International Journal of Computer and Information Technology (ISSN: 2279 – 0764)
Volume 02– Issue 03, May 2013

Scheduling Manager for Mobile Cloud Using Multi-Agents

Naif Aljabri*
*naifkau {at} gmail.com

Fathy Eassa

Abstract- **Mobile Cloud Computing (MCC) is emerging as one of the most important branches of cloud computing. In MCC, a group of mobile devices connected via WLAN and cooperating to execute the user's jobs and acting as a computing resource provider. However, unlike could computing (which use a powerful servers to execute the jobs) , mobile devices are not reliable to execute extensive jobs in the MCC due to its limitation of battery and connectivity. To overcome this limitation and keep the execution of the job up and running even if the mobile device leave the cloud, we need to reschedule the job and continue the execution on another available mobile device in the MCC. Firstly, the impact of job and node characteristic on the performance of the MCC will be discussed. Secondly, we introduce a new scheduling techniques used by the scheduling manager component in the MCC. At last, evaluate and compares our approach with similar approaches. All of these may contribute to make the MCC more efficient, reliable and feasible to be used as a powerful cloud platform to execute the user's extensive jobs.**

Keywords: Mobile cloud computing, job scheduling, Multi-agents , mobile device , offloading.

I. INTRODUCTION

The use of mobile phones has increased dramatically in recent years. The number of mobile users reached 6 billion in 2012 [1]. At the same time, the computing power of mobile devices has increased, and recently we saw a multicore processor with 1 GB memory with support of LTE (4G networks) enter the market. For example, Apple iPhone 5 comes with Dual-core 1.2 GHz and 1 GB RAM. Also, Samsung I9300 Galaxy S III comes with Quad-core 1.4 GHz and 1 GB RAM [2]. These advancements in performance and computing power make use of mobile devices as computing resource provider feasible [2],[3].

However, these advancements in performance and application have resulted in other problems with mobile phones, specifically the power consumption and connectivity (due to mobility) . This problem has been addressed by researchers by computation offloading: sending heavy computation to resourceful servers and receiving the results from these servers [4] [5]. A similar concept used in this paper without the use of a resourceful servers. Our proposal depends on the cooperation between the mobile devices itself to execute the jobs in the MCC.

Several definitions exist for MCC, with different research alluding to varying concepts of the MCC. The term MCC is defined as: a mobile device that functions as a thin client, using the Internet to connect to the remote cloud in order to access applications running on the resource rich servers located in the cloud [6]. The user will pay for accessing a remote cloud via 3G internet connection. For example, Google Gmail mobile edition, Facebook's location aware services, and Twitter for mobile, mobile weather widgets etc. Fig. 1 illustrates this approach.

Another definition of MCC is the cloudlet concept. A cloudlet consists of several resource-rich computers, or a cluster of computers that are connected to a remote cloud server. Typically, these cloudlets would be situated in crowded areas such as coffee shops [7]. The mobile devices can connect and function as a thin client to the cloudlet, as opposed to a remote cloud server, and can offload its workload to the local 'cloudlet'. "Fig. 2" illustrates this approach.

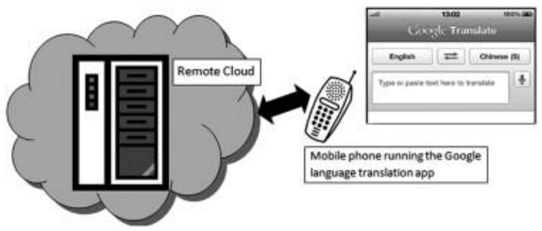

Figure 1. A remote cloud server catering to mobile devices though the internet.

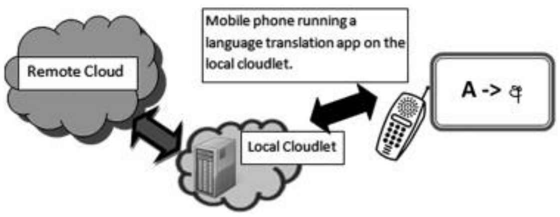
Remote Cloud
Mobile phone running a language translation app on the local cloudlet.
Local Cloudlet
A -> ϥ

Figure 2. A virtual resource cloud made up of mobile devices in the vicinity.

