Dynamic Job Scheduling Using Ant Colony Optimization for Mobile Cloud Computing

Rathnakar Achary[1], Dr. V. Vityanathan[1], Dr. Pethur Raj[2], S. Nagarajan[1]

[1]Department of CSE, SASTRA UNIVERSITY, Thanjavur, India
[2]Infrastructure Architect, IBM Global Cloud CoE IBM, India, Bangalore
a.rathnakar@gmail.com

Abstract: Cloud computing has been recognized as one of the prominent new computing paradigm. The ability of cloud to provide on demand access to software (SaaS), application platform (PaaS) and infrastructure (IaaS) in the form of scalable services has attracted considerable interest in the industry. With the current scenario there is no doubting the incredible impact that mobile technologies have had on both business and personal applications. Employees preferred to use smart phones not just for communication or entertainment purposes, but also to access the company's key applications. The integration of mobile applications and emerging cloud computing concept is Mobile Cloud Computing (MCC). It has been introduced to be a potential technology for mobile service. A prominent challenge by using mobile devices and the mobile cloud (Andreas.k et. al, 2010) is resource constraints of these handheld devices. Comparing to the desktop computers the key issues in the mobile devices are smaller screen size, less memory capacity, lower processing capacity and low battery backup. Due to these resource limitations most of the processing and data handlings are carried out in the cloud, which is known as SaaS cloud. The smart phones are used to access cloud resources by using the browser. Performance of this mobile cloud is impaired by the time varying characteristics such as, latency, jitter and bandwidth of the wireless channel. In this research we proposed a modified task scheduling mechanism called Ant Colony Optimization (ACO) to address the issues related to the performance of the mobile devices (L.Liu et. al, 2011) when used in a cloud environment and Hadoop. However there are bottlenecks related to the existing task scheduling techniques in MCC model which uses the built in FIFO algorithm for large amount of tasks. The proposed Ant Colony Optimization algorithm improve the task scheduling process by dynamically scheduling the tasks and improve the throughput and quality of service (QoS) of MCC.

[R. Achary, Vityanathan, Pethur Raj, S. Nagarajan. **Dynamic Job Scheduling Using Ant Colony Optimization for Mobile Cloud Computing.** *World Rural Observ* 2019;11(4):72-79]. ISSN: 1944-6543 (Print); ISSN: 1944-6551 (Online). http://www.sciencepub.net/rural. 12. doi:10.7537/marswro110419.12.

Keywords: Mobile Cloud Computing (MCC), Ant Colony Optimization (APO), Hadoop, Quality of Service (QoS), Software as a Service (SaaS)

1. Introduction

It's the earliest of days of smart phone cloud computing. But its time has arrived as demonstrated by a group of researchers who have sowed how smart phones can be used to create a self contained cloud computing network (Rajkumar et. al, 2009). When smart phones are used to create a cloud infrastructure, each smart phone has a function of the processing power as that of server elements in a remote cloud also; there are benefits of using mobile devices over a network. Mobile users accumulate rich experience of various services from mobile applications, which run on the device and/or on remote servers via wireless network. The wireless devices are facing many challenges due to their resources and the characteristics of the communication channel. The limited resources significantly impede the improvement of service qualities. Cloud computing (Rajkumar et. al, 2009) has been widely recognized as one of the next generation computing infrastructure (IaaS) and software (SaaS) provided by the cloud providers (Google, Amazon and Salesforce), based on the end users demand with the explosion of mobile applications and the support of cloud computing (Rajkumar et. al, 2009) for a verity of services for mobile users, mobile cloud computing is introduced as an integration of cloud computing (Rajkumar et. al, 2009) and mobile communication technology to explore new types of services and facilities for mobile users to take full advantages of cloud computing. The key challenges for MCC are;

- Higher network throughput required for the real time data transmission between the master and the slave.
- Lower delay characteristics of the network such as latency and jitter which cannot be tolerable by certain applications.
- Segregating the application functions across the cluster and the wireless device to optimally utilize the resources.
- In a cloud the Hadoop typically run on server. Data and file transmission from the client and the master requires a very high bandwidth of the order of gigabits per second, which is many times more than

that of the bandwidth of the WiFi links. This results the MapReducer (H.C. Yang et. al, 2007) job on a wireless mobile cluster (J. Dean and S. Ghemawat 2004) would be expected to perform much worse than a traditional cluster.

