AN ADVANCE ALGORITHM FOR TASK MANAGEMENT ON ACTIVITY BASED COSTING IN CLOUD COMPUTING

ASHUTOSH S. INGOLE

Abstract: Cloud computing is a structure of database servers, application servers and delivery network built in order to maximize the efficiency of system. This algorithm works on activity based costing. A complete job from customer called as task is divided into number of small activities. We check for dependencies within the activities of same task. We consider the tasks are independent on each other. After checking for dependencies we shuffle the activities in a proper execution sequence. We calculate cost for each activity and then cost for a complete task. Cost of an activity depends on two parameters, estimated execution time of an activity and its resource cost. Estimated execution time for an activity can be calculated from the estimated execution time of that task. We decide the final priority for a task on two parameters, total cost and arrival time of the task. We also divide priorities in three separate queues, high priority queue, low priority queue and mid priority queue. We will be also maintaining the two main tables, universal resource table and activity table. This survey is done in activity table where record of all existing tasks is maintained. The cost of resources changes continuously as their demand changes. This algorithm is very effective in cloud computing where systems are busy all the time and handles huge database and traffic is heavy. For systems with very less traffic and sufficient number of servers this algorithm may not show any significant difference than the traditional scheduling algorithms.

Keywords: activity based costing, activity table, advance algorithm, cloud computing, resource cost, task management, universal resource table.

Introduction: Cloud Computing is the technology specially adopted for an efficient data providing services under some critical circumstances such as, very less resources with great number of clients, unstable demand of resources etc.

Now what do we mean by 'unstable demand'? Suppose some 'X' organization is handling huge database for its client. Now depending on customer requirement we provide data to them. Now at certain point of time, say, they need resources R1 mostly. Then after a particular period of time they do not require those R1 types of resources. Then why do we waste our space in storing and maintaining those resources for rest of the time. Can we use that space and those application servers for some other purpose? Yes we can! That's where concept of cloud computing arises. Cloud computing helps to implement and execute queries more dynamically. It dynamically keeps busy all available resources and servers. But what is more important than that is, is our cloud efficient and economical to provide services to customers with limited number of resources and in such adaptive conditions? Definitely that will depend on how well we handle incoming requests for data. How well we allocate resources to queries.

The goal can be achieved by proper sorting and scheduling of tasks. This can be done by using the concept of activity based costing (ABC). In activity based costing, we divide task in number of activities and calculate cost of each activity and then cost of a complete task. The cost of activity is based on two parameters, total cost of resources and estimated execution time required by an activity. While evaluating final priority for a task, we consider two parameters, arrival time and original priority of the task. The original priority of the task depends on the type of client. We divide clients on four different categories according to their priority or we can say according to their subscription.

Related Work: Cloud computing is defined in three models, Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) [1]. The general architecture of cloud computing includes master server, scheduler server, database server and virtual machines [7].Load balancing has always been a research subject whose objective is to ensure that every computing resource is distributed efficiently and fairly and in the end improves resource utility. In traditional modes of task scheduling in operating system, we never considered activity based costing. To reduce the system load we divided application login among the number of application servers and virtual machines (VMs) [6]. The major advantage with virtual machines is we can dynamically increase or decrease number of resources as demand changes according to the period and clients.

Request from client arrives at master server first in the form of query or queries. This request is called as job or task which is divided into number of subtasks called as activities. Activities of the same task may or may not be dependent on each other. If activities are dependent on each other, then using the concept of directed acyclic graph (DAG) [10], we need to solve

the dependency issue and sort the activities in a proper execution sequence. Many scheduling algorithms doesn't consider the concept of dependency which is the major concern in task scheduling in cloud computing.

In cloud computing, there are three different modes of renting the computing capacities from a cloud provider [10].

- Advance Reservation (AR): Resources are reserved in advance. They should be available at a specific time;
- Best-effort: Resources are provisioned as soon as possible. Requests are placed in a queue.
- Immediate: When a client submits a request, either the resources are provisioned immediately, or the request is rejected, based on the resource availabilities.

Initial priority of the task depends on the priority of the client. Clients can be divided in different categories according to their subscription, which means client paying more money to access the cloud or to use its services has higher priority initially.

Architecture of Cloud Computing System: Architecture we have proposed for our cloud computing system includes master server, scheduler, application servers with virtual machines and record container. Each of the components has its own importance and functions. Along with the functioning of components, we will see flow of our algorithm. We will see the detail functioning of each of the above.

