THE DESIGN OF MOBILE-SOURCING FRAMEWORK FOR THE ENERGY EFFICIENT COMPUTATION WITH MULTI-OBJECTIVE ANT COLONY OPTIMIZATION

MR. KATCHAGUY AREEKIJSEREE

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF MASTER OF ENGINEERING (COMPUTER ENGINEERING)
FACULTY OF ENGINEERING
KING MONGKUT'S UNIVERSITY OF TECHNOLOGY THONBURI
2013

The Design of Mobile-sourcing Framework for the Energy Efficient Computation with Multi-objective Ant Colony Optimization

Mr.Katchaguy Areekijseree B.Eng. (Computer Engineering)

A Thesis Submited in Partial Fulfillment of the Requirement for
the Degree of Master of (Computer Engineering)
Faculty of Engineering
King Mongkut's University of Technology Thonburi
2013

Thesis Committee

..... (Asst. Prof. Marong Phadoongsidhi, Ph.D.)	Chairman of Thesis Committee
..... (Assoc. Prof. Tiranee Achalakul, Ph.D.)	Member and Thesis Advisor
..... (Assoc. Prof. Naruemon Wattanapongsakorn, Ph.D.)	Member
..... (Prof. Simon Chong Wee See, Ph. D.)	Member
..... (Sivadon Chaisiri, Ph.D.)	Member

Copyright reserved

Thesis Title	The Design of Mobile-sourcing Framework for the Energy Efficient Computation with Multi-objective Ant Colony Optimization
Thesis Credits	12 credits
Candidate	Mr. Katchaguy Areekijseree
Thesis Advisor	Assoc. Prof. Dr. Tiranee Achalakul
Program	Master of Engineering
Field of Study	Computer Engineering
Department	Computer Engineering
Faculty	Engineering
Academic Year	2013

Abstract

Volunteered computing is one of the most popular distributed computing concepts in the recent years. The basic idea is to allow computer owners to donate computing power and storage to scientific applications. Currently, the advancement of the mobile technology has made possible the utilization of mobile devices for serious computations. In this research, we attempted to exploit the power of mobile devices. We believed that by outsourcing the computation, the energy consumption at data centers can be reduced while keeping the execution time within a reasonable deadline. We then designed a mobile-sourcing framework to utilize devices' capability and capitalize on the CPU cycles contributed from mobile users under the concept of volunteer computing. Due to the availability of volunteers, hardware limitation, and wireless network connectivity, task scheduling becomes another crucial process. Thus, this research also emphasizes the design and the implementation of a scheduling algorithm for the framework. We adopted Multi-Objective Ant Colony Optimization (MOACO) to find a good balance between energy consumption and the runtime. In order to validate the effectiveness of the scheduler, we performed a set of experiments. The results showed that the accuracy of the prediction models within the scheduler was approximately above 95% high for both energy and time. In other words, when the created schedules were executed, the actual observed time and the estimated time were similar. In addition, we found that the auto-generated schedules gained about 25% quality improvement with MOACO in comparison to the round-robin technique. Lastly, the scalability of the designed platform was investigated with different sizes of scientific workflows and different numbers of volunteered devices. From the results, it can be concluded that the quality of the created schedules was sufficiently good regardless of the workflow sizes and the number of devices.

Keywords : Ant Colony Optimization / Mobile-sourcing / Optimization Algorithm / Scheduling Algorithm / Workflow Scheduling

หัวข้อวิทยานิพนธ์	การออกแบบเฟรมเวิร์คสำหรับการลดพลังงานในการประมวลผลด้วยอุปกรณ์พกพา
หน่วยกิต	12
ผู้เขียน	นายกรัชกาย อารีกิจเสรี
อาจารย์ที่ปรึกษา	รศ. ดร. ธีรณี อจลากุล
หลักสูตร	วิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
ภาควิชา	วิศวกรรมคอมพิวเตอร์
คณะ	คณะวิศวกรรมศาสตร์
ปีการศึกษา	2556

บทคัดย่อ

ในปัจจุบันความก้าวหน้าทางเทคโนโลยีบนอุปกรณ์พกพาทำให้ผู้ใช้ได้รับประโยชน์มากมาย อุปกรณ์พกพาเหล่านี้มีประสิทธิภาพเทียบเท่ากับคอมพิวเตอร์ส่วนบุคคล ซึ่งทำให้เกิดแนวคิดที่จะนำทรัพยากรของอุปกรณ์เหล่านี้มาใช้ในการประมวลผลให้เกิดประโยชน์ การประมวลผลแบบอาสาสมัครเป็นรูปแบบหนึ่งของการประมวลผลแบบกระจายที่กำลังได้รับความนิยม หลักการของการประมวลผลแบบอาสาสมัครคือการกระจายงานบางส่วนไปประมวลผลบนอุปกรณ์ของอาสาสมัคร ในงานวิจัยนี้เรามุ่งเน้นการลดพลังงานที่ใช้ในการประมวลผลของศูนย์ข้อมูล (ดาต้าเซนเตอร์) อีกทั้งควบคุมเวลาให้อยู่ในช่วงที่เหมาะสม ทีมนักวิจัยได้ออกแบบโมบายซอร์สซิงเฟรมเวิร์คเพื่อที่จะใช้ประโยชน์จากการประมวลผลบนอุปกรณ์แบบไร้สายให้มากที่สุด แต่อย่างไรก็ตามการจัดสรรงานเพื่อกระจายงานไปประมวลผลบนอุปกรณ์ไร้สายนั้นเป็นขั้นตอนที่สำคัญมาก เนื่องจากมีหลายปัจจัยที่ทำให้ใช้พลังงานหรือเวลามากขึ้น ไม่ว่าจะเป็น จำนวนเครื่องของอาสาสมัคร ข้อจำกัดของฮาร์ดแวร์ที่แตกต่างกัน หรือ การเชื่อมต่อระหว่างเซิฟเวอร์และอุปกรณ์ ดังนั้นในการวิจัยนี้จึงเล็งเห็นความสำคัญของการออกแบบและพัฒนาอัลกอริทึมสำหรับการกระจายงานสำหรับโมบายซอร์สซิงเฟรมเวิร์ค ทางทีมวิจัยได้เลือกใช้มัลติออพเจ็กทีปแอนท์โคโลนีออพติไมซ์เซชันอัลกอริทึม เพื่อหาโซลูชันในการกระจายงานที่จะลดพลังงานและเวลาที่ใช้บนดาต้าเซนเตอร์ ผลการทดลองแสดงให้เห็นว่าแบบจำลองมีความแม่นยำถึง 95% และยังให้ผลลัพธ์ที่อยู่ในเกณฑ์ที่น่าพอใจเมื่อขนาดของเวิร์คโฟล์ (workflow) และจำนวนของอาสาสมัคร มีจำนวนที่แตกต่างกัน ทางทีมวิจัยทำการตรวจสอบความมีประสิทธิภาพและความเป็นไปได้ของเฟรมเวิร์คโดยทดลองเปรียบเทียบกับราวร็อบบิน ซึ่งแสดงให้เห็นว่าเทคนิคที่เลือกใช้สามารถประหยัดพลังงานของดาต้าเซนเตอร์ได้ดีกว่า 24.28 % จากการทดลองต่างๆนั้นจึงสรุปได้ว่าเฟรมเวิร์คนี้มีความเป็นไปได้ที่จะนำอุปกรณ์พกพามาช่วยในการประมวลผลเพื่อลดการใช้พลังงานภายในช่วงเวลาที่กำหนดได้

คำสำคัญ : การกระจายงานสำหรับเวิร์คโฟลว์ / โมบายซอร์สซิง / อัลกอริทึมสำหรับการกระจายงาน / ออปติไมซ์เซชันอัลกอริทึม / แอนท์โคโลนีออปติไมซ์เซชัน/

ACKNOWLEDGEMENTS

I would like to express my deep gratitude to Assoc. Prof. Tiranee Achalakul, my advisor, for her patient guidance, enthusiastic encouragement and useful critiques on this research work. I would like to thank you for encouraging me to do this research and for allowing me to grow as a research scientist. Your advice on both research as well as on my career have been priceless.

I would also like to extend my thanks to the Miss Sikana Tanupabrungsun and my parents for their support and encouragement throughout my study. Lastly, I would especially like to thanks all the CASTlab members for all the great moments during my study.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

Mobile devices usually refer to handheld or pocket-size computing devices such as smartphones, phablets and tablets. At present, such devices have become a part of our daily life. People use these devices for different purposes, for example, social networking, messaging, documentation, browsing and photo editing. Statistics from the International Telecommunication Union revealed that the number of active mobile-broadband subscriptions in the world had increased from 268 million in 2007 to 2,096 million in 2013 [1]. In other words, the annual growth rate of mobile usage is approximately 130%.

The advancement of mobile technology has brought about tremendous benefits to the mobile users. The functionality of device is rapidly developed and integrated with PC-like capabilities. Not only the functionality but also performance has been greatly improved. The computing ability of the mobile devices is now comparable to those of regular desktop computers. Thus, it is now possible to utilize mobile devices for actual computation, such as image processing, 3D rendering, 3D game, and mathematical modeling. Moreover, mobile users rarely turn off their devices but rather leave them in idle state for several hours each day, for instance, an overnight battery charging may leave the device in an idle state for up to 7 hours. To put them to good use, mobile users can contribute the computing power to be public resources for Sciences.

As the capability of mobile device has been greatly improved, people are now making use of them on a variety of applications, including scientific ones. However, Sciences have not fully utilized the devices as they should. One of the most popular science-related applications is for data collecting. Even though the application supports the scientists to work more easily and efficiently, still, the computing capability of mobile device is far beyond that. One possibility is to support the scientific workflow execution.

In general, a scientific workflow is a series of computation nodes and concurrent data processing. The order represents the interdependencies of nodes. It is usually presented as Directed Acyclic Graph (DAG) where nodes represent a series of atomic tasks which can be either small units of computation or data manipulation. The atomic task includes data generation, transformation, aggregation, analysis and visualization. In general, the task consumes the input data transferred from the ancestor nodes, processes and produces the output for the descendent nodes. The edge between nodes in a workflow represents the dependency between those atomic tasks.

In general, a great deal of execution times and resources are required to serve the needs of an application. For instance, a workflow may require up to a month to complete an execution. However, for an application like weather or epidemic forecast, the results are frequently in need and it is almost impossible to wait for the long execution to complete.

The advancement in the high performance computing (HPC) platform has been continuously contributing to Sciences. For instance, it plays a crucial role in data analytic, modeling, and simulation tasks. Moreover, scientific organizations usually utilize a cluster of computers or cloud computing to optimize the execution time of

data- and compute-intensive applications. However, HPC usually consumes a great deal of energy. For instance, power consumption per rack can be as high as 30 kW. From the study [2], it shows that energy consumption by the data centers worldwide is increasing every year. Thus, it would be beneficial for scientific organizations to have the community lightening the energy loaded from the data center.

In this research, we attempted to reduce the energy consumption at the data center and restrict the execution time in the timeframe. We designed the mobile sourcing framework to utilize devices' capability and capitalize on the CPU cycles contributed from mobile users under the concept of volunteer computing – a distributed computing platform constructing from the resource donation of the users around the world. The more resources being donated, less amount of energy at the data center is consumed. The basic idea of our mobile-sourcing framework is to allow the atomic tasks of the scientific problems to be packed and distributed to mobile devices and use mobile power to solve the problem in conjunction with the computers at the data center.

For the volunteers, they can donate their computing powers any time under the join-and-leave concept. Specifically, the computing unit does not always available for the workflow execution but the platform has to request the mobile users to donate the resources. In this work, we define a few assumptions to simplify the framework design. Firstly, we assume that the mobile user only donates the resource when the mobile is being charged overnight. Also, the mobile is not running any application which means that it is left in the idle state. Thus, the processing power of the donated mobile can be fully utilized. Secondly, as the mobile is plugged, it can be considered as in the stationary phase and the network connectivity is stable.

During the process of framework design, several challenges arose. First is the variation of mobile device from different mobile users. Different models of mobile devices usually have different profiles; for example, architecture, computing power, memory and battery capacity. Thus, these profiles must be taken into consideration for the scheduling. Secondly, to distribute the tasks to the mobile devices, the executable and input files must be transferred to the assigned device. The transfer time heavily relies on the speed of network connection and file size. Thus, the transfer time must be precisely predicted as it must be considered by the scheduler. Third, we've learnt that scattering too many tasks sometimes takes too long an execution due to the overhead of task transfer. Thus, the execution schedule should be carefully designed.

With the aforementioned challenges, tasks scheduling to the mobile devices is a crucial process due to the availability of volunteers, hardware limitation, and wireless network connection. This research did not only intend to design the mobile-sourcing framework but also put an emphasis on the design and implementation of a scheduling algorithm for the platform.

Specifically, we adopted the Multi-Objective Ant Colony Optimization (MOACO) algorithm to obtain the optimal execution schedule. The first objective was to maximize the energy saving by distributing tasks to the network of volunteered mobiles. In other words, we aimed to minimize the power consumption at the data center. Second, the algorithm attempted to optimize the total computing time by utilizing mobile sources appropriately. For the multiple objectives schema, we applied the weighted sum method which is the simplest and best known technique. In summary, we

designed the framework of mobile sourcing and put an emphasis on the scheduling algorithm.

We hope to contribute in the optimization community with the design of the scheduling algorithm. It could also contribute to the high performance community as we are proposing a novel concept for the optimal workflow execution with the mobile-sourcing framework.

The organization of this thesis is as follows: Chapter 2 presents a literature survey and related research studies together with the background study on the optimization technique summarized at the end of chapter. The proposed framework and evaluation methods are explained in Chapter 3. Then, the experimental setups and results are presented in Chapter 4. Chapter 5 is the conclusion of this research.

