An Adaptive Data Mapping Storage Selection Algorithm in Mobile Cloud Computing

Ahmed. A. A. Gad-ElRab
Faculty of Science
Al-Azhar University, Cairo, Egypt

Eman. H. Zaky
Faculty of Science
Al-Azhar University, Cairo, Egypt

Neveen I. Ghali
Faculty of Science
Al-Azhar University, Cairo, Egypt

ABSTRACT

Mobile cloud computing (MCC) is a new computing paradigm which tends to transfer the data storage and the data processing from a mobile device to a cloud server on the Internet. The cloud server may be a block sever or a file server. In MCC, due to the limited resources of a mobile device as processing power, battery power, and memory, the main challenge is how to map data items into a cloud server and select the best mapping server among block and file servers. In this paper, the difficulties in mapping data items are addressed and a new adaptive data mapping storage scheme is proposed. The proposed scheme can select a block or file mapping for mobile data items based on a defined cost model which takes into account the energy consumption and the total time delay for sending and retrieving of data to/from the cloud server. In addition, the proposed scheme can select the optimal number of blocks and files of data items that adaptively changes with their cost models. The simulated results show that the proposed algorithm achieves a better mapping performance with minimum cost compared to the mapping data items without any selection mechanism.

Keywords

Cloud computing, Mobile cloud computing, File level storage, Block level storage.

1. INTRODUCTION

The advanced developments in web technology, information technology (IT), and wireless communications present a new architecture of computing which is called *Cloud Computing* (CC). CC will be the next generation of IT Enterprise. In the traditional solutions of computing architecture, the IT services are maintained with suitable personnel, logical and physical controls. While, CC manages and processes the data and services optimally by moving the databases and application software to the large data centers on the Internet which are called clouds. It works on storing information of fixed machine's hard drive or other local storage device, into a remote database. The Internet provides the connection between the computer and the database [1], [2]. Mobile cloud computing is similar to cloud computing though there is no correlated definitions.

Recently, a novel computing mode consists of mobile computing and cloud computing which is called mobile cloud computing (MCC) is developed. MCC provides cloud based services to users through the Internet and mobile devices. Fig. 1 shows the architecture of MCC which can be simply divided into cloud computing and mobile computing [3]. In MCC, an access point or a base station with GPRS, WIFI, or 3G capabilities is used to connect the mobile devices with the Internet. Based on this architecture, by using a web browser or desktop application, mobile users send service requests to a cloud server. Then the cloud server establishes a connection by allocating the required resources for these requests. At the same time, to ensure the *Quality of Service* (QoS), the cloud server implements the calculating and monitoring operations until the connection is completed [3].

Therefore, the MCC extends the capabilities and reflects advantages of cloud computing and develops the functionalities of mobile computing. So, MCC is a combination of the two technologies for developing a lot of centralized, distributed, and grid applications. In MCC, the developing, running, deploying and using of mobile applications have been totally changed. Because, intensive computing, data storage and mass information processing have been transferred from the mobile device to the cloud. As a result, computing capability and resources of the mobile devices can be saved. In addition, the cloud service are not restricted to fixed devices, but the mobile devices with people like smartphone, iPad, Tablet, and PDA are suitable to access and use these cloud services.

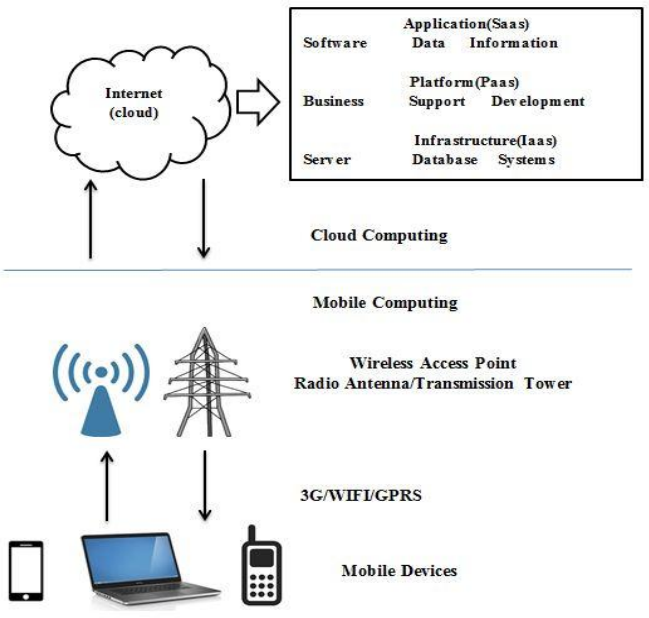

Fig 1: Architecture of Mobile Cloud Computing (MCC)

In MCC, the cloud server can be a block severs which stores all data items as blocks or a file server which stores all data items as files. Due to the limited resources of a mobile device as processing power, battery power, and memory, the main challenge is how to map data items into a cloud server and how to select the best mapping server among block and file servers for each data item.