A. Master Server

Tasks from all clients arrive first to the master server. As soon as master server collects the first task, it will start dividing task into activities. Activities are nothing but the smallest queries required to solve to get the result for a complete task. After separating different activities, master server checks for dependencies among the

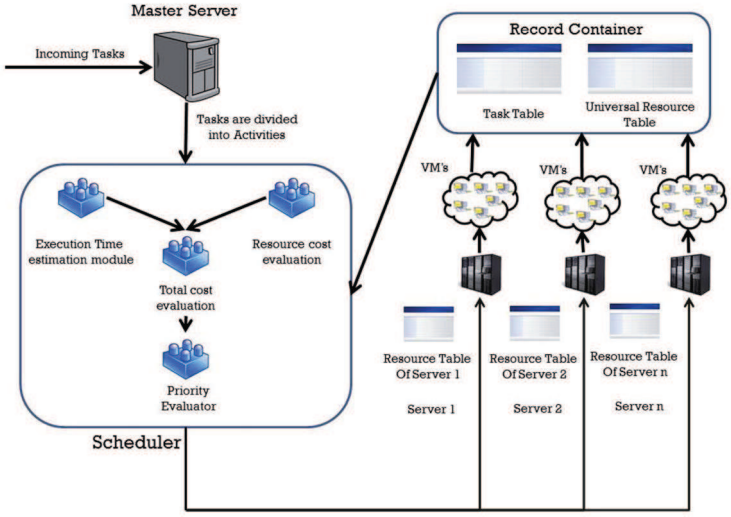

Fig.1 Architecture of cloud computing

activities of the same task using Directed Acyclic Graph (DAG). To understand DAG more clearly, consider an example given in Figure.2.

a_{14}	a_{13}	a_{12}	a_{11}

Fig.2 Activities of task T_1

These are four activities of task T_1 in initial sequence of arrival. Consider sequence right to left. According to this sequence we have to execute activity a_{11}first,

a_{13}	a_{14}	a_{12}	a_{11}

then a_{12}, a_{13} and at last a_{14}. Consider that activity a_{13} is dependent on the output of a_{14}, and in such case we have to rearrange the sequence of execution so that a_{13} should get execute after a_{14}. This can be done using DAG. Let's see how DAG works.

A DAG $T= (V, E)$ consists of a set of vertices V, each of which represents an activity in the task, and a set of edges E, showing the dependencies among the activities. The edge set E contains edges eij for each activity $vi \in V$ that activity $vj \in V$ depends on. Given an edge eij, vi is the immediate predecessor of vj, and vj is called the immediate successor of vi. An activity only starts after all its immediate predecessors finish. Activities with no immediate predecessor are entry-activities, and activities without immediate successors are exit-activities [10].After rearranging the activities we get the final proper sequence of execution as shown in Figure.3.

Fig.3 Final execution sequence of task T_1

DAG is handy in detecting deadlocks as they illustrate the dependencies amongst a set of processes

and resources (both are nodes in the DAG). Deadlock would happen when a cycle is detected.

- If the cycle exists, all the activities related to that particular task should be moved from the queue and the activities from the next task are processed.
- If no cycle exists, then the order of the activities from the task is decided and activities are placed in the task queue for further calculations.

1. Get the activity dependencies form master server.
2. Calculate the DAG from the given information.
3. According to the DAG fill up the task queue.
4. An activity only starts after all its immediate predecessors finish.
5. Activities with no immediate predecessor are entry-activities, and activities without immediate successors are exit-activities.

For example, suppose we have a task which consists of 4 activities A, B, C, D in which activity B is dependent on A and C. D is dependent on B. So from the given data we generate DAG and check for the cycles.

This can be represented as a DAG. Once you have the DAG in memory, DAG is drawn and the order is decided i.e. B is dependent on A and C and D is dependent on B. Thus A or D should be processed before B and B should be processed before D. We will use DAGs to ensure proper build order of the activities in respective task. So the final execution order for the above example is A-C-B-D.

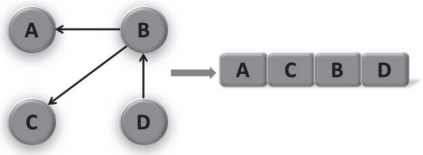

Fig.4 DAG

B. Scheduler Server

Scheduler server contains three modules, resource cost evaluation, estimated execution time evaluation, total cost evaluation and prioritization. Activities coming from master server first arrive at resource cost evaluation module. This module checks for the resources required by each activity, and calculate the cost of resource at that instance. Then these activities are forwarded to estimated time evaluation modules which will evaluate the approximate time required by each activity to get execute. Once we found the estimated time, we can calculate the cost of execution of each activity and hence for a complete task. Using the total cost of a complete task and initial priority of the task we can assign a priority to the task with the help of prioritization module. After assigning the priority we can place the task at proper place according to the priority in any one of the priority queues. The detail operations to be performed to get the total cost of a task and hence its priority are described in this paper later.

