Volume 3, Issue 11, November 2013 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

A Survey of Proposed Job Scheduling Algorithms in Cloud Computing Environment

Rohit O. Gupta
Computer Science & Engineering,
L.D.College of Engineering, India

Tushar Champaneria
Assistant Professor of Computer Engg. Department,
L.D.College of Engineering, India

Abstract — Cloud computing provides reliable, customized and guaranteed dynamic computing environments for end-users, so that more and more people tend to adapt the cloud environment in their workplace. Cloud computing is becoming an increasingly popular enterprise model in which computing resources are made available on-demand to the user as needed. The unique concept of cloud computing creates new opportunities to align Business and IT goals. Job scheduling is an essential requirement in cloud computing environment with the given constraints. Some intensive researches have been done in the area of job scheduling of cloud computing. The scheduling algorithms should order the jobs in a way where balance between improving the performance and quality of service and at the same time maintaining the efficiency and fairness among the jobs. This paper aims at studying various scheduling algorithms recently proposed in cloud computing.

Keywords — Cloud Computing, Job Scheduling, Scheduling Algorithm.

I. INTRODUCTION

The US National Institute of Standards and Technology (NIST) has developed a working definition that covers the commonly agreed aspects of cloud computing. The NIST working definition summarizes cloud computing as [3]:

"Cloud computing is a model enabling ubiquitous, ,convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."

Cloud Computing provide the opportunity to access IT resources and services with high speed and efficiency. It implies a service oriented architecture through offering software and platform as services, reduced information technology overhead for the end-user, reduced ownership cost, higher flexibility, on demand services etc. Cloud services enable individuals and businesses to use software and hardware from remote locations that are managed by third parties. A Cloud service provider provides cloud services based on the concept of "pay per use service" available via public or private network to various clients. The main enabling technology for cloud computing is virtualization. Virtualization generalizes the physical infrastructure and provides it as a soft component that is easy to be used and managed. It helps to speed up IT operations, and reduces cost by increasing recourse utilization. [1] Cloud computing also borrows some concepts from utility computing so that it can provide metrics for the services used. Such metrics are at the core of the public cloud pay-per-use models. The huge growth in cloud computing technologies reflects the increasing number of jobs that require the computing services. Traditional job scheduling algorithms are not able to provide scheduling in the cloud environments. Various types of scheduling algorithms have been applied on various data sets and measured with different parameters to evaluate the performance. Most of the tasks scheduling algorithms are developed to achieve two things. The first one is to improve the quality of services of computing environment and provide the expected output on time. The second is to maintain efficiency and fairness among the jobs. [2]

Figure 1 illustrates the cloud frame-work based on job scheduling which consists of three tiers, namely, the cloud provider, the internet and the connected clients. There are various study works available which propose different scheduling algorithms. Some of these proposed algorithms are specifically for scheduling jobs in a cloud computing environment and some are adapted in the cloud environment. This survey paper aims at studying and analysis various scheduling algorithms recently proposed in cloud computing.

