Task Scheduling in Cloud Computing using ATM Approach

Raja Manish Singh
Abhishek Kumar
Priyanka Karn
ME Research Scholar
BIT Mesra, Ranchi, India

Dr. Sanchita Paul
Assistant Professor
BIT Mesra, Ranchi

***Abstract*— Cloud computing is a pool of large, inter-related and virtualized resources that can be accessed over internet on demand by the user. Users get charged according to their use of cloud services. When we perform scheduling of tasks we must have to know about how much time a particular task spend for getting its desired resource from the pool of resources from the point of their submission for execution. In this paper we proposed an algorithm that not only calculates the total execution cost, total execution time as well as the time for respective tasks for finding their cost efficient resources.**

***Keywords*— Scheduling; Task; ATM**

I. INTRODUCTION

Cloud computing in present time is one of the well-known facts for all the peoples as every people in world are somehow related to cloud. The reason behind its popularity is that it provides a simplest way for their users to use the services of cloud. We can say that cloud computing provides service that has no limitations. Users use the cloud services according to their need and also pay accordingly to use.

Cloud computing provides scalable service over internet by using the concept of virtualization, distribution etc. Virtualization means that users do not aware from where it is getting services. Hardware as well as software is virtualized for providing cloud services. It gives users a variety of storage, networking and computing resources in the cloud computing environment via Internet, users put a lot of information and accesses a lot of computing power with the help of its own computer.

Clouds provide a very large number of resources, including platforms for computation, data centers, storages, Networks, firewalls and software in form of services. At the same time it also provides the ways of managing these resources such that users of cloud can access them without facing any kind of performance related problems.

When we talk about of scheduling tasks to various available resources from the pool of resources we talk about a lot of constraint that we have to follow when we develop an approach. Some users want the services cost must be as minimum as possible, some wants reliable services and some wants the services in quick time.

In this paper we proposed an algorithm that schedules the tasks to their cost efficient resources and also at the same time measures the time at which tasks find their cost efficient resources. So, we can get a clear view for all the tasks that how much time they take to find their desire resources from their submission for execution. The remaining paper is organizes as follows: the proposed algorithm is discussed in section II. Section III contains the evaluation of the proposed algorithm and section IV concludes the paper.

II. ACTUAL TIME OF MAPPING ALGORITHM (ATM)

Scheduling of tasks has to be considered under various conditions as there are lots of factor that can affect the scheduling ex: availability of resources, communication between the tasks, and arrival of tasks. The proposed algorithm considers the fact that a task can be completed by various resources but the cost at different resources differs with each other. We keep in mind the fact that whenever a task is ready for execution the availability of resources will change i.e. it gets a new sequence as compare to previous one. Each task has some requirements and cost of execution over resources is completely depending on the fact that how fast a resource provides the service.

Algorithm is developed under the scenario that each tasks are independent to each other, a resource can completely fulfill the requirement of each tasks and batch mode processing. The task when arrives it goes to the task queue and when the space of task queue completely exhausted then the tasks are submitted for execution according to their arrival. When a task is assigned is assign with a resource for execution, the resource will available for that task until the task complete its execution over it.

The ATM algorithm measure the total execution time of executing all the tasks on available resources, total execution cost and most importantly the time at which a task gets its desired resource for its execution i.e. actual task-resource mapping time for each tasks.

2.1 Algorithm for finding the Actual time of mapping tasks to resources:

Step 1. Initialize a global time variable at the starting of code by creating a java calendar instance. (Say calendar is the current object)

Step 2. Get current time in milli-seconds by using calendar.getTimeInMillis(); method call.

Step 3. Store this variable in a Long variable [say ‘now’].

Step 4. Run the task scheduling algorithm as mentioned in section 2.2.

4.1 If cbest is reached, get current time in milli-seconds by using calendar.getTimeInMillis() method call.

4.2 Store this variable in a Long variable [say 'mid']

4.3 Calculate difference of (mid-now) which represents the actual timing of getting cBest (best cost for a cloudlet among virtual machines.)