C. Application Servers

Application server is the most important component of the architecture as it is responsible for execution of activities. Each application server handles number of virtual machines and its own resource table. Each application server has access to limited number of resources and it has to maintain the record of these resources. It may happen that a particular resource is accessed by multiple application servers. In such case if any application server updates the record of such particular resource then record of that resource must get updated at all application servers using that resource. And for the same reason application servers must be connected to each other to communicate properly. According to the demand of resources we can add or remove resources, increase or decrease the number of virtual machine on application servers.

D. Record Container

Record container maintains the records of resources and activities in the form of tables.

It contains two tables, universal resource table and activity table. Universal resource table contains the record of total resources available in the cloud. Activity table contains all tasks in the priority queue or in the execution queue of application server along with their required resources and required application servers to get execute. This record container is very well connected to all other components of the system such as scheduler server, application servers in order to update its table every time when new task arrives or execution completes. The detail description of the tables and their contents is given in this paper later.

Activity Based Costing: Activity based costing is the concept which never been used in traditional task scheduling algorithms such as First in First out (FIFO), Shortest Job First (SJF), Round Robin (RR) etc. In activity based costing we calculate cost for each activity and hence cost of a complete task, which helps to prioritization and schedule the tasks efficiently. Below we will see step by step evaluation of cost of tasks and prioritization.

A. Resource Cost

Cost of the resource changes at every new demand of resource by new task. Using activity based costing we can conclude that which resource is frequently used and which one is rarely. To evaluate cost of each resource, resource cost evaluation module must be very well connected to the universal resource table. After evaluating the cost the table must be updated. We consider the following scenario to understand

how to calculate resource cost and how the cost changes with every new demand.

Scenario: consider our cloud contain five different resources R_1, R_2, R_3, R_4 and R_5.Consider our first request is task T_1 which contains four activities a_1, a_2, a_3 and a_4. Below in Table.1 it is shown the resources required by each activity. Table.2 is showing total number of resources available in cloud.

Table.I Resources required

	R_1	R_2	R_3	R_4
a_1	4	2	3	-
a_2	2	-	4	5
a_3	6	3	1	-
a_4	-	2	3	4

Table.II Resource table

	R_1	R_2	R_3	R_4
Total	20	20	20	20
Assigned	-	-	-	-
Available	-	-	-	-

Activity a_1 comes first in the execution sequence and it requires resources as shown in Table.1. We use the equation (1) to calculate the resource cost. Consider any random i^{th} resource R_i, then

$$\text{Cost of Ri} = \frac{\text{Total demanded Ri instances}}{\text{Total Ri instances initially}} \quad \ldots\ldots (1)$$

When activity a_1 arrives, it needs 4 instances of R_1, 2 instances of R_2 and 3 instances of R_3. We calculate cost for each resource using equation (1).

Cost of R_1= 4/20 = 0.20

Cost of R_2= 2/20 = 0.10

Cost of R_3= 3/20 = 0.15

We have calculated cost for every resources needed by activity a_1, and to get total resource cost for activity a_1 we just do addition of cost of every resource. Hence the total cost for activity a_1 is 0.45.After calculating the cost for each resource we update the Table.2 and it is shown in Table.3.

Table.III Updated resource table

	R_1	R_2	R_3	R_4
Total	20	20	20	20
Assigned	4	2	3	-
Available	16	18	17	-

Activity a_2 comes second; again we calculate cost for resources required by activity a_2.

Cost of R_1= 6/20 = 0.30

Cost of R_3= 7/20 = 0.35

Cost of R_4= 5/20 = 0.25

After calculating the cost for each resource we update the resource table again and it is shown in Table.4. While calculating cost for activity a_2, we added number of resource instances needed by a_2 in number of resource instance assigned to a_1. Hence we can see the difference in the cost after arrival of activity a_2.

Table.IV Updated resource table

	R_1	R_2	R_3	R_4
Total	20	20	20	20
Assigned	6	2	7	5
Available	14	18	13	15

We have seen how to calculate the resource cost and how the demand of resources affects its cost. The demand may exceed the total available instances of resource, and for such case we have maintained activity table. We will see what the significance of activity table in this paper is later.