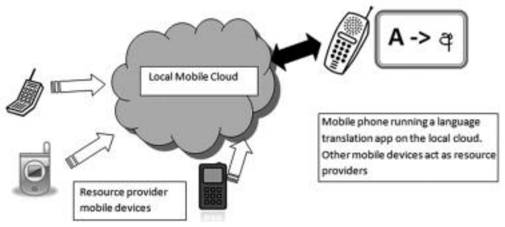
Local Mobile Cloud
A -> क
Mobile phone running a language translation app on the local cloud. Other mobile devices act as resource providers
Resource provider mobile devices

Figure 3. A virtual resource cloud made up of mobile devices in the vicinity

The third definition of MCC is to consider mobile devices as resource providers for the cloud. In this case, collective resources of various mobile devices in the local vicinity would be utilized using a peer-to-peer network [8]. This approach is based on component model systems representing systems made up of interoperable local components, rather than offloading jobs to the remote cloud [9]. In most research, this is called Virtual Mobile Cloud or Local Mobile Cloud [10]. This paper focuses primarily on this approach, illustrated by “Fig. 3”.

Based on the third approach described above, we define the MCC as “The aggregation of mobile devices connected via wireless network that cooperate to execute user’s jobs" [5],[17].

The MCC utilizes key strengths of cloud computing, which can be described in terms of the services offered by cloud service providers: software as a service (SaaS), platform as a service (PaaS), and infrastructure as a service (IaaS) [11].

In this paper, we present two rescheduling techniques used by the scheduling manager component in the MCC. The proposed MCC scheduling manager component is implemented and managed using multi-agents software technology. Agent software technology has a lot of advantages including mobility, adaptability, transparency, ruggedness and self-start [13] [14]. Multi-agents approach allows the system designer to implement the system, with each agent specialized for a particular task. For example, an e-commerce application might be used simultaneously by buyer agents, seller agents, stocking agents, database agents, email agents, etc. All of these agents need to communicate with each other and must have the capability of working together to achieve a common set of goals [12].

II. RELATED WORK

Integration between mobile devices and cloud computing is presented in several previous works. Marinelli [6] introduce Hyrax, a mobile cloud computing client that allow mobile devices to use cloud computing platforms. The platform derived from Hadoop that supports cloud computing on Android smartphones. The main focus of this work is to port a client application into a mobile device to enable the integration. The author introduce the concept of using mobile devices as resource provider, but further experimentation is not included. Similar work has been done in gird computing. Luo [18] introduces a framework that allow using of the cloud computing power to enhance the mobile device capabilities. The author introduce a new task partitioning schema and show the feasibility of such implementation. The best point in this paper is using the cloud to back mobile computing. Chun and Maniatis [19] introduce the use of cloud computing to execute mobile applications on behalf of the device. they create a clone VMs to run the applications in the same way that they run on mobile device to avoid inconsistencies produced when run part of the application in different infrastructure. They introduce the same concept of distributed file system. Huerta-Canepa and Lee [17] introduce the feasibility of using mobile devices resource provider. Their concept is to utilize the computing power of the mobile devices to introduce a virtual cloud computing. The best point in this paper is avoiding a connection to infrastructure-based cloud provider. Their results don’t show speedup, but they save the power consumption, since the processing time is less than when executed in a single mobile device. Finally, the impact of the job and node characteristic on cloud efficiency is not discussed in the previews works.

III. SCHEDULING MANAGER FRAMEWORK ARCHITECTURE

The proposed scheduling manager framework in this paper consists of laptop and a group of mobile devices (nodes) connected via WLAN and acting as MCC. This environment uses agent software technology to manage the jobs inside the MCC. Since this system is written completely in Java, it is portable and allows the execution of the agents across multiple heterogeneous mobile platforms. agents communicate and interact by exchange of messages. The cloud managed by three main agents located in the laptop. these agents called (*Scheduling Manager*, *Resource Manager* and *Job Handler*). in addition , there is one tracking agent called *Node Tracker* located in every node (mobile device) inside the cloud. “Fig. 4”, illustrates the scheduling manager environment.

Due to availability problem with mobile devices, the laptop is used to keep the management of the cloud centralized and available all the time. this will give the MCC a strong control of the running jobs on the mobile devices.

A. Framework Components

Our proposed framework consists of four agents (*Scheduling Manager, Resource Manager* , *Job Handler* and *Node Tracker*). All these agents are located in the laptop except the last one which located in the mobile device. “Fig. 5”, illustrates the Framework Components.

