International Journal of Scientific Engineering and Technology (ISSN : 2277-1581)
Volume No.2, Issue No.10, pp : 1062-1068 1 Oct. 2013

A Dynamic Optimization Algorithm for Task Scheduling in Cloud Computing With Resource Utilization

Ram Kumar Sharma,Nagesh Sharma
Deptt. of CSE, NIET, Greater Noida, Gautambuddh Nagar, U.P. India

Abstract - It is a style of computing where massively scalable resources are delivered as a service to external customers using Internet technologies. Scheduling in cloud is responsible for selection of best suitable resources for task execution, by taking some static and dynamic parameters and restrictions of tasks' into consideration. Cloud computing provides us with the massive pool of resources in terms of pay-as-you-use policy. Cloud delivers these resources on demand through the use of network resources under different load conditions. Cloud computing is the next stage in the Internet's evolution, providing the means through which everything — from computing power to computing infrastructure, applications, business processes to personal collaboration — can be delivered to you as a service wherever and whenever you need. Cloud computing has emerged as a popular computing model to support on demand services. As the users will be charged based on their usage the effective utilization of resources poses a major challenge. To accomplish this, a service request scheduling algorithm which reduces the waiting time of the task in the scheduler and maximizes the Quality of Service (QoS) is needed. Our proposed algorithm is based on 3-tier cloud architecture (Consumer, Service Provider and the Resource Provider) which benefits both the user (QoS) and the service provider (Cost) through effective schedule reallocation based on utilization ratio leading to better resource utilization. Performance analysis made with the existing scheduling techniques shows that our algorithm gives out a more optimized schedule and enhances the efficiency rate. The main objective of this paper we are showing the maximum utilization on client and server side accessing the cloud environment.

***Keywords*: Cloud; Service Request; Service Provider; Consumer; Scheduler Units Cloud Scheduling, Optimal Scheduling, Dynamic task execution.**

1. Introduction

Cloud computing an emerging and an enabling technology which made us to think beyond what is possible. Cloud computing services are offered based on 3-tier architecture. The entire architecture of a cloud with respect to service request scheduling comprises of the resource provider, the service providers and the consumers. In order to service the request given by the consumer, the service provider needs either to procure new hardware resources or to rent it from resource provider.

NIET,Greater. Noida

The service provider hires resources from the resource provider and creates Virtual Machine (VM) instances dynamically to serve the consumers. Resource provider takes on the responsibility of dispatching the VM's to the physical server. Charges for the running instance are based on the flat rate (/time unit). Users submit their request for processing an application consists of one or more services. These services along with the time and cost parameters are sent to the service provider. In general the actual processing time of a request is much longer than its estimated time as there incurs some delay at the service provider site. As the cloud is a form of "pay-as- you-use" utility, the service provider needs to reduce the response time and delay. Over here service request scheduling becomes an essential element to reduce maximize the profit of service provider and to improve the QoS offered to the user.

Scheduling process in cloud is generalized into three stages namely:-

- **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

Our algorithm ERUA for service request scheduling schedules the task units based on the utilization ratio of the queue and greedy algorithm for Dynamic scheduling. It always ensures that the utilization ratio always falls within 1 leading to better resource utilization and enhancing the efficiency through enabling the task units to finish up its execution within their deadline. With our sample set of data, ERUA proves to be more optimal than the existing algorithms for service request scheduling. 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. The simplified scheduling steps mentioned above are shown in **Figure 1** balance this biasing to form an optimized scheduling

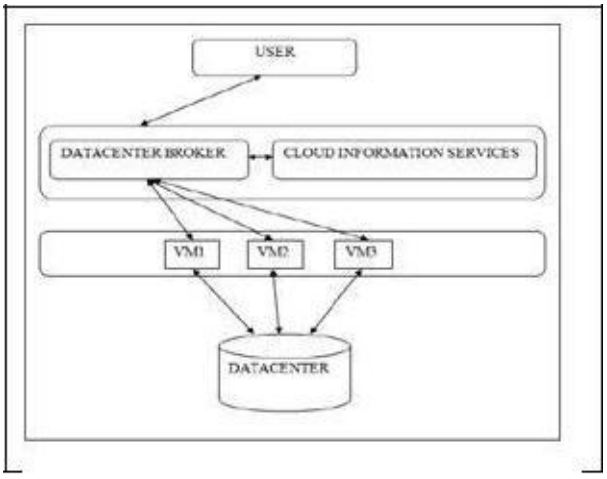


Figure 1. Scheduling in Cloud

The remainder of the paper is sectioned as follows: Section 2 Related work Section 3 proposed methodology for request and response scheduling Section 4 proposed algorithm, Section 5 discusses about the results and Section 6 concludes this paper.