B. Estimated Execution Time

Understanding the basic difference between activities and tasks is very important. A single task may contain a single activity or many activities. So simply, to complete a particular task we need to execute its all activities.Our algorithm will schedule the tasks on the basis of cost. And that cost will be of an entire task, not a single activity. But while evaluating the cost for an entire task we need to evaluate cost of every activity we perform to complete the task. Hence to evaluate that cost we need to find out estimated time also. Again it is a simple mathematics that the total time for a particular task will be the summation of time taken by its every activity. Now the question is how to find out the estimated time required for an entire task. Our system knows which kind of client it handles and what kind of services we provide. We keep track of tasks for a certain period so that maximum number of verity of tasks arrives and note down the time taken by each task. Then we find out two tasks, one with minimum execution time and other with maximum execution time. We calculate the time difference between these two tasks and divide it into 10 windows. This means now we have ten timing windows. Again we go back to our record, check every task's execution time and sort them in these timing windows. Now we have ten different categories of tasks and this record will be held as standard to compare and get execution time for incoming tasks in the future. This is the one solution to get approximate time of execution for a task. This method is slow but very effective according to the time.As already mentioned in the above paragraph

that total time of execution for an entire task is the total time taken by its all activities to get execute, it's very simple now to get estimated execution time for an activity. Consider that there is a certain task T_i, having total number of activities five, a_1, a_2, a_3, a_4 and a_5. Now we have estimated execution time for an entire task T_i, and divide it by 5, so that we will get estimated execution time for an activity. It is shown in the equation (2).$Tij = \frac{Tj}{Kj}$ ..... (2)

Where,

T_{ij}: estimated execution time for i^{th} activity of j^{th} task

Tj: estimated execution time for complete j^{th} task

K_j: total number of activities in j^{th} task

So from the above discussion we can conclude that estimated execution time for all activities of the same task is same.

C. Activity and Task Cost

We have seen how to calculate resource cost and estimated execution time for an activity. It is very important while calculating complete activity cost to consider estimated execution time separately for every activity though the time is same for activities of same task. Because though the estimated time of execution is same for all activities of same task, the resource cost is different as the resources needed by different activities are different.

Cost of an activity is mainly based on two parameters, total resource cost and estimated execution time. It can be calculated using equation (3).

$Cij = (\text{resource cost}) * (\text{estimated execution time})$ ..... (3)

Where, C_{ij}: total cost of i^{th} activity of j^{th} task

We have seen how to calculate total cost of an activity. Now the total cost of a task is just the summation of total costs of its all activities. We will use equation (4) to calculate the total cost of an entire task. $Cj = \sum_{i=0}^{n} Cij$ ..... (4)

Where, C_j: total cost of j^{th} task

C_{ij}: total cost of i^{th} activity of j^{th} task

We have to understand that whatever calculations we have done to calculate cost of each activity were just important to get total cost of an entire task. But the scheduling we are going to do in this paper is based on this total cost of a task.

D. Determining Final Priority

Evaluating final priority of a task is the last part of our calculations. Once we get the final priority, we just have to place it in priority queue. We have studied traditional algorithms of task scheduling in significance of waiting queue below here, by considering a scenario.

Scenario: let's say we have 50 activities in low priority queue and there are two dedicated application servers say serevr1 and server2. As the activities are coming to the low priority queue and we are sending them to their respective application server's execution queue.

a_{42}	a_{22}	a_{32}	a_{12}	a_{21}	a_{31}	a_{11}

Figure.5 Low priority queue

Now consider that activities a_{11} and $a_{31}$are placed in execution queue of application server1 and a_{21} needed

operating system such as priority scheduling. In priority scheduling we have number of tasks along with their original priority and we use the same priority to schedule the tasks. In traditional priority scheduling we schedule number of tasks present at a particular instance of time and task with highest priority among them will get execute first. If a new task arrives in between the execution of this highest priority task then newly arrived task and remaining tasks from previous scheduling again get rescheduled. Hence there is big possibility of starvation of a particular low priority task in this traditional way of priority scheduling.

Starvation is the major problem which is solved somewhat in Round Robin scheduling. But still it is not that efficient as a very high priority task may have to wait for a long time to get execute completely. Hence we consider arrival time along with total cost to decide final priority. To evaluate final priority we use the following equation (5).

