A Collaborative Framework of Enabling Device Participation in Mobile Cloud Computing

Woonghee Lee, Suk Kyu Lee, Seungho Yoo, and Hwangnam Kim*

School of Electrical Engineering, Korea University
Seoul, Korea
{tgorevenge,sklee25,pen0423,hnkim}@korea.ac.kr

Abstract. Cloud Computing attracts much attention in the community of computer science and information technology because of resource efficiency and cost-effectiveness. It is also evolved to Mobile Cloud Computing to serve nomadic people. However, any service in Cloud Computing System inevitably experiences a network delay to access the computing resource or the data from the system, and entrusting the Cloud server with the entire task makes mobile devices idle. In order to mitigate the deterioration of network performance and improve the overall system performance, we propose a collaborative framework that lets the mobile device participate in the computation of Cloud Computing system by dynamically partitioning the workload across the device and the system. The proposed framework is based on it that the computing capability of the current mobile device is significantly enhanced in recent years and its multi-core CPU can employ threads to process the data in parallel. The empirical experimentation presents that it can be a promising approach to use the computing resource of the mobile device for executing computation-intensive tasks in Cloud Computing system.

Keywords: Mobile Cloud Computing, Multi-Thread, Parallel Computing.

1 Introduction

Cloud Computing is the technology, the service or the system for serving computing and storage capacity to the end-recipient [1]. One of important usage scenarios for Cloud Computing system is that the system executes computation-intensive tasks instead of the mobile device with the data which a user provides. This scenario assumes that the Cloud server has very high computational power and it produces the required results in a shorter time[1]. Mobile Cloud Computing (MCC) is the system concept added the mobility feature to Cloud Computing, where the transmission involves to wireless connectivity [2]. Therefore, the transmission delay becomes an additional important factor determining the overall system performance.

On the other hand, the hardware and the software of mobile devices have been developed significantly in recent years. For example, the smart-phone equipped with

* Corresponding author.

[1] Note that we focus on this scenario in this paper.

quard-core-CPU and 2GB RAM has been released recently. The enhanced mobile device can do multitasking -- it can run several programs at the same time such as surfing the Internet with listening to music -- and it can process the data in parallel with the multi-thread. Users can thus exert high performance and the latest mobile device can handle much higher computation than those of the past. Therefore, the MCC system should pay attention to this development at the mobile device side in order to overcome the current limitations that (*i*) the network delay may negatively affect the overall performance when the entire user data is transferred back and forth between the Cloud server and the mobile device and (*ii*) the mobile device wastes its computing cycles till it receives any response from the server.

These observations motivate us to devise a collaborative framework on computation for MCC, which exploits the mobile device as an additional computing element of MCC. Thus, we propose in this paper a collaboration framework of using the computational power of the mobile device as an additional resource of MCC in order to improve the system performance by using the mobile device's resource to the fullest. We constructed an empirical test-bed to present that the proposed framework can outperform the traditional computing framework for Cloud Computing. The experimentation results indicate that the proposed framework can enhance the performance of most Cloud Computing system. Note that the proposed framework can be applied to computation-intensive services, not simple browsing or accessing data services.

The remaining part of this paper is organized as follows. We briefly explain the MCC system and the related works in Section 2. We then present the proposed framework and detail the process of framework in Section 3. We evaluate the proposed framework and also state a brief scheduling scheme in Section 4. We conclude the paper with Section 5.

2 Preliminary

In this section we describe some features of Mobile Cloud Computing system, remind that the fundamental strength of parallel computing for computation-intensive tasks, even in the mobile device, and summarize the relevant work that are used in subsequent sections.

2.1 Computation-Intensive Service at Mobile Cloud Computing

The MCC is the newest mobile computing technology for overcoming the limitation of resource shortcomings at the mobile device [3]. As for the computation intensive services at MCC, the entire data is transferred to the Cloud server from the mobile device, the server then processes it with its own resources, and finally the device receives the processed data. In this procedure, MCC has the following features:

1. Network environment dependency: MCC is a concept that the wireless feature is added to Cloud Computing, so that the data transmission involves to the wireless medium. Therefore, the transmission delay dependent on underlying network conditions becomes a more important element of MCC. Note that the wireless network condition is time-varying and hard to estimate [4].
2. Cloud server dependency: To overcome the limitation of the mobile device such as processing and memory capabilities, almost the whole data processing is

performed on the Cloud server, so that the entire processing performance relies on Cloud server's capability. However, the capability may dynamically change with several reasons such as the difference in number of users, which also cannot be controlled and managed by the end mobile device.