4.4 Add these cost and node id of cloudlet and virtual machines to cloudLet_cBest hashmap.

2.2 Overall algorithm for calculation of Total Execution time and total Execution cost:

Algorithm (cloudlet_list, vm_list)

{

Step 1 Initialization:

vm_list: virtual machines list.

vid: virtual machine id.

vmac: represents single virtual machine among vm_list.

cloudlet_list: cloudlet list.

cid: current cloudlet Id

cloudlet: represents single cloudlet machine among cloudlet_list.

costPath: represents cost path for current node CiVi.

cloudLet_cBest: is Hashmap which contains mapping of node CiVi and its cost cBest(initially this map is empty).

cTotal: total cost of a cloudlet which zero at initial condition. // initially it is 0(zero) //

cbest: best cost for a cloudlet among virtual machines. // initially it's a very large value like 12354.//

Step 2. Initialize a global time variable at the starting of code by creating a java calendar instance (say calendar is the current object)

Step 3. Get current time in milli-seconds by using calendar.getTimeInMillis(); method call.

Step 4. Store this variable in a Long variable [say 'now'].

Step 5. Generate a random number and on the basis of that random number fetch one cloudlet from the cloudlet_list.

Step 6. Start For Each (cloudlet: cloudlet_list) Loop // this loop iterates over whole cloudlet_List.

6.1 get the cloudlet ID of current cloudlet [say it 'cid'].

6.2 start For Each (vmac: vm_list) Loop choose a random virtual machine vmac among the vm_list. // this loop iterates over whole vm_list randomly.//

6.2.1 Create the cost path for current cloudlet Ci and current virtual machine Vi [say it costPath].

6.2.2 Fetch the current cost for CiVi from task_mat.get(costPath) method call [say it currcost].

// task_mat.get(costPath) returns the current cost of node CiVi.//

6.2.3 Compare currcost with cbest by (currcost > cbest)? cbest: currCost method .//this comparison provide minimum cost //

6.2.4 Add cBest to a list [say list1].

6.2.5 Get the time in milli-seconds as from algorithm 2.1 [say t= 'mid-now' is time taken to achieve current best cost].

6.2.6 Map CiVi and 't' to HashMap cloudLet_cBest.//this map holds current mapping of best cost and node CiVi.

End For Each. ///end of inner virtual machines loop

6.3 create new time variable and update 'now' by now=Calendar.getInstance().getTimeInMillis() method call.

