performance evaluation results.

1.5 Thesis Layout

The remainder of this thesis are organized as follows and represented in Figure 1.4.

- Chapter 2 aims to review the research undertaken in the field of mobile computational outsourcing. The chapter provides knowledge of MCC environment, reviews mobile computation outsourcing approaches to identify and classify granular resources and determine the deficiencies of current solutions. We investigates heterogeneity and granularity features of the MCC to gain insight into the benefits and challenges. The taxonomy of heterogeneity beside heterogeneity handling techniques are presented in this chapter. Heterogeneous granular mobile computation outsourcing approaches are presented and several research problems are identified as future research directions.

- In Chapter 3, we investigate and analyse the impacts of heterogeneous granular cloud-based resources in performance of Resource-intensive Mobile Application (RMA). Using series of experiments on android-based mobile device and Amazon cloud virtual machines, we identify the impact of scalability, locality, and multi-

plicity on performance of mobile applications. We verify the research problem and demonstrate its significance.

- In Chapter 4, we propose a lightweight horizontally heterogeneous MCC framework in convergence of coarse-, medium-, and fine-grained cloud-based resources for this research. The schematic presentation of the framework is demonstrated and the functional and non-functional properties of the main system components are explained. Significance of the proposed framework is highlighted and performance evaluation setup is described.

- Chapter 5 presents the performance evaluation methodology. We describe two evaluation methods, namely benchmarking and statistical modelling that have been used to evaluate and validate the performance of the proposed framework. The benchmarking model is explained and the methodology to devise statistical model is described. The method to validate the statistical models is also described.

- In Chapter 6, we present the results of our performance evaluation and discuss the findings from two main perspectives of application round-trip time and energy efficiency. We compare and contrast the benchmarking results with the results of statistical modelling (which is validated) to validate the performance of the proposed framework.

- Finally, Chapter 7 concludes the thesis by describing how the aim and objectives of the research are fulfilled. The main contributions are summarized and significance of the research and the framework proposed in this research are highlighted. We list the publications including conference and journal articles that are produced from the research undertaken in this work. The limitation and future works conclude the chapter.

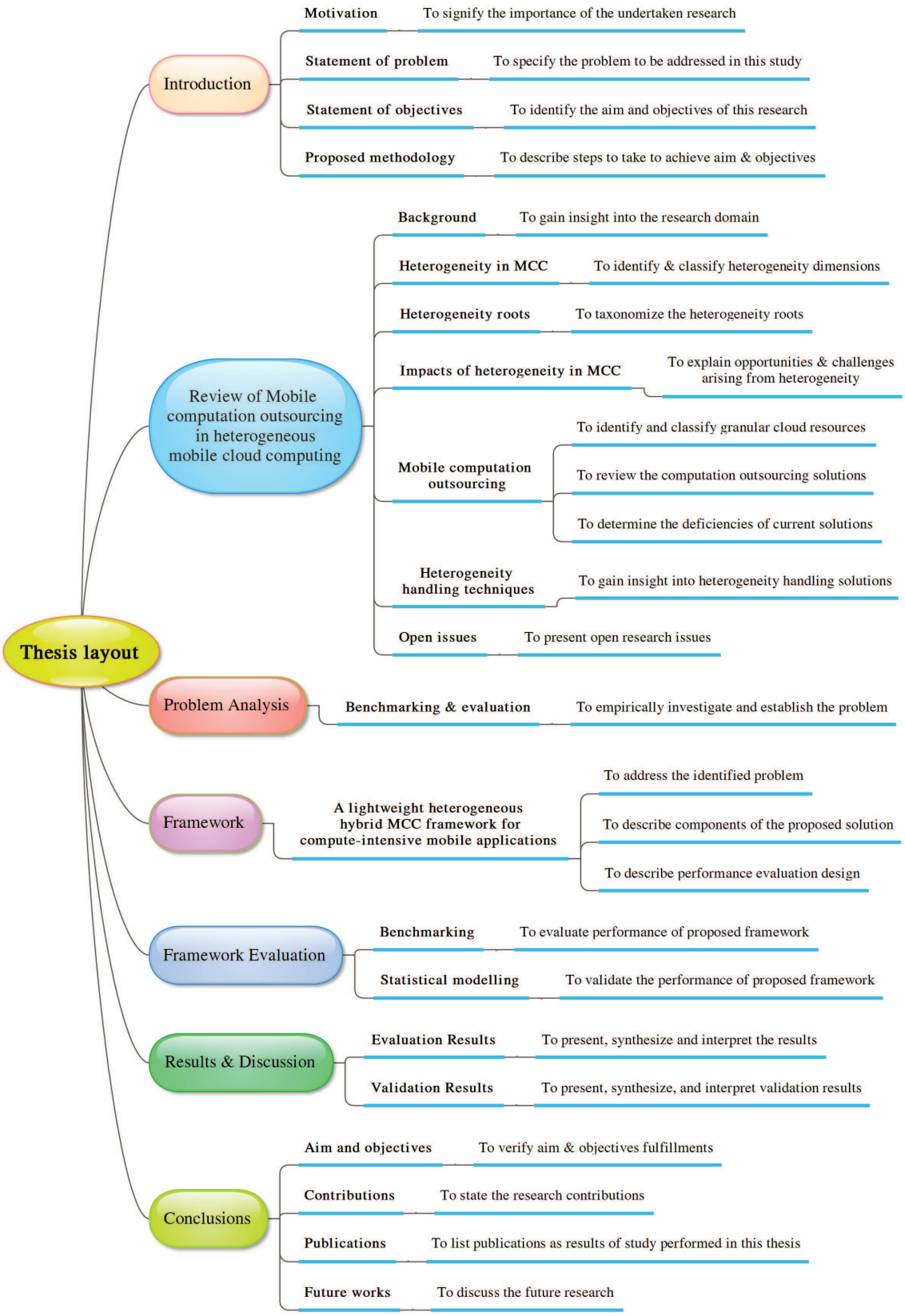


Figure 1.4: Schematic presentation of the thesis layout

CHAPTER 2

A REVIEW ON MOBILE COMPUTATION OUTSOURCING IN MOBILE CLOUD COMPUTING: HETEROGENEITY AND GRANULARITY

This chapter reviews the Mobile Cloud Computing (MCC) on recent mobile computation outsourcing efforts from heterogeneity and granularity perspectives to devise taxonomies. The MCC domain is reviewed from heterogeneity point of view to gain insight into the problems that diversity and inhomogeneity of mobile devices, cloud computing resources, and communication networks impose on computation outsourcing. Taxonomy of heterogeneity dimensions in MCC is presented and roots of heterogeneity are identified. The problems that are stemmed from heterogeneity are highlighted and the efforts to alleviate the problems are critically reviewed. Four varied MCC architectures are taxonomized and described from heterogeneity point of view. Several heterogeneity handling techniques and open research challenges in MCC, including computation and communication latencies of heterogeneous granular resources are identified.

Section 2.1 presents a brief description on mobile computing, cloud computing, and mobile cloud computing. Section 2.2 presents MCC definition and dimension and the taxonomy of heterogeneity in MCC is presented in Section 2.3. Impacts of heterogeneity in MCC are investigated in Section 2.4 and several heterogeneity handling approaches are reviewed in Section 2.6. Several open research challenges are highlighted in Section 2.7. The chapter is concluded in Section 2.8.

2.1 Background

In this section we present a brief introduction on mobile computing, cloud computing, and MCC efforts.

2.1.1 Cloud Computing

Cloud computing is a new era of computing after grid computing which is directed to deliver varied services over the internet. The cloud services including infrastructure, platform and software are provided by several cloud providers. Each service has the ability to provision scalability according to different demands, to run complex and heavy functionality for users regardless of dealing with underlying technologies, to leverage vast verity of physical infrastructure for using pool of rich resources by help of virtualization technology, and pay as you use principle.

Cloud computing is the state of the art technology to deliver a scalable, reliable, secure and sustainable varied infrastructure for hosting various Web-based applications services. Infrastructure as a Services (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) are those services that are offered by cloud providers (Buyya et al., 2009; Mell & Grance, 2011; Armbrust et al., 2010). To benefit from economies these services stated as a computing utility (Armbrust et al., 2009) like traditional utility services e.g. gas, water and electricity. Clouds mainly aim to increase the power of data warehouses which scattered as a pool of resources, and to provide a virtual services like hardware, database and user interface by leveraging virtualization technology (Barham et al., 2003). Therefore, applications can take the benefit of parallel and distributed algorithms supported by cloud servers and run in an isolated environment in top of virtualization layer independent from beneath layers' architecture.

Cloud users also may cross the channel of internet and use hardware, platform and software services on requirement which is provided as a subscription-based in model of pay as the customer use service. Hence, in the cloud computing ecosystem users have the ability to access and deploy the application and data anywhere with availability of synchronization across several devices that use and pay the costs depending on Quality of

Service (QoS) metrics that required (Buyya, Chee Shin, & Venugopal, n.d.). Developers for developing new idea that necessitates specific hardware with high processor, memory, storage, and software no longer need to spend time and cost for making a large infrastructure to deploy their services or user investment to operate it (Armbrust et al., 2009). For example Amazon Elastic Compute Cloud (EC2) (*Amazon Elastic Compute Cloud (EC2)*, n.d.) is one of the well-known cloud computing platforms that offers distributed IaaS for users to run their varied Operating Systems (OS), whereas Amazon S3 (*Amazon Web Services,Amazon Simple Storage Service (Amazon S3)*, n.d.) supports storage and provides highly scalable, secure, fast and reliable. Eucalyptus is a software platform which empowers cloud users to have a private cloud infrastructure as a service. Eucalyptus as an open source cloud computing platform, compatible with Amazon web Service (AWS) can create scalable cloud resources for computing (like EC2) , network and storage (like Simple Storage Service (S3)) for images and user data (Milojicic & Wolski, 2011; *Eucalyptus Systems, Inc.,Eucalyptus Cloud Computing Software*, 2011).

Google App Engine (GAE) (*Google, Inc.,Google App Engine*, n.d.) also as a famous PaaS provides hosting platform and allows clients to deploy and run their specific web applications based on what GAE can support (currently Python and Java are two programming language is supported by GAE). Facebook and DropBox (*DropBox*, n.d.) are illustrious web-based cloud-based applications with “web 2.0” as the key technology behind the realization of software as services.

In summary, cloud computing is a high performance computing technology which significantly reduces costs of ownership by eliminating necessities for maintenance of large-scale parallel and distributed servers including their power consumption and cooling systems. While the cloud landscape has a vast opportunity to implicate different computing devices and infrastructure, it brings a heterogeneous environment featuring varied

programming languages, hardware, architecture, and business policies. Most of cloud systems have their own proprietary policies with invisibility of infrastructure for researchers. Also cloud services delivered by cloud service providers are completely and tightly integrated with the underlying platforms. For instance, Google Application Engine and the Apple Cloud (iCloud) support their own mobile platforms, respectively Android and iOS. This phenomenon manifests as fragmentation and portability issues due to heterogeneity in MCC.

2.1.2 Mobile computing

Advance technologies in mobile device have conveyed a vast variety of hard-ware, software with data transmission technologies that have led to proliferations of mobile devices, especially smartphones whereas they surpass notebook and desktop PCs (Albanesius, 2011; *Smartphones Will Surpass PC Shipments In Two Years*, 2010). Improvements in CPU, memory, power consumption, sensors, size and quality of screen on one hand and growth in mobile applications because of development in mobile OSs and wireless technologies e.g. 3G and 4G that provide higher rates of data transmission on the other hand are contributed toward delivering enhanced services to mobile users on the go.

Despite such advancement and release of modern mobile platforms such as Android and iPhone, still current smartphones have intrinsic limitations on processing power, battery lifetime, storage capacity and display size due to miniature, lightness, and mobility. Hence, complex functionality, heavy processing and storing huge amount of data on mobile devices are non-trivial. Mobile devices with high quality of digital camera to capture a photo and record video are suffering from low memory and battery life time, and also developers do not have the opportunity to fulfill user requirements and engage them with rich mobile applications. Therefore, computation outsourcing through cloud-based resources

is emerging to realize the vision of mobile computing.

2.1.3 Mobile Cloud Computing (MCC)

Mobile Cloud Computing is a rich mobile computing technology that leverages unified elastic resources of varied clouds and network technologies toward unrestricted functionality, storage, and mobility. It serves a multitude of mobile devices anywhere, anytime through the channel of Ethernet or Internet regardless of heterogeneous environments and platforms based on the pay-as-you-use principle. MCC is an amalgam of three heterogeneous foundations, namely cloud computing, mobile computing, and networking.

The state-of-the-art mobile cloud computing (MCC) paradigm has gained a momentous ground to mitigate mobile devices' shortcomings (i.e., computing and energy) by outsourcing resource-intensive mobile tasks to the cloud.

Vision: MCC is the state-of-the-art mobile computing technology that aims to augment a multitude of mobile devices, especially smartphones and alleviate their resource poverty. This futuristic accomplishment will be employed in several areas like healthcare (e.g. telemonitoring and telesurgery), education, IT Business, rural and urban development, and social networking. Technological advancement in manufacturing high-end mobile resources is slower than the ever-growing expectations of mobile users and application requirements. Hence, soft resource augmentation is necessary for delivering user-centric computing capabilities (Satyanarayanan, 2001) equal to user expectations. We advocated that cloud computing is the predominant technology recently deployed to augment smartphones by reducing application resource requirements. Several efforts such as (Cuervo et al., 2010; X. W. Zhang, Kunjithapatham, Jeong, & Gibbs, 2011; B.-G. Chun & Maniatis, 2009; March et al., 2011; Lu, Li, & Shen, 2011; Kemp, Palmer, Kielmann, & Bal, 2010) deploy cloud computing technology to enhance the capability of smartphone.

Moreover, cloud computing is beneficial in enhancing information safety and security. Storing data in smartphones' local storage is a hazardous practice due to their susceptibility to theft, loss, and physical damage. Cloud data storage is envisioned to enhance data safety and security, provide pervasive accessibility, and facilitate data portability and synchronization among several devices (e.g. smartphones and PCs). DropBox[1], SugarSync[2], and Box[3] are examples of cloud storage services. People exploit such huge data warehouses to store and retrieve their data (bulk data) which are accessible from various devices. Users can even access their data through the Internet by utilizing public devices and providing unique credentials.

The advent of MCC has advanced into the technological revolution providing profitable opportunities for several domains such as healthcare, e-learning, and the tourism industry. It connotes the impression to reduce development cost and stimulate execution of resource-intensive mobile applications by leveraging distant rich resources to enhance the quality of user experience.

Figure 2.1 illustrates a conceptual view of MCC and depicts its usability in several domains such as healthcare, social networking, urban development, and vehicular technology. It shows the possibility of utilizing geographically distributed private clouds (e.g. medical and biological research groups), public clouds (e.g. Google[4] and Facebook[5]), and hybrid clouds (that can be generated by converging private and public clouds) in global roaming. Furthermore, Figure 2.1 illustrates heterogeneity in MCC among varied mobile devices, communication networks, and clouds. Although heterogeneity has been existing

[1] https://www.dropbox.com/

[2] http://www.sugarsync.com/

[3] http://www.box.com/

[4] https://www.google.com

[5] http://www.facebook.com

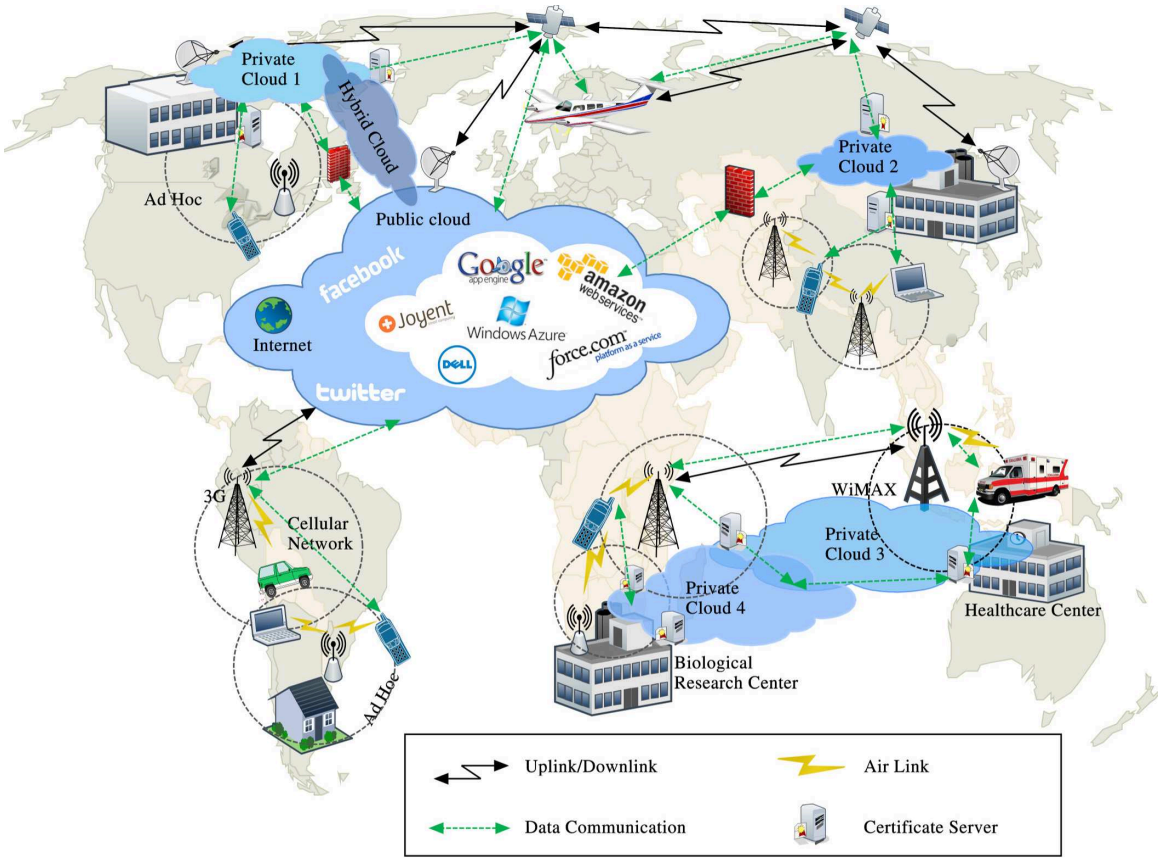


Figure 2.1: A conceptual view of mobile cloud computing.

in mobile and cloud computing domains, accumulated intensity and complexity make it a unique, challenging feature within MCC that necessitates advance study.

MCC is a heterogeneous environment consists of three heterogeneous sub-domains of mobile computing, cloud computing, and communication networks (to augment smartphones). This amalgam of heterogeneities deems could create benefits and challenges. In next section, we investigate MCC from heterogeneity perspective to better gain an insight into heterogeneity aspect of MCC and identify possible benefit and challenges that are either stemmed or intensified by heterogeneity.

2.2 Heterogeneity in MCC

MCC consists of three heterogeneous various components (i.e., mobile devices, clouds, and wireless networks) which these components should seamlessly cooperate and commu-

nicate with each other to benefit mankind. In this section, we present a brief overview on MCC heterogeneity and classify it into two dimensions, namely vertical and horizontal, based on type of heterogeneity risen in mobile devices, cloud, and wireless networks.

2.2.1 Definition

Heterogeneity in MCC is the existence of differentiated hardware, architectures, infrastructure, and technologies of mobile devices, clouds, and wireless networks. The cutting edge technologies are expected to initiate and facilitate collaboration among these heterogeneous computing devices toward unrestricted mobile computing.

Heterogeneity in Mobile Devices: Software, hardware, and technology variation among mobile devices cause heterogeneity in this domain. Moreover, increasing popularity of smartphones creates a dynamic and demanding market that disperse them to different dimensions, e.g. brand, hardware, OS, feature, and communication medium. Consequently, device-level collaboration becomes more challenging in MCC.

Heterogeneity in Clouds: Numerous cloud vendors provide different services with custom-built policies, infrastructures, platforms, and APIs that make the cloud landscape heterogeneous. Such variations cause interoperability and portability (Hogan, Liu, Sokol, & Tong, 2011) as major challenges in cloud computing. There is a notion (D.Durkee, 2010) that business competition also diversifies cloud providers with their heterogeneous frameworks, exacerbating heterogeneity on the cloud side.

Heterogeneity in Wireless Networks: In MCC, the majority of communications take place in the wireless network environment which is a heterogeneous communication medium. Variations in wireless networks and their related technologies impact the delivery of cloud services and affect mobility, augmentation, and usability of smartphones.

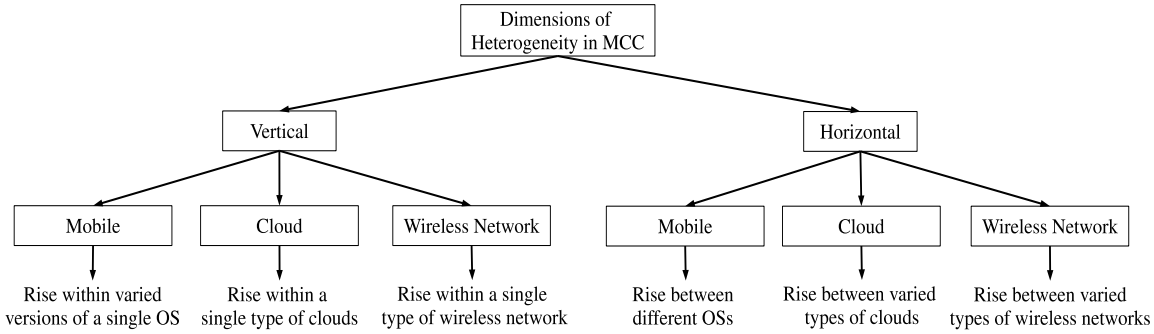


Figure 2.2: Dimensions of Heterogeneity in MCC.

2.2.2 Dimensions

Heterogeneity in MCC is classified into two categories, namely vertical and horizontal based on the variations in mobile, cloud, and network environments. The proposed heterogeneity model in Figure 2.2 depicts how three underlying components of MCC are influenced by two types of heterogeneity. Figure 2.3 shows three examples of vertical and horizontal heterogeneity in MCC.

Vertical Heterogeneity: When differentiation is within a single type of mobile OS, cloud service, or wireless network it is named vertical heterogeneity.

- *Mobile Devices:* Among mobile devices, vertical heterogeneity appears within a similar family of products. Different flavors of the OSs offer some unique features and services that are not compatible with other versions. The vertical oval shape in Figure 2.3(a) shows vertical heterogeneity within different flavors of Android OS. Android 4.0.3 offers social, calendar, and visual voice mail APIs which are totally new compared to Android 3.x (*Android API Levels*, n.d.). Similarly, in various BlackBerry mobile products, different features and hardware specifications are deployed that prevent the application portability among devices from the same manufacturer. Therefore, the application developed for one OS and deployed in one specific product cannot be portable to the same family of products with different

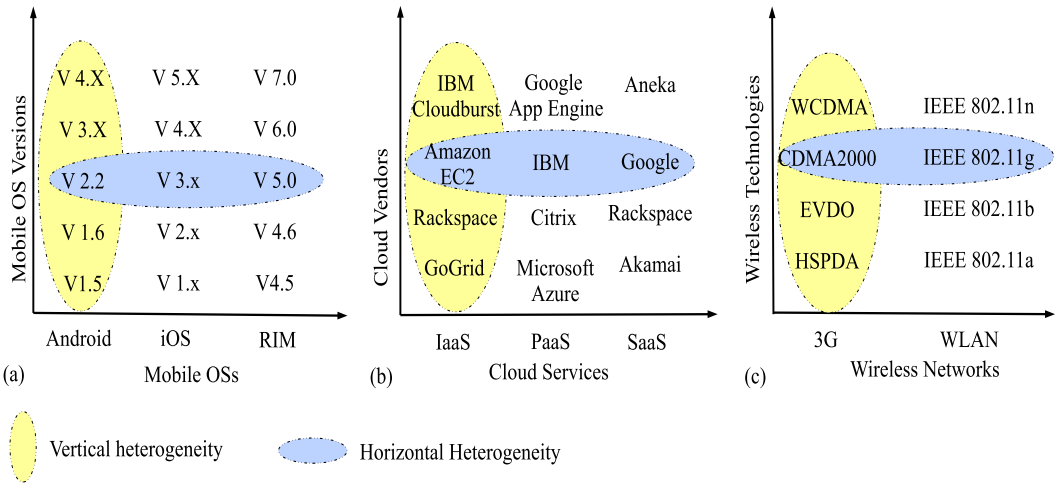


Figure 2.3: Vertical and horizontal heterogeneity in three dimensions within MCC: (a) mobile OSs and their versions, (b) cloud services and vendors, and (c) wireless networks and related technologies.

versions of the same OS.

- *Clouds:* In the cloud, vertical heterogeneity occurs within a single type of clouds that provides similar services, e.g. IaaS (Infrastructure as a Service) or PaaS (Platform as a Service). The vertical oval shape in Figure 2.3(b) shows vertical heterogeneity within various IaaS service vendors. Though Amazon EC2 and Rackspace are IaaS clouds, they are built on different pillars: internal infrastructures, technologies, and business policies. Therefore, demand for switching between these two cloud services incurs redundant cost, even though both vendors provide IaaS. It also creates data and application portability issues and hinders easy code and data migration within a single type of clouds. Cloud users are forced to adhere to specific cloud service provider(s) (Sheth & Ranabahu, 2010). However, standardization efforts like the Open Virtualization Format (OVF) (*"Open Virtualization Format Specification (OVF)"*, 2009) are emerging to alleviate problems and facilitate the deployment of virtual appliances in various clouds.

- *Wireless Networks:* Among different wireless technologies, horizontal handoff is a

well-known phenomenon caused due to vertical heterogeneity in mobile wireless environments including MCC. In MCC, horizontal handoff happens in the situations when a cloud-mobile user is moving across heterogeneous access points within a single type of wireless network to access cloud services. For instance, this happens when cloud-mobile user is moving between IEEE 802.11a and IEEE 802.11g Wireless Local Area Network (WLAN), or between Code Division Multiple Access (CDMA) 2000 and Wideband Code Division Multiple Access (WCDMA) 3G technologies. In majority of offloading algorithms in MCC, the network characteristics highly influence on offloading decision (Sharifi et al., 2011). Hence, any change in networking technologies directly impacts on efficiency and effectiveness of the offloading decision and overall process. The vertical oval shape in Figure 2.3(c) shows vertical heterogeneity within various cellular technologies. Data transmission in cognitive wireless access networks (Demestichas, Stavroulaki, Boscovic, Lee, & Strassner, 2006) which is configured with a set of different Radio Access Technologys (RATs) and Frequencys (Fs) is an illustrious example of vertical heterogeneity. When an application or data is tended to change the environment, the transceiver may need to change RAT and F, just change RAT and maintain F, or reverse. Ultimately, the user session is maintained continuously and consistently during mobility by leveraging horizontal handoff process (Nasser, Hasswa, & Hassanein, 2006).

Horizontal Heterogeneity: When differentiation is across different types of mobile OSs, cloud services, or wireless networks it is named horizontal heterogeneity.

- *Mobile devices:* Among mobile devices, horizontal heterogeneity appears between different platforms: two or more OSs (e.g. Android and RIM) or brands (e.g. Samsung and Nokia). The horizontal oval shape in Figure 2.3(a) shows horizon-

tal heterogeneity between different OSs. For instance, the application developed for BlackBerry Torch[6] (RIM V6.0) is not executable in Android V3.x products.

Horizontal heterogeneity is usually more challenging compared with the vertical heterogeneity. Portability is exacerbated when development of applications such as Cloud-Mobile Hybrid (CMH) applications is concerned. To develop CMH applications, developers should design the application for the cloud as well as the mobile side. This development process must be repeated for various platforms, which is an exasperating effort for the developer (A.Manjunatha, A.Ranabahu, A.Sheth, & K.Thirunarayan, 2010). Developers should consider various mobile platforms, cloud vendors and supporting programming languages to decide how to develop the application which is an irksome impediment.

- *Clouds:* In the cloud, horizontal heterogeneity occurs between different types of clouds that provide heterogeneous services, like IaaS and PaaS. The horizontal oval shape in Figure 2.3(b) shows horizontal heterogeneity between various types of cloud services. In a scenario that some PaaS vendors offer free limited storage, if a new application utilizes such storage that is incidentally coupled with specific data structure like Google App Engine[7] (the only Google Query Language (GQL)-based PaaS cloud), such dependency locks the application in the cloud. Hence, porting rapidly growing data to an IaaS cloud (for less hosting cost) which is non-GQL-based IaaS (e.g. Structured Query Language (SQL)-based cloud) is hardly possible and imposes upfront investment.

  This type of heterogeneity, similar to the mobile side, is more difficult to address as compared with vertical heterogeneity because of switching difficulties between

[6] http://us.blackberry.com/smartphones/blackberrytorch/

[7] https://developers.google.com/appengine/

various service providers with different patterns, architectures, APIs, and business policies. Vertical hand of is the best

- *Wireless Networks:* Horizontal heterogeneity in wireless networks occurs when a mobile client is travelling across various networks like cellular and WLAN. Changing network node and supporting mobility, termed “vertical handoff”, highlighting a dilemma in horizontal heterogeneous wireless network, which is a more challenging task with presence of different infrastructure. In this situation, signal handoff processes are more difficult than vertical heterogeneity due to the switching process between different types of network (Nasser et al., 2006). Unlike the decision making algorithm in vertical handoff that relies on several parameters (like energy efficiency, Received Signal Strength (RSS), accessible bandwidth, security, financial cost, and user preference), decision in horizontal handoff is made based on the RSS only (Yan, Ahmet Sekercioglu, & Narayanan, 2010). Such increased complexity obliges vertical handoff optimization toward seamless connectivity in MCC which will enhance on-demand services and increase the quality of user experience. The horizontal oval shape in Figure 2.3(c) shows horizontal heterogeneity between various types of networks.

Review of challenges in MCC highlights that heterogeneity has remarkable impacts on mobile computation offloading, seamless connectivity, long WAN latency, mobility management, and vendor/data lock-in which encumber resource-intensive computing on the go and necessitate in-depth analysis. Also, managing billing systems and Service-Level Agreements (SLAs) in MCC become more complex considering heterogeneity among varied entities which demand upfront consideration on several parameters. Hence, mitigating the impact of heterogeneity can significantly enhance quality of MCC and extend usability

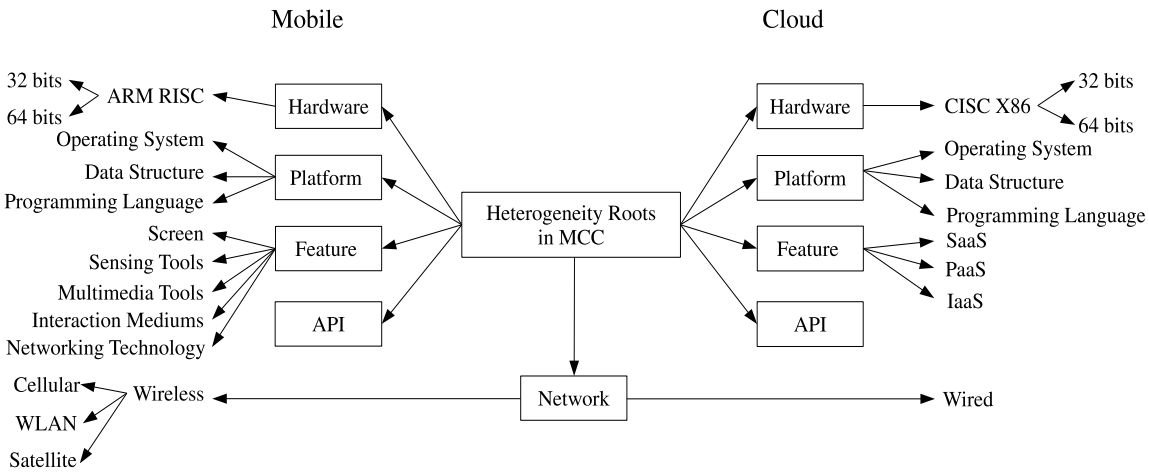


Figure 2.4: Taxonomy of heterogeneity roots in MCC.

of mobile devices to more resource-intensive computing areas. Due to its vital influence, we comprehensively analyze roots of heterogeneity in the following section.

2.3 Taxonomy of Heterogeneity Roots in MCC

In this section, we comprehensively study heterogeneity in MCC by analysing roots and dimensions of heterogeneity. We identify heterogeneity roots as hardware, platform, feature, API, and network and devise a taxonomy depicted in Figure 2.4.

2.3.1 Hardware Heterogeneity

Variety of hardware with different inward architecture between mobile devices, cloud servers, and network infrastructures (e.g. access points, radio transceivers, and routers) trigger hardware heterogeneity in MCC.

In the cloud environment, cloud providers maintain different infrastructures and architectural design to enhance quality of their service. Servers use X86 CISC (Complex Instruction Set Computer) architecture with two variations of 32-bit and 64-bit. Moreover, cloud infrastructures gradually grow more heterogeneous due to upgrade and replacement. The emerging growth of cloud computing will increase the number of geographically distributed cloud nodes that intensifies hardware heterogeneity among cloud providers.

The inward architecture and resource specifications of mobile devices such as processor speed, internal memory, radio specification, and battery capacity vary widely among different brands and models. The majority of smartphones are built based on 32-bit ARM RISC (Reduced Instruction Set Computer) processor architecture, but there is a large variation in terms of speed, number of cores, and amount of processor cache which suits them in specific domains. For instance, among recent Cortext series processors, Cortex-A is ideal for computing intensive multi user applications while Cortex-R is more suitable for real-time data processing scenarios (*"The ARM Cortex-A9 Processors"*, 2009). In the near future, the 64-bit ARM processors are expected to increase the heterogeneity among smartphones (*"ARM Discloses Technical Details Of The Next Version Of The ARM Architecture"*, 2011).

Hardware and architectural heterogeneity among mobile devices and cloud servers hamper direct deployment of cloud resources and services in mobile devices and leads to several problems as below.

- *Imbalanced quality and performance:* Variation in computing resources and their implementations diversify performance and quality of cloud services. Therefore, business contribution among cloud providers will be negatively affected by such quality divergence. It also makes decision making harder for users when choosing the appropriate vendor from available options. To facilitate the user's decision making, a comprehensive study and comparison among reputed cloud vendors is presented in (*Find the Best: Compare Cloud Computing Providers*, n.d.). Users can compare and contrast the service quality of each vendor from various aspects.

- *Data management and integrity:* The increasing number of very large scale geographically distributed data warehouses and the non-similarity of data structures

complicate data management. Integrating huge distributed data and providing virtually unified storage for mobile users is becoming more complicated with the ever increasing heterogeneity in MCC (Sakr, Liu, Batista, & Alomari, 2011).

- *Interoperation:* Data interoperation is the ability of connecting heterogeneous systems (based on wired or wireless), understanding geographical information resources, and exchanging data between/across two or more heterogeneous systems (Blair, Paolucci, Grace, & Georgantas, 2011). However, in MCC infrastructure diversity among various clouds on one hand and dissimilarities between cloud and mobile infrastructures with existence of wired against wireless network hardware systems on the other hand, have created data integration and interoperation problems in the absence of interface's standards and uniform platforms. For example, when Alice moves from current city to another while utilizing a nearby cloud 'A' via her Android mobile device, remote data (in part or whole) might be migrated to a nearer cloud 'B' for the sake of performance. In this situation, if the cloud 'B' fails to connect to the cloud 'A', if cloud 'B' cannot understand cloud 'A' database information after establishing a connection, or fail to exchange data with cloud 'A', the Alice computing experience will be degraded because of data interoperation problem.

- *Portability:* Codes are not easily movable and executable to/on heterogeneous hosts and the privilege of "write once run anywhere" is divested from developers. For instance, the application written for quad-core processor is not executable on dual-core processor due to architectural and hardware dissimilarities. Similarly, the applications developed for the ARM architecture cannot be executed on X86 without code modification and re-configuration.

- *Accurate energy estimation:* One of the most important aims of MCC is to conserve

mobile battery power. Thus, resource-intensive tasks are offloaded from mobile device to the cloud to deliver long-lasting online mobile services. However, before offloading, a decision making system needs to determine whether offloading computation can save energy or not (Kumar & Lu, 2010). If the remote execution can save energy, then the offloading is performed, otherwise the application is either terminated or executed locally. However, estimating energy efficiency of offloading is a non-trivial task due to heterogeneity of wireless technologies and infrastructures. Metrics such as power and bit-rate of wireless modems, activating time of interface, activation and deactivation delay of interface, and traffic pattern complicate accurate energy estimation. Also, varied hardware technologies and implementations are different in terms of power consumption (Perrucci et al., 2011). Therefore, precise estimation of required energy for application execution is difficult in different platforms.

- *Cloud-mobile application development:* Developing cross-platform components (i.e., cloud, mobile, and hybrid) for cloud-mobile application is a complicated task. Mobile components should be able to move among various smartphones while cloud components must be portable to all cloud infrastructures. The hybrid components should easily travel among various smartphone and cloud platforms.

To address these problems, Stone et al. (Stone, Gohara, & Shi, 2010) propose OpenCL as a parallel programming standard for heterogeneous computing devices. OpenCL enables developers to create desktop applications executable on various types of computing elements like multicore CPU, Graphical Processing Unit (GPU), and other accelerators. The same approach can be maintained in the cloud which shrinks hardware heterogeneity. Applications developed for certain architectures using OpenCL can be ported to other

architectures with guaranteed correct functionalities. Using multiple programming strategies, the application can query hardware specifications and capabilities of the hosting machine and choose an appropriate kernel to increase correctness and compatibility. However, developing and managing several kernels incur extra cost and encumbrance beside occupying lots of space.

To dilute the impact of hardware variety in the cloud and smartphones, Madhavapeddy et al. (Madhavapeddy, Mortier, Crowcroft, & Hand, 2010) propose a cloud OS called Mirage, based on virtualization technology. Mirage runs on top of a hypervisor to produce cross-platform applications that are portable to a plethora of mobile devices and cloud servers. In the Mirage, applications are developed on normal OS like Linux and then compiled into a kernel that is able to run directly on mobile devices and virtual clouds. Figure 2.5 depicts the layered architecture of Mirage that links the microkernel to an application on top of the hypervisor. Mirage microkernel leverages Xen hypervisor to lessen the impact of architecture heterogeneity of mobile and PCs on mobile applications. However, creating, maintaining, and destroying VM over a smartphone consume local resources and shorten battery life.

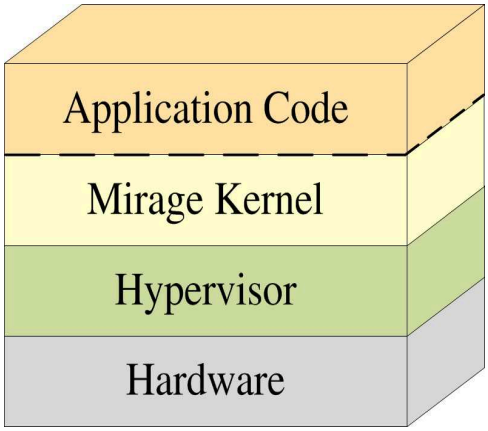


Figure 2.5: Mirage approach with statistically-linked kernel and application.

Huerta-Canepa and Lee (Huerta-Canepa & Lee, 2010) create a virtual cloud computing platform in the absence of a real cloud to augment mobile devices using an ad-hoc

network of mobile phones. In this approach impact of hardware heterogeneity is mitigated by leveraging smartphones as remote servers instead of using desktop machines or cloud servers. Similarly, Marinelli (Marinelli, 2009) propose a MCC platform composed of Android smartphones called Hyrax to lessen the architectural impact of non-ARM (X86) devices. However, using smartphones as cloud providers is hindered by two main challenges. Firstly, giant cloud processing resources are exchanged for limited resources of smartphones. Secondly, the lack of appropriate security and billing mechanisms for individual smartphone owners, discourage the sharing of scarce resources. However, mutual benefits of individual smartphone clients will likely encourage resource sharing in this context.

Furthermore, several application transition solutions like Marmalade and PhoneGap aim to mitigate the impact of heterogeneity on application portability by automatically generating compatible codes for various platforms. Though these approaches reduce the impact of heterogeneity on the application development process, is not congruous with the definition of portability offered by the InterNational Committee for Information Technology Standards (INCITS) (INCITS, n.d.) as portability is the capability of transferring one application from one device to a wide range of devices with little or no modification and conversion.

2.3.2 Platform Heterogeneity

Platform heterogeneity is the availability of various OSs, programming languages, and data structures in MCC. Currently, a plethora of heterogeneous mobile OSs such as Google's Android and Apple's iOS, each with multiple versions, developed to provide rich, compelling services to end-users. Each platform supports different combinations of programming language and data structures. For instance, Android offers Java language,

native code with JNI, and C/C++, while iOS supports Objective-C (Tarkoma & Lagerspetz, 2011).

In the context of cloud computing, the most widespread cloud providers such as Amazon Web Service (AWS)[8], Google App Engine[9], and Microsoft Azure[10] offer different OSs, programming languages, and data structures. Windows Azure supports a multiple choice of languages like .NET, PHP, Ruby, Phyton, and Java (*Windows Azure Developer Center*, 2012), whereas Google App Engine supports Java and its products plus Ruby and Python. AWS supports most developing languages and offers SDK for Android and iOS smartphones (*Amazon Web Services: Mobile Developer Center*, n.d.). Azure is built on the SQL, while the App Engine datastore includes GQL, and Amazon's Dynamo system (DeCandia et al., 2007) has a different structural design and partitioning scheme. Cloud providers enforce restrictions in databases to provide more flexible services.