3. Service availability: The Cloud server is responsible for responding to millions of user's requests with processing the massive data simultaneously. It is an essential prerequisite to continuously support the MCC services even when the number of users exceeds the maximum allowable limit, which should be possible with acceptable performance degradation [5].

We can discover additional features derived from the above main features. (*i*) We cannot expect the constant service turn-around time because the number of service users and the amount of resources for each user request are dynamically changing. (*ii*) Computing resources of the mobile device may be wasted till any response from the Cloud server arrives even though it can be used and more productive in the service provisioning. (*iii*) The communication procedure between the MCC system and the user is fixed, regardless of the device capability, the service type, the network condition, and the current usage of the Cloud server.

2.2 Parallel Processing Schemes for Mobile Device

We briefly explain the current state of the art of parallel processing that we would like to apply to the proposed framework.

Threads. Due to the development of software and hardware, the average size of data is bigger than before and a number of computation-intensive processes appear. Especially, any work that deals with images or videos such as the digital image processing needs many arithmetic operations in handling big data and furthermore requires real-time processing. Therefore, it is possible to process the data using multi-core or multi-thread owing to improvement in hardware like CPU and memory. Parallel computing using multi-thread to process the work divided up into small segments is suitable for computation-intensive processes and it has become the essential method for improving the computing performance [6].

Simultaneous Processing and Transmission. We conducted a simple experiment for processing and transmission using multi-thread simultaneously before we propose a collaborative framework. There are the sender and the receiver in the experiment. The sender starts the process with setting to work. The sender does processing and sends some parts of the processed data at the same time, and then the receiver receives them from the sender and simultaneously does other processing with the received data. All processes are performed by using multi-thread.

2.3 Edge Detection and Discrete Cosine Transform

We used two tasks in our empirical studies in order to demonstrate the benefit of the proposed collaborative framework. One is Edge Detection and the other is Discrete Cosine Transform (DCT). Edge Detection is a fundamental tool in the computer vision and image processing and it is a process for extracting the significant properties of objects in the image. It can be used to do feature detection, feature extraction and identification of the physical phenomena, and therefore it can be employed by any application in 3D reconstruction, motion recognition, image enhancement, image restoration, image compression and so on [7]. DCT is a mathematical tool to express a sequence of finite data using the combination of cosine functions. It is very important in science and engineering such as image processing and compression of audio/image data [8].

Both tasks are widely used for image processing and image compression. Image processing on mobile devices is a new field because the latest phones are equipped with camera, and high performance CPU [9]. Additionally, in terms of the wireless network, image compression is an essential process. Therefore, we choose them as feasible tasks in mobile devices. However, the proposed framework can be applied to any other task in mobile computing environment in order to achieve performance improvement.

2.4 Advantages of Parallel Processing

In order to present the advantages of the proposed framework of letting the mobile device participate in computation in Mobile Cloud Computing environment, we conducted an empirical test with two scenarios. In case A, Sending/receiving are performed after processing is complete, and in case B, they are performed simultaneously. Each of the sender and the receiver has the 4 threads for processing and 1 thread for sending/receiving in case B. As for the task, we use the Discrete Cosine Transform. The DCT for image is generally performed for the block resulted from splitting the image. The block is a square 8 pixels on a side. The sender performed 2D-DCT using 4 threads in parallel and sent the processed blocks, and then the receiver received the processed blocks and performed 2D inverse DCT using 4 threads in parallel. This experimentation was performed in a single PC using two terminals for the sender and the receiver.

We present the empirical result in Table 1. The total processing time of Case B is much shorter than that of Case A. These results show that it is very efficient to do processing image blocks and sending/receiving them simultaneously.

Table 1. The result for processing and sending/receiving simultaneously

	A		B	
	Sender	Receiver	Sender	Receiver
1	240	253	42	69
2	237	257	42	59
3	240	253	52	75
4	239	257	54	61
5	237	253	53	77
6	238	257	56	56
7	241	248	46	61
8	242	260	46	70
9	246	263	54	73
10	238	254	49	67
mean	239.8 ms	255.5 ms	49.4 ms	66.8 ms

Since the latest mobile devices are able to deliver high performance and to use multi-thread like PCs, we were motivated to use the computation capability of the mobile device to improve the overall performance of MCC.