Every agent is assigned a certain role as follows :

- *Resource Manager:* every new node connected to the cloud will communicate with this agent to provide the node characteristics such as: CPU speed , battery level, network signal level and node stability. Then, the *Resource Manager* give a rate value for the new node based on the node characteristics. This rate value is very important to the *Scheduling Manager* to decide which node will be assigned based on the job size. If the job size is large, then the *Scheduling Manager* use a high rate node. A low rate node used if the job size is small. This will reduce the job rescheduling and increase the cloud efficiency.

- *Job Handler:* this agent will handle the jobs that submitted to the cloud and pass them to the execution queue. If the job size is larger than the average of the nodes memory capacity, the *Job Handler* split this job to smaller parts.

- *Scheduling Manager:* The main role for this agent is assigning the jobs to the nodes. This agent read the jobs from the execution queue. Then, based on the job size, assign the job the proper idle node (Every idle node has a rate value as discussed above). When using the *Recovery Technique* , all the assigned nodes communicate with this agent and send the job stat periodically. If the communication lost with the node , then the *Scheduling Manager* assign the last job stat to another idle node. In case of using the *Fault-tolerance Technique,* the *Scheduling Manager* assign each job to two nodes and sends a periodic heartbeat call to the *Node Tracker* in the two nodes to show that the nodes are alive. if one of the nodes lost (node leave the cloud) , the Scheduling *Manager* read the current job stat from the active node and assign it to a new idle node to continue the job execution together with the active node.

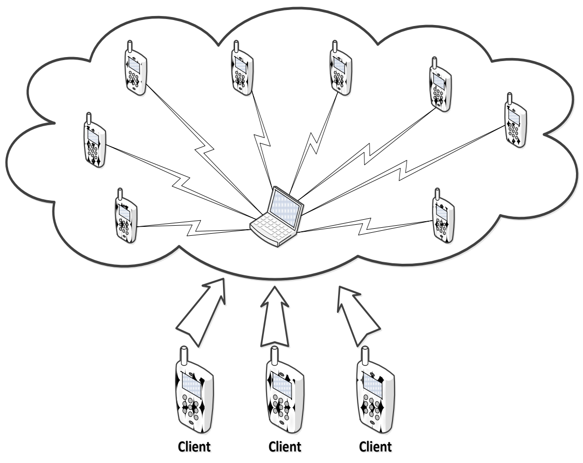

Figure 4. The Scheduling Manager Framework

- *Node Tracker:* the main role of this agent is tracking the node (mobile device) and provide some information about the node to the *Resource Manager* and *Scheduling Manager* . this agent communicate with the *Resource Manager* if a new node connected to the cloud . In this case , The *Node Tracker* send the node characteristics such as: battery level, network signal level, CPU speed and node stability to *Resource Manager*. The *Resource Manager* need these information to give the new node a rate value based on the node characteristics . after the node assigned a job, the *Node Tracker* switch the communication to *Scheduling Manager* . In this case, the *Node Tracker* send the job status to the *Scheduling Manager* periodically. "Fig. 5", Shows the Main Framework Components.

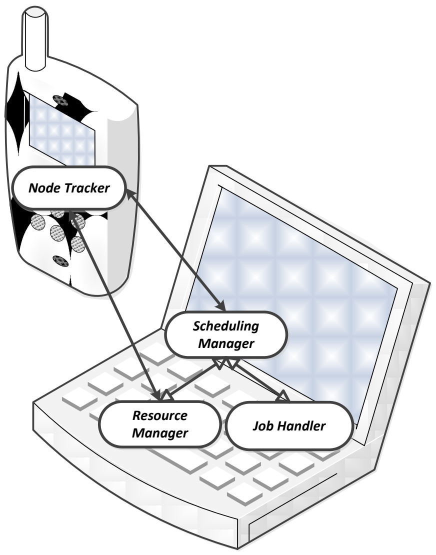

Figure 5. Framework Components

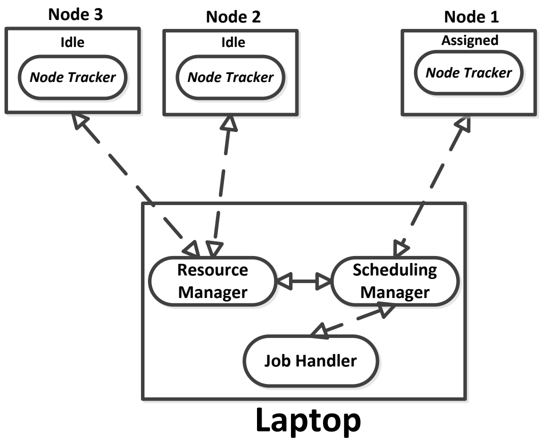

Figure 6. The Agent-Based Architecture for the proposed environment

IV. SCHEDULING TECHNIQUES

Scheduling techniques defined as the decision that should be taken by *Scheduling Manager* when the assigned node leave the cloud. in this paper we propose two scheduling techniques. These techniques are:

A. Recovery Technique

In this technique, the *Scheduling Manager* assign the jobs to the proper node based on the job size and the node rate value. While the job is running, the *Node Tracker* communicate with *Scheduling Manager* and sends the job stat periodically. The *Scheduling Manager* will save the stat temporary until the job is done. During the job execution, if the communication lost with the *Node Tracker* (node leave the cloud) , the *Scheduling Manager* assign the last job stat to a new idle node and continue the job execution from the last stat. “Fig. 7”, shows the scenario for this technique and “Fig. 9”, illustrate the activity diagram for this technique.

B. Fault-tolerance Technique

In this technique, the *Scheduling Manager* use two nodes for each job. Every node run a copy of the same job. The *Scheduling Manager* use a periodic heartbeat call to the *Node Tracker* to show that the nodes are alive. if one of the nodes lost (node leave the cloud) , the *Scheduling Manager* read the current job stat from the active node and assign it to a new idle node to continue the job execution together with the active node. “Fig. 8”, shows the scenario for this technique and “Fig. 10”, illustrate the activity diagram for this technique.

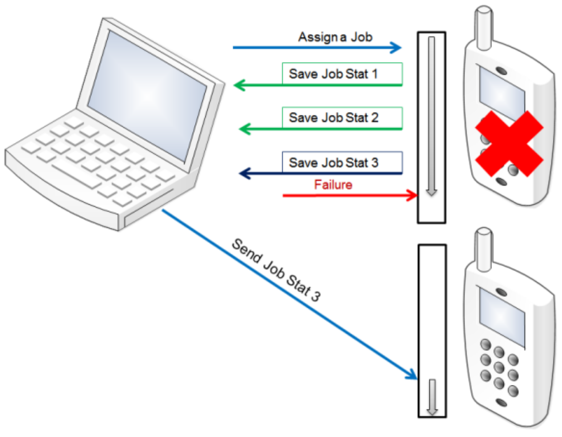

Figure 7. Recovery Technique Scenario

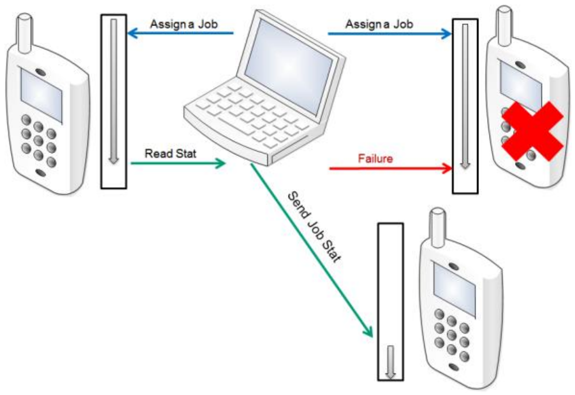

Figure 8. Fault-tolerance Technique Scenario

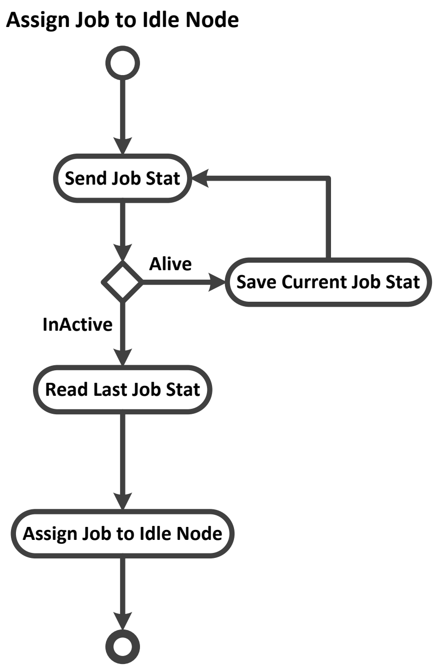

Figure 9. Recovery Technique Activity Diagram

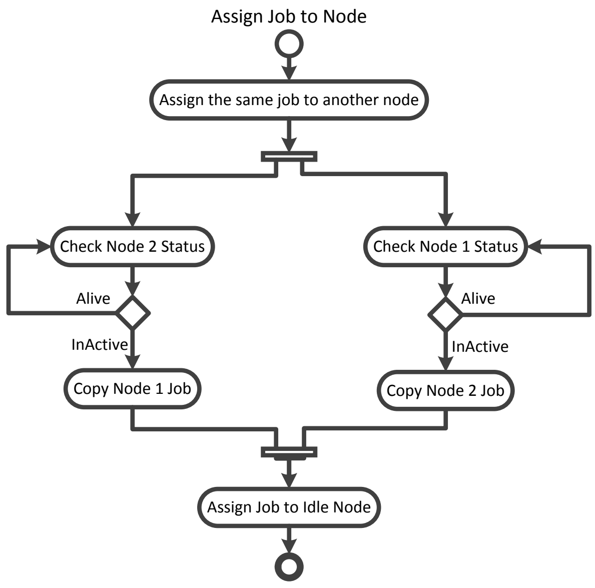

Figure 10. Fault-tolerance Technique Activity Diagram

V. Evaluation Setting

Since our framework is proposal , we use assumption-based evaluation setting to evaluate our proposed framework efficiency. We consider that we have 5 nodes and one laptop connected through WLAN with transfer speed of 100Mbps. The *Node Tracker* agent is installed on every mobile device and the main agents (*Scheduling Manager, Resource Manager and Job Handler)* are installed on the laptop to manage the cloud. Every node has different processing speed and rated by the *Resource Manager* as shown in "Table 1". The rate value based on CPU speed only. We assume that the *Job Handler* split two large jobs (job1 = 600 kb and job2 =900 kb respectively) to be 5 jobs in the queue with size between 100kb to 500kb as shown in "Table 2". The time require to process the entire job without any failure defined as in (1) and the required time to transfer the job from the failed node to the new idle node is defined as in (2) for each occurrence. Here we assume the node is failed after 50% of the job execution. So the *job_size* used in (2) is 50% of the remaining job size. Furthermore, we assume different values for the number of failure occurs in each job.