CHAPTER 2 LITERATURE SURVEY

In this work, we attempted to design the mobile sourcing platform to support the volunteer computing concept. In order to utilize the computing power from mobile sources, the method must be carefully designed. We were particularly interested in compute-intensive scientific applications, where computation can be divided into a large amount of independent tasks. These tasks can then be distributed over a large network of mobile sources volunteered by their owners. To assign tasks appropriately, we focused on the design and implementation of the scheduler which is based on the optimization algorithm

This chapter presents the related research studies. In the first section, we will explain the previous works and their challenges related to integration of mobile and grid environment. Then, we will study the volunteer computing platforms which have been quite popular in the past decades. Also, as we put an emphasis on the optimization algorithms for task scheduling, the novel techniques are surveyed and summarized. In addition, we present the background study on the optimization technique at the end of this chapter.

2.1 Literature Review

2.1.1 Integration of Mobile Devices and Grid Environment

As the use of mobile devices grow exponentially, recent literature has been focusing on an integration of the mobile devices to the grid computing infrastructure. The first research was proposed by Litke et al. [3]. They discussed the efficient adoption of mobile devices in the grid computing environment that inspired them to design the mobile grid computing platform. This platform was a combination of grid computing infrastructure and mobile devices. Grid computing could be considered as a distributed, high performance computing and data handling infrastructure while mobile device was a portable and wireless processing unit. In particular, they were interested in the computing capabilities of grid computing and attempted to make a good use of the device's mobility.

Later on, Phan et al. [4] extended the concept of mobile grid computing by adopting the platform to solve the scientific problems. The underlying concept was similar to the previous; the proposed platform utilized ordinary mobile devices by integrating them to the grid computing environment. The proposed platform consisted of a proxy component residing in the grid computing unit, namely, *Interlocutor*. Interlocutor was designed to scatter the tasks to the mobile devices, namely, *Minions*. After being assigned, Minion would execute the task per requested. Once completed, Interlocutor retrieved the results from Minions and submitted to the grid. The experimental results showed that integrating mobile devices with grid environment had, however, degraded the overall performance of the grid computing infrastructure. They discussed that the degradation was caused by the low performance of the mobile devices back in that time. Specifically, it was reported that the available devices had low computing power, low memory, unstable wireless connectivity and high battery consumption. However, it was expected that the platform could be significantly improved by the advancement in mobile technology. In addition, they discussed the potential motivation of the users to

contribute their resources to the grid by suggesting the economic model, e.g. Game Theory, as a potential encouragement for the future growth of mobile grid computing platform.

Later on, Marinelli [5] proposed a novel framework of mobile grid computing, namely, Hyrax. The proposed framework was designed to allow computation to be separately executed on heterogeneous networks of smartphones and servers. The platform supported only the Android operation system. The implementation of Hyrax was based on the Hadoop framework. They discussed the main advantages of Hadoop's core functionality which were global data access, distributed data processing, scalability, fault-tolerance and data-local computation. Still, there were drawbacks. First, Hadoop was originally designed for the server computer. In other words, the platform itself required a large amount of CPU and memory usage. Second, Hadoop applied the technology that was not fit well with the mobile devices, e.g. XML, and required an expensive cost for parsing. Thus, the results showed that the overhead costs of running Hadoop were too high for Android phones due to the limitation of memory. For instance, Hadoop allocated memory buffers approximately 10 to 100 MB but the heap memory of Android application was limited to the maximum of 16 MB. The results showed that it would take up to 1000 units of Android G1s to achieve the performance of a single server.

Another work on mobile grid computer was Mobile OGSI.NET framework. This research was proposed by Chu [6]. In this research, they adopted OSGI (Open Grid Services Infrastructure) to integrate mobile devices to grid computing infrastructure. OSGI was developed on .NET Compact Framework which was an environment of Windows CE mobile. The research pointed out limitation issues of the mobile devices, e.g. resources limitation and network connectivity. The Mobile OGSI.NET architecture consisted of three layers as follows; Mobile Web Server, Grid Services Module and Grid Services. First, the Web Server component was designed to handle the connection between the endpoints. Second, the Grid Services Module unit was designed to handle message parsing and select an appropriate processing unit namely, Grid Services. The experimental results showed that the network latency of Mobile OGSI.NET had dominated the actual processing time of OGSI.NET on regular computers. In addition, they discussed that the overhead of task assignments was significant. For instance, the execution time could be degraded if they adopted too many devices for a small search space (for i.e. under 10,000). Thus, selecting the appropriate number of devices was a crucial issue.

More on this, Litke et al. [7] focused on fault tolerance of the mobile grid computing platform. In this research, they presented schema based on task replication. They adopted the Weibull reliability function to estimate the number of replicas that should be scheduled in order to guarantee a specific fault tolerance level.

From the study, we found that the concept of mobile grid computing takes the advantages of two greatly developed technologies. Still, to achieve the high efficiency, the platform requires large number of mobile devices. As suggested by Phan et al. in [8], one can potentially motivate the mobile users to contribute their devices' power with the correct model. One great practice of the incentive model has been derived to the concept of volunteer computing and will be discussed in the next section.

2.1.2 Volunteer Computing Platform

With the concept of mobile grid computing, the traditional grid computing infrastructure can be utilized with the contribution of mobile devices. However, it is compulsory that a great deal of mobile resources should be available for the platform to achieve that high efficiency. A survey from the previous literature review reveals that this type of sourcing can be accomplished in certain types of scientific applications. Even though none of the previous projects had used sourcing mobile devices, the concept and underlying infrastructure should be similar.

SETI@home [9] has been one of the biggest volunteer computing projects from 1998 up until present. Basically, it utilized millions of computers around the world to analyze the radio signals from the space. The execution itself was a compute-intensive task and required an unpredictable large amount of computational power and time. Within the first week after launch, there were over 200,000 mobile owners participating in a program to make a contribution. After the first 12 months, it was recorded that SETI@home's volunteers had processed 221 million work units. On average, the throughput was approximately 27.36 TFLOPS.

The platform design was simple enough for the end users. Specifically, to make a contribution, volunteers only needed to install the client software on the computer. This software could run either as a screensaver or in parallel with others. The client software got a work unit from the central server, processed the data and submitted the result once completed.

Later on, several @home projects were established. An outstanding one was Folding@home [10], starting in 2000, the project aimed to simulate protein folding and other bio-molecular phenomena. The architecture of this platform was based on a client-server model. The process started when the client had made a request for the work unit, which was a set of input files, from the "work servers". Then, the client needed to download the computational core from "web server" and began the execution. Once the task was completed, the client sent the result back to the work servers. However, if the work server was unreachable, all the data and log files would be passed to the "Collection Server". From the statistics, the volunteered computing units had scaled the performance of Folding@home up to 1000 times comparing to the local computing.

There also have been many more projects on volunteered computing projects designed to solve problems in different domains. For instance, ABC@home [11] was developed to find triples related to the ABC conjecture which was one of the greatest open problems in mathematics. BURP [12] project was to support the 3D rendering animation. Cosmology@home [13] was to perform the findings of the most appropriate and accurate models that could describe the universe. MindModeling@home [14] was to build a cognitive model of the human mind. FreeHAL [15] was to compute the information and generate the human conversation.

All in all, the aforementioned projects have one concept in common. Firstly, they call for the computing power from millions of computer owners worldwide. The client software runs either as a screen saver or in parallel with others. The client software gets a work unit from the source, analyzes, and sends the results back to the central server.

This computing model allows Sciences to benefit from free computing resources donated from all over the world. Examples of volunteer computing projects are summarized and listed in the Table 2.1.

Table 2.1 Volunteer Computing Projects

Project	Launched	Category	# of active processing units	TeraFLOPS
SETI@home	1999	Astrobiology	225,534	593.65
Folding@home	2000	Molecular biology	426,787	8,588
BURP	2004	Art	154	0.19
FreeHAL	2006	AI	9,522	14.019
ABC@home	2006	Mathematics	9,866	12.33
Cosmology@home	2007	Astronomy	8,124	9.12
MindModeling@home	2007	Cognitive Science	791	0.006

To develop a volunteer computing platform, one of the most important underlying components is the middleware infrastructure. Middleware is a software platform for both volunteer and grid computing. It was originally designed to abstract the resource management layer out of the computation. One of the most popular resource management middleware for the volunteer computing platform is Berkeley Open Infrastructure for Network Computing (BOINC) [16] which was originally designed to support the SETI@home project. It was designed as a client-server model. The server was designed to be a resource management unit and communicates with the BOINC clients over Internet. The client component was designed as a runtime system and must be installed on the volunteered computers. Basically, the client provides the process management, graphic control, file access and other functions. The BOINC core client is also capable of utilizing Graphics Processing Unit (GPU) as a scalable resource.

There was a study on the performance of BOINC. Anderson and Fedak [17] studied and analyzed over 330,000 mobile devices participating in a volunteer project based on BOINC platform. The measurement focused on the processing power, memory, storage space, network connectivity, client availability, and resources limitation. The results showed that the host pool had provided a great deal of computing resources with approximately 95.5 TFLOPS of processing rate and 7.74 Petabytes of storage with the speed rate at 5.27 Terabytes per second.

Bayanihan [18] is another middleware infrastructure which was developed by applying the web technology and utilizing Java's object-oriented functionality. For the volunteers, they have to either install the Java Applet plugin of the web browser or install the desktop application from the command line before executing task. There are two types of client which are categorized by roles. First is the Worker client whose role is to perform the execution. Second is the Watcher client whose role is to monitor the process and wait for the results. The client and server machines exchange messages by using HTTP protocol.

Another web-based volunteer computing middleware was proposed by [19]. This work was to allow the execution to be indeed portable, in other words, no plugin and no installation. To achieve the best performance of web technology, they suggested that

the execution unit should be developed with HTML5 and have the task executed on the background.

The studies show that the volunteer computing paradigm is an efficient platform for the computations of both data- and compute-intensive tasks. Even though the literature only showed about personal computer sourcing but the concept of mobile sourcing is similar. Thus, there exists the possibility of the successful sourcing on mobile computing power.

Still, the number of volunteered mobile devices does not always guarantee the performance of the framework. For instance, the communication overhead between mobile device and the platform may degrade the performance of system when too many devices are being employed. In order to truly utilize the computing power of mobile device and put it to good use, scheduling algorithm is then studied in order to select the optimal number of computing units and appropriate schedule.

2.1.3 Scheduling Algorithm

In this work, we were interested in the idea of volunteered computing. Also, we emphasized the scheduling process to find the optimum schedule as mobile sources will only be utilized if the cost can be saved and the execution time is still acceptable. Numerous works on mobile scheduling were then studied.

The classical blind search algorithm was adopted by Yuan Zhang et al. in [20]. They used depth-first and linear time searching to maximize the energy saved in mobile devices. An algorithm was designed based on the observation that when a task is offloaded, the subsequent invocations will have a high probability of being offloaded. The experiment results showed excellent performance with training data but it was not adaptive when there was a change in environment. Thus, they pointed out that the algorithm itself was not adaptive by the dynamic computing environment which is generally seen in the volunteered computing platform. Moreover, it might consume a large amount of time in case of a large search space

Another widely used algorithm was the Integer Linear Programming (ILP) algorithm. The works in [21], [22], and [23] designed the platform of mobile cloud computing which was to offload some compute-intensive tasks to the computers. These three studies implemented the scheduler based on the ILP algorithm with similar objective functions. They aimed at minimizing the energy consumption of mobile device and data transfer between mobile and computer. The results presented in [20] suggested that even though the runtime performance of ILP was better than the depth-first search, the results' accuracy from ILP was much lower than those from depth-first search.

In order to improve the quality of the schedules created, meta-heuristic algorithms were adopted. Qingfeng Liu et al. [24] used Genetic Algorithm (GA) for scheduling the tasks. The research aimed to maximize the speed and throughput of data streaming. However, this research ignored some factors such as CPU resources and network bandwidth. The experiment results showed that the optimization could improve the performance of processing power by 2x. It was also remarked that GA could obtain a near global optima solution, but the algorithm itself was compute-intensive. In addition, GA consumes high resource during the execution. Thus, they pointed out that GA potentially creates the overhead before offloading.

Another efficient algorithm class is Swarm Intelligence (SI). The work in [25] studied and compared the characteristic of several bio-inspired algorithms such as Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO) and Artificial Bee Colony algorithm (ABC). The study shows that various algorithms were designed differently and fit different types of problem formulation as shown in Table 2.2.

Table 2.2 Comparisons of Swarm Intelligence (SI) Algorithms

Algorithm characteristic	**ACO**	**PSO**	**ABC**
Solution representation	Graph/Similar structure for path-covering of ants	Real-valued	Real-valued
Type of decision variables	Binary values	Real Values	Discrete and Real values
Applicability to problems	Linear/nonlinear	Linear/nonlinear	Linear/nonlinear

We were particularly interested in the scheduling problem with binary variables, in other words, the solution can represent the decision whether to scatter the task to mobile. Also, the problem is formulated as Directed Acyclic Graph (DAG). With these preferences, ACO is the best-fit algorithm as it was designed in a similar manner.

A great amount of research had adopted ACO for the shortest path optimization problems. It was suggested by [26], the SI required lower resources and computational time than GA. This claim was also supported by [27] where ACO was proposed to obtain the routing path for wireless sensor. They aimed to minimize the computation and memory usage of the sensor nodes. The challenge in this study was that the sensor nodes had low CPU power and limited battery power. Moreover, the routing algorithm must be processed on an individual sensor. The results presented that ACO was capable of execution with low processing power and resource.

Advantages of ACO are also presented by [26], [28] and [29] where ACO was adopted to optimize the network routing algorithm. These pieces of research aimed to obtain the shortest path. The results were similar in a manner that ACO could present excellent scalability, high robustness, and adaptability for different environments.

From the characteristics of several algorithm discussed in this chapter, we were interested in adapting the ACO algorithm for use with scheduling problem in volunteer mobile sourcing platform. For better understanding of the readers, background on ACO is presented in the next section.

2.2 Background Study on Ant Colony Optimization (ACO) Algorithm

ACO is a meta-heuristic algorithm that belongs to the swarm intelligence class. The algorithm was inspired by the behavior of the ants searching for food around the colony. The remarkable benefit of ACO is the capability of searching for the optimal path with the limited computational resources [30].