In this paper, the difficulties in mapping data items are addressed and a new mapping selection scheme is proposed, the proposed algorithm can select a block or file mapping of mobile data items based on a defined cost model which takes into account the energy consumption and the total time delay for sending and retrieving the data to/from the cloud server. In addition, the proposed scheme can adjust the optimal number of blocks and files of data items that adaptively changes with their cost models.

The rest of the paper is organized as follows: Section 1, introduces a brief description for file and block storage levels and reviews the related works. Section 2, describes proposed model for data mapping selection problem in MCC. Section 3, introduces the proposed algorithm. Section 4, presents the conducted simulations and discusses their results. Finally, Section 5 concludes the paper.

2. RELATED WORK

The online storage of data on the cloud is called cloud storage. With this cloud storage, the mobile data is transferred and stored in the cloud and the mobile device can access these data from the cloud by sending a service or data requests. The *file level storage* and *block level storage* are the two most common technologies of the cloud storage system. These two storage levels are described as follows.

File level storage: it is common in hard drives and Network Attached Storage (NAS) devices. Here data read/write happens with a client-server model. The client requests data from the storage by using attributes, such as the data filename, directory location, and URL. The server receives the client request and looks up data storage locations where the data is stored and retrieves it by using storing level functions. Here, data is stored as bytes in a file format. Also, file level storage is simple to use and implement which stores files and folders and the visibility is the same to the clients accessing and to the system which stores it. File level storage has a low maintenance cost compared to block level storage. Also, file level storage can handle access control, integrate with corporate directories, so it is the base in building in network attached storage systems. Scale out NAS is a type of file level storage that merges a distributed file system that can scale a single volume with a single name-space across many nodes, and scale out NAS file level storage solutions can scale up to several petabytes all while handling thousands of clients. As capacity is scaled out, performance is scaled up.

Block level storage: it finds its major application in a Storage Area Networks (SAN) environment. Here, data is stored as blocks in hard drives, which are installed in a remote storage arrays accessible to the network computers using Fiber Channel or iSCSI. Also, the filing system sends a request to the storage to write data to certain blocks and then retrieve it. Typically, Logical Unit Numbers (LUN) are created, which are treated like storage blocks and given to a server where, the server views it as a local disk and reads from and writes to it. Each storage block/storage volume can individually be treated as an independent disk drive and can be controlled by an external server operating system, and formatted with the file system. Also, block level storage can be used to store files and work as storage for special applications like virtual machine file systems and databases. In SAN or storage area network environment, to offers boot-up of systems which are connected to them, block level storage is usually used. Also, it supports various formatting of file systems like NFS, NTFS or VMFS (VMware) or SMB (Windows) which are required by the applications. The iSCSI and FCoE protocols are used in block level storage for data transfer. In addition, data transportation in block level storage is much reliable and efficient.

In [4], checking for deduplication secrete sharing scheme used for data fragmentations. While, convergent key approach is proposed to encrypt data before deduplication in [5]. DROPS Concept was introduced in [6], which is used for fragmentation and Replication of data. In [6], coloring algorithm for placing node is given. So, reliability is achieved. While authors in [7] presents auditing algorithms for achieving integrity.

For several decades, the database research community has faced the challenge of consistent and scalable data management. The first generic solution to this challenge is the distributed database systems [8], [9]. This solution ensures the global serializability when transacts with data which not bounded to a single machine [10], [11].

Recently, software infrastructures and business models of Internet services has been changed by using cloud computing technologies. These cloud technologies provide and manage various resources of computation and data storage over the network that minimize the costs of data processing and accessing operations [12], [13]. In [14] the way people access computers and network services has been significantly changed due to publicity of smart devices and mobile networks. Also, many advanced smartphone applications shows the effects of mobile cloud computing in building and developing of these applications. In [15], the cloud computing offers great convenience to users by transferring data to the cloud. In this situation, users do not need to manage the required resources and hardware. The well-known examples of cloud computing vendors are Amazon Elastic Compute Cloud (EC2) and Amazon Simple Storage Service (S3).

In [16], huge amounts of storage space and customizable computing resources are provided by using the Internet-based online services. At the same time, this eliminates the responsibility of local machines for data maintenance. In addition, the cloud service providers manage the availability and integrity of the data for their users. In [17], to enable convenient, ubiquitous, and on-demand network access to a shared group of computing resources (e.g., services, applications, servers, storage, and network) a cloud computing model was proposed. This model can be deployed easily and it minimizes the costs of management effort or service provider interaction. In [18] authors tried to ensure file integrity across multiple distributed servers, erasure-coding and block-level file integrity checks are used. However, their scheme only considers static data files and do not explicitly study the problem of data error localization.