Such non-uniformity makes an irksome impediment for cloud-mobile application developers and users. Portability and data integrity problems are also exacerbated in MCC. Therefore, mobile users, especially corporate mobile users with bulk sensitive data face problems in transferring a huge amount of data between heterogeneous clouds because it is a costly, time consuming, and risky process (A.Ranabahu & A.Sheth, 2010). Millions of records stored in a cloud database cannot be utilized in another cloud without compromising privacy and integrity when there is a difference between file systems and encryption techniques. Figure 2.6 depicts heterogeneous platforms and programming technologies in the cloud and mobile domains, and challenges for application programmer in selection of a suitable development environment or a programming language.

For MCC applications, we need a unified application environment similar to Aneka

[8]http://aws.amazon.com

[9]https://appengine.google.com/

[10]http://www.windowsazure.com

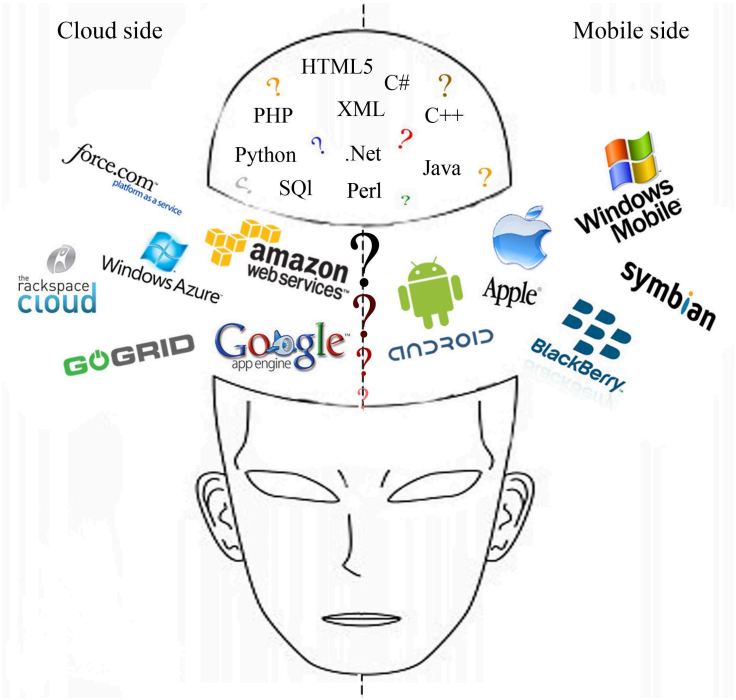


Figure 2.6: Platform heterogeneity in MCC and challenges for application developers.

(Vecchiola, Chu, & Buyya, 2009a), which allows one to develop cloud applications and deploy them on multiple cloud infrastructures such as AWS, Azure, and GoGrid[11] in a seamless manner. Although Aneka was developed to serve stationary clients, minor extensions and modifications (depending on the needs of client applications) can be performed to leverage Aneka in mobile environments.

2.3.3 Feature Heterogeneity

Feature heterogeneity in MCC is a result of feature variations in the mobile and cloud domains. Feature heterogeneity in smartphones is due to variation in native features like multimedia, sensing, and interaction tools, visualization area, and networking technologies. For instance, HTC Sensation[12] possesses an 8MP camera while BlackBerry Curve 8520[13] provides 2MP camera. Hence, the development process and performance of ap-

[11] http://www.gogrid.com/

[12] http://www.htc.com/us/products/sensation4g-tmobile/

[13] http://us.blackberry.com/smartphones/blackberrycurve8500/

plications on various devices are different. For example, the bar code reader application whose functionality depends on the quality of captured image may not offer similar functionality in HTC Sensation and BlackBerry Curve 8520.

Differentiation in sensing apparatuses and interaction features intensify application portability and usability. Recently, sensing tools such as tilting sensors have received noticeable attention from academia and the industry to enhance the quality of interaction in smartphones. In various proposals (Valberg & Christensen, 2009; Jones, Alexander, Andreou, Irani, & Subramanian, 2010) authors exploit sensing tools such as accelerometer to augment the interaction capabilities of smartphones towards delivering rich user experience. However, feature variation and lack of feature upgrading facility in smartphones obscure usability of feature-dependent applications on various devices.

Moreover, variation in the data visualization area remains a challenge for developers to design and deliver a common user interface for all smartphones. Omitting redundant and less important data according to the preference and context of individual users is an approach in some proposals like event-based semantic image adaptation scheme (Yin, Luo, & Chen, 2011). Authors identify important objects in an image and collect user preference using a simple feedback mechanism. An adaptation algorithm integrates important objects with user preferences to produce a suitable version for target mobile devices. This approach not only reduces the volume of presentation data by omitting non-important objects, but also offers a superb feeling of customization to mobile users.

Similarly, Chen et al. (Chen, Xie, Ma, & Zhang, 2005) propose a page split method to adapt web content for mobile devices. This approach is implemented inside the mobile browser to adapt the web page content according to the screen size and semantics of the content. However, components like page analysis, content detection, page splitting, and index page generation consume a high volume of local resources. Programming level

solutions such as screen support APIs in Android 3.2 provides more control to developers to adapt screen content for different Android devices with varied screen sizes (*Android 3.2*, n.d.).

Feature heterogeneity in the cloud domain arises from variations in the services (e.g. infrastructure, platform, software and security as a service) offered by different vendors. For example, Google App Engine (PaaS vendor) and Microsoft Windows Azure (PaaS) provide dissimilar security features; though they offer paid backup storage service, critical data privacy is only offered by Azure (*Find the Best: Compare Cloud Computing Providers*, n.d.). Therefore, users, especially corporate users, face difficulties in moving from one vendor to another.

2.3.4 API Heterogeneity

Application Programming Interface (API) is an interface supplied by OS vendors or service providers that allows an application written in a high-level language to access specific data or functions from the API distributor. Programmers, including mobile application developers, are usually in a hurry, while mobile users are increasingly demanding a rich computing experience. Therefore, APIs play an important role in delivering a rich experience to mobile users. Mobile platforms such as Android, BlackBerry, and iOS offer a gigantic number of APIs to assist programmers with developing rich mobile applications without direct access to the kernel. However, application portability has become an irksome practice for developers due to inward dissimilarities of APIs. Tarkoma and Lagerspetz (Tarkoma & Lagerspetz, 2011) study the role of APIs in mobile devices and argue that “The marketplace has a clear need for a common API that unifies network connectivity, energy awareness, and the user experience”.

Similarly, on the cloud side, the majority of cloud providers develop and deploy their

own proprietary APIs to describe syntax of specific operations to be utilized by their clients. A drastic growth in the number of cloud providers has created a huge silo of different APIs that intensifies the difficulty of developing applications due to interpreting semantics of data and operations. This outlook, results in API variation intensifying interoperability and portability issues. To mitigate the impact of API heterogeneity on the cloud, several regulatory and research unions endeavour to provide common cloud APIs through, including the European Telecommunications Standards Institute Technical Community (ETSI TC Cloud)[14], DMTF[15], and Cloud Audit[16].

2.3.5 Network Heterogeneity

The composition of various wireless technologies such as WiFi, 3G, and WiMAX makes MCC more complicated compared to cloud computing. Unlike desktop computers, smartphones utilize wireless communication which is comparatively more intermittent and unreliable while offers lower bandwidth. Client mobility among varied network environments intensifies communication deficiencies and stems complex issues like signal handover. Inappropriate decision making during the handover process like (i) less appropriate selection of network technology among available candidates and (ii) transferring the communication link at the wrong time, increases WAN latency and jitter which directly degrade the quality of service, especially for delay sensitive data and functions (Yan et al., 2010).

To tackle these challenges, the concept of seamless connectivity among heterogeneous wireless technologies plays a vital role that necessitates reliable intra-system and inter-system handoff schemes (Nasser et al., 2006). Intra-system handover is a less chal-

[14]http://www.etsi.org/WebSite/Technologies/GRID_CLOUD.aspx

[15]http://www.dmtf.org

[16]http://cloudaudit.org/CloudAudit/Home.html

lenging task due to inward homogeneity of engaging technologies, while addressing inter-system handover is more complicated due to signal transmission difficulties between heterogeneous environments. To realize seamless connectivity across heterogeneous wireless networks, the burgeoning concept of next generation wireless networks (Akyildiz, Jiang, & Mohanty, 2004) with the notion of all Internet Protocol (IP)-based infrastructures is emerging. In the absence of seamless connectivity, the quality of user experience is decreased because of decrements in communication quality and increments in code execution and application response time.

In addition, convergence of wireless and wired networks creates data bottleneck problem. The problem happens when large data streams from a wired network flood the limited bandwidth of wireless networks. This congestion not only increases packet drop ratio and prolongs data transfer between cloud and mobile, but also rises control and maintenance operations which demands enhancement in current network architecture and design for mobile operators.

However, network operators are employing powerful helping nodes like wall-connected WiFi hotspots to relay data packets and reduce the intensive load from congested cells. The helping nodes' bandwidth is not infinite because very large bandwidth cannot effectively enhance data trafficking and is limited to a certain upper bound (Li & Fang, 2012). Hence, a large number of such helping nodes are needed to alleviate ever increasing wireless traffic. Although, MCC imposes overhead on operators' network, strategies like Cisco's innovative Next-Generation Hotspot (NGH) (*"The Future of Hotspots: Making Wi-Fi as Secure and Easy to Use as Cellular"*, 2012) can reduce data traffic and hike the MNOs' revenue by increasing market share through subscriber retention. NGH is an advance approach to provide mobile network optimization by offering WiFi as a side mechanism for secure mobile access of data traffic, while aims to enrich the user experience.

2.4 Impacts of Heterogeneity in MCC

This section discuss several opportunities and challenges raised due to variations in smartphones, clouds, and networking technologies

2.4.1 Opportunities

The opportunities arising from heterogeneity in MCC are explained as follows:

2.4.1 (a) Performance Gain

User perceived performance from online mobile applications is highly influenced by computing performance of server and wireless networks in MCC (Huang et al., 2010). While it is financially impossible to avoid infrastructure heterogeneity within a single cloud (due to maintenance, update, and technological advancements), cloud service providers use such heterogeneity as an opportunity to enhance performance and cost of their services. Rosenberg and Chiang (Rosenberg & Chiang, 2010) study two clusters of computers with identical mean speed. In one cluster, all CPU speeds are similar while in the other one, heterogeneous CPUs are utilized. The authors analyze execution performance of both clusters and mathematically demonstrate that the cluster with heterogeneous CPU speed outperforms another cluster with homogeneous CPUs.

Moreover, because varied applications have different architectural preferences, combination of heterogeneous processors with dissimilar architectural features (e.g., pipeline depth, in-order versus out-of-order execution, and superscalar width) can efficiently meet application preferences toward better performance (M. Guevara, 2013) (further explanation is out of the scope of this thesis). For instance, Amazon leverages dissimilar processing entities such as Intel Xeon E5507 and AMD Opteron 2218HE for creating an instance of VM to fulfil various computing requirements of different mobile applications with least possible cost. Exploiting heterogeneous computing resources in creating Virtual Machine

(VM) instances enhances execution performance of online applications in MCC (Rosenberg & Chiang, 2010; M. Guevara, 2013; Ou, Zhuang, Nurminen, Ylä-Jääski, & Hui, 2012).

Additionally, researchers and industrialists can leverage benefits of heterogeneous communication and networking technologies such as 2G and 3G towards efficient communication. In enterprise organization, running communication-intensive cloud-mobile applications on 3G-ready devices and rest of the applications on 2G devices can reduce capital and operational costs of buying 3G-enabled mobile device and long-lasting batteries (Motorola, 2008). In academia, researchers (Rahimi, Venkatasubramanian, Mehrotra, & Vasilakos, 2012) study execution time and cost of code offloading in MCC. They exploit heterogeneous communication technologies to access remote computation resources located near/far to/from the mobile users and could remarkably enhance application performance and responsiveness by leveraging both 3G and WiFi. The authors propose a 2-tiered cloud infrastructure consist of distant clouds and nearby cloudlets aiming to alleviate effectiveness and efficiency of computation offloading process. They leverage WiFi technology to save energy while communicating with nearby cloudlets and use 3G to access distant giant clouds. The authors report 32% lower delay and 40% cost-reduction in offloading computation-intensive tasks to two-tiered cloud with heterogeneous infrastructure compare to single-tiered cloud.

Thus, heterogeneous co-existence of various communication technologies can originate a performance-energy trade off which can benefit mobile clients, cloud vendors, and network operators. Future mobile communication technologies will likely enable mobile nodes to negotiate with network infrastructures for optimized communication technology. Therefore, heterogeneous selection of infrastructures in mobile-cloud environment provides a higher performance and service range to MCC community.

2.4.1 (b) Enhanced Application Response Time

Reducing cloud-mobile application response time is one of the most important requirements of MCC which is likely achievable by enhanced performance of leveraging heterogeneous resources while outsourcing computation. Heterogeneous computation hardware in the cloud datacenters, remarkably decreases computation delay in server side resulting less cloud-mobile execution latency and better overall response time. Researchers in (M. Guevara, 2013) explore that leveraging combination of three heterogeneous processor architectures can reduce response time violation by 12X leading to crisp application response which is critical in MCC. Therefore, leveraging heterogeneous computing resources can increase application responsiveness in MCC.

2.4.1 (c) Cost Efficiency

One of the most critical metrics in successful MCC adoption is the cost of utilizing cloud services by mobile users which includes the amount of native resources, energy, and time used as well as monetary cost of inter-system communication and computation. Efficient match of communication requirements and available wireless technologies, and migrating resource-intensive tasks to high-performance heterogeneous clouds, reduce overall application execution cost.

Researchers in (Ou et al., 2012) observe heterogeneity within computing entities in Amazon EC2 for two period in 2011-2012 and explore that Amazon EC2 leverages various processors in creating a single VM to reduce costs and achieve efficiency. For example, in large VM instances, heterogeneous CPU models, namely Intel Xeon E5507, E5430, E5645, and AMD Opteron 2218HE and 270 are being used. Experimental analysis of CPU, memory, and disk performance of various VM instances resulted different performance outcomes in varied computation loads. The authors conclude that efficient

selection of service instance from pool of heterogeneous instances, can deliver up to 30% cost saving to Amazon end-users. Therefore, by assuming rich computing capabilities in cloud and seamless high bandwidth connectivity between mobile and cloud, mobile application responsiveness, local resource consumption, and utilization cost could be enhanced towards more efficiency.

2.4.2 Challenges

Several important challenges resulting from, or intensified by, heterogeneity, are explained below.

- *Interoperability:* API Heterogeneity of mobile and cloud systems beside inward variation in cloud system structure originate interoperability as a major challenge in MCC (Hogan et al., 2011). The challenge intensifies vendor lock-in problem and obscures data migration and code transition across a multitude of existing processing units. In MCC, providing collaboration among various mobile and cloud processing unites with different interfaces is a non-trivial matter. Figure 2.7 depicts mobile-cloud interoperability across a silo of computing devices and illustrates the inter-cloud collaboration between different cloud providers as essential requirements in MCC.

- *Portability:* A lack of standards, technologies, and solutions to handle heterogeneity in MCC implicitly creates the portability problem. In the cloud, providers offer various computing services with different structures and programming languages. Similarly, smartphone vendors develop various approaches and technologies to enhance the quality of their products. Therefore, porting an application to various computing devices in a the heterogeneous MCC domain has become more difficult.

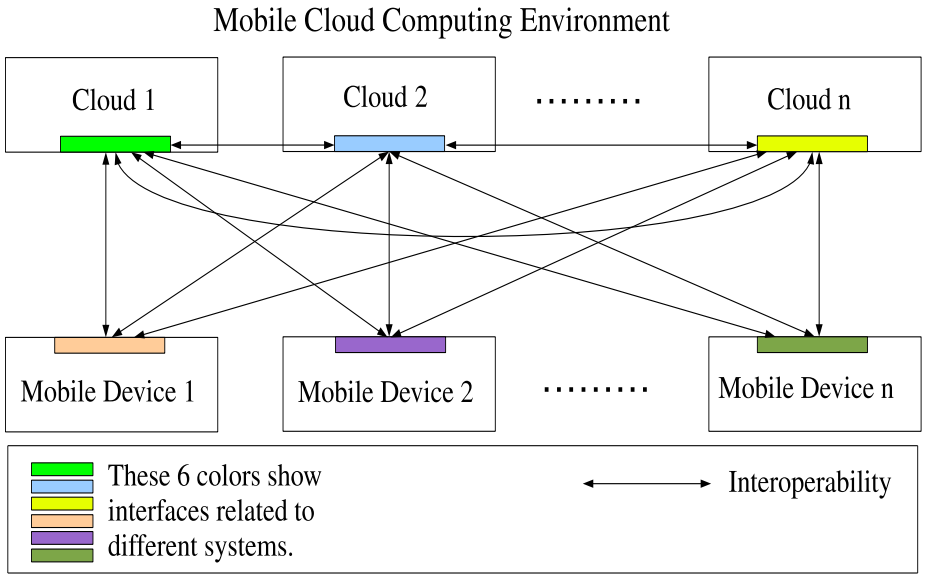


Figure 2.7: Interoperability in MCC: Collaboration of inter-cloud and mobile-cloud systems with varied interfaces provides interoperability.

Ideally, data and application should be able to cross a multitude of clouds and smartphones with no or little configuration and conversion (INCITS, n.d.). However, it is almost impractical to port native codes to the cloud and transfer resource-intensive codes from clouds to the weak smartphones. Therefore, we limit portability in MCC to the ability of (i) migrating cloud components from one cloud to other clouds, (ii) migrating mobile components from one smartphone to other smartphones, and (iii) migrating data across heterogeneous clouds and smartphones. Figure 2.8 depicts portability in MCC.

- *Developing cost:* Developing one application for several platforms is a costly process that demands knowledge of several programming languages and enforces redundant design and programming tasks. In the presence of heterogeneity, the market share for a single version of an application is diminished to a fraction of total customers that decrease revenue and demotivates individual programmers.

- *Time lag:* In many scenarios, although the financial cost of developing a multi-platform application is affordable and welcomed due to its business opportunity, pro-

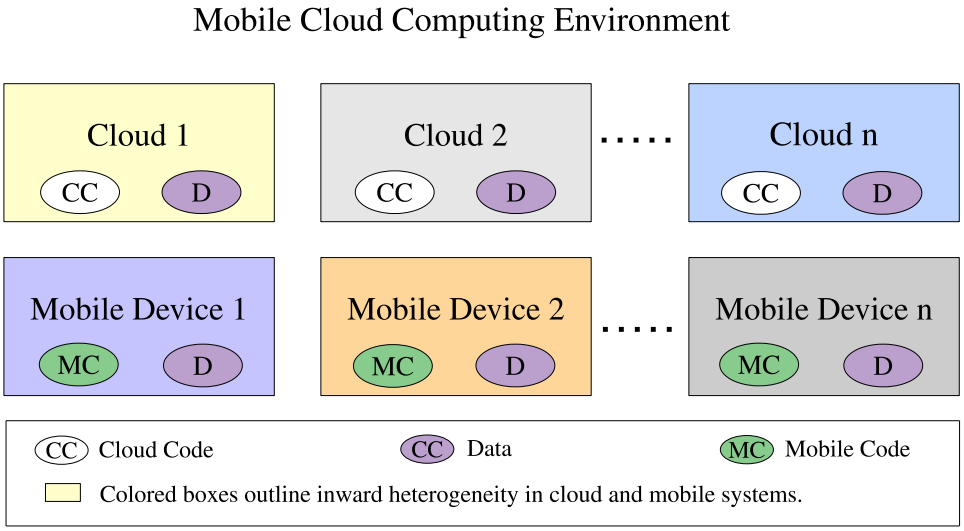


Figure 2.8: Portability in MCC: Data should be portable to all cloud and mobile devices. Cloud codes should move between clouds while mobile codes should move between a multitude of mobile devices regardless of the inward heterogeneity of hosting machines.

longed application development process procrastinates service offering to a wider community and leads to financial loss.

- *Application maintenance:* When there are multiple versions of a single application, each for a different platform, if one application is enhanced or changed, the modifications should be reflected to all versions. Usually a single development model cannot be employed and extended to different platforms. Hence, different modification approaches should be undertaken for each platform which is a hectic job for developers.

- *Communication:* The heterogeneity of wireless technologies in MCC has created communication problems such as signal handover. Bandwidth variation between wireless and wired technologies are the source of data bottlenecks when huge amounts of data stream from wired (cloud servers) into a wireless medium (mobile devices). Therefore, continuous, consistent connectivity and scalable, accessible networking services are necessary to enhance the quality of mobility and communication between a wide range of mobile devices and clouds.

- *Security and privacy:* Although MCC can enhance the quality of mobile computing and increase the usability of smartphones, a drastic rise in security glitches and cyber crimes (Cachin & Schunter, 2011) is diminishing trust among cloud users in MCC. Storing confidential information (e.g. banking information, medical records, and social security numbers) in cloud infrastructures and remote access to them via the Internet and wireless mediums threatens mobile users in the presence of numerous hackers. Therefore, to increase trust among cloud-mobile users and a secure collaboration process between different cloud service providers and consumers, strong authentication, authorization, and communication protection are required. For example, data migration from one cloud to another (serially) or operation across multiple clouds (simultaneously) should be secured by cloud providers. During communication processes, personal information and personally identifiable information require protection by cloud providers, mobile network operators, and trusted third parties. Identity provisioning and access management through different environments are a sample of the security keys which manifest the necessity of secure intercommunication in MCC.

2.5 Mobile Computation Outsourcing Architectures

We review Mobile Computation Outsourcing (MCO) approaches and identify four architectures, considering existing heterogeneity in granularity of scalability, locality, and multiplicity. Granularity in resources has three main aspects of scalability, proximity (location), and multiplicity. Scalability is a major aspect in cloud-based resources. The higher scalability and elasticity of the resources, the larger granular resources. Those resources are called coarse-grained resources. Locality aspect refers to the distance from cloud-based resources to mobile users. Lastly, multiplicity concerns the number of resources;

the larger is the number of resources, the finer would be their multiplicity granularity. We classify the existing heterogeneities in granularity into two classes of vertical and horizontal illustrated in Figure 2.10 and described as follows.

2.5.1 Vertically Heterogeneous Mobile Computation Outsourcing

Vertically Heterogeneous Mobile Computation Outsourcing (VHMCO) referred to those computation outsourcing approaches that have used vertically heterogeneous cloud-based resources. As described in section 2.2.2, in vertically heterogeneous cloud-based resources, differentiation is among one silo of cloud-based resources. For instance, in a VHMCO solution, only heterogeneous VM instances of amazon EC2 cloud are utilized.

Researchers in recent works have leveraged three main types of vertically heterogeneous cloud-based resources, including CGRs, MGRs, and FGRs that can perform heterogeneous computations. Therefore, we classify existing works in three classes of coarse, medium, and fine granular approaches that are described in below.

2.5.1 (a) Coarse Granular

VHMCO approaches that leverages CGRs as computing resources to undertake computation outsourcing for CMA execution are classified in this category. Coarse-grained resources depicted in Figure 2.9(a) are those computing resources that are located in distant from majority of mobile users, feature high scalability and elasticity. CGRs are significantly low in number and are located in very few geographical regions, but their computing powers are infinite. Thus, their multiplicity is low. Amazon EC2 and Google App Engine are two example of coarse-grained resource providers. Efforts such as (B.Chun, S.Ihm, P.Maniatis, & M.Naik, 2010; Cuervo et al., 2010) are using coarse-granular resources.

The advantages of using coarse-grained resources in MCC are high availability, scalability, elasticity, reliability, and security that make them suitable resources for extremely

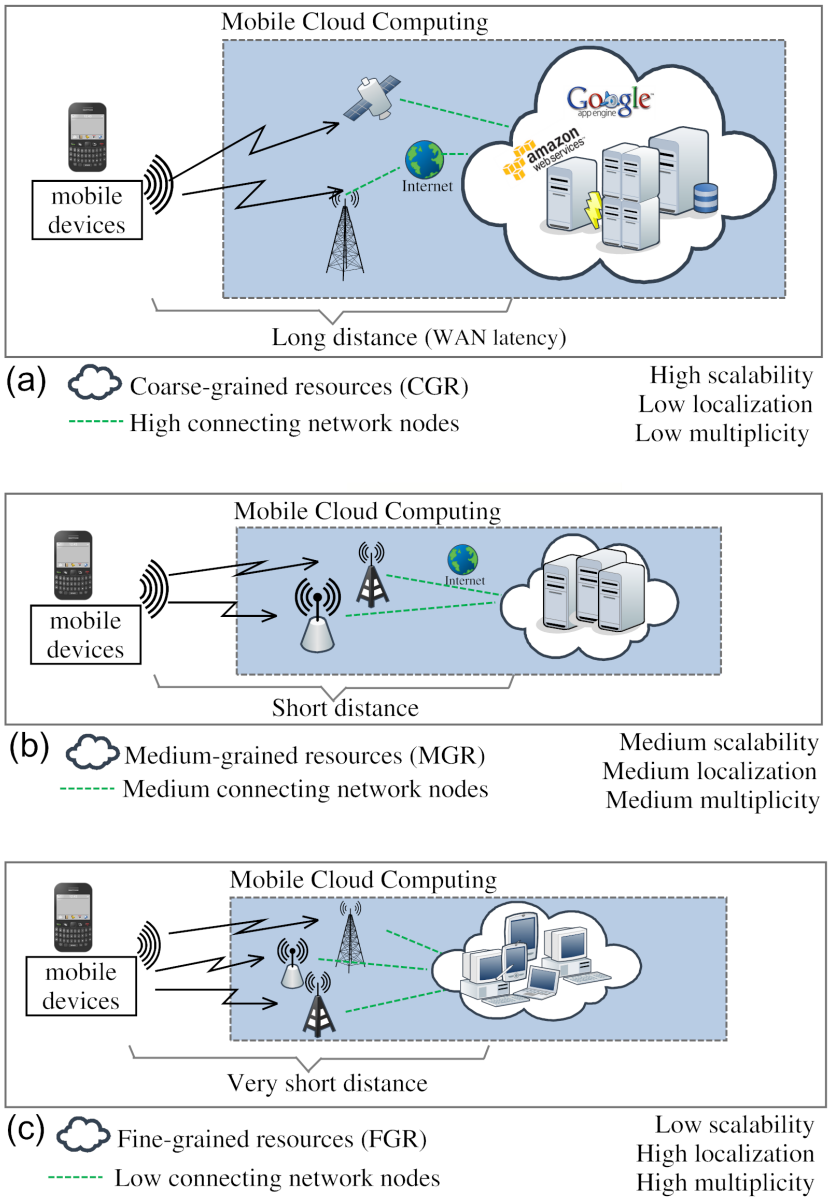


Figure 2.9: Conceptual view of mobile cloud computing architectures.

intensive computational task. However, leveraging these resources from mobile devices is encumbered by long WAN latency (Abolfazli, Sanaei, Alizadeh, Gani, & Xia, 2014). Due to low multiplicity, CGRs are often far from users and accessing them via WAN through Internet is time, energy, and money intensive. This is the best architecture for substantially compute-intensive tasks which are not time-sensitive so that communication latency does not impact much on user experience. However, this architecture is not suitable for data-intensive and communication-intensive applications.

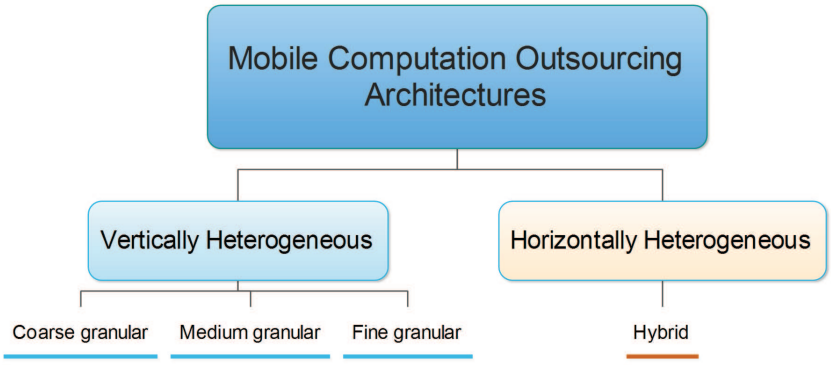


Figure 2.10: Taxonomy of mobile computation outsourcing architectures

2.5.1 (b) Fine Granular

Fine granular architecture is another emerging type of MCO architecture in which fine-grained cloud-based resources are leveraged as remote resources. FGRs shown in Figure 2.9(c) are located in user proximity and feature low scalability and elasticity. Smartphones, tablets, laptops, and car-mounter computers, and desktop computers are good examples of FGRs.

Multiplicity of FGRs is significantly higher than CGRs and is expected to grow in the presence of insatiably popularity of mobile devices. They are located almost everywhere, but their computing powers is very limited. MOMCC (Abolfazli, Sanaei, Shiraz, & Gani, 2012) and Hyrax (Marinelli, 2009) are exemplary efforts that use FGRs for outsourcing in MCC.

The advantages of using FGRs are their great multiplicity and short network latency. However, their high proximity to mobile users make them suitable resources for extremely time-sensitive interactive small applications. Huge number of mobile devices and rapidly increasing shipment of mobile devices compared to desktop computers, make it feasible to build a cloud of mobile devices and use their accumulated power for MCO.

2.5.1 (c) Medium Granular

Medium granular architecture exploits medium-grained computational resources for remote computation. Medium-grained resources illustrated in Figure 2.9(b) are computing resources that are located in nearer location compared to the CGRs, but are not as near as FGRs. They feature medium scalability and elasticity more than fine-grained resources and less than coarse-grained resources.

Multiplicity of these resources is more than CGRs and less than FGRs. Number of MGRs is more than CGRs, they are located in more geographical regions, and their computing powers is medium. Cloudlet (Satyanarayanan et al., 2009) is one of the most credible efforts that has exploited computing power of nearby desktop computers.

The advantages of using MGRs are their lower WAN latency and higher multiplicity. These resources are numerically more available than CGRs. But their performance is limited due to limited scalability, elasticity, reliability, and security. These resources are suitable for compute-moderate delay-moderate tasks.

2.5.2 Horizontally Heterogeneous Mobile Computation Outsourcing

Unlike VHMCO in which cloud-based resources are selected from one type of grained resources, in Horizontally Heterogeneous Mobile Computation Outsourcing (H2MCO) solutions, the resources are composed of multiple classes of heterogeneous granular cloud-based resources.

H2MCO are referred to those MCO approaches that have used horizontally heterogeneous cloud-based resources. In H2MCO, there is a high granularity and resource differentiation among various utilized cloud-based resources for compute-intensive mobile applications. For instance, in a H2MCO solution, VM instances of Amazon EC2 cloud, cloudlets, and mobile devices are used which are placed in three separate layers with

different functional and non-functional characteristics (e.g., computational attributes, location granularity, and resource multiplicity). Mobile applications built based on H2MCO can take the benefits of all three classes of granularities. In H2MCO, the computing and communication latencies can be trade-offed to gain better performance.

2.5.2 (a) Hybrid

Hybrid MCO depicted in Figure 2.11 is a feasible MCO architecture in which computing resources are composed of CGRs, MGRs, and FGRs. Hybrid resources are classified in category of H2MCO since these resources in contrast to vertically heterogeneous cloud-based resources, use multi-tier horizontally heterogeneous resources. MapCloud (Rahimi et al., 2012) and SAMI (Sanaei, Abolfazli, Gani, & Shiraz, 2012) are efforts using 2- and 3-tiered resources (CGRs, MGRs, and FGRs). The former which is an offloading solution aimed to reduce the time and price cost of offloading compute-intensive tasks to remote outsources. However, though it can minimize the offloading overhead, it cannot entirely omit it, and hence the offloading overhead (especially code migration) remains partially. The latter, however, aimed to entirely dismiss the overhead of code offloading by deploying SOA that calls already-available codes in server instead of migrating code to the server. MOCHA (Soyata, Muraleedharan, Funai, Kwon, & Heinzelman, 2012) is another effort that uses fixed and greedy partitioning algorithms; In the former algorithm, the task is partitioned into equal sizes and distributed among all Cloudlet and cloud servers. In the latter, the response latency of computing devices is considered for allocation. Thus, the task is partitioned and distributed among computers based on time; the first partition is sent to the device with shortest latency while the last partition is sent to the device with highest latency. In comparison, the execution time of partitions is better in greedy than fixed, especially when Cloudlet is utilized in outsourcing lifecycle and numerous cloud datacen-

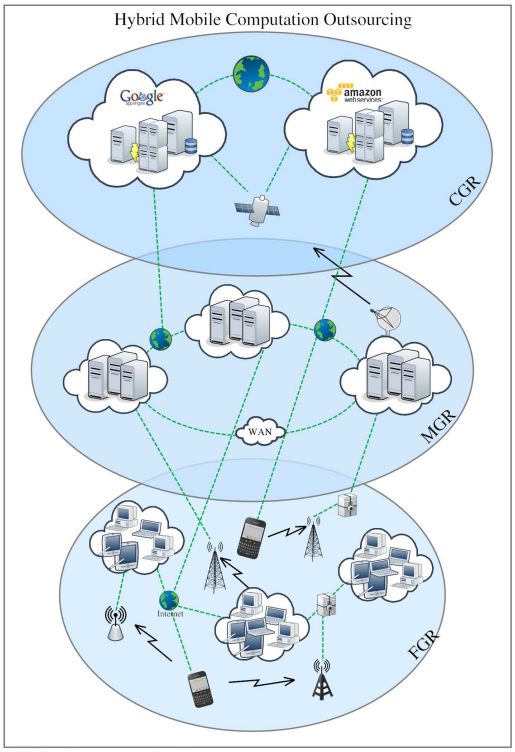


Figure 2.11: Conceptual view of hybrid mobile computation outsourcing architecture

ters with heterogeneous response time exist. Nevertheless, mobile devices should acquire prior knowledge of the server's latency which is a resource-hungry and time-consuming task that can degrade the performance gain. Finally, hybrid resources are aimed to overcome limited scalability and locality capabilities problems caused by VHMCO resources, and finally to decrease response time and energy consumption of intensive applications. Therefore, to advance cloud computing platform for mobile devices, there is a need for a horizontally heterogeneous hybrid MCO model composed of varied types of granular resources. This hybrid cloud-based platform deemed could accumulate strengths and benefits of each granular resource class and develop a multi-layered cloud platform to achieve high performance at the time of computing resource-intensive mobile applications.

The computation outsourcing systems based on hybrid resources, significantly gain execution performance (discussed in section 2.4.1 (a)) of the existing heterogeneous ca-

pabilities to better match the resources with the computing requirements of the requested tasks from resource-intensive mobile application. If the task is significantly compute-intensive and requires very large resources, the hybrid system can offer coarse-grained resources and if the task demands crisp response fine-grained resources can be allocated.

Moreover, the hybrid system can have the trade-off ability between computation and communication to handle the scalability problem at the time of increasing the number of requests to the server, and long WAN latency between cloud-mobile consumer and cloud service provider.

2.6 Heterogeneity-Handling Techniques

Due to remarkable proven benefits of heterogeneity, it is beneficial not to dilute or remove such differences. However, non-addressed heterogeneity can originate severe challenges that encumber success and adoption of MCC solutions. Therefore, the existing variations should be handled for excess benefits. In this section we describe three major techniques to handle heterogeneity in MCC.

2.6.1 Service Oriented Architecture (SOA)

Service-Oriented Architecture (SOA) is a well-known design philosophy independent from specific technology, vendors, and business policies that incorporates different services towards generating complex applications and services. Web services are well-known SOA implementation models that could successfully integrate heterogeneous services from various service providers like Facebook, Google, and Yahoo[17] and enable interoperability across them. For instance, Facebook delivers multimedia YouTube[18] content regardless of inward differentiations.

[17]http://yahoo.com

[18]http://youtube.com

Service Oriented Computing (SOC) (Huhns & Singh, 2005) is a service-driven approach to generate service-based applications with least dependency to the specific platform. In service-based applications, functions are defined, implemented, and combined as services to enhance application granularity and modularity, flexibility, scalability, and reusability. This approach encourages the development of elastic applications meaning that users are able to extend and shrink functionality on-demand which is parallel to the vision of cloud computing. Aneka (Vecchiola, Chu, & Buyya, 2009b) is an example of service-oriented solution to automatically manage distributed resources (clouds and grids). Aneka is designed to be robust against variations in application models, security solutions, and communication protocols such that client choices can be applied at any time without affecting the existing system.

In summary, loosely coupled SOA-driven services have a proactive potential to integrate heterogeneous resources in MCC. Loutas et al. (Loutas et al., 2010) observe interoperability issues between cloud systems and perceive the ability of SOA plus semantics for a new cloud landscape. They present architecture for heterogeneous clouds, called RASIC, to enable semantic interoperability among clouds. The authors propose a user-centric paradigm to facilitate developing and deploying of SOA-based services in a large-scale, resource-intensive environment hosted by different cloud providers. It is concluded that utilizing SOA-based design philosophy towards a common API standard for cloud can eliminate vendor lock-in problem, ease content migration across heterogeneous clouds, and reduce the cost of porting data and application from one cloud to another.

2.6.2 Middleware/Adapter

Adapter is an intermediate tool or approach to smooth out the impact of heterogeneity in a specific domain like software engineering and distributed computing environments. A

'plug adapter' is a well-known example that helps an international traveller to overcome the incompatibility problem caused by dissimilarity of power outlets in different countries. In software engineering, adapter is a pattern designed to address the common problems caused by heterogeneity in many situations (Wolfgang, 1994). Distributed computing middleware such as Object Request Broker (OBR) acts as an arbiter to enable communication between heterogeneous object systems regardless of their inward differences.

Emergent middleware as Blair and Grace (Blair & Grace, 2012) describe, is the contemporary approach to tackle problems caused by extreme heterogeneity. The idea is advocated by enormous accomplishments in academia and industry. Emerging efforts such as (Chandrakant, Bijil, Shenoy, Venugopal, & Patnaik, 2012; Andriescu, Speicys Cardoso, & Issarny, 2011; Gupta, Zeldovich, & Madden, 2011; Walraven, Truyen, & Joosen, 2011; Bromberg, Grace, Réveillère, & Blair, 2011; Rellermeyer & Küpfer, 2011) in academy beside commercial products such as Oracle Fusion Middleware 11g[19] and Open Middleware Adapter (OMA)[20] advocate suitability of adapters and more specifically middleware technology to tackle the heterogeneity-made problems. However, contemporary middleware require great deal of research and development to become an appropriate heterogeneity handling technique in MCC.

2.6.3 Virtualization

Virtualization (VMware, n.d.) is one of the cornerstone technologies of MCC promising to reduce the negative impacts of hardware, feature, and platform heterogeneity. Using the virtualization approach, a VM manager (hypervisor) is deployed on top of a cloud, mobile, or both to host desired platform(s) in order to create a homogeneous execution environment between various smartphones and clouds. Mirage is an example effort that

[19] http://www.oracle.com/us/products/middleware/index.html

[20] http://ultra-ats.com/products/open-middleware-adapter-oma-software-development-kit-sdk/

exploits hypervisor in both the cloud and smartphone to build cross-platform applications regardless of variations in underlying devices. However, VM deployment and management impose excessive overhead on resource-constrained mobile devices (Shiraz & Gani, 2012).

Moreover, recent application offloading approaches like (Cuervo et al., 2010; B. Chun et al., 2011) leverage virtualization technology to offload mobile application (entirely or partially) to remote cloud resources for execution. Similarly, efforts like (Lu et al., 2011) exploit virtualization technology to separate screen rendering tasks from the presentation layer and migrate them to the cloud. On top of a VM inside a cloud, a remote server can render screen-related processing tasks and send the result to the device. The authors aim to address the feature heterogeneity (screen size) of smartphones using a virtual screen rendering approach. However, virtualization gives rise to several security threats such as VM hopping and VM escape (Owens, 2009). VM hopping is a virtualization threat through which an attacker can exploit a VM and attack other VM(s) on the same host. VM escapes can violate the security of VMs when an attacker accesses control over the hypervisor. Therefore, several open challenges such as VM deployment and management (Shiraz & Gani, 2012), and security establishment (Takabi, Joshi, & Ahn, 2010; Chen, Paxson, & Katz, 2010) should be alleviated before virtualization technology can be fully established as grounding technology in MCC.

2.7 Open Issues

This section presents some of the research directions in MCC, especially those which are more complex due to the heterogeneity. Addressing these open issues is vital to alleviating the restrictions caused by heterogeneity.

2.7.1 Architectural issues:

A reference architecture for heterogeneous MCC environment is a crucial requirement for unleashing the power of mobile computing towards unrestricted ubiquitous computing. Employing a unified or a joint architecture composed of varied architectures requires further studies. A generic architecture might be a little optimistic when market competition enforces business policies for mobile manufacturers and cloud providers. However, it is achievable by leveraging technology-neutral design approaches such as SOA. Several research communities like NIST (L.-J. Zhang & Zhou, 2009), HP (Bohn, Messina, Liu, Tong, & Mao, 2011), and IBM (Behrendt et al., 2011) endeavour to address the open challenges of cloud computing by proposing conceptual reference architectures. In the absence of such reference architecture for MCC, ubiquity of mobile computing is diminished. Several open challenges can be alleviated in the presence of the reference architecture to release the power of mobile devices.

2.7.2 Mobile Computation Offloading Issues:

Leveraging varied cloud resources to augment computing limitations of multitude of mobile devices towards realizing the vision of unrestricted functionality, storage, and mobility in current diverse communication environment is a non-trivial task. Realizing cloud-based augmentation vision is impeded by multi-dimensional overhead of identifying and efficiently partitioning resource intensive components, VM (Virtual Machine) creation and migration, and monitoring the overall outsourcing process (Shiraz, Gani, Hafeez, & Buyya, 2012). Moreover, a plethora of hurdles in utilizing cloud resources such as communication latency, heterogeneity, security (both offloaded code and cloud permanent software), code portability, and cloud-mobile interoperability intensifies the situation.