Under the mobile cloud computing environment in regard to multi-user with small amount of task scheduling from master to slave nodes Hadoop based architecture is used. This has certain advantages and it also uses FIFO algorithm for scheduling the tasks. Being next generation MCC platform needs with large amount of different granularity concurrent tasks to be processed in both master and slave nodes. The static task scheduling like FIFO is not suitable to these types of applications. Dynamic task scheduling is one of the solutions to mitigate these problems. The purpose of this research is to implement a dynamic job scheduling technique using Ant Colony Optimization as an effective algorithm to schedule the tasks for mobile cloud computing infrastructure on Hadoop (Andreas.k et. al, 2010, White T, 2009). This is an accomplishment of MapReduce (H.C. Yang et. al, 2007) by Google. Rest of this paper is organized as follows: In section 2 we explained the related work about mobile cloud computing (MCC), in section 3 an overview of the Mobile Cloud Computing (MCC), section 4 explore the MapReduce. We then detail our design and analysis Ant Colony Optimization algorithm in section 5. Section 6 gives the details of results and performance evaluation. The section 6 concludes.

2. Related work

In (Andreas.k et. al, 2010) and White T, 2009) the author explained the objectives of Apache Hadoop. It is an open source frame work in the form of MapReduce parallelization (H.C. Yang et. al, 2007). It was provided as a protocol, to list and evaluate a huge number of web contents. By default all the tasks scheduled by Hadoop, is using FIFO queue. This mechanism is very inefficient. The modified scheduler in Hadoop is Hadoop on Demand (HoD) is to address this concern by providing a private MapReduce (H.C. Yang et. al, 2007) scheduling system. This technique also failed because it desecrated the design features like data locality of the initial MapRedcue scheduler and added an additional overhead. In (L. Bianchi et. al, 2002) and (M. Dorigo et. al, 2006) the author analyzed the implementation of ACO for parallel problem solving and computational intelligence. ACO is one of the efficient techniques with strong job distribution capability, scalability and parallelism. In the existing Hadoop cloud computing system, if the number of nodes entering into the cloud increases, it degrades its performance and also there is a probability of failure. We combined the advantages of ACO into MCC to achieve an efficient and scalable job scheduling mechanism.

3. Mobile cloud computing

MCC refers to an infrastructure whereas both the data storage and the data processing happen outside the mobile device. Mobile device applications (A. Garcia and H. Kalva 2011) move the computing power and data storage away from mobile phones and into the cloud. i.e, in MCC the data processing and storage are moved from the mobile device to powerful centralized computing platforms located in cloud. These centralized applications are then accessed over the wireless connection based on a thin native client or web browser on the mobile devices. The key benefits of MCC we have included on the following section;

3.1 Extended battery life

Battery is one of the main resources in any mobile device. The CPU manufacturer and software developer have introduced several solutions to enhance the performance of the mobile devices and to reduce the power consumption. However these solutions require changes in the structure of mobile devices, or it requires an upgrading in the hardware. Computation off loading technique is one of the techniques introduced with the objective of migrating the large computation and complex processing from resource limited devices like mobile devices to resourceful machines (i.e., server in cloud). This avoids taking a long application execution time on mobile devices which results in large amount of power consumption (E. Vartiainen and K.V. Mattila, 2010). The cloud computing can save the energy significantly.

3.2 Improved data storage capacity and processing power

Mobile devices (L. Liu et. al, 2011) have limited storage capacity. MCC is developed to enable mobile users to store/ access the large data on the cloud through wireless networks. Users can upload the data to the cloud and can access them from any devices with the cloud user can save considerable amount of storage space on their mobile devices.

3.3 Improving reliability

Storing user's data or running applications on clouds is an effective way to improve the reliability of the systems, because the data and applications are stored and backed up on a number of computers. This reduces the chances of data and application lost on the mobile devices.

3.4 The other advantages are

3.4.1 Dynamic provisioning

Dynamic on demand provisioning of resources, it is a flexible way for service providers and mobile users to run their applications with advanced reservation of resources.

3.4.2 Scalability

The deployment of mobile applications can be performed and scaled to meet the unpredictable user demands due to flexible resource provisioning. Any up gradation or modification of the applications by the service providers can be done easily without or with little constraint on the resource usage.

3.4.3 Multi-tenancy

Service providers (network operators and data center owners) can share the resources and costs to support a variety of applications and large number of users.