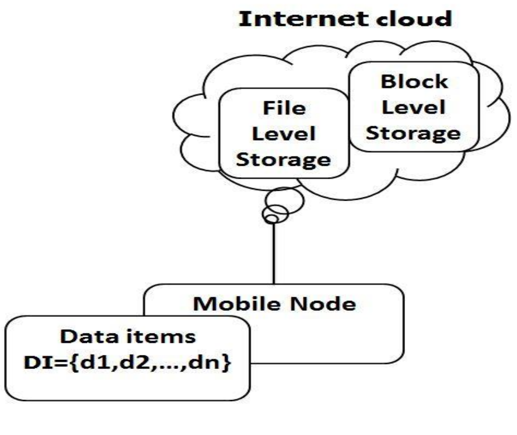

Fig. 2. The proposed MCC model

3. PROPOSED MODEL FOR DATA MAPPING STORAGE SELECTION PROBLEM IN MCC

In MCC, based on the existing of these two storage levels, the problem is how to map mobile data item by selecting the most appropriate storage level for each data item by taking into account the limited resources of a mobile device. This problem is called ***Data Mapping Storage Selection (DMSS)*** problem.

In the rest of this section, the assumptions and models are introduced then the DMSS problem will be formulated.

3.1 Assumptions, and Models

Here, MCC system model consists of (1) mobile node, *MN*, (i.e., mobile device with a user) which has a set of data items *DI* = {d_i, 1<= *i* <= *n*}, and each data item d_i represents different types of data (e.g., text, image, video, etc.). (2) a file cloud server, *fs*, which stores all received data elements d_i as files. (3) a block cloud server, *bs*, which stores all received data elements d_i as blocks. The characteristics of file and block servers as available storage, processing power, and how many stored files/blocks of data in the server are assumed to be known in advance. The energy consumed for sending and receiving a data item d_i to/from a server *s* are denoted as $se(d_i; S)$ and $re(d_i; S)$, respectively. Where, *s* can be a file server *fs* or a block server *bs*.

Also, The time delay for sending and receiving a data item d_i to/from a server *s* are denoted as $std(d_i; s)$ and $rtd(d_i; s)$, respectively. Fig.2 shows the proposed MCC Model.

3.2 Cost Model for DMSS

In this subsection, the cost model for DMSS is described

for using the file and the block servers.

3.2.1 Cost by using block server

In case of a block server, it is needed to classify each data item before storing it in the appropriate block. So, the cost for sending includes sending, classifying, and searching costs and the cost for receiving includes retrieving and searching costs.

- Energy cost: the cost of the consumed energy for sending *DI* to the file server *bs* is calculated as follows

$$\mathbf{SE_{bs}(DI) = \sum_{d_i \in DI} se(d_i, bs)} \quad (1)$$

and the cost of the consumed energy for receiving DI from the file server bs is calculated as follows

$$\mathbf{RE_{bs}(DI) = \sum_{d_i \in DI} re(d_i, bs)} \quad (2)$$

By using Equations 1 and 2, the total energy cost by using bs can be defined as follows.

$$\mathbf{TED_{bs}(DI) = SE_{bs}(DI) + RE_{bs}(DI)} \quad (3)$$

Time delay cost: the cost of the time delay for sending DI to the block server bs is calculated as follows

$$\mathbf{STD_{bs}(DI) = \sum_{d_i \in DI} std(d_i, bs)} \quad (4)$$

and the cost of the time delay for receiving DI from the file server bs is calculated as follows

$$\mathbf{RTD_{bs}(DI) = \sum_{d_i \in DI} rtd(d_i, bs)} \quad (5)$$

By using Equations 7 and 8, the total time delay cost by using bs can be defined as follows.

$$\boldsymbol{TTDC_{bs}(DI) = STD_{bs}(DI) + RTD_{bs}(DI)} \quad (6)$$

3.2.2 Cost by using file server

In case of a file server, the cost for sending includes sending without classification and searching costs and the cost for receiving includes retrieving and searching costs.

- Energy cost: the cost of the consumed energy for sending

DI to the file server fs is calculated as follows

$$\mathrm{SE}_{fs}(\mathrm{DI}) = \sum_{d_i \in DI} se(d_i, fs) \quad (7)$$

and the cost of the consumed energy for receiving DI from the file server fs is calculated as follows

$$\mathrm{RE}_{fs}(\mathrm{DI}) = \sum_{d_i \in DI} re(d_i, fs) \quad (8)$$

By using Equations 7 and 8, the total energy cost by using fs can be defined as follows.

$$TEC_{fs}(DI) = SE_{fs}(DI) + RE_{fs}(DI) \quad (9)$$

- Time delay cost: the cost of the time delay for sending DI to the file server fs is calculated as follows

$$\mathrm{STD}_{fs}(\mathrm{DI}) = \sum_{d_i \in DI} std(d_i, fs) \quad (10)$$

and the cost of the time delay for receiving DI from the file server fs is calculated as follows

$$\mathbf{RTD_{fs}(DI) = \sum_{d_i \in DI} rtd(d_i, fs)} \quad (11)$$

By using Equations 7 and 8, the total time delay cost by using fs can be defined as follows.