$$comp\_time = \frac{job\_size \times node\_rate}{cpu\_speed} \qquad (1)$$

$$transfer\_time = \frac{job\_size}{bandwidth} \qquad (2)$$

For the job monitoring, the periodic time for sending the job stat from *Node Tracker* to *Scheduling Manager* assumed by ¼ of *comp_time* which approximately ¼ of *job_size* with *Recovery Technique*. For example , if the *job_size=100 kb* the *Node Tracker* will send the job stat 3 times in (75 kb , 50 kb and 25 kb) . The total processing for each job in the cloud using the *Recovery Technique* can be defined as in (3) . The zero value assumed for the periodic heartbeat call to the *Node Tracker* when using the *Fault-tolerance Technique* (since no job stat reading). Equation (3) is used to calculate the total processing time when using the *Fault-tolerance Technique* except the last part of the equation as defined in (4).

$$total\_procss = comp\_time + \sum_n^1 \frac{job\_size}{(bandwidth \times 2^n)} + \frac{6 \times job\_size}{bandwidth \times 4} \qquad (3)$$

$$total\_procss = comp\_time + \sum_n^1 \frac{job\_size}{(bandwidth \times 2^n)} \qquad (4)$$

Where *n* is the number of failures during the job execution.

TABLE 1. The nodes proccessing speed

Node Name	**Node CPU speed**	**Node Rate**
Node1	1.2 GHz	1
Node2	0.8 GHz	3
Node3	0.4 GHz	5
Node4	1 GHz	2
Node5	1.2 GHz	1

TABLE 2. The job list in the queue

Job Name	**Job Size**
Job11	100 kb
Job12	200 kb
Job13	300 kb
Job21	400 kb
Job22	500 kb

VI. RESULTS

The results in “Table 3” and “Table 4” shows the MCC processing results using the *Recovery Technique* and *Fault-tolerance Technique*. We notice that the *Scheduling Manager* assign the large jobs to the a high rate nodes. For example, job22 is given to Node5 , the total processing time is 416. If we give this job to node3 , the total processing time is 6250. We always get the optimum time based on the jobs and nodes characteristics. Furthermore, in case of failure, the large jobs take longer time to transfer to a new node. To minimize this time, we need to achieve large jobs faster. In the similar works [6],[18],[19],[17] assigning based on job and node characteristic is not discussed.