3 Collaborative Framework

As shown in Section 2.4, we have got a hint that we could improve the service performance by letting the mobile device take some parts of the high computational processes.

3.1 Assumed Configuration

It is fundamental to employ threads for all processing in the proposed framework. There are two thread types. One is the task thread for processing the data and the other is the transmission thread for sending/receiving the data. We just use one thread for transmission thread, but one or more number of task threads for processing the data. Even though the processing performance is generally increased with the number of threads, the improvement becomes stalled and then degraded after reaching a threshold. The main reasons of such the degradation are stretching the CPU to handle too many threads and managing the memory access [6]. We found through much experimentation that the optimum number of threads is in the range between 3 and 5 in our experiments. Naturally, the tasks (edge detection and DCT) that we have chosen for the experimentation are processed in parallel.

There is a pair of the server and the client, the server takes the role of MCC and the client takes the role of user's mobile device. *The server is a virtual server whose capability represents the overall capability of the Mobile Cloud Computing infra*. The total processing starts if the client initiates the processing and it ends when the client has total processed data. The overall performance means the turn-around time, meaning the time required for finishing all processes. The overall performance is influenced by network conditions because MCC is running in wireless network environment in which there are many unpredictable variables. Note that the underlying network is assumed to have a wireless connectivity.

The client can choose the amount of data to process at the client and the rest of data is processed at the server. Therefore, the amount of data to be processed at the client varies depending on the situation: network conditions, the mobile device's performance, and the server's state.

3.2 Operational Procedure

The procedure of the proposed framework is described in what follows. (*i*) First of all, the client partitions the task into two parts (where the one is to be processed in the server and the other is to be done at the mobile device). The ratio varies depending on factors influencing the performance of the system such as the capability of the client and the server, and network situations. It is the most important stage for improving the efficiency for the client to decide the optimal ratio. (*ii*) After the ratio has been chosen, the client transfers the server part of data to the server via the transmission thread and starts processing the rest of data with the task threads simultaneously.

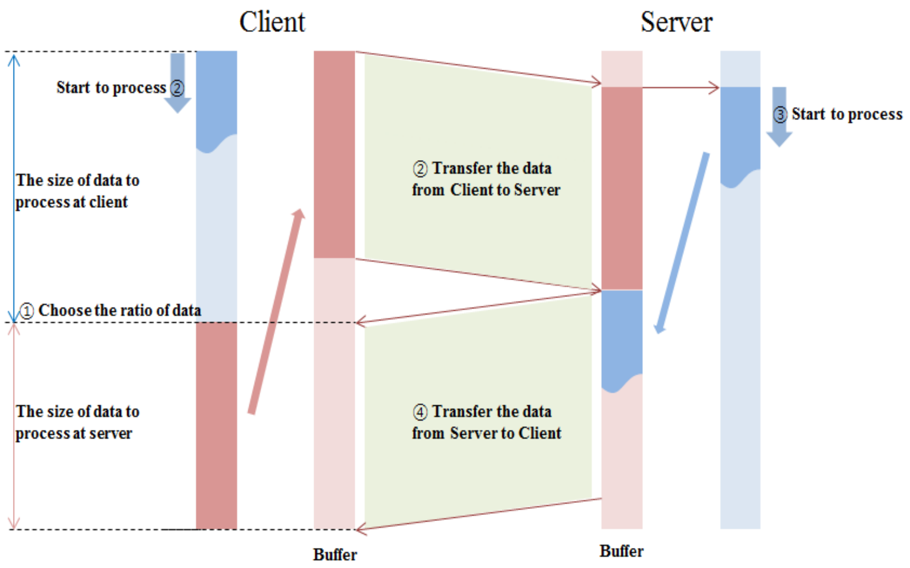
Client
Server
Start to process ②
The size of data to process at client
① Choose the ratio of data
The size of data to process at server
② Transfer the data from Client to Server
④ Transfer the data from Server to Client
③ Start to process
Buffer
Buffer

Fig. 1. An execution flow of the proposed framework

(*iii*) The server starts the work with received data. Once all the data is processed, the transmission thread sends the data back to the client. (*iv*) The client receives the server data from the server. When all processes of both sides are finished, the client converges the two parts from different locations into the final data. After that, all of the processes are done. Figure 1 shows the procedure briefly.