3.4.4 Ease of integration

Cloud helps to integrate the different services from service providers without much difficulty through the Internet to meet the end users requirements.

4. Mapereduce

MapReduce (H.C. Yang et. al, 2007) address the programming challenges, large data processing and scheduling in a cloud environment. It was initially proposed by Google for large scale data processing in a distributed computing environment. MapReduce model provides a simple programming abstraction that hide low level details from the programmer at runtime, tuned to the basic architecture. In this the user simple specify a Map function that processes data to generate a set of intermediate key/value pairs, and a Reduce function that processes all intermediate values associated with the same intermediate key.

The system consists of a master server and several wireless client terminals. The server monitors and keeps track of the user data and applications, task scheduling and assigning them to the respective clients. The clients are responsible for requesting work from the server, retrieving the proper modules and data, then performing the execution on them and returning the results to the server. In this client-server environment, the data gets passed from the device over wireless network to the cluster where it gets processed and the result is sent back to the device. The common issues in the wireless channels are; the network is quite slow and causing latency. The requirements for this type of wireless networks to how MapReduce (H.C. Yang et. al, 2007) would work with the data being kept close to the device. In some situations, then it is also found that the data could be processed faster using the smart phone cluster. But, if a node goes down, the quality of the network decreases considerably.

One of the main challenges of implementing MapReduce on cell phones is that in a regular cloud with servers, the failure rate is very low, and the latency of a signal transmitted between the nodes is relatively low. This may not in the case of cell phones. It means that among other things as the failure rate of any one node in the system decreases the performance of the entire network exponentially. The MapReduce (H.C. Yang et. al, 2007) is a simple programming framework. In which we have to write the map and reduce functions. The system will automatically handle the distribution of codes, distribution of data, executing the tasks in parallel, work scheduling and all other complicated and distribution systems issues. The MapReduce approach has been proved to be an effective programming approach for developing machine learning, data mining, and search application in cloud data centers. Its advantage is that it allows programmers to abstract from the issues of scheduling, penalization, partitioning replication and focus on developing their applications.

In the MapReduce (H.C. Yang et. al, 2007) mode shown below map and reduce are the data processing functions. The parallel map tasks are run on input data which is partitioned in to fixed size blocks and produce intermediate output as a collection of <*key, value*> pairs. These pairs are shuffled across different reduce tasks based on <key, value> pairs. Each reduce tasks accept only one key at a time and process data for that key and output the result as <key, value> pairs. Hadoop MapReduce architecture consists of one job tracker (Master) and many Task Trackers (workers) (B. Rochwerger et. al, 2009, W. Tsai and J. Balasooriya, 2010). The job tracker receive the job as an input submitted from user, divides it into map and reduce tasks, assigns these tasks to the task trackers, monitors the progress of the task tracers, and finally when all the tasks are completed, the user will get the job completion report.

Each Task Tracker has a fixed number of maps and reduces tasks slots that determine how many map and reduce tasks it can run at a time.

In cloud computing large amount of different granularity concurrent tasks are to be processed. Using only static job scheduling technique like FIFO used in Hadoop is not suitable to this application. Dynamic job scheduling algorithm has great stochastic performance in this scope. The presence of master node mechanisms overall task scheduling Hadoop (White T, 2009) can split the input file into many blocks and dispatch them to different slave nodes and implement data locality to avoid large scale of data shuttle and save lots of processing time and input/output time. Hadoop includes a built in FIFO algorithm which sequentially execute the tasks according to the priority parameters and the arrival time. This results into a larger waiting time for small granularity tasks, when large granularity tasks are under execution. This disadvantage would turn out many resource fragments, under utilization and poor flexibility. The first solution to this problem was

Hadoop on Demand (HoD) White T, 2009) which provides private MapReduce clusters over a large physical cluster. In an infrastructure based wireless cloud (MCC), the master node has higher performance than slave nodes. The objective of job scheduling is to dispatch parallel jobs to slave nodes according to scheduling policy and priority constraints to reduce total execution time and minimizing the computation cost, which intern improve the performance and the Quality of service (QoS).