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 priority 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 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.

3. Proposed Methodology For Request And Response Scheduling

3.1 Response based Scheduling

i) 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].** We can say that it is a behavior that supports the creation of 'sets of tasks' by some form of commonality. In the proposed framework tasks are grouped on the basis of constraint which can be deadline or minimum cost. Once the tasks are grouped, they can be judged for their priority and scheduled accordingly. Grouping, if employed to combine several tasks, reduces the cost-communication ratio

ii) Prioritization: Priority determines the importance of the *element with which it is associated. In terms of task scheduling, it determines the order of task scheduling based on the parameters undertaken for its computation [3]. In the present framework, the deadline based tasks are prioritized on the basis of task deadline. The tasks with shorter deadline need to be executed first. So they are given more priority* in scheduling sequence. The task list is rearranged with tasks arranged in ascending order of deadline in order to execute the task with minimum time constraint first. The cost based tasks are prioritized on the basis of task profit in descending order. This is appreciable as tasks with higher profit can be executed on minimum cost based machine to give maximum profit.

iii) Greedy Allocation : Greedy algorithm is suitable for dynamic heterogeneous resource environment connected to the scheduler through homogeneous communication environment [4]. Greedy approach is one of the approach used to solve the job scheduling problem.
According to the greedy approach –

> "*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*" [5].

iv) Deadline Constrained Based - To improve the completion time of tasks greedy algorithm is used with aim of minimizing the turnaround task of individual tasks,
resulting in an overall improvement of completion time.

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

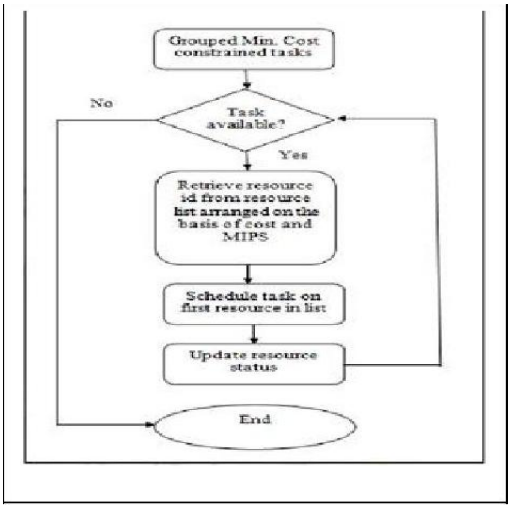


Figure 1.1 Scheduling of Deadline Constrained Tasks

After calculating the turnaround time for each resource, the resource with minimum turnaround time is selected and task is executed there. The scheduler locates the best suited resource that minimizes the turnaround time. The turnaround time is calculated on the basis of expected completion time of a job. 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.

v) 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. Thus the selection of task and target resource is sequential once they are prioritized according to user needs.

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

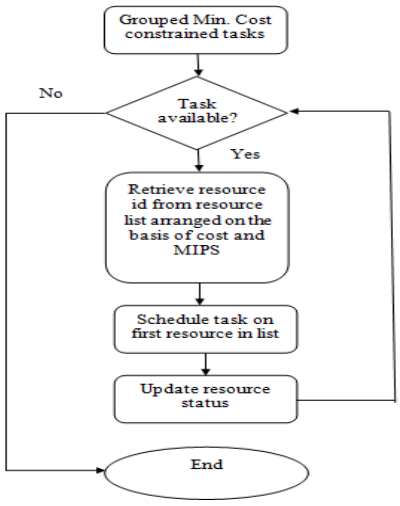


Figure 1.2 Scheduling of Cost Based Tasks

3.2 Request based Schedule

Process a) Scheduling Process

The process of scheduling can be viewed as service request scheduling (service provider and the Consumer) and resource scheduling (service provider and resource provider). The process of service request scheduling occurs as:

a) Users submit their request to the service provider.
b) Service provider executes the request.
c) Process the request in the service request architecture.
d) Dynamic VM generation and dispatch at the resource provider site.