It begins with a set of ants working from the starting point and randomly moves around to find a food source. During their walk, the ants deposit pheromone on the explored paths. The pheromone is a chemical which ants use to communicate with one another. Consequently, the majority of ants will travel the path with the highest pheromone concentration as it is assumed to be a path leading to the best food source. The pheromone values will be updated in each iteration due to both pheromone that is deposited and is evaporating, which can occur concurrently.

The overall workflow is presented in Figure 2.1.

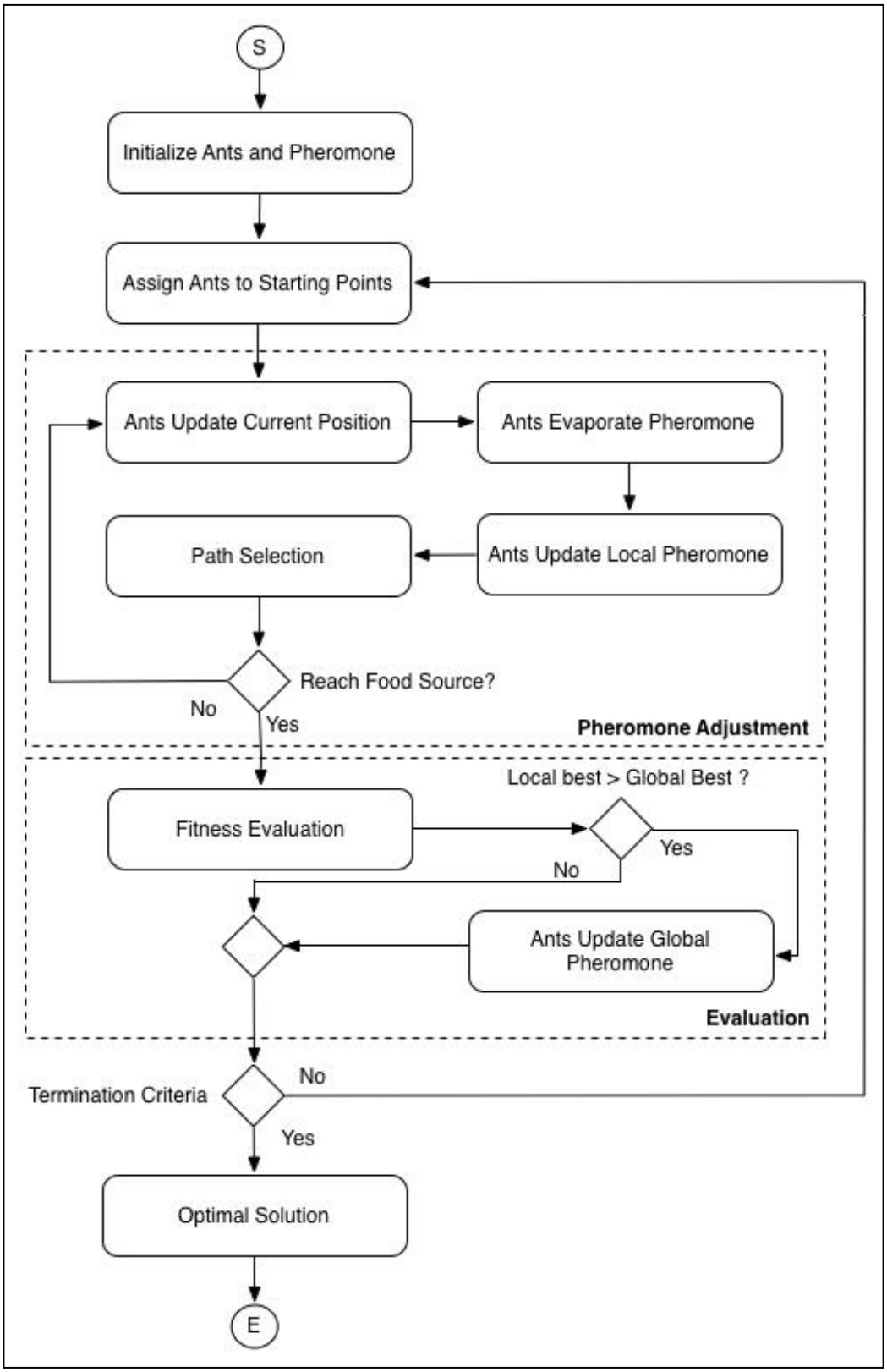

Figure 2.1 ACO algorithm

In this optimization problem, a path leading to each food source represents a feasible solution and the pheromone concentration in each trail is considered a fitness value. There are two main parts in ACO: path selection and pheromone updates.

2.2.1 Path Selection

ACO starts by randomly initializing the pheromone trails. The ants will then select paths according to the pheromone values (probability value) as in Equation 2.1. The path with higher pheromone values will have more chance of being selected.

$$p_{i,j}^{k}(t) = \frac{[\tau_{i,j}(t)]^{\alpha}[\eta_{i,j}(t)]^{\beta}}{\sum_{j \in N_i^k}[\tau_{i,j}(t)]^{\alpha}[\eta_{i,j}(t)]^{\beta}} \quad (2.1)$$

where:

$p_{i,j}^{k}(t)$ denotes the probability that an ant *k* goes from *i* to *j* at time *t*.
$\tau_{i,j}(t)$ denotes the pheromone value on the path from *i* to *j* at time *t*.
$\eta_{i,j}(t)$ denotes the heuristic information on the path from *i* to *j* at time *t*, which is used to select *j* when an ant is at *i*.
N_i^k denotes the feasible neighborhood of the ant *k*.
α denotes the weight of pheromone.
β denotes the weight of heuristic information.

2.2.2 Pheromone Updates

Once the ant selects the path, the pheromone values on the selected path will be updated due to both the pheromone depositing and evaporating using Equations 2.2 and 2.3. The most explored path will have the highest level of pheromone. The pheromone will be decreased on the less visited path.

$$\tau_{i,j}(t) = \rho\tau_{i,j}(t-1) + \sum_{k=1}^{n}\Delta\tau_{i,j}(t) \quad (2.2)$$

and

$$\Delta\tau_{i,j}(t) = \begin{cases} \frac{Q}{L_k(t)} & , if\ the\ path\ from\ i\ to\ j\ is\ chosen\ by\ ant\ k \\ 0 & , otherwise \end{cases} \quad (2.3)$$

where:

ρ denotes the pheromone trail evaporation. It can be in range of 0 to 1.
n denotes the number of ants.
Q denotes a constant for pheromone updating.
$L_k(t)$ denotes the cost of tour by the ant *k*.

From the equations, more pheromone is evaporated on the long path as ants have to spend more time travelling. Therefore, shorter paths will more likely to be selected. The algorithm will run recursively until the stopping criteria are reached, for i.e., the number of maximum iteration, number of evaluated solutions, execution time and fitness value. The shortest path, therefore, represents the optimal solution.

It is notable that there are various controlling parameters which can significantly affect the performance of an algorithm as presented below. These parameters are highly recommended to be fine-tuned with the problem

- ρ is a pheromone evaporation rate
- Q is a constant which affects the evaporation rate
- T_0 is an initial pheromone to be deposited
- The number of ants

In conclusion, this research attempted to design the mobile-sourcing framework to lighten the energy consumption at data centers. The framework aimed at utilizing the capability of mobile devices. From the studies, we were interested in the architecture of the framework proposed in [4] which consists of *Interlocutor* (server) and *Minion* (client) components. We also aimed to adopt the concept of volunteer computing where the computing power can be donated from the people across the globe. To reduce the effects of communication overhead, we emphasized the implementation of the scheduler based on the ACO algorithm. The main advantage of this algorithm is the capability to obtain the near optimal solution and the algorithm itself consumes less energy and resources. The design of the proposed framework and evaluation criteria are presented in the next chapter.

CHAPTER 3 PROPOSED WORK

In this work, we designed the mobile sourcing framework aiming to lighten the work load and energy consumption at the data centers. The framework attempts to distribute tasks to mobile devices while maintaining the execution timeframe. The design of this proposed framework is to support the volunteer computing concept by allowing the mobile users to contribute their devices to be public resources for Sciences.

The proposed framework attempts to assign the workflow tasks to appropriate computing units of two classes: volunteered mobile sourcing, which is free of charge, and computers at a data center, which can be costly. Thus, the goal is to use as much free sources as possible while satisfying the time constraint. The chapter presents the design of the proposed framework and evaluation methods.

3.1 Framework Design

The designed mobile sourcing framework is a composition of resource management and volunteered mobile devices as illustrated in Figure 3.1. The resource management unit will reside on the data center to manage the workflow execution. The input of this platform is the targeting workflow to be executed. Initially, the workflow is required to be presented in Extensible Markup Language (XML) format. When the input file is uploaded for the first time, application profiler constructs the abstract workflow as an application's profile. This profile is independent to the computing environment of data center and the current volunteered devices. Thus, it can be reused in any environment.

Meanwhile, the volunteered device profiler collects the profiles of the available volunteered devices by broadcasting the message. The application's profile is then used in conjunction with device profiles by the scheduler to obtain the optimal execution schedule. Once the optimality has been reached, execution node starts the execution by assigning the tasks to the targeting computing units, which can be either computer at data center or volunteered device. After the assignment, it will collect the results to complete the workflow execution.

On the mobile side, the volunteers are required to install the client application on their devices before making the contribution. The application includes the device profiler component which gathers the information, such as model and Internet connection status, and creates a device's profile. The profile is then submitted to the volunteered device profiler unit as discussed earlier. Once the scheduler has obtained the optimum schedule, the task is then assigned to the targeted device. The execution node starts the execution using local resources, collects, and submits the results accordingly.

The overall framework execution is illustrated in the following figure. Details of each component are presented in the next section.

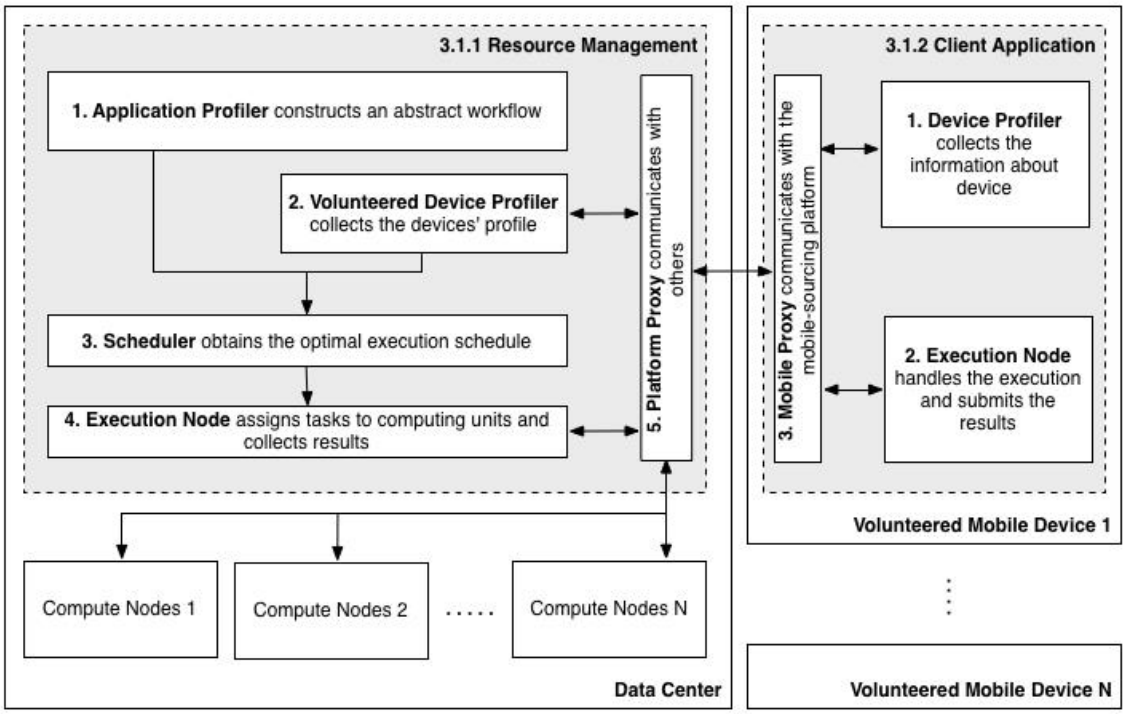

Figure 3.1 Mobile-sourcing Framework

3.1.1 Resource Management

This component is designed to be a core management module of the mobile-sourcing framework. It should be running on the data center to facilitate the workflow execution. The design of this component is based on the following assumptions.

- When the mobile device is not in used, it is always being charged and stationary. Thus, the battery will never run out.
- There is no application running on the mobile. Thus, the CPU is always in the idle state before the execution of any task.
- The memory on the mobile is sufficient.
- The network connectivity is in a stable condition.

Basically, it consists of 5 sub-components: application profiler, volunteered device profiler, scheduler, execution node and platform proxy. Details are as presented below.

1. Application Profiler

This component is designed to create the profile of the targeting application regardless of the computing unit. In other words, the constructed profile can be reused with any data center and any hardware. In this work, the profile of an application is presented as an abstract workflow in DAG format.

Abstract workflow is the representation of an application in an actual execution computing environment. In general, the user executes the application and records the computer's specification (i.e. memory and processor speed) and runtime of each task. This information is considered the baseline profile. Similar to any workflow, it consists of two parts – vertex and edge. In this work, the abstract workflow is constructed from the uploaded XML file as explained in the following section.

The XML input file defines the baseline information structure of the workflow, task dependencies, input/output, and runtime of each sub-task. In general, the XML input file consists of three main parts; header, task information and dependency.

Header (line 2-8)

The header of an XML file gives general information about the workflow which includes name, total tasks, baseline computing environment (processor speed and memory). In the example below, this workflow is interpreted as the *example* workflow consists of 3 tasks and it was executed on Intel Xeon E5-2440 with processor speed of 2.5 GHz and 2GB memory.

Task information (line 10 - 25)

The second part is the list of tasks in the workflow. Each task is presented with *job* tag. The header of each task describes task name and baseline runtime. For this example, the ID000 task is called *mProjectPP* and the runtime was 13.67 seconds when it was executed on Intel Xeon E5-2440.

Also, if the task requires input and/or output files, they are presented with *uses* tag. This tag includes name, size and input/output flag. In this example, task ID000 requires an input file of size 304 Bytes and produces 2 output files of size 104 and 12 MB.