2.7.3 Communication and Computation Latency Issues:

Latency adversely impacts on the energy efficiency (Miettinen & Nurminen, 2010) and interactive response (Lagar-Cavilla, Tolia, Lara, Satyanarayanan, & Hallaron, 2007) of cloud-mobile applications by consuming excessive mobile resources and raising transmission and computation delays. In cellular communication, distance from the base station (near or far) and variation in bandwidth and speed of various wireless technologies affect the energy efficiency and usability of MCC devices. For example, data transfer bit-rate consumes comparatively more energy in cellular networks than WLAN. The higher the transmission bit-rate, the more energy efficient the transmission (Miettinen & Nurminen, 2010). Moreover, leveraging wireless Internet networks to offload mobile intensive applications to distant cloud resources creates a bottleneck. Consequently, when the long WAN latency is increased, the quality of user experience is decreased due to communication delays.

Moreover, computation latency is challenging issue raised through research works leveraging fine- and medium-grained cloud-based resources. This type of resources such as desktops in coffee shops or laptop computers are closer to mobile users, but feature low scalability and hence utilizing their resources originate computing latency that impacts on computation outsourcing performance in MCC. Although, they can provide services with low communication overhead because of their vicinity to the mobile users, this kind of resources impose computation overhead due to their medium or low scalability based on computing, processing, and storage capabilities of devices. Thus, addressing the communication and computation latencies remains a challenging task in MCC. Future efforts are deemed to leverage heterogeneous hybrid granular resources utilizing optimized scheduling algorithms (Javanmardi et al., 2014) to address the resource deficiencies of mobile devices by remotely performing intensive components of cloud-based mobile applications

via computation outsourcing.

2.7.4 Energy Constraint Issues:

Energy is the only unreplenishable resource in mobile devices that cannot be restored spontaneously and requires external resources to be renewed (Satyanarayanan, 2005). Current technologies can increase battery capacity by only 5% per annum (Robinson, 2009). Several energy harvesting efforts have been in progress since the 1990s to replenish energy from external resources like human movement (Flinn & Satyanarayanan, 1999) and wireless radiation (R.Avro, 2009), but these intermittent resources are not available on-demand (Pickard & Abbott, 2012). Alternatively, application offloading (Satyanarayanan, 2001; Xia et al., 2014) and fidelity adaptation (Rajesh Krishna, 2004) approaches are proposed to conserve local mobile resources, especially energy. However, application offloading is a risky, resource-intensive approach that needs further research and development to be deployed in real scenarios (Sharifi et al., 2011). Fidelity adaptation solutions compromise quality to conserve local resources which impoverish quality of user experience in MCC. Researchers (Cuervo et al., 2010; B. Chun et al., 2011; X. W. Zhang et al., 2011) endeavoured to mitigate application offloading challenges by exploiting secure, reliable, elastic cloud resources instead of insecure, limited surrogate's resources. However, cloud-based application offloading cannot always save energy with current developments which demands further efforts (Kumar, Liu, Lu, & Bhargava, 2012; Kumar & Lu, 2010; Sharifi et al., 2011). Therefore, the energy constraint of mobile nodes remains a problem in MCC.

2.7.5 Elasticity Issues:

Cloud providers confront situations in which there are more demands than available resources. Adverse impact of cloud-resource unavailability and service interruption for MCC clients is more severe than stationary clients connected to the wall power and fixed

network. Frequent suspension of energy-constraint mobile clients due to resource scarcity, not only shrinks usefulness of cloud outsourcing for MCC end-users, but also divests privilege of intensive computation anytime, anywhere from mobile users.

Therefore, several challenging tasks (e.g. resource provisioning without service interruption, quick disaster recovery, and high service availability) need to be realized since service unavailability and interruption prolong execution time, increase monitoring overhead, and deplete smartphones' local resources, especially battery. Solutions such as Reservoir (Rochwerger et al., 2011) and using different kind of resources can be employed to expand cloud resources on demand with main focus on mobile clients.

2.7.6 Mobile Communication Congestion Issues:

Mobile data traffic is tremendously hiking by increasing user demands for exploiting cloud resources which impact on MNOs. Data storage/retrieval, application offloading, and live video streaming are examples of cloud-mobile operations that drastically increase traffic, leads to excessive congestion and packet loss.

Furthermore, employing MCC in several domains such as VANET, wireless sensor networks, and M2M communications lead to new research domains such as vehicular cloud computing (Whaiduzzaman, Sookhak, Gani, & Buyya, 2014) and further increase data volume across the network. Hence, managing such huge data becomes challenging, especially when offloading mobile data are distributed among helping nodes to commute to/from the cloud.

Although utilizing heterogeneous wireless spectrum (Min, Zhang, Choi, & Shin, 2012) and leveraging regional hotspots as helping nodes (to relay traffic at peak hours) can contribute to smooth traffic, several decisions need to be made like how efficient is relaying offloading data packets? How secure are helping nodes and in what extend security

of shifted MCC data would be protected? How latency of data migration to other node impacts on interactive applications' responsiveness? Such kinds of questions, not only necessitate intelligent systems to manage offloaded MCC data, but also might alter systems' overall architecture and network structure. Because, sharp MCC data hike, necessitates cost-effective, efficient deployment of infrastructures (e.g. hotspots) and innovative strategies with least overhead and latency.

Therefore, a cognitive system within MCC that likely pre-identifies congestion issues and considers factors like MCC application types (e.g. data-intensive, computation-intensive, and communication-intensive) to determine the best action(s) to relay traffic, is imperative as the future research direction.

2.7.7 Trust, Security, and Privacy Issues:

Trust is an essential factor for the success of the burgeoning MCC paradigm. Constructing a trustable, secure environment is an open issue which is exacerbated when the Internet is utilized as the bridge between front-end and back-end devices (over wireless and wired networks). Provisioning security and providing data integrity and reliability beside delivering essential services (e.g. always on connectivity and cloud services) over the heterogeneous distributed systems, wireless networks, and the Internet require novel lightweight methods. Trust establishment based on the service provider's reputation (i.e. cloud, mobile, and Internet provider) and aggregation of trust from each service node would be a valuable approach that requires future research.

Privacy is exclusively a big issue in the vast convergence of several network technologies, which is exacerbated when cloud users trust the cloud providers and store sensitive information on public data warehouses. Hence, absorbing user trust is an important criterion leading to yet another challenge, that of how cloud service providers can ensure

confidentiality of user information.

2.8 Conclusions

In conclusion, mobile cloud computing (MCC) is aimed to provide rich functionality for resource-intensive mobile applications by leveraging cloud computing. In this chapter, we presented an overview of MCC and discussed dimensions of heterogeneity in MCC. Also, some of the major MCC problems are described based on literature. It was argued that MCC is a more heterogeneous domain compared to cloud computing due to amalgam of computing (mobile computing and cloud computing) and networking technologies. According to the types of heterogeneity in each landscape, we categorized heterogeneity of cloud computing, mobile computing, and wireless networks into two classes, namely vertical and horizontal. The taxonomy of heterogeneity in MCC was also devised. Our investigation results unveil that dissimilar platform performances (i.e. CPU performance in mobile devices as well as cloud servers) beside differing power consumption and bitrate of heterogeneous wireless technologies can affect the overall performance of remote processing approaches and mechanisms.

The literature advocates that there are several academic solutions for executing compute-intensive mobile applications by leveraging different classes of granular cloud-based resources; Coarse-grained cloud resources feature high scalability and low localization that originates network latency, medium-grained resources provide medium scalability and locality breeding network and computing latency, and fine-grained resources offer low scalability and high localization that leads to computation latency. Such network and computation latencies negatively impact on energy efficiency and response time of compute-intensive mobile applications leading to mobile application performance degradation. Considering importance of time- and energy-efficiency of compute-intensive mobile applica-

tions like m-health and m-learning, we investigate and analyse the network and computation latency problems in-order to propose a new MCC framework to alleviate these problems.

CHAPTER 3

PERFORMANCE ANALYSIS OF MOBILE COMPUTATION OUTSOURCING USING VERTICALLY HETEROGENEOUS GRANULAR CLOUD RESOURCES

In this chapter, we aim to analyse the performance of compute-intensive mobile applications when outsourcing to heterogeneous granular cloud-based resources. Benchmarking is employed to investigate the impact of latency of vertically heterogeneous cloud-based resources in MCC. Using series of benchmarking experiments, we demonstrate the impact of computing and communication latencies of performing compute-intensive mobile applications utilizing coarse-, medium-, and fine-grained cloud-based resources in MCC. The results of analysis and synthesis are presented and findings of our experiment are discussed.

The structure of this chapter is as follows. Section 3.1 presents the testbed used for benchmarking analysis. Our benchmarking model is described in Section 3.1.1 and the data design is presented in Section 3.1.2. The results of this experimental benchmarking are reported in Section 3.2 and the chapter is concluded in Section 3.3.

3.1 Benchmarking

In this section,we describe experimental model, mobile client and cloud servers specification, communication infrastructure/medium, and data design. Data design including performance metrics, prototype applications, workloads, data generation process, and data collection apparatus are presented. Here, we analyze the impact and verify the severity of computation and communication latencies in vertically heterogeneous MCC.

Considering location granularity in cloud service provider, we select three vertically heterogeneous cloud service providers from different worldwide location featuring var-

ied granularity levels. The computing services of Amazon Web Service (EC2) are used for vertically computation outsourcing computing in which the resources placed in varied level of location based on mobile node position. California, Ireland, and Singapore instances are used for the testbed.

3.1.1 Benchmarking Model

In this analysis the two testbed modes are named as local execution mode and vertical execution mode. We test our compute-intensive prototype application in these two modes. In local execution mode, all the application components are executed locally inside mobile device, whereas in vertical executions modes, the intensive component of the prototype mobile application are called for execution into (i) fine-grained cloud-based resource, (ii) medium-grained cloud-based resource and (iii) coarse-grained cloud-based resource. It is noticeable that the classification of these three granular resources is done based on cloud-mobile user's position. In vertical execution mode, intensive computations are performed inside the vertical resources and results are returned back to the mobile client for integration. Following we specify all components used in both modes.

Our graphical representation of the benchmarking model is depicted in Figure 3.1 and its components are described as follow.

3.1.1 (a) Mobile Client

The mobile device used in this experiment is a HTC Nexus One featuring Qualcomm QSD8250 Snapdragon 1 GHz Scorpion processor, 512 MB RAM, Wi-Fi 802.11 a/b/g, running Android v2.3.4 with API level 10. The software components we deploy in our mobile device are described as below.

Computing Outsourcing Engine: Computing Outsourcing Engine (COE) receives the name of the required service from the user application and searches the database to

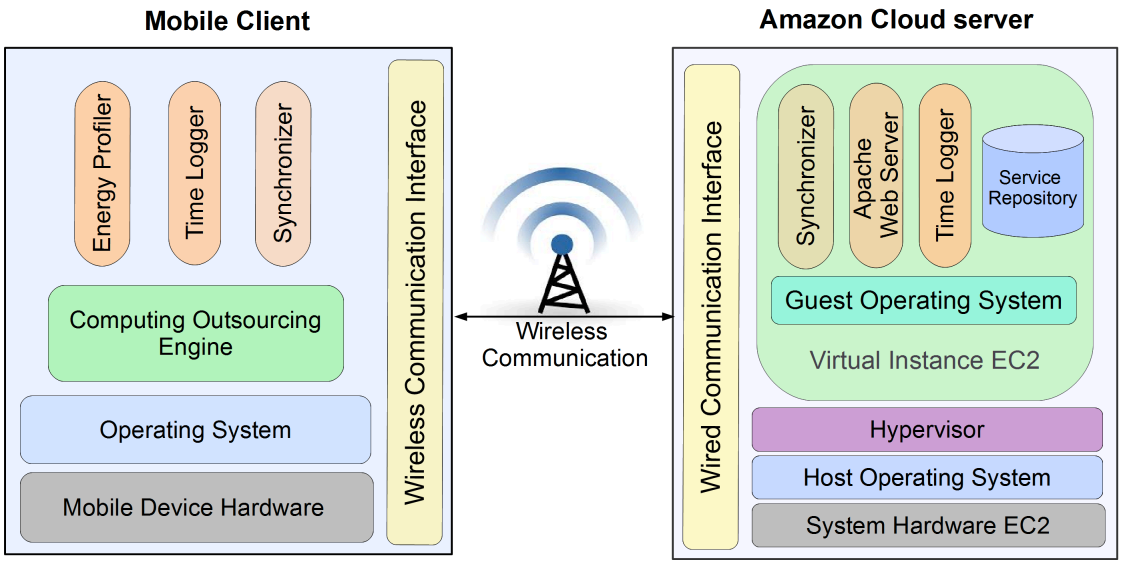


Figure 3.1: Schematic representation of benchmarking model

determine the IP address of the outsourcing cloud VM (Amazon). The database in the mobile device deposits the IP addresses of all available cloud VMs. When the execution of the application reaches the compute-intensive tasks, the execution pauses and the request is sent to the COE to determine the IP address of the cloud server. The COE marshals the user data and corresponding IP address of the outsource server as a request and sends it to the cloud VM (Amazon). COE monitors the remote execution till it receives the response from the server to integrate it into the system. Upon outsourcing failure, the outsourcing task is terminated and the error message is issued to avoid wrong data collection.

Wireless Communication Interface: Wireless Communication Interface (WLCI) is the communication interface of our mobile client node that receives the request from COE and forwards it to the cloud VM. Upon successful execution, the WLCI gets the response from the cloud VM and sends it to the Synchronizer component for the integration of the results into the client node.

Synchronizer: The Synchronizer aims to maintain integration of the memory state of the client node and the cloud VM. Synchronizer collects the response from the cloud server and reintegrates it into the native application so the COE can present the results to

the end-user.

Time Logger: Time Logger is an internal clock that generates the round-time of the application. The generated round-trip time data are stored in local storage for analysis of the system performance.

Energy Profiler: Energy profiler contains PowerTutor 1.4[1] tools that is a predominant energy collection tool for Android-based smartphones. Before the execution of the application, the tool runs in the mobile device and keeps collecting energy data while execution continues. After complete execution, we stop the PowerTutor application and save the log file for future processing.

3.1.1 (b) Cloud-based Resources

In this analysis, we built our server side using the following components. On top of the bared metal hardware and host operating system in cloud, several layers and components exist which are explained below.

Wired Communication Interface (WCI): Wired Communication Interface (WCI) is the communication interface of the cloud VM. The client request is delivered to the server VM via this interface point. The request is forwarded to reach the Apache Web server for execution. On successful execution of the compute-intensive task in the VM, the WCI receives the results of the execution and transfer it to the client device.

Hypervisor: Hypervisor or VM manager is a cloud-side application that manages the creation, execution, and destroying of the VMs. Maintaining cloud services is feasible via Hypervisor only. Hypervisor builds a layer over the host operating system to provide virtual processing infrastructures to the mobile service consumers.

Virtual Machine Instance In order to provide computing powers in our servers,

[1]http://ziyang.eecs.umich.edu/projects/powertutor/

we utilize three heterogeneous granular VM instances featuring varied computing powers located in dissimilar locations with different proximity levels that are described as follows. Technical specifications of our coarse-grained, medium-grained, and fine-grained VMs are described as follows and tabulated in Table 3.1.

The coarse-grained server located in Ireland is a m1.large VM instance of Amazon EC2 featuring 4 Elastic Compute Units (ECU) processor (Intel Xeon) with 2 vCPU cores, 7.5 GB RAM, running Microsoft Windows Server 2008 Base 64-bit OS.

The medium-grained server is a cloud m1.medium VM instance of Amazon EC2 featuring 2 ECU processor with 1 vCPU cores, 3.75 GB RAM, running Microsoft Windows Server 2008 Base 64-bit OS, which is located in California.

The fine-grained server is a cloud t1.micro VM instance of Amazon EC2 featuring variable ECU (variable*5)[2] up to 2 ECUs with 1 vCPU cores, 615 MB main memory, and low-performance I/O interface, running Microsoft Windows Server 2008 Base 64-bit OS, which is located in Singapore.

The actual computing in VMs is performed with the help of Virtual Processing Unit (VPU). The operating system installed in the VM, can only see the VPUs and allocate them to the task for execution.

Synchronizer: This component is built to communicate with the relevant components in the mobile side to ensure integrity of the native code and data in client and the execution in the cloud server. Synchronizer forwards the processing data to the mobile device for reintegration. In the absence of synchronizer component, the application and data integrity is compromised.

Time Logger: Time Logger is a timer installed in the cloud VM to generate the time

[2]It allows the instance to operate at up to 2 EC2 ECUs (one ECU provides the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor)

Table 3.1: Technical specifications of grained cloud resources used in benchmarking analysis

Device	VM Instance	ECU	vCPU Cores	RAM	OS
Coarse-grained Resource	m1.large	4	2	7.5 GB	Win Server 2008 64-bit
Medium-grained Resource	m1.medium	2	1	3.75 GB	Win Server 2008 64-bit
Fine-grained Resource	t1.micro	up to 2	1	615 MB	Win Server 2008 64-bit

data when computations perform in the VM. This low-footprint software timer produces the time for each successful computation in the cloud VM.

Apache Web Server: Apache web server is deployed in our remote servers to host the web services and manage their execution. Using the web server, our servers will be able to listen to the specified port for call from the mobile device to receive the request and initiate its execution. The apache is being used because of its lightweight nature and open source license.

Service Repository: Similar to the Universal Description Discovery and Integrity (UDDI) in SOA-based systems that stores web services for public usage, we have deployed a local service repository that contains core service codes and their required libraries for successful execution inside the remote VMs.

3.1.1 (c) Wireless Communication

We perform wireless communication using a Cisco Linksys WRT45GL wireless router feature firmware 4.30.16 build 6. The wireless link speed is 54 Mbps and indoor communications take place via 2.4 GHZ band. The router is connected to a high speed LAN network to communicate with medium and coarse grained remote servers.

3.1.2 Data Design

This part presents the in-detail description of our performance metrics, prototype applications, workload values, and data collection procedures.

3.1.2 (a) Performance Metrics

Application Response Time (ART), Application Computation Time (ACT), and Consumed Energy (CE) are the most important performance metric for typical mobile applications which are used in this experiment. The metrics are summarized in Table 3.2 and described as follows:

Application Computing Time (ACT) is the computing time to perform the desired task over the given workload in millisecond (ms). ACT excludes the communication time between client and server.

Application Response Time (ART) is the total time from initiation stage to completion stage of the prototype application for one workload in millisecond (ms). Unlike ACT, ART covers both the ACT and Communication Latency (CL).

Consumed Energy (CE) is the total energy consumed to complete entire prototype application for a workload that is presented as millijoule (mJ).

Table 3.2: Performance Metrics Analysed in This Experiment

Performance Metrics	Unit
Application Computation Time (ACT)	ms
Application Response Time (ART)	ms
Consumed Energy (CE)	mJ

3.1.2 (b) Prototype

The prototype application is a power math application that receives two values of base and exponent to calculate the value of $base^{exponent}$. The prototype application is developed using jQuery mobile 1.2.1 and PhP and runs in client-server architecture. In server side, the Apache v2.4.4 web server is running on top of the OS and port 80 is utilized to perform Hypertext Transfer Protocol (HTTP) client-server communication over the network. For execution on the mobile device, PhP v6.4.11 is installed that manages communication and execution over port 8080.

3.1.2 (c) Workloads

Benchmarking workloads include 30 different workloads in three major intensity levels of low, medium, and high. The workload values are summarized in Table 3.3.

3.1.2 (d) Data Collection Tools

Data collection tools are designed carefully to minimize the man-made mistakes. Data of time nature (ACT and ART) are collected using automatic loggers which start before execution and ends after ending the task.

Energy data are collected using auto logging feature of the PowerTutor 1.4 tool which is an open source tool for Android handsets.

3.2 Results and Discussion

The results of our experiment are presented in this section in two parts. Firstly, in time results we present the results of ART and ACT. Secondly, we present the results of energy analysis.

Table 3.3: 30 workloads analysed in this experiment

Workload #	Intensity	Request	
		Base	Exponent
1	Low	999	3000
2		999	3111
3		999	3222
4		999	3333
5		999	3444
6		999	3555
7		999	3666
8		999	3777
9		999	3888
10		999	3999
11	Medium	999	13000
12		999	13111
13		999	13222
14		999	13333
15		999	13444
16		999	13555
17		999	13666
18		999	13777
19		999	13888
20		999	13999
21	High	999	23000
22		999	23111
23		999	23222
24		999	23333
25		999	23444
26		999	23555
27		999	23666
28		999	23777
29		999	23888
30		999	23999

3.2.1 Time Results

Tables 3.4 and 3.5 summarize descriptive analysis of our results including minimum, maximum, and mean values of ART and ACT, respectively. In each Table, results are categorized in four different intensity levels of low, medium, high, and mean (mean of all three intensity levels) for four different execution modes. As results suggest, performing remote execution process using Singapore cloud is significantly beneficial compared to the local execution and execution using medium- and coarse-grained cloud resources like California and Ireland Clouds. Leveraging fine-grained Singapore cloud reduces the ART up to 42.4% for all intensity levels. The minimum time saving using Singapore cloud for low, medium, and high intensity workloads are 20%, 37.6%, and 47.8% respectively. The maximum ART reduction for low, medium, and high intensity levels are 35.3%, 46.2%, and 42.4% respectively.

Utilizing California and Ireland clouds are not beneficial due to the long WAN latency

Table 3.4: Descriptive analysis of benchmarking ART results

Workloads Exponents	Resources	Minimum (ms)	Maximum (ms)	Mean (ms)
3000-3999	Local ART	175.3	269.9	213.84
	Singapore ART	139.1	174.6	147.96
	California ART	1061.2	1138.9	1093.49
	Ireland ART	1344.4	1654.7	1516.46
13000-13999	Local ART	814	1141.9	992.61
	Singapore ART	507.8	614.4	542.54
	California ART	1541.3	2012.2	1628.58
	Ireland ART	2108.6	2227.9	2193.72
23000-23999	Local ART	2092	2303.5	2238.72
	Singapore ART	1091.6	1326.3	1258.43
	California ART	2225.8	2297.9	2250.27
	Ireland ART	2838.5	3407.1	3051.47
Mean	Local ART	175.3	2303.5	1148.39
	Singapore ART	139.1	1326.3	652.98
	California ART	1061.2	2297.9	1657.45
	Ireland ART	1344.4	3407.1	2253.88

between the mobile client and cloud VM. Using California resources for low, medium increases the ART about 401%, 64%, and almost no impact on ART in high intensity workloads compare to local execution. The WAN latency of using such resources is significantly higher for low intensity workloads and this negative impact decreases as workloads increase. Similarly, using Ireland resources for low, medium, and high intensity workloads increases the ART about 609%, 121%, and 36% more than local execution which is worse that California instance due to the excess distance. These tendencies suggest that utilizing coarse-grained resources for low intensity workloads not only does not save ART, but also significantly increase it. Leveraging coarse-grained cloud resources for extensively intensive tasks is feasible to be beneficial.

In order to demonstrate the above reported ART prolonging, we have presented the ACT in Table 3.5 and communication latency in Table 3.6. The results in Table 3.5 shows that computation time for all workload intensities is remarkably lower when performing

Table 3.5: Descriptive analysis of benchmarking ACT results

Workloads Exponents	Resources	Minimum (ms)	Maximum (ms)	Mean (ms)
3000-3999	Local ACT	144.4	206	168.86
	Singapore ACT	35.4	54.7	44.824
	California ACT	30.4	64.3	50.266
	Ireland ACT	21.2	33.2	27.10
13000-13999	Local ACT	768.8	1089.7	942.633
	Singapore ACT	365.7	413.9	388.910
	California ACT	320	373.8	339.11
	Ireland ACT	220	257.3	243.16
23000-23999	Local ACT	2036.6	2251.5	2183.89
	Singapore ACT	941.2	1003.4	971.028
	California ACT	814.4	864.4	840.714
	Ireland ACT	570.2	597.7	586.872
Mean	Local ACT	144.4	2251.5	1098.457
	Singapore ACT	35.4	1003.4	468.25
	California ACT	30.4	864.4	410.03
	Ireland ACT	21.2	597.7	285.714

remote execution that are more powerful from the mobile device. However, in contrast with ACT results, the results of Table 3.6 shows significant difference in WAN latency of utilizing heterogeneous granular cloud VMs. Utilizing Singapore originates minimum of 102.9*ms* overhead for the low intensity level while it is 997*ms* and 1317*ms* for California and Ireland. In average utilizing Ireland resources yields 12 and 8.7 times more WAN latency compared to Singapore and California.

To better present the results of this study, we have plotted the ART, ACT, and communication latency results of executing 30 workloads in four different execution modes in Figure 3.2, 3.3, 3.4, and 3.5. Green checkered bars in Figure 3.2 represent the ART of local execution mode. Blue diagonally stripped bars show ART of Singapore mode, horizontally stripped bars represent California and the red diagonally back stripped ones show ART of Ireland cloud execution mode. As the bars in Figure 3.2 demonstrate, the highest execution time is always when the Ireland cloud resources are utilized followed by California. It also shows that utilizing fined-grained clouds resources (Singapore) is beneficial in all modes. Moreover, the Figure shows more anomalies in ART when using

Table 3.6: Descriptive analysis of benchmarking communication latency results

Workloads Exponents	Resources	Minimum (ms)	Maximum (ms)	Mean (ms)
3000-3999	Singapore WAN Latency	102.9	124.1	113.160
	California WAN Latency	997.0	1079.5	1043.230
	Ireland WAN Latency	1317.7	1622.5	1489.353
13000-13999	Singapore WAN Latency	135.9	200.5	153.630
	California WAN Latency	1204.4	1657.6	1289.477
	Ireland WAN Latency	1888.5	1972.6	1950.560
23000-23999	Singapore WAN Latency	150.5	347.3	287.410
	California WAN Latency	1390.2	1433.5	1409.557
	Ireland WAN Latency	2261.8	2811.5	2464.603
Mean	Singapore WAN Latency	102.9	347.3	184.733
	California ART	997.0	1657.6	1247.421
	Ireland WAN Latency	1317.7	2811.5	1968.172

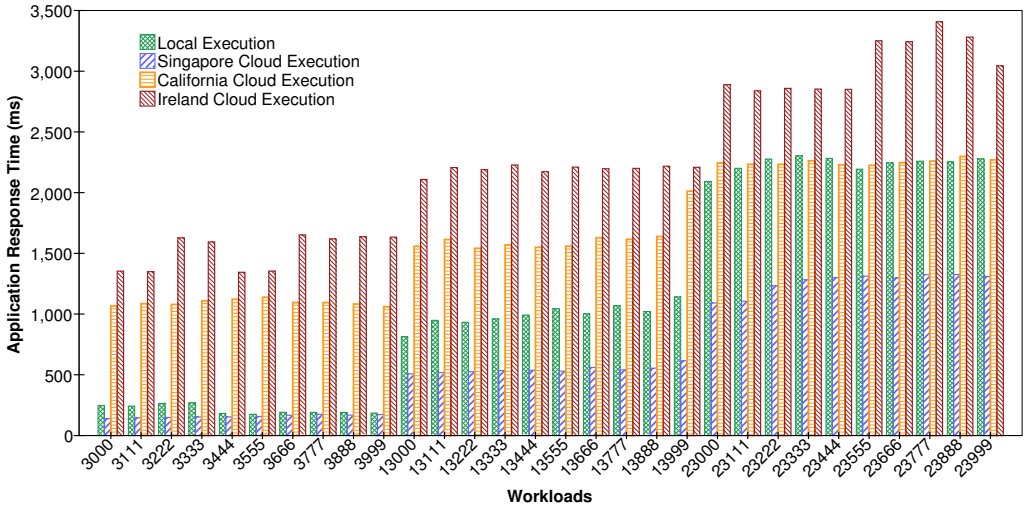


Figure 3.2: Application response times of 30 workloads in 4 execution modes

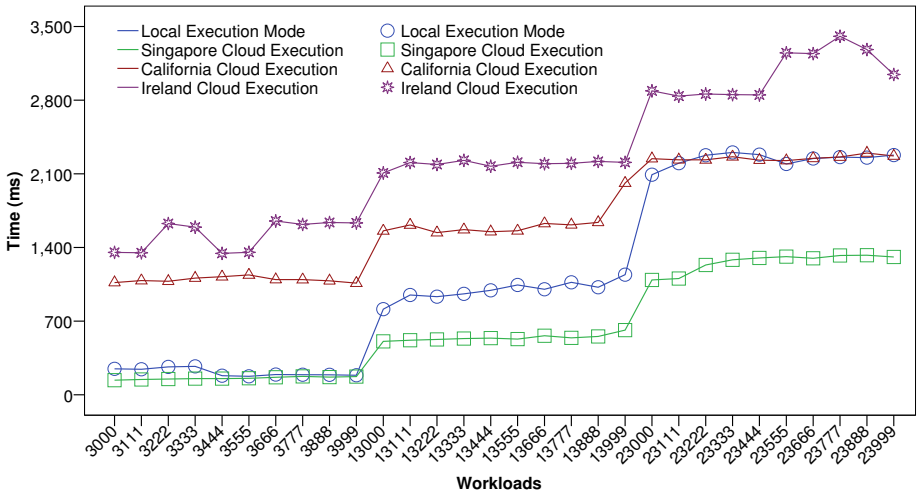


Figure 3.3: Comparison of application response time for 30 workloads in 4 execution modes

Ireland and California while there is no visible anomaly in Singapore mode.

Figure 3.3 demonstrate the scattered diagram of ART with interpolation lines to better compare the ART in different execution modes. For the sake of data collection reliability, we have 30 times repeated the execution of each workload. Each bar in these charts represents mean value of 30 iteration of each workload's execution.

Results of Figure 3.4 show ACT in four execution modes. As expected the ACT is the highest in local execution modes and is lowest in Ireland cloud due to their native computing power. Among cloud VMs, the Ireland is the most powerful, followed by

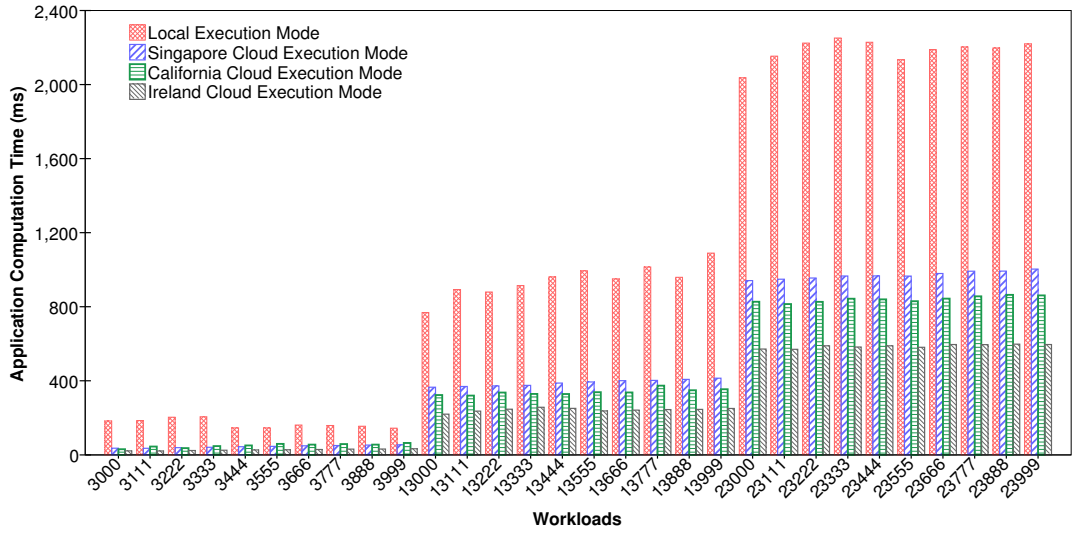


Figure 3.4: Application computing time of 30 workloads in 4 execution modes

California and Singapore clouds. However, utilizing Ireland resource is not yet beneficial despite computing superiority compared to other resources. The communication latency of utilizing varied resources are drawn in Figure 3.5. The gray chattered bars represent the I/O delay of mobile device when running the application locally. The red diagonally stripped, green horizontally, and purple back diagonally stripped bars are communication latency of using Singapore, California, and Ireland VMs, respectively.

The results clearly demonstrate highest latency in Ireland cloud followed by Califor-

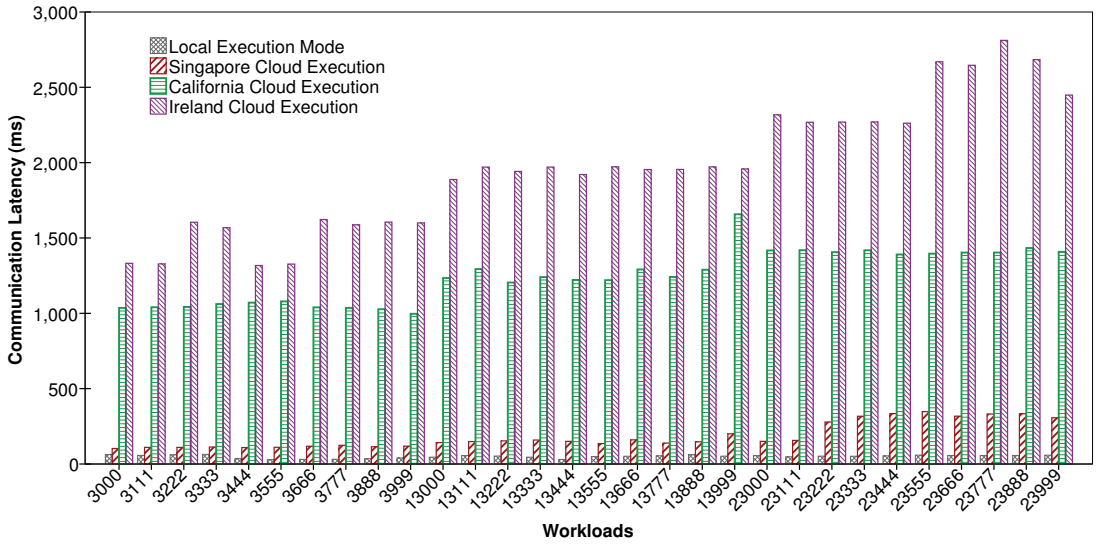


Figure 3.5: Communication latency of 30 workloads in 4 execution modes

nia. Even the WAN latency of utilizing Singapore resources is more than I/O latency of using local resources.

In order to demonstrate the proportions of ACT and communication latency in ART, we have drawn stacked bar charts for each intensity level in Figures 3.6, 3.7, 3.8. In each bar, the blue chunk shows the ART for 10 workloads. The green chunks in each bar shows the computing latency and the beige color part represents the communication latency. Bars in Figure 3.6 show the results of low intensity workloads. In these workloads the computation are minimal and hence the green layer is very small compared to the communication latency.

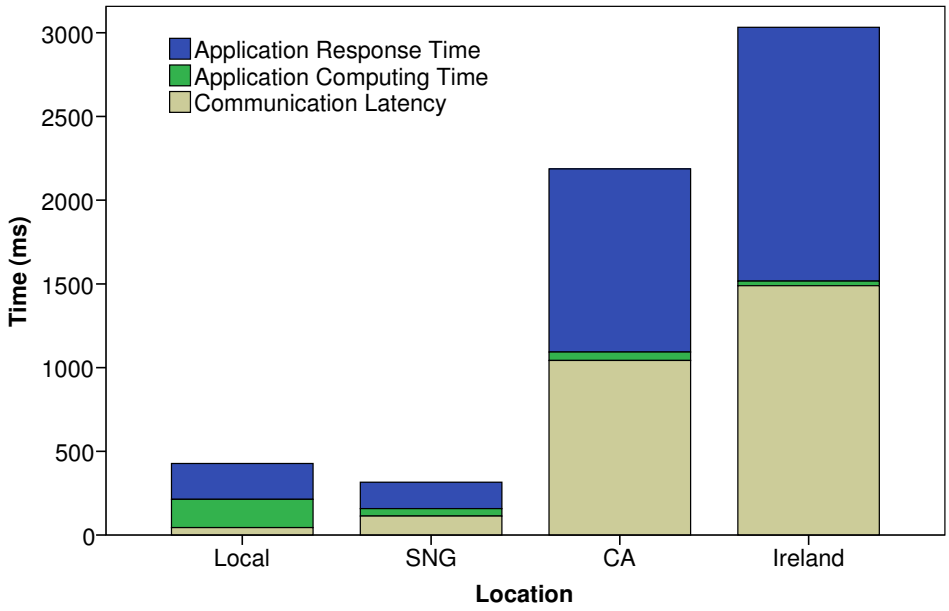


Figure 3.6: Time comparison of ART, ACT, and CL for low intensity workloads in 4 execution modes

However, results of medium intensity workloads depicted in Figure 3.7 shows the difference in computing capabilities of different execution modes. The least communication latency and the most computing delay belongs to the local execution mode whereas the most communication latency and the least computing delay is for the Ireland cloud. The graphical comparison of the green chunks in these four bars illustrates the computing

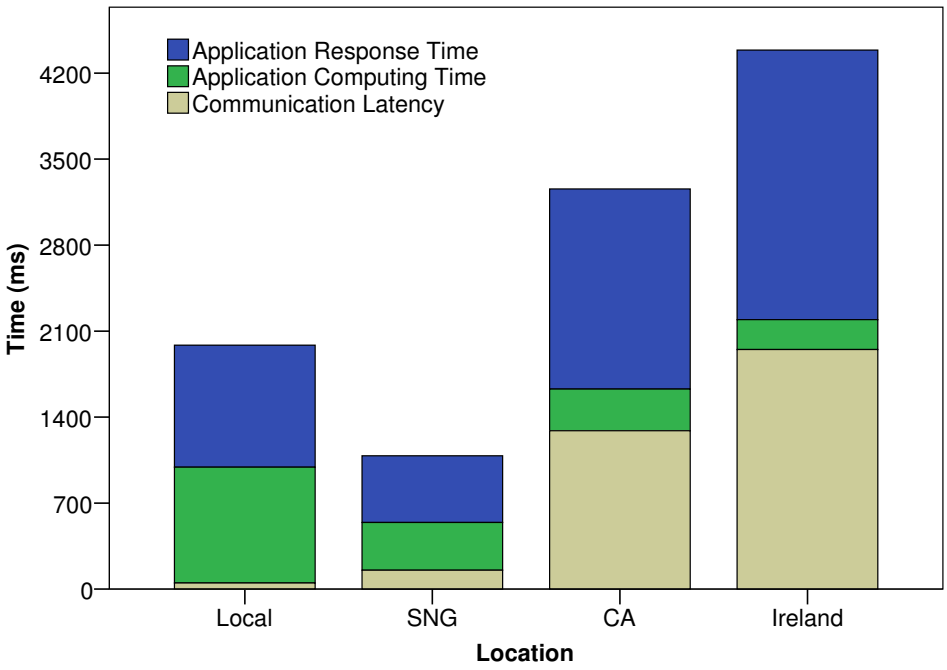


Figure 3.7: Time comparison of ART, ACT, and CL for medium intensity workloads in 4 execution modes

performance of local, Singapore, California, and Ireland computing devices. Such information are better visible in Figure 3.8 where the results of high intensity workloads are plotted. As the workloads increase the application computing time also increase, which demonstrate increasing computing complexity of workloads and computing performance of different servers.

Comparison of the results in Figures 3.6, 3.7, 3.8, advocates that coarse-grained resources can reduce the computing latency of remote execution. However, the benefit of ACT reduction is jeopardized in most of the cases due to the high WAN latency of sending request and receiving response to the distant coarse-grained cloud VMs. Therefore, we can conclude that despite their computing power, utilizing coarse-grained resources is hindered by long communication latency associated with them. This analysis suggests feasibility of the computing-communication trade-off when utilizing cloud-based resources in MCC.

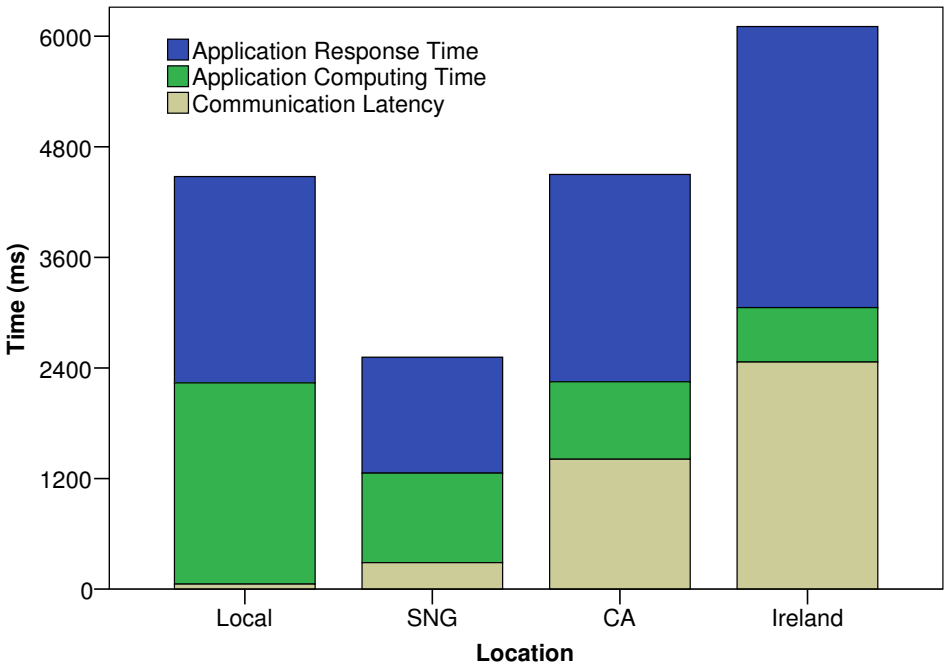


Figure 3.8: Time comparison of ART, ACT, and CL for high intensity workloads in 4 execution modes

3.2.2 Consumed Energy Results

Results of our energy analysis are presented in Tables 3.7. The table summarizes descriptive analysis of our results including minimum, maximum, and mean values of consumed energy in four execution modes. Results are categorized in four different intensity levels of low, medium, high, and mean (mean of all three intensity levels).