End For Each. //end of outer cloudlet list loop//

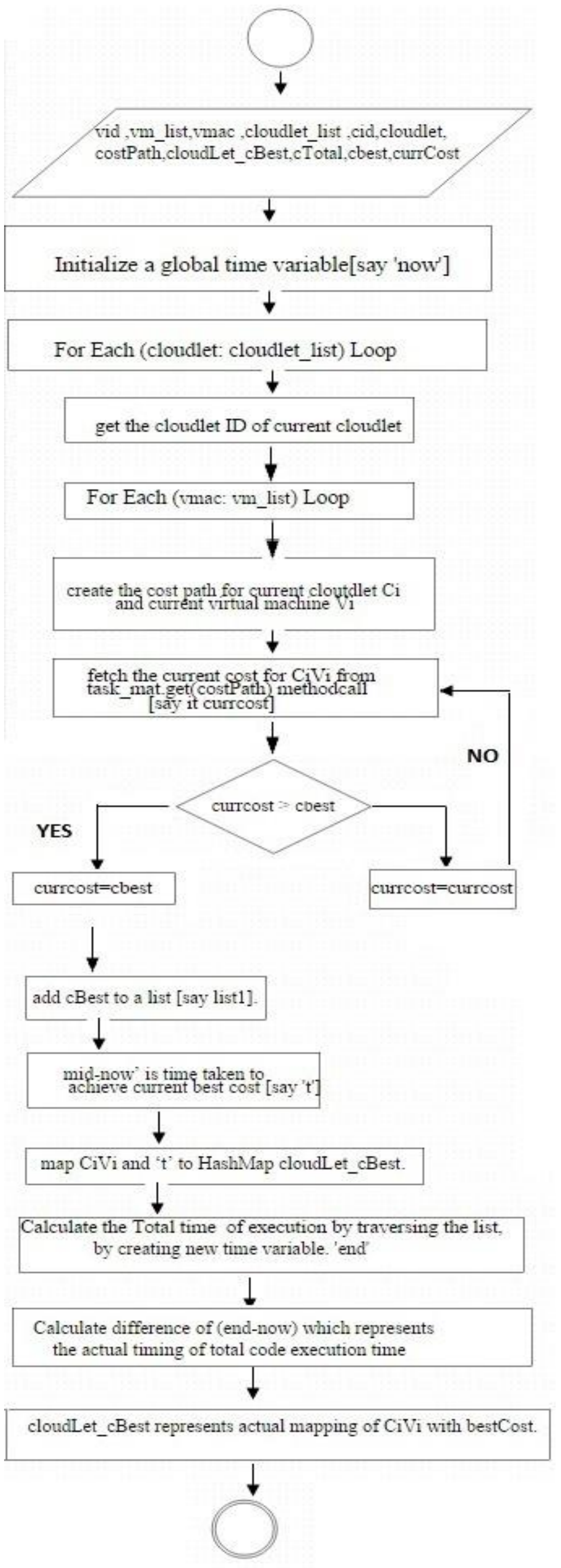
Step 7.

7.1 Calculate the Total Time of execution by traversing the list by creating new time variable 'end'. //end=Calendar.getInstance().getTimeInMillis() method call.//

7.2. Calculate difference of (end-now) which represents the actual timing of total code execution time (time at which all virtual machines and cloudlets gets exhausted.)

7.3 cloudLet_cBest represents actual mapping of CiVi with bestCost.

III. FLOW CHAT OF ATM ALGORITHM


vid ,vm_list,vmac ,cloudlet_list ,cid,cloudlet,
costPath,cloudLet_cBest,cTotal,cbest,currCost
Initialize a global time variable[say 'now']
For Each (cloudlet: cloudlet_list) Loop
get the cloudlet ID of current cloudlet
For Each (vmac: vm_list) Loop
create the cost path for current cloutdlet Ci and current virtual machine Vi
fetch the current cost for CiVi from task_mat.get(costPath) methodcall [say it currcost]
NO
currcost > cbest
YES
currcost=cbest
currcost=currcost
add cBest to a list [say list1].
mid-now' is time taken to achieve current best cost [say 't']
map CiVi and 't' to HashMap cloudLet_cBest.
Calculate the Total time of execution by traversing the list, by creating new time variable. 'end'
Calculate difference of (end-now) which represents the actual timing of total code execution time
cloudLet_cBest represents actual mapping of CiVi with bestCost.

IV. PERFORMANCE EVALUATION

In this section, we are showing the metric, the experimental setup and the result analysis.

4.1 Performance metric

For measuring performance we used cost as well as time as metrics. We compute the total execution cost, total execution time and time of actual mapping of tasks to their cost efficient resources by using the proposed algorithm.

4.2 Data and Implementation

Table: the values shown in the table are an example of the experiment run.

The values in different field signifies that the cost of complete execution of a task on that virtual machine. We take different values by keeping in mind the fact that each vm differs in their MIPS and time for completing their jobs.

TABLE I. TASK-VM COST ASSUMPTION TABLE

	Task 1	Task 2	Task 3
v1	5	10	7
v2	13	5	6
v3	2	4	15
v4	7	14	8
v5	6	7	16
v6	11	8	10
v7	8	6	9
v8	10	13	20
v9	9	12	4
v10	4	9	5

4.3 Experiments and results

For our experiment we have developed a simulation program in java that is coded in java for a cloud computing environment which consists of a data center. Data center consists of ten resources, differs in their processing speed so, prices also differ. We evaluated our scheduling algorithm in different scenario. We keep on changing the sequence the list of tasks and also the sequence of resources over which a task has to be executed. As different vms cost differs for each task we get the total execution time of executing all tasks and also execution time of each tasks from a number of tasks.