Dependency (line 30 - 36)

The last part of the file presents dependency between tasks. The dependency is presented with *child* and *parent* elements, and the *ref* attribute that specifies the task ID. For this example, task ID001 and ID002 are children of task ID000.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2   < adag xmlns="http://pegasus.isi.edu/schema/DAX"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     name="example"
5     jobCount="3"
6     processorName="Intel Xeon E5-2440"
7     processor="2.5"
8     memory="2GB" >
9
10       < job id="ID000"
11          namespace="Montage"
12          name="mProjectPP"
13          runtime="13.67" >
14              < uses file="File_0.fits "
15                 link="input"
16                 size="304"/>
17                 link="input"
18                 size="304"/>
19              < uses file="File_1.fits "
20                 link="output"
21                 size="104000000"/>
22              < uses file="File_2.fits "
```

```
23                  link="output"
24                  size="12000000"/>
25          </job>
26
27          < job id="ID001" > ... </job>
28          < job id="ID002" > ... </job>
29
30          < child ref="ID001">
31                  < parent ref="ID000"/>
32          </child>
33
34          < child ref="ID002">
35                  < parent ref="ID000"/>
36          </child>
37
38  </adag>
```

Figure 3.2 Example of an XML File

Once the XML file is uploaded by a user, the abstract workflow is constructed using the Java's graph data structure. In the abstract workflow, each vertex represents a task (T_i) where the weight represents the baseline runtime. The incoming degree of vertex refers to the number of ancestors nodes while the outgoing degree is the number of children nodes. Each edge represents the dependency where the weight is the amount of data transferred between tasks.

From an XML example in Figure 3.2, the generated abstract workflow consists of three vertexes corresponding to task ID000, ID001 and ID002 plus start and terminal vertexes. The first executable task is ID000 which requires an input file of size 304 Bytes. The baseline runtime is 13.67 seconds. Once ID000 is completed, it produces two output files of size 104 and 12 MB for the children, ID001 and ID002. These two tasks can be executed in parallel. ID001 requires File_1 as an input file and the baseline runtime is 10.56 seconds. ID002 requires File_2 as an input and the baseline runtime is 23.03 seconds. Once these two are completed, the execution reaches terminal vertex and terminates. The example of abstract workflow is illustrated in Figure 3.3.

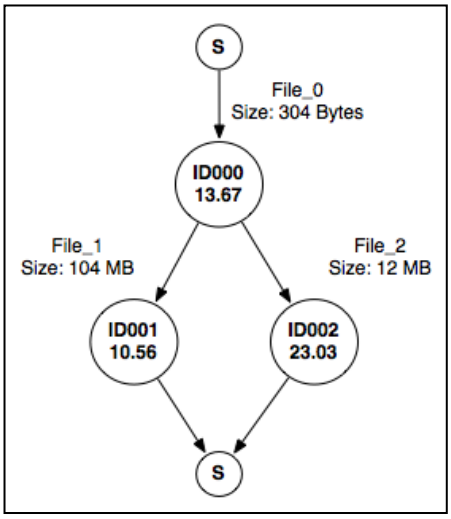

Figure 3.3 Example of Abstract Workflow

2. Volunteered Device Profiler

This component is designed to collect the profiles of the available volunteered devices. Prior to scheduling, the component broadcasts the message to the network of volunteered devices through the platform proxy. The request message is to ask the device for the generated profile. The profile includes the device's model, processor speed, memory, and network connectivity. This information will be used by the scheduler to construct a concrete workflow.

3. Scheduler

The scheduler component is designed to obtain an optimal execution schedule for the workflow. The schedule plan describes where task T_i should be executed. However, before the optimization, it has to extend the abstract workflow to be the concrete one by configuring the computing environment to be the current data center and available mobile devices. To do so, it relies on the baseline runtime storing in the vertex and the current computing environment. Thus, the profiles of the available devices will be acquired from volunteered device profiler beforehand.

Once the concrete workflow is constructed, the scheduler will perform the optimization to find the optimal execution plan. Thus, this component performs 2 steps, concrete workflow construction and workflow optimization. Details are presented as follows.

3A. Concrete Workflow Construction

Again, the input of this framework is an XML file which is used to generate the abstract workflow. At this stage, the abstract workflow is extended to a concrete one. The figure below illustrates the overall process of workflow construction from the uploaded XML file to the concrete level.

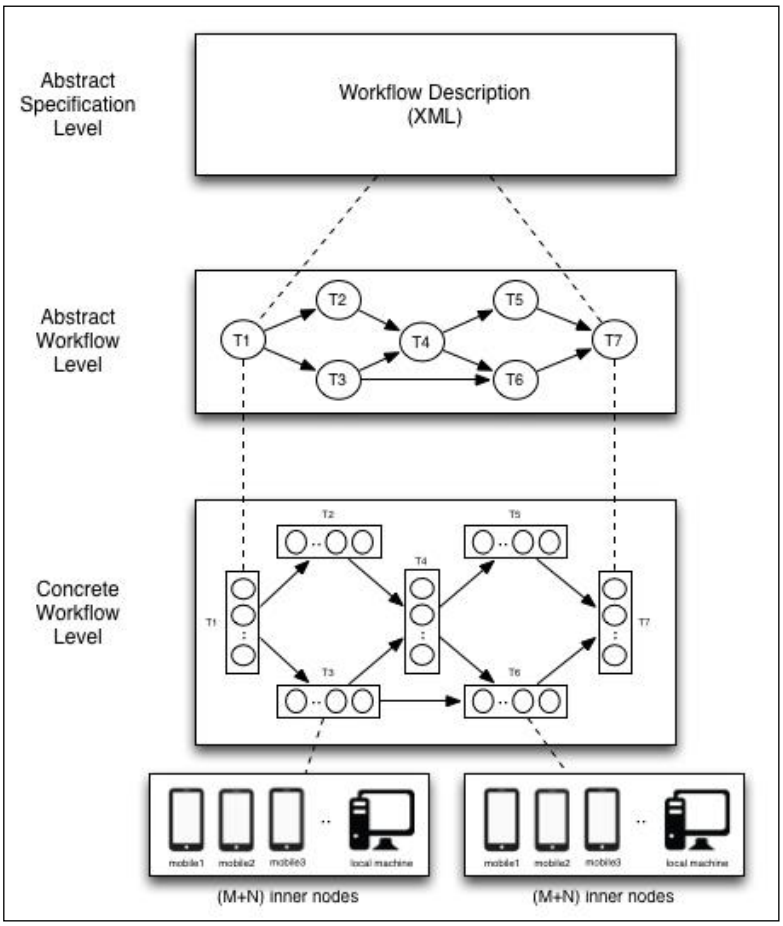

Figure 3.4 Workflow Construction Process

In this process, each vertex in the abstract workflow is mapped with the available machines to extend to the concrete node. The weight on each task T_i will be converted from the baseline runtime to the predicted execution time.

Notice that each vertex carries a set of weights since there are various possibilities for execution time depending on the processor it is scheduled to. If the available machines consist of M local processors and N mobile processors, the weights will be a vector of $(M+N)$ values.

The example below shows how task ID001 is extended to concrete vertex, assuming that the current available network consists of three mobile devices and one computer at the data center. The runtime labeled for each device is predicted by the table lookup process (details are presented in the following section).

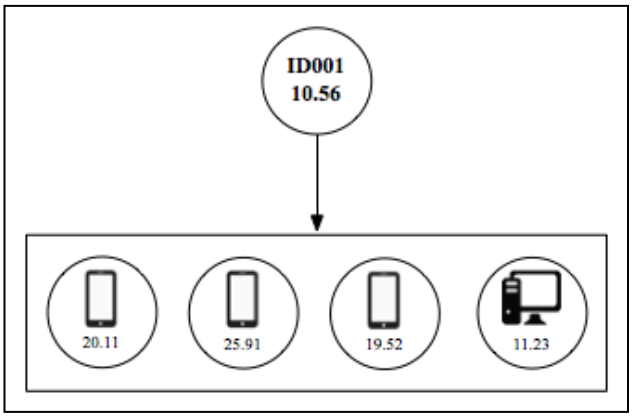

Figure 3.5 Extended Vertex

The structure of the workflow schedule can be explained as followed, let $G = (V, E)$ is the completely connected graph that represents the search space.

- V is a finite set of task $V = \{T_0, T_1 \ldots, T_n\}$, where n is the number of total tasks of the input workflow. Each task T_i $(0 \leq i \leq n)$ has n set of available machines $S_i = \{S_i^1, S_i^2, \ldots, S_i^k\}$, where $S_i^k (1 \leq k \leq M + N)$ represents runtime of task i on an available machine, k.
- E is a finite set of relations and dependencies of tasks. The weight on an edge represents an amount of data (in bytes) transferring between source and target nodes.

We assume that a data center is homogeneous. Runtimes on each machine for any task T_i will thus be the same. However, due to the variety of mobile devices, the runtime is estimated as a summation of processing time and data transfer time as expressed in Equation 3.1.

$$Total\ runtime = T_d + T_p \qquad (3.1)$$

where:

T_d denotes the data transfer time in second (s), calculated by Equation 3.2.
T_p denotes the processing time in second (s), calculated by Equation 3.4.

Data transfer time (T_d)

As we do not have access to the actual network of the service providers, we consider the network of the data center and volunteered mobile devices as star topology as illustrated in Figure 3.6.

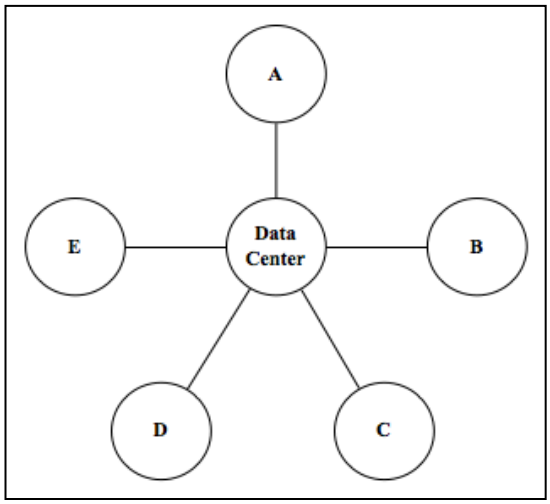

Figure 3.6 Star Topology

With this topology, we can estimate the transfer time as a function of network bandwidth and file size. The transfer time is estimated with Equation 3.2.

$$T = \frac{X * 8}{B} \tag{3.2}$$

where:
T denotes the estimated transfer time in second (s).
X denotes the transferred file size is megabyte (MB).
B denotes the network bandwidth in megabit per second (Mb/s).

To obtain a network bandwidth, the scheduler pings each volunteered device with a small packet to obtain turnaround time first. The turnaround time is the round trip time of a packet (from data center to device plus from device to data center). Once we have obtained the turnaround time, the network bandwidth of data center and the device can be calculated with Equation 3.3.

$$B = \frac{(2 * S_p)}{T_t} * 8 \tag{3.3}$$

where:
B denotes the network bandwidth in megabit per second (Mb/s).
S_p denotes the packet size in megabyte (MB).
T_t denotes the turnaround time in second (s).

For example, for a packet of 1 MB and turnaround time is 1.04 seconds, the network bandwidth can be calculated as follows.

$$B = \frac{(2 * S_p)}{T_t} * 8$$
$$B = \frac{(2 * 1)}{1.04} * 8 = 15.38$$

By repeatedly pinging all devices, the scheduler can obtain the network bandwidth of the current volunteer network. From a topology in Figure 3.6, assume that the turnaround times of the data center and devices are as in Table 3.1.

Table 3.1 Turnaround Time between Data Center and Devices

Device	Turnaround Time (Second)
A	1.04
B	2.00
C	0.59
D	1.79
E	1.40

The network bandwidth of the volunteer network is as follow.

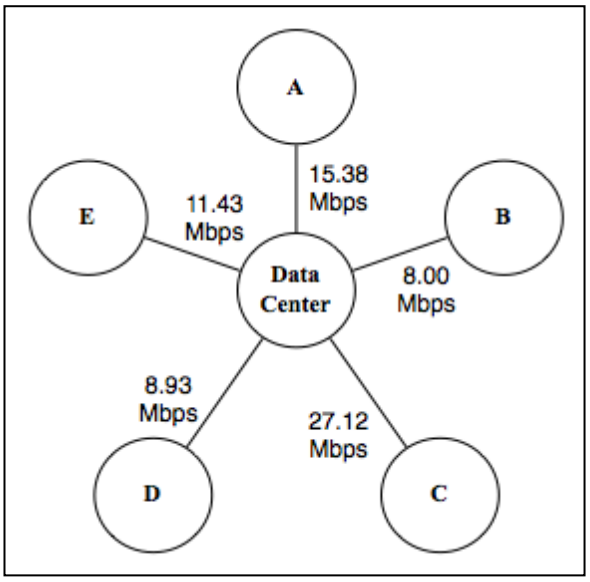

Figure 3.7 Network Bandwidth

Processing time (T_p)

Once the computing environment is defined, the mobile device specifications and network bandwidth are known. The processing time of each vertex can be estimated based on the baseline clock speed and runtime labeled on the abstract workflow. We used the input baseline numbers and performed a table lookup in the processor benchmarking data in www.passmark.com. Some examples of processor score are presented below.

Table 3.2 Processor Scores

	CPU	Score
1	Intel Xeon E5-2440 @ 2.40GHz	9,628
2	iPhone5s	5,075
3	Samsung Galaxy S4 (SHV-E330S)	4,432
4	Samsung Galaxy Note (SM-N9005)	4,303
5	HTC One	3,203

Form this table, it can be interpreted that the Intel Xeon is approximately three times faster than HTC One. To perform the table lookup, we referred to the baseline information and simply performed the Rule of Three with the following expression.

$$T_e = \frac{T_b * S_b}{S_e} \quad (3.4)$$

where:

T_e denotes the expected processing time on the targeting CPU in second (s).
T_b denotes the processing time on the baseline CPU in second (s).
S_e denotes the score of the targeting CPU.
S_b denotes the score of the baseline CPU.

For example, assume that the workflow was executed on Intel Xeon 2.4 GHz Processor (CPU type 1) as a baseline profile. The baseline runtime of the task ID001 is 10.56 minutes. Thus, the processing time when it is being executed on iPhone5s (CPU type 2) can be estimated as follows.

$$T_e = \frac{10.6 * 9268}{5075} = 20.11$$

3B. Workflow Optimization

Once the scheduler has constructed the concrete workflow, it then performs the optimization to find the optimal execution schedule for the workflow. In order to solve the scheduling problem for mobile-sourcing, all possible combinations of task assignments (candidate schedules) form a search space.

The objective of the optimization is to find the execution schedule which maximizes the cost saving at the data center (energy saving) while restricts the total runtime within the deadline. Note that if a scheduler outsources more tasks, the energy consumption at the data center will be lower, but the runtime may be higher. As we know that the performance of mobile device is not greater than computers, the execution time on the mobile device can be longer. In addition, there is an overhead of sending the data over wireless network, which takes a significant amount of time. Thus, the poor schedule may take too long for the overall execution process. In this work, we defined the deadline of the execution to be 1.5x of the runtime on local computer. For example, if the workflow takes 30 minutes executing on the local computer, the deadline constraint will be 45 minutes. Thus, any candidate schedule that took more than 45 minutes to complete would then be rejected.

We adopted the ACO algorithm to search for the best optimum schedule. ACO is meta-heuristic algorithm that was inspired by the behavior of the ants searching for food around the colony. The problem formulation of an algorithm is explained in Sections 1-3.

Section 1. Solution Representation