$$\mathbf{TTDC_{fs}(DI) = STD_{fs}(DI) + RTD_{fs}(DI)} \quad (12)$$

In MCC, the most important for the mobile user may be the consumed energy, the time delay for sending and receiving data, or both of them. To represent these priorities, the total cost for each data item di ∈ DI by using fs and bs are reformulated and the weighted accumulated costs are proposed which are defined as follows.

$$\mathbf{tc_{fs}(d_i) = w_1 \frac{tec_{fs}(d_i)}{E_{Max}} + w_2 \frac{ttdc_{fs}(d_i)}{TD_{Max}}} \quad (13)$$

and

$$\mathbf{tc_{bs}(d_i) = w_1 \frac{tec_{bs}(d_i)}{E_{Max}} + w_2 \frac{ttdc_{bs}(d_i)}{TD_{Max}}} \quad (14)$$

where tecfs(di), ttdcbs(di), tecbs(di), and ttdcbs(di) are the energy and the time delay costs (i.e., for sending and receiving) by using fs and bs for di, respectively. Their values can be calculated by using Equations 3, 6, 9, and 12 for one data item di instead of using the whole set of data items DI. EMax and TDMax represent the maximum consumed energy and the maximum time delay that are accepted by the mobile user or the system model for each data item di ∈ DI, respectively. Also, w1 and w2 are the weight values for consumed energy and time delay, respectively, such that

$$\mathbf{w_1 + w_2 = 1} \qquad (15)$$

These weight values represents the importance degree of the consumed energy and the time delay for the mobile user.

By using Equations (13) and (14), the total weighted accumulated costs by using fs and bs are defined as follows.

$$\mathbf{TC_{fs}(DI)} = \sum_{\mathbf{d_i \in DI}} \mathbf{tc_{fs}(d_i)} \qquad (16)$$

and

$$\mathbf{TC_{bs}(DI)} = \sum_{\mathbf{d_i \in DI}} \mathbf{tc_{bs}(d_i)} \qquad (17)$$

3.3 Problem Formulation

In MCC, the main goals of DMSS problem are (1) minimizing the energy consumed for sending and retrieving data and (2) minimizing the time delay which spent for sending and retrieving data. So, based on the previous described cost model in section 3, the goal is finding the set of data items, Dbs ⊆ DI, which will be stored in the block server bs and the set of data items, Dfs⊆ DI, which will be stored in the file server fs. Such that there is no any data item can be mapped to bs and fs at the same time. Therefore, The **DMSS** can be formulated as follows:

Objective: find D_{bs} ; D_{fs}

such that

$$|DI| > 0 \qquad (18)$$

$$TC_{bs}(D_{bs}) + TC_{fs}(D_{fs}) \le TC_{bs}(DI) \qquad (19)$$

$$TC_{bs}(D_{bs}) + TC_{fs}(D_{fs}) \le TC_{fs}(DI) \qquad (20)$$

$$DI = D_{bs} \cup D_{fs}, \qquad D_{bs} \cap D_{fs} = \emptyset \qquad (21)$$

Where, Equation 18 means that the set of data items is not empty. Equation 19 means that the total cost of mapping data items to the file and block servers is less than or equal the total cost of mapping data items to the block server only. Equation 20 means that the total cost of mapping data items to the file and block servers is less than or equal the total cost of mapping data items to the file server only. Equation 21 means that some of data items are mapped to block server and the other data items are mapped to file server. Also, there is no any data item can be mapped to both server in the same time.

4. ADAPTIVE DATA MAPPING SELECTION ALGORITHM

In this section, to solve the DMSS problem which was formulated in the previous section, a new mapping selection Algorithm called Adaptive Data Mapping Storage Selection Algorithm **(ADMSSA)** is proposed.

4.1 Basic Idea

To solve the DMSS problem, the basic idea of ADMSSA is based on the following three issues: (a) determining the importance degree of consumed energy and time delay for each data item di ∈ DI by defining the values of w1 and w2, (b) estimating the total mapping cost of each data item di ∈ DI on the file and block servers by calculating their costs which were determined by using Equations (13) and (14), and (c) comparing the calculated costs and selecting the most appropriate mapping server based on the needs of a mobile user (e.g., minimum mapping costs).

4.2 The Proposed Algorithm

To satisfy the basic idea of ADMSSA, in this section, the proposed algorithm and its description will be presented.

ADMSSA consists of two phases: (1) Estimating phase which estimates and predicts the energy and time delay costs for all generated data items by mobile device with the user of file and block mapping servers, separately, and (2) Selecting phase which select the best appropriate mapping server for each generated data item that satisfy the user needs. The steps of these two phases are described as follows.

A) **Estimating phase**: this phase consists of the following steps

A1. Collecting the profile information for the file and the block servers (*fs* and *bs*).

A2. Defining values of w_1 and w_2 for data item d_i.