As the results suggest, performing remote execution process using Singapore cloud is always beneficial, especially for high intensity workloads compared to the local mode and executing using medium- and coarse-grained cloud resources like California and Ireland Clouds. Leveraging fine-grained Singapore cloud, reduces the CE up to 41.8% for all intensity levels. The minimum energy saving using Singapore cloud for low, medium, and high intensity workloads are 0.06%, 34.7%, and 49.6% respectively. The maximum energy consumption reduction for low, medium, and high intensity levels using Singapore cloud are 16.7%, 38.2%, and 41.8% respectively.

Utilizing California cloud is less beneficial especially in low and medium intensity

Table 3.7: Descriptive analysis of consumed energy results of benchmarking analysis

Workloads Exponents	Resources	Minimum (mJ)	Maximum (mJ)	Mean (mJ)
3000-3999	Local Execution Mode	126.0	153.9	142.66
	Singapore Cloud Mode	118.2	128.1	122.43
	California Cloud Mode	120.9	170.3	142.62
	Ireland Cloud Mode	203.1	266.8	235.87
13000-13999	Local Execution Mode	202.4	270.8	229.36
	Singapore Cloud Mode	132.1	167.2	147.71
	California Cloud Mode	201.7	215.0	206.53
	Ireland Cloud Mode	342.1	412.5	375.78
23000-23999	Local Execution Mode	312.6	372.1	346.56
	Singapore Cloud Mode	157.4	216.4	196.23
	California Cloud Mode	254.3	295.8	271.03
	Ireland Cloud Mode	597.0	690.6	640.16
Mean	Local Execution Mode	126.0	372.1	239.54
	Singapore Cloud Mode	118.2	216.4	155.45
	California Cloud Mode	120.9	295.8	206.73
	Ireland Cloud Mode	203.1	690.6	417.27

levels due to the high ART discussed earlier in this study. Using California resources for low, medium, and high intensity workloads decrease the mean CE about 0.0%, 0.1%, and 21.7% compared to local execution. Utilizing Ireland resources is not at all beneficial in this study. Ireland cloud increases the mean CE in low, medium, and high intensity levels as much as 65%, 63.8%, and 84.7% compared to local execution.

We have plotted the Figure 3.9 using the energy data of this experiment. The green checkered bars in this Figure represent the CE of local execution mode. Blue diagonally stripped bars show CE of Singapore mode, horizontally stripped bars represent California and the red diagonally back stripped ones show CE of Ireland cloud execution mode. As results in this chart highlight utilizing Ireland cloud is not beneficial for any workload and is more than the amount of energy consumed for local execution. This is due to excess communication latency of utilizing Ireland cloud that directly impacts on the ART and consequently on CE.

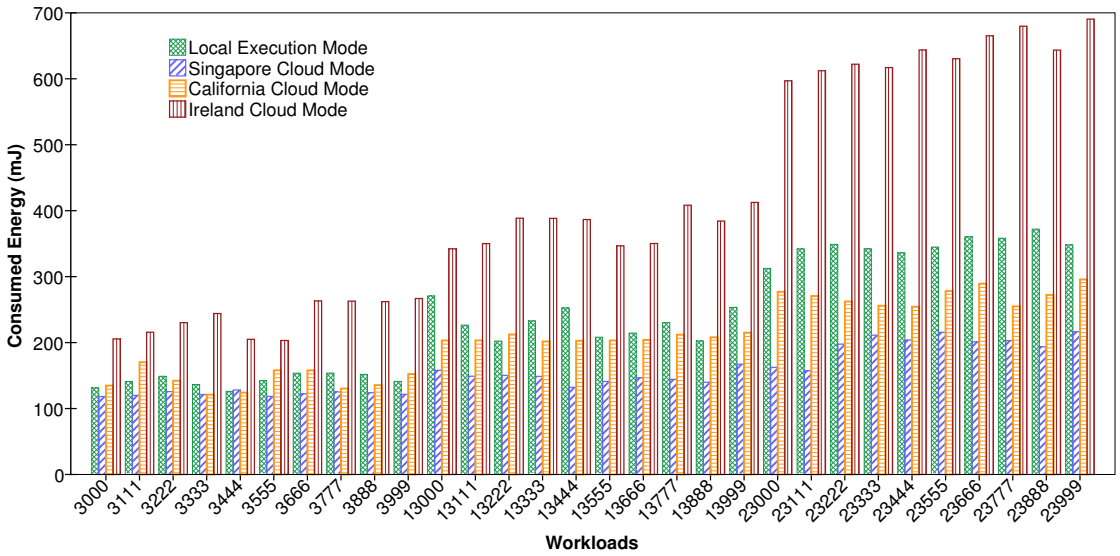
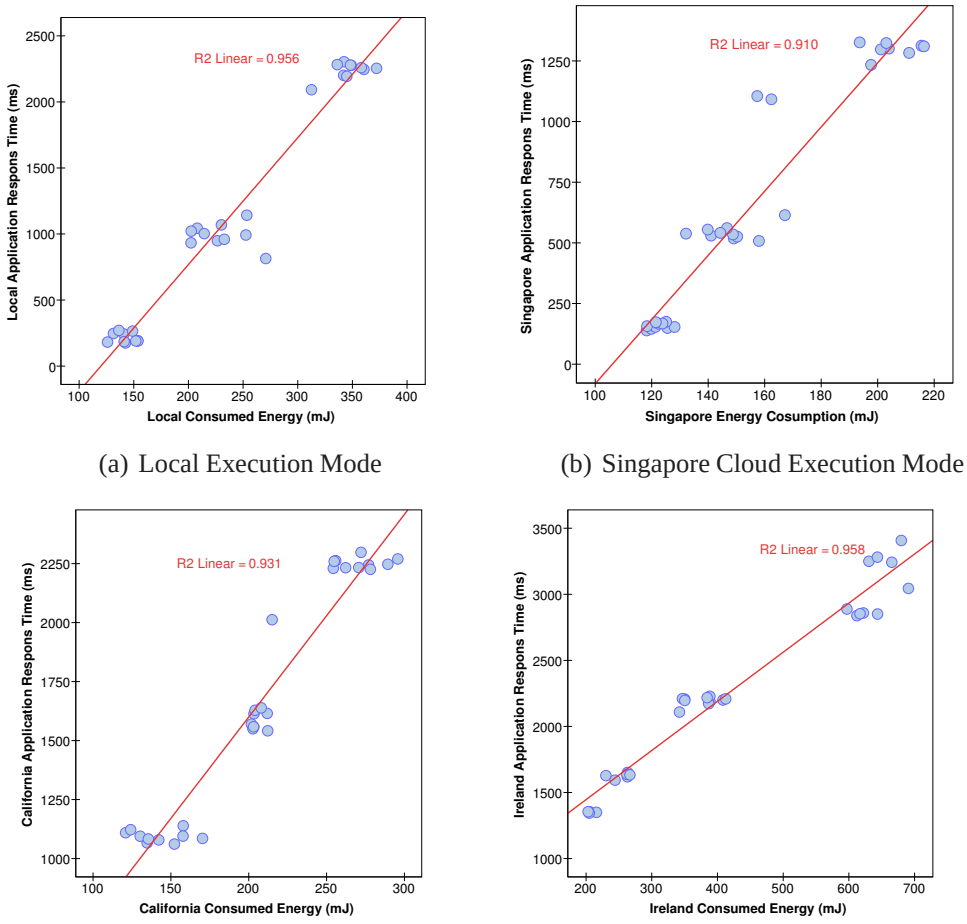


Figure 3.9: Comparison of consumed energy for 30 workloads in four execution modes

The slightly different results are evident for California cloud. In low intensity workloads, the California cloud consumes more energy than local. However, this trend reverses in medium and high latency workloads. The energy saving is at peak when using high intensity level workloads. However, using Singapore cloud is beneficial for all workloads and the achievements are increasing linearly as the workloads increase.

The results of ART and CE analysis suggest a linear correlation between time and energy. To analyze this correlation, we performed further analysis that its results are presented in Figure 3.10. Blue remarks in each chart represent the corresponding CE and ART values of 30 workloads. The red fit line and R^2 values show the correlation between the ART and CE.

As the results testify, CE of the applications when leveraging remote resources follows identical pattern similar to the ART (due to its tight direct dependency to time). Moreover, as the fit line slops indicate, for the local, Singapore, and California, the slops that represents the significance of the correlation is more sharply laid on the remarks than of Ireland cloud. Such difference in the fit line slope can interpret the implications of utilizing coarse-grained resources in mobile augmentation. Therefore, the assumption that



(a) Local Execution Mode

(b) Singapore Cloud Execution Mode

(c) California Cloud Execution Mode

(d) Ireland Cloud Execution Mode

Figure 3.10: Correlation between the ART and Consumed Energy for 30 Workloads in Four Execution Modes

utilizing coarse-grained cloud resources can generally increase CE of applications remains valid. Energy results suggest that utilizing combination of fine, medium, and coarse grain computing resources is likely more efficient and beneficial for remote execution in MCC.

3.3 Conclusions

In this chapter, we investigate the impact of utilizing fine-grained, medium-grained, and coarse-grained cloud resources on ART, ACT, and CE of compute-intensive mobile applications. We demonstrate how computation and communication latencies of utilizing heterogeneous granular resources can increase or decrease ART and CE of compute-intensive mobile applications when running on cloud-based remote resources.

The results of this investigation confirmed that though leveraging cloud-based resources can improve ART and CE of mobile applications, varied computation and communication latencies of heterogeneous granular resources remarkably impact on the outsourcing performance. Our analysis unveil that utilizing fine-grained resources featuring low scalability and high proximity originates high computing and low communication latencies. Exploiting coarse-grained resources originates high communication latency and lower computing latency compared to the fine-grained resources, since coarse-grained resources are highly scalable located far from end-users. Lastly, we found that employing medium-grained resources creates medium computing and computation latency due to their lower scalability and proximity compared with coarse-grained and fine-grained resources, respectively. Thus, inhomogeneous computations and communication latencies among vertically heterogeneous granular resources negatively impact on energy efficiency and response time of compute-intensive mobile applications leading to mobile application performance degradation.

Moreover, from the results of this analysis, it is deemed utilizing amalgam of horizontally heterogeneous granular resources composed of three classes of fine-, medium-, and coarse-grained cloud resources can yield high performance efficient of computation outsourcing in MCC by providing opportunity to perform a computation-communication tadeoff while selecting remote resources. As a result, time and energy consumption of performing resource-intensive mobile applications on horizontally heterogeneous granular computing infrastructures can be remarkably decreased. Therefore, exploiting combination of horizontally heterogeneous granular cloud-based resources can remarkably enhance performance of MCC solutions toward optimal outcome.

CHAPTER 4

LIGHTWEIGHT HETEROGENEOUS HYBRID MOBILE CLOUD COMPUTING FRAMEWORK FOR COMPUTE-INTENSIVE MOBILE APPLICATIONS

This chapter presents an in-depth explanation of our proposed heterogeneous hybrid MCC framework for compute-intensive mobile applications. The framework is consist of three major building blocks of service consumer, system arbitrator, and cloud-computational infrastructure composed of horizontally heterogeneous granular computing resources, aiming to augment computation capabilities of mobile devices, especially smartphones. In this study we assume mobile network operators (MNOs) and their dealers can play a vital role in providing computing services to the mobile users due to their functional and non-functional attributes (i.e., computing and locality). Later, we express the important of this assumption with introducing MNOs' roles, schematic representation of layered grained cloud-based resources and methods used in this framework. Significance and novelty of the proposed framework is also presented. Moreover, we present the data design for evaluation of the proposed framework.

The remainder of this chapter is as follows. Section 4.1 presents our proposed framework and describes its building blocks and components. Section 4.2 highlights the significance and novelty of the proposed framework. Data design for evaluation of the performance of the proposed framework is presented in section 4.3 and the chapter is concluded in section 4.4.

4.1 Lightweight Heterogeneous Hybrid Mobile Cloud Computing Framework

We propose a lightweight heterogeneous hybrid mobile cloud computing framework consists of horizontal heterogeneous cloud-based resources for compute-intensive mobile applications to address the inefficiencies, particularly computation and communication latency of vertical heterogeneous MCC solutions demonstrated in chapter 3.

In design and development of this framework, we have employed Resource Oriented Architecture (ROA) and cloud computing principles to generate elastic platform-neutral solution. Therefore, the cloud-based mobile applications based on this framework feature higher portability, scalability, and elasticity. In this MCC environment, every mobile application is a service-based composite application that incorporates several prefabricated services. The resource-intensive services are stored and executed outside the mobile devices on computing entities of built hybrid cloud resources.

Typical ROA-based frameworks feature three main building blocks, namely service provider, service broker, and service consumer. Service provider plays two roles of developing and delivering the services to the service consumers. However, convergence of service development and delivery necessitates acquisition of programming and software engineering skills by service providers. In order to address this limitation and enabling

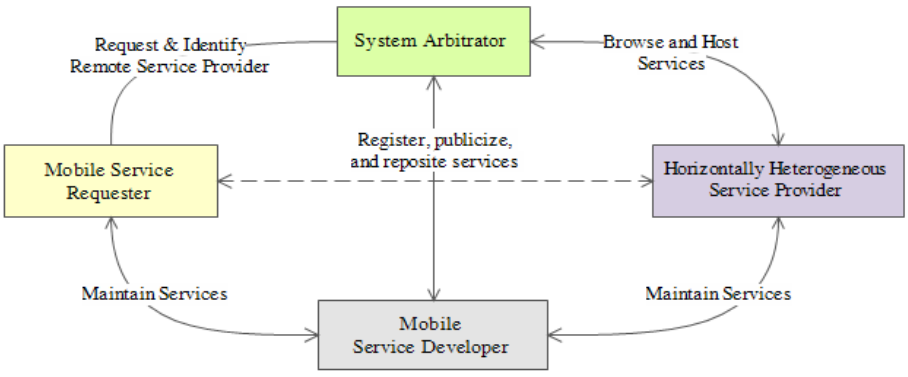


Figure 4.1: The block diagram of hybrid mobile cloud computing framework

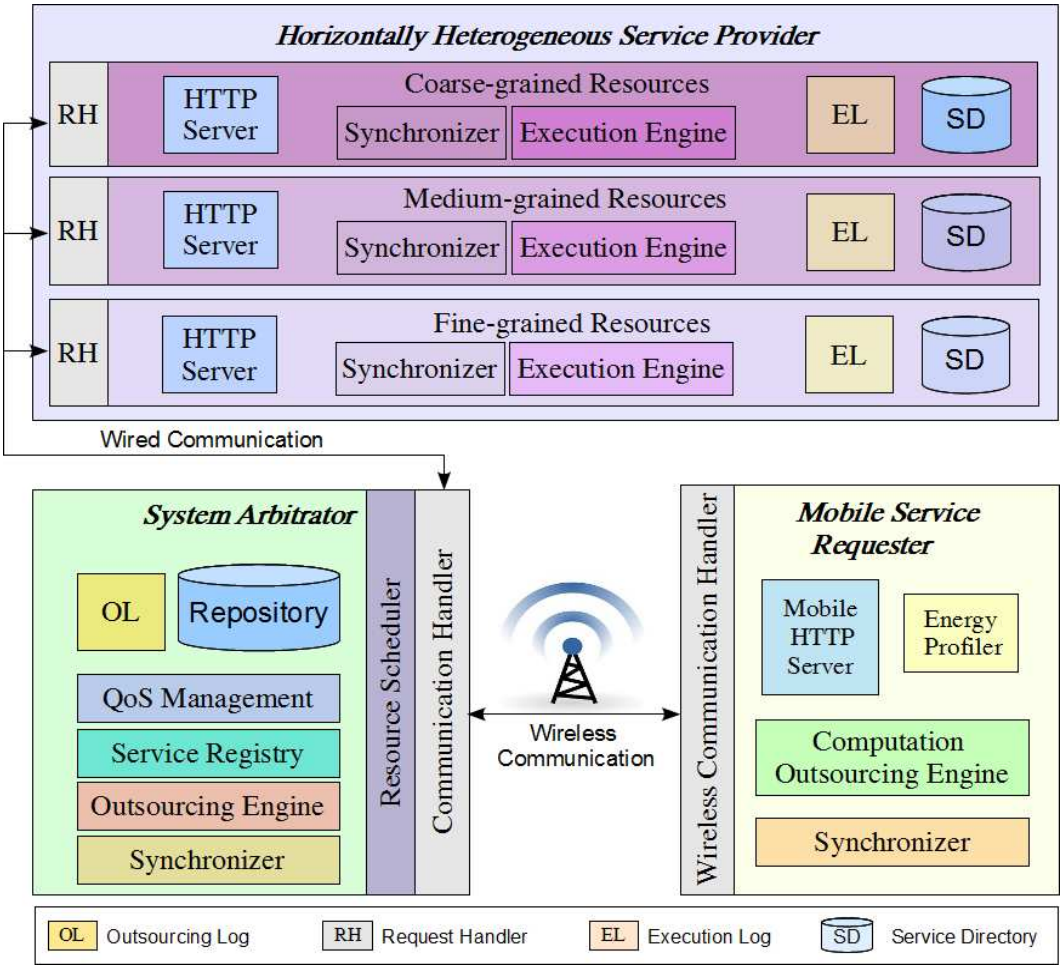


Figure 4.2: Schematic presentation of heterogeneous hybrid mobile cloud computing framework

multitudes of computing entities to host and deliver the services to the service consumers, we have separated these two roles from each other. Hence, our proposed framework features four major building blocks, including service developer, mobile service requester, system arbitrator, and horizontally heterogeneous service provider, which are depicted in Figure 4.1. Furthermore, in order to realize the vision of this framework, we have designed and developed several components for major building blocks. Figure 4.2 shows schematic presentation of our heterogeneous hybrid mobile cloud computing framework, along with its main components that are described as follows.

Figure 4.3 illustrates the sequence diagram of operations among major building blocks in our framework to perform successful outsourcing.

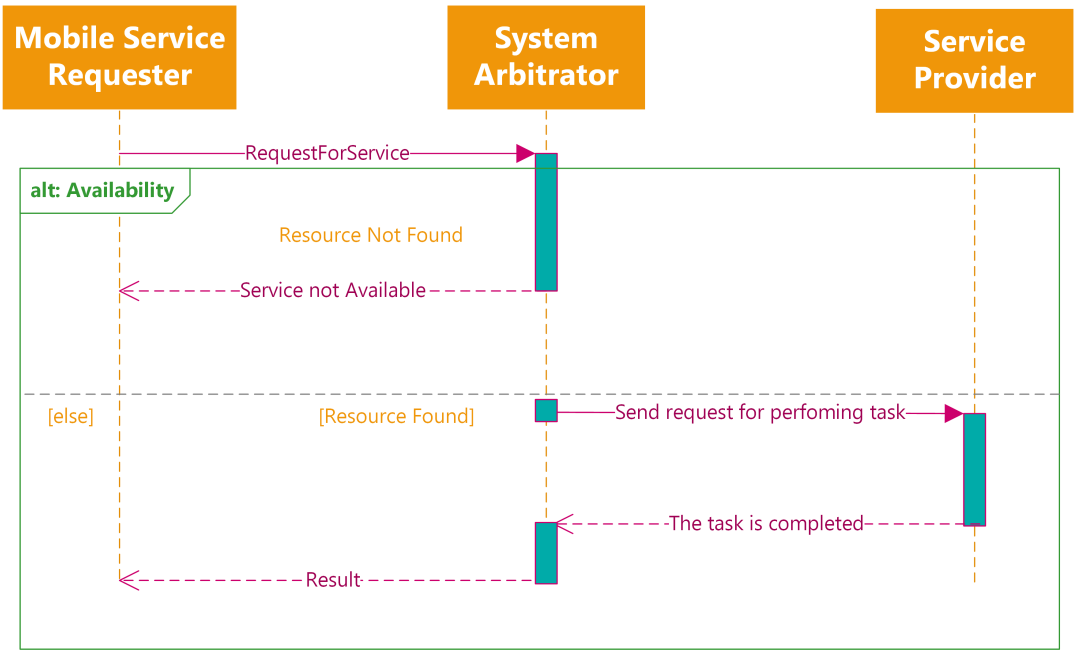


Figure 4.3: Sequence diagram of operations among main building blocks in proposed framework

4.1.1 Service Developer

The service developer building block is responsible to develop heterogeneous granular computing ROA-based services (or services in short) and utilize them to build loosely coupled complex applications based on pre-fabricated services. ROA-based services are platform-independent and can be hosted and executed on any resource-rich computing device that features web service execution. Developers implement heterogeneous granular services and contact the central arbitrating entity (called System Arbitrator) to register and publicize their services. Moreover, service developers are responsible to update and maintain services to ensure consistency, integrity, and availability of services. Though service developers are still capable of hosting and providing services to the consumers, service provisioning is separated from the developers and they are treated as two different entities in this framework.

4.1.2 Horizontally Heterogeneous Service Provider

Service provider in this framework hosts pre-fabricated services that are built by application and service developer to provisioning computing services on-demand based on the cloud computing principles. The separation of service development and provisioning enables non-technical service providers to conveniently host and provision the required services on-demand without software engineering expertise. As illustrated in Figure 4.1, hosted services in service provider are continuously monitored by the service provider for frequent update and maintenance.

Service providers in this framework are horizontally heterogeneous hybrid granular computing entities that feature varied computing elasticity, multiplicity, and different proximity levels to the service consumers. Thus, efficiency of service delivery is improved in the presence of numerous horizontally heterogeneous service providers that can fulfil varied quality requirements of different mobile service requesters. In the following, we explain these heterogeneous hybrid granular computing entities.

Hybrid Cloud Resources (HCR): The Hybrid Cloud Resource (HCR) in this framework are 3-tier complex resources that creates a horizontally heterogeneous granular computing environment. In this research, these 3-tier resources are considered as coarse-grained, medium-grained, and fine-grained computing resources that are illustrated in Figure 4.4 and explained below. Resources are classified based on three parameters of locality (low as country-level, medium as city-level, fine as cell-level), scalability (low, medium, high), and multiplicity (low, medium, high).

Coarse-Grained Resources (CGRs): The first type of HCRs shown in Figure 4.2 are giant cloud service providers such as Amazon EC2. Although these resources are often located in long distance to mobile users that causes communication latency, they offer elastic resources with high computing power. Moreover, they are low in multiplicity since

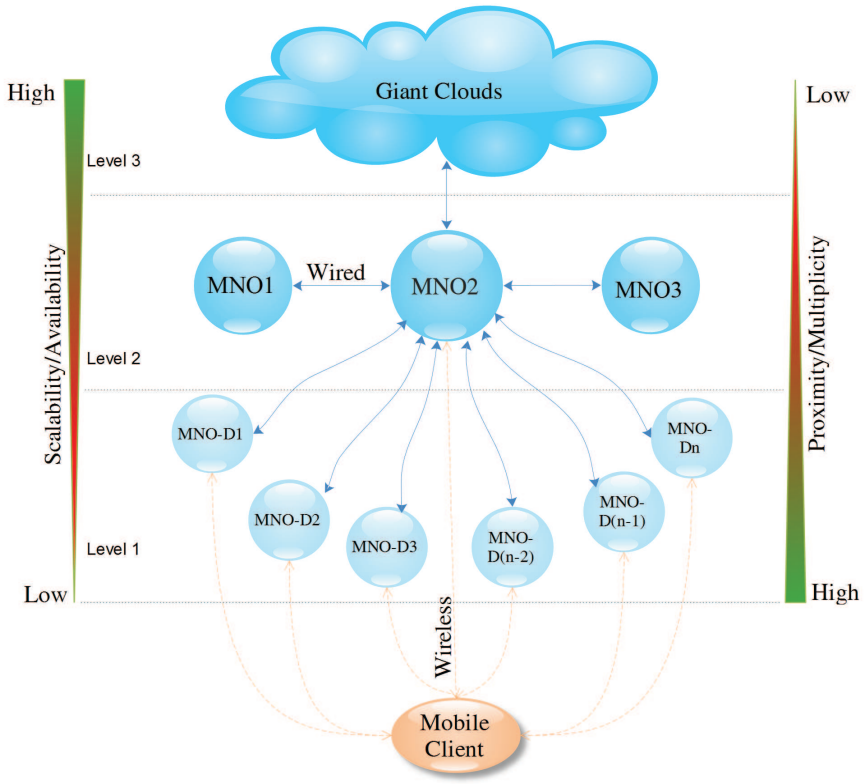


Figure 4.4: Graphical representation of multi-level heterogeneous computation outsourcing.

number of cloud datacenters is limited. These resources are located in the top most part of the Figure 4.4. CGRs feature high availability, scalability, and elasticity and typically levy low financial cost to the service consumers compared with other resources based on cloud computing principal. Utilizing these resources necessitates communication through the wired networks and the channel of Internet and accessing their resources requires passing through large number of intermediate hops that increases communication latency and degrades service utilization quality.

Medium-Grained Resources (MGRs): The medium-grained resources have usually lower processing power compared to CGRs, but can provide more proximate cloud services to mobile users. These resources have medium proximity to mobile users and hence can be considered to be at city level. We have selected MNOs as MGRs in this research

due to several uniquely beneficial features (we utilize computing capability of a desktop in the absence of MNO server). MNOs are able to provide continuous services to their client without need to access to the Internet. They are usually well-established and reputed business stakeholder that can play important roles in our framework as follows.

- *Multi-service providers:* MNOs can provide elastic, on-demand computing, storage, and network as a service to their end-users according to the negotiated service level agreement as they have already started (e.g., Verizon and AT&T) diversifying their services to mobile users by providing IaaS.

- *Cloud brokers:* MNOs can play the role of service mediators between cloud providers such as Amazon and mobile service requester. Also, they have the capability of being broker across other MNOs. In order to increase communication efficiency from latency perspective, MNOs can interoperate with each other to serve larger community of mobile service requesters.

- *Cloud consumers:* MNOs can utilize giant cloud resources if their proprietary resources are not enough to serve their clients. In this situation, they leverage cloud services like IaaS from IT giants' cloud service providers based on a negotiated service level agreement (SLA).

- *Carrier cloud:* MNOs as a carrier cloud meet the needs for connectivity, reliability, and accountability between heterogeneous wireless networks and wired systems in the cloud which are highlighted in mobile communication congestion issues in MCC domain explained in chapter 2.

- *Reputation trust:* MNOs such as BT (since 1846)[1], Telefonica (since 1924)[2], Maxis

[1] http://www.btplc.com/Thegroup/Ourcompany/index.htm

[2] http://www.telefonica.com/en/home/jsp/home.jsp

(since 1993)[3], Verizon(since 1983)[4], and NTT DoCoMo (since 1991)[5] have been existing since the beginning of cellular communication and could establish reputation trust among their end-users. Leveraging such reputed entity as the system arbitrator can remarkably enhances the system adoption.

Fine-Grained Resources (FGRs): The third resource tier in our framework is the closest cloud-based infrastructure to the mobile service requester. In this layer MNO's authorized dealers can play the role of trusted surrogates which are able to serve computation to mobile service requesters in vicinity. MNO authorized dealers are those service outlets having wall-connection and direct communication with MNOs that are scattered in this urban or rural areas to serve and support the MNOs' clients better.

Large number of authorized dealers are mainly scattered in different business spots such as shopping mall, market areas, airports, and commercial buildings where plenty of mobile users are operating. Currently Maxis, one of the largest Malaysian MNOs, has 1372 authorized dealers[6] which are noticeable number of wall-connected computing entities with uninterrupted power source and wired backend network connection. Utilizing computing infrastructures of these numerous ever-increasing nearby resources is deemed to not only decrease communication latency and execution time, but also to increase computing capability of mobile devices. However, their computing resources (i.e., CPU, memory, and storage) are limited and accessing them may be limited to the operating hours and weekdays, that necessitate a scheduling algorithm.

Utilizing horizontally heterogeneous hybrid cloud-based resources model provides a trade-off between computation and communication latency. Providing cloud services

[3]http://www.maxis.com.my

[4]http://www.verizon.com/home/verizonglobalhome/ghp_landing.aspx

[5]https://www.nttdocomo.co.jp/english/

[6]http://www.maxis.com.my/personal/latest/bpl%5Ffigurines/dealers.html

closer to the mobile user, will decrease communication latency. Here, mobile users can connect to MNO, and then MNO do the job or handle the job by its dealers or giant clouds, and finally the response returns from MNO to the mobile devices. Therefore, horizontally heterogeneous grained-resources can provide computation-communication trade-off to achieve near-optimal energy and time consumption for compute-intensive cloud-based mobile applications.

In order to realize functionality of service providers in our framework we have designed and implemented several components that are depicted in Figure 4.2 and described as follows. These low footprint components are installed on any computing device that is utilized as service provider in our framework with identical functionality.

4.1.2 (a) Request Handler

The request handler component plays the role of communication interface on service providers. The request handled by this module is prepared by system arbitrator and includes the service name, user data, and user preferences that should be processed in the cloud-based resources. All the communications in this framework follow the asynchronous trait of the ROA where restful communications takes place. Hence, the system can perform communication in background without interrupting their foreground transactions. This feature is vital for MNO dealers so they can perform their daily routine work without interaction disturbance.

This component in service providers is contacting its counterpart in arbitrator to handle inter-communications via wired network. Each arrived request should be validated by the handler to avoid resource wastage of the service providers. After passing thorough request validation, the received information from the arbitrator is transferred to the execution engine for execution. The response from the execution engine is sent to the request

handler so that the computing results of the execution can be sent back to the arbitrator.

4.1.2 (b) Execution Engine

Execution engine receives valid requests from arbitrator via request handler to perform certain service using the given name, input value, and user preferences. The execution engine is responsible to inquire the local service directory of the hosting service provider to identify the desired service. Upon successful identification, the input data is sent to the desired service loaded on top of the HTTP server. Upon successful execution of the required service the results are sent back to the execution engine to be embedded into the response. The response from execution engine is forwarded to the arbitrator via request handler.

4.1.2 (c) HTTP Server

The HTTP server is required at the time of service execution, because we have deployed ROA for design and development of our framework and applications. HTTP server is installed on top of the host OS in service provider machine and hosts the code and libraries of the loaded services. This server monitors the port number 80 (default port), receives and validates the Unified Resource Identifier (URI) of the requested service, and performs execution of received requests if validated. Upon successful execution, the results are sent to the execution engine. We have used Apache tomcat as a lightweight open-source HTTP server in this framework.

4.1.2 (d) Service Directory

Service directory is a local database that contains information and implementation of all the services that are hosted by the service provider. ROA obliges to store the code and binded library files in each service provider so that execution of requests can be started without need to transmit the code. The same codes can be used for execution without

limitation (based on agreement between arbitrator and developer). As the Figure 4.1 illustrates, the developer is communicating with the service providers to update/maintain the codes for efficient execution.

4.1.2 (e) Synchronizer

Synchronizer is designed to be deployed in all computing entities which are employed as service providers. The role of synchronizer in service providers is to synchronize the memory state and produced results between the service provider and service requester via system arbitrator. Once the execution state is saved on the persistent storage, the data is automatically synchronized with the system arbitrator and ultimately with mobile service requester. Without synchronizer component, the application integration is violated.

4.1.2 (f) Execution Log

In order to maintain the resource utilization in our service providers and adhere to the service level agreement between the service requester and provider, the execution log is embedded into the service provider machine. For each execution, the execution log component stores the resource information such as time and energy. This component is also used in our data collection task to produce the computing latency and time of the servers.

4.1.3 Mobile Service Requester

In this framework mobile service requester can be any mobile computing device including smartphone, laptop, tablet, wearable computers, car-mounted computers, and any other nomadic computing device working on battery accessible via wireless technologies that requires computational resources for execution of compute-intensive services. In the testing phase, we use smartphone as a mobile service requester.

Service requesters host applications that are built by application and service developers to perform desired computing task on the go. The execution of applications usually starts from the service requester device which hosted the application and execution continues on the device until it reaches a compute-intensive task. While execution continues, a request is sent by the requester to the arbitrator to identify an appropriate service provider capable of executing the desired compute-intensive service(s). Upon complete execution, the results are sent back to the service requester. In the following, we describe functionality of each component deployed in mobile service requester.

4.1.3 (a) Computation Outsourcing Engine

Computation outsourcing engine component is managing the entire outsourcing process in the mobile service requester device. When the execution reaches the intensive task, it collects data and user preferences to build a request and forward it to the arbitrator via wireless communication handler component. Once the results of remote computation are provided by the arbitrator, the computation outsourcing engine integrates the results into the local application with the help of synchronizer.

4.1.3 (b) Synchronizer

The synchronizer component embedded into the mobile service requester device is communicating with its counterpart in the arbitrator to ensure integrity of the results during the outsourcing process. In the absence of such component, the results of remote computation is not properly reflected into the native application.

4.1.3 (c) Wireless Communication Handler

The wireless communication handler component performs as a wireless communication interface on the mobile service requester. The typical communications handled by this component is asynchronous transmission of the request including service name, user

data, and user preferences to the arbitrator and receiving the results of remote execution. All the communications in our proposed framework undertaken asynchronously to enable mobile service consumer maintain seamless interaction with the rest of applications in the device.

4.1.3 (d) Mobile HTTP Server

Similar to the service provider that is hosting HTTP server, we have designed and developed a lightweight HTTP server that can listen to the assigned ports for local execution of services since our proposal is based on ROA. This server receives the URI of the desired service from the client part of the application for execution and validates the request. In case of successful validation the input values and memory state is provided to the service for execution.

4.1.4 System Arbitrator

System arbitrator plays the roles of a central arbitrary entity that manages the entire outsourcing operation. One of the roles arbitrator plays in our proposal is the UDDI (Universal Description Discovery and Integrations) that exist in typical SOA-based frameworks. Moreover, the overall system arbitration and supervision is taking place. Due to several key characteristics and benefits of MNOs in mobile computing, we exploit MNOs and deploy arbitrator in MNOs to arbitrate and supervise the outsourcing operations between front-end (service consumers or cloud-mobile users) and back-end (service providers) entities.

The system arbitrator in our proposal consists of eight major components, namely communication handler, outsourcing engine, service registry, resource scheduler, QoS management, synchronizer, repository, and outsourcing logger that are depicted in Figure and explained as follows:

4.1.4 (a) Communication Handler:

Communication handler in system arbitrator is the communication interface of the arbitrator that enables communications between compute-intensive mobile applications and other building blocks in this framework. Similar to the mobile service requester and service providers, the entire communications perform asynchronously so that building blocks can communicate with each other in background while system is usable by the users. Communication handler is responsible to receive the requests from the mobile service requesters through wireless network and forwards it to the destination component. It also receives response from the components and forwards it to the mobile service requester wirelessly.

Moreover, communication handler has a wired network interface that provides a communication channel to the service provider. When arbitrator identifies an appropriate service provider to provision computing resource for execution of the given task, the communication handler forwards the user data to the identified service provider for execution. It receives the response from the service provider upon completion and return to the mobile service requester after performing relevant tasks in the arbitrator.

4.1.4 (b) Service Registry:

Service registry component in this framework receives service registry requests from service developers (who has implemented the service) and maintains relevant data in a local repository inside arbitrator. The service registry mimics the roles of UDDI in typical SOA-based system. However, our service registry works using lightweight asynchronous communication style unlike heavy synchronous Simple Object Access Protocol (SOAP) style used in UDDI.

When a new service is registered in the system, the arbitrator establishes negotiation

with potential service providers (among horizontally heterogeneous servers) to identify an appropriate server for its execution (similar to white page registration in public service repositories). Upon successful server allocation, the service code and its associated libraries are transmitted to the server for future reference and the unique accessing address of the service is registered in the service directory as URI.

4.1.4 (c) Repository:

Repository plays the role of a database that can store information about all services and service providers in our framework. It also stores the code and libraries of all the services that are registered with this framework. Information stored in repository is utilized by service registry component, outsourcing engine, and QoS management to maintain the entire system.

4.1.4 (d) Resource Scheduler:

The resource scheduler in our framework plays the role of identifying appropriate service provider based on the user need and preferences (e.g., cost and time). At runtime, the request from mobile service requester is sent to the system arbitrator asking to identify the service providers that are capable to perform desired services considering current user's location and its preferences. For this component, we use priori decision making method in which user preferences are given to the scheduler in advance. To minimize the communication latency and database delay, mobile device forwards the list of all services at once asking arbitrator to provide the list of service providers that can satisfy all the desired services. So the request is an ordered array of service names and maximum execution cost, and the response is an ordered array of binding information for all services satisfying the total cost and is executed in least execution time. We assume that mobile user mobility during the application execution does not affect the server allocation deci-

sion. The service providers are ranked using lexicographic order method based on their computing cost and time. The first user priority is considered to be the computing cost of service providers and the second requirement is time (as imposed by the thesis's aim and objectives). The response is prepared so that the cost and time are satisfied for the entire services, not individual services.

In order to mitigate the search delay, the scheduler assumes that the required services are available and hence directly contacts the repository to fetch the records that fulfil the requested service and desired criteria (e.g., service type and operation cost), and performs scheduling on the fetched records. Otherwise, the arbitrator should contact the database to validate the service name before asking scheduler to find service provider which increases latency. Upon complete resource scheduling (either success or failure), the results (either 'service not found' error or the binding information) are sent to the outsourcing engine and scheduling ends. The pseudo code of resource selection is presented in Algorithm 1.

4.1.4 (e) Outsourcing Engine:

Outsourcing engine inside the system arbitrator is one of the center components of the framework. When a mobile service requester sends a discovery request for the desired service to the arbitrator, it passes through communication handler and reaches the resource scheduler component in the system arbitrator.

Once the resource identification is completed by the resource scheduler component and the appropriate resource is identified, the binding information is sent to the outsourcing engine. Herein, the outsourcing engine utilizes the binding information to make an indirect request call (on behalf of the mobile service requester) to the given resource provider asking for remote execution or it can send the binding information to the mobile service requester for direct contact (for evaluation purpose, we used the former approach).

Algorithm 1: Pseudo code of Finding Service Provider

1: Begin;
2: **Data**: $ser\_names[...] = service\ name\{s1, s2, s3, ..., sn\}$;
3: $loc = user\ location$;
4: $cost = user\ maximum\ cost$;
5: **Var:** $ser\_provider\_cost_{si} = cost\ of\ executing\ service\ i\ in\ the\ service\ provider$
6: **Result**: The array of binding information of all the service providers
;
7: Select * from database where Ser_provider_location=loc and ser_provider_cost<cost ;
8: Select all $Ser\_names[s1, s2, s3, ..sn]$ where $ser\_provider\_cost_{s1} + ser\_provider\_cost_{s2} + ser\_provider\_cost_{s3} + ...ser\_provider\_cost_{sn} < cost$ and store them in $Serv\_provider[n][s1, s2, s3..., sn], 0 < n < m$
; // m= maximum number of alternative service provider combinations
9: Find the maximum $ser\_provider\_execution\_time$ in each combination from $Serv\_provider[n][s1, s2, s3..., sn]$;
10: Sort the $Ser\_provider[n][s1, s2, s3, ..sn]$ using lexicography technique;
11: Select the minimum of identified maximum execution time of $Ser\_provider[n][s1, s2, s3, ..sn]$ and store in *Response* ;
12: Retrieve the binding information of the service providers in *Response* as the final results;
13: Forward the array of *Response* to the outsourcing engine;
14: End.

Upon completion of the service execution in the remote servers, the results are received by the outsourcing engine. Outsourcing engine unmarshalls the response and uses the inside results to build a response message for the mobile service requester and forwards it through the wireless network using communication handler interface. The same process is performed for failed resource discovery.

4.1.4 (f) Synchronizer:

The synchronizer component deployed in the system arbitrator communicates with its counterpart in the mobile service requester and service providers to ensure integrity of the results during the outsourcing process. In the absence of such component, the results of remote computation inside the service provider machine are not properly reflected into the native application in the mobile service requester machine.

4.1.4 (g) QoS Management:

Using QoS management component, we measure and collect QoS metrics for each successful or failed service execution in our framework. The system arbitrator needs to keep track of functional and QoS attributes of all resource providers including giant clouds, MNOs, and MNO authorized dealers. These QoS information are beneficial at the time of resource allocation by the resource scheduling. The resource scheduler utilizes QoS values of all available service providers to choose the near-optimal service provider for each call. Major QoS metrics utilized in this proposal are availability (success/failed execution), network latency to reach the service provider, and computing latency (time to execute the service by service provider).

4.1.4 (h) Outsourcing Log:

In order to complete the tasks designed for QoS management component, the outsourcing log is required to log the QoS information for each outsourcing execution. The information from this logger and execution logger in each remote service provider are used by the QoS management component for ranking service providers.

Figure 4.5 illustrates the flow of runtime computation outsourcing from local execution initiation till the execution completion in our framework. The Figure consists of three main building blocks, namely mobile service requester, system arbitrator, and horizontally heterogeneous service providers. The flowchart starts from start state located in top leftmost and flows towards down and right side of the chart during runtime and it ends at the bottom leftmost terminate state. The blue process boxes are executing in parallel; the synchronizer executes in background and application execution in foreground.

