Double Level Priority based Optimization Algorithm for Task Scheduling in Cloud Computing

Shachee Parikh
Gujarat technological university
kalol institute of technology and research center
kalol, Gujarat, india

Richa Sinha
Gujarat technological university
kalol institute of technology and research center
kalol, Gujarat, india

ABSTRACT
Cloud computing is the fastest new paradigm for delivering on demand services over internet and can be described as internet centric software. In cloud computing there are many tasks that needs to be executed by the available resources to acquire high performance, reduce task completion time, minimize response time, utilization of resource usage and etc.

Scheduling theory for cloud computing is gaining a lot of attention with increasing popularity in this cloud era. Service providers like to ensure that resources are utilized to their fullest and best capacity so that resource power is not left unused. This paper proposes a priority based scheduling optimization algorithm which addresses these major challenges of task scheduling in cloud. The incoming tasks are grouped on the basis of data and requested resources by the task and prioritized. Resource selection is done on the basis of its cost and turnaround time both using greedy approach. Task selection on the basis of a priority formula. This way of resource selection and task selection gives more better results over sequential scheduling.

General Terms
Task scheduling, cloud computing, priority based algorithm.

Keywords
Priority based scheduling, task scheduling, cloud computing, optimization algorithm.

1. INTRODUCTION
Cloud computing is a very current topic and the term has gained a lot of attention in recent times. It can be defined as on demand pay-as-per-use model in which shared resources, information, software and other devices are provided according to the clients' requirement when needed [1]. Human dependency on cloud is evident from the fact that today's most popular social networking, email, document sharing and online gaming sites are hosted on cloud. Google, Microsoft, IBM, Amazon, Yahoo and Apple among others are very active in this field[5].

Scheduling theory for cloud computing is gaining consideration with day by day hike in cloud popularity. In general, scheduling is the process of mapping tasks to available resources on the basis of tasks' characteristics and requirements. It is an essential aspect in efficacious working of cloud as many task parameters need to be considered for proper scheduling. The available resources should be utilized efficiently without affecting the service parameters of cloud. Scheduling process in cloud can be generalized into three stages nam**e**ly–

- Resource discovering and filtering – Datacenter Broker discovers the resources present in the network system and collects status information related to them.
- Resource selection – Target resource is selected based on certain parameters of task and resource. This is deciding stage.
- Task submission -Task is submitted to resource selected[5].

The simplified scheduling steps mentioned above are shown in Figure 1

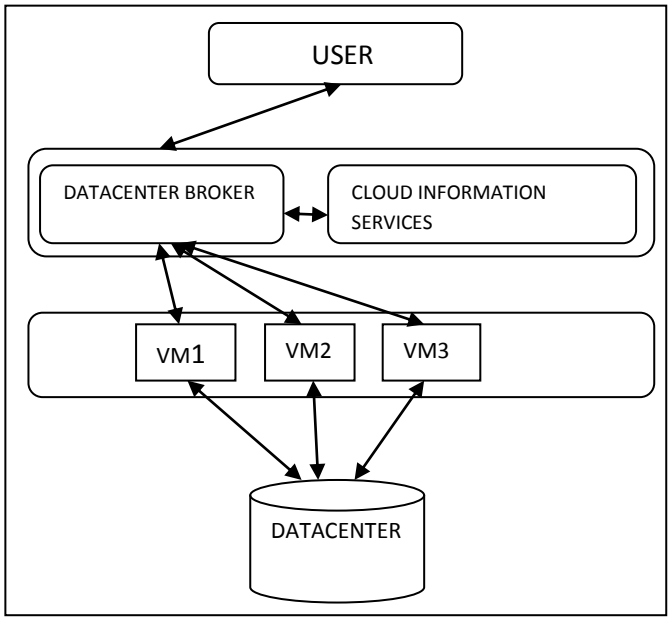

Figure 1. Scheduling in Cloud

The rest of this paper is organized as follows: Section 2 briefly discusses related work followed by proposed framework in Section 3. Next, Section 4 presents the proposed scheduling algorithm and its strategy.