b) System Architecture

The major components in the service request scheduling are (Figure 2):

i. Classifier: Receives user request, process and classifies into smaller task units. These task units can be scheduled directly onto the scheduler but before that it needs to get assigned with random priorities. Priority can either be based on system state or the task characteristics. Once each task gets its unique priority these task units can be sent to the scheduler component to be scheduled.

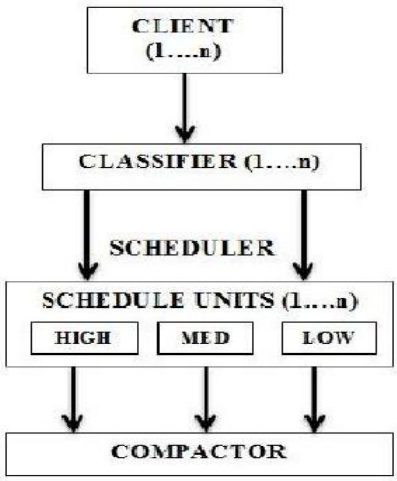


Figure 2. System Architecture of ERUA.

ii. Scheduler: Each scheduler contains several schedule units, each having its own priority based on the system design and the real situation. Scheduler pushes up the task units into appropriate schedule units based on the idleness and the saturation of each and every schedule unit. Scheduler units execute the task units based on the algorithm. The task unit with the lower deadline will be scheduled first to optimize the result.

iii. Compactor: Summarizes the completed task units during each cycle and sends it to the resource provider.

c)The Process of Service Request Scheduling

Users submit their request for executing their application which consists of one or more services to the SP. Now the SP has to perform the service request scheduling process with these requests and has to operate on a massive set of data. So the SP requires a scheduler to efficiently schedule these request maximizing the QoS to the user and the profit on the SP site. The process of service scheduling starts here. Each request will be spliced into task units and are assigned with some random priority in the classifier. Classifier pushes these task units into an appropriate scheduler units based on the state of the scheduler units. Scheduler units execute the task unit based on some algorithm. Our algorithm considers utilization ratio as the deciding factor for priority reassignment. Let us consider an example for priority reassignment **(Figure 3)**

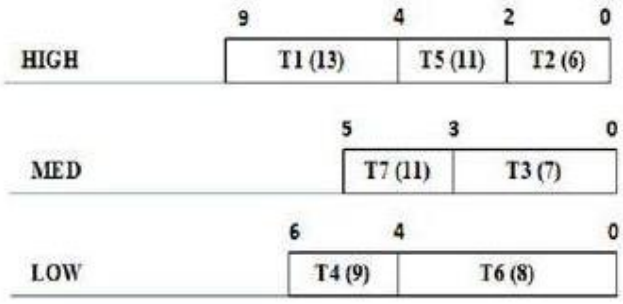


Figure 3. Example of scheduling task in prioritized queue.

Task units T2 (6), T3 (14) and T6 (8) having the lowest deadline will be assigned for execution at first in high, medium and low priority scheduler units respectively. Now after a cycle the remaining task units will be high queue - T5 (11) and T1 (13), medium queue - T7 (11) and low queue - T4 (9) and T6 (8). If there is a task T8 (19) with the execution time of 7 ms in the higher priority queue that needs to be executed after T1 (13), it can be scheduled to execution in the medium priority scheduler unit after T7 (11) as it frees up after 5 ms. This can be done by analyzing the remaining jobs and the completion time of the current jobs scheduled in the queue, thus minimizing the delay of 1 ms, while enhancing the processor utilization. Now, T8 (14) completes its execution by 12 ms within its deadline. Whenever the queue frees up irrespective of the priority class, the tasks can be scheduled onto any one of them based on the state of the scheduler units. Priority reassignment based on deadline gives us a better way of maximizing the throughput and the performance of the system through effective resource utilization.