The pseudo code of the proposed framework is presented in the Algorithm 1.

4 Performance Evaluation

We employed two evaluation scenarios according to the amount of data processing in order to evaluate the proposed framework. One is the low computational task (Task A) and the other is the high computational task (Task B). Edge detection is used for the low computational task, and DCT is added to the low computational task for the high computational task. Edge Detection mainly consists of integer computations and has few processes, whereas DCT is composed mainly of real number computations and has a large amount of processes. The PC is equipped with 'Intel core i5-2410M', DDR3 4GB RAM, and the mobile device runs Google Android OS version 4.03 in the platform of 'Qualcomm 1.5GHz dual-core MSM8660 Snapdragon', 1GB DDR2 SDRAM. Table 2 shows the approximate average run-time difference between the PC and the mobile device used in the experiment.

Table 2. The average run-time in diverse cases

	Task A	Task B
Mobile device	1000 *ms*	12600 *ms*
PC	80 *ms* (x 12.5)	280 *ms* (x 45)

Algorithm 1. *Pseudo code of the framework*

```
                Server                                  Client
A = The data assigned to the server;
B = The data assigned to the client;
byte buf[]; // buffer array for receiving the data
byte $D_{process}$[]; //array for the processed data
class ProcessThread{                    class ProcessThread{
    P = first position;                     do processing B;
    while (true){                           P = first position;
       if(P >= size of A)then               while(true){
        break;                                 if(P >= size of A)then
       if(buf[P] != null)then                   break;
        do processing buf[P];                  if(buf[P] != null)then
        save the processed                      move the data in
         data to $D_{process}$[P];                    place;
        P += next position;                     P += next position;
    }                                       }
}                                       }
class TransThread{                      class TransThread{
    connect to client;                      connect to server
    receive the A;                          send the A;
    while(true){                            T = 0;
      if($D_{process}$ != null)then               while(true){
       send $D_{process}$ to Client;              if(T >= size of A)then
       break;                                   break;
    }                                          receive the data;
}                                              T += received data;
class Mainclass{                            }
    void main(){                        }
       ProcessThread Pthread;           class Mainclass{
       TransThread Tthread;                 void main(){
       Pthread.start();                        ProcessThread Pthread;
       Tthread.start();                        TransThread Tthread;
       Pthread.join();                         Pthread.start();
       Tthread.join();                         Tthread.start();
    }                                          Pthread.join();
}                                              Tthread.join();
                                            }
                                        }
```

PC's performance is much better than that of the mobile device. In comparison with the run-time, the PC shows better computation capability in Task B (45 times) than in Task A (12.5 times). It means high computational processes such as real number computations overhead much upon the mobile device and processing the high computational operations at the server is more desirable. As mentioned in Section 3, there are the server and the client in our experiments. The PC takes the role of the server and the mobile device takes the other.

Performance Evaluation w.r.t. Low Computational Task. First experiment is performed with 7 configurations in each of which the ratio of data partitioning is changed by 1/6. Both the server and the client have 3 threads for processing. We conducted 5 experimentation runs for each configuration, and so the experimentation results in subsequent explanation are average values. Connectivity between the server and the mobile device is established through Wi-Fi.

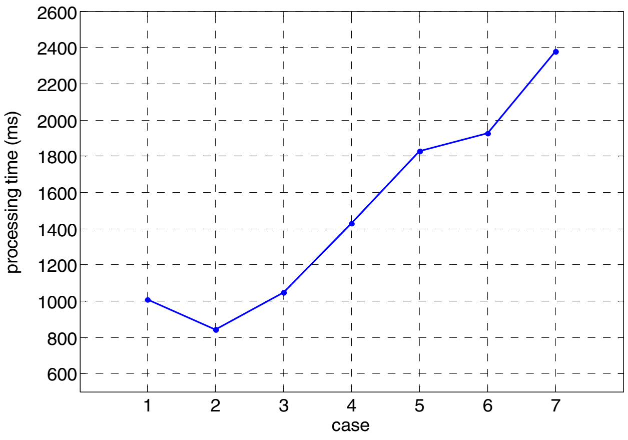

Fig. 2. Turn-around time for each configuration of task partition for Task A

Figure 2 presents the turn-around time of the whole task for each configuration. As shown in the figure, the case 1 (where the entire data is processed at the mobile device) took shorter time than case 7 (where the server is entitled to process the total data). This is because the transmission part took more time than the processing part due to wireless environment; in other words, the delay in the transmission part became the dominating value because of the short processing time.