Furthermore, we notice that the Periodic and node Transfer time (column 5,6 in “Table 3” and column 5 in “Table 4”) values are very small. The reason is that we use small jobs with large WLAN transfer rate (100 Mbps). We can get better results if we use larger jobs or using different network architecture as in similar works [6],[18],[19],[17].

A. *Recovery Technique Results*

The results in “Table 3” shows the MCC processing results using *Recovery Technique*. . The fourth column is the number of failures occurs during the job execution. The Periodic Transfer is the time used to send the job stat from the *Node Tracker* to the *Scheduling Manager*. The last column is the time used to transfer the last job stat of the failed node to the new idle node. As defined in (3), the total processing time is the total of the column 3,4 and 5 is approximately 3819. As discussed before, It is the optimum time based on the jobs and nodes characteristics.

B. *Fault-tolerance Technique Results*

The results in “Table 4” shows the MCC processing results using *Fault-tolerance Technique*. Here there is no periodic time as discussed in “Session V”. In this technique, the *Scheduling Manager* assign two nodes for each job. We notice that in ”Table 4”.

First, the largest jobs (Job21, job22) assigned to the 4 nodes (high rate nodes). Then after job21 finished, job11 assigned to Node1 and Node4 , and so on. As discuses before, we notice that the *Scheduling Manager* assign the large jobs to the high rate nodes to get the optimum processing time. Since each job is assigned to two nodes, this technique needs large number of nodes to give better performance .

From the table, we notice that job11 waiting time is approximately 333 (the processing time of job21). Also, the waiting time for job12 and job13 is 416 (the processing time for job22 and job11). As defined in (3), the total processing (total of the column 3 and 5 and the waiting time for the jobs) is approximately 2413. With this technique we enhance the MCC performance by 37% than the previous technique. “Fig. 11”, shows the performance of *Fault-tolerance Technique* compared to *Recovery technique*.

TABLE 3. THE CLOUD PROCCESSING RESULTS WITH RECOVERY TECHNIQE

Job	**Node**	**Computing Time** (1)	**No. of failures (n)**	**Periodic Transfer** (2)	**Node Transfer** (2)
Job11	Node3	1250	3	0.15	0.0875
Job12	Node2	750	2	0.3	0.15
Job13	Node4	600	1	0.45	0.15
Job21	Node1	800	3	0.6	0.35
Job22	Node5	416	2	0.75	0. 375

TABLE 4. THE CLOUD PROCCESSING RESULTS WITH FAULT-TOLERANCE TECHNIQUE

Job	**Node**	**Computing Time** (1)	**No. of failures**	**Node Transfer** (2)
Job21	Node1	333	1	0.2
Job21	Node4	800	2	0.3
Job22	Node5	416	2	0.375
Job22	Node2	1875	3	0.4375
Idle	Node3	-	-	-
Job21 finished (Node1 and Node4 released) start Job11				
Job11	Node1	82	3	0.0875
Job11	Node4	166	2	0.075
Job11, job22 finished (All nodes released)				
Job12	Node5	166	2	0.15
Job12	Node2	750	1	0.1
Job13	Node1	250	3	0.2625
Job13	Node4	600	2	0.225
Idle	Node3	-	-	-

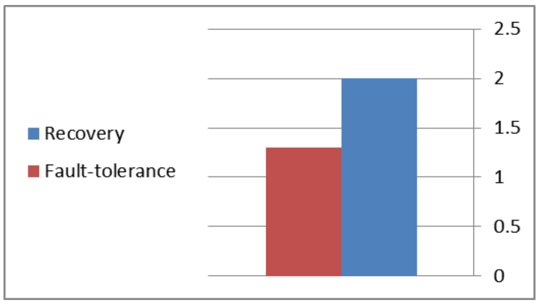

Figure 11. Performance of Fault-tolerance Technique compared to Recovery technique

VII. CONCLUSIONS

In this paper, we show how the impact of the job and node characteristic plays a major role in the MCC efficiency. The optimum processing time is guaranteed if we assign the large jobs to high rate nodes. Moreover, we can reduce the failures occur in the MCC. We believe this will give a significant improvement if used with other frameworks discussed in related works [6],[18],[19],[17]. Furthermore, we introduce two rescheduling techniques that can be used to enhance the MCC efficiency. In our results, the *Recovery Technique* doesn't show a significant improvement to the optimum processing time. With *Fault-tolerance Technique,* we enhance the MCC performance by 37%.