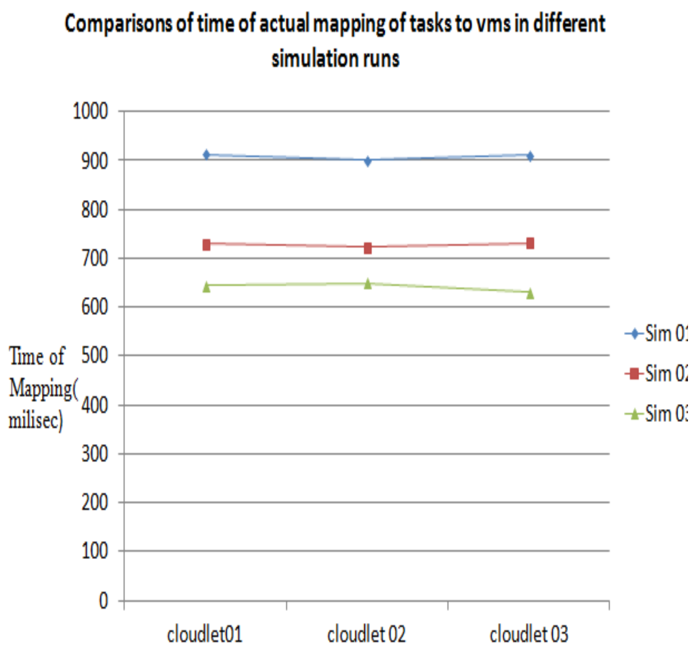


Figure 1: Actual time of mapping of Task to desired Vm

Sequence of cloudlet list in different Simulations [2,3,1], [2,1,3], [3,1,2]

Simulation 01:- C1V3 913, C2V3 900, C3V9 911

Sequence of vms for different tasks according to their arrival:

For C2, sequence is [3,8,1,4,6,9,10,2,5,7]

For C3, sequence is [6,4,5,3,7,2,1,10,8,9]

For C1, sequence is [8,6,1,3,7,2,10,5,4,9]

Simulation 02:- C1V3 731, C2V3 723, C3V9 732

Sequence of vms for different tasks according to their arrival:

For C2, sequence is [4,5,8,10,2,3,9,6,7,1]

For C1, sequence is [8,6,10,4,2,1,7,9,5,3]

For C3, sequence is [4,6,3,5,1,7,2,9,8,10]

Simulation 03:- C1V3 644, C2V3 650, C3V9 630

Sequence of vms for different tasks according to their arrival:

For C3, sequence is [6,4,3,1,2,8,9,7,5,10]

For C1, sequence is [2,5,6,8,7,9,1,10,4,3]

For C2, sequence is [2,6,7,10,9,4,3,1,5,8]

The figure signifies the fact that the times at which different tasks find the desired cost efficient resources for their execution. The time we are calculating in form of millisecond. If we consider the simulation 01, C1V3 913 signifies the fact that task number 1 finds its cost efficient resource after 913 milliseconds prior to start of the working of algorithm. We are using the different sequence of resources for different-different tasks by keeping the fact in mind that when a new task starts its execution the availability of different resources will change.