Isam Azawi Mohialdeen / Journal of Computer Science 9 (2): 252-263, 2013

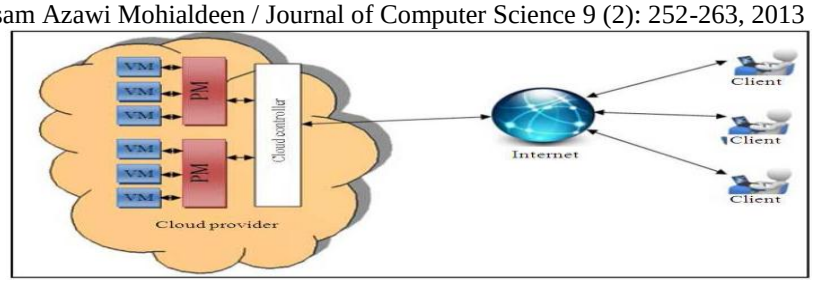


Figure 1: The proposed cloud framework

II. SCHEDULING

There have been different types of job scheduling algorithm applied in the cloud environment with suitable modification. The main aim of job scheduling algorithm is to improve the performance and quality of service and at the same time maintaining the efficiency and fairness among the jobs and reduce the execution cost. Traditional job scheduling algorithms are not able to provide scheduling in the cloud environments. Based on a simple classification, job scheduling algorithms in cloud computing can be categorized into two main classes; Batch mode heuristic scheduling algorithms (BMHA) and online mode heuristic algorithms. In BMHA, Jobs are taken into a collection when they arrive in the system. The scheduling algorithm starts its work after a fixed amount of time. Examples of BMHA based algorithms is Round Robin algorithm (RR).In On-line mode heuristic scheduling algorithm, Jobs are scheduled individually as soon as they arrive in the system. The on-line mode heuristic scheduling algorithms are more suitable for a cloud environment Because of the cloud environment's heterogeneity and varying processor speed, Example of On-line mode heuristic scheduling algorithm is Most fit task scheduling algorithm.[6]

.

A. First Come First Serve Algorithm

Jobs are served in queue as they come. This algorithm is simple and fast.

B. Round Robin algorithm

In the round robin scheduling, processes are given a limited amount of CPU time called a time-slice or a quantum in FIFO manner. If a process does not complete execution before its CPU-time expires, the CPU is pre-empted and given to the next process waiting in a queue. And the preempted process is placed at the end of the ready queue.

C. Random Algorithm

In random algorithm, the selected jobs are randomly selected for execution and assigned to Virtual Machine. The algorithm does not take into considerations the status of the VM, which will either be under heavy or low load.

D. Max – Min algorithm

This algorithm chooses larger tasks to be executed firstly, which make small task delays for long time.

E. Priority scheduling algorithm

Each process is assigned a priority, and then based on priority processes are allowed to be executed. Equal-Priority processes are scheduled in FCFS manner.

F. Most fit task scheduling algorithm

Job which fit best in queue based on some parameters is executed first.

III. MATERIALS AND METHODS

In this study, five proposed job scheduling algorithms of Cloud computing were carefully selected for study and evaluation, namely,

- Short Job Scheduling
- Job Scheduling Model based on Multi-Objective Genetic Algorithm
- Priority based Job Scheduling Algorithm
- SLA-Tree
- Enhanced Max-min Task Scheduling Algorithm

A. Short Job Scheduling

Poonam Devi [4] proposed an efficient process allocation algorithm Short Job scheduling in multiple clouds. It is capable to handle the over load condition and for that the load distribution scheme is presented in this work. To perform the effective allocation, some priority is assigned to each cloud.

The objectives of proposed system are following:

1) Create an Intermediate Architecture that will accept the user request and monitor the cloud servers for their capabilities.
2) Scheduling of the users requests is performed to identify the order of allocation of the processes.
3) Performing the effective resource allocation under defined parameters and the cloud server capabilities.
4) Define a dynamic approach to perform the process migration from one cloud to other.
5) Analysis of the work using graph under different parameters

Figure 2 demonstrate the architecture of proposed job scheduling system.

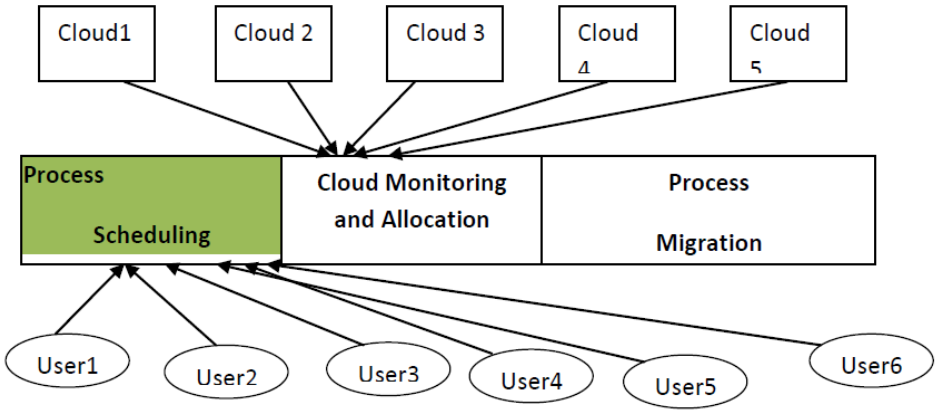


Figure 2: Short job scheduling system

This system is middle layer architecture to perform the cloud allocation in case of under load and overload conditions. The process migration will handle the over load conditions. The middle layer will exist between the clouds and the clients. The middle layer accepts the request coming from the user side and the analysis of the cloud servers is performed by this middle layer. The middle layer is responsible for three main tasks:

1) Perform the allocation of process and Monitor the cloud server's capabilities
2) Process Migration in overload conditions
3) User requests scheduling

The detailed steps of Short job scheduling are illustrated in Fig. 3. The algorithm input includes two sets of information, 1.User process requests with some parameters specification, 2.Number of Clouds and its associated Virtual Machines. First the requests are arrange in order of their memory requirement. Than one by one each process is initially allocated to particular VM and the cloud if it has enough available memory.
This step doesn't consider the dead line of process. In second step the free time slot of cloud are identify for reallocation of process. And in final step processes with finishing time greater than its deadline are migrated to new virtual machine having enough free memory and time slot.

Algorithm

1. Input the M number of Clouds with L number of Virtual Machines associated with Each cloud.
2. Define the available memory and load for each virtual machine.
3. Assign the priority to each cloud.
4. Input N number of user process request with some parameters specifications like arrival time, process time, required memory etc.
5. Arrange the process requests in order of memory requirement
6. For i=1 to N
7. {
8. Identify the priority Cloud and Associated VM having AvailableMemory > RequiredMemory(i)
9. Perform the initial allocation of process to that particular VM and the Cloud
10. }
11. For i=1 to N
12. {
13. Identify the Free Time slot on priority cloud to perform the allocation. As the free slot identify, record the starttime, process time, turnaround time and the deadline of the process.
14. }
15. For i=1 to N
16. {
17. If finishtime(process(i))> Deadline(Process(i))
18. {
19. Print "Migration Required"
20. Identify the next high priority cloud, that having the free memory and the time slot and perform the migration of the process to that particular cloud and the virtual machine.
21. }
22. }

Figure 3: Short job scheduling algorithm

B. Job Scheduling Model based on Multi-Objective Genetic Algorithm

Jing Liu, Xing-Guo Luo, Xing-Ming Zhang, Fan Zhang and Bai-Nan Li [5] proposed a scheduling model for cloud computing based on MO-GA algorithm to minimize energy consumption and maximize the profit of service provides

under the constraint of deadlines. In this scheduling model, the energy consumption and the profits of the service providers are taken into account, and providing a dynamic selection mechanism of the most suitable scheduling scheme for users according to the real-time requirements.

Fig. 4 illustrates the functional architecture of the scheduling model.

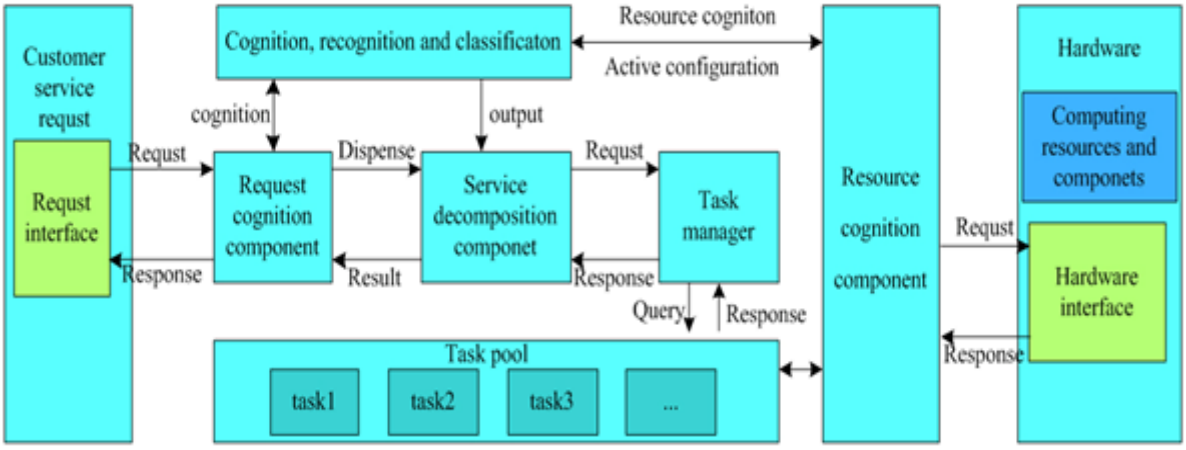


Figure 4: Functional Architecture of Job Scheduling Model for Cloud Computing

The Functionalities of the main components are as follows:

Request cognition component should have information about needs for different businesses, which include the computing, storage and communication requirements for computing, security and privacy requirements, Quality of service etc. Service decomposition component decomposes the service request into different level of granularities with different processor preferences. Task manager analyse the resource requirements of each granularity, and mapping it on to optimal processors to reach an effective solution. It is responsible for task status management (start, stop, cancel…), determining the scheduling sequence and resource assignment for the requests and allocating suitable resources to each job under the help of the scheduling algorithm. Resource cognition component manage the available resources, monitoring of the performances of resources, dynamic optimization of scheduling strategy and error notification.

Following operations of Genetic Algorithms are used:

1) Encoding Rule

Each schedule is expressed as a 2XM matrix, where, M is the length of the chromosome. The first row of the matrix represents the requested applications, and second of the matrix is the corresponding number of the cloud where the application is executed.

2) Population Initialization

The population initialization is an important step in algorithm because it affects the quality of the future generations. This step is conducted by combing the random and greedy initialization methods.

3) Individual Evaluation

The fitness of chromosomes is deduced from the energy consumption and profits of the service providers.

Following operations are performed over the chromosomes:

1) Selection operation

The selection operation is based on tournament operator of k individuals, with two strategies: elitism and crowding.

2) Crossover Operation

The crossover operator uses two individuals s1, s2 to generate two new individuals s1', s2'.

3) Mutation Operation

The mutation operation chooses two tasks in an individual randomly, and swaps their allocation position to generate a new individual

Implementation Steps

Based on the above, the implementation steps of this algorithm are listed following:

1) Initial the population by greedy and random methods;
2) Modify the individual during the evolution process of the MO-GA algorithm according to the above mentioned GA operators and store the results to external Pareto archive;
3) Select the optimal solution according to the vector and implement the scheduling result to distributed cloud federation.

Several experiments are conducted with MO-GA and other scheduling algorithms and their results are evaluated and compared to validate the efficiency of this scheduling model.

C. Priority based Job Scheduling Algorithm

Shamsollah Ghanbari and Mohamed Othman [6] proposed a priority based job scheduling algorithm named it "PJSC" which can be applied in cloud environments. This Scheduling algorithm is consisted of three levels of priorities including: scheduling level (objective level), resources level (attribute level) and job level (alternative level), as shown in Fig 5.