2. RELATED WORK
Target resources in a cloud environment can be selected in various ways. The selection of resources can be either random, round robin, greedy (resource processing power and waiting time based) or by any other means. The selection of jobs to be scheduled can be based on FCFS, SJF, priority based, coarse grained task grouping etc. Scheduling algorithm selects job to be executed and the corresponding resource where the job will be executed. As each selection strategy is having certain flaws work could be done in this direction to extract the advantageous points of these algorithms and come up with a better solution that tries to minimize the drawbacks of resultant algorithm.
The existing algorithms are beneficial either to user or to cloud service providers but none of them takes care of both. Each have their own advantages and disadvantages. Like greedy and priority based scheduling are beneficial to user and grouping

based scheduling is concerned with better utilization of available resources. But the priority based scheduling may lead to long waiting time for low prioty tasks. Greedy scheduling from users point of view lead to wastage of resources whereas greedy scheduling from service providers point of view may lead to disappointment for user on QoS parameters. Similarly task grouping may have the disadvantage of considerable task completion time due to formation of groups. Thus we see that some scheduling strategies are biased to users while others to service providers. There is an emerging requirement to balance this biasing to form an optimized scheduling solution.
New scheduling strategy need to be proposed to overcome the problem posed by network properties and user requirements. The new strategies may use some of the conventional scheduling concepts to merge them with some network and requirement aware strategies to provide solution for better and more efficient task scheduling[5].

3. PROPOSED FRAMEWORK

Task Grouping: Grouping means collection of components on the basis of certain behavior or attribute. By task grouping in cloud it is meant that tasks of similar type can be grouped together and then scheduled collectively[2] .
Activities will be performed on virtual operating systems and the resources are provided over these virtual systems by the original system. There might be tasks which are totally independent or dependent on the other task. There might be some tasks whose all required resources are not available on any single data center. So after considering all possibilities, divide the task into different groups. These groups are, a) Available b) Partially available.

- Available: Dependent and Independent.
- Partially Available: cat1, cat2, cat3…….catN.

This grouping is done on the basis of resources. Available is the group of tasks which can be complete performed on a single data center. And partially available is the group of tasks which will require resources from other data centers. Again independent is the group of tasks which are independent on the other task's result. And hence dependent require some results from previous.
Now the partially available is the group of tasks which needed data from different data centers. Hence further we have sorted them in different categories, cat1, cat2, cat3… and so on till N number of categories. These categories are done on the basis of data need. Cat1 tasks will need the data from same data centers. Similarly cat2 tasks will need the data from same data centers and so on[4].

Implementing task scheduling queue structure

There will be one parent queue in which it will store the tasks according to their arrival time, i.e. first in first out approach. Then it will check for data and requested resources by the task and sort them into two different queues, available and partially available. Again in available queue it will check if the task is dependent or independent and according to that it will store them in their respective queues. Now in partially available queue they are having tasks which will need data resources from other data centers. Then it will sort them in different queues named cat1, cat2… and so on. This will be done on the basis of resources they need. For example if task1 need resources from data center location1 and data center location3 and similar for task2 and task4. Then tasks 1, 2, 4 will be in one category say cat1. Similarly for other tasks we will make cat2, cat3 and so on till we finish the tasks in the partially available queue.

Now they have major queues as, independent, dependent, and cat1, cat2…..catN. For every major queue they will again make three different queues based on priority, High, Mid, Low. Now the question is how to decide the priority?[4]. Decide using below formula.

Prioritization:

The priority level can be sorted by the ratio of task's cost to its profit.
Parameters are defined as followed:
(1) Ri,k: The ith individual use of resources by the kth task.
(2) Ci,k: The cost of the ith individual use of resources by the kth task.
(3) Pk: The profit earned from the kth task.
(4) Lk: The priority level of the kth task.
The priority level of each task can be calculate as in formula (1), the total individual resources use is supposed to be n, so the priority level of the kth task is:[3]