A3. Predicting the consumed energy cost, $tec_{fs}(d_i)$ and time delay cost, $ttdc_{fs}(d_i)$ for d_i by

A4. Calculating the file server mapping cost $tc_{fs}(d_i)$ and the block server mapping cost, $tc_{bs}(d_i)$ by using Equations 13 and 14.

B) **Selecting phase**: this phase consists of the following steps

B1. Comparing the calculated file server mapping cost, tcfs(di) with the calculated block server mapping cost, tcbs(di).

B2. Selecting the mapping server with the minimum mapping cost for data item di.

B3. Sending the data item di to the selected mapping server.

Fig. 3. shows the architecture of ADMSSA phases

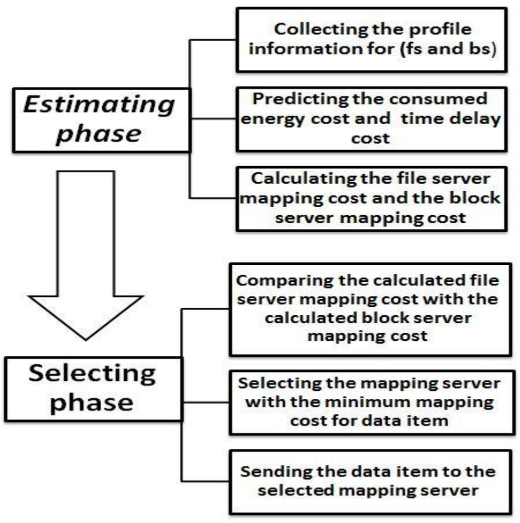
Estimating phase
Collecting the profile information for (fs and bs)
Predicting the consumed energy cost and time delay cost
Calculating the file server mapping cost and the block server mapping cost
Selecting phase
Comparing the calculated file server mapping cost with the calculated block server mapping cost
Selecting the mapping server with the minimum mapping cost for data item
Sending the data item to the selected mapping server

Fig. 3. The proposed ADMSSA phases

Algorithm 1 shows the steps of ADMSSA with its two phases. As shown in Algorithm 1, the input parameters for ADMSSA are the set of data items DI, the file server fs, the block server bs, the weight values w1 and w2, the maximum accepted consumed energy EMax, and the maximum accepted time delay TDMax. In steps 1 and 6, ADMSSA collects all profiles information (e.g., # of stored files or blocks, processing power, bandwidth, etc.) for bs and fs in profbs and proffs , respectively. In steps 3 to 6, ADMSSA initializes the sets of mapping data items (Dfs and Dbs) and the estimated total costs (TCfs and TCbs) for fs and bs, respectively. In steps 8 to 13, ADMSSA executes the steps of its estimating phase by predicting and calculating the mapping cost for each data item di ∈ DI. In steps 14 to 22, ADMSSA executes the steps of its selecting phase by comparing mapping cost of the file and the block servers for each data item di ∈ DI and filling the sets of mapping data items (Dfs and Dbs) based on the result of the comparison as shown in steps 15 and 19. Also, in steps 16 and 20, ADMSSA adds the mapping cost of di to the estimated total costs for fs or bs. Then, ADMSSA sends the mapped data item to the selected server as shown in step 17 in case of fs is selected and in step 21 in case of bs is selected.

Finally, ADMSSA produces the set of mapped data items to the file server, Dfs, the set of mapped data items to the block server, Dbs, the mapped cost for the file server, TCfs(Dfs), and the mapped cost for the block server, TCbs(Dbs).

Algorithm 1: Adaptive Data Mapping storage Selection Algorithm

[ADMSSA]

```
Input: A set of data items DI, a file server fs, a block server bs, w1, w2, EMax, and TDMax
1: profbs ← collects bs profile
2: proffs← collects fs profile
3: Dfs ← ∅
4: Dbs ← ∅
5: TCfs(Dfs) ← 0
6: TCbs(Dbs) ← 0
7: for each di ∈ DI
8: tecfs(di) ← se(di; fs) + re(di; fs)
9: tecbs(di) ← se(di; bs) + re(di; bs)
10: ttdcfs(di) ← std(di; fs) + rtd(di; fs)
11: ttdcbs(di) ← std(di; bs) + rtd(di; bs)
12: tcfs (di) ← $\mathbf{w_1}\frac{\mathbf{tec_{fs}(d_i)}}{\mathbf{E_{Max}}} + \mathbf{w_2}\frac{\mathbf{ttdc_{fs}(d_i)}}{\mathbf{TD_{Max}}}$
13: tcbs (di) ← $\mathbf{w_1}\frac{\mathbf{tec_{bs}(d_i)}}{\mathbf{E_{Max}}} + \mathbf{w_2}\frac{\mathbf{ttdc_{bs}(d_i)}}{\mathbf{TD_{Max}}}$
14: if (tcfs(di) ≤ tcbs(di)) then
15:     Dfs ← Dfs ∪ {di}
16:     TCfs(Dfs) ← TCfs(Dfs) + tcfs(di)
17:     send(di ; fs)
18: else
19:     Dbs ← Dbs ∪ {di}
20:     TCbs(Dbs) ← TCbs(Dbs) + tcbs(di)
21:     send(di; bs)
22: end if
23: end for
Output: : Dfs, Dbs, TCfs(Dfs), and TCbs(Dbs).
```

As a result, by using ADMSSA the mobile device can select adaptively and dynamically the most appropriate mapping cloud server for each data item to satisfy the user needs as minimizing the energy consumption and the time delay of sending and receiving data to/from the cloud servers.