Additionally, we observed that the time required for sending/receiving total data is over 1 second and the time required for processing total data at mobile device took 1 second approximately, and so it looks inefficient using Mobile Cloud Computing.

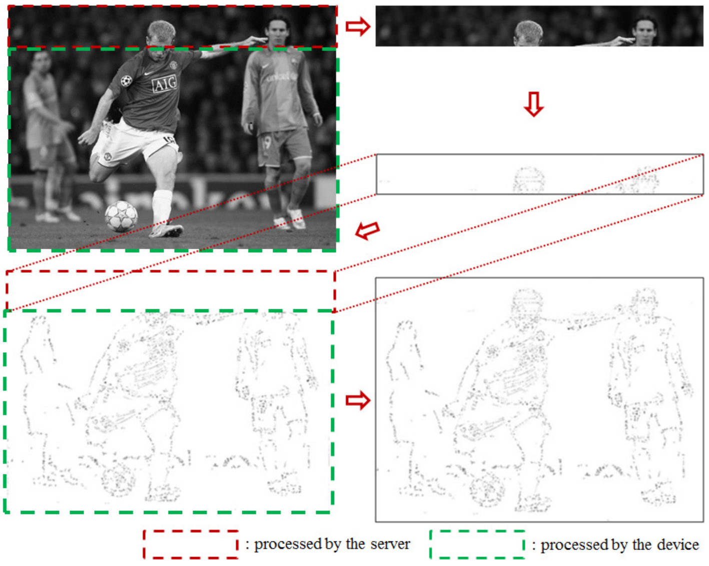

Fig. 3. An execution procedure at case 2 for Task A

However, the computational power of the PC (Server) is much better than mobile devices as shown Table 2, so that it could improve the overall efficiency of processing if the proper allocation of data processing is made for the server. It is shown in the case 2 where one sixth of the data is processed at the server and results in the shortest total processing time. The turn-around time of case 2 is shorter than that of the case 1 by about 150*ms*. It means one sixth of the data is the best allocation between processing and sending/receiving in this experiment. Figure 3 shows the execution procedure for the case 2 of task A.

Performance Evaluation w.r.t. High Computational Task. The next experimentation is done with the high computational Task. Figure 4 shows the result of the experiment in this case. The experiment was conducted in the similar way to the previous one. As shown in the figure, the case 7 where the entire data is processed at the server took shorter time than the case 1 where the data is processed only at the mobile device. This comparison shows that mobile device's computational capability could be overwhelmed by the plenty of computation such as DCT. It is more efficient to exploit server computation facility to the fullest. It looks like the advantage assumed in general Cloud Computing system. However, it is not quite the same with that. Even though the case 7 has to show the most efficient result (according to the intention of the current Cloud Computing system), the case 6 (where the mobile device took part in the required computation) shows the shortest time in this experiment. Thus, the ratio of data in case 6 is most pertinent.

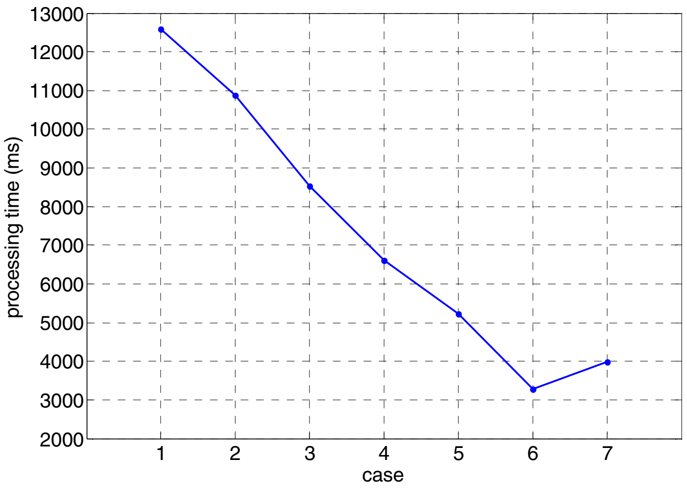

Fig. 4. Turn-around time for each configuration of task partition for Task B

Remarks on Advantages. Recall the issues of existing Mobile Cloud Computing presented in Section 2. The proposed framework can alleviate those issues as follows.