4. Proposed Algorithm

An optimum scheduling algorithm is proposed and implemented in this section. The proposed algorithm works as follows Incoming tasks to the broker are grouped on the basis of their type- deadline constrained or low cost requirement. After initial grouping they are prioritized according to deadline or profit. This is required because the tasks with shorter deadline need to be scheduled first and similarly the tasks resulting in more profit should be scheduled on lost cost machines. Thus, the prioritizing parameter is different based on the nature or type of task.

A. For each prioritized task in deadline constrained group -

i) Turnaround time at each resource is calculated taking following parameters into account.

- Waiting Time
- Task Length
- Processing power of virtual machine

ii) The virtual machine with minimum turnaround time that is capable to execute the task is selected and task is scheduled for execution on that machine.

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

B. For cost based group

i) Virtual Machine are selected on the basis of processing power of machine and its cost

ii) For each virtual machine cloudlets from the group are scheduled till the resource capacity is permitted.

iii) Resource capacity and waiting time are updated accordingly

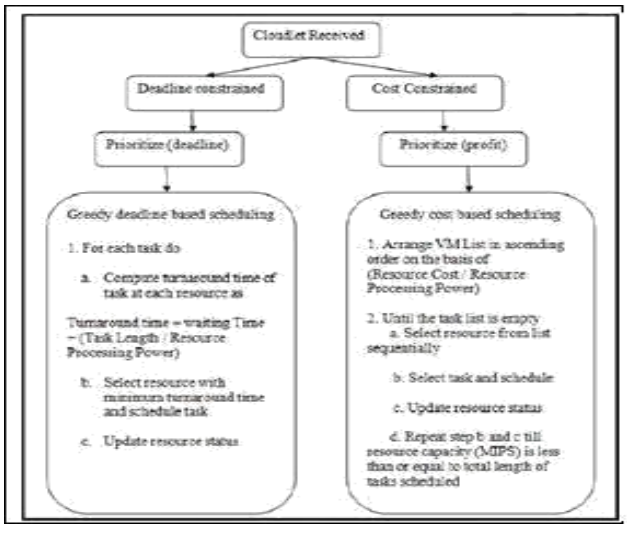


Figure 4 Proposed Algorithms

5. Results and Discussions

The CloudSim toolkit is used to simulate heterogeneous resource environment and the communication environment [6, 7]. 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	Times Shared

The configuration of Virtual Machines used in this experiment is as shown in **Table 3**.

Table 2 Configuration of VMs

Virtual	**VM1**	**VM2**
Machine	5054	5024
RAM(MB)	22000	11000
Processing	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**.

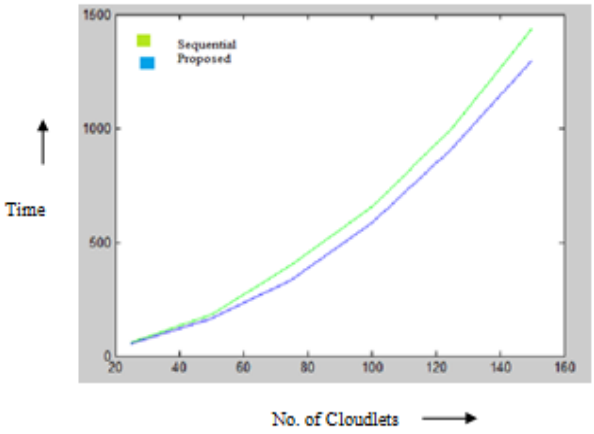
Table No. 3

NO. of Cloudlets'	**Proposed Algorithm**	**Sequential Assignment**
25	565.91	735.36
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 algorithm**	**Sequential Algorithm**
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



User submits their request to the service provider and it enters the scheduling architecture through the classifier.