5. EXPERIMENTAL RESULTS AND ANALYSIS

This section evaluates the performance of the proposed method for data mapping selection using ADMSSA algorithm through comparing it with two mapping methods, FMAP method: which maps all data items to file server only and, BMAP method: which maps all data items to block server only. The OMNet ++ [19] simulator was used to evaluate the proposed algorithm ADMSSA. Also, each experiment is repeated 5 times and the average was taken.

5.1 Time Delay Cost

Fig. 4 shows the time delay cost against different number of data items when the number of mobile hosts is fixed to 2 nodes, the time delay cost increases as number of data items increases. This is because a large number of data items needs more time in sending and receiving processes. Also, the time delay cost for ADMSSA is less than BMAP method and is larger than FMAP method. This is because block server mapping needs more time delay cost for classification and searching in sending and receiving processes and this time delay cost will be increased in ADMSSA if the number of mapped data items to block server increases.

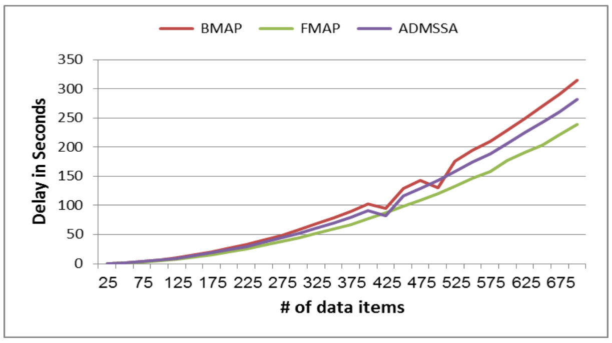

Fig. 4. The time delay cost vs. number of data items

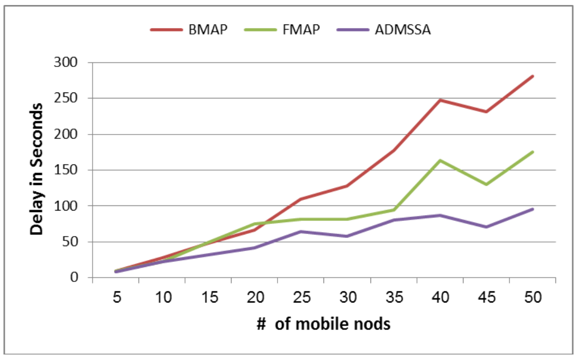

Fig. 5. The time delay cost vs. number of mobile nodes

Fig. 5 shows the time delay cost against different number of mobile hosts when the number of data items is fixed to 25 data items, the time delay cost increases as number of mobile hosts increases. This is because the existence of large number of mobile hosts generates a high number of data items in total. Therefore, the number of mapped and stored data items in the file and block servers will increase which will affect on the time delay cost of sending and receiving processes for each mobile host. Also, the time delay cost for ADMSSA is much lower than FMAP and BMAP methods. This is because ADMSSA can select the best mapping server for a fixed number of generated data items by each mobile host.

5.2 Energy Consumption Cost

Fig. 6 shows the energy consumption cost against different number of data items when the number of mobile hosts is fixed to 2 mobile nodes, the energy consumption cost increases as number of data items increases. This is because a large number of data items consumes more energy in sending and receiving processes. Also, the energy consumption cost for ADMSSA is less than FMAP method and it is noticed to be larger than BMAP method. This is because file server mapping consumes more energy consumption cost in sending and receiving processes and this energy consumption cost will be increased in ADMSSA if the number of mapped data items to file server increases.

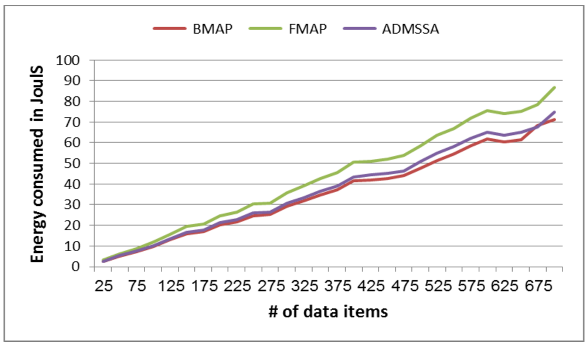

Fig. 6. The energy consumption cost vs. number of data items

Fig. 7 shows the energy consumption cost against different number of mobile hosts when the number of data items is fixed to 25 data items, the energy consumption cost increases as number of mobile hosts increases, this is because the existence of large number of mobile hosts generating a high number of data items in total. Therefore, the number of mapped and stored data items in the file and block servers will increase which will affect on the energy consumption cost of sending and receiving processes for each mobile host. Also, the energy consumption cost for ADMSSA is much lower than FMAP method and is almost the same as BMAP method. This is because ADMSSA can select the best mapping server for a fixed number of generated data items by each mobile host.