We believe these techniques will give better performance if used with other frameworks in the related work. For example, in [17] the job rescheduling take about 44% of the processing time. we can minimize this number if we use these rescheduling techniques.

Further experiments (using larger jobs and different network architecture) are needed to support our results and show the feasibility of using these techniques in different MCC frameworks.

REFERENCES

[1] International Telecommunication Union, ITU World Telecommunication/ICT Indicators Database, ITU data release June 2012.

[2] GSMArena.com, "Samsung I9300 Galaxy S III Full phone specifications", Accessed March 2013, http://www.gsmarena.com/samsung_i9300_galaxy_s_iii-4238.php

[3] GSMArena.com, "Apple iPhone 4S Full phone specifications", Accessed March 2013, http://www.gsmarena.com/apple_iphone_4s-4212.php

[4] K. Kumar and L. Yung-Hsiang, "Cloud Computing for Mobile Users: Can Offloading Computation Save Energy?," IEEE Computer , vol.43, no.4, pp.51-56, April 2010. doi: 10.1109/MC.2010.98.

[5] W. Vogels "A Head in the Clouds the Power of Infrastructure as a Service." In Proceedings of the 1st Workshop on Cloud Computing and Applications, CCA, 2008.

[6] E.E. Marinelli, Hyrax: Cloud Computing on Mobile Devices using MapReduce, Masters Thesis, Carnegie Mellon University, 2009.

[7] M. Satyanarayanan, P. Bahl, R. Caceres, N. Davies, The Case for VM-based Cloudlets in Mobile Computing, IEEE Pervasive Computing 8, pp.14–23, 2009.

[8] Y. Sasaki, Y. Shibata "A Disaster Information Sharing Method by the Mobile Servers in Challenged Networks." In Advanced Information Networking and Applications Workshops, WAINA, 26th International Conference, pp.1048–1053, 2012.

[9] S. Zachariadis, C. Mascolo, W. Emmerich "Satin: a Component Model for Mobile Self-Organization." In R. Meersman, Z. Tari (Eds.), On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE, in: Lecture Notes in Computer Science, vol. 3291, Springer, Berlin, Heidelberg, 2004, pp. 1303–1321. http://dx.doi.org/10.1007/978-3-540-30469-2_31.

[10] R. Kemp, N. Palmer, T. Kielmann, and H. Bal "Cuckoo: a Computation Offloading Framework for Smartphones." In Proceedings of The Second International Conference on Mobile Computing, Applications, and Services, MobiCASE, 2010.

[11] J. Carolan, S. Gaede, J. Baty, G. Brunette, A. Licht, J. Remmell, L. Tucker, and J. Weise "Introduction to Cloud Computing Architecture." White paper, 2009.

[12] E. Aimeur "Mobile Agents for wireless-commerce applications (MAWA)" Artificial Intelligence in 6261, winter 2003.

[13] Agent Oriented Software Group. The Agent Oriented Software Group (AOS), http://www.agent-software.com/, 2006.

[14] E.M. Kristensen "Agent Technology, tdt 4735 Software Engineering, Depth Study." Master's thesis, NTNU, 2005.

[15]

[16] M. Satyanarayanan "Mobile Computing, Computer 26" pp. 81–82, 1993.

[17] E.E. Marinelli "Hyrax: Cloud Computing on Mobile Devices Using MapReduce," Masters thesis, Carnegie Mellon University, 2009.

[18]

[19] G. Huerta-Canepa and D. Lee "A Virtual Cloud Computing Provider for Mobile Devices." In Proceedings of the 1st ACM Workshop on Mobile Cloud Computing & Services: Social Networks and Beyond, MCS'10, ACM, New York, NY, USA, pages 6:1–6:5, 2010.

[20] X. Luo, "From Augmented Reality to Augmented Computing: A Look at Cloud-Mobile Convergence," International Symposium on Ubiquitous Virtual Reality, vol. 0, 2009, pp. 29-32.

[21] B. Chun and P. Maniatis, "Augmented Smartphone Applications Through Clone Cloud Execution," HOTOS workshop, USENIX, 2009.a