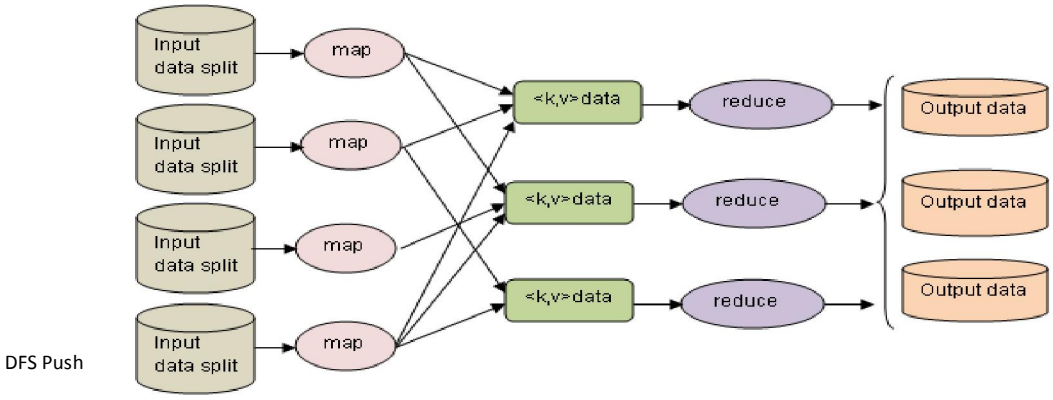

Fig.1. Hadoop MapReduce

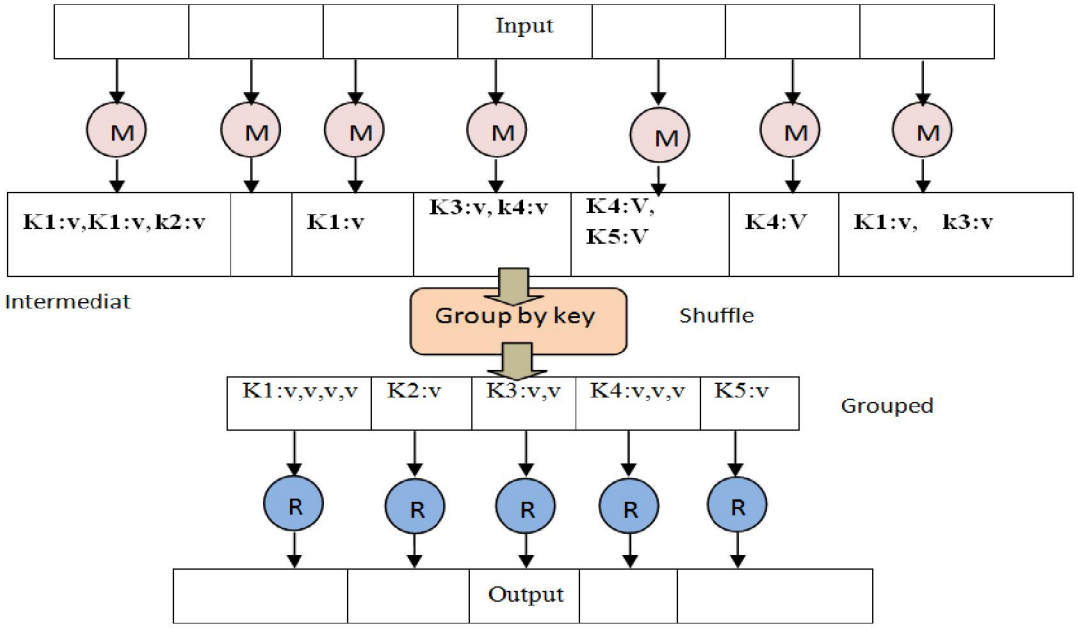

Fig.2. MapRedcue work flow

5. Ant colony optimization

Ants are insects which live together; they find the shortest path from nest to food with the aid of pheromone. It is a chemical material deposited by the ants, which serve as a critical communication media among ants, their by guiding the determination of the next movement. On the other hand ants find the shortest path based on the intensity of pheromone deposited on different paths. The intensity of deposited pheromone is one of the most important factors for ants. In the algorithm routing information is organized in pheromone table $P^k{}_{ij}$ it is a two dimensional matrix. An entry $P^k{}_{ij}$ of this pheromone table contains information about the route from node *i* to destination d, over neighbor *j*. This information includes the pheromone value $P^k{}_{ij}$ which is a value indicating the relative goodness of going over node *j* when travelling from node *i* to the destination node *d*, as well as statistical details about the path and possible virtual pheromone. Apart from a pheromone table

each node also maintains a neighbored in which it keeps track of which nodes it has wired or wireless links to.