- Low network dependency: The mobile network environment is relatively unstable so its quality cannot be guaranteed. As the occasion, transmission time can be a more dominant factor deciding the overall performance than processing time as shown above. When the network state is stable, sending more data to the server is reasonable. If not, processing more data at the mobile device improves the overall system performance. Also, it can be entirely processed by the mobile device in the extreme case that any connection is not feasible.
- Low Cloud server dependency: Cloud Computing aims at improving the overall performance of processing using the server's computation capability and resources but it is useless if the server could not provide those sufficiently. In this case, it is better to use more resources of the mobile device. Mobile device's resource can be controlled by users and its state is easily ascertainable than that of the server. Processing the data at the mobile device reduces the Cloud server dependency.
- Maximization of mobile device's resources: Entrusting the server with the entire task causes mobile devices idle. Processing some parts of data at the mobile device is one of the ways that optimize the resource usage at the device.
- High flexibility: Factors influencing the performance of Cloud Computing are various. The fixed way for providing services to users is one of causes degrading the overall stability of the service. Users can choose the degree of participation of the user's device in computation in the light of the mobile network condition, device's and server's performance, characteristics of processing. Therefore, these make system flexible and stable, which allows the Cloud server to provide more guaranteed services for users.
- Optimization of Efficiency: It is most important to determine the optimal workload partitioning as shown above, and the efficiency can be achieved by changing the

ratio. The empirical results show that the optimal ratio can be determined and to what extent the optimization is possible.

- Reducing the burden of the server: Distributing the workload across the server and the mobile device can save the server capacity, so that the server is able to serve other users' requests and thus it achieves higher service availability.

A Sketch on Scheduling Algorithm. There are several factors that influence on the proposed framework: the processing capability of the mobile device and the server, network conditions, and characteristics of processing. By considering those factors, we can present a sketch on how to partition workload across the device and the Cloud server based upon the premise that the server maintains steady-state.

Firstly, we determine scheduling parameters as specified in Table 3, which can be estimated by the current network conditions and the computational capability of the server and the device, and then we decide the optimal ratio with those parameters as follows.

Table 3. Scheduling parameters

Notation	Description
T_d	The time required for processing the entire data at the device
T_s	The time required for processing the entire data at the server
T_t	The time required for sending the entire data between the server and the device
T_c	The time required for converging the two parts from different locations.
X	The ratio of work for processing at the server ($0 \leq X \leq 1$)

The total time required at the server is shown in equation (1), whereas the total time required at the device is shown in (2).

$$2XT_t + XT_s\ . \tag{1}$$

$$(1\text{-}X)T_d + XT_c\ . \tag{2}$$

If the work of the server is finished earlier than that of the device, it means that we do not utilize the server's resource efficiently. On the other hand, if the work of the device is finished earlier than that of the server, it means that the more work can be processed at the device. Therefore, the work can be done most efficiently when either the server or the device is not idle. The optimal ratio X is obtained by using (4).

$$2XT_t + XT_s = (1\text{-}X)T_d + XT_c\ . \tag{3}$$

$$X = T_d\ /\ (\ 2T_t + T_s + T_d - T_c\ )\ . \tag{4}$$

In Eq. (4), $2T_t + T_s$ means that the time required in case 7, and T_d stands for the time required in case 1. T_c can be negligible for obtaining the X approximately in (4) because T_c is much smaller compared to T_d, T_s, T_t. Therefore, we disregarded T_c to obtain the X more easily. The ratio X is roughly 0.29 in the experimentation for low computational Task and it is between 1/6 and 2/6. Additionally, The X is roughly 0.75 in the experimentation for high computational Task and it is between 4/6 and 5/6. Therefore, the optimal ratio obtained by using (4) accords closely with results of our experiment.