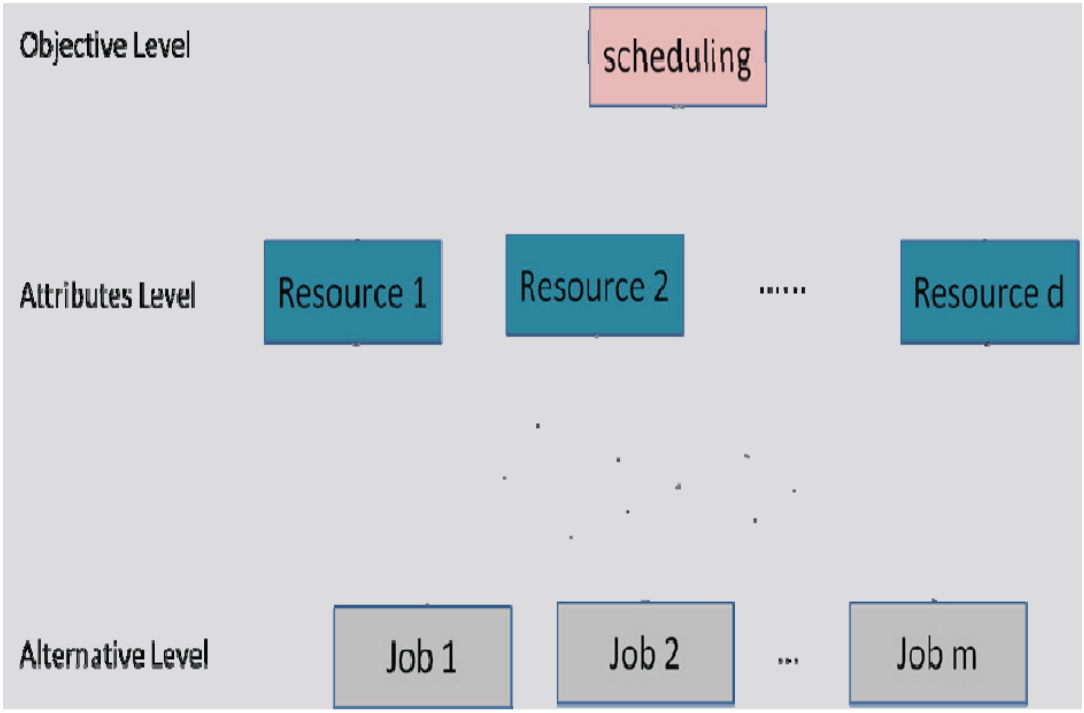


Figure 5: Priority level of jobs and services in cloud computing

The detailed steps of Priority job scheduling are illustrated in **Fig. 6.** The algorithm take a set of jobs J= {j1, j2,.., jm} that request resources in a cloud environment and a set of resources C={c1,c2,…,cd} available in cloud environment as input where (d<<m). Each job requests a resource with a determined priority. The priority of each job is compared with other jobs separately. For example, suppose the ratio of priority of Ji to Jj for gaining a particular resource such as Cg is 7, in this case $Pg^{ij}=7$ and $Pg^{ji}=1/7$;

$$Pg^{ij} = \begin{cases} \frac{1}{Pg^{ji}} & i \neq j \\ 1 & i = j \end{cases}$$

In above Equation Pg denotes a matrix with m rows and m columns known as comparison matrix. And for each resource such comparison matrixes of jobs are created according to priority of resource accessibilities say Q1, Q2 ... Qd. Now for each of comparison matrixes a priority vector (vector of weights) Aw is computed by following equation:

$Aw= \lambda max * \omega$

where λmax is denoted the principal eigenvalue of Matrix A and ω is denoted the corresponding eigenvector.

Suppose W1, W2 ... Wd are corresponding priority vectors of Q1, Q2 ... Qd respectively. And using these priority vectors normal matrix of jobs level are defined as

Δ= [W1, W2 ... Wd]

The next step of the algorithm is to make a d×d comparison matrix for resources based on their priorities. This matrix helps to determine which resource has higher priority than others based on decision maker(s). Say M is comparison matrix for resources level and β will be defined as priority vector of M. Then PVS is calculated which is denoted as priority vector of scheduling jobs. PVS can be calculated by following equation:

PVS= Δ .β

Finally, the maximum element of PVS is selected and then select corresponding job of J in order to allocate a suitable resource and after that list of jobs is updated.

A general algorithm for PJSC

1) Enter $J=\{j_1,...,J_m\}$ a set of jobs
2) Enter $C=\{c_1,...,C_d\}$ a set of resource
3) For all jobs make consistent comparison matrix according to priority of resources accessibilities(d matrix with *m* row and *m* column)
4) Compute priority vector for all d matrixes in step 3 based on Eq. (3).
5) Make a matrix with priority vectors in step 4 based on Eq. (8) and name it Δ .
6) For C compute a consistent comparison matrix according to decision maker(s).
7) Compute priority vector for matrix in step 6 same as step 4 and name it Y .
8) Compute *PVS* which is a vector included value of priority of jobs from Eq. (9).
9) Choose a job with maximum amount of priority value based on *PVS* and allocate it suitable resource.
10) Update the list of jobs.
11) End.

Figure 6: Priority based Job Scheduling Algorithm

Experimental result of this algorithm indicates that the proposed algorithm has reasonable complexity. In addition, deterministic value for finish time (makespan) of PJSC cannot be calculated. It depends on the priory of jobs and may change from the worst value to the best value. Improving the proposed algorithm in order to gain a reasonable and less finish time is considered as future work.