$Pj = Cj + Aj$ ..... (5)

Where, P_j: final priority of j^{th} task

C_j: total cost of j^{th} task

A_j: arrival time of j^{th} task

We have seen how to evaluate the final priority of task. Now we know that each task has its original priority and that depends on the priority of the client. Client's priority is decided from the type of subscription, which means for what kind of service, how much amount client pays will decide its priority. According to that we will divide our clients in three types of priorities, high priority, mid priority, low priority. Hence we have three priority queues, high priority queue, mid priority queue and low priority queue. There are separate dedicated resources and application servers to every priority queue. After evaluating final priority we check for its original priority and place it accordingly in one of the priority queues. But where to place in that priority queue will be decided by final priority we have evaluated above using equation (5).Remember, we have done scheduling using the total cost of a complete task, hence if a task contains five activities then all five activities will together while placing in the priority queue and the sequence among the activities of same task will be the same as we derived using DAG.

We will be maintaining a separate waiting queue along with every priority queue. We will see the

to place on server2. But a_{21} is dependent on a_{31} and hence it will not execute unless a_{31} completes its execution. In such scenario if we send a_{21} to execution queue of server2 then it will have to wait until a_{31} finishes its execution at server1. And there will be starvation of activities following a_{21} in the execution queue of server2. Hence in such case instead of sending a_{21} at server2, we keep it in waiting queue and as soon as it receives the feedback of completion of a_{31} it will be send to server2. In this way, waiting queue helps to avoid the unnecessary starvation of activities. To avoid this kind of situation we have given waiting queues to each type of priority queue and there is no dependency among the waiting queues of different priority queues. Now how does feedback mechanism works to alert the activities in waiting queue we will see in this paper later.

Record Tables: Record tables provide information about all types of resources in our cloud computing system and its database. It keeps track of initially available, assigned and available resources in our system. It also keeps track of all tasks, their required resources and servers on which they are going to execute. The information in record tables is very important during the feedback and survey procedure. We have two record tables, one is called as universal resource table and other is activity table.

Table.V Universal resource table

	R_1	R_2	R_3	R_n
Total				
Assigned				
Available				
Server				

Table.5 is showing universal resource table. It contains the total resources, assigned resources and available resources. The server row contains the number of servers on which that particular resource is available. As soon as the activity finishes its execution, a feedback is send to the universal resource table and updates are made in the table. While evaluating resource cost, resource cost evaluation module takes the data from universal resource table for evaluation.

Table.VI Activity table

Activity/resource	R_1	R_2	R_n	S_1	S_n
a_{11}					
a_{22}				Y	
a_{32}					Y
a_{43}					
a_{xy}				Y	

Table.6 is showing activity table. This table contains activities and their corresponding required resources. It also shows server columns, Y entry means that activity requires that server for execution. This table is needed when we do survey and particularly when any activity is in waiting queue to check the servers needed are free or not. As the execution of any activity completes, a feedback is send to the activity table to delete the entry of that activity from the activity table.

The major advantage of maintaining these tables is we can search any activity or resource any time to know the status. As the changes are made in these two tables, the corresponding changes are made in the resource tables of the application servers also.

Feedback and survey : The concept of feedback was necessary for the system to maintain and frequently keep on updating the information about the resources and the tasks (completed or under progress). The universal resource table maintains the record about the resources of various application servers. The individual resource table of each application servers is in sync with the universal table. Thus, changes are made in this universal table accordingly.

Before actually going for the process and need of the feedback, the knowledge of the survey is crucial. The separation of the dependent and independent tasks is to be done as the dependent tasks are placed in the waiting queue unless the respective independent tasks are carried out and the required resources are made available. Thus, a survey of what type of task it is and in which queue the task should be placed in is done. The survey helps in managing the tasks assemble for their respective purpose in the required queue in order to avoid the deadlock situation.

After the task is provided with its required resources and gets completed, the resources of this completed task get free. This new update is carried out in both the tables. Thus, the communication between the individual resource table and the universal resource table can be seen. Now, the current status of availability of resources should be provided to the incoming new tasks. So, instead every task, when completes its required work, sends a feedback regarding its completion and, as a matter of fact, making its resources available to use. The feedback is sensed by the required tasks in the waiting queue, tasks in main priority queue, universal table and also by the individual resource table of each application server. Then the required changes in the various tables are made and the calculation of the cost of new incoming task at that instance of time is calculated. The feedback works as an acknowledgement about completion of a task and resources made available. The feedback and survey is a handy mechanism for reducing the complexity of the working by making the things look so easy and clear to understand. The use of feedback brings the dynamicity in calculating

the cost of the incoming task at any point of time. The task doesn't have to wait at all. Its cost is calculated by taking the demand of resources at that particular instance

Conclusion : In the present study, we propose a new strategy for scheduling the incoming tasks. The scheduling can be done by keeping the cost as the concern. At the first stage, the incoming jobs are taken in a pool. Each job is assigned a cluster independently. The job is then broken into tasks and DAG of each job is obtained in order to avoid deadlock. Considering the DAG, the sequence of tasks is obtained.