As the service provider rectangles are depicted in the figure, there are three heterogeneous service providers which are called at runtime for executing the prototype application

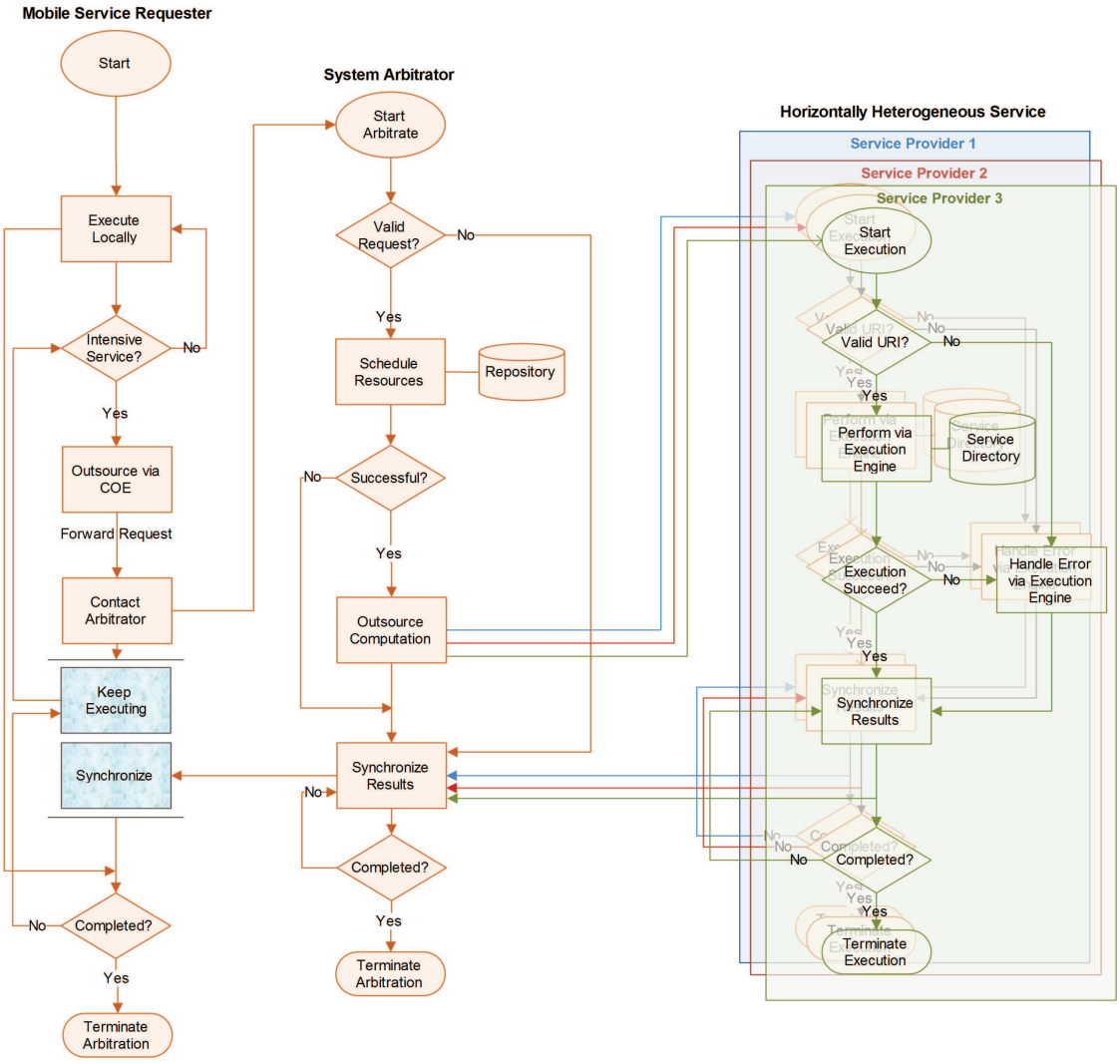


Figure 4.5: Flowchart diagram of runtime mobile computation outsourcing in the proposed framework

for our performance evaluation purpose. It is noteworthy that the flow chart contains only the states necessity for successful runtime computation outsourcing. Thus, components such as loggers and QoS management are disregarded to avoid complexity.

4.2 Significance of The Proposed Framework

The proposed framework has several significant features that are described as follows.

- **Portability:** One of the most significant features of this framework is platform-independence which is inherited from the SOA. The framework is both vertically

and horizontally platform-independent. The former ensures that applications built based on this framework can be executed on various version of certain OS, such as Android. The latter means that applications built based on this framework can be executed on and ported to varied mobile operating systems without considerable reconfiguration and modification. Employing SOA in this framework not only enhances the portability but also omits the virtualization overhead. In the absence of cross-platform solutions, the service providers and mobile service requesters need to employ virtualization technology which is a costly and heavy approach.

- **Lightweight Compute-intensive Mobile Application:** In design and development of this framework, we have employed Resource Oriented Architecture (ROA) and cloud computing principles. The service-based development generates loosely coupling of resource-intensive services with the rest of the application. So, overhead of identifying, partitioning, and offloading application to remote resources are omitted and hence the application execution originates less temporal and energy costs. Therefore, the mobile applications developed based on this framework feature higher portability, scalability, and elasticity and are lightweight.

  The term 'Lightweight' in our framework refers to the low temporal and energy overheads of utilizing heterogeneous granular cloud-based resources for efficient computation outsourcing of compute-intensive mobile applications. Furthermore, the proposed hybrid framework imposes low burden for development of application, maintenance of cloud services, and porting applications to different devices for mobile users which are also considered as lightweight features.

- **Adaptiveness:** The proposed architecture is the premier multi-tier horizontally heterogeneous granular MCC architecture that employs three tiers of cloud-based re-

sources with different granularity levels of coarse, medium, and fine. Employing horizontally heterogeneous granular resources in this framework enhances the system adaptiveness and flexibility by performing latency trade-off between computing and communication when leveraging remote resources. The coarse-grained resources feature low computing latency but high communication latency whereas fine-grained resources provide high computing latency beside low communication latency. The medium-grained resources feature medium computation and communication latencies. Hence, convergence of these horizontally heterogeneous resources can efficiently and effectively realize the computing requirements of multitudes of inhomogeneous compute-intensive task using a computation-communication trade-off toward a highly adaptive solution.

- **Trustworthiness:** Exploiting MNOs and their authorized dealer in hosting arbitration operation is a novel significant feature of this framework that significantly enhances its trustworthiness. MNOs are trustworthy mediator who has been serving mobile end-users from the beginning of the telecommunication technology. Thus, MNOs could develop a reputation and historical trust over the years among end-users. Leveraging such trustful arbitrator to supervise the outsourcing operation and its trusted dealers not only improves the reliability and availability of the framework, but also enhances the trust among end-users. Moreover, direct and indirect benefits of this framework for MNOs encourage its successful adoption.

- **Seamlessness:** Asynchronous communication technology used in this framework enables unobtrusive distraction-free interaction between the mobile service requester, system arbitrator, and service provider. While the execution of compute-intensive task continues outside the device, the mobile end-user can keep interacting with the

rest of application and also with other functions in the device. Hence, user experiences a seamless interaction and outsourcing in this framework.

- **Centralized Architecture:** Considering implications of using ad-hoc and peer-to-peer architectures, utilizing centralized architecture is remarkably enhancing the complexity and management process in computation outsourcing. Utilizing central arbitrator and transferring the complexity of interacting with heterogeneous service providers to arbitrator improves management and complexity of the framework and omit the overhead from the mobile service requester. In the absence of such centralized architecture, the overhead of service provider discovery by mobile service requester could neutralize the benefits and performance gain of outsourcing.

4.3 Performance Evaluation System Design

In this part we explain the characteristics of our performance evaluation system including performance evaluation metrics and methods. We introduce and describe our performance evaluation metrics. These metrics are selected to efficiently evaluate the lightweight properties of our MCC frameworks.

Moreover, we explain the methods to evaluate and validate the performance of the proposed framework.

4.3.1 Performance Metrics

We present round-trip time and energy consumption as our performance evaluation metrics here. Application round-trip time and consumed energy are two commonly used metrics in evaluating the performance of the MCC outsourcing systems that are presented in Table 4.1 and explained as follows. We also describe tools and methods for collecting time and energy data in this experiment.

- **Round-Trip Time (RTT):** is the total time taken from initiation stage of a compute-intensive application to completion stage for one workload in millisecond (ms). The RTT for local execution mode is referred to as Local Round-Trip Time (LRTT) and for hybrid execution mode is called Hybrid Round-Trip Time (HRTT) in this study.

- **Energy Consumption (EC):** is the total energy consumed to complete entire prototype application for a workload that is presented as millijoule (mJ).

4.3.2 Data Collection Tools

Among different tools and methods of generating execution time, including manual and automatic data collection we have designed and developed automatic logging timers that start counting at the beginning of the application execution and ends when the application successfully completes. Auto logging is beneficial in avoiding man-made mistake and enhancing data integrity, reliability, and accuracy at analysis and synthesis stages.

Table 4.1: Performance Metrics Analyzed in This Experiment

Performance Metrics	Unit
Round-Trip Time (RTT)	ms
Energy Consumption (EC)	mJ

The energy data is collected using Power Tutor 1.4 which is capable of energy profiling for the computing and wireless communications. The energy consumed by the other components such as storage and LCD is disregarded in this study, because our tested prototype is a compute-intensive application. In order to avoid man-made mistakes, the energy values are extracted and acquired from the log files created by the Power Tutor.

4.3.3 Evaluation Methods

The performance of this framework is evaluated via benchmarking experiments on real android device. Using 30 synthetic workloads, the data are collected for analysis. The

results of our benchmarking are validated via statistical model. We produce the statistical model using independent replication model to train the regression model. The identified statistical model is validated using split-sample approach. The validated models are used to generate the execution times and energy consumption data. Data analysis and synthesis testify the performance of the proposed framework.

4.4 Conclusions

In this chapter, we have presented description of our proposed framework. We provide a schematic presentation of the proposed framework and its major components. The functional and non-functional characteristics of each component are described as an individual entity. Overall operation of the proposed model also is described using sequential diagrams and flowchart. Several significant aspects of the proposed framework are highlighted. Utilizing MNOs as service provider and arbitrator in this framework has enhanced its trustworthiness, adoption, and utilizing ROA architecture helped us to effectively achieve our lightweight feature of the proposed framework. Performance evaluation system design that can be used to evaluate and validate performance of the proposed work is described. We have determined the application round-trip time and energy consumption as our performance evaluation metrics in this study. Benchmarking and statistical modelling are determined for performance evaluation purpose. Benchmarking method is used to evaluate performance of the framework and its findings are validated via statistical modelling. The apparatus for collecting round-trip time and energy consumption are explained.

CHAPTER 5

PERFORMANCE EVALUATION

In this chapter, we describe the approaches to evaluate the performance of the proposed framework. For this purpose, we analyse execution of a compute-intensive mobile application considering two performance metrics, namely Round-Trip Time (RTT) and Energy Consumption (EC) in two execution models of local and hybrid. The hybrid execution mode represents our proposed framework. Using series of benchmarking experiments in real environment, we evaluate the performance of our framework. The performance evaluation results of our framework are validated using statistical modelling. To devise our statistical model, we leverage regression analysis and employ independent replication model to produce a dataset to train the regression model. In order to validate our statistical model of our framework, we use split-sample approach. The validated statistical model is used to validate the findings of our performance evaluation results.

Section 5.1 presents our benchmarking experiment. In Section 5.2, we describe how the statistical model is created and validated. The chapter is concluded in Section 5.3.

5.1 Benchmarking

In this section, we describe the hardware and software used in the experiments, and explain the methodology used to benchmark the local and hybrid execution modes when collecting the round-trip time and profiling energy consumption.

The mobile device used in this experiment is a HTC Nexus One smartphone featuring Qualcomm QSD8250 Snapdragon with 1 GHz Scorpion processor, 512 MB RAM, 512 MB ROM and standard Li-ion 1400 mAh battery that is running Android Operating

System (OS) 2.3.4.

The wireless access point used in this study is Cisco Linksys WRT54G running firmware Ver. 4.21.5 that compiles with 802.11g at 2.4 GHz frequency. Three heterogeneous servers are selected in this experiment in different granularity levels. The coarse-grained server is a compute optimized *c1.xlarge* virtual machine instance of the Amazon EC2 located in Singapore. The server runs 64-bit Microsoft Windows Server 2008 OS with 8 vCPU featuring 20 elastic computing unit (equal to about 24 GHz CPU speed), 7 GB RAM, 4 $\times$ 420 hard disk with high performance Input/Output (I/O).

Table 5.1: Technical specifications of the client and servers used in benchmarking analysis

Device	Machine Type	CPU Type	CPU Speed	CPU Cores	RAM	Storage	OS
Smartphone	Smartphone	Qualcomm QSD8250 Snapdragon	1 GHz Scorpion	1	512 MB	512 MB	Android 2.3.4
Coarse-grained Server	Cloud VM	c1.xlarge Amazon EC2 VM	20 ECU equals to 24 GHz	8 vCPU	7 GB	4*420	Win 2008 server 64-bit
Medium-grained Server	Desktop Dell	Intel i5-2500	3.3 GHz	4	8GB	1 TB	Win 7 32-bit
Fine-grained server	Laptop Dell XPS14z	Intel 2450M Processor	2.5 GHz	4	4 GB	1 TB	Win 7 64-bit

The medium-grained server is a DELL desktop computer featuring Intel i5-2500 quad core processor working at 3.3 GHz speed, having 8 GB of RAM, and 1 TB storage, running 32-bit Microsoft Windows 7.

The fine-grained server, is a DELL Laptop XPS14z featuring Intel 2450M Processor 2.5GHz clock speed, 4GB RAM, 1 TB storage, and 64-bit Win 7 home. Table 5.1 presents summary of the systems specifications.

The prototype mobile application developed for evaluation in local and hybrid mode is a service-based client-server application consists of three utility services of factorial (calculates the factorial of the given workload), prime (verifies if the given prime number if prime), and power (it raises 999 as base into the given exponents).

In local execution mode, both client and server components are executed solely on

the mobile device, whereas in hybrid execution mode, the client components are executed locally and server components hosted on remote servers. To build the client side components we used jQuery 1.8.0 which is low footprint language for developing cross-platform systems. jQuery language is beneficial in resource-constraint mobile devices because it consumes less processing time and energy to complete the task. The PHP 5.4.24 was used to build the server side components which is predominant general-propose server side language that suits HTTP-based applications. In the server side, Structured Query Language (SQL) server is installed as a database management system. The webserver to host the utility services is Apache Tomcat web server for the all remote servers.

5.1.1 Local Execution

This section describes the process of generating evaluation data when the prototype application is running in the mobile device. In coming two sections, we discuss how the application round-trip time and consumed energy data are collected. During data collection, all the secondary applications in the mobile device are uninstalled to avoid unpredicted interruption.

5.1.1 (a) Round-Trip Time (ms)

In order to collect accurate data for running the application inside the mobile device, we deploy internal timers that are working with mobile internal clock. The timers start right before the under investigation code starts and ends when the codes finish. The timer values are stored without user interpretation to avoid man-made mistakes.

5.1.1 (b) Energy Consumption

PowerTutor 1.4 is used to monitor and profile the energy values from the mobile device. For accurate energy measurement, the CPU EC is only measured considering the fact that in local execution mode, wireless communication transmitter is off. The LCD

energy consumption is disregarded to more effectively analyze the energy consumption of computing the task. In this study, we observe that battery level of the mobile device is important in energy consumption of the device. If the battery is too low, the energy data are unreliable. Hence, we ensure that throughout the data collection in this mode, mobile device's battery does not drop below 50 % according to literature and our tests. The USB cable is disconnected and mobile device is working in room temperature between 20-22 Celsius. Also, the mobile device is fixed on the table to ensure no movement during energy profiling. Data collection is paused every few minutes, to ensure that excess CPU heat does not impact on the reliability of collected data.

5.1.2 Hybrid Execution

We explain the process of generating evaluation data when the prototype application is running at hybrid execution mode. The application execution initiates from the mobile device and ends in the mobile device by presenting the results on the screen. In coming two sections, we discuss how the data of application round-trip time and consumed energy are gathered. During data collection, all the secondary applications in the mobile device are uninstalled to avoid unpredicted interruption.

5.1.2 (a) Round-Trip Time

Similar to the local execution mode, we collect data of running the application in hybrid execution mode using internal timers inside the mobile that are working with mobile internal clock. The timers start inside the mobile right before the under investigation code starts and ends when the codes finish. The timer values are stored without user interpretation to avoid man-made mistakes.

In hybrid mode, because the communication delay is entertained, we have used timers in server systems as well. The timers start when the server services are called for execution

and end when finish computation. The times are automatically logged. To log the arbitration time, there is a counter inside the arbitrator that starts counting when the arbitrator calls the scheduler and stops when the schedulers identifies the servers and return their IP addresses to the arbitrator.

5.1.2 (b) Energy Consumption

For collecting energy data, the PowerTutor is used that monitors and profiles the energy values from the mobile device. In hybrid execution mode, apart from energy consumption of CPU, we collect energy consumed by WiFi transmitter. The LCD energy consumption is disregarded to more effectively analyze the energy consumption of computing the task. During the data collection process, we ensure that mobile device's battery does not drop below 50 % -as explained in local execution mode. The USB cable is disconnected and mobile device is fixed on the table without movement and working in the room temperature of between 20-22 Celsius. Data collection is paused every few minutes, to ensure that excess CPU heat does not impact on the reliability of collected data. Since execution of intensive components is performed outside the device, the processor work in less temperature and hence, frequency of pause during data collection is less.

5.2 Statistical Modelling

In this section, techniques that were used to provide statistical modelling of analysing round-trip time and execution time of application are discussed in two modes of local and hybrid. The prototype application used in this effort composed of three compute-intensive tasks, namely factorial, power (x,y), and prime services. In local mode, the complete code of application and web server is running inside of the smartphone, whereas in hybrid mode, the compute-intensive part of application (factorial, power(x,y), and prime services), along with arbitrator, scheduling algorithms, and database are running outside of the mobile

device, in the heterogeneous hybrid cloud resources.

The statistical model is validated using split-sample model and use part of the dataset for building the model and use the rest to validate the model. Using SPSS 21, we divide our dataset into two random subsets whose data are randomly selected using uniform function of the SPSS 21. The statistical model is applied to the subsets of main dataset and the results are used to validate the model. The supporting results are validation proof of model.

5.2.1 Local Mode

5.2.1 (a) Local Round-Trip Time (LRTT)

The application's round-trip time (RTT) in local mode called $LRTT$ is the total time consumed to execute all compute-intensive services in mobile devices. Therefore, for each workload i, the $LRTT$ is,

$$LRTT_i = RTT_{fc_i} + RTT_{pw_i} + RTT_{pr_i} \tag{5.1}$$

where RTT_{fc}, RTT_{pw}, and RTT_{pr} are execution time of factorial, power, and prime services, respectively. Therefore, first we formulate expected time for each of them. These three algorithms have different time complexity. So, to characterize growth rate of different functions according to their workloads, the Big O notation is used. However, Big O notation can only provide upper bound estimation to the growth rate of the function which is not accurate enough to perform evaluation. Therefore, specific equation for each function identify the both upper and lower bounds, along with constant and coefficient expectation values related to specific device, running the algorithm.

In order to evaluate the execution time and energy consumption of the mobile application, we leverage observation-based prediction method using supervised regression as the most common approach. In this approach, we produce a workload-time dataset by

measuring the execution time of workloads in the specific execution environment. The dataset is used to train the regression function to understand the correlation between time and workload intensity. The result of regression training ensures us about type of correlation between time and related workloads, and energy and corresponding workloads. This is useful for determining an accurate round-trip time equation. In the following we present the result of regression analysis and round-trip time equation for each services.

Factorial's Algorithm: Analyzing the algorithm results the Big O notation as $O(f^2)$ because, the highest degree of loops in the algorithm is two. Therefore, the factorial algorithm is classified as quadratic algorithm and its round-trip time Equation (RTT_{fc}) is a quadratic equation as below:

$$RTT_{fc_i} = Af_i^2 + Bf_i + C \tag{5.2}$$

where, f is the workload and RTT_{fc} is the total time in (*ms*) to execute factorial service on mobile devices. Herein, first we show the correlation between RTT and corresponding workloads which is depicted in Figure 5.1.

The results in this figure shows that the existing correlation is quadratic. Therefore, it is feasible and appropriate to perform quadratic regression analysis to determine the equation.

To identify the coefficients and constant values, we analyze the growth rate of execution time and its correlation with workloads using curve estimation regression in SPSS. The summary results of regression and Anova test presented in Table 5.2 advocate that the quadratic regression model is as accurate as %99 and is fitted into the expected equation and time complexity of $O(f^2)$.

The *R Square*, F and *Sig.* values in the Table 5.2 show significant direct correlations

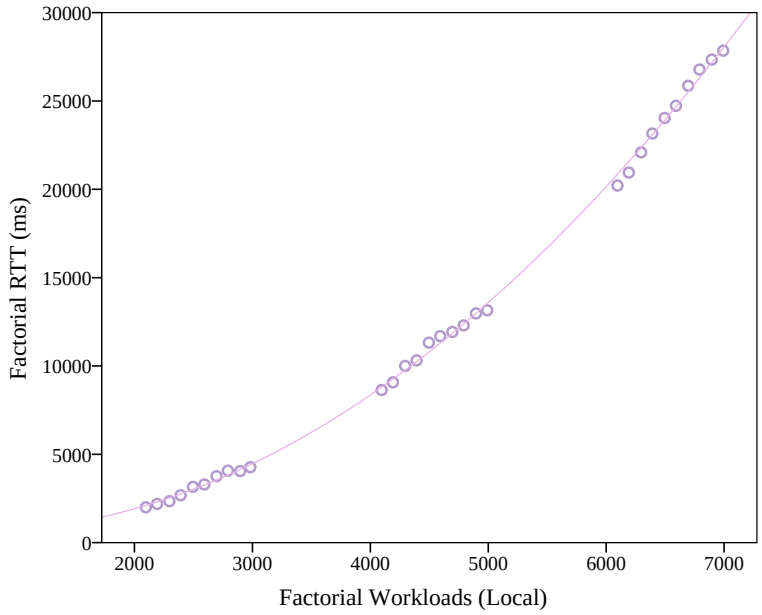


Figure 5.1: Quadratic correlation between RTT and corresponding workloads of factorial algorithm.

between the workloads and their corresponding execution times. So, by replacing the coefficient *A*, *B* with b2, b1, respectively and the constant value of *C* in Equation (5.2) with Constant value from Table 5.2, we have,

$$RTT_{fc_i} = 7.09E-4\,f_i^2 - 0.978\,f_i + 1013.97 \tag{5.3}$$

Here, is noticeable that the coefficient of f^2 as $(7.09E-4)$ is a scientific notation. Later, the results from Equation (5.3) for each workload are replaced with RTT_{fc_i} in Equation (5.1).

Table 5.2: The summary results of quadratic regression model for factorial application in local mode.

Model Summary			Parameter Estimates		
R Square	F	Sig.	Constant	b1	b2
0 .997	1776.96	0.000	1013.97	-0.978	7.09E-4

In order to validate the devised model for factorial, we perform split-sample validation model that successfully demonstrates validity of the devised model based on the

results of the analysis reported in Table 5.3.

Table 5.3: Results of split-sample validation approach of factorial service in local mode

Metrics	split = 1.00 (Selected)	split = 0.00 (Selected)	Non-split Sample
R	1	1	1
R^2	1	1	1
Adjusted R^2	1	1	1
df	15	13	29

The results show strong correlations between the factorial workloads and the execution time in all the three cases.

In the Table, the value of R, R^2, and adjusted R^2 are shown beside the df. Df row shows the number of cases used in each test. In the first case where split is 1, the df=15 shows that there are 16 cases in this sample. The correlation between the workloads and RTT in this sample is full.

The second case with df=13 where the split value equals to 0 contains 14 samples with fully supporting adjusted $R^2 = 1$ value, and in last case with full sample size of 30 workloads (df=29), the adjusted is $R^2 = 1$. In split-sample validation, there should be at most 5% difference in adjusted R^2 of splits and the full sample. Since the difference between the R^2 values of split samples and full sample are less than 5%, it can be concluded that the proposed model remains valid.

Power's Algorithm: Analyzing the power algorithm results the time complexity of $O(p^3)$ to run the big workloads specially with large exponents for power function in resource-constrained mobile device environment. Based on this result, we expected that the execution round-trip time of power function (RTT_{pw}) is calculated by a cubic equation as follows:

$$RTT_{pw_i} = Ap_i^3 + Bp_i^2 + Cp_i + D \tag{5.4}$$

where, p is the workload's exponent and RTT_{pw} is the total time in (ms) to execute

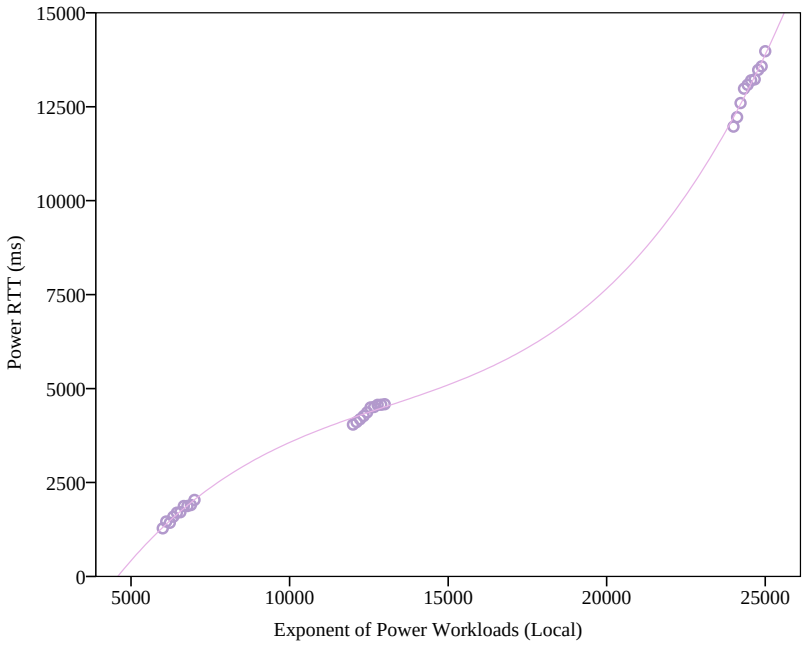


Figure 5.2: Cubic correlation between RTT and workloads of power algorithm

power service on local mode. Before performing regression, we show the correlation between RTT and corresponding workloads for the power algorithm via scatter diagram to ensure the type of correlation between RTT and workloads. Figure 5.2 shows this correlation. As it was expected, the results in this figure show that the existing relationship between total execution time and workloads is cubic. Therefore, performing cubic regression model is feasible for this correlation.

Also, to identify the coefficients and constant values, we use curve estimation regression in cubical mode. The summary results of regression and Anova test presented in Table 5.4 advocate that the cubic regression model is as accurate as %99 and is fitted into the expected equation and time complexity of $O(p^3)$. The *R Square*, F and *Sig*. values in the Table 5.4 show significant direct correlations between the workloads and their corresponding execution times. Therefore, by replacing the coefficient A, B, C, and constant value of D in Equation (5.4) with b3, b2, b1, and Constant value from Table 5.4, respectively, we have,

$$RTT_{pw_i} = 3.52E-9p_i^3 - 1.38E-4p_i^2 + 2.080p_i - 6.98E3 \tag{5.5}$$

The round-trip time calculated for each workload by this formula is replaced with RTT_{pw_i} in Equation (5.1).

Table 5.4: The summary results of cubic regression model for power application in local mode.

Model Summary			Parameter Estimates			
R Square	F	Sig.	Constant	b1	b2	b3
0.998	8351.26	0.000	-6.98E3	2.080	-1.38E-4	3.52E-9

Similar to the factorial mode, the devised statistical model of power is also validated with the same approach. The results of the validation are reported in Table 5.5.

Table 5.5: Results of split-sample validation approach of power local RTT

Metrics	split = 1.00 (Selected)	split = 0.00 (Selected)	Non-split Sample
R	1	1	0.999
R^2	0.999	0.999	0.999
Adjusted R^2	0.999	0.999	0.999
df	15	13	29

As the results of our study in the table indicate, there is a full support in correlations between the power workload values and the RTT time in all the three cases. In the first case where split is 1, the df=15 shows that there are 16 cases in the sample. The reported correlation between the workload and RTT in this sample is as significant as adjusted $R^2 = 0.999$. The second case where the split value equals to 0, the adjusted $R^2 = 0.999$ for 14 samples, and in full sample size of 30 workloads (df=29), the adjusted is $R^2 = 0.999$. Since the difference between the R^2 values of split samples and full sample are less than 5%, the proposed model is valid.

Prime's Algorithm: For this service we select the prime workloads to have a compute-intensive application with longer response time compare with workloads that are not prime. Analyzing the algorithm results the time complexity of $O(r)$ for running prime workloads. Based on this result, the linear equation is expected for round-trip time of

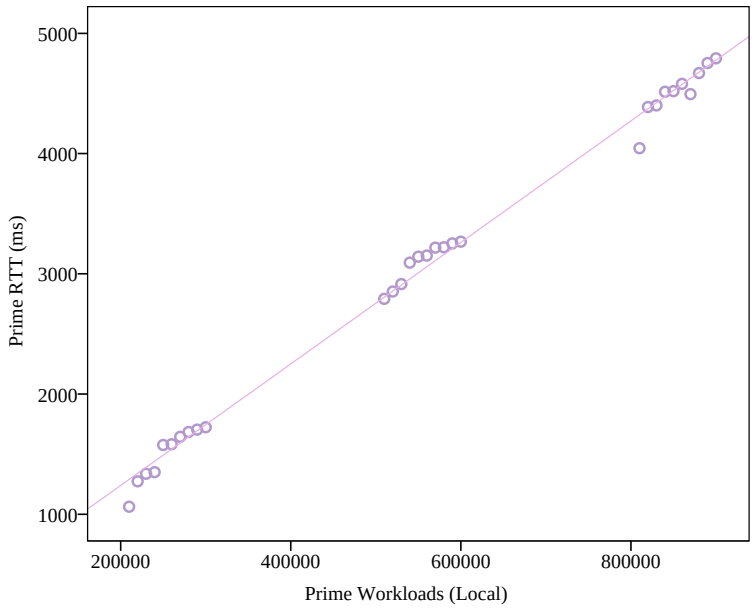


Figure 5.3: Linear correlation between RTT and corresponding workloads of prime algorithm.

application as RTT_{pr} as follows:

$$RTT_{pr_i} = Ar_i + B \tag{5.6}$$

where, r is the prime workload and RTT_{pr} is the total time in (ms) to execute prime service. To ensure the linearity correlation between RTT and corresponding workloads for the prime algorithm we show the scatter diagram plotted in Figure 5.3. As it was expected, the results in this figure shows that the the linear relationship between RTT and related workloads. Therefore, the linear regression model is feasible for this correlation.

We use linear estimation regression to identify the correlation and constant values. The summary results of regression and Anova test presented in Table 5.6 validate that the linear regression model is as accurate as %98 and is fitted into the expected Equation 5.6. Also, the *R Square*, F and *Sig.* values in the Table 5.6 show significant direct correlations between workloads and their corresponding execution time. Therefore, by replacing the coefficient A and constant value of B in Equation (5.6) with b1 and Constant value from

Table 5.6, we have

$$RTT_{pr_i} = (5.39E-3)r_i + 111.25 \tag{5.7}$$

The round-trip time calculated for each workload by this formula is replaced with RTT_{pr} in Equation (5.1).

Table 5.6: The summary results of linear regression model for prime application in local mode.

Model Summary			Parameter Estimates	
R Square	F	Sig.	Constant	b
0.989	1141.71	0.000	111.25	5.39E-3

The devised prime model is validated using split-sample approach. The results of the analysis are reported in Table 5.7.

Table 5.7: Results of split-sample validation approach of local prime RTT

Metrics	split = 1.00 (Selected)	split = 0.00 (Selected)	Non-split Sample
R	1	1	1
R^2	1	1	1
Adjusted R^2	1	1	1
df	15	13	29

As the results in the Table show, there is a strong correlations between the independent and dependent variables in all three sample sizes. In the first case where split is 1, the df=15 shows that there are 16 cases in the sample. The adjusted R^2 value in table shows perfect correlation between the RTT and prime workload value in this sample. The second case with df=13 with split value of 0, the adjusted $R^2 = 1$ which is an evidence on full collaboration and support. In last group of full sample size (df=29), the adjusted is $R^2 = 1$. Thus, because there is no difference between the adjusted R^2 values of split samples and full sample, the criteria for split-sample validation (difference less than 5%) is met and hence we can conclude that the proposed model is valid.

We demonstrate that our devised models are valid and hence we can leverage these models to generate the total local RTT by substituting the right side of Equations (5.3), (5.5), and (5.7) in Equation (5.1. Thus, the $LRTT$ is:

$$\begin{aligned} LRTT_i &= RTT_{fc_i} + RTT_{pw_i} + RTT_{pr_i} \\ &= (7.09E-4f_i^2 - 0.978f_i + 1013.97) \\ &\quad + (3.52E-9p_i^3 - 1.38E-4p_i^2 + 2.080p_i - 6.98E3) \\ &\quad + (5.39E-3r_i + 111.25) \end{aligned} \tag{5.8}$$

By calculating the sum of constant values, we simplify the Equation (5.8) as below:

$$\begin{aligned} LRTT_i &= (7.09E-4f_i^2 - 0.978f_i) \\ &\quad + (3.52E-9p_i^3 - 1.38E-4p_i^2 + 2.080p_i) \\ &\quad + (5.39E-3r_i) + 5.96E3 \end{aligned} \tag{5.9}$$

The above statistical model is used to generate round-trip times of 30 different workloads that will be presented in the next chapter.

5.2.1 (b) Local Energy Consumption (LEC)

The most part of local application energy consumption on mobile devices is comprised of the energy consumed by CPU and LCD at application run-time. In this study we consider only CPU usage which is completely dependent to processing time of services running on mobile devices. Therefore, to calculate the total energy consumption LEC (mJ) consumed for the local execution, we expect to have an equation as follows,

$$LEC_i = LRTT_i \times (TEC_{cpu_i}) \tag{5.10}$$

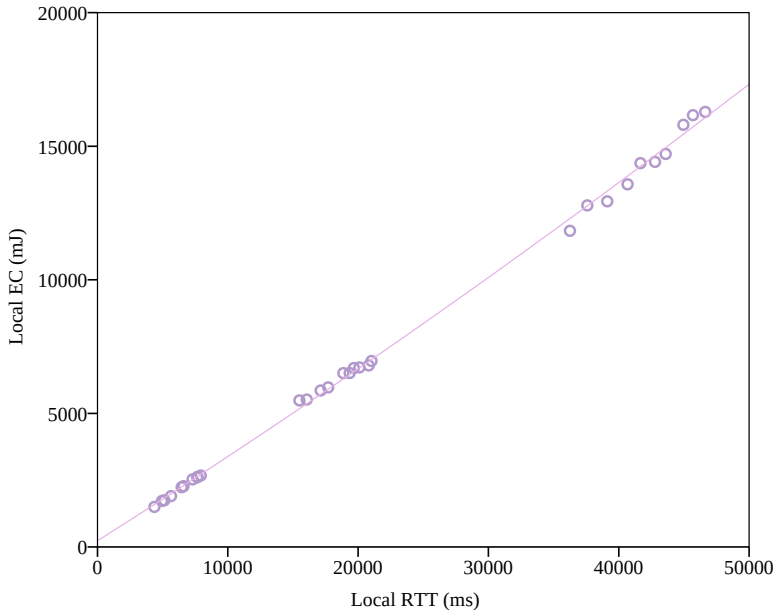


Figure 5.4: Linear correlation between round-trip time and energy consumption of application in local mode.

Herein, we put the identified value of $LRTT_i$ observed from Equation (5.9) in the Equation 5.10. Herein, to ensure the type of correlation between LRTT and LEC, we use scatter diagram presented in Figure 5.4. The fit line of diagram ensures the linearity of this correlation. Therefore, the linear regression model is feasible for this correlation and expected energy consumption equation.

Therefore, to have the mean value of energy consumed by CPU per millisecond TEC_{cpu} to process tasks on mobile device, we use the linear regression model for a dataset of workloads. The summary results of regression and Anova test presented in Table 5.8 shows that the linear regression model is as accurate as %99 and is fitted into the expected Equation (5.10) with considering R^2 equal with 0.999 form model summery. Also, the F and Sig. values in the Table 5.8 show significant direct correlations between workloads and their corresponding execution time. The 0.344 (mW) shows estimated power consumed by CPU per millisecond to tally with the execution time unit. Replacing the b value from

this table with TEC_{cpu} in Equation (5.10) is resulted as follows:

$$LEC_i = LRTT_i \times 0.344 \tag{5.11}$$

where the unit of LEC is (mJ) and shows the total energy need to execute the mobile application at total time of $LRTT$ in (ms) for the i^{th} workload.

Table 5.8: The summary results of linear regression model for local energy consumption.

Model Summary			Parameter Estimates
R Square	F	Sig.	b
0.999	13489.84	0.000	0.344

To validate the model the split-sample approach is performed on the dependent and independent variables and the existing correlations in the split samples and full sample are studied. The results of the analysis are reported in Table 5.9.

Table 5.9: Results of split-sample validation approach of local CE model

Metrics	split = 1.00 (Selected)	split = 0.00 (Selected)	Non-split Sample
R	1	1	1
R^2	1	1	1
Adjusted R^2	1	1	1
df	15	13	29

As the results of our study in the table indicate, there is a strong support in correlations between the independent and dependent variables for all the three cases. The df row shows the number of cases in each test. In the first case where split is 1, the df=15 shows that there are 16 cases in the sample. The reported correlation between the RTT and energy consumption in this sample is as significant as Adjusted $R^2 = 1$. The second case with df=13 where the split value equals to 0, the Adjusted $R^2 = 1$, and in full sample size of 30 workloads (df=29), the Adjusted is $R^2 = 1$. Since the difference between the R^2 values

of split samples and full sample are less than 5%, we can conclude that the proposed model is valid. The trained values of energy consumption of 30 different workloads by this statistical model will be presented in the next chapter to compare with measured value of energy consumption in real experiment.

5.2.2 Hybrid Mode

5.2.2 (a) Hybrid Round-Trip Time (HRTT)

In hybrid mode the $HRTT$ for executing the cloud-based mobile application (CMA) is calculated via considering the arbitration, communication overhead, and maximum time needs to process three services of factorial, power, and prime in remote resources. In hybrid mode, mobile device calls arbitrator and sends along the name of desired services asking to find appropriate resources for remote execution; arbitrator use its scheduler component to find the IP address(es) of the appropriate service providers able to perform mobile device task; one found, the scheduler return results to the arbitrator. Then, arbitrator makes asynchronous calls to the identified remote resources along with data asking for execution. Once the results are completed and received to the arbitrator, it forwards them to the mobile device. Therefore, for calculation of HRTT we need to calculate the Arbitrator Time (AT), Remote Execution Time (RET), and the total communication time to complete CMA.

So, the $HRTT$ equation for i^{th}workload is as follows:

$$HRTT_i = AT_i + RET_i + CT_i \tag{5.12}$$

where $HRTT_i$ is the total time to execute each workload in hybrid mode, AT_i is the arbitrator time, RET_i is the remote execution time, and CT_i is the communication time for i^{th} workload.

The AT_i is the total time taken for i^{th} workload by the mobile device to send the request to the arbitrator for IP addresses of the remote resources, along with delay of scheduling task and fetching data from the database. Whereas, the scheduling time is independent of workloads, then we consider only one mean value for all workloads. Also, its value highly depends on the wireless network and performance of computing device that hosted the system arbitrator. Therefore, we replace AT_i to AT. In order to identify the mean value of system arbitrator, we measured the arbitrator delays. The calculated mean value is 1041.39 (ms). For the sake of simplicity, we round the delay to 1041 (ms) per call. Therefore, we have,

$$AT = 1041(ms) \tag{5.13}$$

RET_i in the Equation (5.12) is the total round-trip time of executing all three services for i^{th} in multi-layered remote servers including the total time taken from the moment that arbitrator performs remote calls until it receives the results. It includes communication delay of transmitting workload values to the remote servers and receive their results plus computing time in the servers. Hence, we can write the RET_i equation as below:

$$RET_i = RET_{fc_i} + RET_{pw_i} + RET_{pr_i} \tag{5.14}$$

where RET_{fc}, RET_{pw}, and RET_{pr} are remote execution time of factorial, power, and prime for i^{th} workload, respectively. Because, there are more than one remote server to perform remote computation, it is beneficial to perform asynchronous calls from the system arbitrator, so that, remote executions perform in parallel. Therefore, RET_i equals to the execution time of the longest service. Hence, we have

$$RET_i = Max(RET_{fc_i}, RET_{pw_i}, RET_{pr_i}) \tag{5.15}$$

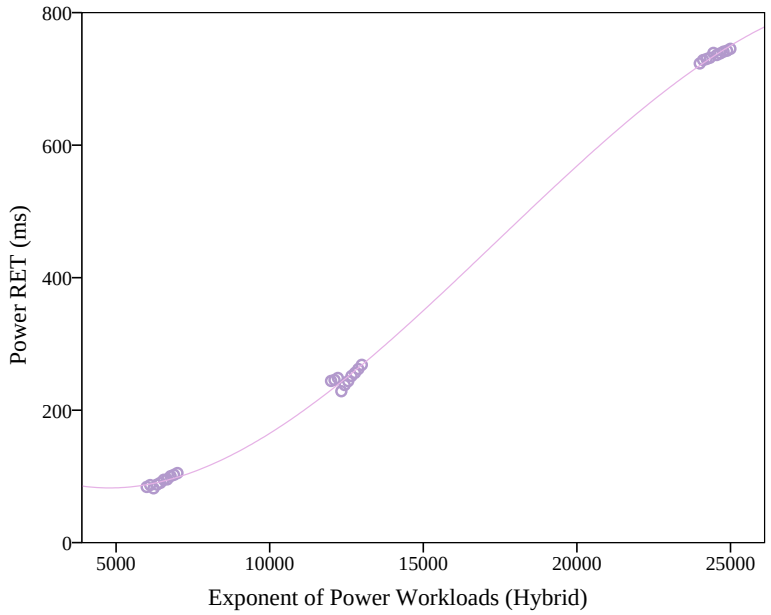


Figure 5.5: Cubic correlation between RTT and workloads of power algorithm.

