Automating Mobile Task Offloading with Agent Based Auctions

A thesis

presented to

the Faculty of the Department of Engineering Technology

University of Houston

In partial fulfillment

of the Requirements for the Degree

Master of Science

in Engineering Technology

By

Nidhi Niket Shah

December 2016

Acknowledgement:

First and foremost, I express my sincere gratitude to my advisor Dr. Ricardo Lent for his support, patience, guidance and constant encouragement throughout my study and thesis work. It has been a constant learning process for me under his expertise. I would never thank him enough for his time and dedication he paid for my work.

Besides my advisor, I would like to thank my thesis review committee: Dr. Driss Benhaddou and Dr. Xiaojing Yuan, for their time, interest, and helpful comments. I want to extend my special gratefulness towards Dr. Xiaojing Yuan for always motivating me and giving me the opportunity to attend conferences and meet so many interesting people.

I also wish to express my heartfelt thankfulness to my friends, especially Palak Desai, Manvika Mohan, Harika Siruvolu, Susmitha Sama, Sachin Sahane, Amar Singh, Mydhili Palagummi, Sonal Harsh, Gandhimathi Velusamy, Andia Karimipoor, Govind Prasad and many who have rendered their continuous support for the successful completion of the project, both explicitly and implicitly.

Lastly, I would like to thank my family for all their love and encouragement, especially my loving daughter Manya. Her smiling face kept me tension free through difficult times. Most of all a special appreciation to my husband Niket Shah for being patient and supportive throughout my degree program. He is my biggest critique. His experience and knowledge always inspired me to work hard.

Abstract:

Mobile Cloud Computing has been introduced to be a potential technology for mobile services. It solves mobile computing's fundamental problems such as resource scarcity, frequent disconnections, and mobility. Mobile cloud computing can address these problems by executing mobile applications on resource providers external to the mobile device i.e. by offloading them onto the cloud servers. Cloud computing allows users to use infrastructure, platforms and software in an on-demand fashion. Thus, data storage and processing services in clouds, eliminates the need for mobile users to have a powerful device configuration (e.g. CPU speed, memory capacity etc.), as all resource-intensive computing can be performed on the cloud. The data center hardware and software is referred as the cloud and the cloud can be made available "pay-as-you-use" manner. This payment for services to cloud is based on services we require for cloud to process. These services are further differentiated according to types, availability and quality. For instance, to run any particular application, one cloud provider offers less price than other cloud provider but its quality is compromised. In some cases, single cloud is not enough to meet mobile user's demands. Therefore, new scheme is needed in which the mobile users can utilize multiple clouds in a unified fashion and get opportunity to choose the best cloud provider to serve them.

The present research addresses this issue and presents design scenario for the agent based auction model where mobile users get chance to receive best services from one of the cloud providers which offers minimum cost including price and quality of the service. This

quality of service refers to energy and latency optimization. The agent based framework has been designed on JADE framework, to provide robustness to the model. Concept of auction market and auction manager has been introduced in this research that helps mobile users to get energy-latency optimization with optimal price to be paid to the cloud providers for accessing their resources. The proposed mechanism has been evaluated for different scenarios and it gives optimal solution for all the cases. Thus, proposed approach can be scaled in future for more features.

Table of Contents:

List of Figures:

List of Tables:

List of Abbreviations:

MCC	Mobile Cloud Computing
CC	Cloud Computing
MCO	Mobile Cloud Offloading (MCO)
JADE	Java Agent DEvelopment Framework
MTSB	Multi-dimensional, two sided, Sealed Bid Auction
FIPA	Foundation for Intelligent Physical Agents
DF	Directory Facilitator
RMA	Remote Monitoring Agent
AMS	Agent Management System
IMTP	Internal Message Transport Protocol
RMI	Remote Method Invocation
MTP	Message Transport Protocol
IIOP	Inter-ORB Protocol
HTTP	Hypertext Transfer Protocol
ACL	Agent Communication Language
SLA	Service Level Agreement
SB	Sealed-Bid
GUI	Graphical User Interface
TIM	truthful incentive mechanism
EDA	Efficient Design of Auction

Chapter 1

Introduction

Mobile Cloud Computing (MCC) is concept of combining Cloud Computing (CC) into the mobile environment. It provides new services for mobile users taking full advantages of cloud computing, thereby overcoming their inherent limitations [1, 2]. There are many popular cloud-based mobile applications deployed on cloud platforms such as in Amazon Silk [51], Apple iCloud [50] etc. Thus, mobile cloud computing appears as an appealing paradigm to accommodate demands for running computation intensive power-hungry applications over resource-constrained mobile devices in storage, energy and networking.

Mobile cloud offloading (MCO) is becoming a promising method to reduce execution time and extend battery life of mobile devices, along with the maturity of mobile cloud computing [53]. Offloading is an efficient way to overcome the resources and capabilities constraints of the mobile devices as it releases mobiles from intensive processing and increases performance of the applications, in terms of latency [54]. There are many potential benefits that offloading brings, such as energy saving, reliability improvement, performance improvement, ease for the software developers and better exploitation of contextual information [55].

Next section of this chapter describe the research objective of this work followed by contribution made to carry this research and thesis organization.

1.1 Research Objective

Response time i.e. latency and energy consumption are two primary aspects for mobile systems that must be considered when making offloading decisions. Moreover, in cloud computing, to satisfy requesting in sufficient manner, all resources should be made available. An efficient distributed resource management is so an important feature to enable scalable services to end users during high load on cloud resources [52]. Various cloud resources can be encapsulated as services to provide manufacturing services for clients on-demand and charges users as "Pay-as-you-use" manner. Thus, price metric can also play an important role in offloading task from mobile to cloud platform. Furthermore, if mobile user gets chance to pay minimum price for dumping an application to cloud server, it will give a benefit to get best service at optimal price. The objective of this research is to provide platform which will automate the decision making for resource allocation that will allow cloud providers to compete with each other so as to offer the lowest cost to the mobile user without impacting performance. To achieve this goal, following contributions explained in next section have been made in this research.

1.2 Contributions

The main contributions of the present research are:

- Comprehensive literature has been reviewed to decide framework to design a scenario for competitive platform.

- Agent based JADE framework and auctioning method have been studied to get insight for designing a model for providing optimal solution.
- Agent based auction approach have been proposed which runs an auctioning mechanism to provide offloading decisions for application. This auctioning mechanism is designed such a way that it will automate decision based on balanced energy-latency and price metrics.
- Proposed auctioning mechanism is evaluated for different scenarios to analyze its performance.

1.3 Thesis organization

The present thesis is structured mainly in five chapters organized as follows:

Chapter 2 describes characteristics of mobile cloud computing and related work. General experimental analysis has been carried out to formulate the problem statement. Existing research efforts are surveyed concerning mobile offloading on time saving, energy saving, and time and energy combined saving, resource provisioning mechanisms, frameworks for task migrations and price to be paid to cloud providers for accessing their resources.

Chapter 3 discusses technical groundworks of the framework selected for the model design.

Designing of proposed model is presented and explained in detail in Chapter 4.

A mechanism of proposed model has been described in detail in chapter 5 with its performance evaluation.

Finally, chapter 6 concludes the work with remarks on future improvements of the model.

Chapter 2

Background and Motivation

Significant research has been carried out in the area of general/mobile cloud computing systems. This chapter reviews background and related work in the area of mobile cloud computing which led to decide on problem statement for present research which motivates to further review the current literature to decide best approach to solve the research problem.

2.1 Background of Mobile Cloud Computing

According to [5], The Mobile Cloud Computing can be defined as below:

"Mobile Cloud Computing at its simplest, refers to an infrastructure where both the data storage and the data processing happen outside of the mobile device. Mobile cloud applications move the computing power and data storage away from mobile phones and into the cloud, bringing applications and mobile computing to not just smartphone users but a much broader range of mobile subscribers" [5].

Figure 2.1 presents general scenario of mobile cloud computing. Thus, MCC provides the data processing and storage services in clouds for mobile users. That means, mobile devices

do not need a powerful configuration such as memory capacity, CPU speed etc. since all the complicated computing processes will be processed in the clouds.

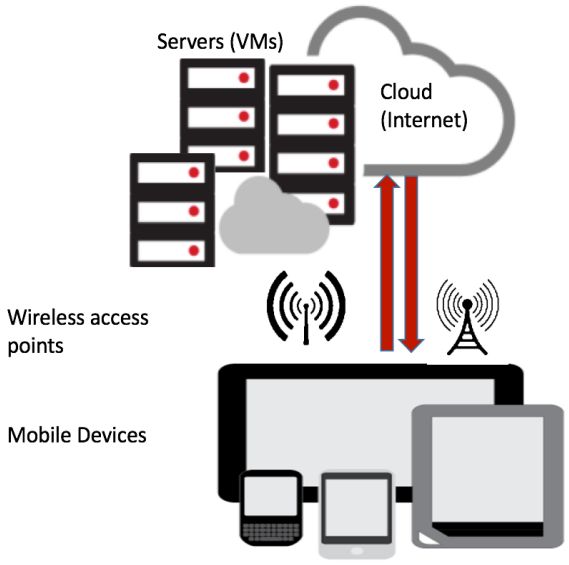

Figure 2.1 Mobile Cloud Computing Architecture

2.2.1 Advantages of MCC

The main advantage of mobile cloud computing is that it improves capacity of data storage and processing power of mobile devices. MCC is established to facilitate mobile users for accessing and storing large amount of data on the cloud. Amazon S3 [55] is such an example, which provides file storage service. Image Exchange is another example, which uses the large storage space in clouds for mobile users [56] and it enables mobile users to upload images to the clouds. MCC also helps to reduce the cost for compute-intensive applications that take longer execution time and large amount of energy.

The main concern for mobile devices is battery. Several solutions have been proposed to reduce power consumption, manage disk space and improve CPU performance. Many offloading techniques are proposed in recent literatures for migrating the large computations and complex processing from mobile devices to cloud servers. This reduces long application execution time on mobile devices which results in large amount of power consumption.

Furthermore, MCC can help reducing the chance of data and application lost on the mobile devices by storing them on cloud servers and backed up on several computers. It also facilitates mobile users to run their applications without advanced reservation of resources, hence, delivers dynamic provisioning.

With MCC, scalability and multitenancy can easily be achieved as cloud service providers can easily add and expand an application and service without or with little constraint on the resource usage. [5] They can share the resources and costs to support a variety of applications and large number of users.

With these advantages, there are some challenges which address some research questions in the area of MCC explained in related work presented in next section.

2.2 Related Work

Mobile Cloud Computing refers to the concept of offloading the tasks of mobile applications which will be partly offloaded to servers in cloud, which leads to the improved performance of mobile applications [1, 2]. In [3], the definition and discussion of general cloud computing systems were presented. [4] focused on cloud computing implemented on mobile network infrastructures, i.e., mobile cloud computing systems. Also, [5] presented a comprehensive survey of mobile cloud computing including the system architecture, applications, resource allocation and other issues. Survey carried out in [31, 45-48], addresses current challenges in the area of MCC. One of the key issues is Energy-latency optimization. As tasks with high computation demand should have longer execution time and consume more energy than tasks with low computation demand, it is necessary to identify how much application data can be offloaded to the cloud servers and what part of application should be processed on mobile platform. This identifies the problem statement of the present research which is described in following section.

2.3 Problem Statement

As the main concept of MCC is using powerful computing means to enhance capabilities of mobile devices and provide better user experiences, an effective approach is needed which is task migrations. Decision making is an important characteristic of migrations which affects the feasibility and effectiveness of task migrations. Thus, it is important that how to offload an application to cloud with energy-latency optimization.

To recognize the importance of task migration i.e. why offloading is important in MCC, an experimental analysis has been carried out as a starting point of this research. The main aim of this experiment is to find the answer of whether it is feasible to offload whole process on cloud? If it is not, then how to decide that how much of process can be offloaded to cloud server to achieve balanced energy-latency.

The experiment led to measurement of energy consumption with end-to-end delay metric, which is different from the schemes presented in recent literatures for complex tasks i.e. Image retrieval, voice recognition, gaming, graphic generation, navigation etc. which are typical tasks, that take more processing time as well as consume more energy. Thus, measurement of energy with consideration of delay metric gives optimized performance.

Generation of fractal images is one of the of typical tasks that consumes more energy and takes more processing time for generation of images. Fractal image is "a curve or geometric figure, each part of which has the same statistical character as the whole. Fractals are useful in modeling structures in which similar patterns recur at progressively smaller scales, and in describing partly random or chaotic phenomena such as crystal growth, fluid turbulence, and galaxy formation" [57].

Mandelbrot set is famous examples of rendering fractals in mathematics, which are normally generated by initializing complex numbers and iterating them numerously. Hence, the choice of Mandelbrot set is justified as a complex task for this experiment as it gets more complex when number of iterations increases. It consumes more energy and

takes more processing time with increase of number of iterations such as other typical applications. Additionally, it is easy to handle by limiting number of iterations. Thus, it is easy to test the application for all scenarios and compare the results.

2.3.1 Mandelbrot Set

The Mandelbrot set is the set of complex numbers for which the function given in following equation does not diverge when iterated from z = 0.

$$\mathbf{f_c(Z) = Z^2 + c} \qquad \mathbf{(2.1)}$$

where,

c = complex numbers

Mandelbrot set images may be created by sampling the complex numbers and determining, for each sample point *c*, whether the result of iterating the above function goes to infinity. Treating the real and imaginary parts of each number *c* as image coordinates, pixels may then be colored according to how rapidly the sequence diverges, with the color 0 (black) usually used for points where the sequence does not diverge [49].

Figure 2.1 represents initial image of a Mandelbrot set zoom sequence with a continuously colored environment.

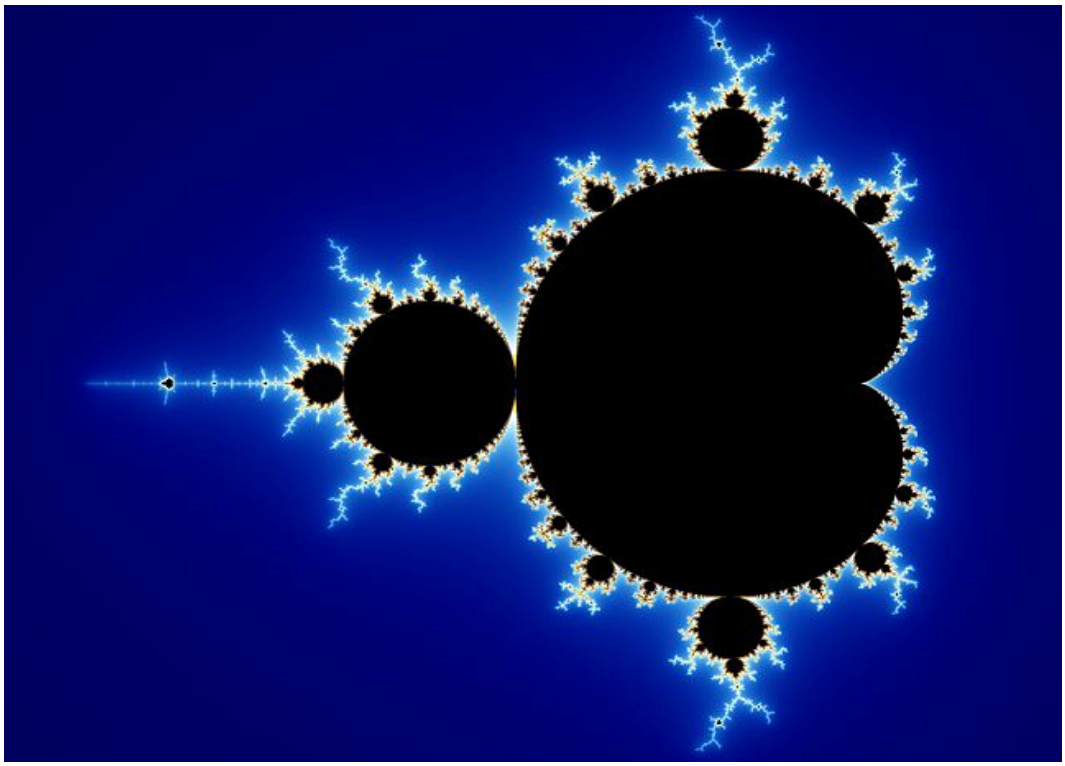

Figure 2.2 Mandelbrot Set image [49]

The Mandelbrot set has become popular outside mathematics both for its aesthetic appeal and as an example of a complex structure arising from the application of simple rules. It is one of the best-known examples of mathematical visualization. [49]

2.3.2 Experimental Setup

Experimental setup to measure energy consumption with end-to-end delay metric to generate Mandelbrot set image is shown in figure 2.3. As shown in figure, there are three systems i.e. ck102a, ck102c and ck103a, which are Ubuntu 14.04 servers and an android phone. Android phone presents mobile platform and ck103a system acts as a cloud server from remote platform. System ck102c acts as a power measuring unit with android phone

attached to it. System ck102a acts as a power measuring unit for cloud server ck103a attached to it. LabJack Tool is used as a power measuring tool running through python script on ck102a and ck102c.