Figure 5 shows the change of X depending on the alteration of T_t, T_s, and T_d. The bigger T_d/T_s means that the Cloud server has more computational power than the device and the smaller T_t/T_s means that the network condition is better. Therefore, the bigger T_d/T_s or the smaller T_t/T_s is, the more data can be processed at the Cloud server more efficiently.

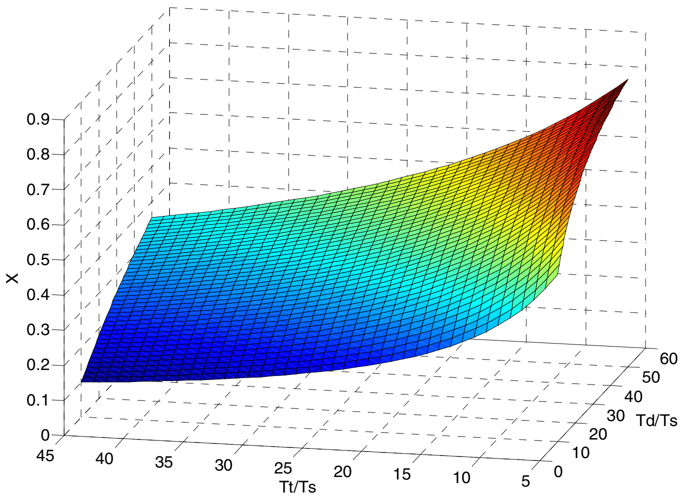

Fig. 5. The change of X depending on the alteration of factors

5 Conclusion

Based on the observation that there are several disadvantages of existing Cloud Computing such as high network environment dependency, high Cloud server dependency, idle mobile devices, low flexibility, and large burden of the server, we proposed a new collaborative framework of letting the mobile device participate in computation-intensive tasks in Mobile Cloud Computing environment. The participation ratio is determined depending on the factors of the system capability, the mobile device's performance, the network state, and characteristics of processing. We ascertained with empirical studies that the proposed framework makes the system achieve better performance and more flexible. It can also alleviate the server workload by using the device capability.

We have several directions as future work. We will elaborate the scheduling algorithm for the proposed framework. We plan to implement the proposed framework in the specific Mobile Cloud Computing infra such as Hadoop-based Cloud Computing system. We also would like to implement the proposed framework in various mobile device platforms such as iPhone and Windows phone.

Acknowledgement. This research was supported in part by the National Research Foundation of Korea (NRF) Grant funded by the Korea government (MEST) (No. 2010-0014060), and in part by the KCC (Korea Communications Commission) Korea under the R&D program supervised by the KCA (Korea Communications Agency) (KCA-2012-08-911-05-001).

References

1. Buyya, R., Yeo, C.S., Venugopal, S.: Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities. In: 10th IEEE International Conference on High Performance Computing and Communications, HPCC 2008 (2008)
2. Sanaei, Z., Abolfazli, S., Gani, A., Khokhar, R.H.: Tripod of requirements in horizontal heterogeneous Mobile Cloud Computing. In: Proc.1st Int'l Conf. Computing, Information Systems, and Communications (2012)
3. Sanaei, Z., Abolfazli, S., Gani, A., Shiraz, M.: SAMI: Service-Based Arbitrated Multi-Tier Infrastructure for Mobile Cloud Computing. In: Mobicc. IEEE Workshop on Mobile Cloud Computing, Beijing, China (2012)
4. Woo, S., Kim, H.: Estimating Link Reliability in Wireless Networks: An Empirical Study and Interference Modeling. In: 2010 Proceedings IEEE INFOCOM (2010)
5. Zhu, J., Jiang, Z., Xiao, Z.: Twinkle: A fast resource provisioning mechanism for internet services. In: Proceedings of IEEE INFOCOM (2011)
6. Tullsen, D.M., Eggers, S.J., Levy, H.M.: Simultaneous multithreading: maximizing on-chip parallelism. In: Proceedings of the 25th Annual International Symposium on Computer Architecture, ISCA 1998, pp. 533–544 (1998)
7. Vincent, T.: On Edge Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence (1986)
8. Ahmed, N.: Discrete Cosine Transform. IEEE Transactions on Computers (1974)
9. Wells, M.T.: Mobile Image Processing on the Google Phone with the Android Operating System, http://www.3programmers.com/mwells/main_frame.html