According to the results of our independent replication, the power function is the heaviest service that takes the longest time. So, the maximum execution time between all services is belong to power function. Then, the equation can be rewritten as:

$$RET_i = RET_{pw_i} \tag{5.16}$$

Here, also we follow the same approach like local mode and leverage observation-based prediction method using supervised regression model. As explained in local execution mode, the complexity of power algorithm is cubic and the equation will be cubic. Therefore, we will have

$$RET_{pw_i} = Ap_i^3 + Bp_i^2 + Cp_i + D \tag{5.17}$$

Figure 5.5 depicts the correlation between remote execution time of power function and related workloads to ensure the type of correlation. As it was expected, the results show a cubic relationship between RET_{pw} and its related workloads. Therefore, the curve regression model of cubic is feasible for this correlation.

In order to identity the coefficients, we perform the curve estimation regression. Table 5.10 shows the summary of regression analysis results.

Table 5.10: The summary results of cubic regression model for remote execution time.

Model Summary			Parameter Estimates			
R Square	F	Sig.	Constant	b1	b2	b3
0.996	3091.113	0.000	17.56	0	1.79E-006	-2.59E-011

The *R Square*, F and *Sig*. values in the Table 5.10 show significant direct correlations between workloads and their corresponding remote execution time. Therefore, by replacing the coefficient A and constant value of B in Equation (5.17) with b1 and Constant value from Table 5.10, respectively, we have,

$$\begin{aligned} RET_{pw_i} &= (-2.59E-011)p_i^3 + (1.79E-006)p_i^2 + (0)p_i + 17.56 \\ &= (-2.59E-011)p_i^3 + (1.79E-006)p_i^2 + 17.56 \end{aligned} \tag{5.18}$$

Therefore, based on Equation (5.16) we have,

$$\begin{aligned} RET_i &= RET_{pw_i} \\ &= (-2.59E-011)p_i^3 + (1.79E-006)p_i^2 + 17.56 \end{aligned} \tag{5.19}$$

Before we leverage the result produced using the above model, we validate the devised model using split-sample approach that is used in earlier validation efforts in this chapter. The split-sample approach is applied to identify the correlation between the RET_i and the value of P_i and compare the existing correlations in the split samples and full sample. The results of our analysis are reported in Table 5.11 and interpreted as follows.

As the results in Table 5.11 show, the existing correlations between the p_i and RET_i are significant for all the three cases. In the first case where split is 1 and df=15, the reported correlation in this sample is full. The second case with df=13 where the split

Table 5.11: Results of split-sample validation approach for RET_i in hybrid mode

Metrics	split = 1.00 (Selected)	split = 0.00 (Selected)	Non-split Sample
R	1	1	1
R^2	1	1	1
Adjusted R^2	1	1	1
df	15	13	29

value equals to 0, the adjusted $R^2 = 1$, and in full sample size of 30 workloads (df=29), the adjusted is $R^2 = 1$ too. Thus, due to full similarity of the adjusted R^2 value for both split samples and the full sample and different less than 5%, the proposed model is valid.

The total communication delay CT_i is the existing delay caused through wireless communication during the entire hybrid mode cycle. This delay includes the communication time of sending mobile device request to the arbitrator, the delay of arbitrator to perform the remote calls to the remote servers, delay of taking all the results and send them back to the mobile device.

Whereas, our prototype is compute-intensive so the communication volume in our experiment is fixed and does not grow by workload increase. Hence, there is no correlation between the CT_i and workloads. This delay depends only on the quality of communication medium; the medium is fixed for all workloads and executions. In order to estimate its value, we observed communication delay of our dataset and calculated its mean value as $134.41(ms)$ that we round it to 134. Since our prototype application is not data-intensive, the potential wireless fluctuations do not originate substantial impact on communication delay. Therefore, we have,

$$CT_i = 134(ms) \tag{5.20}$$

So, by replacing the right side of Equations (5.13), (5.19), and (5.20) in Equation

(5.12), we have,

$$\begin{aligned} HRTT_i &= 1041 + (-2.59E-011)p_i^3 + (1.79E-006)p_i^2 + 17.56 + 134 \\ &= (-2.59E-011)p_i^3 + (1.79E-006)p_i^2 + 1192.56 \end{aligned} \tag{5.21}$$

The above validated statistical model is used to generate hybrid round-trip times of 30 different workloads that will be presented in the next chapter.

5.2.2 (b) Hybrid Energy Consumption (HEC)

To evaluate the energy consumption during CMA execution in hybrid mode we only consider energy usage by CPU and WiFi data transmission without LCD consideration like the local mode evaluation. Therefore, to calculate the total energy consumption HEC (mJ) for the hybrid execution, we have,

$$HEC_i = TEC_{cpu_i} + TEC_{w_i} \tag{5.22}$$

where HEC_i is the amount of energy consumed by mobile devices to execute CMA for i^{th} workload through heterogeneous hybrid remote resources. The TEC_{cpu} and TEC_w are the Total Energy Consumed (TEC) by CPU and WiFi , respectively. The TEC_{cpu_i} is resulted from multiplication of CMA's round-trip time into the mean value of energy consumed by CPU per millisecond as EC_{cpu} for the i^{th} workload. The TEC_w is also calculated by multiplication of CMA's round-trip time into the mean value of energy consumed by WiFi per millisecond called EC_w for each workload. Therefore,

$$TEC_{cpu_i} = HRTT_i \times EC_{cpu_i} \tag{5.23}$$

$$TEC_{w_i} = HRTT_i \times EC_{w_i} \tag{5.24}$$

Whereas, the tested application was focused on intensive computation so the uplink-bytes and downlink-bytes are always low (at most 1163 Bytes) in this study. Thus, the WiFi state is low and based on the literature and our investigation, the under test mobile device consumes 34 (mW/s) power as long as is connected to wireless network by WiFi. Therefore, we put the value of 0.034 (mW/ms) for EC_{w_i} in the Equation (5.24). Then,

$$TEC_{w_i} = HRTT_i \times 0.034 \tag{5.25}$$

Herein, to ensure the type of correlation between round-trip time and CPU energy consumption in hybrid mode, we use scatter diagram presented in Figure 5.6. The diagram shows linearity correlation between total time taken for remote execution and energy consumed by mobile device's CPU for each of 30 workloads. So, the linear regression model is feasible for this correlation and expected equation for calculating total energy consumption by CPU .

Therefore, considering Equation (5.23) we are expected to have

$$TEC_{cpu_i} = HRTT_i \times b + \alpha \tag{5.26}$$

Here, to find the coefficient b as the mean value of energy consumed by CPU per millisecond as EC_{cpu} for executing CMA, we use the linear regression model for a dataset of workloads. The summary results of regression and Anova test presented in Table 5.12 shows that the linear regression model is as accurate as %93. Also, the R^2, F, and $Sig.$ values in the Table 5.12 show significant direct correlations between workloads' corre-

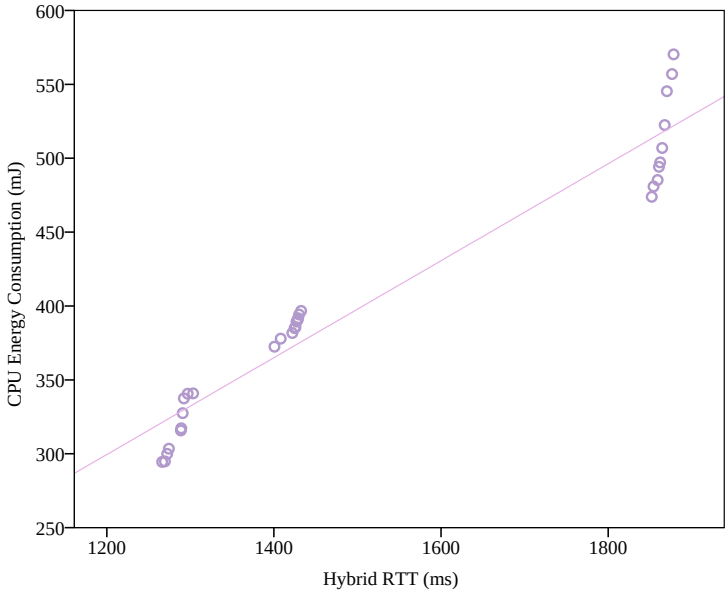


Figure 5.6: Linear correlation between round-trip time and energy consumption of CPU in hybrid mode.

sponding execution time and energy usage.

Table 5.12: The summary results of linear regression model for energy consumption of CPU in hybrid mode.

Model Summary			Parameter Estimates	
R Square	F	Sig.	Constant	b
0.934	398.26	0.000	-94.10	0.328

Substituting the b value and constant from this table with Equation (5.26), we have

$$TEC_{cpu_i} = (HRTT_i \times 0.328) - 94.10 \tag{5.27}$$

From Equations (5.22), (5.25), and (5.27) we derive:

$$\begin{aligned} HEC_i &= (HRTT_i \times 0.328 - 94.10) + (HRTT_i \times 0.034) \\ &= HRTT_i \times 0.362 - 94.10 \end{aligned} \tag{5.28}$$

where the energy calculated through HEC_i (mJ) formula is the total energy needs to

execute the CMA at total time of $HRTT_i$ in (ms).

Now, we can put the $HRTT_i$ from Equation (5.21) in Equation (5.28) and calculate the HEC_i.

To validate the devised model, we deploy split-sample approach on the dependent and independent variables and investigate the existing correlations in the split samples and full sample. The results of our investigation are reported in Table 5.13 and interpreted as follows.

Table 5.13: Results of split-sample validation approach of hybrid CE model

Metrics	split = 1.00 (Selected)	split = 0.00 (Selected)	Non-split Sample
R	1	1	1
R^2	1	1	1
Adjusted R^2	1	1	1
df	15	13	29

As the results of our study in the table indicate, there is a strong support in correlations between the independent and dependent variables for all the three cases. The df row shows the number of cases in each test. In the first case where split is 1, the df=15 shows that there are 16 cases in the sample. The reported correlation between the RTT and energy consumption in this sample is as significant as Adjusted $R^2 = 1$. The second case with df=13 where the split value equals to 0, the Adjusted $R^2 = 1$, and in full sample size of 30 workloads (df=29), the Adjusted is $R^2 = 1$. Because the difference between the R^2 values of split samples and full sample are less than 5%, we can conclude that the proposed model is valid. The amount of energy consumption for 30 different workloads by this statistical model will be presented in the next chapter to compare with measured value of energy consumption in real experiment.

5.3 Conclusions

This chapter presented the methodology used for evaluation and validation of results collected from analysing the performance in two modes of local and hybrid. Benchmarking experimentation is the method used to evaluate the RTT and EC for each mode. Also, statistical modelling is performed to validate all the results achieved by benchmarking for both modes. Regression analysis as the dominant observation-based method is used to devise our statistical model. We validate the statistical model using split-sample approach and compare the correlations between the dependent and independent variables, and advocate the validity of our statistical model. In the next chapter, we present the numerical and statistical results, along with analysis of findings that achieved from two parts of benchmarking and statistical modelling for both local and hybrid modes.

CHAPTER 6

RESULTS AND DISCUSSION

Evaluation results of the proposed framework prototype through benchmarking experiment and statistical analysis are reported in this chapter. Round-trip time and energy consumption data are presented, analysed and synthesized for two modes of local and hybrid mode in three levels of workload intensity (i.e., low, medium, and high). Finally, the evaluation results are validated with statistical modelling using independent replication method.

The reminder of this chapter is as follows. The results of benchmarking experiment are presented and evaluated in section 6.1. Statistical modelling results are presented in section 6.2 and further discussions are provided in section 6.3 that compares the time and energy results between statistical model and benchmarking. Section 6.4 concludes the chapter.

6.1 Performance Evaluation Results

The round-trip time and energy consumption results that are generated using benchmarking experiment are provided in this section. The results are presented with the help of descriptive statistics, tables, and charts. These results are used to evaluate our proposed heterogeneous hybrid cloud-based resources.

6.1.1 Round-Trip Time (RTT)

The RTT results of benchmarking experiment for the local and hybrid execution modes are presented in Tables 6.1 and 6.2. To ensure reliability of our generated data, we repeat the execution of each workload for 30 times.

The columns in the table summarize the mean RTT values of 30 iterations of each workload, standard deviation (SD) in values of each workload, error estimation, and mean RTT values of 30 iterations with 99% confidence interval for each workload (total 30 workloads).

Presenting results with 99% confidence interval ensures the reliability of the results. For example, for the workload 7, the local RTT is 7300.9(+/-)133.2 which means that the RTT is between 7300.9-133.2 and 7300.9+133.2. Thus, RTT for 7^{th} workload is 7300.9-133.2 < RTT < 7300.9+133.2 or 7167.7< RTT < 7434.1. This range means that with 99% confidence, the RTT value for execution of 7^{th} workload is between 7167.7 and 7434.1 if repeated again.

For local execution of every workload in Table 6.1, there is a corresponding remote execution in Table 6.2. For instance, the RTT for the workload number 7 is 7300.9 ms and 1290.83 in local and hybrid execution modes, respectively. For each workload, the individual time saving can be calculated. For example, the RTT saving when executing the 7^{th} workload in hybrid mode is as significant as 6010.07 ms which is 82.3 %.

The Table 6.3 depicts descriptive statistics of measured RTT data (ms) for 30 workloads in both local and hybrid execution modes. The statistical modelling results are classified in three categories of intensity level: low, medium, and high, along with minimum and maximum RTT data for all three levels of intensity in the last row of the table. Each category represents the measured mean values of the minimum and maximum RTT, beside measured RTT average for each 10 workloads as column N shows the number of workloads.

Based on measured results reported in the Table 6.3, the hybrid execution mode can reduce execution time of compute-intensive mobile application about 71% to 84%, 91% to 93%, and 95% to 96%, in low, medium, and high intensity levels categories, respectively,

Table 6.1: Round-Trip Time (RTT) values with 99% confidence interval in local execution mode

Workloads	Mean RTT (ms)	SD_RTT	Error Estimate	RTT with 99% CI
1	4364.4	109.5	51.6	4364.4(+/-)51.6
2	4952.7	168.2	79.2	4952.7(+/-)79.2
3	5125.3	121.3	57.2	5125.3(+/-)57.2
4	5640.1	263.1	123.9	5640.1(+/-)123.9
5	6457.4	224.4	105.7	6457.4(+/-)105.7
6	6606.2	270.1	127.2	6606.2(+/-)127.2
7	7300.9	282.8	133.2	7300.9(+/-)133.2
8	7653.9	238.6	112.4	7653.9(+/-)112.4
9	7682.4	248	116.8	7682.4(+/-)116.8
10	7925.2	277.2	130.6	7925.2(+/-)130.6
11	15498.2	224.2	105.6	15498.2(+/-)105.6
12	16059.3	196.1	92.4	16059.3(+/-)92.4
13	17121.4	214.7	101.1	17121.4(+/-)101.1
14	17702.8	132.9	62.6	17702.8(+/-)62.6
15	18861.1	349.7	164.7	18861.1(+/-)164.7
16	19355.4	282.1	132.9	19355.4(+/-)132.9
17	19671.1	251.7	118.5	19671.1(+/-)118.5
18	20102	325.5	153.3	20102(+/-)153.3
19	20816.7	268.3	126.4	20816.7(+/-)126.4
20	21021.5	226.3	106.6	21021.5(+/-)106.6
21	36251.2	381.4	179.7	36251.2(+/-)179.7
22	37582.3	307.5	144.9	37582.3(+/-)144.9
23	39114.4	367.3	173	39114.4(+/-)173
24	40673.5	385.9	181.8	40673.5(+/-)181.8
25	41662.6	388.7	183.1	41662.6(+/-)183.1
26	42784.1	338.9	159.6	42784.1(+/-)159.6
27	43602.2	377.3	177.7	43602.2(+/-)177.7
28	44954.6	265.1	124.9	44954.6(+/-)124.9
29	45692.6	289.9	136.6	45692.6(+/-)136.6
30	46631.9	362.7	170.9	46631.9(+/-)170.9

Table 6.2: Round-Trip Time (RTT) values with 99% confidence interval in hybrid execution mode.

Workloads	Mean RTT (ms)	SD_RTT	Error Estimation	RTT with 99% CI
1	1266.2	49.4	23.3	1266.2(+/-)23.3
2	1269.63	50.9	24	1269.6(+/-)24
3	1272.33	54.7	25.8	1272.3(+/-)25.8
4	1274.3	47.4	22.3	1274.3(+/-)22.3
5	1288.67	47.7	22.5	1288.7(+/-)22.5
6	1289.1	49.1	23.1	1289.1(+/-)23.1
7	1290.83	50.4	23.8	1290.8(+/-)23.8
8	1292.27	47.3	22.3	1292.3(+/-)22.3
9	1296.83	43.8	20.6	1296.8(+/-)20.6
10	1303.3	39.2	18.5	1303.3(+/-)18.5
11	1400.7	42.8	20.2	1400.7(+/-)20.2
12	1408.13	35.4	16.7	1408.1(+/-)16.7
13	1422	11.3	5.3	1422(+/-)5.3
14	1424.77	38.8	18.3	1424.8(+/-)18.3
15	1425.87	17.9	8.4	1425.9(+/-)8.4
16	1427.07	15.8	7.4	1427.1(+/-)7.4
17	1428.3	17.9	8.4	1428.3(+/-)8.4
18	1429.03	26.7	12.6	1429(+/-)12.6
19	1430.1	16	7.5	1430.1(+/-)7.5
20	1432.53	17	8	1432.5(+/-)8
21	1852.13	41.4	19.5	1852.1(+/-)19.5
22	1854.3	41.6	19.6	1854.3(+/-)19.6
23	1859.13	42.9	20.2	1859.1(+/-)20.2
24	1860.77	48.4	22.8	1860.8(+/-)22.8
25	1862.2	46.1	21.7	1862.2(+/-)21.7
26	1864.6	45.2	21.3	1864.6(+/-)21.3
27	1867.63	48.6	22.9	1867.6(+/-)22.9
28	1870.2	52.1	24.5	1870.2(+/-)24.5
29	1876.37	50.5	23.8	1876.4(+/-)23.8
30	1878.27	50.6	23.8	1878.3(+/-)23.8

Table 6.3: Descriptive statistics of RTT for local and hybrid mode in real environment

Intensity	Mode	N	Min RTT (ms)	Max RTT (ms)	Mean RTT (ms)
Low	Local	10	4364.40	7925.20	6370.85
	Hybrid	10	1266.20	1303.30	1284.35
	Valid N (listwise)	10			
Medium	Local	10	15498.20	21021.50	18620.95
	Hybrid	10	1400.70	1432.53	1422.85
	Valid N (listwise)	10			
High	Local	10	36251.20	46631.90	41894.94
	Hybrid	10	1852.13	1878.27	1864.56
	Valid N (listwise)	10			
All	Local	30	4364.40	46631.90	22295.58
	Hybrid	30	1266.20	1878.27	1523.92
	Valid N (listwise)	30			

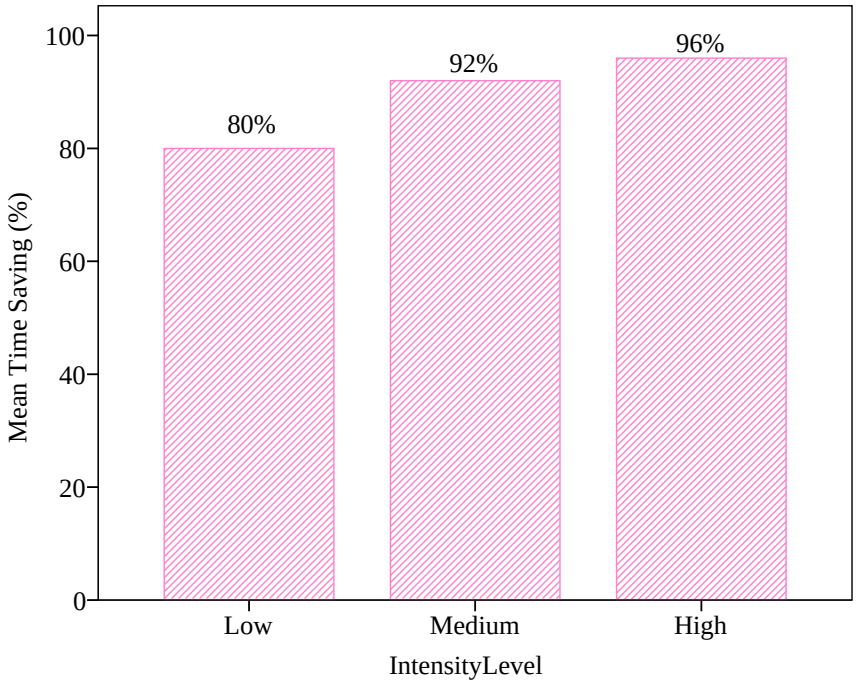


Figure 6.1: Time saving average in hybrid mode from measured data

compare with local execution mode. Therefore, these results validate that our proposed model can execute the CMA in about 80% to 96% shorter time than local mode similar to results in statistical modelling.

The Figure 6.1 that is based on mean RTT results from benchmarking that shows considerable achievements in RTT reduction to execute CMA like previous section. This

is a proof to indicate that time saving achievements are increased about 16% with increase of workloads' intensity in hybrid mode.

Therefore, it validates that the results calculated from statistical modelling are very close to benchmarking results in real environment. So, we can conclude that leveraging remote computation resources for executing higher resource-intensive tasks on mobile devices can recommended through our proposed model.

In local mode, the results in Table 6.3 show that when the workload intensity is low, the mean RTT is equal to 6370.85 (ms) ($\simeq$ 6s). With increase of workloads in medium and high intensity levels, the RTT significantly grows. The low-medium and low-high RTT differences are as high as 12250.10 (ms) ($\simeq$ 12s) and 35524.09 (ms) ($\simeq$ 36s). By contrast, in hybrid mode when the workload is low, the mean RTT is equal to 1284.35 (ms) ($\simeq$ 1s). Unlike local execution mode, in hybrid mode increase in workload intensity, causes insignificant rise in RTT values. Low-medium and low-high RTT differences are as low as 138.50 (ms) ($\simeq$ 0.1s) and 580.21 (ms) ($\simeq$ 0.6s) in second and third workloads intensity categories, receptively.

Also, Table 6.4 verifies the significant differences between total time execution of local mode and hybrid mode by extending our analysis through performing Paired Sample T-Test. This test ensures that there is 98% correlation between execution time of local and hybrid datasets as well as the test for statistical results. Also, the Sig. (2-Tailed) value equal to zero, expresses that there is a statistically significant difference between the measured values of mean RTT for application execution in local mode and hybrid mode environment.

Table 6.4: Paired Sample T-Test: Local RTT & Hybrid RTT from measured data

Paired Samples Correlations				**Paired Samples T-Test**	
Pair	N	Correlation	Sig.	t	Sig.(2-tailed)
(Local RTT & Hybrid RTT)	30	0.986	0.000	7.626	0.000

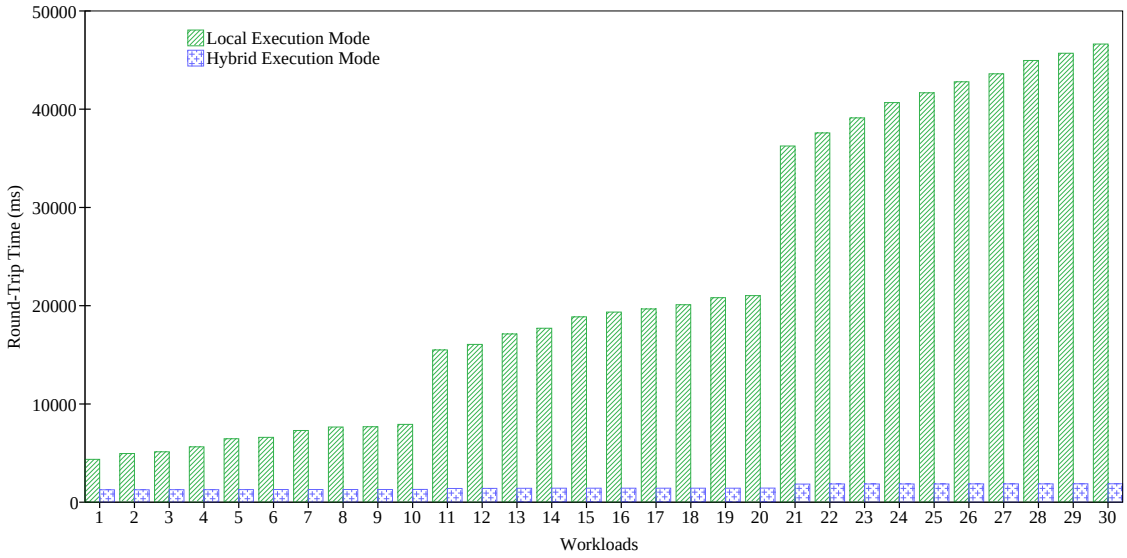


Figure 6.2: Comparison of application round-trip time for 30 workloads using measured data: Local mode vs Hybrid mode

The Figure 6.2 compares the measured round-trip time for 30 different workloads with three intensity level (i.e., low (1-10), medium (11-20), and high (21-30)) to execute compute-intensive application in two execution modes of local and hybrid. The bar lines in this figure represent the mean value of 30 execution iterations for each workload. The green stripped bars are showing the RTT in local execution mode, whereas the blue plus pattern bars are related to RTT in hybrid execution mode when the computation-intensive part(s) of the application are running outside of mobile device in remote computing servers.

As can be seen in the Figure 6.2, in local execution mode, low-intensity workloads (1-10) complete application execution in significantly shorter time compared to the medium- and high-intensity workloads that take longer to finish execution in mobile device. Because, the tested application logic is CPU- and memory-bound, increasing workload intensity leads to the execution time grows in local mode due to resource-poverty of mobile devices. However, despite of scheduling engine in arbitrator and data transmission through wireless communication, increase in workload intensity in hybrid mode has low impact(s)

on execution time growth of the CMA, using high computing processors with large storage and memory.

Also, the Figure 6.2 demonstrates, the hybrid MCC is a time-efficient platform for cloud-based mobile augmentation compared with local mode application execution. Moreover, the higher compute-intensity of workloads, the greater is the time saving in hybrid mode. Here, it is noticeable that the monetary cost of using CMA like RTT is one of the major factors for mobile users. Therefore, affordable cost provided by mobile user as a priority parameter for outsource service scheduling can indirectly impact on RTT based on computing capability of the selected remote resources.

Figure 6.3 depicts average time of computation and communication of intensive services on coarse-, medium-, and fine-grained resources for low, medium, and high intensity workloads. Bars and their segmented areas clearly demonstrate computation-communication trade-off when leveraging heterogeneous granular resources for low, medium, and high intensity workloads. The highest communication overhead belongs to distant resource stated at the coarse level, while the rest of resources have very low communication overhead (hardly seen on diagram).

This is noteworthy that factorial service is the most computing-intensive service when executed in local execution mode, whereas in hybrid execution mode power service has consumed the largest computing time. This is because the computing power of the coarse-grained resource is substantially higher than medium and low. Therefore, the factor computing task quickly complete while execution of power service continues in the medium-granular service. Figure 6.3 clearly shows the existing communication-computing trade-off in hybrid resources. By distributing computation-intensive tasks to different resources, considering the mobile user preferences, we can significantly reduce the WAN latency in all three services, but the computing time of the services running on medium- and fine-

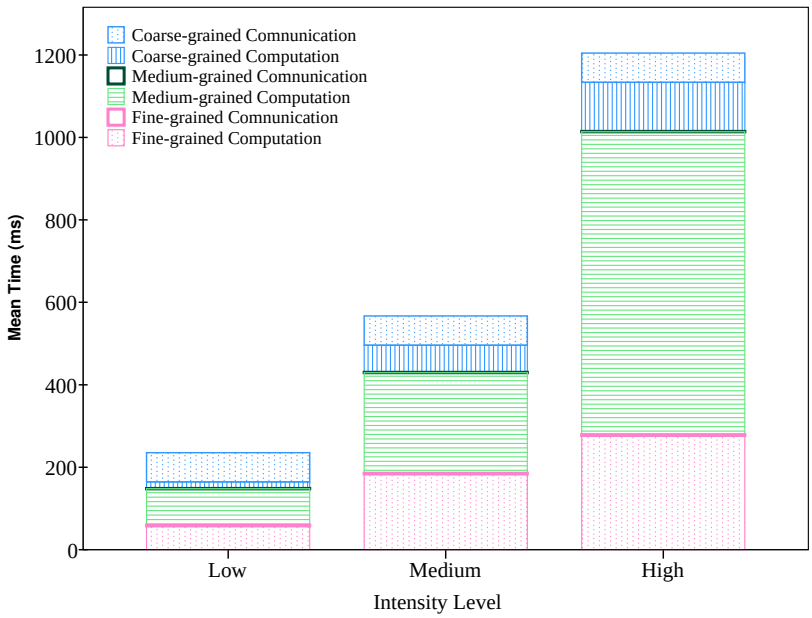


Figure 6.3: Computation-communication trade-off between three classes of heterogeneous grained resources in three level of workload intensity.

grained resources slightly increase compared to the coarse-grained resource.

In conclusion, the analysing measured results of RTT unveils that the proposed hybrid computing infrastructure can complete compute-intensive task execution in very shorter time compared with local execution mode. Therefore, leveraging heterogeneous granular hybrid cloud-based resources can enhance RTT toward optimal execution of resource-intensive applications on mobile devices (execution starts locally and intensive tasks are executed in remote resources).

6.1.2 Energy Consumption (EC)

The EC results of benchmarking experiment for the local and hybrid execution modes are presented in Tables 6.5 and 6.6. To ensure reliability of our generated data, we repeat the execution of each workload for 30 times and presented the results with 99% confidence interval.

Columns in the two Tables of 6.5 and 6.6 summarize the mean EC values of 30 iterations of each workload, standard deviation (SD) in values of each workload, error

Table 6.5: Energy consumption values with 99% confidence interval in local execution mode.

Workloads	Mean EC (mJ)	SD_EC	Error Estimate	EC with 99% CI
1	1499.1	24.9	11.7	1499.1(+/-)11.7
2	1723.1	72.3	34	1723.1(+/-)34
3	1743.6	27.8	13.1	1743.6(+/-)13.1
4	1903.1	16.8	7.9	1903.1(+/-)7.9
5	2234.5	39.8	18.8	2234.5(+/-)18.8
6	2272.3	30.1	14.2	2272.3(+/-)14.2
7	2531.4	49	23.1	2531.4(+/-)23.1
8	2608.5	65.9	31.1	2608.5(+/-)31.1
9	2622.2	90.6	42.7	2622.2(+/-)42.7
10	2671.8	35.1	16.5	2671.8(+/-)16.5
11	5487.1	68.9	32.5	5487.1(+/-)32.5
12	5517.3	66.9	31.5	5517.3(+/-)31.5
13	5855.9	43.4	20.4	5855.9(+/-)20.4
14	5971.3	57.9	27.3	5971.3(+/-)27.3
15	6506.3	108.2	51	6506.3(+/-)51
16	6509.8	146.8	69.1	6509.8(+/-)69.1
17	6695.2	64.1	30.2	6695.2(+/-)30.2
18	6718.7	75.9	35.8	6718.7(+/-)35.8
19	6792.4	68.7	32.4	6792.4(+/-)32.4
20	6961.5	114.1	53.8	6961.5(+/-)53.8
21	11834.2	161.3	76	11834.2(+/-)76
22	12782.5	314.3	148	12782.5(+/-)148
23	12937.5	202	95.1	12937.5(+/-)95.1
24	13575.3	357.5	168.4	13575.3(+/-)168.4
25	14368.4	317.1	149.4	14368.4(+/-)149.4
26	14414.7	343.5	161.8	14414.7(+/-)161.8
27	14707.9	215	101.3	14707.9(+/-)101.3
28	15800.7	297.5	140.2	15800.7(+/-)140.2
29	16164	165	77.7	16164(+/-)77.7
30	16285.1	331.1	156	16285.1(+/-)156

estimation, and mean EC values of 30 iterations with 99% confidence interval for each workload (total 30 workloads).

To statistically demonstrate and highlight the findings, descriptive statistics of mea-

Table 6.6: Energy consumption values with 99% confidence interval in hybrid execution mode.

Workloads	Mean EC (mJ)	SD energy	Error Estimation	EC with 99% CI
1	337.6	1.3	0.6	337.6(+/-)0.6
2	338	4.7	2.2	338(+/-)2.2
3	343.2	0.7	0.3	343.2(+/-)0.3
4	346.8	2.2	1	346.8(+/-)1
5	359.6	1.9	0.9	359.6(+/-)0.9
6	361	3.8	1.8	361(+/-)1.8
7	371.4	2.4	1.1	371.4(+/-)1.1
8	381.4	3.1	1.5	381.4(+/-)1.5
9	384.7	4	1.9	384.7(+/-)1.9
10	385.1	3.6	1.7	385.1(+/-)1.7
11	420.1	3.2	1.5	420.1(+/-)1.5
12	425.8	2.7	1.3	425.8(+/-)1.3
13	430	2.7	1.3	430(+/-)1.3
14	433.5	3.2	1.5	433.5(+/-)1.5
15	434.1	2.4	1.1	434.1(+/-)1.1
16	437.9	2.2	1.1	437.9(+/-)1.1
17	439	2.6	1.2	439(+/-)1.2
18	440.3	3.4	1.6	440.3(+/-)1.6
19	442.8	2.9	1.4	442.8(+/-)1.4
20	445.4	3.2	1.5	445.4(+/-)1.5
21	536.9	4.6	2.1	536.9(+/-)2.1
22	543.9	6.4	3	543.9(+/-)3
23	548.4	5.4	2.5	548.4(+/-)2.5
24	557.5	4.4	2.1	557.5(+/-)2.1
25	560.5	9.9	4.7	560.5(+/-)4.7
26	570.3	4.9	2.3	570.3(+/-)2.3
27	585.9	4.1	1.9	585.9(+/-)1.9
28	608.9	4	1.9	608.9(+/-)1.9
29	620.7	5.1	2.4	620.7(+/-)2.4
30	634.2	3.4	1.6	634.2(+/-)1.6

Table 6.7: Descriptive statistics of energy consumption in local and hybrid mode in real environment.

Intensity	Mode	N	Min EC (mJ)	Max EC (mJ)	Mean EC (mJ)
Low	Local	10	1499.10	2671.80	2180.96
	Hybrid	10	337.60	385.10	360.88
	Valid N (listwise)	10			
Medium	Local	10	5487.10	6961.50	6301.55
	Hybrid	10	420.10	445.40	434.89
	Valid N (listwise)	10			
High	Local	10	11834.20	16285.10	14287.03
	Hybrid	10	536.90	634.20	576.72
	Valid N (listwise)	10			
All	Local	30	1499.10	16285.10	7589.85
	Hybrid	30	337.60	634.20	457.50
	Valid N (listwise)	30			

sured EC data (mJ) for 30 workloads in both local and hybrid execution modes are presented in Table 6.7. Results are classified in three categories of intensity level: low, medium, and high, along with minimum and maximum EC data for all three levels of intensity in the last row of the table. Each category represents mean values of the minimum and maximum EC, beside mean EC.

The energy consumption for completing CMA in hybrid execution mode is conserved as high as 78% (min) to 86%(max), 92%(min) to 94%(max), and 95%(min) to 96%(max), in low, medium, and high intensity levels, respectively, compared with local execution mode. In total, our proposed model can save the energy usage from 78% (min) to 96% (max) for hybrid execution of compute-intensive application compared with local execution mode.

The Figure 6.4 is drawn from measured EC results in the Table 6.7 states considerable energy usage reduction in hybrid mode. Figure 6.4 shows that in all intensity levels the proposed hybrid model can reduce energy usage to complete the task execution in more than 83% compared with task execution in local mode.

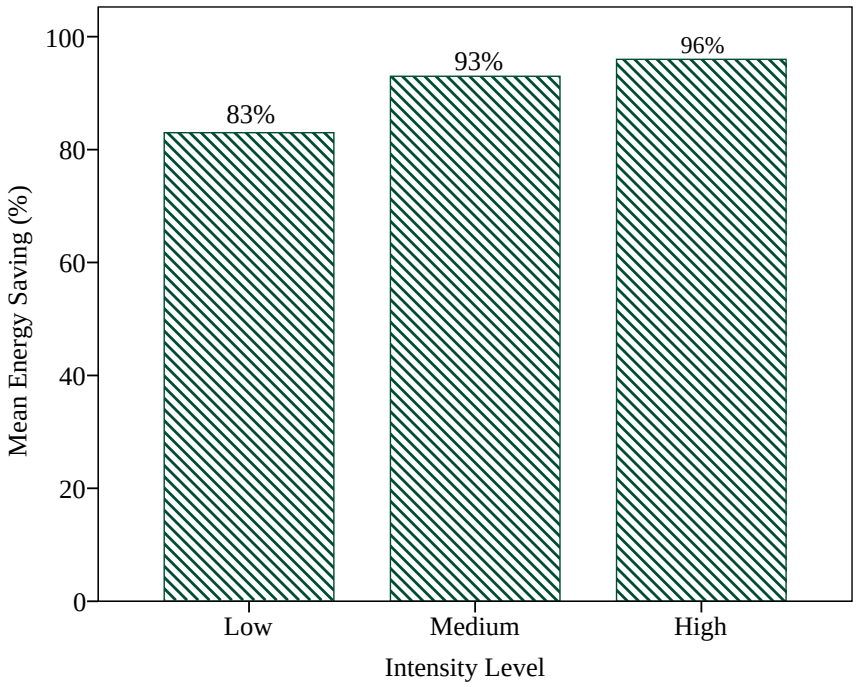


Figure 6.4: Energy saving average in hybrid mode from measured data.

Energy saving grows from low to high workload intensities as of 83%, 93%, and 96%, respectively. This indicates that energy saving is increased about 13% from the lowest level to the highest level of workloads' intensities in hybrid mode. So, it advocates that maintaining hybrid execution mode for compute-intensive application can significantly augment mobile devices and conserve their battery. Based on these results, we can conclude that for the performing high computation-intensive tasks on mobile device as a native application, more battery usage is expected that can quickly drain the battery even before completing the application execution (in worst case).

In local mode, the mean EC results for the low workloads is equal to 2180.96 mJ ($\simeq$ 2J), while with increase of workloads, the EC raises 4120.59 mJ ($\simeq$ 4J) and 12106.07 mJ ($\simeq$ 12J) in second and third workloads' category, receptively. By contrast, in hybrid mode when the workload is low, the mean EC is equal to 360.88 mJ ($\simeq$ 0.4J $\ll$ 2J), while with increase of workloads, the EC raises only 74.01 mJ ($\simeq$ 0.07J $\ll$ 4J) and 215.84 mJ ($\simeq$ 0.2J $\ll$ 12J) in second and third workloads' category, receptively. Finally, as can be seen,

the energy consumption in all intensity levels is less than 1 J which is more interesting compared with local mode where energy values vary from 2 J to 14 J. Therefore, these results unveil the advantage of utilizing cloud-based remote execution through our proposed hybrid model for CMA.

Also, we extend our analysis for measured EC with performing Paired Sample T-Test whose results are shown in Table 6.8. This test verifies the significant difference of energy usage between local and hybrid mode. The results of the test ensure that there is 99% correlation between mean energy consumption of local and hybrid datasets. Also, based on the Sig (2-Tailed) value equal to zero, we can conclude that there is a statistically significant difference between the mean EC of application execution for the local and hybrid mode environment.

Table 6.8: Paired Sample T-Test for measured data: Local EC & Hybrid EC

Paired Samples Correlations				**Paired Samples Test**	
Pair	N	Correlation	Sig.	t	Sig.(2-tailed)
(Local EC & Hybrid EC)	30	0.995	0.000	7.657	0.000

The Figure 6.5 demonstrates the average results of measured energy consumption for 30 different workloads with three intensity levels of low, medium, and high to execute compute-intensive application in two execution modes of local and hybrid. Execution of each workload is reiterated for 30 times to ensure data reliability and the EC value of each workload is mean value of 30 iterations. The red checkered bars are showing the EC related to local execution mode, whereas the green patterned bars are related to EC of hybrid execution mode. The energy consumption in hybrid mode is include of both data communication and CPU energy usage on mobile device.

Comparing two Figures 6.5 and 6.2 suggests a relationship between the RTT and energy consumption. By analysing the results, we observe that in local execution mode,

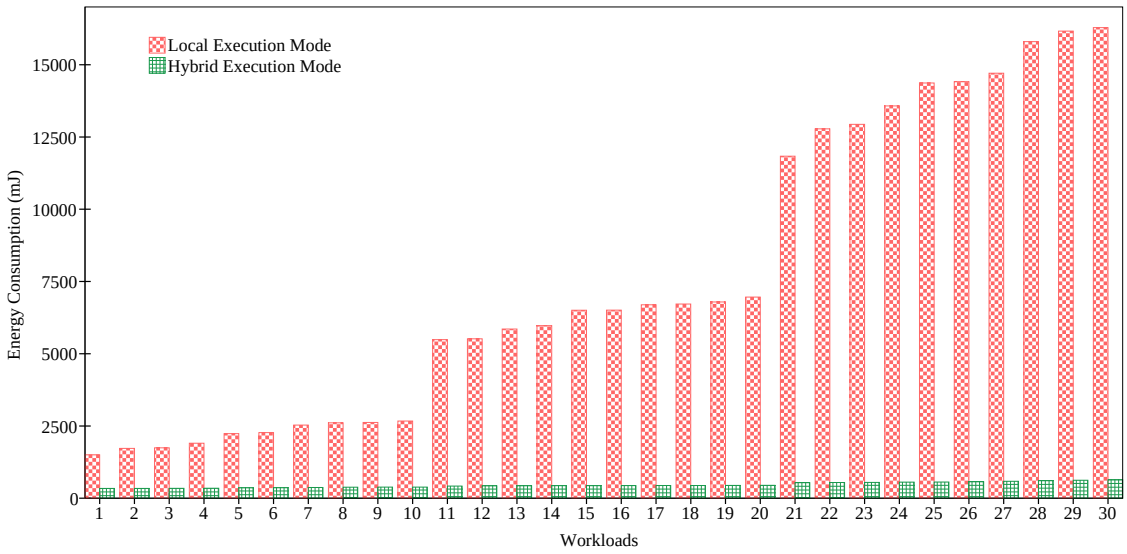


Figure 6.5: Comparison of application energy consumption for 30 workloads using measured data: Local mode vs Hybrid mode

shorter RTT executions consume less energy compared with longer RTT for medium- and high-intensity workloads in mobile device. Since the application is compute-intensive in this experiment, it uses noticeable processing power. Therefore, any rise in workload intensity on mobile devices, has high impact on RTT growth, results in increase in the energy usage.