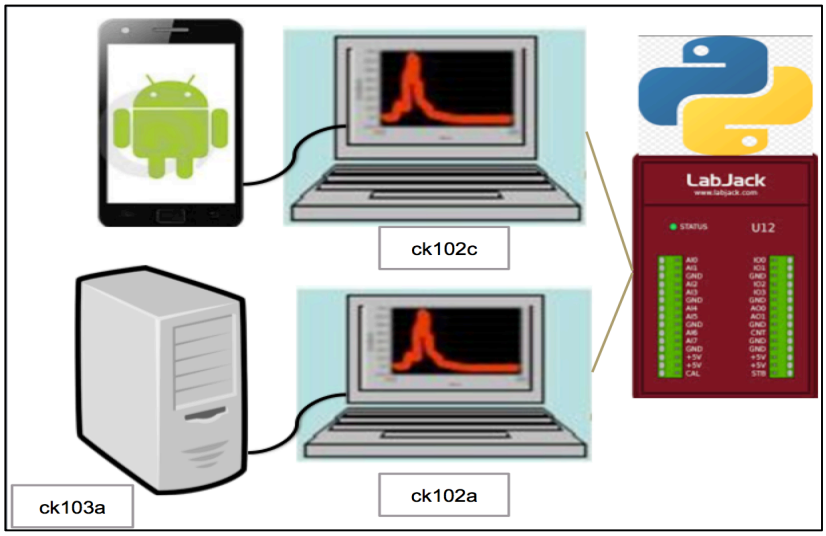

Figure 2.3: Experimental setup

The Mandelbrot set image has been generated by running python script on both platforms i.e. mobile and cloud server for different iterations ranges i.e. from 10 to 1000 iterations. Percentage increase in energy has been calculated to run the process and round trip time is measured for latency measurement for different values of N. N represents number of iterations. Experiments were run 1000 times to get optimum results. Following figures represent output graphs for mobile and cloud server. As shown in figure 2.4, cloud server performs better in both scenarios energy as well as latency.

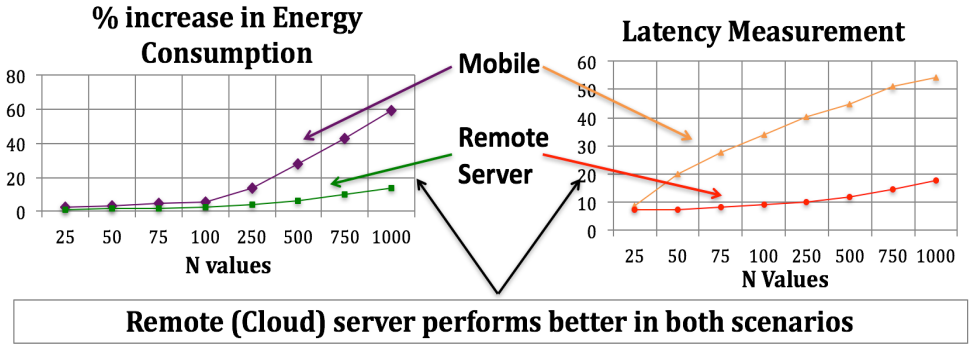

Figure 2.4: Energy and latency evaluations for mobile and cloud servers

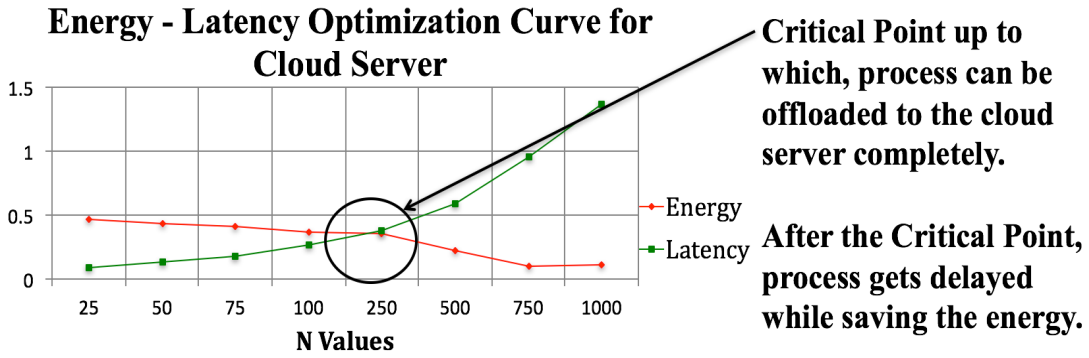

Figure 2.5: Energy and latency optimization curve for cloud server

As shown in figure 2.5, energy latency optimization curve has been created which is based on equation 2.2. This equation presents combined energy and average delay metric, which is the measure of optimum end to end delay and energy consumption that minimizes the combined metric. The reason to evaluate optimization function is to achieve critical point at which one of the metric changes its behavior as process gets complex, which can help to make decision for offloading the task to get balanced energy-delay scenario for present experiment.

$$\textbf{Optimization function} = \alpha E + \beta D \qquad (2.2)$$

Where,

E = Energy at all iterations (N) for cloud server.

D = Delay at all iterations (N) for cloud server.

N = number of iterations

α = Optimization variable. It is energy consumed per one iterations

β = end-to-end delay measured per one iterations.

Figure 2.5 shows an energy and latency optimization curve for cloud server. It is seen on the curve that, after N = 250 i.e. when Mandelbrot set equation (2.1) is iterated 250^{th} times, energy is getting decreased but process gets delayed. Thus, dumping whole process on cloud server is not a good idea as it gives increase in delay on cloud server i.e. that cannot be an optimum solution and the point at which process gets delayed while saving an energy is called critical point of the server. To balance energy – latency optimization, it is necessary to decide what percentage of application should be offloaded on cloud. This research problem has been addressed in the present work.

Furthermore, services offered from cloud are further differentiated according to types, availability and quality. Thus, sometimes single cloud cannot adequately provide solutions to mobile user's requirements, which led to increased payment for the resources usage from the cloud platform for mobile users. Therefore, the new system approach is needed in which the mobile users can use multiple clouds in an incorporated manner and get

opportunity to choose the best cloud provider that will provide best services at minimum cost to serve them. To design such a model, significant literature survey should be carried out, to further decide on framework to be used, which is presented in next section.

2.4 Literature Survey

Many researchers and scholars have done various achievements to provide resource provisioning offloading algorithms in mobile cloud computing.

In [32], a relatively comprehensive survey on task migration approaches and analysis on their solutions for decision making have been presented. They differ from each other in application migration algorithms, granularities, decision-making factors, and decision modes i.e. overall decision centric or distributed decision at a server level or task level. Research in [14], [16], [2], [18-27] provide offloading algorithms to address Energy-latency issues in MCC. [15] addresses storage capacity issue in MCC. In [17], CPU speed has been addressed for MCC. Context-aware MCC system has been proposed in [28], that takes the advantages of both nearby cloudlet, local mobile device cloud, and public cloud computing services in the remote platform to provide an adaptive and seamless mobile code offloading service.

Auction is a general and effective way to address resource allocation issue. A detailed introduction of auction theory is presented in [10]. An auction mechanism for a single auction commodity scenario was developed in [6], and proposed the concepts of individual

rationality and incentive compatibility in auction mechanism designs. [7] extended the model in [6] to a multiple commodity auction scenario. However, the auction commodities in [7] are independent with each other and indivisible. [8] discussed an auction model with two complementary auction commodities. [9] explored the auction designs for substitutable commodities. [11] proposed an auction technique to optimize cloud resource distribution in a cloud computing system. A second-price auction mechanism is proposed in [12] to allocate a single type of cloud resource, i.e., computational capacity. An analytical auction model is analyzed in [13] for the resource allocation problem in a mobile cloud computing system. The complementary and substitutable cloud resources have been considered. It has analyzed the model, and solved the optimization problem to maximize the revenue of the service provider given the proposed auction mechanism.

However, due to the dynamic behaviors of some users, the traditional cloud pricing models cannot well support such popular applications as Mobile Cloud Computing (MCC) [29]. As MCC is the combination of wireless access services and cloud services, it is feasible to apply auctions in MCC markets. A work on combinatorial and double auctions is presented in [29], and then reviewed the use of auctions to implement resource allocation and pricing in cloud and MCC markets. An auction mechanism is described in [13] with premium and discount factors for resource allocation in MCC systems. A combinatorial double auction mechanism is considered in [29] for MCC markets, which enables mobile users and MCC providers to submit bids and asks simultaneously and supports users to bid sets of commodities at one time.

According to [30], truthfulness and system efficiency are two crucial properties in addition to computational efficiency, individual rationality and budget balance. Two double auction mechanisms, TIM and EDA have been proposed in [30], which coordinate the resource trading between mobile devices as service users (buyers) and cloudlets as service providers (sellers).

Thus, based on the present literatures, it is concluded that, auctioning mechanism provides a general solution to the problem of discrete resource allocation in MCC. As described in [10], There are many types of auctions including single-good, multi-unit, combinatorial auctions, and double auctions. However, as presented in [5], the business model including pricing has to be developed for MCC. These payments for services to cloud is based on services we require for cloud to process. These services are further differentiated according to types, availability and quality. For instance, to run any particular application, one cloud provider offers less price than other cloud provider but its quality is compromised. As explained earlier in such cases, single cloud is not enough to meet mobile user's resource demands. Therefore, the new scheme is needed in which the mobile users can utilize multiple cloud in a unified fashion and get opportunity to choose the best cloud provider to serve them [5].

Hence, based on this assessment, auctioning mechanism can be applied with agent based scenario to provide more generic and practical solution. JADE can be one of the solutions for this scenario. JADE [36] is an energy aware computation offloading system for Android mobile devices. It transports computation which contains remote-able tasks from an

Android device to servers running on the cloud. At runtime, JADE gathers devices' information and tasks' information and uses multi-level scheduling algorithm to offload tasks and balance the workload of servers. JADE was evaluated with two applications (face detection and path finding), and the result showed that it can effectively reduce up to 35 % of average power consumption for mobile devices [36]. According to some of the recent research [31-34], JADE can be used as the mobile agent development platform due to its prevalence in mobile agent computing and support for multiple platforms including Android OS.

Moreover, Previous works have been successfully shown various technologies to offload computation capabilities to the cloud but, security concern is the limiting factor in these researches. Security measures were solely left for the cloud service provider, depending on the Service Level Agreement (SLA) of the provider and the client. As explained in [33], By using the java based JADE technology, mobile agents can be used which are powerful elements to improve security of distributed processing of data. JADE create platforms that have containers to act as independent virtual machines which can be java virtual machine or Dalvik virtual machine for execution of mobile agent's code. JADE allows the mobile agents to have capabilities of communicating with other agents in remotely located containers. Another capability of mobile agents is to migrate to remote locations. They can be equipped with intelligent security features of detecting tampering of one's state or bundled code [37, 38]. Upon detection, mobile agents can migrate to other containers and continue execution in these locations.

2.5 Research Motivation

Following a problem statement in section 2.3, a detailed literature has been reviewed in section 2.4 for different strategies which can be helpful to create a system model that can provide balanced energy-latency-price model. To design such model, a rich decision policy with robust framework is required. By reviewing the current research works, it has been clear that the auctioning can be the best policy for offloading decisions and JADE can provide full-bodied framework with its efficient agent based scenario.

2.6 Summary

This chapter presents the contextual information of MCC which formulated the problem statement of the present research to design a model which will provide solution of offloading decision in context of energy, latency and price to be paid for the services accessed from cloud platforms. To decide the best approach to design such model, detailed review of current literature has been carried out. Based on the literature, it has been concluded that further research can be done in this area, combining auctioning mechanism with JADE, where mobile users get chance to receive best services among the cloud providers offering minimum cost including price and energy-latency optimization.

Chapter 3

Technical Preliminaries

As the main objective of this research is to design a model which identifies efficient offloading strategy to provide balanced energy-latency at optimal price to be paid to cloud providers for accessing their resources for particular application, it is necessary to understand technical aspects of the framework to be used for model design. This chapter represents the technical groundwork for agent based auction model to be proposed in the present research. It signifies importance of agent based scenario, JADE framework details and concept of auctioning strategies in following sub sections.

3.1 Agent-based Computing

A noteworthy definition of agents in presented in [44], according to which an agent can be defined as,

"*agents reside on a platform that, consistent with the presented vision, provides the agents with a proper mechanism to communicate by names, regardless of the complexity and nature of the underlying environment (i.e. operating systems, networks, etc.)*"

A detailed descripting on mobile agent has been described in [34]. A mobile agent is a software program with mobility, which can be sent out from a computer into a network and

roam among the computer nodes in the network [34]. It can be executed on those computers to finish its task on behalf of its owner. When an application using mobile agents needs to request a service from a remote server, it gathers the required information and passes it to the agent's execution environment [34]. Mobile agents provide several advantages over other distributed computing models such as the client- server model, especially in the case of mobile clients.

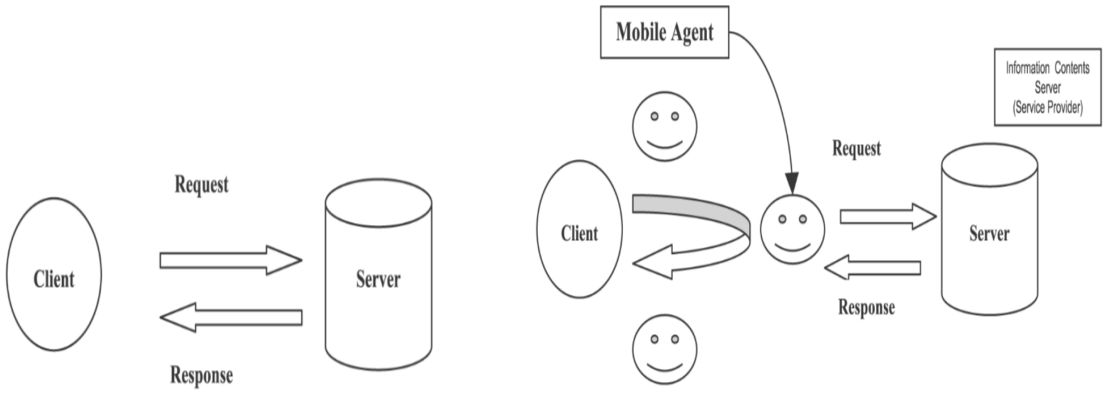

Figure 3.1(a) Client Server Communication, (b) Mobile agent Communication [34]

Figure 3.1 demonstrates the difference between the mobile agent paradigm and the client-server paradigm in terms of the communication involved upon the request of a client to a data server. As seen in the figure, while the client has to maintain a communication link with the service provider for the transfer of data in the client-server architecture, a mobile agent is shipped to the service provider and returns to the client only when it has collected all the requested data in the mobile agent paradigm [34].

As explained in [34], agent based computing can offer certain advantages. Agents can provide better support for mobile clients. As explained in [34], mobile devices are intermittently connected to a network, especially in the case of roaming in areas with low GSM/CDMA coverage. Even when the cellular signal is strong, data communication costs set a limit to users' willingness to use applications involving continuous transfer of data from/to a remote server. Due to the limited processing capacities of mobile devices, applications running on them need to offload computationally intensive operations to more powerful servers. This makes mobile agents, which are based on asynchronous interaction with servers, ideal for mobile client systems. Other advantage of agents is that it facilitates real-time interaction with server. In the case of high network latency, executing a program on a server which provides resources the program needs access to, will be faster than transferring information over the communication link with that server. This makes agent computing better fit for satisfying the real-time requirements of applications than traditional approaches that maintain constant communication between the client and the server [34]. One of the biggest advantage of agent based framework is that, mobile agent computing offers greater reliability than the client-server paradigm due to two main reasons: asynchronous communication provides reliable transport between the client and the server without requiring reliable communication, and the mobile agent is capable of dealing with a server's unavailability to provide service (in which case it would be routed to a different server with the required service) [34]. Thus, Agent-based queries/transactions can be more robust. Furthermore, agent-based transactions avoid the need to preserve process state.