$$L_k = \sum_{i=0}^{n} R_{i,k} \times C_{i,k} / P_k \qquad (1)$$

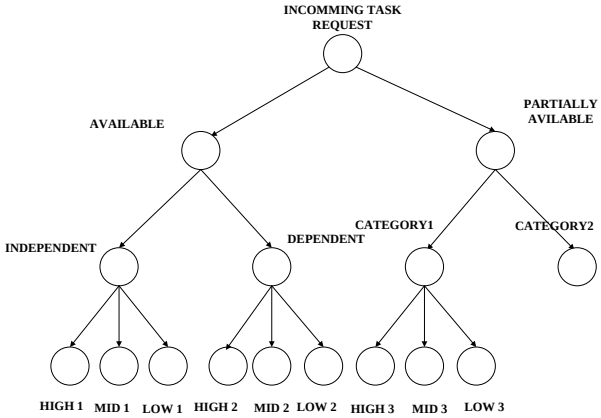

Fig.2 Task Scheduling Tree Structure

While calculating priorities of the task, they will need to check newly calculated priority with the previous ones. Then according to that they will place them into priority queues.
When they will finish placing tasks in their queue then they can select task from High priority queue and when this task will be finished, the first task of the Mid priority queue will be shifted to High priority queue. And in this way tasks from all the queues will start executing.
One question still remain there that what will be the sequence of task selection, as there will be more than one High priority queues. Now it will use a simple logic, as below:

- Compare priority of all tasks which are on number one of every High priority queue.
- Then select the highest priority and assign resources and space.
- Then check if resources are remaining then again choose the next task with the same strategy and allocate the resources and space.
- Repeat the above procedure till the end of all resources.
- When all resources get allocated then waits for any task to finish and as soon as any task finishes then again choose next one and allocate the resources [4].

Greedy Allocation : Greedy algorithm is suitable for dynamic heterogeneous resource environment connected to the scheduler through homogeneous communication environment [7]. Greedy approach is one of the approach used to solve the job scheduling problem. According to the greedy approach –

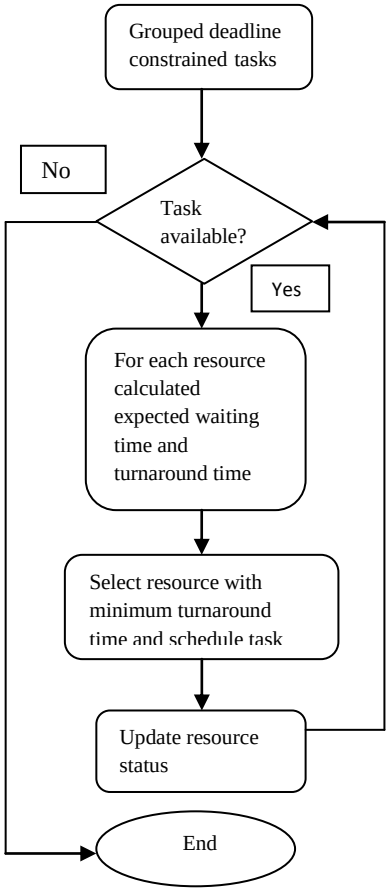

Fig 3.1 Scheduling of Deadline Constrained Tasks

"*A greedy algorithm always makes the choice that looks best at that moment. That is, it makes a locally optimal choice in the hope that this choice will lead to a globally optimal solution*"[8].

Minimum Turnaround Time Based - To improve the completion time of tasks greedy algorithm is used with aim of the resource with minimum turnaround time is given priority than others, resulting in an overall improvement of completion time.

Turnaround Time = Resource Waiting Time + Task Length / Proc. Power of Resource

The task list is rearranged with tasks arranged in descending order of priority in order to execute the task with high priority constraint first. Once the scheduler submits a task to a machine, the resource will remain for some time in processing of that job. The resource status is updated to find out when the resource will be available to process a new job.