By contrast, in hybrid mode increase in workload intensity has low impact on mobile device's energy usage. Although in this mode, some energy are dissipated to transfer data through WiFi interface between client and server, the high processing power of servers and distributing jobs between heterogeneous service providers noticeably decrease the total energy consumption in mobile device to execute CMA. Figure 6.6 depicts the average energy consumed by WiFi and CPU of mobile device in our tested hybrid prototype for three intensity levels of; low, medium, and high. The bars show that, although WiFi communication energy usage is increased from low to high level, the higher is the computation outsourcing rate, the higher is the WiFi energy consumption. Therefore, utilizing more powerful cloud resources can reduce response time which leads to saving energy of mo-

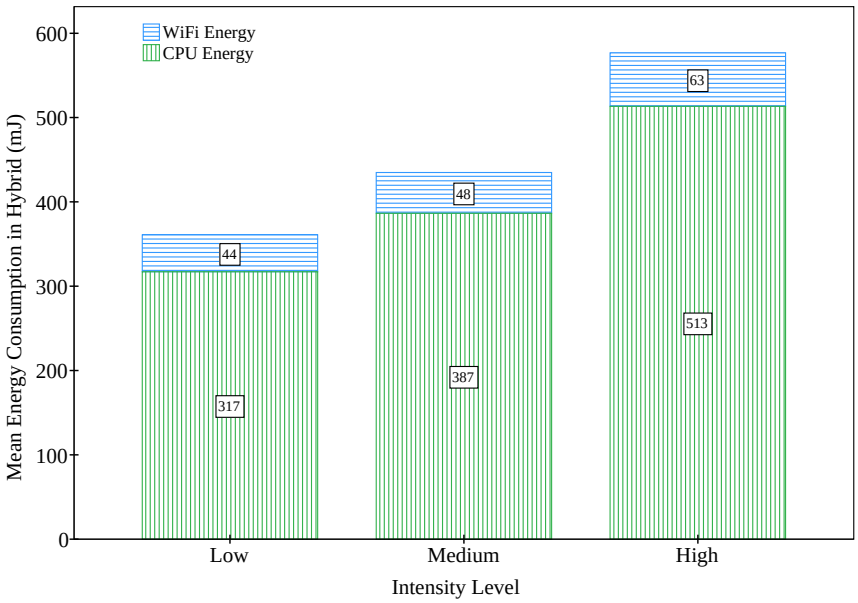


Figure 6.6: Average energy consumption of mobile device's WiFi and CPU in hybrid mode

bile device when transferring data and waiting for remote computation.

6.2 Validation Results

The time and energy results that are generated using statistical modelling are presented in this section. The results are presented with the help of descriptive statistics, tables, and charts.

6.2.1 Round-Trip Time (RTT)

The RTT data are generated via statistical modelling are presented in Table 6.9.

The results advocate significant improvement in RTT when execution takes place on our hybrid execution mode. Table 6.10 shows descriptive statistics of calculated RTT data (ms) for 30 workloads in both local and hybrid execution modes. Results are classified in three categories of intensity levels: low, medium, and high, along with minimum and maximum RTT data for all three levels of intensity in the last row of the table. Each category represents calculated mean values of the minimum and maximum RTT, beside RTT average for each 10 workloads as column N shows the number of workloads.

Table 6.9: RTT results generated via statistical modelling when executing CMA in local and hybrid modes

Workload Number	Local RTT (ms)	Hybrid RTT (ms)
1	4615.54	1251.41
2	4955.8	1253.5
3	5325.5	1255.62
4	5689.51	1257.77
5	6085.13	1259.96
6	6472.81	1262.18
7	6894.53	1264.43
8	7306.07	1266.71
9	7754.05	1269.02
10	8156.95	1271.37
11	15956.85	1405.56
12	16512.99	1409.1
13	17122.77	1412.66
14	17705.59	1416.24
15	18344.37	1419.84
16	18954.15	1423.46
17	19622.17	1427.1
18	20258.87	1430.76
19	20956.32	1434.44
20	21620.14	1438.13
21	37993.85	1865.56
22	38962.49	1870.13
23	40010.64	1874.69
24	41012.33	1879.26
25	42095.79	1883.82
26	43130.5	1888.38
27	44249.43	1892.93
28	45317.38	1897.49
29	46471.95	1902.04
30	47573.21	1906.58

The hybrid execution mode can save the time to complete the CMA about 73% to 84%, 91% to 93%, and 95% to 96%, in low, medium, and high intensity levels, respectively compared with local execution mode. In total, the proposed model can execute the compute-intensive tasks at least about 73%, at most 96% and in average 93% quicker than local mode.

The Figure 6.7 figured from mean RTT results in the Table 6.10 states considerable achievements in RTT reduction to execute CMA. Figure 6.7 shows that in all intensity

Table 6.10: Descriptive statistics of RTT in local and hybrid mode via statistical model

Intensity	Mode	N	Min RTT (ms)	Max RTT (ms)	Mean RTT (ms)
Low	Local	10	4615.54	8156.95	6325.59
	Hybrid	10	1251.41	1271.37	1261.20
	Valid N (listwise)	10			
Medium	Local	10	15956.85	21620.14	18705.42
	Hybrid	10	1405.56	1438.13	1421.73
	Valid N (listwise)	10			
High	Local	10	37993.85	47573.21	42681.76
	Hybrid	10	1865.56	1906.58	1886.10
	Valid N (listwise)	10			
All	Local	30	4615.54	47573.21	22570.92
	Hybrid	30	1251.41	1906.58	1523.00
	Valid N (listwise)	30			

levels the proposed hybrid model can save the time about 80 % compared with task execution in local mode. Time saving grown up from low level to high level workload intensity as 80%, 92%, and 96%, respectively. This indicates that time saving achievements are increased about 16% with increase of workloads' intensity in hybrid mode, which is a positive point to leverage remote execution for executing higher resource-intensive tasks on mobile devices.

In local mode, the results in Table 6.10 show that when the workload intensity is low, the mean RTT is equal to $6325.59(ms)$ ($\simeq 6s$). With increase of workloads in medium and high intensity levels, the RTT significantly grows. The low-medium and medium-high RTT differences are as high as $12379.83(ms)$ ($\simeq 12s$) and $36356.17(ms)$ ($\simeq 36s$).

By contrast, in hybrid mode when the workload intensity is low, the mean RTT is equal to $1261.20(ms)$ ($\simeq 1s$). Unlike local execution mode, in hybrid mode increase in workload intensity, causes insignificant rise in RTT values. Low-medium and medium-high RTT differences are as low as $160.53(ms)$ ($\simeq 0.2s$) and $624.9(ms)$ ($\simeq 0.6s$) in second and third workloads' intensity groups, receptively. Therefore, these results signify the advantages of utilizing cloud-based remote execution through our proposed hybrid model

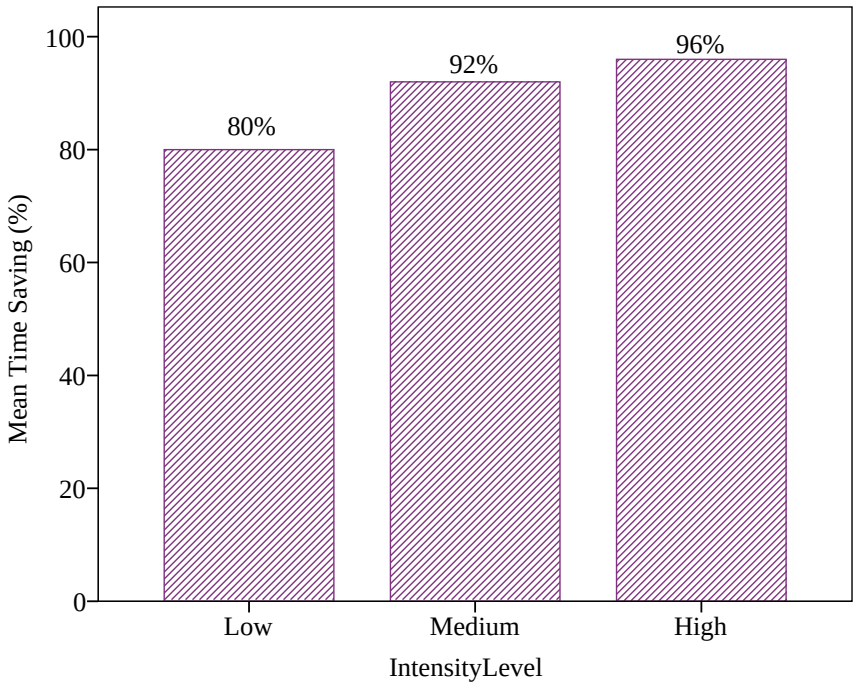


Figure 6.7: Time saving average in hybrid mode with statistical modelling

for mobile cloud applications.

In conclusion, based on these results, it can be concluded that for performing high computation-intensive tasks on mobile device (as a native application), there should be significantly higher execution time, if processing of request does not fail due to insufficient resources (CPU and RAM). From comparing Figure 6.7 with Figure 6.1 it is observed that in all intensity levels the proposed hybrid model can save the time 80 % more compared with task execution in local mode.

However, in order to better demonstrate the significance of time achievement, we extend our analysis to Paired Sample T-Test. The results of Paired Sample T-Test are presented in Table 6.11 verify this significant RTT differences. The test reports 99% correlation between execution time of local and hybrid datasets. Also, based on the Sig (2-Tailed) value equal to zero, we can conclude that there is a statistically significant difference between the mean RTT of application execution for the local mode and hybrid mode environment.

Table 6.11: Paired Sample T-Test: Local RTT & Hybrid RTT from statistical modelling

Paired Samples Correlations				**Paired Samples Test**	
Pair	N	Correlation	Sig.	t	Sig.(2-tailed)
(RTT_Local & RTT_Hybrid)	30	0.990	0	7.565	0

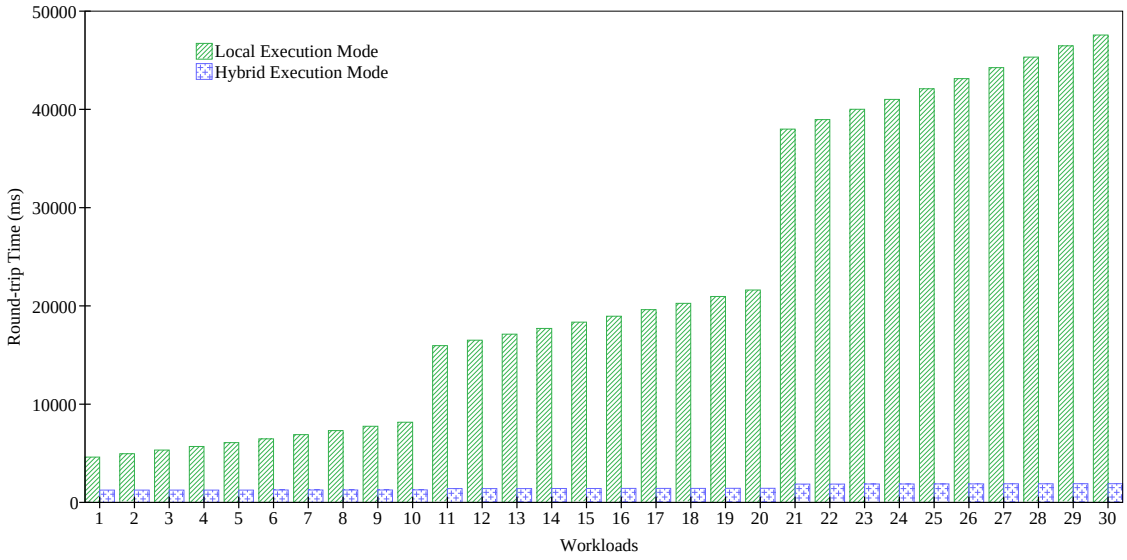


Figure 6.8: Comparison of application round-trip time for 30 workloads generated using statistical modeling: Local mode vs Hybrid mode

The Figure 6.8 compares the calculated round-trip time for 30 different workloads with three intensity levels (i.e., low, medium, and high) to execute compute-intensive application in two execution modes of local and hybrid. The bars in this figure represent the mean value of 30 execution iterations for each workload. The green stripped bars are showing the RTT related to local execution mode, whereas the blue plus-marked bars are related to hybrid execution mode's RTT when the computation-intensive part(s) of the application are running outside of the mobile device in remote computing servers.

In local execution mode, low-intensity workloads complete application execution in significantly shorter time compared to the medium- and high-intensity workloads that take longer to finish execution in mobile device. In local execution mode, increase in workload intensity has high impact(s) on RTT growth due to CPU and memory constraint of mobile devices. By contrast, increase in workload intensity in hybrid mode has low impact(s) on

RTT growth of the CMA, using high computing processors with large storage.

It is interesting to note that though the chart 6.8 -for these workloads- demonstrates that hybrid MCC is always beneficial, but when the compute-intensity of workloads is high, the time saving is more. Therefore, considering implications of executing CMA in hybrid mode (e.g., cloud service and wireless communication cost, security and privacy), using this platform is not recommended for very low level compute-intensive mobile applications.

In conclusion, results validate that the proposed hybrid mode can complete compute-intensive task execution in very shorter time compared with local execution mode. Therefore, leveraging heterogeneous granular cloud-based resources leads to RTT reduction of outsourcing resource-intensive tasks on mobile devices.

6.2.2 Energy Consumption (EC)

The results of EC of workloads generated via statistical modelling related to local and hybrid execution mode are presented in Table 6.12. The results highlight significant achievements in conserving energy which is supporting the results of benchmarking. Descriptive statistics of energy consumption (EC) data in (mJ) for 30 workloads are summarized in Table 6.13 in both local and hybrid execution modes. Results are classified in three categories of intensity level: low, medium, and high along with minimum and maximum EC data for all three levels of intensity in the last row of the Table. Each category represents calculated mean values of the minimum and maximum EC, beside EC average for each 10 workloads as column N shows the number of workloads.

The results of Table 6.13 shows that hybrid execution mode can reduce the energy consumption for performing CMA about 76% (min) to 86% (max), 92% (min) to 94% (max), and 95% (min) to 96%(max), in low, medium, and high intensity levels, respec-

Table 6.12: EC results generated via statistical modelling when executing CMA in local and hybrid modes

Workload Number	Local EC (mJ)	Hybrid EC (mJ)
1	1501.35	364.27
2	1703.73	365.51
3	1763.1	366.48
4	1940.19	367.2
5	2221.35	372.4
6	2272.53	372.56
7	2511.51	373.18
8	2632.94	373.7
9	2642.75	375.35
10	2726.27	377.7
11	5331.38	412.95
12	5524.4	415.64
13	5889.76	420.67
14	6089.76	421.67
15	6488.22	422.07
16	6658.26	422.5
17	6766.86	422.95
18	6915.09	423.21
19	7160.94	423.6
20	7231.4	424.48
21	12470.41	576.37
22	12928.31	577.16
23	13455.35	578.91
24	13991.68	579.5
25	14331.93	580.02
26	14717.73	580.89
27	14999.16	581.98
28	15464.38	582.91
29	15718.25	585.15
30	16041.37	585.83

tively compared with local execution mode. In total, the proposed model can save the energy usage at the time of executing compute-intensive applications at least about 76%, at most 96% and in average 94% compare with local mode.

The bar chart 6.9 figured from mean EC results in the Table 6.13 depicts considerable energy usage reduction in CMA execution. Figure 6.9 shows that in all intensity levels the proposed hybrid model can reduce energy usage to complete the task execution in more than 83 % compare with task execution in local mode. Energy saving grows from low level

Table 6.13: Descriptive statistics of energy consumption in local and hybrid mode for statistical method .

Intensity	Mode	N	Min EC (mJ)	Max EC (mJ)	Mean EC (mJ)
Low	Local	10	1501.35	2726.27	2191.57
	Hybrid	10	364.27	377.70	370.83
	Valid N (listwise)	10			
Medium	Local	10	5331.38	7231.40	6405.61
	Hybrid	10	412.95	424.48	420.97
	Valid N (listwise)	10			
High	Local	10	12470.41	16041.37	14411.86
	Hybrid	10	576.37	585.83	580.87
	Valid N (listwise)	10			
All	Local	30	1501.35	16041.37	7669.68
	Hybrid	30	364.27	585.83	457.56
	Valid N (listwise)	30			

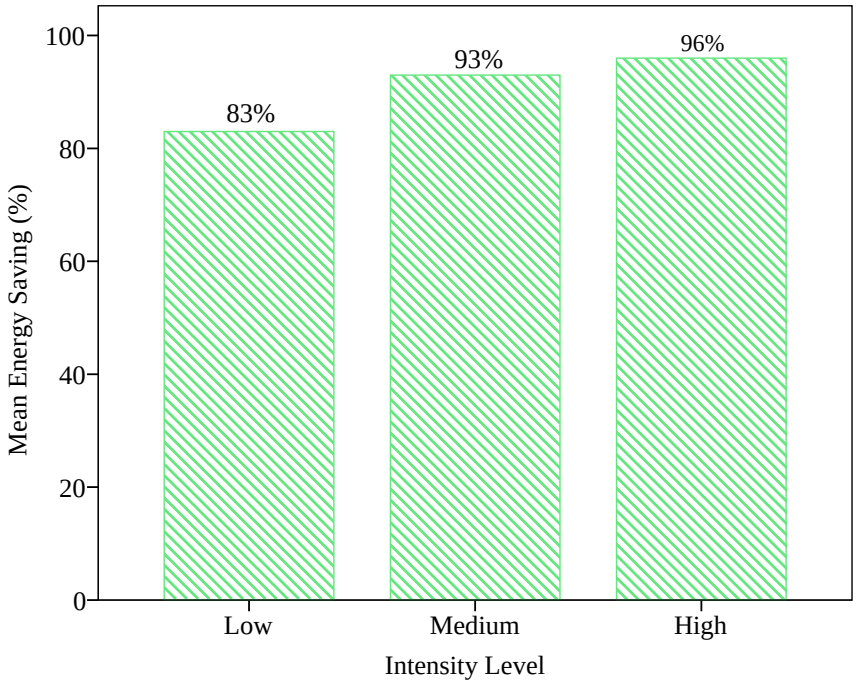


Figure 6.9: Energy saving average in hybrid mode through statistical modelling

to high level workloads' intensity as 83%, 93%, and 96%, respectively. This indicates that energy saving achievements are increased about 13% from lower level to upper level of workloads' intensity in hybrid mode, which is a positive point to leverage our proposed horizontally heterogeneous granular servers for augmenting mobile devices. The proposed work in this thesis realized longer lasting battery on mobile devices for executing higher

resource-intensive tasks.

The results in Table 6.13 advocate that, in local mode, increase in workload intensity significantly raises the energy dissipation. The mean EC for low workload intensity is 2191.57 (mJ) ($\simeq$ 2J), whereas it is 6405.61 and 14411.86 for medium and high. The difference between low and medium energy consumption is as high as 4214.04 (mJ) ($\simeq$ 4J) and the difference between medium and high intensity workloads is as significant as 12220.29 (mJ) ($\simeq$ 12J). However, in hybrid mode when the workload is low, the mean EC is equal to 370.83 (mJ) ($\simeq$ 0.4J $\ll$ 2J), while with increase of workloads, the EC raises 50.14 (mJ) ($\simeq$ 0.05J $\ll$ 4J) and 210.04 (mJ) ($\simeq$ 0.2J $\ll$ 12J) in second and third workloads' category, receptively. Finally, as can be seen, the energy consumption of hybrid mode at all intensity levels are less than 1J (364.47, 585.83, and 456.56 mJ) which is much interesting compared with local mode energy usage varying from 2 J to 14 J. Therefore, these results reveal the advantage of utilizing cloud-based remote execution through our proposed hybrid model for CMA.

The energy results unveil that performing significantly high computation-intensive tasks on mobile device (as a native application) substantially increases the application energy consumption or in worst case drains the battery before the task is being completed.

Moreover, to verify the significant energy saving between local mode and hybrid mode, we extend our analysis with performing Paired Sample T-Test whose results are shown in Table 6.14. This test ensures that there is 98% correlation between mean energy consumption of local and hybrid datasets. Also, based on the Sig (2-Tailed) value equal to zero, we can conclude that there is a statistically significant difference between the mean EC of application execution for the local mode and hybrid mode environment.

The Figure 6.10 compares the calculated average energy consumption for 30 different workloads with three intensity level (i.e., low, medium, and high) to execute compute-

Table 6.14: Paired Sample T-Test: Local EC & Hybrid EC from statistical modelling

Paired Samples Correlations				**Paired Samples Test**	
Pair	N	Correlation	Sig.	t	Sig.(2-tailed)
(EC_Local & EC_Hybrid)	30	0.986	0.000	7.704	0.000

intensive application in two execution modes of local and hybrid. The bars in this figure represent the mean value of energy consumption of 30 execution iterations for each workload. The grey plus pattern bars are showing the EC related to local execution mode, whereas the red checkered pattern bars are related to hybrid execution mode's EC when the computation-intensive part(s) of the application are running outside of mobile device in remote computing servers.

Comparing two Figures 6.8 and 6.10 results the relationship between the RTT and energy consumption. The results in local execution mode confirm that the workload with shorter RTT consumes less energy compared with larger RTT of medium and high-intensity workloads in mobile device. Therefore, any rise in workload intensity on mobile devices has high impact(s) on execution time growth, results to increased energy usage. However, in hybrid mode increase in workload intensity has low impact(s) on mobile device's energy usage. Although this mode consumes battery energy to transfer data through WiFi interface between client and server, the high processing power of servers and distributing jobs between service providers noticeably decrease the total energy consumption in mobile device to execute CMA.

Moreover, the Figure 6.10 demonstrates that the proposed hybrid mode can complete cloud-based mobile compute-intensive task execution with less energy consumption compared with local execution mode. Although wireless communication, client-server data transmission delay, and system arbitrator consume energy, the total energy consumption in hybrid mode is still lower than local mode.

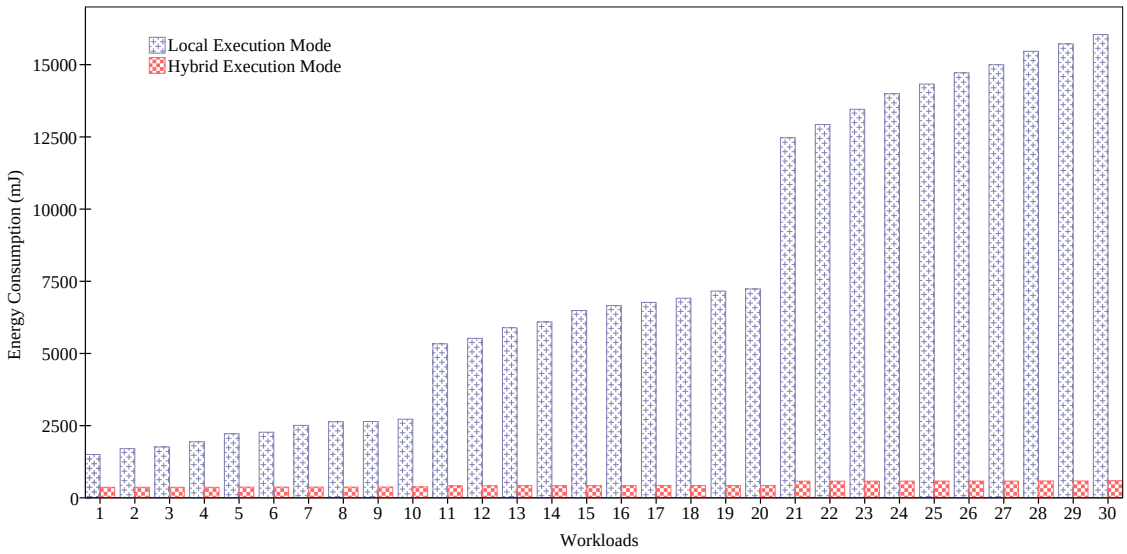


Figure 6.10: Comparison of application energy consumption for 30 workloads generated using statistical modeling: Local mode vs Hybrid mode

6.3 Discussion

The results of benchmarking analysis and statistical modelling presented in section 6.1 and 6.2 are individually supporting the lightweight feature of our propose framework. Significant correlations between data generated via these two evaluation methods help us to ensure that benchmarking and statistical modelling efforts are efficiently and effectively undertaken and the results are reliable. Comparing these two datasets, help us validate the performance gains of the proposed framework. The results are presented in two sections of round-trip time and energy consumption as follows.

Table 6.15: Comparison of RTT values in local and hybrid execution mode: statistical modelling vs benchmarking

Analysis Method	Local RTT		Hybrid RTT		RTT Saving	
					Numeric	Percentage
Statistical Modelling	Min	4615.54	Min	1251.4	3364.14	72.90%
	Max	47573.21	Max	1906.58	45666.63	95.90%
	Mean	22570.92	Mean	1523	21047.92	93.30%
Benchmarking	Min	4364.4	Min	1266.2	3098.2	70.40%
	Max	46631.9	Max	1878.27	44753.63	95.90%
	Mean	22295.58	Mean	1523.92	20771.66	93.10%

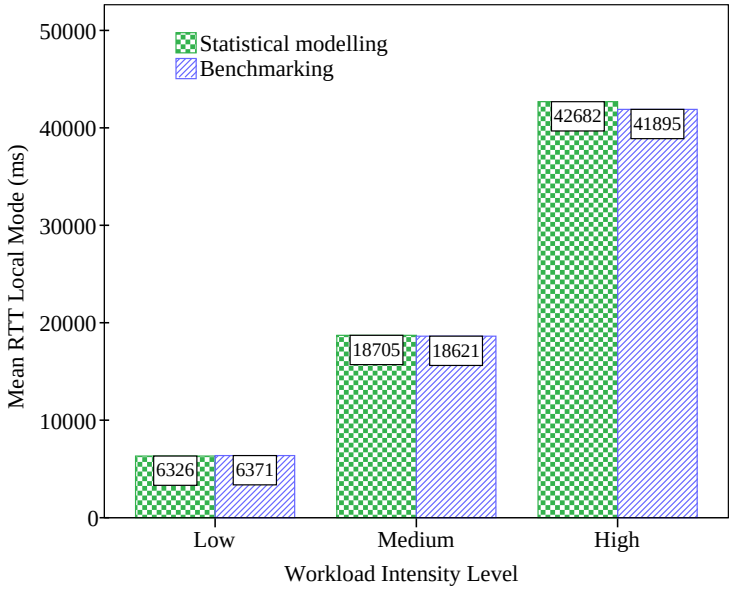


Figure 6.11: Comparison of RTT results in local Mode: Statistical model vs Benchmarking

6.3.1 Round-Trip Time (RTT)

The results of benchmarking and statistical modelling are analysed to verify validity of performance evaluation. Table 6.15 presents descriptive analysis of RTT time in local and hybrid modes collected via benchmarking and statistical modelling. The two most right columns in the table show the numerical and percentile of time saving when using our proposed framework. The minimum RTT saving values are as close as 72.9% and 70.40% when generated via statistical modelling and benchmarking, respectively. The maximum RTT saving values for statistical model and benchmarking is equally 95.90%. The mean time saving is also approximately identical which is 93.30% and 93.10% for statistical modelling and benchmarking respectively.

The illustration of the results of our comparison for RTT time are presented in Figures 6.11 and 6.12. The Figure 6.11, illustrates comparison of mean RTT of local execution mode in three intensity levels when generated using statistical model and benchmarking. The green checkered bars show the statistical data and the blue stripped bars depict the benchmarking results. Each couple of bars clearly supports each other and ensure that

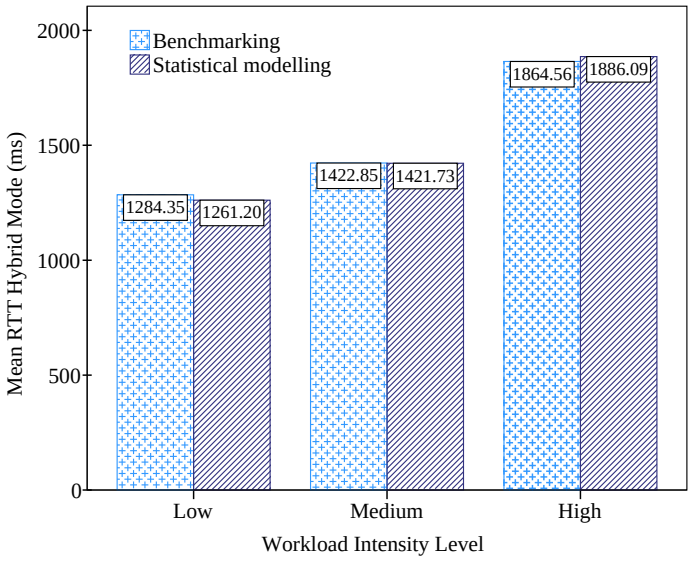
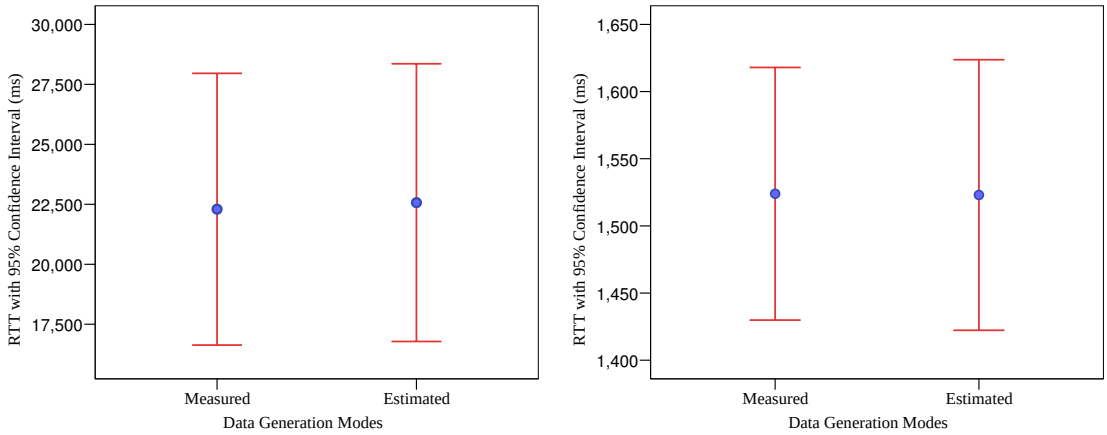


Figure 6.12: Comparison of RTT results in hybrid mode: Statistical model vs benchmarking

data collected using these two approaches are valid. The difference between each bar in each intensity level is negligible advocating adequacy of data collection in local execution mode.

Similarly, the Figure 6.12, illustrates comparison of mean RTT of hybrid execution mode in three intensity levels when generated using statistical model and benchmarking. The blue patterned bars show the benchmarking data and the blue stripped bars depict the



(a) Local RTT (b) Hybrid RTT

Figure 6.13: RTT Results with 95 % Confidence Interval: Benchmarking vs Statistical Modeling

statistical modelling results. Each couple of bars clearly supports each other and ensure that data collected using these two approaches are valid. The difference between each bar in each intensity level is negligible advocating adequacy of data collection in hybrid execution mode and testify validity of performance evaluation undertaken in this research.

Moreover, we present the Error bar charts 6.13 of data related to local and hybrid execution time generated using benchmarking and statistical modeling. The results are presented with 95 % confidence interval. The vertical lines in each chart highlight the 95% confidence interval range of round-trip time. The circle on each line shows the mean RTT value. The charts 6.13(a) and 6.13(b) verify the accuracy of the RTT data collection in our performance evaluation for local and hybrid modes. As shown, the confidence intervals range of benchmarking and statistical modelling has significant overlap that shows negligible differences in RTT.

6.3.2 Energy Consumption (EC)

Table 6.16 presents descriptive analysis of EC in local and hybrid modes collected via statistical modelling and benchmarking. The two most right columns in the Table summarize the numerical and percentile of energy saving when using the proposed framework. The minimum EC saving values respectively generated via statistical modeling and benchmarking are as close as 75.7% and 77.40%. The maximum EC saving values for statistical model and benchmarking are 96.30$ and 96.10% respectively. The mean energy saving is also approximately identical which is 93.90% and 94% for statistical and benchmarking respectively.

The illustrative view of the comparative results for EC is presented in Figures 6.14 and 6.15. The Figure 6.14, illustrates comparison of mean EC of local execution mode in three intensity levels when generated using statistical model and benchmarking. The

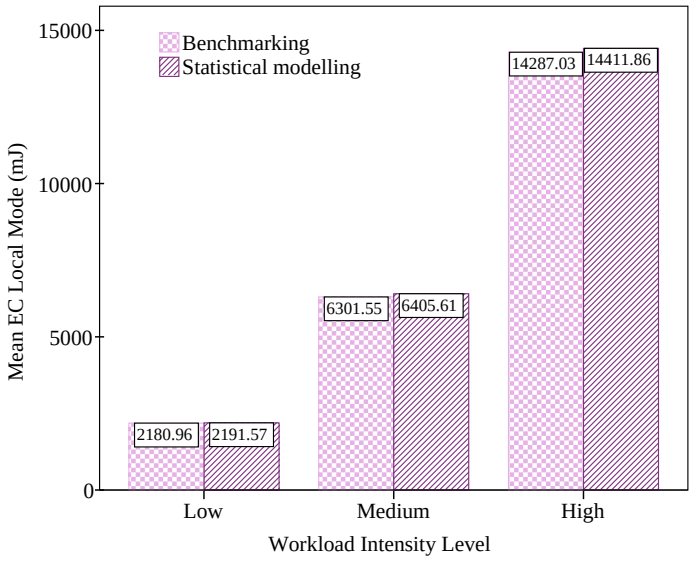


Figure 6.14: Comparison of EC results in local mode: Statistical model vs Benchmarking

purple stripped bars show the statistical data and the pink checkered bars depict the benchmarking results. Each couple of bars clearly supports each other and ensures that data collected using these two approaches are valid. The differences between each bar in each intensity level is negligible advocating adequacy of data collection in local execution mode.

Similarly, the Figure 6.15, illustrates comparison of mean EC of hybrid execution mode in three intensity levels when generated using statistical model and benchmarking. The brown stripped bars show the statistical data and the red checkered bars depict the benchmarking results. Each couple of bars clearly supports each other and ensures that

Table 6.16: Comparison of EC values in Local and Hybrid Execution Mode: Statistical Modeling vs Benchmarking

Analysis Method	Local EC		Hybrid EC		Energy Saving	
					Numeric	Percentage
Statistical Modeling	Min	1501.35	Min	364.27	1137.08	75.70%
	Max	16041.37	Max	585.83	15455.54	96.30%
	Mean	7669.68	Mean	457.56	7212.12	94%
Benchmarking	Min	1499.1	Min	337.6	1161.5	77.40%
	Max	16285.1	Max	634.2	15650.9	96.10%
	Mean	7589.85	Mean	457.5	7132.35	93.90%

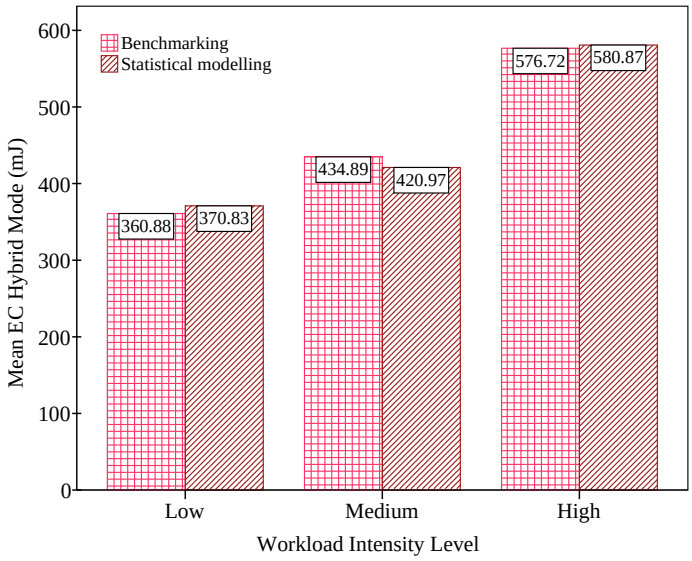


Figure 6.15: Comparison of EC results in hybrid mode: Statistical model vs Benchmarking

data collected using these two approaches are valid. The differences between each bar in each intensity level is minimal which testify adequacy of data collection in hybrid execution mode.

Figures 6.16 shows combined 3-D view of local and hybrid RTT values for three intensity levels and compare the values that are generated using statistical model and benchmarking experiments. The bars in this chart clearly advocate performance gain of the proposed model and validity of data collection approaches in this thesis.

Figures 6.17 shows combined 3-D view of local and hybrid EC values for three intensity levels and compare the values that are generated using statistical model and benchmarking experiments. The bars in this chart clearly advocate performance gain of the proposed model and validity of data collection approaches in this thesis.

Moreover, the error bars chart of data related to local and hybrid consumed energy generated using benchmarking and statistical modelling are illustrated. The results are presented with 95 % confidence interval in Figures 6.18. The vertical lines in each chart highlight the 95% confidence interval range of consumed energy. The circle on each line

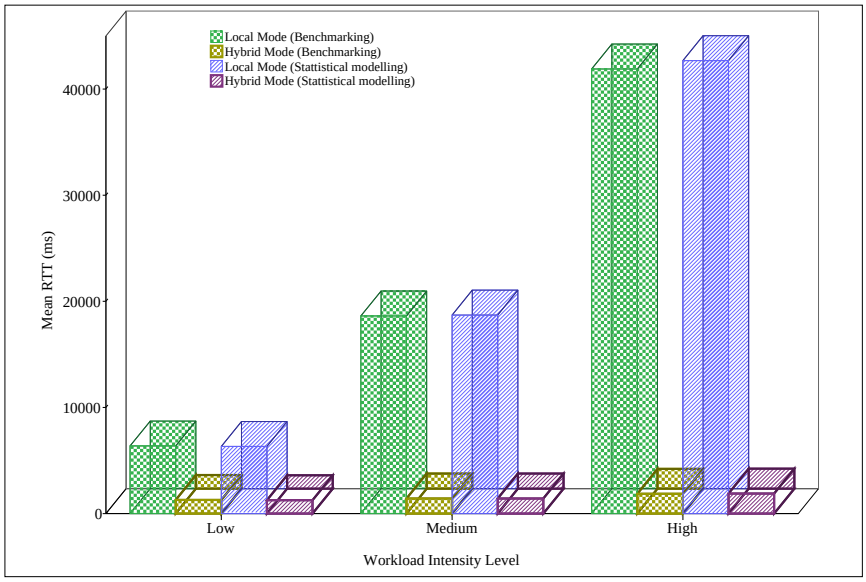


Figure 6.16: Comparison of RTT results in hybrid and local modes: Statistical model vs Benchmarking

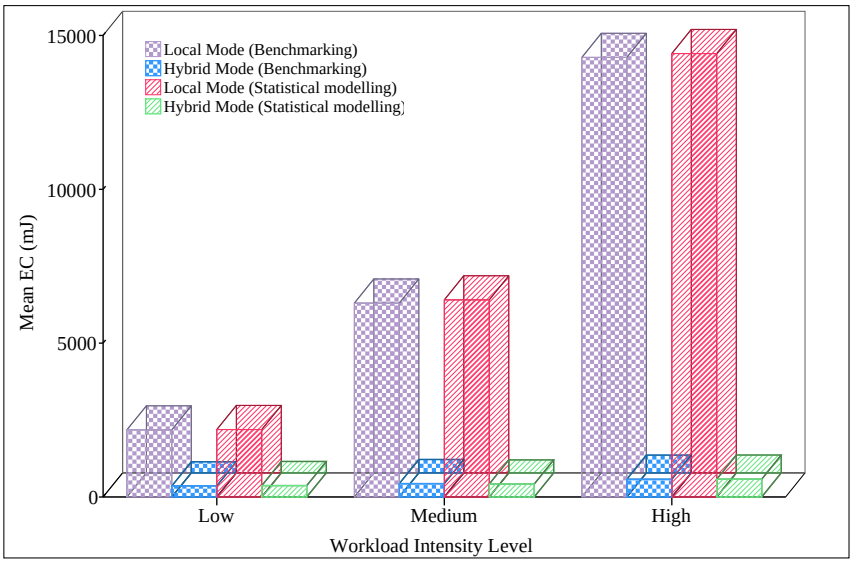
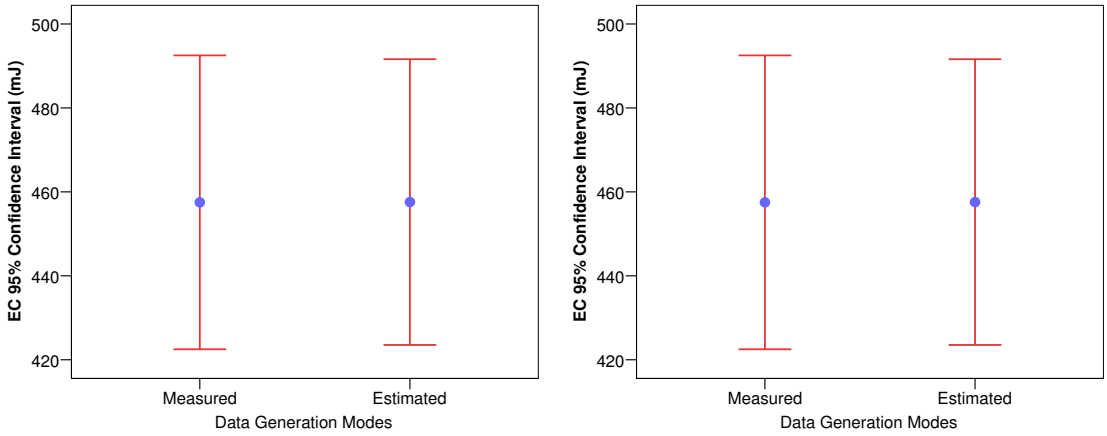


Figure 6.17: Comparison of EC results in hybrid and local modes: Statistical model vs Benchmarking

shows the mean EC value. The charts 6.18(a) and 6.18(b) verify the accuracy of the consumed energy data collection in our performance evaluation for local and hybrid modes. As shown, the confidence intervals range of benchmarking and statistical modelling has significant overlap that shows negligible differences in consumed energy. Small difference in mean values and confidence interval range testifies validity of performance evaluation.