3.2 JADE Agent Development Framework

JADE is an enabling technology that provides a middleware for the development and run-time execution of peer-to-peer applications. It is an agent platform which is FIPA compliant. “The JADE platform provides naming, object serialization and migration services to mobile apps. The JADE agent interface includes notification method calls that are used to convey start and completion of migration”. [31]

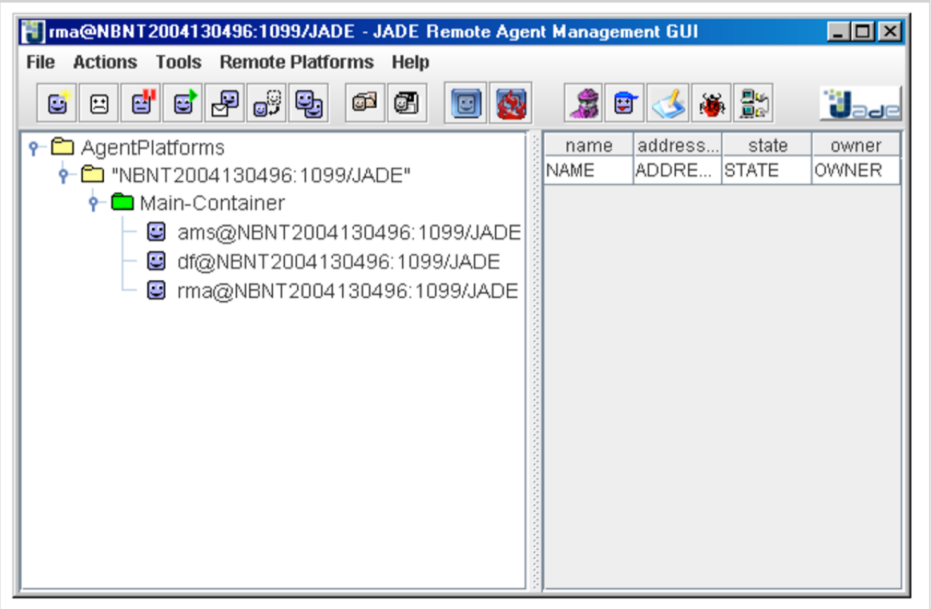

Figure 3.2 JADE Management Console

JADE architecture includes multiple containers which host and execute agents. It consists of at least one “Main Container”. All consequent containers are registered with the main container. A directory called “Global Agent Descriptor Table” of all active agents within the same platform resides in the main container. Each container has its own directory which contains information about agents running on that particular container known as “Local

Agent Descriptor Table". Figure 3.2 represents JADE management console [39]. The Main container contains three by default Agents i.e. DF (Default Facilitator), AMS (Agent Management Service) and RMA (Remote Management Agent). RMA is implements JADE Management console itself.

According to [15], "DF is a platform- wide "yellow pages" service through which all agents can publish their services and lookup services that they require. AMS is a special agent with a role of main authority within an agent platform. It is responsible for all other agents within the platform." IMTP (Internal Message Transport Protocol) provides communication between containers. it is implemented on top of Java RMI (Remote Method Invocation) mechanism. MTP (Message Transport Protocol) provides communication between platforms [40]. There are variety of other protocols on top of them MTP can reside, such as IIOP (Inter-ORB Protocol) and HTTP (Hypertext Transfer Protocol). "Messages passed over MTP are FIPA ACL compliant messages" [41].

3.3 Auctioning Strategy

Auctions are traditional and well-studied mechanisms in all aspects of technologies in recent days, the popularity of internet auctions has prompted wide interest in various aspects of auctions and related mechanisms, including the question of optimizing the total revenue of an auction [41]. Auctioning constitutes a market-driven scheme for the allocation of cloud-based computing capacities [35]. According to recent survey [38], "Auctions and biding have been used as methods for allocating and pouring goods and

services.” “There is a variety of methods of auctions— single-dimensional, multi-dimensional, single-sided, double- sided, first-price, second-price, English, Dutch, Japanese, sealed-bid—and these have been extensively discussed and analyzed in the economics literature, which shown in figure below” [10].

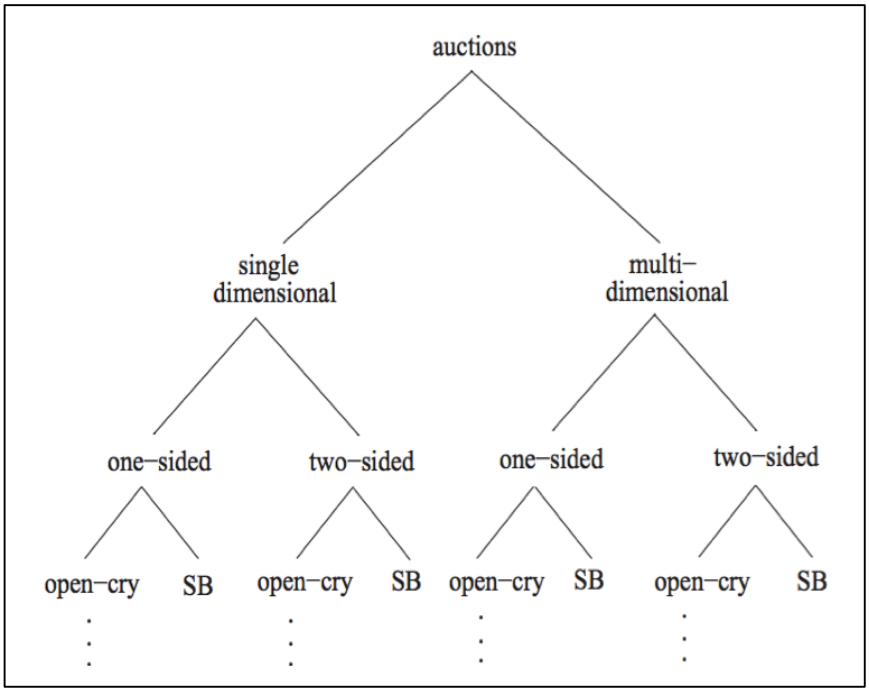

Figure 3.3 Auction Characterization [10]

In [10], all kinds of auctions are described in particular manner. The simplest auction is single dimensional auction. In this kinds of auction, the price offered for a good is the only aspect of a bid. Multi-dimensional auction includes aspects such as quality and delivery date are also distinguished with price offered for the good. Another kind of auction is based on participants of the auctions. In one-sided auctioning, bidding is either buyers or sellers—typically buyers—and the auctioneer has the job of deciding which is the winning bid. The auctioning which allows both buyers and sellers to submit bids and the job of the auctioneer

is to match buyers to sellers is referred as two-sided auction.

There are some auctions which allow every bidder to access every other bid, they are called as open-cry auctions, whereas sealed-bid (SB) auction allows only the auctioneer to access to the bids. In first-price auction, the winning bidder pays the price of the winning bid and k^{th}-price auction allows the winning bidder to pay the price of the bid that is ranked kth.

Some auctions allow to bid for only single good and other allows to bid for several units together. The key auctioning mechanism is combinatorial auction where, multiple, heterogeneous goods are auctioned simultaneously and bidders may bid for arbitrary combinations of goods. This kind of auction allows bidders to express their preferences not just for particular goods but for sets or bundles of goods.

The auctioning type, which has been considered for the present research is combination of (i) Multi-dimensional which includes energy, delay and price metrics of application to be processed (ii) Two-sided as Auction as it allows both mobile and cloud providers to submit bids and the job of the auctioneer is to match buyers to sellers and (iii) Sealed-bid (SB) Auction as it allows only the auctioneer to access to the bids. Thus auctioning scheme presented in this research is called as Multi-dimensional, two sided, Sealed Bid (i.e. MTSB) Auction.

3.4 Summary

This chapter reviewed technical details of JADE framework and auctioning methodologies. Hence, it is concluded that with agent based auction model, resources of sellers i.e. cloud servers can efficiently be distributed to buyers i.e. mobile users in a market at competitive prices. Thus, choice of agent based framework to design an approach is justified. As main container can act as an auction manager, which can have information of all agents reside in other containers. These containers can act as different platforms i.e. mobile platform and cloud platform which are unaware of each other's environments but they can keep track of their individual environment and interact with main container. The model design has been proposed in next chapter.

Chapter 4

Agent Based Auction Model

The agent based auction model has been proposed in this research to provide efficient process offloading with respect to energy and processing delay at effectively low price for MCC with the help of agent based scenario using JADE where decision will be taken based on auctioning.

4.1 Agent based Auction Model Design

Figure represents the agent based auction model design which is based on JADE framework.

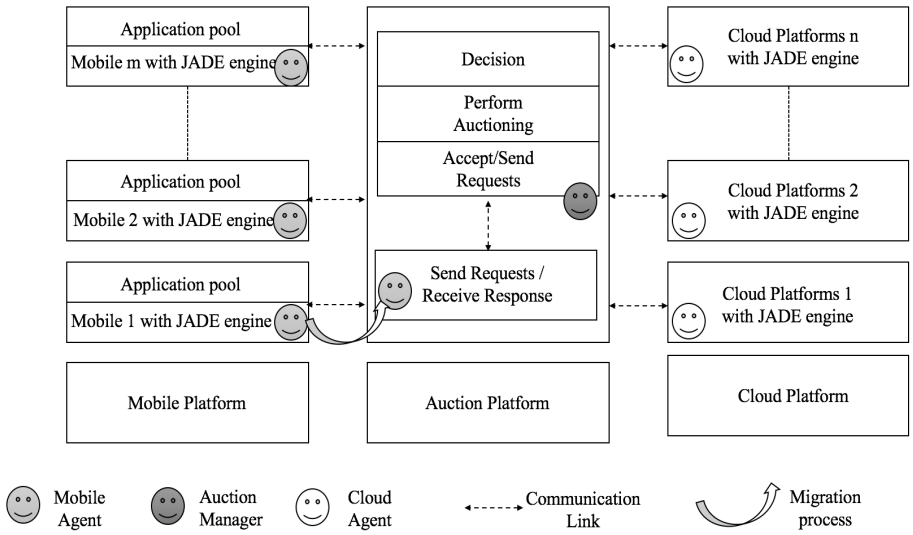

Figure 4.1 Agent based auction model design for mobile cloud computing

As shown in Figure 4.1 an open market of auctioning is presented, in which there are multiple cloud providers (e.g., Windows Azure, Amazon EC2, and Google AppEngine), who offer cost to provide services. This cost is function of energy consumed and average delay to process an application on cloud servers and price to be paid to cloud provider for using their resources. An auctioning manager is introduced here, who can determine the best offloading strategy for a given application at optimal price based on demands of application pools from mobile users.

An auctioning market is modeled as Multi-dimensional, two sided, Sealed Bid (i.e. MTSB) Auction, in which there are n cloud providers, m mobile users and a trustworthy auctioning manager. The auctioning manager starts an auction when mobile user submits requests with cost metric to start bidding and the cloud providers submit their sealed costs and bids to the auctioning market, respectively, so that no participant knows the cost/bids of any others. The auctioning manager makes the decision on offloading and payments/charges for the participants.

As shown in figure, the model comprised of three platforms i.e. mobile platform, auction platform and cloud platform. Auction platform provides platform for agents from all other platforms to communicate with each other. Auctioning platform is trustworthy entity aware of environment and it provides access to all cloud providers for mobile agent from mobile platform.

JADE engine runs on all platforms. JADE engine handles computation offloading

automatically in the background with the help of mobile agent. It allows application developers to focus on the application without implementing a separate computation offloading system. On the cloud platform, servers also contain a JADE runtime engine, which is platform independent, it can run in any virtual machines on the cloud. JADE programming model provides APIs in order for mobile applications to interact with the JADE runtime engine. Each JADE engine contains three components.

1. Communication link: It is responsible for maintaining connection among mobile users, auction market and cloud providers.

2. Profilers: There are two profilers. Device profiler collects information pertaining to device status, such as wireless connection, CPU usage, and battery level and program profiler collects information about applications, such as execution time, energy consumption, memory usage, and data size.

3. Optimizer: optimizes offloading strategy in order to maximize application's performance with auctioning mechanism.

Thus, using above three components, agents from each platform communicate with each other and processes the tasks assigned to them. Each agent has its own behavior to complete its job. They use JADE's behavior class for this purpose.

4.1.1 Agent Behavior

A. Mobile Agent

Mobile agent has mobility behavior with which it can migrate from one platform to another platform. It migrates from mobile platform to auction platform to get best service at optimal price for particular application. It makes a request to auction manager, an auctioning agent resides inside an auction platform to provide application offloading decision. It waits for response from auction manager and acts according response got from auction agent i.e. if decision is to process the task remotely on cloud platform, the mobile agent will further migrate to selected cloud provider, uses its resources to perform the task, pays for the bidding cost to the cloud providers and returns to the mobile platform. It processes remaining task if task is being partially migrated and displays the results.

B. Auction Manager

This is an auctioning agent, plays a key role in auctioning market. It has decision making behavior to provide efficient offloading at optimum price. It receives request from mobile agent for particular application and requests available cloud agents residing different cloud platforms to get bidding from cloud providers to process the application. It makes decision following an MTSB auction method i.e. it follows single bidding, selecting minimum bid among all offers and makes an agreement with the selected platform. The auctioning mechanism also selects best offloading policy associated with optimal price to be paid for

the application to be processed. That means, sometimes it may happen that, partitioning an application and offload some of its part to cloud servers and process rest of the part locally will be the best decision than offloading it completely on cloud. That way, mobile can get to use minimum of its resources at minimum price. It is explained in more details in later section. It sends its decision to all agents i.e. mobile agents and cloud agents.

C. Cloud Agents

These are the agents inside all cloud platforms taking part in auctioning. They send an offer to auction agent, once they get request for particular application. They will wait for the decision from auction agent. If they get selected, they will make an agreement to provide resources for the given application at bided price.

4.2 Experimental Analysis of Model

The experiment presented here, shows how proposed model behaves in JADE environment. It shows how agents behave in particular platform and make decisions.
The scenario is simple as shown in figure below, which follows proposed agent based auctioning model.

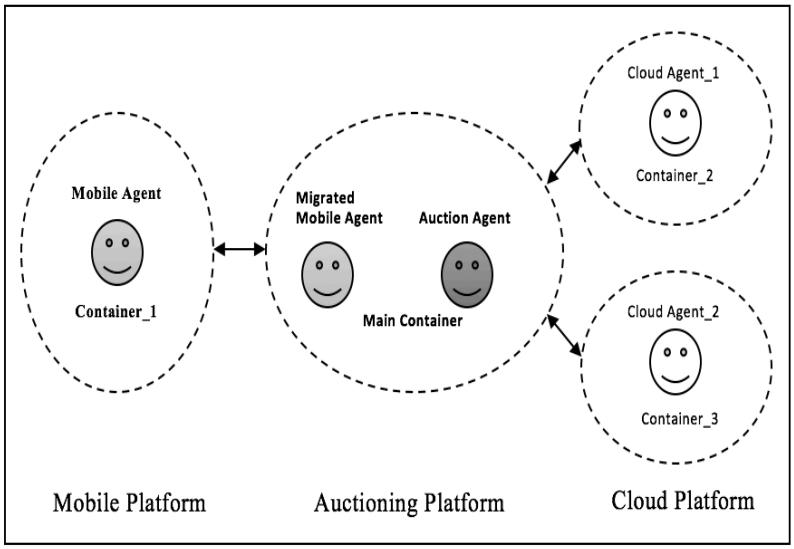

Figure 4.2 JADE based auctioning model

As shown in figure 4.2, there are three JADE containers each acting as a separate platform, connected to main container which is an auctioning market i.e. auction platform. Auction platform is aware of all platforms. All agents from different platforms can communicate with each other via auctioning platform.

In the given example, auctioning behavior is being checked in JADE environment. It is assumed here that, application will be offloaded to the cloud platform, and task is to find best bidder cloud platform. Whichever cloud provider offers less cost for the application, it will get selected by auction manager and it will make an agreement to provide services to the mobile for given application.

The scenario is, mobile agent will migrate to the auction platform and sends request to auction agent to get best bid for given application. Auction manager will get activated and

sends requests to cloud agents for offer. Cloud agents will send their bid to auction manager for particular application.

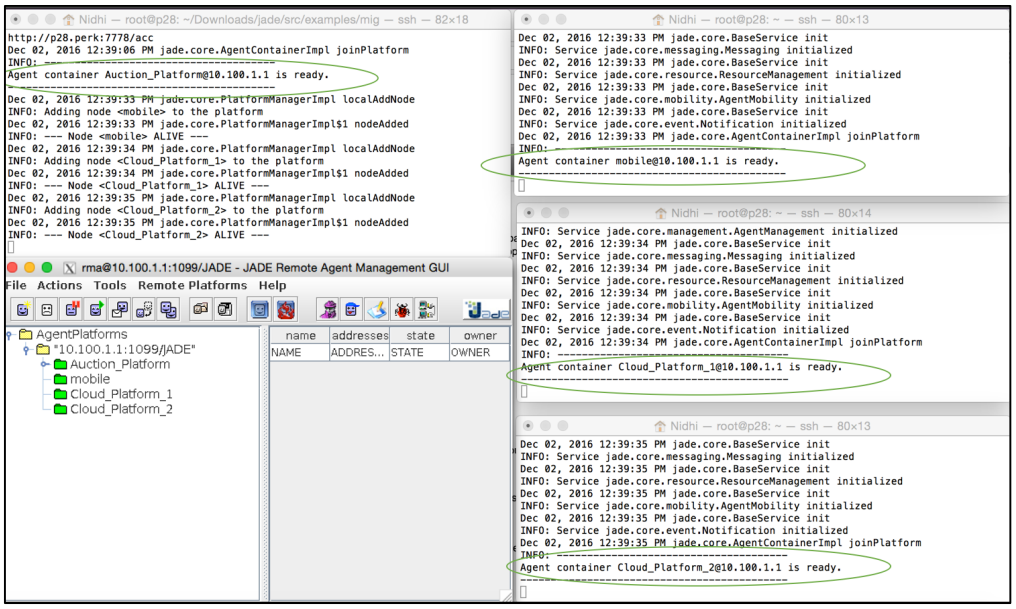

Figure 4.3 JADE management console

Auction manager will follow MTSB auction method and selects best cost received from all cloud providers. Figure 4.3 represents GUI of JADE environment. It shows all platforms are ready in management console where auction platform acts as main container.