Minimum Cost Based - The resource with minimum cost is selected and tasks are scheduled on it until its capacity is supported. After scheduling each task the resource status is updated accordingly

*Cost of Task = (Task length / Proc Power of Resource) * Resource Cost*

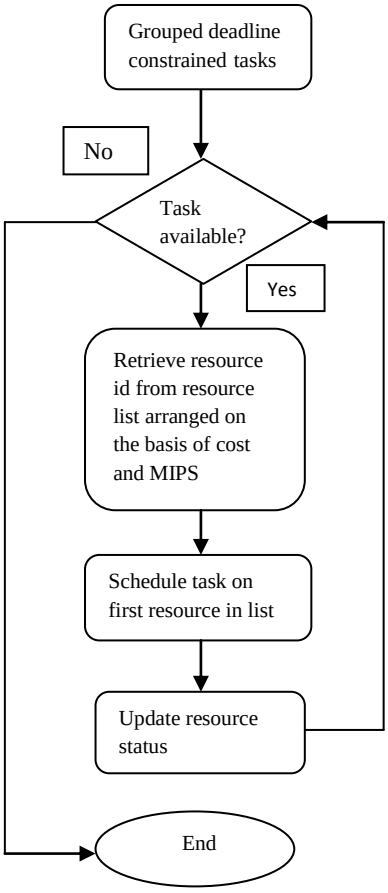

Figure 3.2 Scheduling of Cost Based Tasks

4. PROPOSED ALGORITHM

An optimum scheduling algorithm is proposed and. The proposed algorithm works as follows

1. Incoming tasks to the broker are grouped on the basis of their resources– available and partially available.
Further the grouping is done in available category of task− independent and dependent. And in partially available category, grouping is done on the basis of data need cat1,cat2 and so on.

2. Now we have major queues as, independent, dependent, and cat1, cat2…..catN. For every major queue it will again make three different queues based on priority, High, Mid, Low

3. After initial grouping, the priority level can be sorted by the ratio of task's cost to its profit .

4. i) Virtual Machine are sorted in ascending order on the basis of processing power of machine and its cost .
ii)They are divided into 2 or more equal size groups .

5.starting with first group 1,
i) Turnaround time at each resource is calculated taking following parameters into account.
- Waiting time
- Task length
- Processing Power of virtual machine

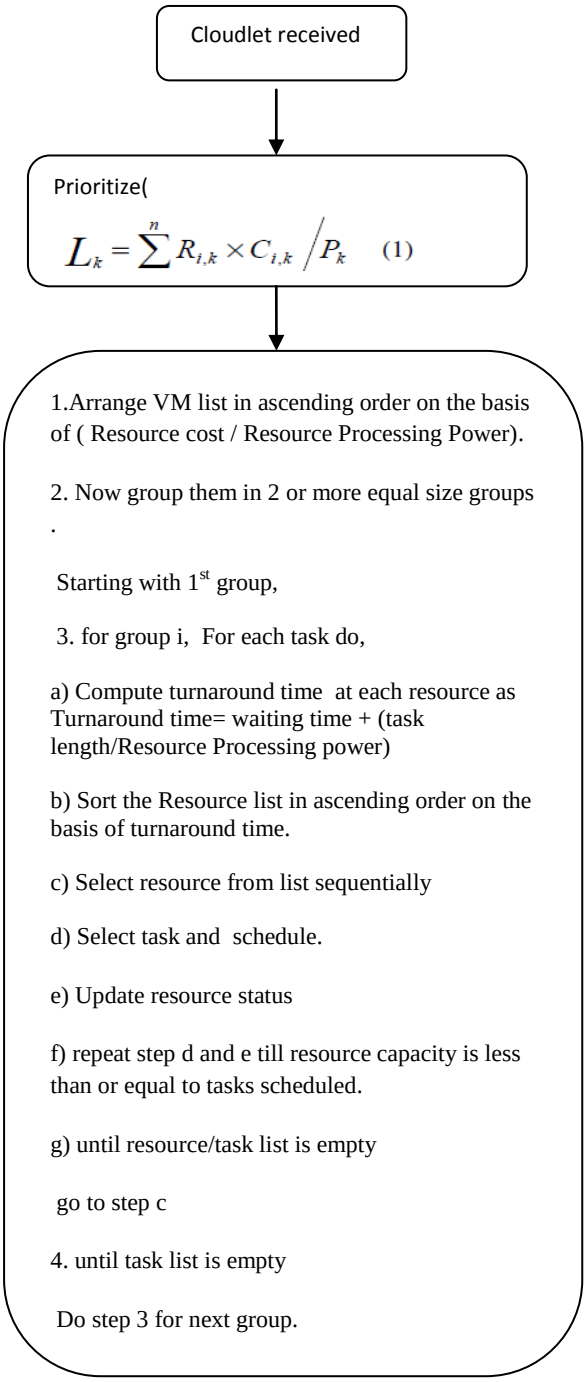

Fig 4.Proposed algorithm

ii)Sort the vm list in ascending order on the basis of turnaround time

iii) select the vm sequentially and schedule task till the resource capacity is permitted