(a) Local Mode (b) Hybrid Mode

Figure 6.18: Mean energy consumption results with 95 % confidence interval: Benchmarking vs Statistical modelling

6.4 Conclusions

This chapter presents the results of our performance evaluation. Data for this evaluation is generated using benchmarking and statistical modelling for RTT and EC performance metrics. The results of RTT analysis generated via statistical model advocate average of 93.3% RTT saving when performing the compute-intensive applications using our proposed framework which is significantly high. The maximum time saving is as high as 95.90%. The RTT results of our benchmarking model which is 93.10% significantly validates the findings of statistical model. These time savings are strong evidences that our proposed framework can significantly improve application responsiveness due to its lightweight nature.

In terms of energy-efficiency, our performance gains are significant. The EC analysis results generated and collected via statistical model testify average of 94% energy saving when performing the compute-intensive applications using our proposed framework which is remarkable achievement toward green mobile computing. The maximum energy saving is as high as 96.30%. The EC results of our benchmarking model which is 93.90% significantly validates the findings of statistical model. These energy savings

are strong evidences that our proposed solution is an energy-efficient MCC solution that can remarkably improve application energy efficiency because of lightweight design and implementation of the work.

The results of performance evaluation advocate that in spite of arbitration and networking overheads in utilizing hybrid cloud-based computing resources, the responsiveness and energy efficiency of compute-intensive mobile applications can be significantly enhanced.

CHAPTER 7

CONCLUSIONS AND FUTURE WORKS

This chapter presents the conclusions of this thesis and identifies the future works. We describe how the aim and objectives of this research are fulfilled. We also present the contributions of this thesis and highlight the significance of the work undertaken in this study. The scholarly publications, including peer reviewed ISI journal articles, conference papers, and other publications that are produced as the results of the work carried out in this thesis are listed. Finally, we present limitations of this study and possible future works.

In section 7.1, we describe how aim and objectives of this study are attained. Section 7.2 presents contributions of this thesis. Significance of this work is presented in section 7.3. The ISI articles that are produced in coarse of this study are listed in section 7.4 and Limitations and future works are appeared in section 7.5.

7.1 Aim and Objectives

In this research, we aimed to achieve efficient computation outsourcing for compute-intensive mobile applications using horizontally heterogeneous granular cloud-based resource. We explain how we could attain the research aim by realizing the following objectives.

7.1.1 Investigate the Recent Cloud-based Mobile Computation Outsourcing Approaches to Identify Current Research Problems

We fulfil the objective and review the most credible works reported in articles collected from scholarly digital libraries using the University Malaya access portal. Ma-

jor mobile and cloud computing conference and journals are visited to ensure thorough browsing of the recent literature. We analyse and synthesize the recent works in mobile computation outsourcing and identify seven problems, including architectural, communication and computation latency, energy constraint, and elasticity problems. We specify the communication and computation latency as the most significant problem to address in this research.

7.1.2 Analyse the identified research problem to determine its impact on energy efficiency and response time of cloud-mobile applications.

In fulfilling this objective, we verify the identified research problem of computation and communication latency created by vertically heterogeneous cloud-based resources to address in this research. In order to analyse the significance of this research problem, an analytical analysis is carried out using real time experiment on Android-based smartphone using Amazon EC2 Cloud computing instances for the execution time and energy consumption of a compute-intensive mobile application. The results of the analysis are reported in chapter 3. In order to gain insight into the problem, we implemented a compute-intensive mobile application and run it for 30 different workloads to harvest their execution time and energy consumptions. In our experiment, execution took place in three runtime environments; one native runtime inside the mobile device and three Amazon EC2 virtual machines located in three heterogeneous granular locations, namely Singapore, California, and Ireland regions. Though analysis of the results of experimental analysis, we testify severity and significance of the selected problem.

7.1.3 Propose a lightweight mobile cloud computing framework to achieve efficiency in response time and energy consumption of compute-intensive mobile application.

The objective is realized by proposing a horizontally heterogeneous hybrid MCC framework to decrease execution time and energy consumption of utilizing cloud computing resources for CMA execution. The main design principle employed in this framework is that building a horizontally heterogeneous hybrid granular resources consists of multitude of heterogeneous computing resources featuring dissimilar granularities (proximity and computing power) creates a localized elastic resourceful cloud that offers efficient computational services to mobile clients. The optimum resource allocation in this hybrid heterogeneous resources is that resources can be allocated to tasks based on their intensity, time sensitivity, and financial cost of utilizing those services.

7.1.4 Evaluate the performance of the proposed solution.

We attain this objective by evaluating the proposed framework via benchmarking experiment using an android-based smartphones, desktop, laptop and a cloud VM instance from Amazon EC2. We performed 30 different workloads in two execution modes of local and hybrid. The execution of each workload is repeated 30 times for the sake of reliability. Round-trip time and energy consumption of the mobile application in both modes are measured and analysed. Out performance analysis results unveil that utilizing our framework to perform compute-intensive application improves the RTT by 93.1% and energy consumption by 94% in average compared with the local execution mode.

We validated the results of performance evaluation via statistical modelling. Regression analysis is used as a common approach to derive accurate statistical models of execution time and energy consumption of mobile application on local and hybrid execution modes. We validate the devised models using split-sample validation approach. The

findings of benchmarking are compared with the statistical model results to validate our framework. Validation results confirm that leveraging our framework to execute compute-intensive mobile application reduces the RTT by 93.3% and energy consumption by 93.9% in average compared with the local execution mode.

7.2 Contributions

In this research, we have produced several contributions to the body of knowledge as following.

7.2.1 Taxonomy of Heterogeneity Roots in Mobile Cloud Computing

We produced the taxonomy of heterogeneity roots in MCC. We comprehensively reviewed MCC from heterogeneity point of view by critically analysing articles extracted from scholarly indices such as IEEE, ACM, and Elsevier that are presenting the state-of-the-art researches to devise the taxonomy of the roots of heterogeneity in MCC. The proposed taxonomy presents five major roots of heterogeneity in MCC as hardware, platform, feature, API, and network. Our study, briefly appeared in chapter 2 and published in (Sanaei, Abolfazli, Gani, & Buyya, 2014), is the early comprehensive survey of heterogeneity in MCC that is published in the literature.

7.2.2 Taxonomy of Heterogeneity Dimensions

We devised the taxonomy of heterogeneity dimensions in MCC. We describe and taxonomize heterogeneity dimensions in MCC into two categories of vertical and horizontal. Analysing roots of heterogeneity in MCC results significant differentiation in silo of mobile devices, cloud, and wireless networks. We distinguish vertical from horizontal heterogeneity in mobile devices, cloud resources and networking infrastructures and elaborate the concept with the help of real examples. The findings of this contribution are presented in chapter 2 and published in the literature.

7.2.3 Taxonomy of Heterogeneous Mobile Computation Outsourcing

We produce the taxonomy of recent heterogeneous granular outsourcing model by comprehensively reviewing the most credible cloud-based mobile computation outsourcing approaches in MCC. We horizontally and vertically analyse and synthesize the recent works in MCC and taxonomized outsourcing models into two categories of horizontal and vertical; the horizontal approaches are consists of those works that utilized vertically heterogeneous resources, namely coarse granular, medium granular, and fine granular. The results of our work are appeared in chapter 2 of this thesis and published in a survey paper listed in section 7.4.

7.2.4 Performance Evaluation of Vertically Heterogeneous Mobile Computation Outsourcing on CMAs

We contributed to the body of knowledge by empirically analysing performance of vertically heterogeneous granular cloud-based resources on compute-intensive mobile applications. With the help of benchmarking experiments, we identify the computation and communication overhead/latency of utilizing heterogeneous granular cloud-based resources when are utilized for execution of compute-intensive application. The results of this analysis are presented in chapter 3. The analysis shows noticeable overhead when inappropriate resources are selected regardless of the application intensity and time sensitivity We found that leveraging coarse-grained cloud-based resources for a low intensity time-sensitive application is not always beneficial and originates high overhead due to the high communication latency of accessing the virtual services provided by distant coarse-grained resources. This overhead typically jeopardizes the performance gain of computation outsourcing, especially for low intensity workloads.

7.2.5 Lightweight Heterogeneous Hybrid Mobile Cloud Computing Framework

We produce a research that features horizontally heterogeneous granular cloud-based resources. While focus in traditional solutions was on vertically heterogeneous MCC resources, this is the first effort that uses horizontally heterogeneous hybrid resources in convergence of coarse-, medium- and fine-grained cloud-based resource. Our framework is designed based on resource-oriented architecture to builds a lightweight framework for efficient execution of compute-intensive mobile applications by employing multitude of varied granular computing resources. This solution meets its aim by performing compute-intensive processing of mobile applications, remotely in multi-layer heterogeneous cloud-based resources. The proposed hybrid MCC framework is reported in chapter 4 and published in the literature.

7.2.6 Performance Evaluation and Validation of the Framework

The empirical and analytical evaluation of the system is produced using benchmarking and statistical models. Performance evaluation using benchmarking analysis is performed on android-based smartphone and three classes of horizontally heterogeneous cloud-based resources. The statistical models are produced via observation-based modelling approach in which dataset of independently replicated data are generated to train the regression model. The models are validated using split-sample approach and the valid model is used to validate the performance of our proposed framework. The performance evaluation and validation models are reported in chapter 5 and the results are presented in chapter 6. Schematic and statistical analysis of the results unveiled the functionality, feasibility, lightweight nature, and high performance of our proposed framework and advocate that objectives of this study are fulfilled and the aim is realized.

7.3 Significance of the Work

Several key features that are considered in design and development of this framework could enhance the significance of this work that are briefly discussed as follows.

- **Cross-Platform:** One significant feature of this proposal is its cross-platform feature that inherits from SOC in design and implementation. Thus, any mobile device including smartphone, laptop, tablet with any operating system including Android, iOS, Blackberry, and Windows mobile can leverage the benefits of this proposal with least possible overhead. This platform-independent design philosophy enables varied computing device that has capability to run the services to serve as cloud-based resources, regardless of its OS and architecture.

- **Horizontally Heterogeneous Hybrid Granular Resources:** This model employs horizontally heterogeneous hybrid granular computing resources in three granularity levels of FGR, MGR, and CGR. The FGR is in proximity of mobile device with limited scalability, while the CGR is located in distance with high scalability. The MGR has medium proximity and scalability. Such hybrid resources provide the opportunity of performing a computation-communication trade-off for performing cloud-based mobile computation outsourcing method.

- **Loosely Coupled Architecture:** Our framework enables development of loosely coupled compute-intensive mobile applications that can be simply separated and integrated without any partitioning overhead. Enhancing execution of service-based mobile applications in MCC produces remarkably least overhead compared to tightly coupled codes in non-service-based mobile applications. Loosely coupled architecture mitigates development complexity and temporal cost of application development, and also characterizes the application with lightweight feature.

- **Arbitrated Architecture:** One of the key features provisioned in this framework is the deployment of arbitrator. Arbitrator is a central trustworthy entity, that is concerned to be mobile network operator in this research- that arbitrates the entire outsourcing process. The arbitrator currently keeps track of all cloud-based resources and arbitrates the entire outsourcing platform to ensure that remote execution takes place with low overhead. Arbitrator's role can be enhanced by incorporating further services to it in future works.

- **Low Communication Overhead:** In this work, the system does not transmit the service code to the cloud-based resources. The compute-intensive codes are already installed in the cloud-based resources and can be called from any device without transmission of the code. Only input values are sent as request and the results are received as response. Hence, the communication overhead is remarkably shrinks. It reduces the mobile owner's data plan and contributes to reduce the wireless network load.

- **Enriched User Interaction:** In design and development of the proposed framework in this thesis, we have employed the asynchronous communication technology where mobile-cloud communications take place in background without freezing the application or the mobile client device. Utilizing asynchronous communication omits interaction distraction and remarkably enhances user interaction experience. While outsourcing process is executing in the background, mobile user can fully utilize the features of the device and applications with no distraction.

7.4 International Scholarly Publications

The list of publications related (in whole or part) to the research undertaken in this thesis is as follows.

Authored Publications

1. Z.Sanaei, S.Abolfazli, A.Gani, R. Buyya, Heterogeneity in Mobile Cloud Computing: Taxonomy and Open Challenges, **IEEE Communications Surveys & Tutorials** (Q1, IF=6.3), Vol 16, No.1 pp. 369-392, 10.1109/SURV.2013.050113.00090, Feb 2014 (ISI/SCOPUS Cited Publication).

2. Z.Sanaei, S.Abolfazli, T. Alizadeh, F. Xia, Hybrid Pervasive Mobile Cloud Computing: Toward Enhancing Invisibility, **Information**, Vol 16, No 11, Nov 2013 (ISI/SCOPUS Cited Publication).

3. Z.Sanaei, S.Abolfazli, A.Gani, Heterogeneity in Pervasive Computing: Taxonomy, Opportunities, and Challenges, **Pervasive and Mobile Computing** (Q1), Under Review, 2014, (ISI/SCOPUS Cited Publication).

4. Z. Sanaei, S. Abolfazli, A. Gani, Hybrid Mobile Cloud Computing Infrastructure for Pervasive Computing, **IEEE Transactions on Parallel and Distributed Computing** (Q1), under review, 2014, (ISI/SCOPUS Cited Publication).

5. Z. Sanaei, S. Abolfazli, A. Gani, Location Granularity of Vertically Heterogeneous Clouds in Mobile Cloud Computing: Energy and Time Analysis, **Supercomputing** (Q2), under review, 2014, (ISI/SCOPUS Cited Publication).

6. Z. Sanaei, S. Abolfazli, A. Gani, Empirical Analysis of Heterogeneous Granular Resources in Mobile Cloud Computing, **Post Graduate Research Excellence Symposium (PGRES)** 2013, University Malaya. **(Best Paper Award)**

7. Z. Sanaei, S. Abolfazli, A. Gani, M. Shiraz, SAMI: Service-Based Arbitrated Multi-Tier Infrastructure for Mobile Cloud Computing, IEEE Workshop on Mobile Cloud Computing **(MobiCC 2012)**, China, 2012, PP. 14-19 (ISI Cited Publication).

8. Z. Sanaei, S. Abolfazli, A. Gani, R. H. Khokhar, Tripod of Requirements in Horizontal Heterogeneous Mobile Cloud Computing, The 1st International Conference on Computing, Information Systems and Communications **(CISCO' 12)**, 2012, Singapore, PP.217-222. (ISI-Cited Publication).

9. Z. Sanaei, S. Abolfazli, A. Gani, A Lightweight Horizontally Heterogeneous Hybrid Mobile Cloud Computing Framework, **ACM MobiCom '14** , Maui, Hawaii, under review, 2014.

Co-Authored Publications

10. S.Abolfazli, Z.Sanaei, E.Ahmed, A.Gani, R.Buyya, Cloud-based Augmentation for Mobile Devices: Motivation, Taxonomies, and Open Challenges, **IEEE Communications Surveys & Tutorials** (World 1st ranked ISI CS journal, Q1, IF=6.31), Vol 16, No.1 pp. 337-368, Feb 2014, DOI:10.1109/SURV.2013.070813.00285 (ISI/SCOPUS Cited Publication).

11. S.Abolfazli, Z.Sanaei, A.Gani, F.Xia, L. T. Yang, Rich Mobile Applications: Genesis, Taxonomy, and Open Issues, **Journal of Network and Computer Applications** (Q1, IF=1.46), In Press, Sept 2013, 10.1016/j.jnca.2013.09.009, (ISI/SCOPUS Cited Publication)

12. S.Abolfazli, Z.Sanaei, A.Gani, An Experimental Analysis on Cloud-based Mobile Augmentation in Mobile Cloud Computing, Minor revision submitted, **IEEE Transactions on Consumer Electronics** (Q2, IF=1.087), accepted for publication, Feb 2014, (ISI/SCOPUS Cited Publication).

13. N. Aminzade, Z. Sanaei, S. Hafizah Hamid, Mobile Storage Augmentation in MCC: Taxonomy, Approaches, and Open Issues, Simulation Modelling Practice and Theory, revision received, 2014.

14. S. Abolfazli, Z. Sanaei, A. Gani, LPMCC: A Lightweight Proximate Mobile Cloud Computing Framework for Energy-efficient Responsive Mobile Computation Augmentation, **ACM MobiCom 2014 (Tier 1)**, under review, Maui, Hawaii, 2014.

15. S. Abolfazli, Z. Sanaei, M. Shiraz, A. Gani, MOMCC: Market-Oriented Architecture for Mobile Cloud Computing Based on Service Oriented Architecture , IEEE Workshop on Mobile Cloud Computing **(MobiCC 2012)**, Beijing, China, 2012, pp. 8-16 (ISI-Cited Publication).

16. S. Abolfazli, Z. Sanaei, A. Gani, Mobile Cloud Computing: A Review on Smartphone Augmentation, The 1st International Conference on Computing, Information Systems and Communications **(CISCO' 12)**, Singapore, 2012, pp. 199-204 (ISI-Cited Publication).

17. S. Abolfazli, Z.Sanaei, M.H. Sanaei, A.Gani, Mobile Cloud Computing: The-state-of-the-art, Challenges, and Future Researches, **Chapter in Encyclopedia of Cloud Computing, Wiley, 2014**, San Murugesan and Irena Bojanova (editors), Accepted, Feb 2014.

18. S. Abolfazli, Z. Sanaei, A. Gani, A Lightweight Mobile Cloud Computing Platform for Resource-intensive Mobile Applications, University Malaya Research Conference **(UMRC'13)**, Kuala Lumpur, Malaysia, Nov 2013.

19. S. Abolfazli, Z. Sanaei, A. Gani, Augmenting Mobile Devices via Lightweight Mobile Clouds, IEEE Transactions on Parallel and Distributed Computing, under review, 2014.

20. M. Shiraz, S. Abolfazli, Z.Sanaei, A.Gani, A study on virtual machine deployment for application outsourcing in mobile cloud computing, **The Journal of Supercomputing** (Q2), Vol 62, No. 3, 2012, DOI 10.1007/s11227-012-0846-y, (ISI/SCOPUS Cited Publication).

7.5 Limitation and Future Work

One of the limitations of this study is that we have not used mobile devices like smartphones as service providers. Considering rapidly growing computing capabilities of mobile devices and their multiplicity, it is feasible to exploit computational capabilities of nearby mobile devices, especially smartphones to perform computation on behalf of

nearby mobile clients. Mobility management also is not considered in this work. However, in provisioning of the system, we intentionally used mobile network operators to facilitate deployment of mobility management strategies via the central mobile network operator. Also, as a shortcoming of this research we can point out integration of alternative/secondary communication technologies such as cellular network.

In our future works, we will consider incorporating mobile devices as fine-grained computing devices and will deploy appropriate mobility management strategy. It is also possible to consider improvement in security and privacy features of the proposed framework to make it trustable so that mobile users can leverage the benefits of the proposed framework.

REFERENCES

Abolfazli, S., Sanaei, Z., Alizadeh, M., Gani, A., & Xia, F. (2014). An experimental analysis on cloud-based mobile augmentation in mobile cloud computing. *IEEE Transactions on Consumer Electronics*, *60*(1), 146-154.

Abolfazli, S., Sanaei, Z., Gani, A., & Buyya, R. (2014). Cloud-based augmentation for mobile devices: Motivation, taxonomies, and open challenges. *IEEE Communication Surveys & Tutorials*, *16*(1), 337-368.

Abolfazli, S., Sanaei, Z., Shiraz, M., & Gani, A. (2012, August). MOMCC: Market-oriented architecture for Mobile Cloud Computing based on Service Oriented Architecture. In *Proc. ieee 1st international conference on communications in china workshops (iccc)* (p. 8-13). Beijing, China.

Akyildiz, I. F., Jiang, X., & Mohanty, S. (2004). A survey of mobility management in next-generation all-ip-based wireless systems. *IEEE Wireless Communications*, *11*(4), 16-28.

Albanesius, C. (2011, February). "*smartphone shipments surpass PC shipments for first time. what's next?"*. Available from http://www.pcmag.com/article2/0,2817,2379665,00.asp

A.Manjunatha, A.Ranabahu, A.Sheth, & K.Thirunarayan. (2010, December). Power of clouds in your pocket: An efficient approach for cloud mobile hybrid application development. In *Proc. IEEE 2nd international conference on cloud computing technology and science (CloudCom'10)* (p. 496-503). Dayton, OH, USA.

Amazon elastic compute cloud (ec2). (n.d.). Available from http://www.amazon.com/ec2/

Amazon web services,amazon simple storage service (amazon s3). (n.d.). Available from http://aws.amazon.com/s3

Amazon Web Services: Mobile developer center. (n.d.). Available from http://aws.amazon.com/mobile

Andriescu, E., Speicys Cardoso, R., & Issarny, V. (2011, December). AmbiStream: a middleware for multimedia streaming on heterogeneous mobile devices. In *Proc. ACM/IFIP/USENIX 12th international conference on middleware (Middleware'11)* (pp. 249–268). Lisboa, Portugal.

Android 3.2. (n.d.). Available from http://developer.android.com/sdk/android-3.2.html#api

Android API levels. (n.d.). Available from http://developer.android.com/guide/appendix/api-levels.html

A.Ranabahu, & A.Sheth. (2010, November). Semantics centric solutions for application and data portability in cloud computing. In *Proc. IEEE 2nd international conference on cloud computing technology and science (CloudCom'10)* (p. 234 - 241). Dayton, OH, USA.

Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., et al. (2010). A view of cloud computing. *Communications of the ACM*, *53*(4), 50–58.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R. H., Konwinski, A., et al. (2009). *Above the clouds: A berkeley view of cloud computing* (Tech. Rep. No. UCB/EECS-2009-28).

"*The ARM cortex-A9 processors"*. (2009). White Paper. ARM. Available from http://arm.com/files/pdf/ARMCortexA-9Processors.pdf

"*ARM discloses technical details of the next version of the ARM architecture"*. (2011, Oct.). SANTA CLARA, CA, USA. Available from http://goo.gl/98kBc

Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., et al. (2003). Xen and the art of virtualization. *ACM SIGOPS Operating Systems Review*, *37*(5), 164-177.

B.Chun, S.Ihm, P.Maniatis, & M.Naik. (2010). CloneCloud boosting mobile device applications through cloud clone execution.

Behrendt, M., Glasner, B., Kopp, P., Dieckmann, R., Breiter, G., Pappe, S., et al. (2011, February). *Cloud computing reference architecture v2.0* (Vol. 1; Tech. Rep.). IBM. Available from https://s3-sa-east-1.amazonaws.com/bucketmanoelveras/manoel_veras_PROFESSIONAL/CLOUDCOMPUTING/Artigos/Artigos_IBM/arq_ibm.pdf

Blair, G., & Grace, P. (2012). Emergent middleware: Tackling the interoperability problem. *IEEE Internet Computing*, *16*(1), 78 -82.

Blair, G., Paolucci, M., Grace, P., & Georgantas, N. (2011). Interoperability in complex distributed systems. In M. Bernardo & V. Issarny (Eds.), *Formal methods for eternal networked software systems* (Vol. 6659, p. 1-26). Springer Berlin Heidelberg.

Bohn, R., Messina, J., Liu, F., Tong, J., & Mao, J. (2011, July). NIST cloud computing: Reference architecture. In *Proc. IEEE 7th world congress on services (SERVICES'11)* (pp. 594–596). Washington DC, USA.

Bromberg, Y., Grace, P., Réveillère, L., & Blair, G. (2011, December). Bridging the interoperability gap: Overcoming combined application and middleware heterogeneity. In *Proc. ACM/IFIP/USENIX 12th international conference on middleware (Middleware'11)* (pp. 390–409). Lisboa, Portugal.

Buyya, R., Chee Shin, Y., & Venugopal, S. (n.d.). Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. In *10th ieee international conference on high performance computing and communications* (p. 5-13).

Buyya, R., Yeo, C., Venugopal, S., Broberg, J., & Brandic, I. (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.

Cachin, C., & Schunter, M. (2011). A cloud you can trust. *IEEE Spectrum*, *48*(12), 28 -51.

Chandrakant, N., Bijil, A., Shenoy, P., Venugopal, K., & Patnaik, L. (2012, December). Middleware service oriented rescue and crime information system (RCIS) using heterogeneous fixed nodes in WSNs. In *Proc. international conference on advanced computing, networking and security (ADCONS'11)* (pp. 389–398). Surathkal, India.

Chen, Y., Paxson, V., & Katz, R. H. (2010, Jan). *WhatâĂŹs new about cloud computing security?* (Tech. Rep. No. UCB/EECS-2010-5). EECS Department, University of California, Berkeley. Available from http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-5.html

Chen, Y., Xie, X., Ma, W., & Zhang, H. (2005). Adapting web pages for small-screen devices. *IEEE Internet Computing*, *9*(1), 50-56.

Chun, B., Ihm, S., Maniatis, P., Naik, M., & Patti, A. (2011, April). CloneCloud: Elastic execution between mobile device and cloud. In *Proc. ACM the european professional society on computer systems (EuroSys'11)* (p. 301-314). Salzburg, Austria.

Chun, B.-G., & Maniatis, P. (2009). Augmented smartphone applications through clone cloud execution. In *Proc. 12th conference on hot topics in operating systems (HotOS'09)* (p. 8).

Cuervo, E., Balasubramanian, A., Cho, D., Wolman, A., Saroiu, S., Chandra, R., et al. (2010, June). MAUI: making smartphones last longer with code offload. In *Proc. ACM 8th annual international conference on mobile systems, applications and services (MobiSys'10)* (pp. 49–62). San Francisco, CA, USA.

D.Durkee. (2010). Why cloud computing will never be free. *Communication of the ACM*, *53*(5), 62-69.

DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., et al. (2007, October). Dynamo: amazon's highly available key-value store. In *Proc. ACM 21st symposium on operating systems principles (SOSP'07)* (pp. 205–220). Stevenson, Washington, USA.

Demestichas, P., Stavroulaki, V., Boscovic, D., Lee, A., & Strassner, J. (2006). m@ANGEL: autonomic management platform for seamless cognitive connectivity to the mobile internet. *IEEE Communications Magazine*, *44*(6), 118-127.

Dropbox. (n.d.). Available from https://www.dropbox.com/

Eucalyptus systems, inc.,eucalyptus cloud computing software. (2011). Available from http://www.eucalyptus.com/

Find the best: Compare cloud computing providers. (n.d.). Available from http://cloud-computing.findthebest.com

Flinn, J., & Satyanarayanan, M. (1999, December). Energy-aware adaptation for mobile applications. In *Proc. ACM 17th symposium on operating systems principles (SOSP'99)* (p. 48-63). SC, USA.

"The future of hotspots: Making Wi-Fi as secure and easy to use as cellular". (2012). Available from http://www.cisco.com/en/US/solutions/collateral/ns341/ns524/ns673/white%5Fpaper%5Fc11-649337.html

Gartner. (2013, Nov). *Gartner says smartphone sales accounted for 55 percent of overall mobile phone sales in third quarter of 2013.* Available from http://www.gartner.com/newsroom/id/2623415

"Global markets for smartphones and PDAs". (2009). Available from http://www.bccresearch.com/pressroom/report/code/IFT068A

Google, inc.,google app engine. (n.d.). Available from http://code.google.com/appengine/

Gupta, P., Zeldovich, N., & Madden, S. (2011, December). A trigger-based middleware cache for ORMs. In *Proc. ACM/IFIP/USENIX 12th international conference on middleware (Middleware'11)* (pp. 329–349). Lisboa, Portugal.

Hogan, M., Liu, F., Sokol, A., & Tong, J. (2011, July). *NIST cloud computing standards roadmap-version 1.0* (Tech. Rep.).

Huang, J., Xu, Q., Tiwana, B., Mao, Z. M., Zhang, M., & Bahl, P. (2010, June). Anatomizing application performance differences on smartphones. In *Proc. ACM 8th international conference on mobile systems, applications, and services (MobiSys'10)* (pp. 165–178). San Francisco, CA, USA.

Huerta-Canepa, G., & Lee, D. (2010, June). A virtual cloud computing provider for mobile devices. In *Proc. ACM 1st workshop on mobile cloud computing & services: Social networks and beyond (MCS'10)* (p. 1-5). San Francisco, California. Proc. ACM 1st Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond.

Huhns, M., & Singh, M. (2005). Service-oriented computing: Key concepts and principles. *IEEE Internet Computing*, *9*(1), 75–81.

INCITS. (n.d.). *Portability definition.* Available from www.incits.org/ansdit/p2.htm

Javanmardi, S., Shojafar, M., Amendola, D., Cordeschi, N., Liu, H., & Abraham, A. (2014). Hybrid job scheduling algorithm for cloud computing environment. In *Proceedings of the fifth international conference on innovations in bio-inspired computing and applications ibica 2014* (pp. 43–52).

Jones, E., Alexander, J., Andreou, A., Irani, P., & Subramanian, S. (2010, April). Gestext: accelerometer-

based gestural text-entry systems. In *Proc. ACM 28th international conference of human factors in computing systems (CHI'10)* (p. 2173-2182). Atlanta, GA, USA.

Kemp, R., Palmer, N., Kielmann, T., & Bal, H. (2010, October). Cuckoo: a computation offloading framework for smartphones. In *Proc. 2nd international conference on mobile computing, applications, and services, (mobicase '10).* Santa Clara, CA, USA.

Kumar, K., Liu, J., Lu, Y., & Bhargava, B. (2012). A survey of computation offloading for mobile systems. *Mobile Networks and Applications*, 1-12.

Kumar, K., & Lu, Y. H. (2010). Cloud computing for mobile users: Can offloading computation save energy? *IEEE Computer*, *43*(4), 51-56.

Lagar-Cavilla, H., Tolia, N., Lara, E. de, Satyanarayanan, M., & Hallaron, D. (2007). Interactive resource-intensive applications made easy. In *Middleware* (Vol. 4834, p. 143-163). Springer Berlin / Heidelberg.

Li, P., & Fang, Y. (2012). On the throughput capacity of heterogeneous wireless networks. *IEEE Transactions on Mobile Computing*, *11*, 2073- 2086.

Loutas, N., Peristeras, V., Bouras, T., Kamateri, E., Zeginis, D., & Tarabanis, K. (2010, December). Towards a reference architecture for semantically interoperable clouds. In *Proc. IEEE 2nd international conference on cloud computing technology and science (CloudCom'10)* (pp. 143–150). Dayton, OH, USA.

Lu, Y., Li, S. P., & Shen, H. F. (2011). Virtualized screen: A third element for cloud-mobile convergence. *IEEE Multimedia*, *18*(2), 4-11.

Madhavapeddy, A., Mortier, R., Crowcroft, J., & Hand, S. (2010, April). Multiscale not multicore: Efficient heterogeneous cloud computing. In *Proc. ACM-BCS international conference on visions of computer science (ACM-BCS '10)* (p. 1-12). Edinburgh,UK.

March, V., Gu, Y., Leonardi, E., Goh, G., Kirchberg, M., & Lee, B. S. (2011). μcloud: Towards a new paradigm of rich mobile applications. *Procedia Computer Science*, *5*, 618–624.

Marinelli, E. E. (2009). *Hyrax: Cloud computing on mobile devices using MapReduce*. Unpublished master's thesis, School of Computer Science, Carnegie Mellon University.

Mell, P., & Grance, T. (2011). *The NIST definition of cloud computing.* Available from `http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf`

M. Guevara, B. L., B. Lubin. (2013, February). Navigating heterogeneous processors with market mechanisms. In *Proc. IEEE 19th international symposium on high-performance computer architecture (hpca).* Shenzhen, China,.

Miettinen, A., & Nurminen, J. (2010, June). Energy efficiency of mobile clients in cloud computing. In *Proc. 2nd usenix conference on hot topics in cloud computing* (p. 4-11). Boston, MA.

Milojicic, D., & Wolski, R. (2011). Eucalyptus: Delivering a private cloud. *IEEE Computer*, *44*(4), 102-104.

Min, A., Zhang, X., Choi, J., & Shin, K. (2012). Exploiting spectrum heterogeneity in dynamic spectrum market. *IEEE Transactions on Mobile Computing*, *11*, 2020–2032.

Motorola. (2008). *"2G and 3G cellular networks: Their impact on todayâĂŹs enterprise mobility solutions and future mobility strategies".* White Paper. Available from `http://www.motorola.com/web/Business/Products/Mobile%20Computers/Handheld%20Computers/%5FDocuments/staticfile/3G%5FWhitepaper%5F0608.pdf`

Nasser, N., Hasswa, A., & Hassanein, H. (2006). Handoffs in fourth generation heterogeneous networks. *IEEE Communications Magazine*, *44*(10), 96-103.

"Open virtualization format specification (OVF)". (2009). Available from http://www.dmtf.org/sites/default/files/standards/documents/DSP0243_1.0.0.pdf

Ou, Z., Zhuang, H., Nurminen, J. K., Ylä-Jääski, A., & Hui, P. (2012, June). Exploiting Hardware Heterogeneity within the same Instance Type of Amazon EC2. In *Proc. 4th usenix conference on hot topics in cloud computing (HotCloud'12)* (p. 4). Boston, MA.

Owens, K. (2009). *"Securing virtual compute infrastructure in the cloud"*. Whitepaper. Available from http://www.savvis.com/en-us/info%5Fcenter/documents/hos-whitepaper-securingvirutalcomputeinfrastructureinthecloud.pdf

Perrucci, G. P., Fitzek, F. H. P., & Widmer, J. (2011, May). Survey on energy consumption entities on the smartphone platform. In *Proc. IEEE 73rd vehicular technology conference (vtc spring)* (p. 1-6). Budapest, Hungary.

Pickard, W. F., & Abbott, D. (2012). Addressing the intermittency challenge: Massive energy storage in a sustainable future [scanning the issue]. *Proceedings of the IEEE*, *100*(2), 317 -321.

Rahimi, M. R., Venkatasubramanian, N., Mehrotra, S., & Vasilakos, A. V. (2012, November). MAPCloud: Mobile applications on an elastic and scalable 2-tier cloud architecture. In *Proc. IEEE/ACM 5th international conference on utility and cloud computing (UCC'12)* (pp. 83–90). Chicago, Illinois, USA.

Rajesh Krishna, B. (2004). Powerful change part 2: reducing the power demands of mobile devices. *IEEE Pervasive Computing*, *3*(2), 71-73.

R.Avro, S. (2009). *"Wireless electricity is real and can change the world"* (No. 10-12-2011). Consumer Energy Report. Available from www.consumerenergyreport.com/2009/01/15/wireless-electricity-is-real-and-can-change-the-world/

Rellermeyer, J., & Küpfer, R. (2011, December). Co-managing software and hardware modules through the juggle middleware. In *Proc. ACM/IFIP/USENIX 12th international conference on middleware (Middleware'11)* (pp. 431–450). Lisboa, Portugal.

Robinson, S. (2009). *"Cellphone energy gap: Desperately seeking solutions"* (No. 30-11-2011). Strategy Analytics. Available from http://www.strategyanalytics.com/default.aspx?mod=reportabstractviewer&a0=4645

Rochwerger, B., Breitgand, D., Epstein, A., Hadas, D., Loy, I., Nagin, K., et al. (2011). Reservoir-when one cloud is not enough. *IEEE Computer*, *44*(3), 44-51.

Rosenberg, A., & Chiang, R. (2010, April). Toward understanding heterogeneity in computing. In *Proc. IEEE international symposium on parallel & distributed processing (IPDPS'10)* (pp. 1–10). Fort Collins, CO, USA.

Sakr, S., Liu, A., Batista, D., & Alomari, M. (2011). A survey of large scale data management approaches in cloud environments. *IEEE Communications Surveys & Tutorials*, *13*(3), 311-336.

Sanaei, Z., Abolfazli, S., Gani, A., & Buyya, R. (2014). Heterogeneity in Mobile Cloud Computing: Taxonomy and Open Challenges. *IEEE Communication Surveys & Tutorials*, *16*(1), 369-392.

Sanaei, Z., Abolfazli, S., Gani, A., & Shiraz, M. (2012, August). SAMI: Service-based arbitrated multi-tier infrastructure for mobile cloud computing. In *Proc. IEEE 1st international conference on communications in china workshops (iccc)* (p. 14-19). Beijing, China.

Satyanarayanan, M. (2001). Pervasive computing: vision and challenges. *IEEE Personal Communications*,

8(4), 10-17.

Satyanarayanan, M. (2005). Avoiding dead batteries. *IEEE Pervasive Computing*, *4*(1), 2-3.

Satyanarayanan, M., Bahl, P., Caceres, R., & Davies, N. (2009). The case for VM-Based cloudlets in mobile computing. *IEEE Pervasive Computing*, *8*(4), 14–23.

Sharifi, M., Kafaie, S., & Kashefi, O. (2011). A survey and taxonomy of cyber foraging of mobile devices. *IEEE Communications Surveys & Tutorials*, *PP*(99), 1-12.

Sheth, A., & Ranabahu, A. (2010). Semantic modeling for cloud computing, part 1. *IEEE Internet Computing*, *14*(3), 81-83.

Shiraz, M., & Gani, A. (2012). Mobile cloud computing: Critical analysis of application deployment in virtual machines. In *Proc. int'l conf. information and computer networks (ICICN'12)* (Vol. 27).

Shiraz, M., Gani, A., Hafeez, R., & Buyya, R. (2012, November). A review on distributed application processing frameworks in smart mobile devices for mobile cloud computing. *IEEE Communication Surveys & Tutorials*, *Accepted*.

Smartphones will surpass pc shipments in two years. (2010). Available from http://techcrunch.com/2010/11/16/meeker-smartphones-pcs/

Soyata, T., Muraleedharan, R., Funai, C., Kwon, M., & Heinzelman, W. (2012). Cloud-Vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture. In *Proc. ieee iscc '12* (pp. 59–66). Cappadocia, Turkey.

Stone, J., Gohara, D., & Shi, G. (2010). OpenCL: A parallel programming standard for heterogeneous computing systems. *Computing in science & engineering*, *12*(3), 66.

Takabi, H., Joshi, J., & Ahn, G. (2010). Security and privacy challenges in cloud computing environments. *IEEE Security & Privacy*, *8*(6), 24–31.

Tarkoma, S., & Lagerspetz, E. (2011). Arching over the mobile computing chasm: Platforms and runtimes. *IEEE Computer*, *44*(4), 22-28.

Tolia, N., Andersen, D. G., & Satyanarayanan, M. (2006). Quantifying interactive user experience on thin clients. *Computer*, *39*(3), 46-52.

Valberg, S., & Christensen, P. T. (2009). *Improving usability of mobile devices by means of accelerometers*. Master thesis.

Vecchiola, C., Chu, X., & Buyya, R. (2009a). Aneka: A software platform for .net-based cloud computing. In G. J. E. W. Gentzsch L. Grandinetti (Ed.), *High speed and large scale scientific computing* (p. 267-295). IOS Press, Amsterdam, Netherlands.

Vecchiola, C., Chu, X., & Buyya, R. (2009b). Aneka: A software platform for .NET-based cloud computing. *in High Speed and Large Scale Scientific Computing*, *W. Gentzsch, L. Grandinetti, G. Joubert (Eds.), ISBN: 978-1-60750-073-5*, 267-295.

VMware. (n.d.). *Intro to virtualization: Get started with ESXi.* Available from http://www.vmware.com/files/pdf/ESX%5FServer%5F3i%5Fpresentation.pdf

Vmware view 5 with pcoip network optimization guide. (n.d.). Available from http://www.vmware.com/files/pdf/view/VMware-View-5-PCoIP-Network-Optimization-Guide.pdf

Walraven, S., Truyen, E., & Joosen, W. (2011, December). A middleware layer for flexible and cost-efficient multi-tenant applications. In *Proc. ACM/IFIP/USENIX 12th international conference on middleware (Middleware'11)* (pp. 370–389). Lisboa, Portugal.

Whaiduzzaman, M., Sookhak, M., Gani, A., & Buyya, R. (2014). A survey on vehicular cloud computing. *Journal of Network and Computer Applications*, *40*, 325-344.

Windows Azure Developer Center. (2012). Available from `http://www.windowsazure.com/en-us/develop/overview`

Wolfgang, P. (1994). *Design patterns for object-oriented software development*. Addison-Wesley.

Xia, F., Ding, F., Li, J., Kong, X., Yang, L. T., & Ma, J. (2014). Phone2Cloud: Exploiting computation offloading for energy saving on smartphones in mobile cloud computing. *Information Systems Frontiers*, *16*(1), 95–111.

Yan, X., Ahmet Sekercioglu, Y., & Narayanan, S. (2010). A survey of vertical handover decision algorithms in fourth generation heterogeneous wireless networks. *Computer Networks*, *54*(11), 1848–1863.

Yin, W., Luo, J., & Chen, C. (2011). Event-based semantic image adaptation for user-centric mobile display devices. *IEEE Transactions on Multimedia*, *13*(3), 432-442.

Zhang, L.-J., & Zhou, Q. (2009, July). CCOA: Cloud computing open architecture. In *Proc. IEEE 7th international conference on web services (ICWS'09)* (pp. 607–616). Los Angeles, CA, USA.

Zhang, X. W., Kunjithapatham, A., Jeong, S., & Gibbs, S. (2011). Towards an elastic application model for augmenting the computing capabilities of mobile devices with cloud computing. *Mobile Networks & Applications*, *16*(3), 270-284.