Figure 4.4 represents all agents in their own platforms. Mobile agent will migrate with request to perform any application to auction platform as shown in figure.

Figure 4.5 represents mobile agent migration from mobile platform to auction platform. It shows that mobile agent has migrated to auction platform to take participation in

auctioning.

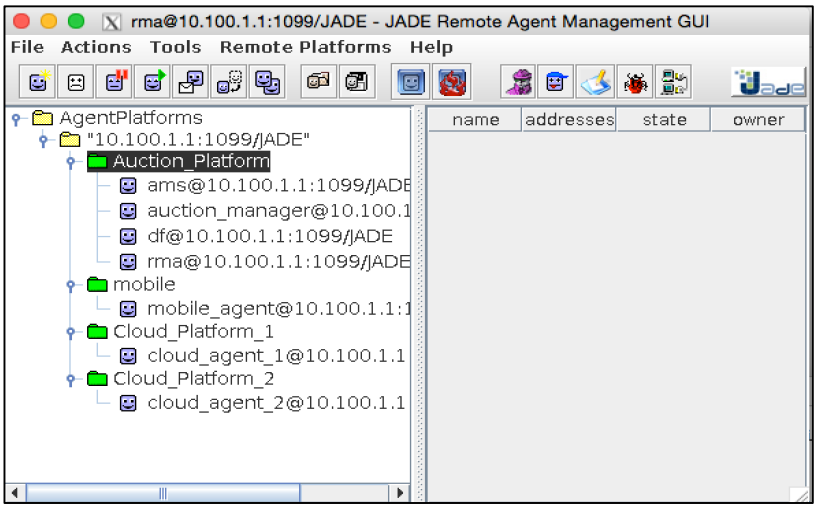

Figure 4.4 Agents residing inside their own platforms

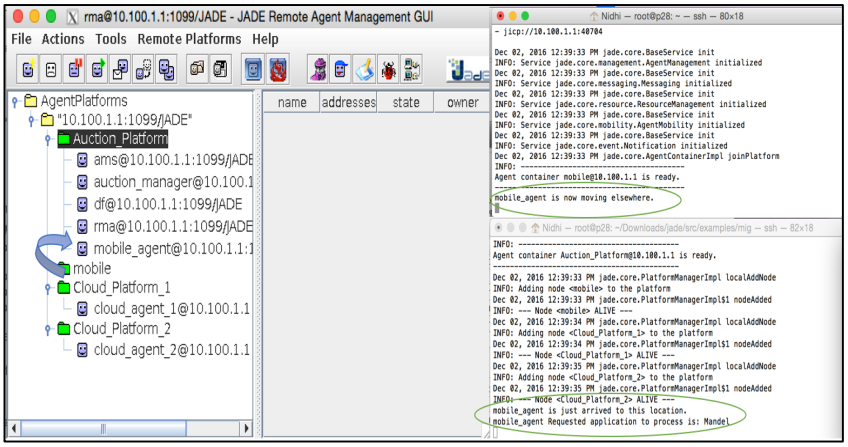

Figure 4.5 Mobile agent migration

As shown in figure, mobile agent has moved from its platform to other platform and auction platform gets updates that mobile agent has moved to its platform and requested bid for application named “Mandel”.

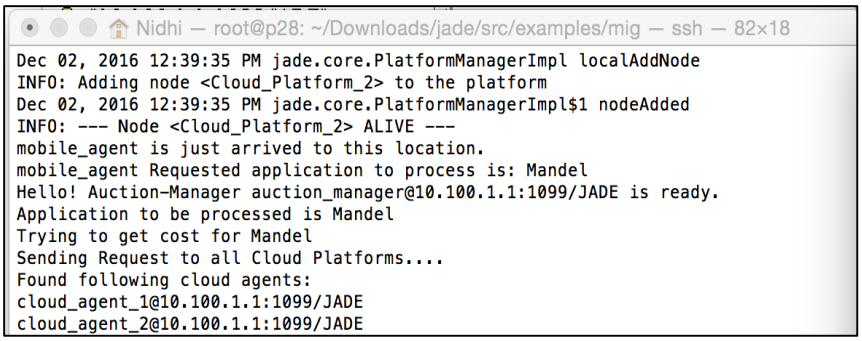

Figure 4.6 Auction manager requesting for offer from cloud agents

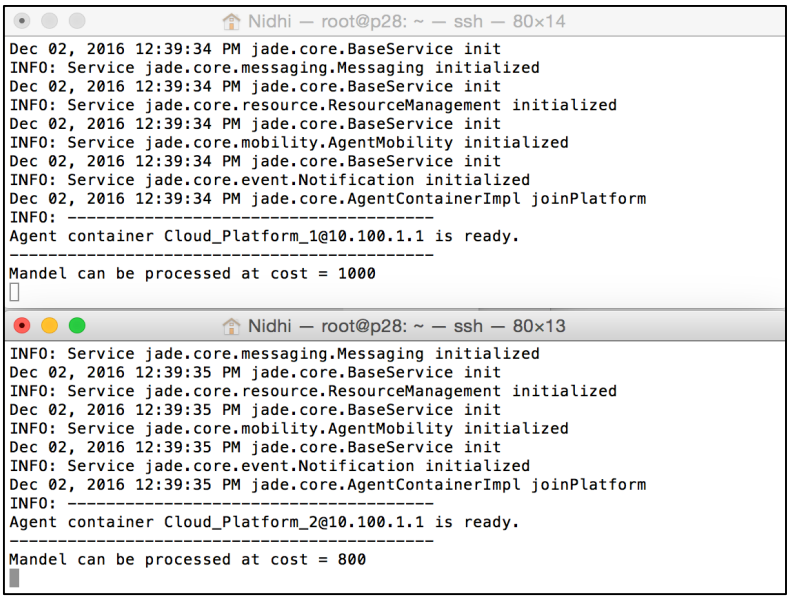

Figure 4.7 Cloud platform offering cost

Auction manager gets activated and it sends request to cloud platforms to bid for “Mandel”. As shown in Figure 4.6 auction manager finds for available clouds. Figure 4.7 shows cloud platform offering cost for application “Mandel”.

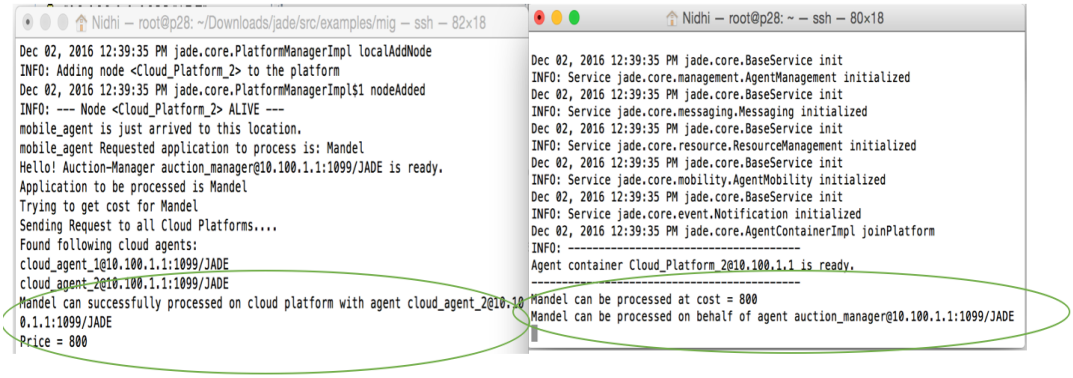

Figure 4.8 Auctioning decision

As shown in figure 4.8, auction manager finds minimum cost following MTSB auction method and displays results. When bid offered by two cloud servers matches then it selects the cloud, whose offer came first. Auction manager selects cloud provider 2 as best bidder and makes an agreement with it. It sends response to mobile agent to further migrate on cloud platform 2 and process it. Figure 4.9 shows mobile agent further migrating on cloud platform 2.

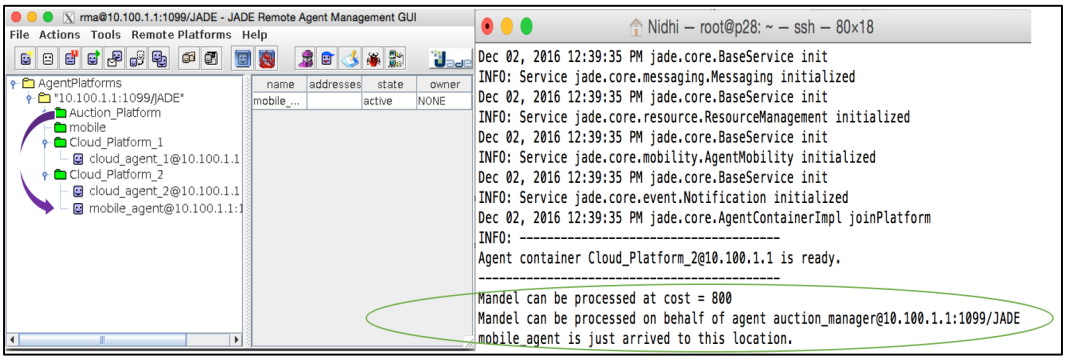

Figure 4.9 Mobile Agent migration to cloud platform 2

Thus, from this simple example, it has been analyzed that, agent based auction model can

work to identify cloud server which provides best cost value.

4.3 Summary

The chapter proposed the agent based auction model based on JADE framework. It can also be used to decide best offloading decision with help of agent based auctioning mechanism. The agent based auctioning mechanism which selects best offloading policy with optimal price is presented in next section with its implementation.

Chapter 5

Agent Based Auction Mechanism

The agent based auction mechanism proposed in this research tries to find optimum cost value of an application, which provides task offloading decision for proposed model. This offloading decision is based on whether to process the application locally i.e. on mobile platform or remotely (on cloud server) or partially on both platforms with respect to optimum cost. The optimum cost is formulated by energy, average delay and price metrics for particular application. Thus, whichever platform provides minimum cost will be selected hence, provides best offloading with minimum energy, delay and price to be paid to the cloud platform. Thus, this mechanism tries to find optimum cost which tries to get best services at best price for mobile user.

As concluded from previous chapter, JADE provides robust structure for agent based technology to develop an agent based architecture, JADE framework has been used for developing the agent based scenario in proposed mechanism. It is assumed that all platforms are running on JADE framework and they can communicate with each other through JADE. Auction mechanism proposed here is processed in all agents, i.e. mobile agents, auction manager and cloud agents according to their behaviors. Detailed steps for the mechanism are explained in next sub section.

5.1 Agent Based Auctioning Mechanism Steps

Step 1: Mobile agent gets activated when it receives any application to run. The mobile agent calculates cost metric i.e. $\mathrm{Cost_{local}}$ for particular application. The Cost metric on mobile platform is calculated as per equation 5.1:

$$\mathbf{Cost_{local} = E_l * D_l = P_l * T_l * D_l} \qquad \mathbf{(5.1)}$$

Where,

E_l = Energy required to process the Task on mobile

E_l = Power (P_l) * Processing Time (T_l) at given λ

D_l = Average Delay to process the Task at given λ

λ = requests / seconds

λ affects the cost metric on all platforms as there may be other processes going on at particular time, when application is requested to perform the particular task.

Step 2: Mobile agent migrates with $\mathrm{Request_{local}}$ (App Data, λ, $\mathrm{Cost_{local}}$) to auctioning platform to get offloading decision from auction manager.

Step 3: It waits for the response array which is decision array from auction manager agent. Response array is decision array. It contains the information regarding offloading. It contains total five fields and represented as Response (0,1,2,3,4).

Response (0) represents optimum cost value. It will be minimum cost value. Response (1) shows how task should be offloaded. It can be “M”, “C” or “P”. “M” represents mobile platform. In this case, an agent will return to mobile platform and process the task locally. “C” represents cloud platform. Agent will further be migrated to cloud platform and processes the task on cloud platform for particular application, returns to mobile platform and displays the results. If this field represents “P” then mobile agent will migrate to the cloud platform first, processes the part of application and returns to the mobile platform. It processes the remaining task on mobile and terminates after completion.

Response (2) presents percentage value of the task to be offloaded. If it is 1, then task will 100% be offloaded to the platform mentioned in Response (1) field. If percentage field ≠ 1, then process will be offloaded partially with percentage value mentioned in the field. The value represents the percentage of task to be processed on mobile and rest will be offloaded to cloud platform. Response (3) gives address of the platform i.e. where process to be offloaded. If the field is 2, then process will be offloaded to cloud platform 2. If mobile platform is selected to offload the process completely, then this field will be 0. Finally Response (4) shows value of λ for which application was requested to be processed.

Step 4: Auction manager will be activated at auction platform, when it receives $Request_{local}$ (App Data, λ, $Cost_{local}$) from mobile agent and search for the available cloud platforms. It is assumed that, Auction Manager is aware of nearby Cloud Platforms and can communicate with cloud agents residing inside all platforms.

Step 5: Auction Manager will send $Request_{remote}$(App Data, λ) to all available cloud agents and waits for Offer($Cost_{remote,}$ λ)from them. Here, auction manager will not forward $Cost_{local}$ received from mobile to cloud agents as the auction is sealed bid i.e. none of participant will know the cost of other participant. Meanwhile it calculates $Cost_{transfer}$ for applications to be transferred from mobile to all cloud platforms.

$$\mathbf{Cost_{transfer} = P_t * T_t * D_t} \quad \textbf{(5.2)}$$

Where,

E_t = Energy required to transfer the application from mobile to Cloud Server= Power (P_t) * Processing Time (T_t) at given λ

D_t = Average Delay to Transfer the application at given λ

λ = requests / seconds

Step 6: Cloud Agents will get activated when they receive $Request_{remote}$(App Data, λ) from Auction Manager. It calculates $Cost_{remote}$ at given λ and sends offer (App Data, $Cost_{remote,}$ λ) to Auction Manager and waits for the Response (). This Response is same as explained in step 3.

$$\mathbf{Cost_{remote} = C_e * E_r * D_r = C_e * P_r * T_r * D_r} \quad \textbf{(5.3)}$$

Where,

C_e = Price to be paid to process the application on cloud platform

E_r = Energy required to process the application on Cloud Server= Power (P_r) * Processing Time (T_r) at given λ

D_r = Average Delay to process the application at given λ

λ = requests / seconds

Step 7: On reception of the offers from cloud agents, Auction Manager adds $\mathrm{Cost}_{\mathrm{remote}}$ with $\mathrm{Cost}_{\mathrm{transfer}}$ values received for all cloud agents.

$$\mathbf{Cost_n = Cost_{remote} + Cost_{transfer}} \qquad \mathbf{(5.4)}$$

Where,

Cost_n= Total Cost for n^{th} cloud agent.

Step 8: Auction manager will calculate $\mathrm{Cost}_{\mathrm{Remote\_min}}$= Min ($\mathrm{Cost}_n$). Thus, one cloud agent with minimum cost will be selected and it may be predicted that given application may be offloaded completely on the cloud platform which offers $\mathrm{Cost}_{\mathrm{Remote\_min}}$.

Step 9: To get, optimum cost, auction manager generates partition array which represents values of all Partition Costs for all possibilities of percentages of application for each cloud agents. It is calculated as below:

$$\mathbf{Partition\ Costs} = \mathbf{p}\left(\mathbf{Cost_{local*p}}\right) + (\mathbf{1}-\mathbf{p})(\mathbf{Cost_{n*p}}) \qquad \mathbf{(5.5)}$$

Where,

p = 0.01 to 1 (all possible percentage values from 1% to 100%)

n = n^{th} cloud agents.

For example, for p = 0.1, it will process 10% of application on mobile platform and 1-0.1 = 0.9 i.e. 90% of application will be offloaded to n^{th} cloud agent.

$(\text{Partition Array})_n$ is collection of all the values of partition costs for particular n and values of p.