To begin the communication process, the source node of the session, controls its pheromone table, to find the routing information available for the requested destination. If it does not it starts a reactive route setup process, in which it sends an ant packet out over the network to find the route to the destination. Such an ant packet is called a reactive forward ant. Each intermediate node receiving a copy of the reactive forward ant forwards it. This is done via uni-casting in case the node has routing information about the ant's destination in its pheromone table, and via broadcasting otherwise. Reactive forward ants store the full array of nodes that they have visited on their way to the destination. The first copy of the reactive forward ant to reach the destination is converted into a reactive backward ant; this retraces the exact path that was followed by the forward ant back to the source. On its way it collects the information about each of the links of the path. At each intermediate node and at the source, and then it updates the routing tables based on this information. This is how the first route between the source and destination is established at computation of the reactive route setup processes.

5.1 ACO algorithm and Analysis

The task scheduling issue is addressed as follows. If there are *n* tasks, it needs to be dispatched from *n* wireless Nodes (Smart phones) to the cloud, where one smart phone takes one task at a time. So different dispatch plans have different execution cost and resource consumption. The proposed algorithm is to find the plan to allot the tasks smoothly to ensure availability and optimality of the resources.

Under the same processor performance condition task complexity is the key factor of influence processing time. Complex task require higher processing time than simple jobs. The different parameters considered for optimization and scheduling is discussed below.

The processing cost for a i^{th} node to complete j^{th} task is;

Cost matrix: $C_{n*n} = \{C_{i*j} \mid C_{i*j} \in C_{n*n}$

$$C_{i*j} \geq 0;$$

$$i = 1.2.3.....n$$

$$j = 1,2,3.....n$$

These values in the matrix are derived from the requested task complexity and processor performance.

$$T_{n*n} = \{T_{ij} \mid T_{ij} \in T_{n*n}$$

These values in the matrix represents the pheromone of that the j^{th} task was dispatched to i^{th} node.

The matrix is initialized to constant matrix or 0 before scheduling.

Performance matrix:

$$V_{n*n} = \{V_{ij} \mid V_{ij} \in V_{n*n}$$

These values in the matrix represents that the j^{th} task was dispatched to i^{th} node. The matrix is initialized as $\frac{1}{C_{j*j}}$ that is to say this matrix is reverse ratio to cost i^{th} Task scheduling matrix:

$$R^{k}{}_{n*n} = \{R^{k}{}_{ij} \mid R^{k}ij \in R^{k}{}_{n*m}$$

$$i = 1,2,3.....n$$

$$j = 1,2,3.....n$$

$$k = 1,2,3.....n$$

This represents the task scheduling plan that the k^{th} ant implements task scheduling. The matrix is initialized as 0, and the value of elements in this matrix is 1 or 0. When, $R_{ij} = 1,\ j^{th}$ task is dispatched to i^{th} node. $R_{ij} = 0,\ j^{th}$ task is dispatched to i^{th} node.

The task dispatching matrix R_{n*n} represents how to dispatch the task to the slave node to complete all the submitted tasks with the minimum cost. If there are N ants to complete all the submitted tasks, ant's in one trip stands for one task dispatching procedure in which they need N walks which demonstrates dispatching of one task. These walks are tagged as's'. When, all the ants complete one trip, and then it can be thought as one loop in completed.

N_c - is the times of loops.

$$Task = \{task1, task2, ...taskn\}$$

$$node = \{node1, node2, ...noden\}$$

Let us introduce an 'n' dimension vector $D^{N_n}{}_n$ which has elements $D^{N_n}{}_k$ that is $D^{N_n}{}_k \in D^{N_c}{}_n$ to stand for the k^{th} ant's cost vector during the $N_c{}^{th}$ algorithm loop. The initial value of $D^{N_n}{}_n$ is 0. The key probability matrix $P^{k}{}_{n*m}$ which has elements $P^{k}{}_{ij}$ that is $P^{k}{}_{ij} \in P^{k}{}_{n*m}$ to stand for the

probability of dispatching j^{th} jobs to i^{th} slave node, and $P^k{}_{ij}$ has relations to its pheromone matrix and its tasks performance matrix with their relationship is;

$$P^k{}_{ij} = \frac{T^\gamma{}_{ij} * V^\gamma{}_{ij}}{\sum_{j=1}^{n} T^\gamma{}_{ij} * V^\gamma{}_{ij} j}$$