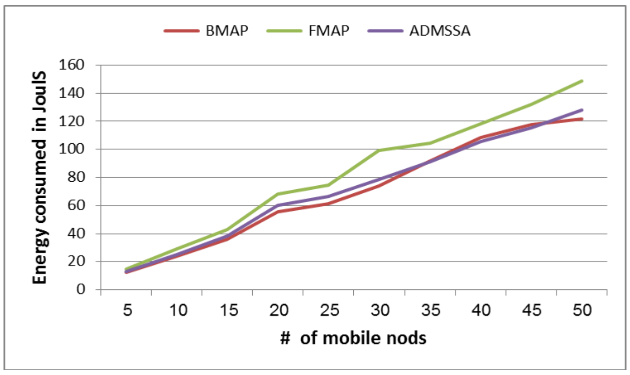

Fig. 7. The energy consumption cost vs. number of mobile nodes

5.3 Total Cost

Figs. 8 and 9 show the total cost which is calculated by Equations 16 and 17 against different number of data items and number of hosts, respectively.

Fig. 8 shows the total cost against different number of data items when the number of mobile hosts is fixed to 2 mobile nodes, the total cost increases as number of data items increases and the total cost for ADMSSA is less than FMAP and BMAP methods which satisfies the required conditions in Equations (19) and(20).

Fig. 9 shows the total cost against different number of mobile hosts when the number of data items is fixed and it was 25 data items. As shown in Fig. 9 the total cost increases as number of data items increases and the total cost for ADMSSA is much lower than FMAP and BMAP methods which satisfies the required conditions in Equations 19 and 20.

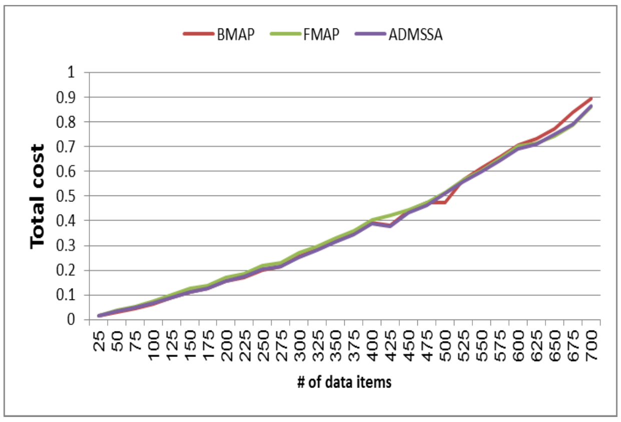

Fig. 8. The total cost vs. number of data items

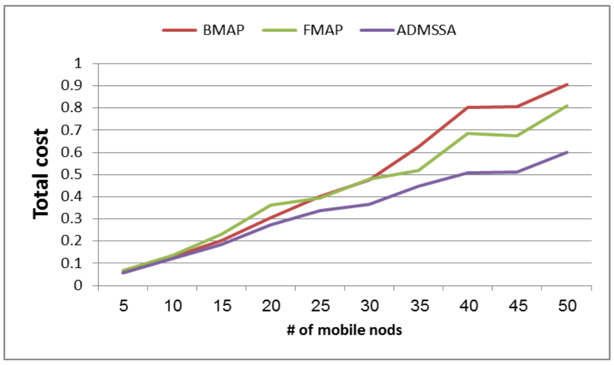

Fig. 9. The total cost vs. number of mobile nodes

Based on these results, ADMSSA can select the most appropriate mapping server adaptively and dynamically by taking into account the time delay and energy costs for each data items and is much better than FMAP and BMAP methods.

6. CONCLUSIONS

In this paper, a new data mapping selection algorithm in MCC called Adaptive Data Mapping Selection Algorithm (ADMSA) is proposed to solve the data mapping storage selection problem in mobile cloud computing (MCC) and to minimize the total energy consumption for data mapping in MCC. ADMSA constructs a cost model by taking into account the time delay and energy consumption costs for sending and retrieving data to/from the cloud servers. Also, many simulation experiments were conducted to evaluate ADMSSA by comparing it with the file and block mapping methods. The simulation results show that the proposed algorithm achieves a better mapping performance with minimum cost compared to the mapping data items without any selection mechanism. In the future work, the proposed method will be modified by adding more metrics in its cost model as data priority or sever trust. Also, the existence of more than one file and block server in the cloud side will be studied.