The solution to this problem is the schedule which can be considered a tasks distribution plan. It is represented by a one-dimensional array, which represents a path or trail that an ant explores. The size of the array equals to the size of the total tasks of the workflow (n). Each value in the array represents where task i should be executed. The figure below is an example of the solution which can be interpreted that task 1 and task 2 are executed on local machine #1, task 3 is executed on mobile device #1 and so on.

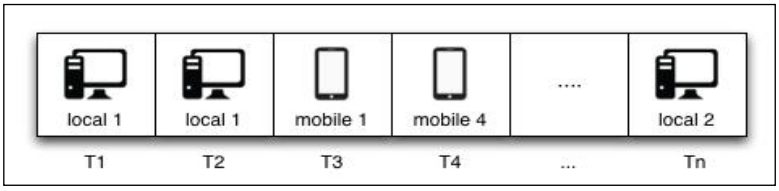

Figure 3.8 Solution representation

In each iteration, a group of P ant agents are attempting to find the optimal solution. The solution represents where the task T_i should be executed. Each node in the graph represents task T_i. Each T_i consists of a set of available machine S_i. Each node in the set S_i connects to every node in the set S_{i+1}.

To construct a solution, the ant uses the information of pheromone in order to select the next node. The ant explores every task assignment until it reaches the end node (E). The process is illustrated in Figure 3.9.

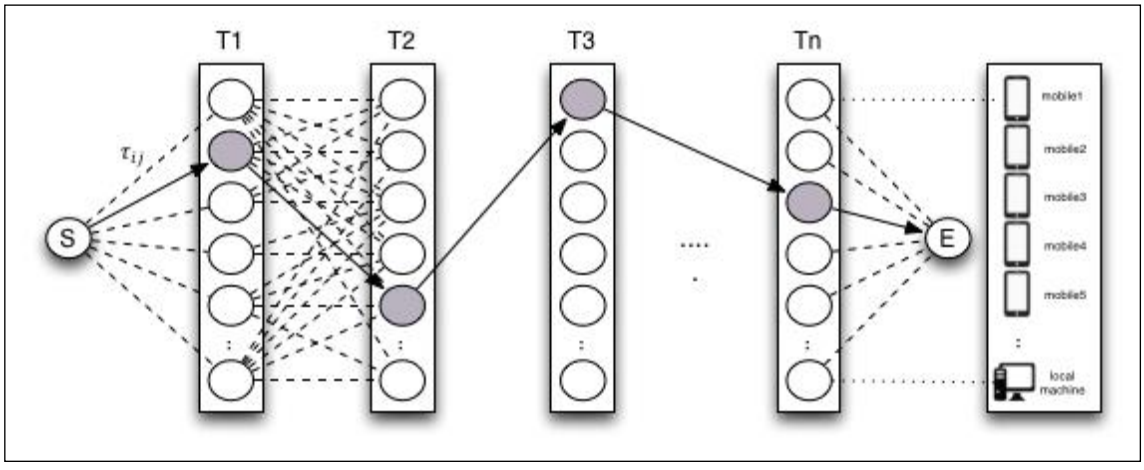

Figure 3.9 Process of creating a new solution

Section 2. Path Selection

At the beginning, every ant is assigned at the starting point S. Then, the ant selects the available machine for each task based on the probability value which has been calculated from the pheromone values. In order to select the appropriate machine for task T_i, we applied the roulette wheel selection method adapted from parent selection in genetic algorithms (GA). The path with a higher probabilistic value will have higher chance of being selected.

Section 3. Fitness Evaluation

When every ant reaches the end node (E), the trail each ant explored represents one solution. Then, all the candidate solutions are evaluated by the fitness function as the function of weighted scores between reduced energy and runtime. Then, the fitness values of *P* ants are compared to find the best candidate of the current iteration. The solution is then kept for comparison in the next iteration. *P* ants repeatedly search for the optimal solutions until the number of maximum iteration is reached.

The fitness value of the candidate solution can be calculated using Equation 3.5.

$$\boldsymbol{Maximize}\ f(x) = w_1 f_1(x) + w_2 f_2(x) \qquad (3.5)$$

where:

$f(x)$ denotes the fitness value of the candidate solution *x*.

$f_1(x)$ denotes the percentage of saved energy by the candidate solution *x*, calculated by Equation 3.6.

$f_2(x)$ denotes the percentage of saved runtime by the candidate solution *x*, calculated by Equation 3.9.

w_1 denotes the weighting factor of $f_1(x)$.

w_2 denotes the weighting factor of $f_2(x)$.

Saved Energy

As we aimed to lighten the energy consumption at the data center, this framework was designed to scatter as much tasks as possible to the network of volunteered mobile devices. However, high level of scattering does not guarantee the minimal energy consumption. One obvious reason is that transferring a task and related file to a mobile device consumes a significant amount of energy depending on the file size and network bandwidth.

Thus, one of the 2 objective functions is to maximize the saved energy at the data center. The percentage of saved energy can be calculated using Equation (3.6).

$$f_1(x) = \left(\frac{E_{local} - \sum_{i=0}^{n} Eexe_i + Erecv_i + Esend_i}{E_{local}} \right) * 100 \qquad (3.6)$$

where:

E_{local} denotes the total energy consumption for executing tasks on local machine.

$Eexe_i$ denotes the energy consumption for executing task, T_i, in joule (J), calculated by using Equation 3.7.

$Erecv_i$ denotes the energy consumption for transferring data from the mobile executing task, T_{i-1}, back to local machine, in joule (J), calculated by using Equation 3.8.

$Esend_i$ denotes the energy consumption for transferring data from local machine to the mobile to execute task, T_{i+1} calculated by using Equation 3.9.

n denotes the total tasks.

The energy consumption of the task execution is calculated from the equation below.

$$Eexe_i = k * W_{exe} * Texe_i \quad (3.7)$$

where:

k denotes the constant (3.6 x 10^6 J/kWatt).

W_{exe} denotes the maximum power of the baseline machine during an execution in kilowatt (kWatt).

$Texe_i$ denotes the runtime of task, T_i, in hour.

The energy consumption of data transfer between the data center and any mobile device is calculated with the equation below.

$$Erecv_i = Esend_i = k * W_{transfer} * \left(\frac{S_i}{B_k}\right) \quad (3.8)$$

where:

k denotes the constant (3.6 x 10^6 J/kWatt).

$W_{transfer}$ denotes the power of the baseline machine in the idle state in kilowatt (kW).

S_i denotes the total transferred packet size of task, T_i, in megabits (Mb).

B_k denotes the network bandwidth of mobile device k, in megabit per second (Mbps).

Saved Runtime

The second objective of our interest is the total runtime saved by the candidate schedule. As previously discussed, the overall runtime might be increased with the poor schedule. From observation, the runtime can be worsened by several reasons. First, it was when we scattered a task with large input and/or output files. The second reason is when a task was scattered to the long-distance or bad network connection device. Thus, it is crucial that the scheduler consider this objective along with the reduced energy consumption.

The percentage of saved runtime for the workflow execution can be calculated by Equation 3.9.

$$f_2(x) = \left(\frac{T_{local} - \sum_{i=0}^{n} Twait_i + Ttransfer_i + Texe_i}{T_{local}}\right) * 100 \quad (3.9)$$

where:

T_{local} denotes the runtime of an execution on local machine, in second (s).

$Twait_i$ denotes the waiting time before executing task, T_i, note that a task may have to wait for a machine to be available or wait for the arrival of input file.

$Ttransfer$ denotes the data transfer time, in second (s).

$Texe_i$ denotes the runtime of task, T_i, in second (s).

n denotes the total tasks.

Multi-objective Schema

To combine these two objectives, we applied weight-sum method to balance between energy and time saved. With this schema, there were two factors to be varied; w_1 and w_2 which are weights of energy and time saved respectively. The weighting factors can be varied between 0.0 and 1.0 but the summation always equals to 1.

The relationship of these two weighting factors indicates the significance of each objective function. For instance, if w_1 is greater than w_2, it means that the energy saved is more emphasized. These factors can be varied upon the execution of the framework to meet the user's preferences.

4. Execution Node

Once the scheduler component has constructed the execution schedule, the solution is transferred to execution node. This component was designed to handle the execution assignment and results gathering. To assign the task to a computing unit, the component packs the executable file and other related files, for i.e. input file, and communicates with a platform proxy to forward to the targeting computing unit.

Once assigned, this component also has to monitor the executing devices. If the task has to produce the output file, the executor will have to submit the result back to this component. Otherwise, execution node only performs the check-pointing to keep track of the execution. The component repeatedly assigns, monitors and gathers the results until the workflow is completely executed.

5. Platform Proxy

This component was designed to handle the communication between the mobile-sourcing platform and other components, for i.e. data center and volunteered mobile devices.

3.1.2 Client Application

As discussed earlier, the volunteers or mobile users are required to install the small application on the device before making a contribution. Once the user has started the donation, this application is launched and signals the data center. This application is a composition of three sub-components as follows.

1. Device Profiler

As discussed earlier, to construct the concrete workflow, the scheduler needs the information about the current network of volunteered devices. Before the construction, the volunteered device profiler on the data center broadcasts a request for the device's profile. Once received the request, this component collects the information of the device itself and sends it to the volunteered device profiler component. The profile includes device model, memory, processing speed, and network connectivity.

2. Execution Node

This component is similar to execution node in the data center. It was designed to handle the execution of an assigned task by using the mobile resources. Also, it monitors the execution and waits for the results. Once the execution is completed, it either sends a signal to the data center or submits a result package.

3. Mobile Proxy

This component is also similar to platform proxy. It was designed to handle the communications between mobile device and the mobile sourcing platform.

3.2 Evaluation Methods

In this work, we designed the mobile-sourcing framework with the objectives of reducing the energy consumption with the deadline constraint. The study focused on the scheduler component, which is crucial, as a poor one can affect the overall performance, for i.e. takes the execution too long. The scheduler involves the predictions of consumed energy and total runtime. Thus, the first evaluation criterion of this framework is the predictive accuracy of our scheduler.

To evaluate the accuracy, we designed the experiment to compare the predicted runtimes against the actual runtimes. Firstly, we set up the experimental environment with the designed network topology. We then implemented the mobile-sourcing framework and embed the predicting models for the scheduler. Initially, scheduler will be executed to find the optimum schedule and estimate the corresponding runtime and energy saving. The obtained schedule will then be used to distribute tasks in an actual environment. The total runtime and the amount of saved energy will then be recorded and compared against the predicted ones.

Besides the accuracy, we also evaluated the effectiveness of the framework. For this purpose, we designed the experiment to compare the quality of results from the proposed framework against the results from the framework with no scheduler.

The last experiment was the framework's scalability. The scalability was considered in two aspects. Firstly, we attempted to investigate the performance of framework when different sizes of input workflow were adopted. Ideally, the algorithm must still obtain a good enough solution for any size of input. Secondly, we attempted to investigate the performance of the framework when the number of volunteered mobile device varied. The expected performance is; more volunteered mobile devices, more energy and time are saved. In addition to the scalability results, we aimed to obtain the optimum number of volunteered devices where energy consumption and runtime were minimized.

All in all, we were particularly interested in three dimensions of the mobile-sourcing framework; accuracy, scalability and effectiveness. The experiments were set up accordingly. Details of the experimental designs and results are presented in the next chapter.

3.3 Usage Scenario