The task arrival and completion moment are tagged as $at(i)$ and $et(i)$ respectively with each job has it's priority, which is tagged with a $\Pr iority(i,t)$ meaning the priority of task i at the moment t . The algorithm is described as below;

If M tasks are arrived during a given period and there are N tasks which are submitted and k tasks are returned due to time out. There are some indexes of QoS as follows;

Average task Execution Time (AET);

$$AET = \{\sum_{i=1}^{n} (et(i) - at(t))\} / N$$

Task weighting Average execution time (WAET);

$$WAET = \{\sum_{i=1}^{n} (et(i) - at(t)) * \Pr iority(i, et(i)))\} / \sum_{i=1}^{n} \Pr iority(i, et(i))$$

The task's losing rate $ls = (k / M)$

The slave node work load ratio which is sampled at every period.

6. Result and performance evaluation

To verify the efficiency of our ACO algorithm we simulated the performance and other QoS parameter against Hadoop's built in FIFO on MCC environment White T, 2009). The master node is a high performing computing device with Linux operating system as the platform and the slave nodes are the smart phones with Android operating systems. The performance is evaluated for Hadoop FIFO scheduling and by the proposed dynamic scheduling technique using ACO algorithm for scheduling the tasks.

The static and dynamic scheduling performance comparison is done by simulating the system using NS2. The simulation system includes 10 wireless nodes and submitted an equal amount of jobs for both Hadoop FIFO algorithm and ACO based dynamic scheduling mechanism. Table (1) represents in ACO based dynamic scheduling the total execution time and loop time, which approximately half of that for the static is scheduling. The simulated result is validated by comparing the Hadoop FIFO and ACO dynamic scheduling White T, 2009). The FIFO algorithm results a best performance for small granularity jobs to achieve local optimization. For large amount of jobs and to attain global optimization, dynamic scheduling using ACO algorithm results better performance.

A lg *orithm* : *Dynamic job scheduling*
Initialize $N_c = 1, Task$, *Node*
Do
for $(K = 1; k \le n; k++)$ *Begine*
for $(s = 1;\ s \le n;\ s++)$ *Begine*
If
the task is tagged $\Pr iority\ (i,t)$
Ramdomely select the nodei to compute the j^{th} *task at a* $\Pr obability\ P^k{}_{ij}$
Tag it as $P^k{}_{ij\max}$
Set $R^k{}_{ij} = 1$
Delete the nodei from nodeset
Delete the taskj fromTaskse t
set $\cos t$ *vector* $D^{N_c}{}_k = D^{N_c}{}_k + C_{ij}$
else
wait for task j $\Pr iority \ge \Pr iority\ (i,t)$
End
End
for $(k = 1;\ k \le n;\ k++)$ *Begine*
Pheromone update : – *Increment in pheromone* ΔT *and ants pheromone is gained to* $\cos t$ *inversly*
$\Delta T = \sum_{k=1}^{n} \frac{Q}{DNCK}$ *and according to the* k^{th} *ant* $R^k{}_{n*n}$ *matrix*
Is $R^k{}_{ij} = 1$ *and set* $T_{ij} = T_{ij} + \Delta T$
End
set volatile parameter $0 < p < 1$ *to* $\lim$ *it the* $\inf$ *inite increment of*
$T = T * (1 - p)$
Find the $\min$ *imum element*
$D^{N_c}{}_{\min}$ *among* $\cos t$ *vector* $D^{N_c}{}_n$
If $D^{N_c}{}_{\min} < D^{N_c}{}_{\min}$
$Nc = Nc + 1$
until $Nc \ge Nc_{\max}$

Fig. 3 ACO Algorithm

In Fig. (4), graph represents the comparison between the processing time and the number of nodes, by varying values of percentage parallelism (P) between the wireless clients and the servers. The graphs are plotted by considering the differences in the slope of the processing time and the task time versus the number of nodes connected. Graphs in fig. (5) & (6) represents the variations of data transfer time with respect to the size of the packets, of small and large jobs. The transfers of the small granular jobs are more costly in terms of number of bytes per second, particularly for the wireless links, because of the additional information provided for small packets becomes an overhead to establish the connection. Fig. (7) Shows the average execution time (AET) for the two algorithms is approximately equal with less number of jobs. As the number of jobs inflow to the system increases the graph representing ACO indicates the enhancement in execution time. In Fig. (8) the graph represents the single job WAET is decreasing along the increasing of the number of jobs, also an improvement in the performance of ACO over FIFO systems. The job losing rate for both ACO and FIFO algorithms are indicated in fig. (9), the dropping curve of ACO represents that there is decrease in the job loss by using ACO algorithm as the number of jobs are increased. The job queue includes jobs with smaller and larger granularity. FIFO algorithm results higher performance for small granularity jobs. The simulation result indicates that ACO algorithm is more adaptable for small and large granularity jobs submitted.