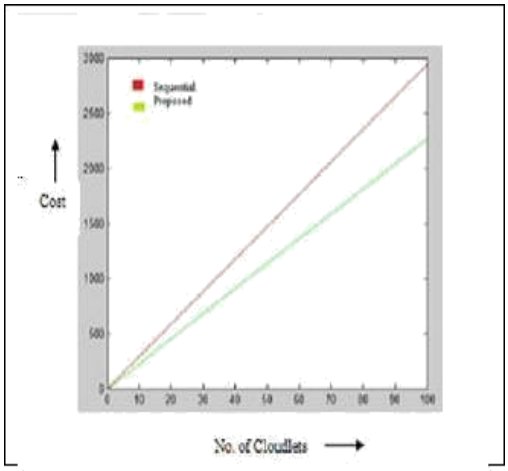
The classifier component split up the user request into several independent task units. Let us consider the following table which consists of several task units of a single request (Table 1). Now, the classifier assigns some initial priority (least deadline) to each task units and schedules them on to the schedule units. Here, we will be having three scheduler queues with high, medium and low priority respectively and this depends upon the design and the current load of the system.

The initial tasks scheduled to execution will be T2 (6) and T6 (11) in high priority queue, T3 (14) and T7 (17) in medium priority queue and T5 (8) and T10 (9) in low priority queue are shown in Figure 3. T8 (13) with the higher priority will be scheduled next to T6 (11) in the high priority queue **(Figure 5).** Always be sure about

Utilization Ratio i(Queue) = (Execution Time $_i$ / Deadline $_i$) □ 1

Table 5. Task Units Schedule.

TASK	EXECUTION TIME	DEAD LINE	PRIORITY
T1	8	50	HIGH
T2	2	6	HIGH
T3	3	14	MEDIUM
T4	4	12	LOW
T5	4	8	LOW
T6	2	11	HIGH
T7	2	17	MEDIUM
T8	5	13	HIGH
T9	3	45	HIGH
T10	2	9	LOW
T11	3	13	LOW



The task units T2, T6, T8, T3, T7, T5 and T10 were scheduled on to execution within their deadline with the utilization ratio of 0.89 (T2, T6 and T8) on high priority queue, 0.33 (T3 and T7) on medium priority queue and 0.72 (T5 and T10) on low priority queue. To keep the queue busy, always ensure that the queue utilization should be within 1. Write down the remaining task that needs to be scheduled **(Table 1) below**.

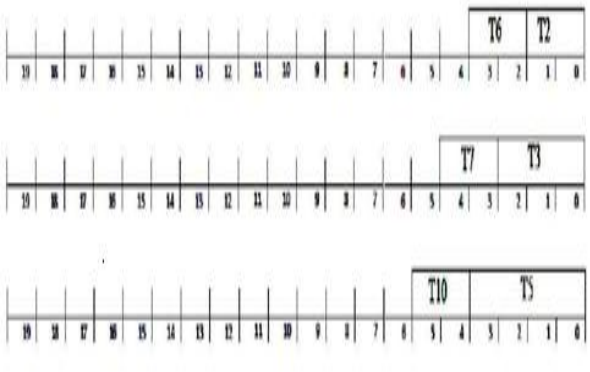


Figure 5. Initial Schedule Based on the Least Deadline.

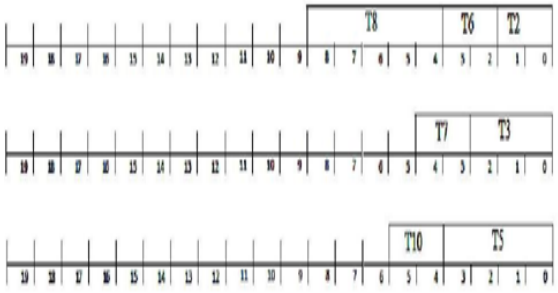


Figure 6. T8 with high priority scheduled on to high priority queue.

PRIORITY	TASK	EXECUTION TIME	DEAD LINE
HIGH PRIORITY	T1	8	50
	T9	3	45
	T4	4	12
LOW PRIORITY	T11	3	13

Table 2. Consolidated task details with priority reassignment based on deadline.