To adopt the mobile-sourcing platform, we assumed that the scientific organization already had a data center. One computer had to be configured as a server for the platform by installing the resource management unit. This computer had to connect to the access point where volunteered devices from the outside could also make a connection over the Internet. In addition, the server computer had to be able to communicate with other computers in the data center.

To allow mobile users to participate in the platform, the organization had to distribute the application to users. One possibility is to distribute through the public application stores, for i.e., Apps Store (iOS), Google Play (Android), and Windows Store (Windows). This application was designed to make the device become a platform's client. The application does not only communicate with the platform's server but it is also able to access mobile's resources. Once the users have installed this application on their device, they can then make the contribution.

Once the platform is configured, when scientists are about to execute the new workflow, firstly they have to upload the input XML file to the resource management unit. Then, the server broadcasts the message to all devices that have the application installed. The mobile user will receive the notification message asking for the permission to use mobile's resources. If the user accepts, the device's profile will be constructed and transferred to the server. Also, the device will be added to the devices pool as a consequence. With the workflow's and device's profiles, the scheduler is executed to find the optimum schedule. The platform then starts the execution by scattering tasks to the client devices. The diagram below illustrates the overall process of the platform.

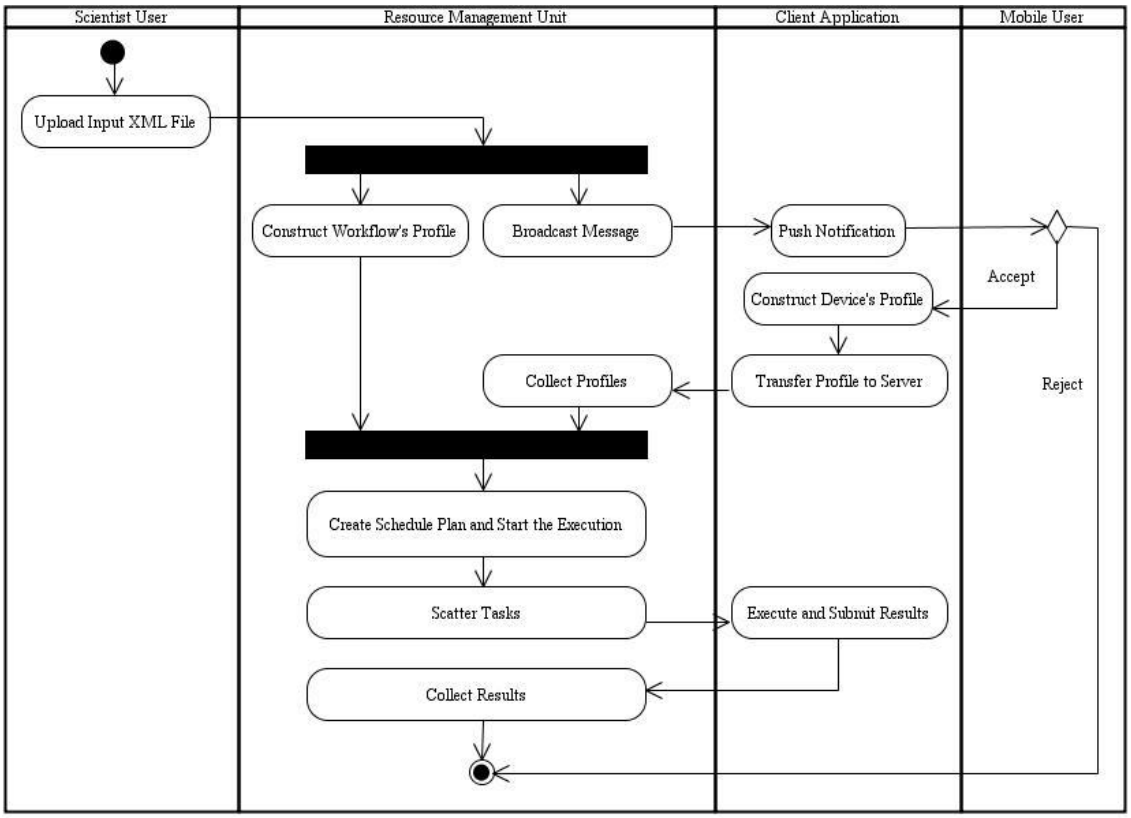

Figure 3.10 Platform Process

CHAPTER 4 DESIGN AND ANALYSIS OF THE EXPERIMENTS

In this work, we attempted to reduce the energy consumption in data centers with the mobile-sourcing framework. We also focused on the design and implementation of the scheduler crucial for the execution to meet a deadline. To achieve the goal, we applied the MOACO algorithm to obtain a near-optimum schedule for scientific workflow execution.

This chapter presents a series of experiments conducted against the evaluation criteria as discussed in the previous chapter. In the first section, we will describe the experiment designed to investigate the predictive power of the generated schedules. The second experiment was to investigate the framework's effectiveness. However, since there is no similar platform; we then decided to investigate an improvement in the results when MOACO scheduler is employed. The last experiment was to investigate the framework's scalability by the number of volunteered devices and size of workflow.

4.1 Model's Accuracy

In the first experiment, we attempted to investigate the predictive accuracy of our scheduler. The proposed scheduler was implemented based on the MOACO algorithm with the objective of minimizing energy consumption with the runtime constraint. The core contributing parameters of the scheduler are the total runtime and consumed energy. These two parameters were estimated by the predicting models as described in Chapter 3. In this experiment, we compared the estimated total runtime against the actual runtime. Similarly, the estimated consumed energy was compared against the actual energy. Details are as follows.

4.1.1 Dataset

In this experiment, we adopted data-intensive workflow downloaded from www.pegasis.isi.edu. This workflow was in the computing project, namely, Montage which was designed to deliver science-grade mosaics of the sky. This workflow was referred to as workflow A. The baseline profile of this workflow is presented in Table 4.1.

Table 4.1 Profile of Workflow A

# of Tasks	25
Energy Used (kJ)	57.8
Execution Time (min.)	3.9
Baseline Machine	Intel Xeon E5-2440 2.4 GHz with 2GB memory

The table below provides examples of tasks in this workflow. For example, the baseline runtime of task ID000 is 13.68 seconds, requires an input of size 304 bytes, and produces an output of size 4.2 MB.

Table 4.2 Example of Montage's Tasks

TaskID	Baseline Runtime (min.)	I/O Flag	File Size (bytes)	Parent Nodes	Child Nodes
ID000	13.68	input	304	-	ID005,ID006, ID007, ID009, ID011, ID0016
			4,222,080		
		output	4,159,489		
			4,159,489		
ID001	13.73	input	304	-	ID007, ID008, ID017
			4,222,080		
		output	4,151,600		
			4,151,600		
ID002	13.80	input	304	-	ID008, ID009, ID010, ID018
			4,222,080		
		output	4,141,450		
			4,141,450		

4.1.2 Experimental Setup

We implemented the prototype of the proposed framework with Java and jGraph (Directed Acyclic Graph manipulation plugin). The parameters setting of MOACO was imported from [31] and as follow: number of ants (p) = 10, alpha (α) = 3, beta (β) = 2, rho (ρ) = 0.01, and Q = 2.0. The stopping criteria were of two conditions. First was the iteration criterion which was to prevent an infinity loop. We set the maximum at 5,000 iterations. Second was the convergence criterion, in this work, the algorithm terminated when there was no better solution within 1,000 iterations. As for the multi-objective method, the weighting factor of saved energy (w_1) was 0.8 and 0.2 for saved time (w_2). The deadline constraint was defined as 1.5x baseline runtime. In other words, the runtime must not exceed 1.5 times of the baseline runtime.

As we attempted to validate the accuracy of models for predicting total runtime and consumed energy, this experiment was designed to compare the predicted values against the actual values. To achieve the goal, we set up an actual environment to simulate the mobile-sourcing framework. For the data center, we assumed that the computing environment consisted of 1 machine with similar specification to the baseline profile of Montage workflow (Intel Xeon E5-2440 2.4 GHz with 2GB memory). This machine also had the resource management component installed and run as a server. The server machine connected to the access point with the Ethernet cable via Gigabit Ethernet card. The maximum speed of the access point over wired connection was 100 Mbps. The client application was installed on multiple mobile devices and connected to the access point over the wireless Internet connection. The maximum speed of the access point over wireless connection was 54 Mbps. The network topology is illustrated in Figure 4.1.

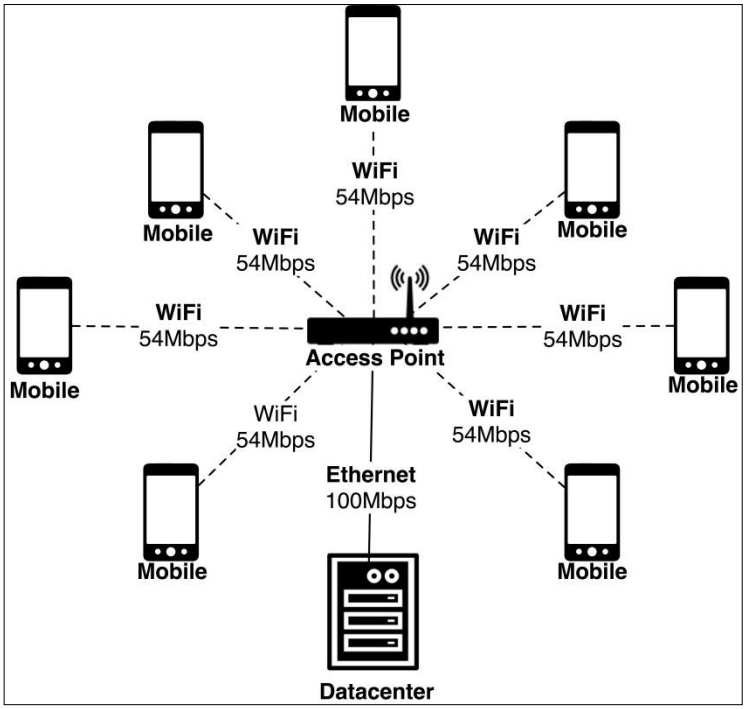

Figure 4.1 Network Topology of an Experimental Environment

With this environment, we had executed the framework 25 times and varied the number of volunteered devices from 1 to 25 with a single step. To be realistic, the network of volunteered devices was assumed to be heterogeneous where device's types were randomly chosen with an equal probability. Specifically, we chose the top four mobile devices available in the market as shown in Table 4.3.

Table 4.3 Types of Mobile Device

Type	Model	CPU	Cores	Memory	Architecture
1	iPhone5S	Apple A7 1.3 GHz	2	1 GB	ARM 64-bit
2	SS Galaxy S4	Cortex-A15 1.6 GHz	4	2 GB	ARM 32-bit
3	SS Galaxy Note3	Cortex-A15 1.9 GHz	4	3 GB	ARM 32-bit
4	HTC one	Krait 300 1.7 GHz	4	2 GB	ARM 32-bit

In each experimental run, the scheduler obtained an execution schedule, predicted runtime and predicted consumed energy. The execution schedule describes where each task should be executed. It was then used to distribute the set of tasks to devices. Once the workflow was completely executed, we recorded the actual runtime and calculated the amount of energy consumed (with Equation 3.7 and 3.8). Thus, we could compare

the predicted runtime and energy obtained by the scheduler against the actual values from the simulated mobile-sourcing environment.

4.1.3 Experimental Results

As described, we recorded the predicted runtime and consumed energy obtained by the scheduler. For each framework execution, we performed three simulations, collected the runtime, calculated the consumed energy and presented as an average in Table 4.4. Also, we calculated the difference in the predicted and actual values of both runtime and energy.

Table 4.4 Algorithm's Accuracy Results

# of Volunteered Devices	Runtime			Energy Consumption		
	Predicted (s.)	Actual (s.)	Accuracy (%)	Predicted (kJ)	Actual (kJ)	Accuracy (%)
1	257	240.84	93.29	46130.67	46430.06	99.36
2	259.28	263.14	98.53	44773.66	44159.35	98.61
3	307.4	296.58	96.35	47609.38	47527.15	99.83
4	257.9	270.58	95.31	43331.95	43425.98	99.78
5	256.6	240.58	93.34	41609.62	41208.05	99.03
6	324.12	348.16	93.10	39667.31	39071.95	98.48
7	304.1	318.11	95.60	36557.62	36958.31	98.92
8	330.84	348.16	95.03	37581.73	37313.32	99.28
9	320.76	315.42	98.31	38807.98	38274.47	98.61
10	272.36	275.54	98.85	35320.53	35311.91	99.98
11	346.5	356.5	97.19	37292.89	37292.9	100.00
12	305.19	315.06	96.87	35539.89	34826.68	97.95
13	330.11	331.67	99.53	37884.14	36140.27	95.17
14	309.45	319.39	96.89	33608.58	33602.9	99.98
15	309.51	319.56	96.86	37206.4	37210.94	99.99
16	325.51	311.86	95.62	35186.3	35821.66	98.23
17	286.52	276.62	96.42	36972.73	35982.18	97.25
18	330.38	344.19	95.99	34065.58	33446.39	98.15

Table 4.4 Algorithm's Accuracy Results (Continue)

# of Volunteered Devices	Runtime			Energy Consumption		
	Predicted (s.)	Actual (s.)	Accuracy (%)	Predicted (kJ)	Actual (kJ)	Accuracy (%)
19	306.39	321.98	95.16	36481.35	31040.25	82.47
20	319.38	309.9	96.94	32965.58	32017.04	97.04
21	338.68	343.63	98.56	36893.16	36588.09	99.17
22	343.2	355.47	96.55	38512.87	37739.3	97.95
23	331.17	343.11	96.52	34462.02	34656.49	99.44
24	349.52	360.13	97.05	37617.32	37678.49	99.84
25	316.47	330.88	95.64	35066.89	35508.14	98.76
		Average	**96.38**		**Average**	**98.13**
		SD.	**1.67**		**SD.**	**3.45**

To analyze if the predicted runtime and energy were different from the actual values, we performed the paired t- tests on mean differences of runtimes and energy values separately. For each test, we compared the predicted values against the actual values with 25 samples. The null hypothesis claimed that the predicted and actual values were not different. Before performing the t-test, we performed the normality test on the results which showed that the results were normally distributed.

The t-test results showed that the null hypothesis could not be rejected at 0.05 significant level for both runtime and energy tests. Thus, we concluded that there was no strong evidence that the predicted runtime was different from the actual runtime. There was also no evidence that the predicted consumed energy was different from the actual consumed energy.

H0: The predicted value equals to the actual value
H1: The predicted value does not equal to the actual value

Table 4.5 T-test Results

Test	T-value	P-value
Runtime	-2.05	0.051
Energy	2.04	0.052

From the tests, the results yielded that the predicting models of both runtime and energy consumption had accurately estimated the key factors for the optimization task. On average, the accuracies of runtime and energy prediction were 96.38% and 98.13%,

respectively, which were sufficiently high. The low standard deviations also showed the accuracy of the models. The results had proved our scheduler to be an accurate and efficient algorithm for the mobile-sourcing framework.

4.2 Solution's Quality

The second evaluation criterion of our interest was the scheduler's effectiveness. As the mobile-sourcing is a novel concept, no literature on adopting such concept to solve the scientific workflow exists. This experiment was to investigate the improvement of solution when MOACO scheduler was integrated into the framework. Initially, we chose ACO to solve the scheduling problem because the literature has revealed that the algorithm itself consumes small amount of resource and time. This characteristic is crucial to our framework as the overhead of task scattering should be minimized.