Table 1. Experimental result of job scheduling

	FIFO Scheduling		ACO based dynamic scheduling	
Trials	Sum of loops	Sum of execution time	Loop time	Execution time
1	19	31ms	8	16ms
2	17	29ms	9	14ms
3	15	26ms	7	13ms

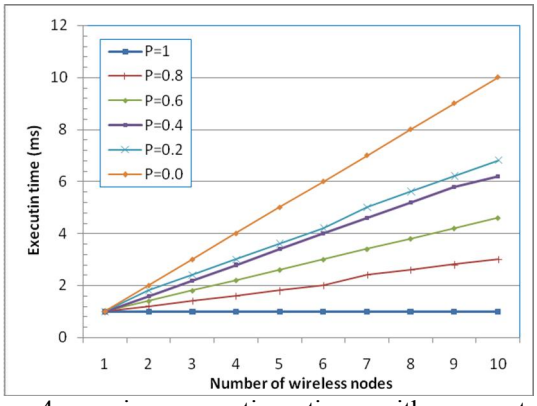

Fig. 4 varying execution time with percentage parallelism.

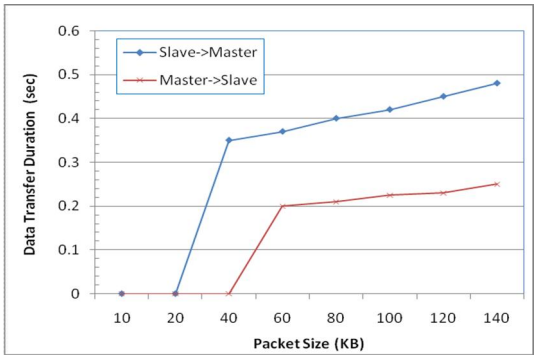

Fig. 5. Network transfer time for small granularity jobs

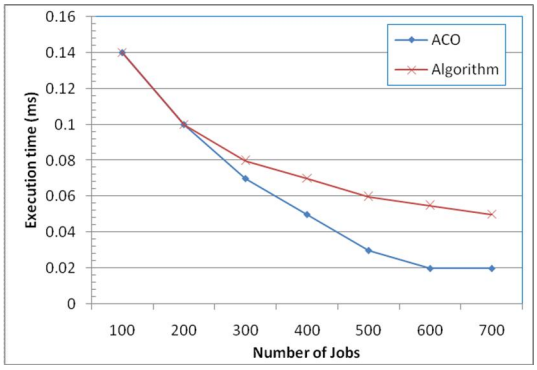

Fig. 6. Network transfer time for large granularity jobs

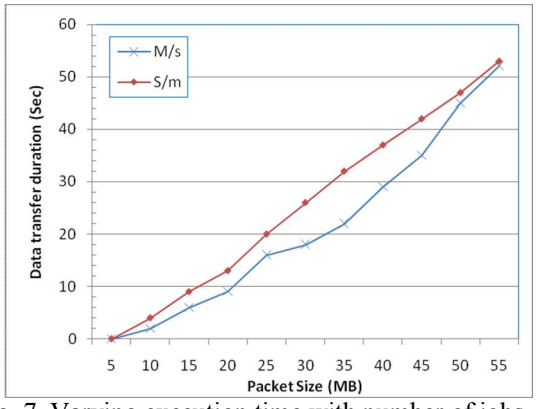

Fig. 7. Varying execution time with number of jobs

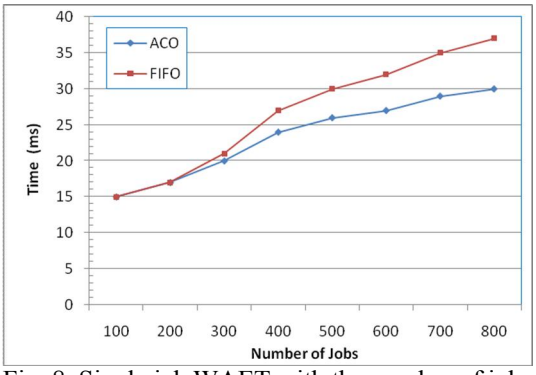

Fig. 8. Single job WAET with the number of jobs

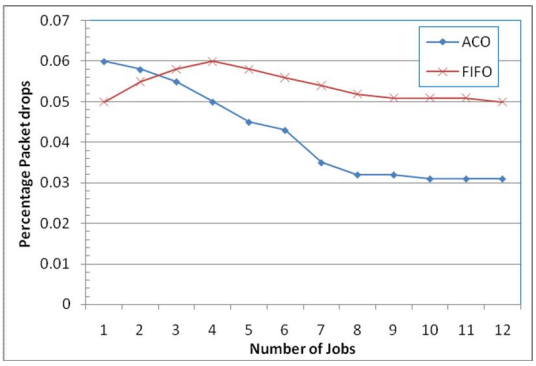

Fig. 9. Percentage job loss as the number of jobs increases

7. Conclusion

In this paper we presented a new paradigm of offering and utilizing the characteristics of cloud computing model and identifying the limitations of Hadoop build in FIFO technique of dealing with the task scheduling for Mobile Cloud Computing model. The proposed Ant Colony Optimization algorithm bridges the gap under Hadoop platform for scheduling the tasks dynamically and enhancing the performance of MCC model for scheduling large number of tasks.

8. Acknowledgement

I would like to thank the Vice-Chancellor of SASTRA UNIVERSITY for the opportunity and the support provided for this research.

References

1. Andreas Klein, Christian Mannweiler, Joerg Schneider, and Hans D. Schotten. "*Access schemes for mobile cloud computing*". Eleventh International Conference on (MDM-, 2010, pages 387–392.
2. L. Bianchi, L. M. Gambardella, M. Dorigo "An ant colony optimization approaches to the probabilistic traveling salesman problem". In *Proceedings of PPSN-VII, Seventh International Conference on Parallel Problem Solving from Nature*, Lecture Notes in Computer Science. Springer Verlag, Berlin, Germany – 2002, pp 883-892.