7. ACKNOWLEDGMENTS

I would like to thank Assoc. Prof. Neveen Ghal for her guidance, efforts, and encouragement. And would like to express my special thanks of gratitude to my teacher Dr. Asaad Ahmed Gad-El-Rab who gave me the golden opportunity to do this work, Without help, this work would have never been possible now came to know about so many new things I am really thankful to him.

8. REFERENCES

[1] Lindsay, B.G., Haas, L.M., Mohan, C., Wilms, P.F. and Yost, R.A.,Computation and communication in R*: a distributed database manager,. ACM Trans. Comput.Syst., Vol. 2(1), pp. 24-38 ,1984.

[2] Rothnie Jr., J.B., Bernstein, P.A., Fox, S., Goodman, N., Hammer, M.,Landers, T.A., Reeve,C.L., Shipman, D.W., Wong, E., Introduction to aSystem for Distributed Databases (SDD-1), ACM Trans. Database Syst.,Vol 5(1), pp. 1-17, 1980.

[3] Ramandeep Singh Rajpal and RaghvendraKumar, Secured Communication Model for Mobile Cloud Computing,International Journal of Computational Engineering Research (IJCER),Vol. 5(5), 2015

[4] Jin Li, Xiao Feng Chen, Xining Huang, Shaohua Tang and Yang Xiang,Secure Distributed Deduplication Systems with Improved Reliability,IEEE Transactions on Computers, pp. 1-12, 2015.

[5] Jin Li, Yan Kit Li, Xiaofeng Chen, Patrick P.C. Lee and Wenjing Lou,A Hybrid Cloud Approach for Secure Authorized Deduplication, IEEEtransactions on parallel and distributed systems, Vol. 26(5), pp. 1206-1216, 2015.

[6] Mazhar Ali, Kashif Bilal,Samee U. Khan,BharadwajVeeravalli,KeqinLiand Albert Y. Zomaya,DROPS:Division and Replication of Data in Cloudfor Optimal Performance and Security, IEEE Transactions on CloudComputing, pp. 1-15, 2015.

[7] Jian Liu, Kun Huang, Hong Rong, Huimei Wang, and Ming Xian,Privacy-Preserving Public Auditing for Regenerating-Code-Based Cloud Storage,IEEE Transactions on Information Forensics and Security,Vol. 10(7),2015.

[8] Bernstein, P.A., Hadzilacos, V. and Goodman, N., Concurrency Controland Recovery in Database Systems, Addison Wesley, Reading, assachusetts,1987.

[9] Weikum, G. andVossen, G.,Transactional information systems: theory,algorithms, and the practice of concurrency control and recovery,MorganKaufmann Publishers Inc. ,2001.

[10] Niroshinie Fernando, Seng W. Loke and Wenny Rahayu , Mobile cloudcomputing: A survey, Future Generation Computer Systems, Vol. 29, pp.84 -106, 2013.

[11] Xinwen Zhang and Anugeeth aKunjithapatham, Towards an ElasticApplication Model for Augmenting the Computing Capabilities of MobileDevices with cloud Computing in Mobile NetwAppl, Springer Science+Business Media, Vol. 16(3), pp. 270 -284, 2011.

[12] Chandra, P. Bahl and Maui, Making smartphones last longer withcode offload, Proceedings of the 8th International Conference on MobileSystems, Applications and Services,MobiSys10, ACM, New York, NY,USA , pp. 49-62, 2010.

[13] Amazon Web Services (AWS), Available from: http://aws.amazon.com[Accessed:13th Mar. 2016].

[14] N. Gohring, Amazons S3 down for several hours, Online at:http://www.pcworld.com/businesscenter/article/142549/amazonss3down for several hours.html, 2008.

[15] Mell P and Grance T. , The NIST definition of cloud computing,National Institute of Standards and Technology, NIST special publication800-145, 2011.

[16] T. S. J. Schwarz and E. L. Miller, Store, Forget, and Check: Using Algebraic Signatures to Check Remotely Administered Storage, Proceedingof ICDCS 06, pp. 12-12, 2006.

[17] Lindsay, B.G., Haas, L.M., Mohan, C., Wilms, P.F., Yost, R.A., Computation and communication in R*: a distributed database manager, ACM Trans. Comput.Syst. Vol. 2(1), pp. 24 -38 1984.

[18] Rothnie Jr., J.B., Bernstein, P.A., Fox, S., Goodman, N., Hammer, M.,Landers, T.A., Reeve,C.L., Shipman, D.W. andWong, E., Introduction to aSystem for Distributed Databases (SDD-1), ACM Trans. Database Syst.,Vol. 5(1), pp. 1-17, 1980.

[19] A. Varga and Andrs, The OMNeT++ Discrete Event Simulation System,,Proceedings of the European Simulation Multi-conference (ESM'2001).Prague, Czech Republic, June 2001.