In this experiment, we compared the solution obtained by our framework with the solution obtained by the framework that has no scheduler. The comparative framework was developed by maintaining all components from our design but the scheduler. Instead of embedding MOACO, we used the round-robin (RR) algorithm to assign tasks to the computing units. The RR algorithm was to simply assign the task to the first available unit without the consideration on neither task's nor machine's profile. For instance, if there are 20 volunteered mobiles available and five tasks to be executed, the tasks will be assigned to the first five mobiles ordered by the arrival times.

For the experimental data, we adopted workflow A (detail in Table 4.1). The number of the volunteer devices was set to 14 units. Two frameworks were then separately executed in an actual environment with the same parameters setting to obtain the runtime and amount of saved energy. Results from the framework with MOACO were compared against results from the framework with RR. For each run, we had performed three executions and recorded the average. Note that the third column presents the percentage of saved energy by the baseline energy of 57,850.5 kJ. Similarly, the fifth column presents the percentage of saved time by the baseline runtime of 234 seconds. Notice that the negative values indicated that the runtime was not saved but only increased. For example, the -30.30% saved runtime means that the runtime increased by 30.30% from the baseline runtime. The last column presents the fitness value (calculated with equation 3.5) which was subjected to maximize.

Table 4.6 Results from MOACO and RR

	$f_1(x)$: Saved Energy (SE)		**$f_2(x)$: Saved Runtime (SR)**		**$f(x) = w_1f_1(x) + w_2f_2(x)$**
	SE (kJ)	**SE (%)**	**SR (s.)**	**SR (%)**	**Fitness Value**
RR	8,354.37	14.44	-243.38	-104.01	-9.25
MOACO	22,729.47	39.26	-70.90	-30.30	25.37
Improvement (%)	24.82		73.71		

The results show that the solution obtained by RR reduced the energy consumption only 14.44% (8,534.37 kJ) while MOACO's solution reduced an energy up to 39.26% (22,729.47 kJ). Thus, the improvement in energy reduction by MOACO accounted for 24.82%. As for the runtime, RR's solution increased the runtime of workflow execution up to 104.01% from the baseline runtime while MOACO's solution increased only 39.29%. Thus, the improvement in runtime minimization by MOACO accounted for 73.71%. Moreover, the runtime of RR's solution was not only poorer than ours but it also violated the deadline constraint (1.5x of baseline runtime).

We then considered on the runtime of the algorithms themselves. We found that the RR algorithm took only 1-2 seconds to obtain the solution because it considered nothing but the arrival times of the computing units. On the other hand, MOACO considered various factors and iterated thousands of times to obtain the optimum solution. It thus took 10-11 seconds to complete. Hence, we concluded that even the RR algorithm could obtain the solution much faster than MOACO; but the quality of RR's solution was not comparable with ours.

All in all, the results showed that the MOACO scheduler had significantly improved the solution's quality for both energy reduction and runtime minimization. We then concluded that the effectiveness of the framework has been contributed by the scheduling algorithm.

4.3 Framework's Scalability

Once the quality of the generated schedules had been validated, the last experiment was designed to investigate the scalability, another important performance metric of the framework. In this experiment, we considered the scalability in two aspects. Firstly, we emphasized the scalability of the framework by the number of volunteered mobile devices. We expected that the energy saving of the data centers should increase when more mobile devices were adopted. Secondly, we were interested in the framework's scalability by the size of input workflow. We expected that the quality of the obtained solution should be maintained regardless of the workflow's size.

In the first part of this experiment, we adopted the same workflow and parameters setting as the previous section. This workflow consisted of 25 tasks (detail in Table 4.1). To analyze the scalability, we executed the scheduler with different number of devices to obtain the runtime and amount of saved energy. The number of devices varied from 1 to 25 with a single step. Thus, we performed 25 experimental runs in total. Note that for each run, we performed three executions and recorded the average.
The figure below illustrates how the energy saving increased by the number of devices. The plot shows that the amount of energy saving steadily increased when we added more volunteered devices. With only five devices, the amount of saved energy increased up to approximately 34%. Even after that point, the saved energy still slightly increased and converged to the steady state at 13 devices. After this point, adding more devices did not further help reducing the energy consumption as the reduction rate was maintained at about 40%.

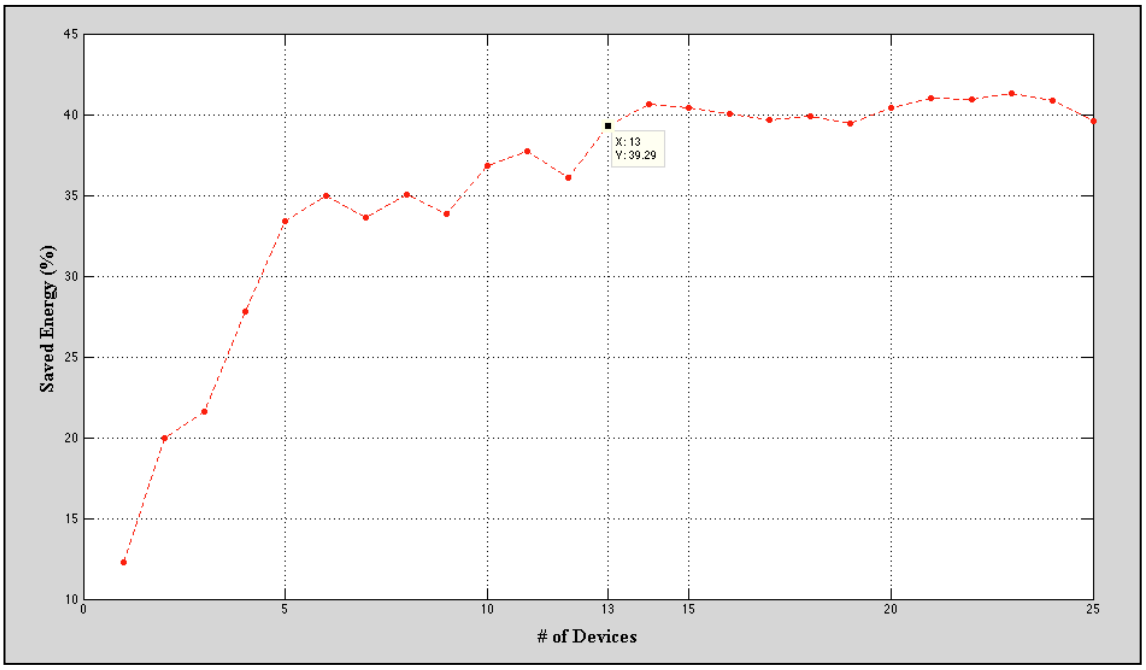

Figure 4.2 Plot of Saved Energy (%) and Number of Devices

The result showed that the solution of our proposed framework was well scaled by the number of devices. We then considered the obtained schedules and found that when there were more than 13 volunteered devices, the obtained schedules always scatter about 21 tasks to mobiles. In other words, even only 13 devices were available; the scheduler suggested scattering 21 tasks, thus, some devices were assigned with more than one task. Also, even we added 25 devices, the scheduler still suggested scattering only 21 tasks, thus, some devices were not used. Indeed, we believed that this was caused by the deadline constraint which was defined as 1.5x of the local execution. For example, if the scheduler scatters all 25 tasks when 25 devices were available, more energy could be saved, however, the runtime may exceed the deadline.

Figure 4.3 shows a plot of runtime versus number of devices. The runtime was presented as the percentage of the baseline runtime. For example, with 1 device, the runtime was 120%; meaning that the runtime was 20% higher than the baseline runtime. For this workflow, the baseline runtime was 30.62 minutes, 120% runtime was thus 36.74 minutes.

The plot shows that the runtime steadily increased by the number of devices until 11 devices were employed. After this point, the runtime steadily decreased until the cut-off point at 14 devices. Since the energy saving was stabilized at about 13 devices, the energy saving of this cut-off point was also optimal. This result drew the conclusion of the optimum number of volunteered devices for this workflow. With 14 devices, the energy saving was maximized while the runtime was minimized.

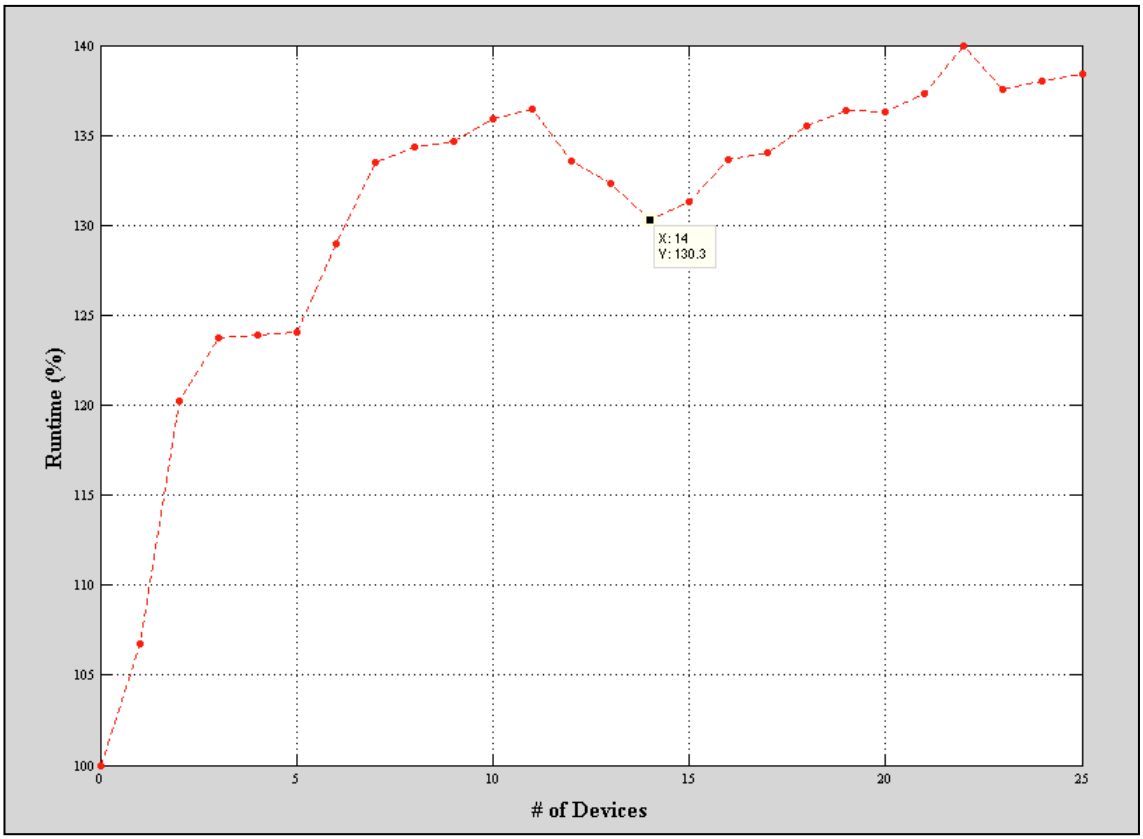

Figure 4.3 Plot of Runtime and Number of Devices

The second part of this experiment was to investigate the scalability of the framework by the size of workflow. Initially, we expected that the quality of solutions should be maintained regardless of the workflow's size. To begin with, we adopted the same workflow as the previous section. We also adopted another workflow of a different size under the same Montage project. This one was referred to as workflow B. The baseline profile of this workflow is presented in Table 4.6.

Table 4.7 Profiles of Workflow B

# of Tasks	50
Energy Used (kJ)	128.58
Execution Time (min.)	8.68
Baseline Machine	Intel Xeon E5-2440 2.4 GHz with 2GB memory

To analyze the scalability, we executed the scheduler in the similar manner with the same parameters setting to obtain the runtime and amount of saved energy. For each workflow, the number of devices was varied from 1 to *n* with a single step, where *n* was the number of tasks in workflow. For example, the number of devices was varied from 1 to 50 in the experiment with the this workflow (50 tasks). Note that for each run, we performed three executions and recorded the average.

As we aimed to validate that the framework could efficiently perform on any size of workflow, we then analyzed the amount of energy saving when it reached the steady

state. In this context, the steady state was when the framework could not achieve any higher reduction rate even more devices could be used. We were interested in an energy saving at this state because it was when the framework's performance was not limited by the number of devices and achieved its maximum efficiency.

For workflow B, the trend of energy saving was similar to the workflow A's results. With only five devices, the amount of reduced energy steadily increased up to 32.2%. The saved energy still gradually increased and converged to the steady state at 30 devices where the energy saving was approximately 42%.

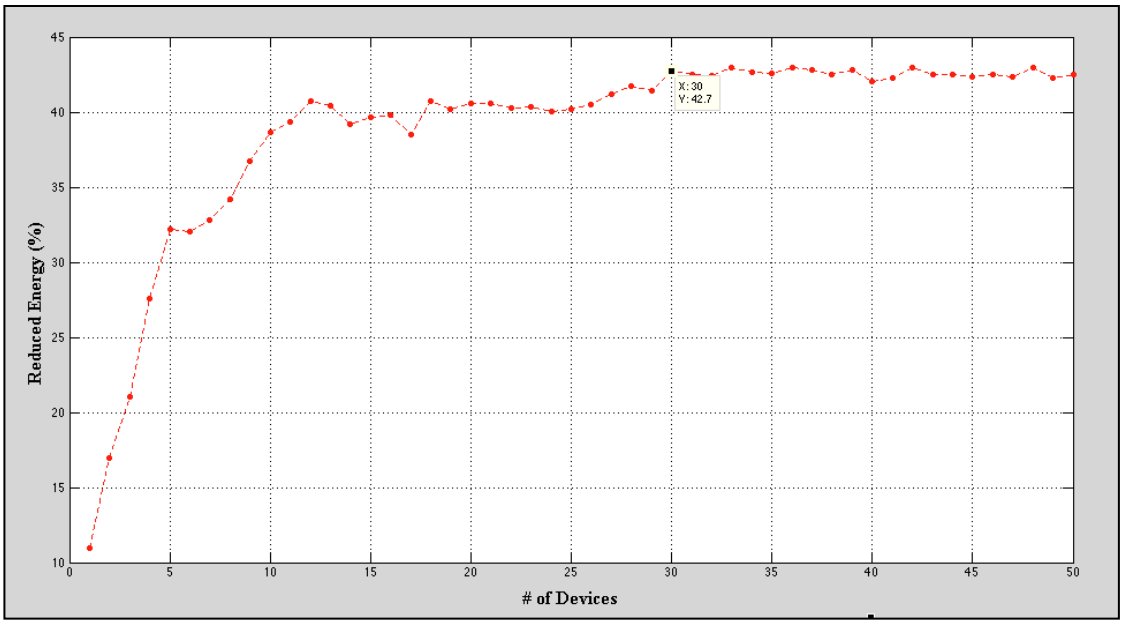

Figure 4.4 Plot of Saved Energy (%) and Number of Devices

From the results, the amounts of saved energy at the steady state for these two workflows were not much different. The reduction rate was even slightly higher for the larger workflow. Thus, we concluded that the framework can perform well on any size of workflow.

In conclusion, the experimental results proved that the proposed scheduler is an accurate and efficient algorithm for the mobile-sourcing framework. The effectiveness of our framework was also validated with an improvement in the results achieved by the MOACO scheduler. In addition, the framework proved to be well-scaled by the number of volunteered devices. With a sufficient number of devices, the framework will be capable of reducing energy consumption at data centers with sufficiently high reduction rate at 40%. The runtime also meets the deadline constraint (1.5x of local execution). The framework was also proved to be well performing on different sizes of input workflow.

CHAPTER 5 CONCLUSION

In this work, we proposed the novel concept to lighten the work load and energy consumption at data centers with the mobile-sourcing framework. We were particularly interested in the scientific workflow execution as it usually consumes large amount of energy and time. The essential component of our framework was the scheduling algorithm as we realized the effects of task scattering's overhead. We adopted the MOACO algorithm with an objective of minimizing both energy consumption and runtime with the deadline constraint.

Our framework was designed based on the volunteer computing concept. The basic idea is to allow mobile users to donate their computing resources when they are not in used. To simplify the design, we made a few assumptions as follow. We assumed that the mobile users donate the computing power only when the device is being charged. Specifically, the mobile is stationary and the network connectivity is stable. In addition, there is no application running on the mobile, in other words, it is idle. Thus, the donated device can be fully utilized.

The designed framework consists of two components; resource management and client application. The resource management unit resides in the data center to handle the workflow execution. It was designed to manage the incoming workflow, create the schedule and scatter tasks to the appropriate computing units. Client application is installed on the volunteered device to handle task assignment, execution and results submission. These two components communicate over the Internet.