Step 10: Auction manager calculates $\text{Cost}_{\text{partition}}$ which is minimum of all $(\text{Partition Array})_n$. Thus, only one value will be selected among all the values for given λ. It will be represented as below.

$$\mathbf{Cost_{(partition)n} = (Min\ (Partition\ Array)_{\boldsymbol{n}}\ , p)} \qquad \mathbf{(5.6)}$$

In above equation, value of $\text{Cost}(0)_{(\text{partition})n}$ fireld presents minimum cost value of all (Participant Array)n and $\text{Cost}(1)_{(\text{partition})n}$ presents percentage value i.e. p% of application data will be processed on mobile and (1-p)% of data will be offloaded on cloud server.

Step 11: Auction manager calculates optimum cost for particular application on given λ as equation 5.7.

$$\mathbf{Cost_{Optimum}} = \mathbf{Min}\ (\mathbf{Cost(0)_{(partition)n}}, \mathbf{Cost_{Remote\_min}}, \mathbf{Cost_{local}}) \quad (5.7)$$

Step 12: Generate Response() based on the $\mathrm{Cost_{Optimum}}$ and Response () will be generated.

If $\mathrm{Cost_{Optimum}} = \mathrm{Cost_{Remote\_min}}$

R will be ($\mathrm{Cost_{Remote\_min}}$, C, 1, n, λ)

If $\mathrm{Cost_{Optimum}} = \mathrm{Cost_{local}}$

R will be ($\mathrm{Cost_{local}}$, M , 1 , 0, λ)

If $\mathrm{Cost_{Optimum}} = \mathrm{Cost(0)_{(partition)n}}$

R will be [$\mathrm{Cost(0)_{(partition)n}}$, P, $\mathrm{Cost(1)_{(partition)n}}$, n, λ]

Step 13: Auction manager sends response array to mobile agent as well as cloud agents to inform them about decision and terminates. It gets again activated once it receives request from mobile.

The mechanism proposed here can give optimum solution for offloading of application and selects optimal price for resource being used from cloud platform. To measure effectiveness of the mechanism, it is implemented and its performance is being evaluated in next section.

5.2 Agent Based Auctioning Mechanism Implementation

Testbed is shown in figure 5.1. There are two Ubuntu 14.04 servers i.e. ck102a and ck102b and an android mobile represented in the testbed. Mobile is connected to system ck102a, which acts as an Auction manager. System ck102b is playing as cloud provider connected with ck102a system. LabJack power tool has been used to measure power.

The Generation of Mandelbrot Set plot has been taken as a critical task for this scenario which has been used for experiment earlier in section 2. Choice of this application has been justified for this experiment, as it is easy to handle and gets high energy consumptions with more processing delay as number of iterations to generate Mandelbrot Set image increses.

The Mandelbrot set image is generated for n = 100, 250 and 1000 for given experiment, which is simulated 10,000 times to get optimum results for each value of λ i.e. number of requests per seconds.

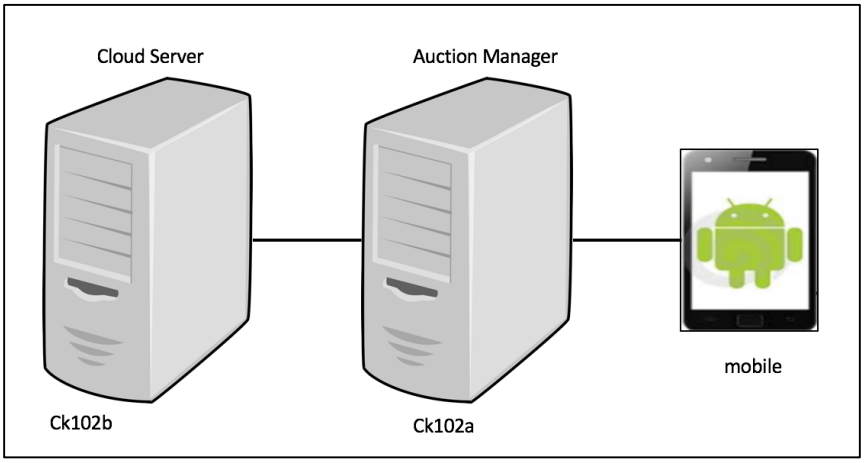

Figure 5.1 Testbed

Energy values have been calculated by measuring power and time taken to execute the application for different values of λ. λ plays an important role in measuring average delay to process the application on any of the system. These requests are simulated in Poisson's distribution i.e. the interval between requests is in poison, to see how given application is reacting to being invoked in poison's distribution manner.

Offloading decisions have been made for each values of λ on each platforms and optimum cost function has been calculated to get best offloading platform which offers best price to process the application in efficient way.

The table 5.1 represents local cost values for all values of λ. As from table it is seen that it increases with value of λ increments. These values are for N = 100 in given table. N is number of iterations to generate Mandelbrot Set image.

Lambda	**Power(P_l)**	**Time(T_l)**	**Energy ($=P_lT_l$)**	**Avg Delay (D_l)**	**ce**	**$Cost_{local}$ = ce*(F(E_l) * D_l)**
0.5	0.935962	5.81938	5.446718544	2255.037555	1	12282.55487
1	0.937938	5.81046	5.449851231	2643.614754	1	14407.30712
1.5	0.936916	5.8094	5.44291981	2845.568923	1	15488.20346
2	0.930578	5.81411	5.410482856	2905.533363	1	15720.33845
2.5	0.930578	5.80905	5.405774131	2947.116591	1	15931.44663
3	0.937594	5.8184	5.45529693	2993.973152	1	16333.01254
3.5	0.943952	5.81228	5.486513331	3019.107184	1	16564.37181
4	0.945314	5.81761	5.49946818	3039.099782	1	16713.43255

Table 5.1 Cost values at mobile platform for N = 100

Lambda	Power(P_r)	Time(T_r)	Energy ($=P_rT_r$)	Avg Delay (D_r)	ce	$Cost_n$ = ce*($F(E_r) * D_r$) + $F(E_t)*D_t$
0.5	69.3819	0.213611	14.82073704	0.260706335	1	3.86386004
1	69.4006	0.213678	14.82938141	0.278275672	1	4.126656083
1.5	69.4102	0.213719	14.83427853	0.306769163	1	4.550699215
2	69.5377	0.213843	14.87015038	0.395357029	1	5.879018471
2.5	69.5703	0.213956	14.88498311	0.404790862	1	6.025305144
3	69.6093	0.218441	15.2055251	0.638471174	1	9.708289466
3.5	69.6854	0.222295	15.49071599	1.006386011	1	15.58963988
4	69.6966	0.222935	15.53781152	1.282754558	1	19.93119854

Table 5.2 Cost values at cloud platform for N = 100

The Table 5.2 gives $Cost_n$ for cloud server. $Cost_n$ represents combined values of Cost calculated as $Cost_{transfer}$ and $Cost_{remote}$ for n^{th} cloud provider. n = 1 for the present scenario.

Graphs represented in figure 5.2, plots cost values for different values of λ for both platforms i.e. mobile and cloud platform.

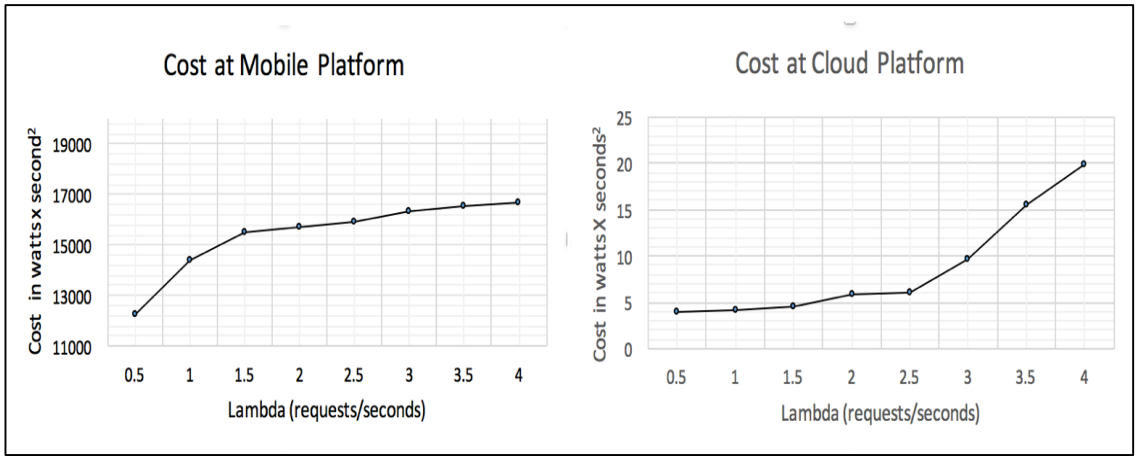

Figure 5.2 Cost values for mobile and cloud platforms

Based on the graphs, following equation can be drawn to represent behavior for cost value of mobile platform and cloud platforms. For cloud platforms equations will be changed based on the Ce values. These equations are presented here to compare the results which are achieved from simulation to measure the performance of the mechanism.

Values of Ce that is fixed price offered by the cloud provider to take part in auction. This price value is decided by cloud providers based on their operation cost and profit. Thus, it will be different for every cloud provider.

Thus, scenario for different values of Ce has been taken into account to evaluate performance of the present mechanism. The value of Ce is 1 for graph represented figure 5.2, it will be multiplied with different values for different scenarios to consider simulation scenarios.

Different scenarios have been considered for performance evaluation of proposed mechanism, Tables in following cases represent resulting optimum cost values from auctioning mechanism. These values are being compared with Cost values of mobile and cloud servers, to check performance of the Auctioning algorithm. Different scenarios have been compared.

Cost function for mobile platform

$$X = 34.555\lambda^3 - 584.23\lambda^2 + 3369.5\lambda + 9567.2 \quad (5.8)$$

Cost Function with cloud server with Ce = 100

$$Y1 = 49.99\,\lambda^2 - 229.19\,\lambda + 627.54 \quad (5.9)$$

Cost Function with cloud server with Ce = 500

$$Y2 = 249.95\,\lambda^2 - 1145.9\,\lambda + 3137.7 \quad (5.10)$$

Cost Function with cloud server with Ce = 4000

$$Y3 = 188.02\lambda^3 - 538.74\lambda^2 + 515.72\lambda + 15794 \quad (5.11)$$

Cost Function with cloud server with Ce = 4000

$$Y4 = 235.03\,\lambda^3 - 673.42\,\lambda^2 + 644.65\,\lambda + 19743 \quad (5.12)$$

Case 1: (Ce = 100)

In this case, cloud agent offers price = 100 for all values of λ. Which is fixed value it offers to auctioning manager to process Mandelbrot application for iterations = 100.

Theoretical Results

Table 5.3 (a) shows result that are calculated from equations 5.8 and 5.9 for cloud server and mobile respectively.

Lambda	Cost at Cloud Server	Cost at Mobile	Optimum Cost	Offloading Decision
0.5	12282.55487	386.386004	386.386004	completely on cloud server
1	14407.30712	412.6656083	412.6656083	completely on cloud server
1.5	15488.20346	455.0699215	455.0699215	completely on cloud server
2	15720.33845	587.9018471	505.3476927	partially to cloud server with 1.0 % on mobile
2.5	15931.44663	602.5305144	505.3476927	partially to cloud server with 1.0 % on mobile
3	16333.01254	970.8289466	505.3476927	partially to cloud server with 1.0 % on mobile
3.5	16564.37181	1558.963988	505.3476927	partially to cloud server with 1.0 % on mobile
4	16713.43255	1993.119854	505.3476927	partially to cloud server with 1.0 % on mobile

Table 5.3(a) Theoretical Output from Auctioning Mechanism for Ce = 100

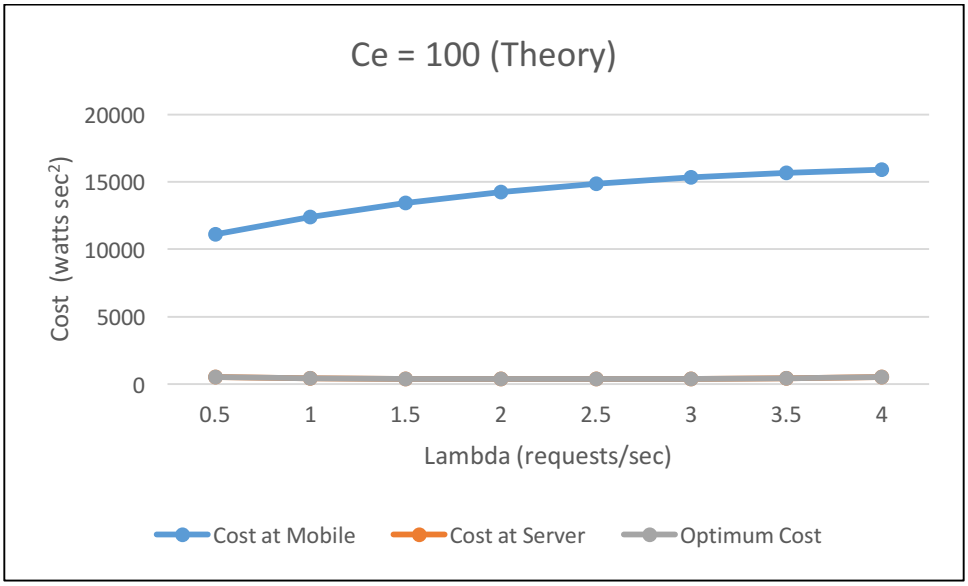

Figure 5.3(a) Theoretical Behavior Auctioning Mechanism for Ce = 100

Simulation Results:

Table 5.3 (b) represents results from simulations mentioned in Table 5.2 with Ce = 100

Lambda	Cost at Cloud Server	Cost at Mobile	Optimum Cost	Offloading Decision
0.5	12282.55487	386.386004	386.386004	completely on cloud server
1	14407.30712	412.6656083	412.6656083	completely on cloud server
1.5	15488.20346	455.0699215	455.0699215	completely on cloud server
2	15720.33845	587.9018471	505.3476927	partially to cloud server with 1.0 % on mobile
2.5	15931.44663	602.5305144	505.3476927	partially to cloud server with 1.0 % on mobile
3	16333.01254	970.8289466	505.3476927	partially to cloud server with 1.0 % on mobile
3.5	16564.37181	1558.963988	505.3476927	partially to cloud server with 1.0 % on mobile
4	16713.43255	1993.119854	505.3476927	partially to cloud server with 1.0 % on mobile

Table 5.3(b) Simulation Output from Auctioning Mechanism for Ce = 100

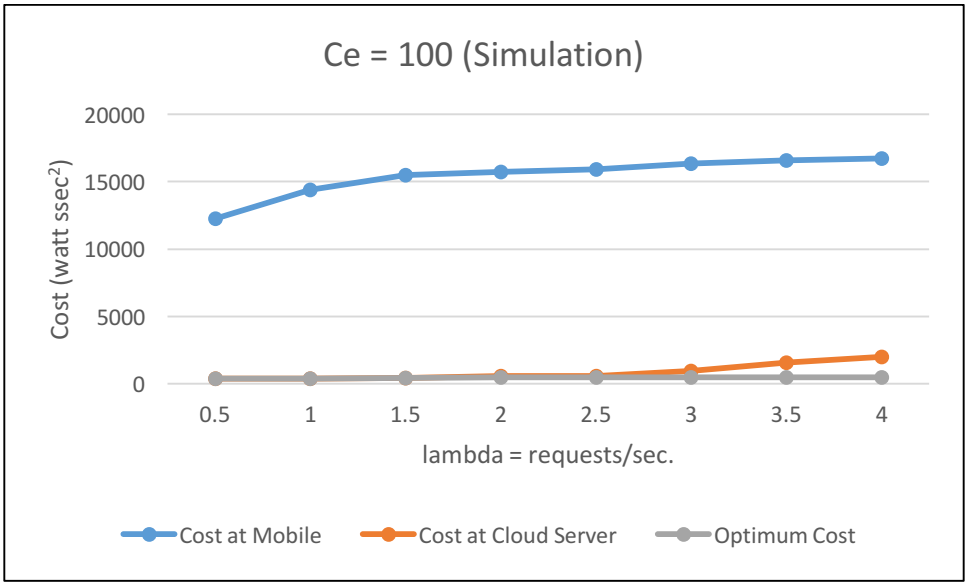

Figure 5.3(b) Simulation Behavior Auctioning Mechanism for Ce = 100

Result Analysis

Above graphs in figures 5.3(a) and (b), represent behaviors of auctioning mechanism for Ce = 100 for both theory and simulation results. It is seen that both graphs present approximately same results with decision of most of the offloading on cloud server as cloud server provides approximately 96% & 94% average reduction in cost than cost of mobile in both theory and simulations results respectively. Results of theory approach are differed with 8% than simulation results.

Case 2: (Ce = 500)

In this case, cloud agent offers price = 500 for all values of λ. Which is fixed value it offers to auctioning manager to process Mandelbrot application for iterations = 100.