The task T4 (12) with the low priority will be scheduled on the medium priority queue after T7 (17) as T7 (17) completes 1ms before T10 (9) in the low priority queue (Figure 5). The task T11 (13) with the low priority will be scheduled on the low priority queue after T10 (9) (Figure 6). The task T9 (45) with the high priority will be scheduled on the high priority queue after T8 (13) as T9 (45) will have the least deadline than T1 (50) (Figure 7). The task T1 (50) with the high priority will be scheduled on the medium priority queue after T4 (12) as T4 (12) completes 4ms before T9 (45).
As per this schedule

ALGORITHM	EFFICIENCY
FCFS	29%
SPSA	74%
EDF	56%
DPSA	84%
ERUA	99%

Table 3. Efficiency (%)

6. CONCLUSION

Users focus on the QoS whereas the service providers rely on maximizing their profit. To satisfy both the userand the service providers we need an efficient service request scheduling algorithm in a cloud computing platform. Our algorithm satisfies the requirement of both the users and the service providers through efficient schedule and priority reassignment. It services the SLA model of the user and the cost model for the service provider through dynamic resource reuse management. Our future work investigates on evaluating the users SLA model and the service provider profit model under different load condition. 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 –

- The future work may group the cost based tasks before resource allocation according to resource capacity to reduce the communication overhead.
- Other factors like type of task, task length could be taken into account for proper scheduling of tasks.

7. **REFERENCES**

i) *R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose and R. Buyya "CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," Software: Practice and Experience, vol. 41, no. 1, pp. 23-50, 2011.*

ii) *N. Calheiros, R. Ranjan, C. A. F. De Rose, and R. Buyya, "Cloudsim: A novel framework for modeling and simulation of cloud computing infrastructures and services," Arxiv preprint arXiv:0903.2525, 2009.*

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

iv) *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.*

v) *J. Chen, C. Wang, B. Zhou, L. Sun, Y. Lee and A. Zomaya, "Tradeoffs between profit and customer satisfaction for service provisioning in the cloud", International Symposium on High Performance Distributed Computing, 2011, pp. 229-238.*

vi) *Buyya, Rajkumar, Rajiv Ranjan, and Rodrigo N. Calheiros. "Modeling and simulation of scalable Cloud computing*

vii) *Computing (CCGrid), 2011, pp 195-204.*

viii) *Linan Zhu, Qingshui Li and Lingna He, "Study on Cloud Computing Resource Scheduling Strategy Based on the Ant Colony Optimization Algorithm", International Journal of Computer Science Issues (IJCSI), 2012, pp 54-58.*

ix) *International Conference on Cluster, Cloud and Grid Computing, pp 15-24. Zhipiao Liu, Shangguang Wang, Qibo Sun,*

x) *Keerthana Boloor, Rada Chirkova and Yannis Viniotis, "Dynamic request allocation and scheduling for context aware applications subject to a percentile response time SLA in a distributed cloud ", IEEE International Conference on Cloud Computing Technology and Science, 2011, pp 464-472.*

xi) *Linlin Wu and Rajkumar Buyya , "Service Level Agreement (SLA) in Utility Computing Systems, Technical*

xii) *Report, 2010. Linlin Wu, Saurabh Kumar Garg and Rajkumar Buyya, "SLA-based Resource Allocation for Software as a Service Provider (SaaS) in Cloud Computing Environments", IEEE/ACM International Symposium on Cluster, Cloud and Grid*

xiii) *Computing (CCGrid), 2011, pp 195-204.*

xiv) *Linan Zhu, Qingshui Li and Lingna He, "Study on Cloud Computing Resource Scheduling Strategy Based on the Ant*

xv) *Colony Optimization Algorithm", International Journal of Computer Science Issues (IJCSI), 2012, pp 54-58.*

xvi) *Noha El.Attar, Wael Awad and Fatma Omara, "Resource Provision for Services Workloads based on (RPOA)",*

xvii) *International Journal of Computer Science Issues (IJCSI), 2012, pp 553-560.*

xviii) *Rajiv Ranjan, Liang Zhao, Xiaomin Wu, Anna Liu, Andres Quiroz and Manish Parashar, "Peer-to-Peer Cloud*

xix) *Provisioning: Service Discovery and Load-Balancing", Cloud Computing Computer Communications and Networks, 2010, pp 195-217.*