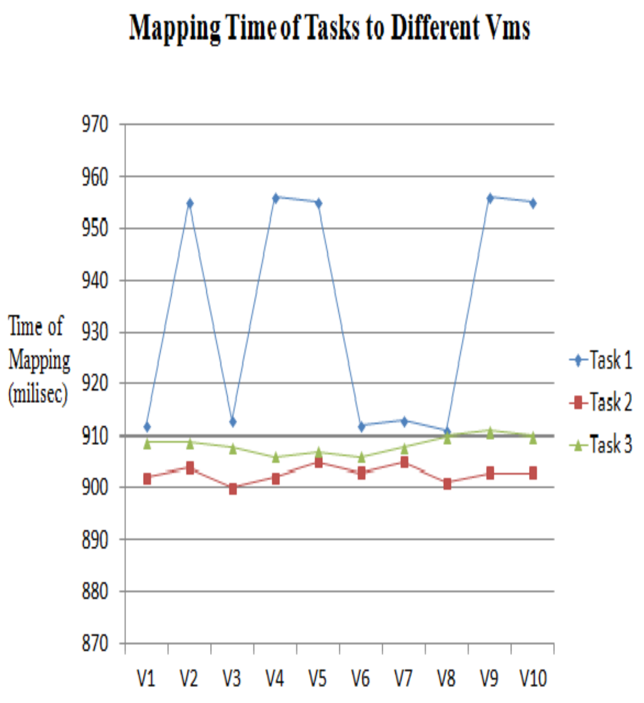


Figure 2: Simulation 01 Result

Figure 2 depicts the fact for one simulation run that shows the time in millisecond how for tasks they reach different-different resources for their execution and after we choose one time of mapping for one task for that particular vm that is most cost-efficient to that tasks for its execution. Table 2 shows the details of graph for our understanding. The field in tables shows the fact that the time in millisecond when the tasks actually reach the resources for their execution from this table we can easily say that when a task actually finds its cost-efficient resource.

TABLE II. ACTUAL MAPPING TIME

	C1	C2	C3
V1	912	902	909
V2	955	904	909
V3	913	900	908
V4	956	902	906
V5	955	905	907
V6	912	903	906
V7	913	905	908
V8	911	901	910
V9	956	903	911
V10	955	903	910

V. CONCLUSION AND FUTURE WORK

In this paper, we have presented Actual Time of Mapping algorithm for scheduling tasks to various resources in cloud computing environment for finding the time at which a task finds its cost-efficient resources. We also calculate the total cost of execution for executing all tasks to resources. The proposed algorithm is developed under strict boundary conditions: tasks are independent to each other, execution of tasks on resources is in non-preemptive manner and we also consider that a resource can completely execute a task when task has been submitted to them for execution. In future we intend to improve our work by considering the fact of dependency between the tasks and the task need more than one resource for their complete execution.

REFERENCES

[1] Zhao, Chenhong, et al. "Independent tasks scheduling based on genetic algorithm in cloud computing." Wireless Communications, Networking and Mobile Computing, 2009. WiCom'09. 5th InternationaConference on. IEEE, 2009.

[2] Guo-ning, Gan, Huang Ting-lei, and Gao Shuai. "Genetic simulated annealing algorithm for task scheduling based on cloud computing environment." 2010 International Conference on Intelligent Computing and Integrated Systems. 2010.

[3] Xu, Yuming, et al. "A multiple priority queueing genetic algorithm for task scheduling on heterogeneous computing systems." High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS), 2012 IEEE 14th International Conference on. IEEE, 2012.

[4] Pandey, Suraj, et al. "A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments." Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on. IEEE, 2010.

[5] Guo, Lizheng, Guojin Shao, and Shuguang Zhao. "Multi-Objective Task Assignment in Cloud Computing by Particle Swarm Optimization." Wireless Communications, Networking and Mobile Computing (WiCOM), 2012 8th International Conference on. IEEE, 2012.

[6] Verma, A., & Kaushal, S. (2014, March). Bi-Criteria Priority based Particle Swarm Optimization workflow scheduling algorithm for cloud. In Engineering and Computational Sciences (RAECS), 2014 Recent Advances in (pp. 1-6). IEEE.

[7] Raja Manish Singh, Sanchita Paul, and Abhishek Kumar. "Task Scheduling in Cloud Computing: Review." (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (6) , 2014, 7940-7944