At stage two, the cost of each task is obtained and the tasks are scheduled accordingly. This new constructive algorithm defines a new way of scheduling the tasks considering the cost. This scheduling method using costs provides a broad vision of further work possible in this field. Additionally, this algorithm can is designed keeping every type of subscriber in mind. This algorithm looks promising in all aspects for providing an effective scheduling.

References

1. Sandeep Tayal, University School of Information Technology, Guru Gobind Singh, Indraprastha University, Delhi- 10006, India "Tasks Scheduling optimization for the Cloud Computing System2011, IJAEST,2011 2.
2. Archana Ganapathi, Yanpei Chen, Armando Fox, Randy Katz, David Patterson Computer Science Division, University of California at Berkeley," Statistics-Driven Workload Modeling for the Cloud", IEEE 2011
3. QI CAO, ZHI-BO WEI, WEN-MAO GONG International School of Software Wuhan University Wuhan, China, "An Optimized Algorithm for Task Scheduling Based On Activity Based Costing in Cloud Computing", IEEE 2009
4. Jinhua Hu, Jianhua Gu, Guofei Sun, Tianhai Zhao, NPU HPC Center Xi"an, China "A Scheduling Strategy on Load Balancing of Virtual Machine Resources in Cloud Computing Environment", IEEE 2010
5. Jiahui Jin, Junzhou Luo, Aibo Song, Fang Dong and Runqun Xiong School of Computer Science and Engineering, Southeast University Nanjing, P.R. China "BAR: An Efficient Data Locality Driven Task Scheduling Algorithm for Cloud Computing" , IEEE 2011
6. GAN Guo-ning, HUANG Ting-Iei, GAO Shuai School of Computer science and engineering Guilin University of Electronic Technology Guilin, China "Genetic Simulated Annealing Algorithm for Task Scheduling based on Cloud Computing Environment", IEEE 2010
7. Suraj Pandey, Department of Computer Science and Software Engineering, the University of Melbourne, Australia, "Scheduling and Management of Data Intensive Application Workflows in Grid and Cloud Computing Environments", Dec 2010
8. Jiayin Li, Meikang Qiu,Jianwei Niu,Wenzhong Gao Ziliang Zong,Xiao Qin,Dept. of Elec. and Comp. Engg.,University of Kentucky, Lexington, KY 40506, USA. School of Computer Science and Engineering, Beihang University, Beijing 100191, China, Dept. of Elec. and Comp. Engr., University of Denver, Denver, CO 80210, USA, Dept. of Math. and Comp. Sci., South Dakota School of Mines, Rapid City, SD 57702, USA ,Dept. of Comp. Sci. and Software Engr., Auburn University, Auburn, AL 36849, USA," Feedback Dynamic Algorithms for Preemptable Job Scheduling in Cloud Systems", ACM International Conference on Web Intelligence and Intelligent Agent Technology, IEEE 2010
9. Thomas Sandholm and Kevin Lai Social Computing Lab, Hewlett-Packard Labs, Palo Alto, CA 94304, USA," Dynamic Proportional Share Scheduling in Hadoop" JSSPP 2010
10. Navjot Kaur, Taranjit Singh Aulakh, Rajbir Singh Cheema, PTU Jalandhar , India" Comparison of Workflow Scheduling Algorithms in Cloud Computing", IJACSA, Vol. 2, No. 10, 2011
11. Mousumi Paul, Debabrata Samanta, Goutam Sanyal, Department of CSE, National Institute of Technology, Durgapur, West Bengal, India" Dynamic job Scheduling in Cloud Computing based on horizontal load Balancing", IJCTA, Vol. 2 (5), 1552-1556
12. Jiahui Jin, Junzhou Luo, Aibo Song, Fang Dong and Runqun Xiong,School of Computer Science and Engineering Southeast University Nanjing, P.R.

***

Pune University
Pune, 411038, India
ashutosh.ingole@gmail.com