6. Waiting time and resource capacity of selected machine are updated accordingly.

7. when resourcelist is empty in group 1, start with next group and repeat step 5.

5. SIMULATION RESULTS

The CloudSim toolkit is used to simulate heterogeneous resource environment and the communication environment [4]. CloudSim(2.1.1) simulator is used to verify the correctness of proposed algorithm. The experiments are performed with Sequential assignment which is default in CloudSim and the proposed algorithm. The jobs arrival is Uniformly Randomly Distributed to get generalized scenario. The configuration of datacenter created is as shown below - Number of processing elements – 1 Number of hosts – 2

Table 1 Configuration of Hosts

RAM(MB)	10240
Processing Power(MIPS)	110000
VM Scheduling	Time Shared

The configuration of Virtual Machines used in this experiment is as shown in Table 2.

Table 2 Configuration of VMs

Virtual Machines	VM 1	VM 2
RAM(MB)	5024	5024
Processing Power(MIPS)	22000	11000
Processing Element(CPU)	1	1

Performance with cost: The tasks execution using the proposed algorithm results in a significant improvement in cost over the sequential allotment as shown in Table 3.

Table 3 Comparison of Execution Cost

No. Of Cloudlets	Proposed Algorithm	Sequential Assignment
25	565.91	735.68
50	1131.82	1471.36
75	1697.73	2207.05
100	2263.6	2942.73

Performance with time: It is evident from the results that proposed algorithm gives better completion time of job in comparison to the sequential approach.

Table 4 Comparison of Task Completion Time

Cloudlets	Proposed Algo	Sequential Algo
25	565.91	735.68
50	1131.82	1471.36
75	1697.73	2207.05
100	2263.6	2942.73
125	910.04	997.99
150	1298.50	1439.75

6. CONCLUSION AND FUTURE WORK

It is observed that the proposed algorithm improves cost and completion time of tasks as compared to Sequential Assignment. The turnaround time and cost of each job is minimized individually to minimize the average turnaround time and cost of all submitted tasks in a time slot respectively. The results improve with the increase in task count.

The proposed algorithm can be further improved by considering following suggestions –

- In computing environment energy efficient scheduling is a more concern. So,They should try to apply dvfs policy for power saving or can add location parameter with the existing deadline and cost.
- Load balancing parameter if needed could be taken into account for proper scheduling of tasks.

7. REFERENCES

[1] Q. Cao, B. Wei and W. M. Gong, "An optimized algorithm for task scheduling based on activity based costing in cloud computing," In International Conference on eSciences 2009, pp. 1-3. Tavel, P. 2007 Modeling and Simulation Design. AK Peters Ltd.

[2] Ashutosh Ingole, Sumit Chavan, Utkarsh Pawde. "An optimized algorithm for task scheduling based on activity based costing in cloud computing" (NCICT) 2011, Proceedings published in International Journal of Computer Applications® (IJCA)

[3] Monika Choudhary, Sateesh Kumar Peddoju "A Dynamic Optimization Algorithm for Task Scheduling in Cloud Environment" IJERA ISSN: 2248-9622 Vol. 2, Issue 3, May-Jun 2012, pp.2564-2568.

[4] Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities. By Rajkumar Buyya.

[5] S. Singh and K. Kant, "Greedy grid scheduling algorithm in dynamic job submission environment," in *International Conference on Emerging Trends in Electrical and Computer Technology (ICETECT),* 2011, pp. 933-936.

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein *Introduction to algorithms*: The MIT press, 2001, pp 16.