Theoretical Results

Table 5.4 (a) shows result that are calculated from equations 5.8 and 5.10 for cloud server and mobile respectively.

Lambda	Cost at Cloud Server	Cost at Mobile	Optimum Cost	Offloading Decision
0.5	11110.21188	2627.2375	2627.2375	completely on cloud server
1	12387.025	2241.75	2241.75	completely on cloud server
1.5	13423.55563	1981.2375	1981.2375	completely on cloud server
2	14245.72	1845.7	1845.7	completely on cloud server
2.5	14879.43438	1835.1375	1835.1375	completely on cloud server
3	15350.615	1949.55	1949.55	completely on cloud server
3.5	15685.17813	2188.9375	2188.9375	completely on cloud server
4	15909.04	2553.3	2553.3	completely on cloud server

Table 5.4(a) Theoretical Output from Auctioning Mechanism for Ce = 500

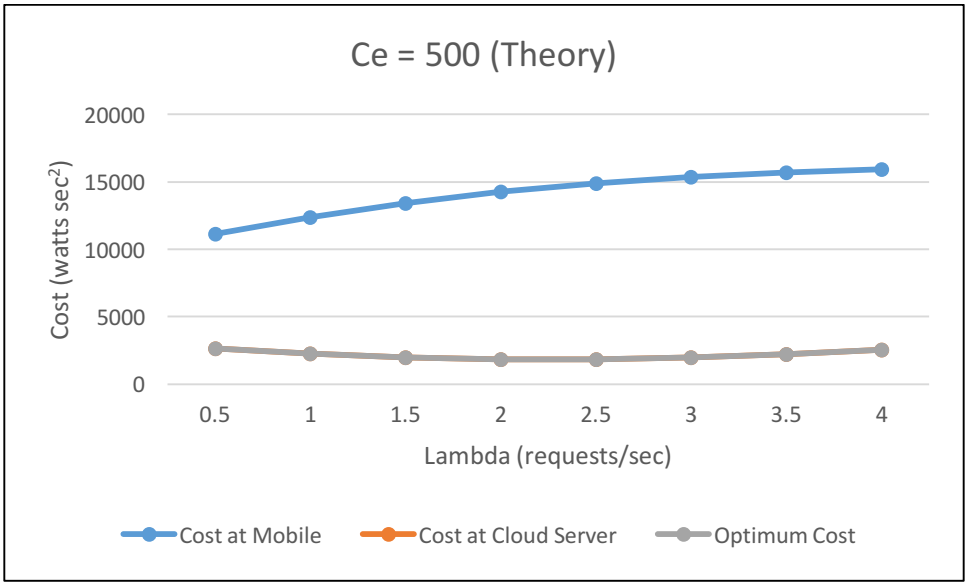

Figure 5.4(a) Theoretical Behavior of Auctioning Mechanism (Ce = 500)

Simulation Results

Table 5.4 (b) represents results from simulations mentioned in Table 5.2 with Ce = 500

Lambda	Cost at Cloud Server	Cost at Mobile	Optimum Cost	Offloading Decision
0.5	12282.55487	1931.93002	1931.93002	completely on cloud server
1	14407.30712	2063.328041	2035.436269	partially to cloud server with 1.0 % on mobile
1.5	15488.20346	2275.349608	2035.436269	partially to cloud server with 1.0 % on mobile
2	15720.33845	2939.509236	2035.436269	partially to cloud server with 1.0 % on mobile
2.5	15931.44663	3012.652572	2035.436269	partially to cloud server with 1.0 % on mobile
3	16333.01254	4854.144733	2035.436269	partially to cloud server with 1.0 % on mobile
3.5	16564.37181	7794.819941	2035.436269	partially to cloud server with 1.0 % on mobile
4	16713.43255	9965.599272	2035.436269	partially to cloud server with 1.0 % on mobile

Table 5.4(b) Theoretical Output from Auctioning Mechanism for Ce = 500

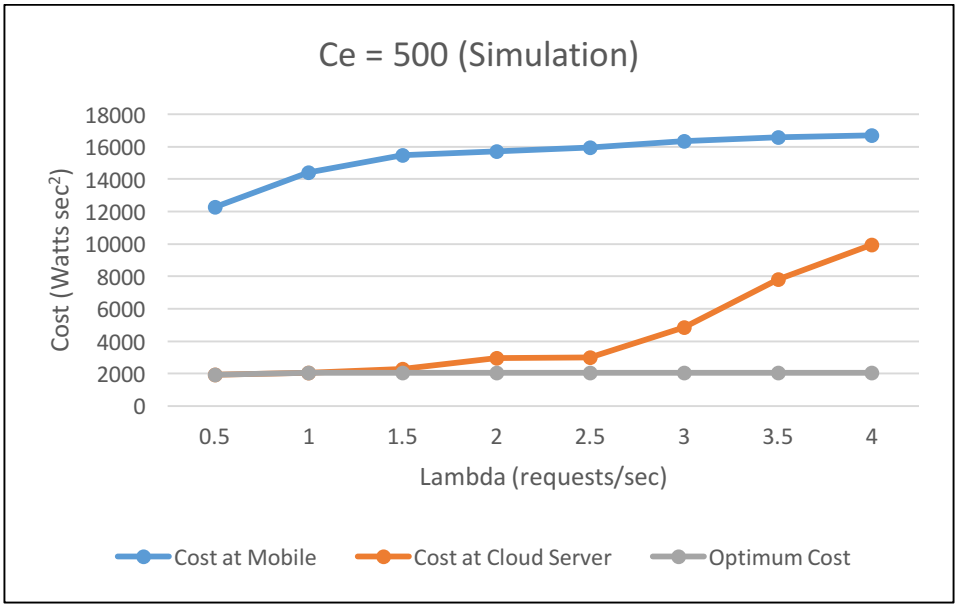

Figure 5.4(b) Simulation Behavior of Auctioning Mechanism (Ce = 500)

Result Analysis

Above graphs in figures 5.4(a) and (b), represent behaviors of auctioning mechanism for Ce = 500 for both theory and simulation results. It is seen that both graphs present approximately same results with decision of most of the offloading on cloud server as cloud server provides approximately 78% & 82% average reduction in cost than cost of mobile in both theory and simulations results respectively. Results of theory approach are differed with 6% than simulation results.

Case 3: (Ce = 4000)

In this case, cloud agent offers price = 4000 for all values of λ. Which is fixed value it offers to auctioning manager to process Mandelbrot application for iterations = 100.

Theoretical Results

Table 5.5 (a) shows result that are calculated from equations 5.8 and 5.11 for cloud server and mobile respectively.

Lambda	Cost at Cloud Server	Cost at Mobile	Optimum Cost	Offloading Decision
0.5	11110.21188	15940.6775	11110.21188	completely on mobile
1	12387.025	15959	12387.025	completely on mobile
1.5	13423.55563	15989.9825	13423.55563	completely on mobile
2	14245.72	16174.64	14245.72	completely on mobile
2.5	14879.43438	16653.9875	14879.43438	completely on mobile
3	15350.615	17569.04	15350.615	completely on mobile
3.5	15685.17813	19060.8125	15685.17813	completely on mobile
4	15909.04	21270.32	15909.04	completely on mobile

Table 5.5(a) Theoretical Output from Auctioning Mechanism for Ce = 4000

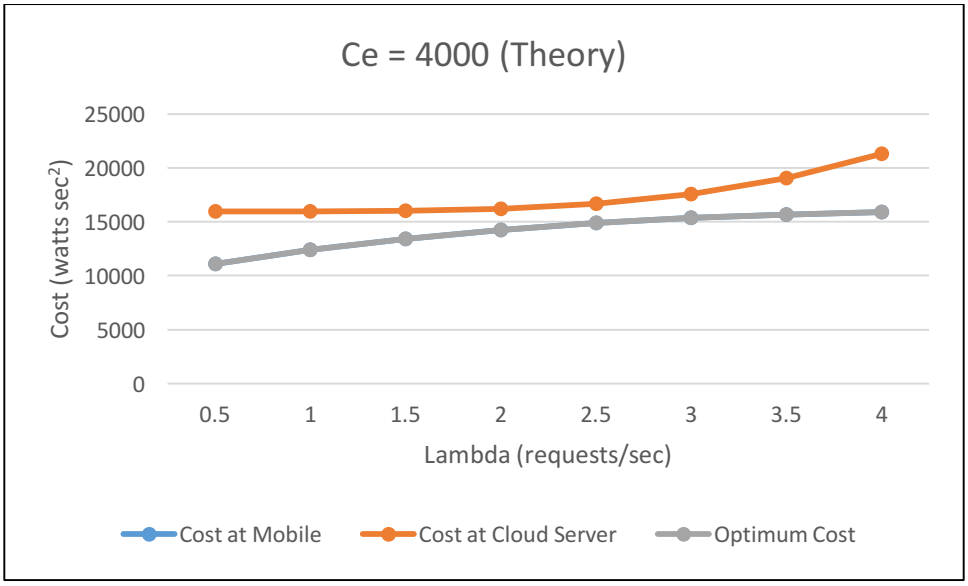

Figure 5.5(a) Theoretical Behavior of Auctioning Mechanism (Ce = 4000)

Simulation Results

Table 5.5 (b) represents results from simulations mentioned in Table 5.2 with Ce = 4000

Lambda	Cost at Cloud Server	Cost at Mobile	Optimum Cost	Offloading Decision
0.5	12282.55487	15455.44016	12282.55487	completely on mobile
1	14407.30712	16506.62433	12314.28372	partially to cloud server with 99.0 % on mobile
1.5	15488.20346	18202.79686	12314.28372	partially to cloud server with 99.0 % on mobile
2	15720.33845	23516.07389	12314.28372	partially to cloud server with 99.0 % on mobile
2.5	15931.44663	24101.22058	12314.28372	partially to cloud server with 99.0 % on mobile
3	16333.01254	38833.15787	12314.28372	partially to cloud server with 99.0 % on mobile
3.5	16564.37181	62358.55953	12314.28372	partially to cloud server with 99.0 % on mobile
4	16713.43255	79724.79418	12314.28372	partially to cloud server with 99.0 % on mobile

Figure 5.5(b) Simulation Behavior of Auctioning Mechanism (Ce = 4000)

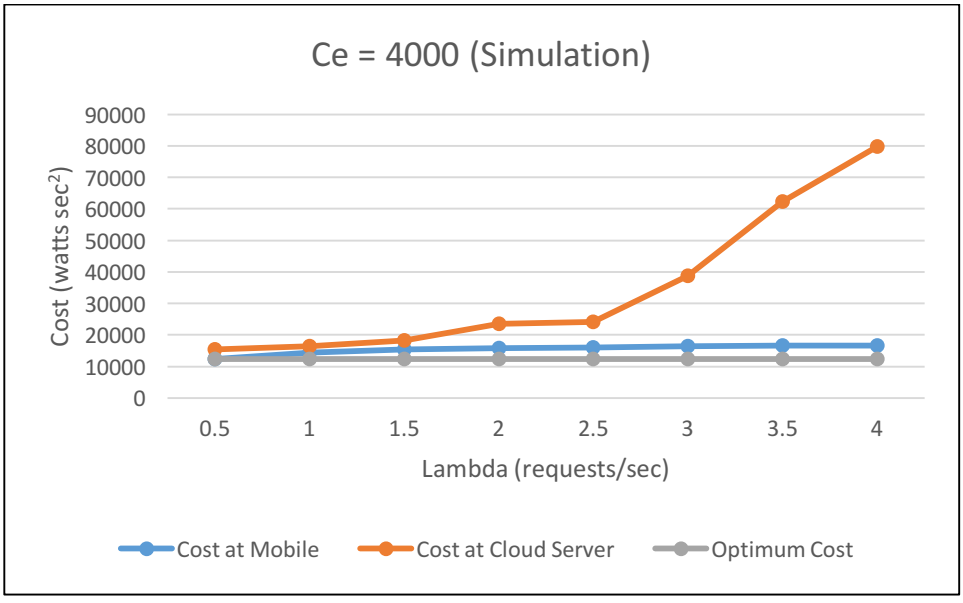

Figure 5.5(b) Simulation Behavior of Auctioning Mechanism (Ce = 4000)

Result Analysis

Above graphs in figures 5.5(a) and (b), represent behaviors of auctioning mechanism for Ce = 4000 for both theory and simulation results. It is seen that both graphs present approximately same results with decision of most of the offloading on mobile as it provides approximately 20% & 40% average reduction in cost than cost of mobile server in both theory and simulations results respectively. Results of theory approach are differed with 10% than simulation results.

Case 4: (Ce = 5000)

In this case, cloud agent offers price = 5000 for all values of λ. Which is fixed value it offers to auctioning manager to process Mandelbrot application for iterations = 100.

Theoretical Results

Table 5.6 (a) shows result that are calculated from equations 5.8 and 5.12 for cloud server and mobile respectively.

Lambda	Cost at Cloud Server	Cost at Mobile	Optimum Cost	Offloading Decision
0.5	11110.21188	19926.34875	11110.21188	completely on mobile
1	12387.025	19949.26	12387.025	completely on mobile
1.5	13423.55563	19988.00625	13423.55563	completely on mobile
2	14245.72	20218.86	14245.72	completely on mobile
2.5	14879.43438	20818.09375	14879.43438	completely on mobile
3	15350.615	21961.98	15350.615	completely on mobile
3.5	15685.17813	23826.79125	15685.17813	completely on mobile
4	15909.04	26588.8	15909.04	completely on mobile

Table 5.6(a) Theoretical Output from Auctioning Mechanism for Ce = 5000

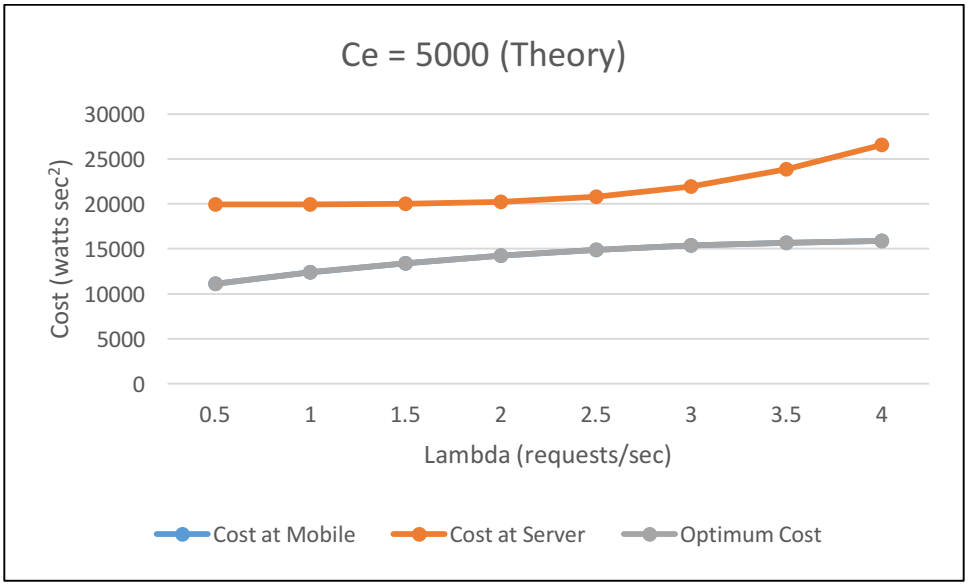

Figure 5.6(a) Theoretical Behavior of Auctioning Mechanism (Ce = 5000)

Simulation Results

Table 5.6 (b) represents results from simulations mentioned in Table 5.2 with Ce = 5000

Lambda	Cost at Cloud Server	Cost at Mobile	Optimum Cost	Offloading Decision
0.5	12282.55487	19319.3002	12282.55487	completely on mobile
1	14407.30712	20633.28041	12352.92232	partially to cloud server with 99.0 % on mobile
1.5	15488.20346	22753.49608	12352.92232	partially to cloud server with 99.0 % on mobile
2	15720.33845	29395.09236	12352.92232	partially to cloud server with 99.0 % on mobile
2.5	15931.44663	30126.52572	12352.92232	partially to cloud server with 99.0 % on mobile
3	16333.01254	48541.44733	12352.92232	partially to cloud server with 99.0 % on mobile
3.5	16564.37181	77948.19941	12352.92232	partially to cloud server with 99.0 % on mobile
4	16713.43255	99655.99272	12352.92232	partially to cloud server with 99.0 % on mobile

Table 5.6(b) Simulation Output from Auctioning Mechanism for Ce = 5000

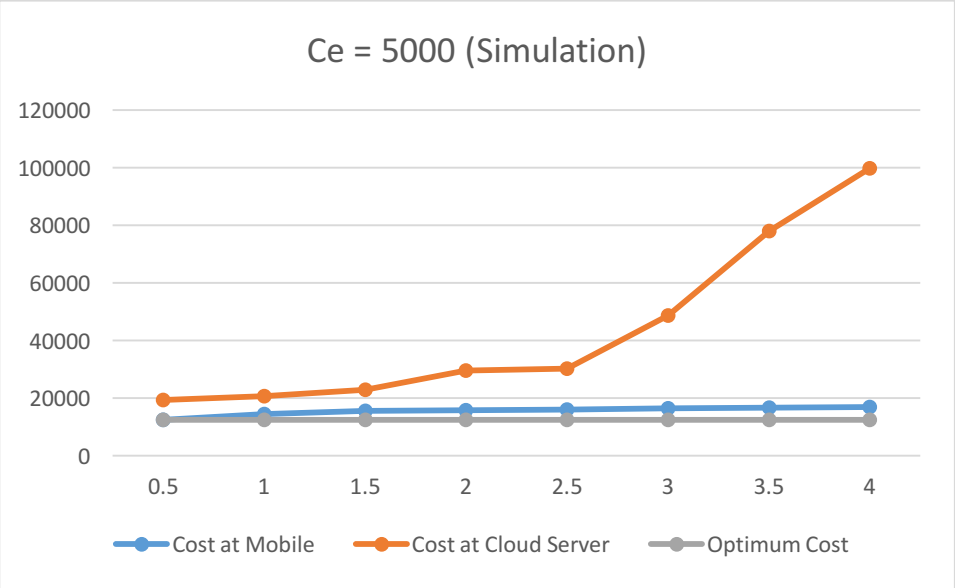

Figure 5.6(b) Simulation Behavior of Auctioning Mechanism (Ce = 5000)

Results analysis

Above graphs in figures 5.6(a) and (b), represent behaviors of auctioning mechanism for Ce = 5000 for both theory and simulation results. It is seen that both graphs present approximately same results with decision of most of the offloading on mobile as it provides approximately 38% & 50% average reduction in cost than cost of mobile server in both theory and simulations results respectively. Results of theory approach are differed with 12% than simulation results.