3. J. Dean and S. Ghemawat. MapReduce: *"Simplified data processing on large clusters"*. OSDI-2004.
4. A. Garcia and H. Kalva, *"Cloud transcoding for mobile video content delivery,"* in Proceedings of the IEEE International Conference on Consumer Electronics (ICCE), pp. 379, March 2011.
5. L. Liu, R. Moulic, and D. Shea, *"Cloud Service Portal for Mobile Device Management,"* in Proceedings of IEEE 7th International Conference on e-Business Engineering (ICEBE), pp. 474, January 2011.
6. Marco Dorigo, Mauro Birattari, and Thomas St¨utzle *"Ant Colony Optimization Artificial Ants as a Computational Intelligence Technique"* Universit ´ e Libre de Bruxelles, BELGIUM.
7. Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. *"Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing* as the 5th utility. Future Generation Computing Systems -25(6):599–616, 2009.
8. B. Rochwerger, D. Breitgand, E. Levy, A. Galis, K. Nagin, L. Llorente, R. Montero, Y. Wolfsthal, E. Elmroth, J. Caceres, M. Ben-Yehuda, W. Emmerich, and F. Galan. *"The RESERVOIR Model and Architecture for Open Federated Cloud Computing"*. IBM Journal of Research and Development, 53(4): 2009.
9. W. Tsai, X. Sun, and J. Balasooriya, "*Service-Oriented Cloud Computing Architecture,"* in Proceedings of the 7th International Conference on Information Technology: New Generations (ITNG), pp. 684-689, July 2010.
10. E. Vartiainen, and K. V. -V. Mattila, *"User experience of mobile photo sharing in the cloud,"* In the Proceedings of the 9th International Conference on MUM-2010, December 2010.
11. H.-C. Yang, A. Dasdan, R.-L. Hsiao, and D. S. Parker. MapReduce-merge: simplified relational data processing on large clusters. SIGMOD-2007, pages 1029–1040.
12. White, T.: Hadoop: The Definitive Guide. O'Reilly, Sebastopol (2009)