Our emphasis is, however, on the scheduler. Our MOACO algorithm was designed to create the optimum schedule for the workflow execution. The crucial element of scheduler is the prediction models for energy consumption and runtime. Prior to the scheduling process, both application's and devices' profile are acquired for the prediction models. The scheduler can estimate the energy consumption and runtime of each candidate solution. The estimated values are then combined with the weight-sum method to be a fitness value. The optimum schedule is used for scattering task to either local computer or mobile device.

The evaluation criteria of our interest are in three aspects; 1) the power of the prediction accuracy, 2) solution's quality and 3) the framework's scalability. For the first criteria, the consumed energy and runtime from the framework's execution in an actual environment were compared against the estimated values. The results showed that our models had 98% accurately predict the energy consumption and 96% accuracy for the runtime. For the second experiment, the solution obtained by our framework was compared with the solution obtained from the framework that has no scheduler. The results showed that ours had outperformed in both aspects of energy consumption and runtime. Lastly, the framework was executed with different numbers of volunteered devices and different sizes of input workflow. The results showed that the framework scaled well by the number of devices. Also, it could perform well with any size of workflow.

The suggestion for improvement can perform as follows. Since the design of this framework is in its early stage and we have only focused on the scheduler, there still

exist areas for continuous development. Firstly, the device profiler component was initially designed to gather only three types of information; device's model, memory, processing speed, and network connectivity. However, the performance of mobile device does not depend on only these factors; there are others that can potentially degrade the performance. For example, malware, computer virus, different versions of the operating system, and background processes. Thus, investigating these factors should improve the accuracy of the scheduling and promote the practicality of the platform.

Secondly, this work has assumed that the data center consists of only one computer. However, the actual data center is usually heterogeneous and consists of a number of computers. Mostly, different computers have different performances and profiles, for i.e., energy consumption, speed. Thus, it would be more practical to study the actual computing environment and take it into account.

Last is the error handling. In this work, we did not consider on any error or fault during the execution. For instance, if the task was already assigned to a mobile but the execution cannot be completed due to some unexpected causes. The framework should have a protocol to prevent and handle such situations. One possibility is to use the heartbeat protocol. This protocol is to continuously monitor the resources by exchanging a message periodically. If the mobile is lost, the server should re-scatter the task to another.

In summary, this framework was initially designed with the emphasis on the scheduler. Even though the experimental results proved the framework to be sufficiently good for the organizations, there still exist areas for further enhancements which potentially promote the practicality of the framework.

REFERENCES

1. ITU, 2013, “Key ICT Indicators for Developed and Developing Countries and the World (totals and penetration rates)”, **ITU World Telecommunication/ICT Indicators database**.

2. Koomey, J. G., 2008, "Worldwide Electricity Used in Data Centers." **Environmental Research Letters**.

3. Litke, A., Skoutas, D. and Varvarigou, T., 2004, "Mobile Grid Computing: Changes and Challenges of Resource Management in a Mobile Grid Environment", **in 5th International Conference on Practical Aspects of Knowledge Management**.

4. Phan, T., Huang, L. and Dulan, C., 2002, “Challenge: Integration Mobile Wireless Devices into the Computational Grid”, **in Proceedings of the 8th annual international conference on Mobile computing and networking**, pp. 271-278.

5. Marinelli, E., 2009, “Hyrax: Cloud Computing on Mobile Devices using MapReduce” **Carnegie-mellon univ Pittsburgh PA school of computer science**.

6. Chu, D. C. and Humphrey, M., 2004, “Mobile OGSI.NET: Grid Computing on Mobile Devices”, **Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing**, pp.182-191.

7. Litke, A., Skoutas, D., Tserpes, K. and Varvarigou, T., 2007, "Efficient Task Replication and Management for Adaptive Fault Tolerance in Mobile Grid Environments", **Future Generation Computer Systems**, pp. 163-178.

8. Phan, T., Huang, L. and Dulan, C., 2002, “Challenge: Integration Mobile Wireless Devices into the Computational Grid”, **in Proceedings of the 8th annual international conference on Mobile computing and networking**, pp. 271-278.

9. Anderson, D. P., Cobb, J., Korpela, E., Lebofsky, M. and Werthimer, D., 2002 , “SETI@ home: An Experiment in Public-resource Computing”, **Communications of the ACM**, pp. 56-61.

10. Beberg, A. L., Ensign, D. L., Jayachandran, G., Khaliq, S. and Pande, V. S., 2009, “Folding@ home: Lessons from Eight Years of Volunteer Distributed Computing”, **in Parallel & Distributed Processing, IPDPS 2009. IEEE International Symposium**, 2009, pp. 1-8.

11. University of Leiden, **ABC@home** [Online], Available: http://abcathome.com/conjecture.php (July, 1).

12. J. Kristensen, **Beta is starting soon** [Online], Available: http://burp.renderfarming.net/forum_thread.php?id=1831 (July, 1).

13. Fendt, W. A. and Benjamin D.W.., 2007, "Pico: Parameters for the Impatient Cosmologist", **The Astrophysical Journal**.

14. Zutter W., "**MindModeling@home: Credit Overview**" [Online], Available: http://boincstats.com/en/stats/63/project/detail [2008, January 31].

15. "BOINC Stats – FreeHAL" (n.d.) In **BOINC Stats**. Retrieved 28 Feb, from http://boincstats.com/en/stats/78/project/detail

16. Anderson, David P., 2004, "Boinc: A System for Public-resource Computing and Storage", **Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on. IEEE**, 2004.

17. Anderson, D. P. and Fedak, G., 2006, "The Computational and Storage Potential of Volunteer Computing", **Cluster Computing and the Grid, CCGRID 06. Sixth IEEE International Symposium**, Vol. 1, pp. 73-80.

18. Sarmenta, L. F. and Hirano, S., 1999, "Bayanihan: Building and Studying Web-based Volunteer Computing Systems Using Java" **Future Generation Computer Systems**, pp.675-686.

19. Krupa, T., Majewski, P., Kowalczyk, B. and Turek, W., 2012, "On-demand Web Search Using Browser-based Volunteer Computing" **Complex, Intelligent and Software Intensive Systems (CISIS)**, pp. 184-190.

20. Zhang, Y., Liu, H., Jiao, L. and Fu, X., 2012, "To Offload or Not to Offload: An Efficient Code Partition Algorithm for Mobile Cloud Computing in Cloud Networking", **IEEE 1st International Conference**, pp. 80-86.

21. Cuervo, E., Balasubramanian, A., Cho, D. K., Wolman, A., Saroiu, S., Chandra, R. and Bahl P., 2010, "MAUI: Making Smartphones Last Longer with Code Offload", **in Proceedings of the 8th international conference on Mobile systems, applications, and services**, pp. 49-62.

22. Kovachev, D., Yu, T. and Klamma, R., 2012, "Adaptive Computation Offloading from Mobile Devices into the Cloud", **Parallel and Distributed Processing with Applications (ISPA)**, 2012 IEEE 10th International Symposium, pp.784-791.

23. Chun, B. G., Ihm, S., Maniatis, P., Naik, M. and Patti, A., 2011, "Clonecloud: Elastic Execution between Mobile Device and Cloud"**, in Proceedings of the sixth conference on Computer systems**, pp. 301-314.

24. Liu, Q. Jian, X, Hu, J., Zhao, H. and Zhang, S., 2009, "An Optimized Solution for Mobile Environment Using Mobile Cloud Computing", **Wireless Communications, Networking and Mobile Computing, 2009. WiCom '09. 5th International Conference**, pp. 1-5.

25. Reddy, J. M. and Kumar, N. D., 2012, "Computational Algorithms Inspired by Biological Processes and Evolution", **Current Science (Bangalore)**, pp. 370-380.

26. Wang, J., Osagie, E., Thulasiraman, P. and Thulasiram, R. K., 2009, "HOPNET: A Hybrid Ant Colony Optimization Routing Algorithm for Mobile Adhoc Network", **Ad Hoc Networks**, pp. 690-705.

27. Saleem, M., Di Caro, G. A., and Farooq, M., 2011, “Swarm Intelligence Based Routing Protocol for Wireless Sensor Networks: Survey and Future Directions”, **Information Sciences**, pp. 4597-4624.

28. Okdem, S. and Karaboga, D., 2009, “Routing in Wireless Sensor Networks Using an Ant Colony Optimization (ACO) router chip”, **First NASA/ESA Conference**, pp. 401-404.

29. Kadono, D., Izumi, T., Ooshita, F., Kakugawa, H. and Masuzawa, T., 2010, “An Ant Colony Optimization Routing Based on Robustness for Adhoc Networks with GPSs”, **Ad Hoc Networks**, Vol. 8, No.1, pp. 63-76.

30. Liu, Q., Jian, X., Hu, J., Zhao, H. and Zhang, S., 2009, “An Optimized Solution for Mobile Environment using Mobile Cloud Computing”, **Wireless Communications, Networking and Mobile Computing, 2009. WiCom'09. 5th International Conference**, pp. 1-5.

31. J. McCaffrey, **Ant Colony Optimization** [Online], Available: http://msdn.microsoft.com/en-us/magazine/hh781027.aspx (2014, January).

CURRICULUM VITAE

NAME	Mr. Katchaguy Areekijseree
DATE OF BIRTH	15 November, 1989
EDUCATION RECORD	
HIGH SCHOOL	High School Graduation, Sri Ayudhya School, 2007
BACHELOR'S DEGREE	Bachelor of Engineering (Computer Engineering) King Mongkut's University of Technology Thonburi, 2012
MASTER'S DEGREE	Master of Engineering (Computer Engineering) King Mongkut's University of Technology Thonburi, 2014
SCHOLARSHIP/ RESEARCH GRANT	NSTDA-University-Industry Research Collaboration: NUI-RC
PUBLICATION	Areekijseree, K. and Achalakul, T., 2014, "Volunteered mobile sourcing with multi-objective ant colony optimization", **Computer Science and Software Engineering (JCSSE)**, **International Joint Conference on IEEE**, pp. 248-253.
	Pattananagkur, T., Tanpabrungson, S., Areekijseree, K., Pumma, S. and Achalakul, T., 2013, "The Design of SkyPACS: a High Performance, Mobile Medical Imaging Solution", **Symposium on GPU Computing and Applications**.
	Tivatansakul, S., Tanupaprungsun, S., Areekijseree, K., Achalakul, T. and Ohkura, M.,

2012, "Intelligent Space Design for the elderly," **in proceedings of the 6th International Symposium South East Asian Technical University Consortium (SEATUC2012).**

Tivatansakul, S., Tanupaprungsun, S., Areekijseree, K., Achalakul, T., Hirasawa, K., Sawada, Sh., Saitoh, A. and Ohkura, M., 2012, " The intelligent space for the elderly - including activity detection", **in proceedings of 4th International Conference on Applied Human Factors and ergonomics (AHFE2012)**.

Tivatansakul, S., Tanupaprungsun, S., Areekijseree, K., Achalakul, T., Hirasawa, K., Sawada, Sh., Saitoh, A. and Ohkura, M., 2012, "The intelligent space for the elderly – Implementation of fall detection algorithm" **in proceedings of the 51st Annual Conference on the Society of Instrument and Control Engineers of Japan (SICE2012)**.

Tivatansakul, S., Tanupaprungsun, S., Areekijseree, K., Achalakul, T. and Ohkura, M., 2012, "The Intelligent Space for the Elderly," **in proceedings of 10th Asia Pacific Conference on Computer Human Interaction (APCHI 2012)**.