Case 5: Multi Server Scenario

In this scenario, cloud servers are competing with each other to provide services for application requested by mobile. Auctioning mechanism will identify which platform would be appropriate for processing the application with both the scenarios.

Theoretical Results

Table 5.7 (a) shows result that are calculated from following cost equations for cloud servers:

Cost equation 5.9 is used for Cloud Server 1 with ce = 100

Cost equation 5.10 is used for Cloud Server 2 with ce = 500

Cost equation 5.11 is used for Cloud Server 3 with ce = 4000

Cost equation 5.12 is for Cloud Server 4 with ce = 5000

Lambda	Cost at server1	Cost at server2	Cost at server3	Cost at server4	Cost at mobile	Optimum Cost	Offloading Decision
0.5	525.4425	2627.2375	15940.6775	19926.34875	11110.211875	525.4425	Completely on cloud server 1
1	448.34	2241.75	15959.0	19949.26	12387.025	448.34	Completely on cloud server 1
1.5	396.2325	1981.2375	15989.9825	19988.00625	13423.555625	396.2325	Completely on cloud server 1
2	369.12	1845.7	16174.64	20218.86	14245.72	369.12	Completely on cloud server 1
2.5	367.0025	1835.1375	16653.9875	20818.09375	14879.434375	367.0025	Completely on cloud server 1
3	389.88	1949.55	17569.04	21961.98	15350.615	389.88	Completely on cloud server 1
3.5	437.7525	2188.9375	19060.8125	23826.79125	15685.178125	437.7525	Completely on cloud server 1
4	510.62	2553.3	21270.32	26588.8	15909.04	510.62	Completely on cloud server 1

Figure 5.7(a) Theoretical Output from Auctioning Mechanism (multi-server)

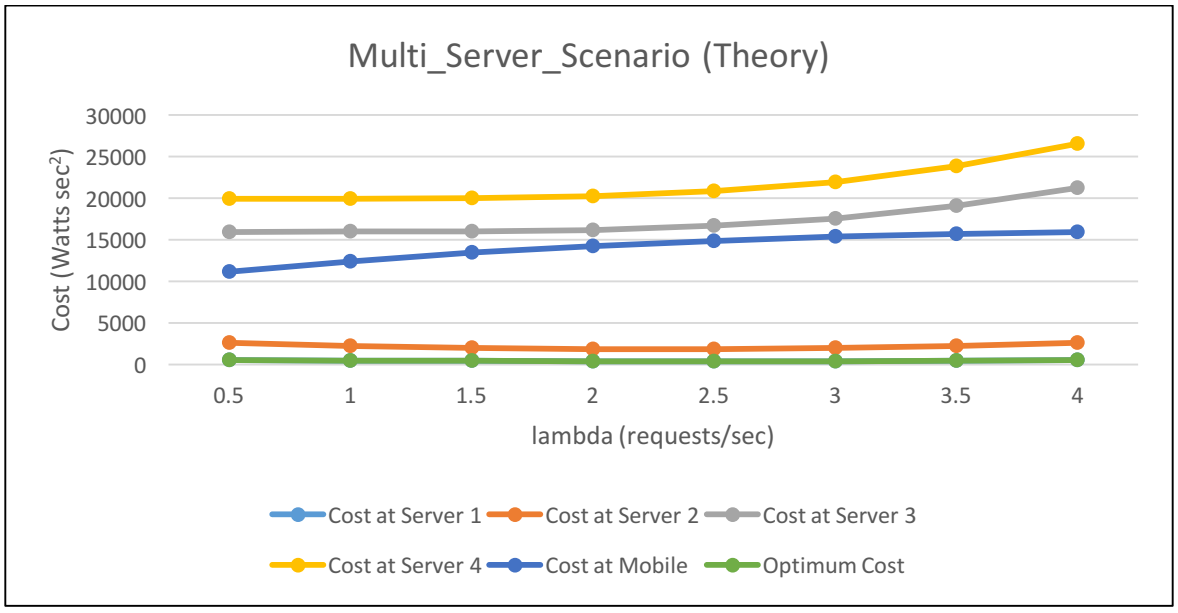

Figure 5.7 (a) Theoretical Behavior of Auctioning Mechanism (multi-server)

Simulation Results

Lambda	Cost at server1	Cost at server2	Cost at server3	Cost at server4	Cost at mobile	Optimum Cost	Offloading Decision
0.5	386.386004	1931.93002	15455.44016	19319.3002	12282.55487	505.34769266	Partially on cloud server 1 with 1.0 % on mobile
1	412.6656083	2063.328041	16506.62433	20633.28041	14407.30712	552.612023417	Partially on cloud server 1 with 1.0 % on mobile
1.5	455.0699215	2275.349608	18202.79686	22753.49608	15488.20346	605.401256885	Partially on cloud server 1 with 1.0 % on mobile
2	587.9018471	2939.509236	23516.07389	29395.09236	15720.33845	739.226213129	Partially on cloud server 1 with 1.0 % on mobile
2.5	602.5305144	3012.652572	24101.22058	30126.52572	15931.44663	755.819675556	Partially on cloud server 1 with 1.0 % on mobile
3	970.8289466	4854.144733	38833.15787	48541.44733	16333.01254	1124.4507825 3	Partially on cloud server 1 with 1.0 % on mobile
3.5	1558.963988	7794.819941	62358.55953	77948.19941	16564.37181	1709.0180662 2	Partially on cloud server 1 with 1.0 % on mobile
4	1993.119854	9965.599272	79724.79418	99655.99272	16713.43255	2140.3229809 6	Partially on cloud server 1 with 1.0 % on mobile

Figure 5.7(b) Simulation Output from Auctioning Mechanism (multi-server)

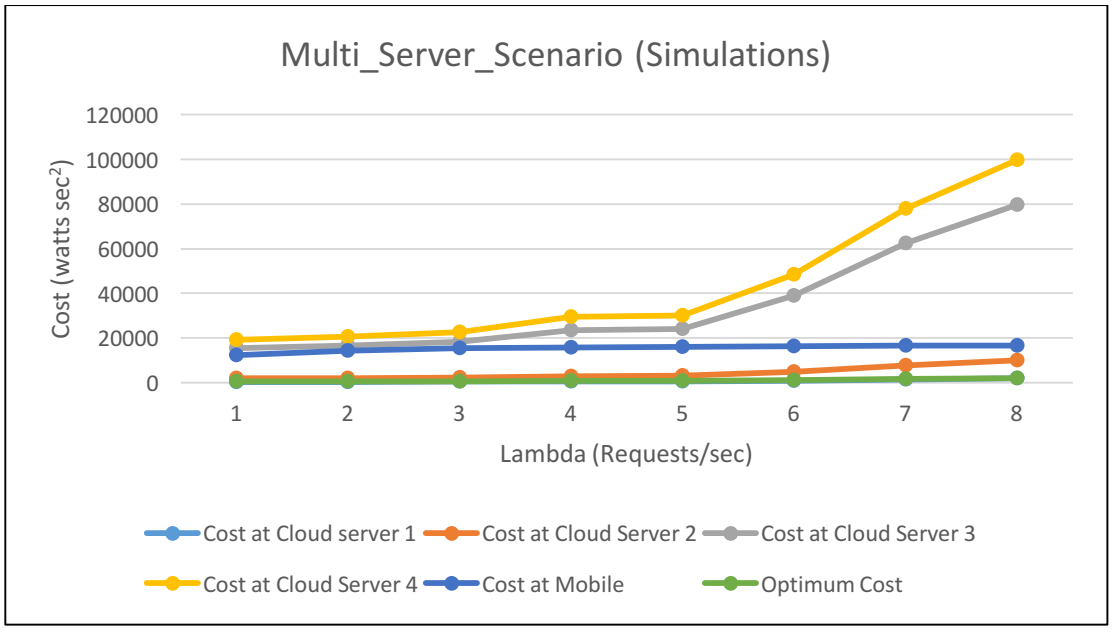

Figure 5.7 (b) Simulation Behavior of Auctioning Mechanism (multi-server)

Result Analysis

Graphs in figure 5.7 (a) and (b) represent behavior of auctioning mechanism for multi-server scenario for both theoretical and simulations approaches. It selects best platform i.e. cloud server 1 for Mandelbrot application to be processed in both cases. Optimum cost theoretically gives approximately 80%, 97%, 98% and 96% of average reduction in cost and 80%, 97%, 98% and 95% by simulation w.r.t. cost offered by server 2, 3, 4 and cost calculated at mobile respectively. The difference between simulation and theoretical approaches here is just 3%.

Case 6: Ce = 500, N = 250

Table 5.8, shows simulation results of cost values for iterations = 250, which was calculated as critical point of the cloud server for all values of λ. In this case, cloud agent offers price = 500 for all values of λ. Which is fixed value it offers to auctioning manager.

Lambda	Cost at Cloud Server	Cost at Mobile	Optimum Cost	Offloading Decision
0.5	23893.44596	11110.21188	11110.21188	Mobile
1	40402.37973	37982.0304	9741.594464	3.0 % on mobile
1.5	45002.6717	46610.7506	8751.211063	Cloud Server
2	64583.7149	57131.8373	9829.279029	2.0 % on mobile
2.5	67455.5395	63356.2017	11245.55235	11.0 % on mobile
3	71179.3879	66541.1990	12248.23735	16.0 % on mobile
3.5	75061.0732	68146.3872	12775.60311	17.0 % on mobile
4	77639.5919	69454.2309	13096.57589	17.0 % on mobile

Figure 5.8 Output from Auctioning Mechanism (Ce = 500, N = 250)

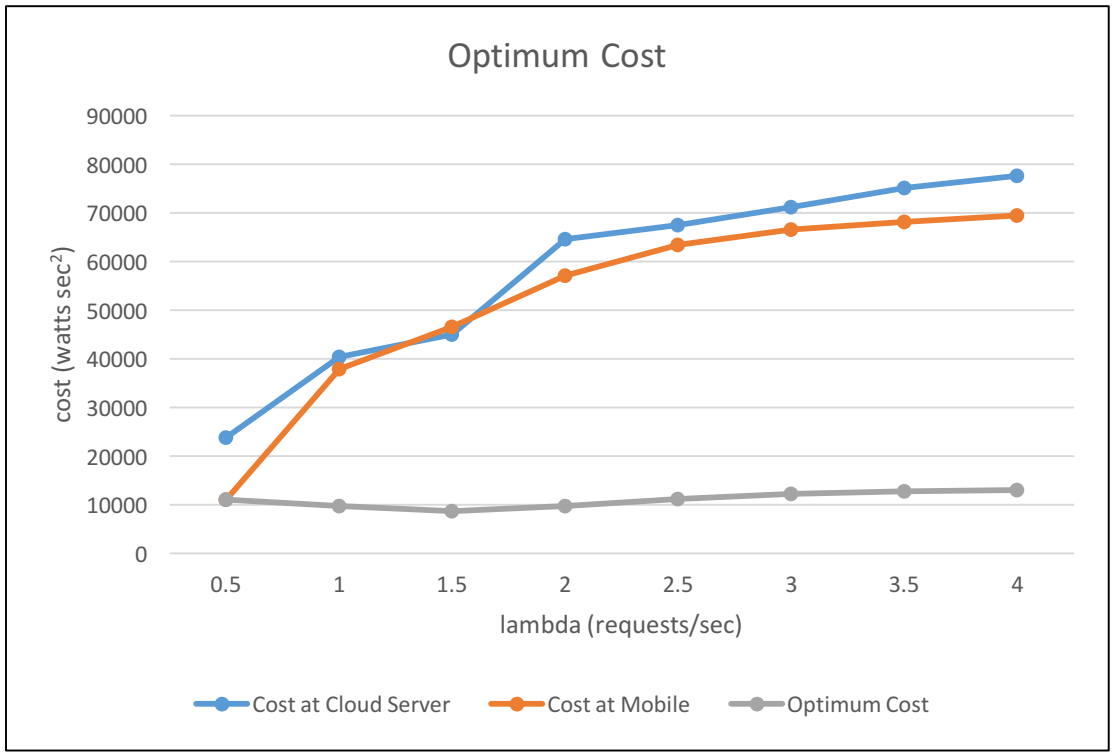

Table 5.8 Behavior of Auctioning Mechanism (Ce = 500, N = 250)

Result Analysis:

Above graph in figure 5.8, represents behavior of auctioning mechanism for Ce = 500 for N = 250. The cost values for cloud server, gives values i.e. in the range from 11.000 to 70,000 watts-second^2 gives approximately 12% reduction in average cost offered by cloud servers i.e. in the range of 23000-78000 watts- second^2 . For $\lambda = 0.5$ optimum cost is selected to be mobile's cost i.e. 100% task processing on mobile platform. For $\lambda = 1.5$, 100% task processing is offloaded on cloud platform since it provides approximately 3% of cost reduction w.r.t. cost value calculated at mobile. Since, the present algorithm chooses minimum of all cost values, it goes for partitioning of application on mobile with different percentage of task migrations, for all values of λ except for $\lambda = 0.5$ and 1.5. which further gives approximately 78% average reduction in cost w.r.t. cost offered by cloud server.

Case 7: Ce = 500, N = 1000

Table 5.9, shows simulation results of cost values for iterations = 250, which was calculated as critical point of the cloud server for all values of λ. In this case, cloud agent offers price = 500 for all values of λ. Which is fixed value it offers to auctioning manager.

Lambda	Cost at Cloud Server	Cost at Mobile	Optimum Cost	Offloading Decision
0.5	14105.59291	11110.21188	11110.21188	Mobile
1	15583.273	12387.025	12387.025	Mobile
1.5	16117.61372	13423.55563	13144.77643	80.0 % on mobile
2	17765.536	14245.72	13423.09621	64.0 % on mobile
2.5	18084.70703	14879.43438	13567.0514	53.0 % on mobile
3	19861.939	15350.615	13656.65443	45.0 % on mobile
3.5	20841.58784	15685.17813	13718.19307	39.0 % on mobile
4	21453.952	15909.04	13762.29934	34.0 % on mobile

Table 5.9 Output from Auctioning Mechanism (Ce = 500, N = 1000)

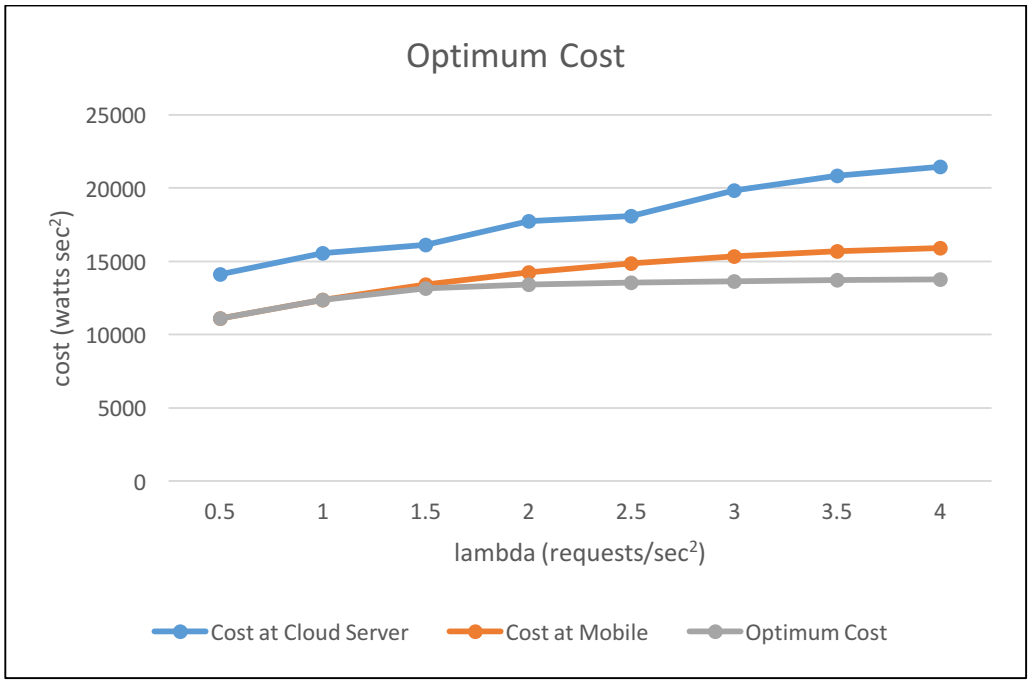

Figure 5.9 Behavior of Auctioning Mechanism (Ce = 500, N =1000)

Result Analysis

Above graph in figure 5.12, represents behavior of auctioning mechanism for Ce = 500 and N = 1000. Cloud server offers higher cost than cost calculated at mobile which is giving approximate average of 22% reduction in average cost for all values of λ, hence optimum value is chosen to be mobile's cost for λ = 0.5 and 1 i.e. 100% task processing is carried on mobile platform. For λ = 1.5 to 4, partitioning of application has been decided as optimum cost is approximately 9% lesser than calculated cost of mobile.

5.3 Summary

This chapter represents detailed steps for proposed auction based auctioning mechanism based on MTSB. Its performance has been evaluated with different scenarios in previous sub section. The model has been tested theoretically and with simulation results. It gives approximately same results in both scenarios. Here, in cases including N = 100 iterations, the offloading decisions are typically choosing either a cloud platform or a mobile platform for almost all price values (Ce values) w.r.t. each λ. The reason is for N = 100, Mandelbrot Set image creation is simple process, so if cloud offers lesser price i.e. Ce = 100 and 500, Optimum Cost offered by cloud server will be less and offloading decision will cloud platform and for higher prices (Ce = 4000 and 5000), it choses mobile platform for processing. Further model has been checked for different numbers of iteration values. As it was concluded from previous experiment explained in chapter 2 that at N = 250, cloud server's performance is reduced if whole process is offloaded to it, it has been justified in

case 6 in this chapter, that with partitioning decision, it gives better performance hence, optimality has been achieved. It is also tested for N = 1000, to show that, when process is getting complexed, partitioning is the best option. Thus, it is concluded that, the proposed algorithm tries to find minimum cost for all scenarios and provide optimum solution for complex tasks.

Chapter 6

Conclusion & Future work

6.1 Conclusion

Mobile cloud computing is a key computing prototype. Many researchers have carried significant research to provide a solid foundation for progress for developing core concepts, techniques, and mechanisms in this area. Task migration is the main approach that offloads applications to powerful cloud servers. Currently, there are number of approaches to decision making for task migrations. In this research detailed survey has been carried out to design a framework which further increases task offloading decisions.

To demonstrate, optimization for both platform is equally important and it plays a vital role in task offloading decision, an experiment have been performed to generate Mandelbrot Set image for different numbers of iterations to identify difference in energy consumption and latency when process gets complexed. Based on this experiment, critical point has been found on cloud server at which process gets delayed while saving energy i.e. performance of cloud server degrades. This experiment helped to design a model, which can help for decision process of task migration with optimization including price factor offered by cloud providers to provide services as it is concluded in chapter 2 that price metrics plays a significant part in decision making of choosing cloud providers.

The agent based auction approach using JADE framework has been proposed in this research, to get optimum solution of offloading decision for provision of balanced energy-latency-price to mobile users. It has been unique approach to combine agent based scenario with auctioning mechanism in area of mobile cloud computing to provide automation in efficient offloading at competitive price and gives solution regarding where and what portion of applications to offload using multi-dimensional, two sided, sealed bid auction.

Sample test model shows how simply this model identifies low cost cloud provider for required service. Performance of agent based auction model has been evaluated for different scenarios to demonstrate its decision making capabilities in all scenarios. The results of optimum cost are compared with costs offered by cloud severs, cost values provided by mobile users and costs calculated for each combinations of portioning of application. It is seen that, the proposed mechanism tries to get optimized results for all scenarios to benefit both mobile and cloud users by providing balanced solution.

6.2 Future work

The proposed agent based auction model motivates to create more robust model in future.

In proposed scheme, the time taken by auctioning manager to make a decision has not been considered which may affect overall performance when it is used in serving more than one complex tasks requested by more than one mobile users. Thus, performance of auction manager can be improved by recovering its limitation with addition of delay caused by auctioning process in cost function for decision making.

The auctioning can be made more scalable in future by considering variable price offered by cloud providers in cost metric i.e. auctioning can be made k^{th} price auction. For example, once mechanism finds optimal solution based on optimum cost, auction manager will perform periodical auction for k times and provide chance to cloud participants to change their bids overtime and decides on best solution based on their results. This costs of the cloud providers can be changed for each k^{th} time with variable price metrics.

Moreover, security can be one of the concern for this scheme. As the tendency of the mechanism is to choose minimum cost all the time, it may choose malicious cloud provider to dump the service and end up compromising the performance. Thus, policies to authenticate cloud provider should be designed for auction manager to provide security in the model in future.

Thus, the concept of agent based auctioning mechanism generalizes to many scenarios in distributed computing and can be explored further in future including Auctioning delay, variable cost and security feature to authenticate cloud user.

References:

[1] B. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Pa., "Clonecloud: elastic execution between mobile device and cloud", In EuroSys, 2011

[2] E. Cuervo, A. Balasubramanian, D. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: making smartphones last longer with code offload," In MobiSys, 2010

[3] M. Armbrust, A. Fox, R. Grith, et al., "A view of cloud computing," Communications of the ACM, vol. 53, pp. 50-58, April 2010

[4] Q. Wang, "Mobile cloud computing," Master of science thesis, University of Saskatchewan, Canada, 2011

[5] D. T. Hoang, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: Architecture, applications, and approaches," Wireless Communications and Mobile Computing, in press, DOI:10.1002/wcm.1203

[6] R. B. Myerson, "Optimal auction design," Mathematics of Operations Research, vol. 6, no. 1, pp. 58-73, February 1981

[7] F. Branco, "Multiple unit auctions of an indivisible good," Economic Theory, vol. 8, no. 1, pp. 77-101, February 1996

[8] J. Levin, "An optimal auction for complements," Games and Economic Behavior, vol. 18, no. 2, pp. 176-192, February 1997

[9] R. Burguet, "The condominium problem; auctions for substitutes," Review of Economic Design, vol. 9, no. 2, pp. 73-90, April 2005

[10] S. Parsons, J. A. Rodriguez-Aguilar, and M. Klein, "Auctions and bidding: A guide for computer scientists," ACM Computing Surveys, vol. 43, pp. 10:1-10:59, February 2011

[11] C. L. Mullins, "Real-time auction of cloud computing resources," U.S. Patent 2010/0076856 A1, 25 March 2010

[12] W.-Y. Lin, G.-Y. Lin, H.-Y. Wei, "Dynamic Auction Mechanism for Cloud Resource Allocation," in 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid), pp. 591-592, May 2010

[13] Zhang, Y., Niyato, D., & Wang, P. (2013, August). An auction mechanism for resource allocation in mobile cloud computing systems. In International

Conference on Wireless Algorithms, Systems, and Applications (pp. 76-87). Springer Berlin Heidelberg.

[14] Keke Gai a, Meikang Qiu a,n, Hui Zhao b, Lixin Tao a, Ziliang Zong c. "Dynamic energy-aware cloudlet-based mobile cloud computing model for green computing", Journal of Network and Computer Applications 59 (2016) 46–54

[15] Nazanin Aminzadeh, Zohreh Sanaei, SiE Hafizah Ab Hamid, "Mobile storage augmentation in mobile cloud computing: Taxonomy, approaches, and open issues", Simulation Modelling Practice and Theory 50 (2015) 96–108

[16] Brad K. Donohoo, Chris Ohlsen, Sudeep Pasricha, Yi Xiang, Charles Anderson, "Context-Aware Energy Enhancements for Smart Mobile Devices," IEEE Transactions On Mobile Computing, Vol. 13, No. 8, August 2014

[17] Yanchen Liu, Myung J. Lee, "An Effective Dynamic Programming Offloading Algorithm in Mobile Cloud Computing System," IEEE WCNC'14 Track 2 (MAC and Cross-Layer Design)

[18] Z. Li, C. Wang, and R. Xu, "Computation offloading to save energy on handheld devices: a partition scheme," in Proc 2001 Intl Conf. on Compilers, architecture, and synthesis for embedded systems (CASES), pp. 238-246, Nov 2001.

[19] G. Chen, B. T. Kang, M. Kandermir, N. Vijaykrishnan, M. J. Irwin, and R. Chandranouli, "Studying energy trade-offs in offloading computation/compilation in Java- enabled mobile devices," IEEE Transactions on Parallel and Distributed Systems, 15(9): 795-806, Sept 2004

[20] C. Xian, Y. H. Lu, and Z. Li, "Adaptive computation offloading for energy conservation on battery-powered systems," in Intl Conf. on Parallel and Distributed Systems, vol. 2, pp. 1, December 2009

[21] K. Kumar and Y. Lu, "Cloud Computing for Mobile Users: Can Offloading Computation Save Energy," IEEE Computer, vol. 43, no. 4, April 2010

[22] B-G. Chun and P. Maniatis, "Dynamically partitioning applications between weak devices and clouds," in Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond (MCS), no. 7, June 2010.

[23] Majid Altamimi, Rajesh Palit, Kshirasagar Naik, Amiya Nayak, "Energy-as-a-Service (EaaS): On the Efficacy of Multimedia Cloud Computing to Save Smartphone Energy," IEEE Fifth International Conference on Cloud Computing, 2012.

[24] Zhang, W., Wen, Y., Wu, J., & Li, H. (2013). Toward a unified elastic computing platform for smartphones with cloud support. IEEE Network, 27(5), 34-40.

[25] Qiang Fu, Weiwen Zhang, Baihua Wang, "Energy Efficient Mobile Cloud Computing and Its Applications", 9th IEEE International Conference on Networking, Architecture, and Storage, 2014

[26] Marco V. Barbera, Sokol Kosta, Alessandro Mei, Vasile C. Perta, and Julinda Stefa, "Mobile Offloading in the Wild: Findings and Lessons Learned Through a Real-life Experiment with a New Cloud-aware System," IEEE Conference on Computer Communications, IEEE INFOCOM 2014

[27] Xue Lin, Qing Xie "Task Scheduling with Dynamic Voltage and Frequency Scaling for Energy Minimization in the Mobile Cloud Computing Environment," IEEE Transactions On Services Computing, 2015

[28] H. Qui, A. Gani, "Research on mobile cloud computing: review, trend and perspective", Second International Conference on Digital Information and Communication Technology and its Applications (DICTAP), 2012

[29] Ke. Xu, Yuchao. Zhang, Xuelin. Shi, Wang. Haiyang, Yong. Wang and Meng. Shen, "Online combinatorial double auction for mobile cloud computing markets", IPCCC 2014, pp. 1-8, 2014

[30] A.-L. Jin, W. Song, P. Wang, D. Niyato, and P. Ju, ''Auction mechanisms toward efficient resource sharing for cloudlets in mobile cloud computing,'' IEEE Trans. Serv. Comput., 2015

[31] Ahmed Abdel Moamen, "Resource and Programmability Issues in Mobile Cloud Computing," Comprehensive Examination Report, University of Saskatchewan, 2014

[32] Zhang, W., Tan, S., Xia, F., Chen, X., Li, Z., Lu, Q., & Yang, S. (2016). A survey on decision making for task migration in mobile cloud environments. Personal and Ubiquitous Computing, 20(3), 295-309.

[33] David I. Fadaraliki, S.Rajendran,"Process Offloading from Android Device to Cloud Using JADE," International Conference on Circuit, Power and Computing Technologies, IEEE - 2015

[34] Angin, P., & Bhargava, B. K. (2013). An Agent-based Optimization Framework for Mobile-Cloud Computing. JoWUA, 4(2), 1-17.

[35] Lampe, U., Siebenhaar, M., Papageorgiou, A., Schuller, D., & Steinmetz, R. (2012, June). Maximizing cloud provider profit from equilibrium price auctions. In Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on (pp. 83-90). IEEE.

[36] Qian H, Andresen D (2015) An energy-saving task scheduler for mobile devices. In: 2015 IEEE/ACIS 14th international conference on computer and information science (ICIS). IEEE, pp 423–430

[37] Ha, K., Lewis, G., Simanta, S., & Satyanarayanan, M. (2011). Cloud offload in hostile environments (No. CMU-CS-11-146). CARNEGIE-MELLON UNIV PITTSBURGH PA SCHOOL OF COMPUTER SCIENCE.

[38] Fahim, A., Mtibaa, A., & Harras, K. A. (2013, September). Making the case for computational offloading in mobile device clouds. In Proceedings of the 19th annual international conference on Mobile computing & networking (pp. 203-205). ACM.

[39] Vyroubal, V., & Kušek, M. (2013, June). Task migration of JADE agents on Android platform. In Telecommunications (ConTEL), 2013 12th International Conference on (pp. 123-130). IEEE.

[40] Vyroubal, V., & Kušek, M. (2013, June). Task migration of JADE agents on Android platform. In Telecommunications (ConTEL), 2013 12th International Conference on (pp. 123-130). IEEE.

[41] Vyroubal, V., & Kušek, M. (2013, June). Task migration of JADE agents on Android platform. In Telecommunications (ConTEL), 2013 12th International Conference on (pp. 123-130). IEEE.

[42] Engelbrecht-Wiggans, R. (1980). State of the art—auctions and bidding models: a survey. Management Science, 26(2), 119-142.

[43] D. Chess, C. Harrison, and A. Kershenbaum, “Mobile agents: Are they a good idea?” in Mobile Object Systems Towards the Programmable Internet, LNCS, J. Vitek and C. Tschudin, Eds. Springer-Verlag, 1997, vol. 1222

[44] Nikraz, M., Caire, G., & Bahri, P. A. (2006). A methodology for the analysis and design of multi-agent systems using JADE.

[45] Ibukun, E., & Daramola, O. (2015). A Systematic Literature Review of Mobile Cloud Computing. International Journal of Multimedia and Ubiquitous Engineering, 10(12), 135-152.

[46] Nitesh Kaushik, Gaurav, Jitender Kumar, “A Literature Survey on Mobile Cloud Computing: Open Issues and Future Directions”, International Journal

Of Engineering And Computer Science ISSN:2319-7242 Volume 3 Issue 5 may, 2014 Page No. 6165-6172

[47] Maurya, S., & Mukherjee, K. A Literature Review on Mobile Cloud Computing Architecture. International Journal of Applied Engineering Research, 10(79), 2015.

[48] Fernando, N., Loke, S. W., & Rahayu, W. (2013). Mobile cloud computing: A survey. Future Generation Computer Systems, 29(1), 84-106.

[49] https://en.wikipedia.org/wiki/Mandelbrot_set

[50] http://www.apple.com/icloud/

[51] https://aws.amazon.com/documentation/silk/

[52] Younge, A. J., Von Laszewski, G., Wang, L., Lopez-Alarcon, S., & Carithers, W. (2010, August). Efficient resource management for cloud computing environments. In Green Computing Conference, 2010 International (pp. 357-364). IEEE.

[53] Liu, F., Shu, P., Jin, H., Ding, L., Yu, J., Niu, D., & Li, B. (2013). Gearing resource-poor mobile devices with powerful clouds: architectures, challenges, and applications. IEEE Wireless communications, 20(3), 14-22.

[54] Flores, H., Hui, P., Tarkoma, S., Li, Y., Srirama, S., & Buyya, R. (2015). Mobile code offloading: from concept to practice and beyond. IEEE Communications Magazine, 53(3), 80-88.

[55] http://aws.amazon.com/s3/

[56] Vartiainen, E., & Väänänen-Vainio-Mattila, K. (2010, December). User experience of mobile photo sharing in the cloud. In Proceedings of the 9th International Conference on Mobile and Ubiquitous Multimedia (p. 4). ACM.

[57] https://en.wikipedia.org/wiki/Fractal