D. SLA-Tree

Yun Chi, Hyun Jin Moon Hakan Hacıgumus, Junichi Tatemura [7] proposes a novel data structure, called SLA-tree to efficiently support profit-oriented decision making.

SLA-tree is built on two pieces of information:

1) A set of buffered queries waiting to be executed, which represents the scheduled events that will happen in the near future.
2) A Service level agreement (SLA) for each query, which indicates the different profits for the query for varying query response times.

By constructing the SLA-Tree, certain profit oriented "what if" questions can be efficiently answered. Answers to these questions in turn can be applied to different profit-oriented decisions in cloud computing such as profit-aware scheduling, dispatching, and capacity planning. In other words by using SLA-Tree, We can schedule the client's requests so that can get maximum profit out of it. The SLA-tree framework is illustrated in Figure 7.

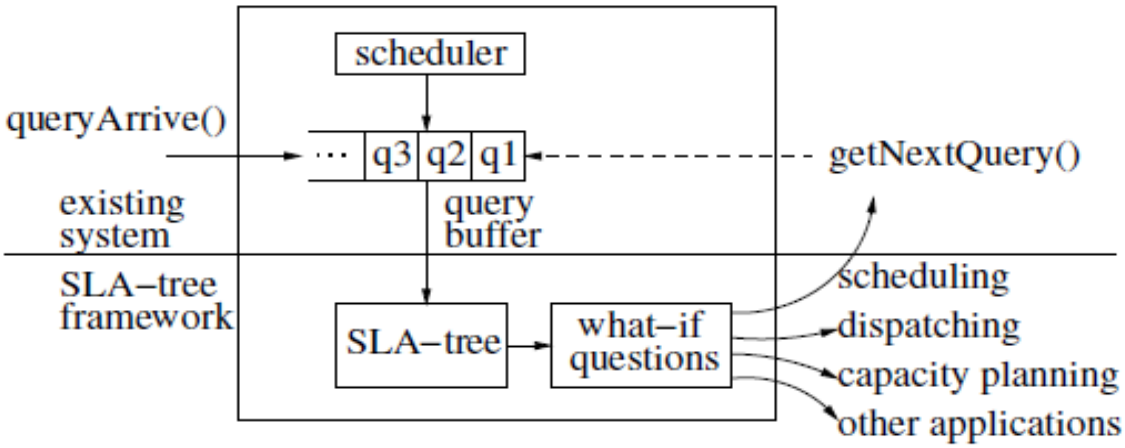


Figure 7: The SLA-tree Framework

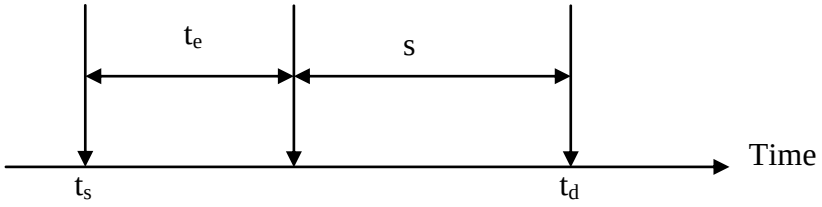
In "SLA-Tree", The SLA metric is used on query response time, which is the time between a query is presented to the system and the time when the query execution is completed.

In SLA-Tree algorithm two key questions are used for scheduling:

- Postpone (m,n,t) : How much profit will be lost if we postpone queries qm, qm+1, . . . , qn by time of t?
- Expedite (m,n,t) : How much profit will be gained if we expedite queries qm, qm+1, . . . , qn by a time of t?

First we generate two types of SLA-Tree called S+ and S- using "Slack Time".
Slack time is calculated as follow: there are three parameters: ts,te and td where

ts is the scheduled starting time for query.
te is the execution time for qi in the database and
td is the deadline for query q. That is, query must be finished before td.
So slack time s = td – ts - te;



In other words, slack si is the time that query can be further postponed (with respect to its originally scheduled starting time) without introducing additional penalties. If the slack is a negative value, we reverse its sign and call the result tardiness, which is again a positive value. In other words, if $s < 0$, then the tardiness -s indicates the time that query has to be expedited (with respect to its originally scheduled starting time) in order to avoid its profit loss.
Positive slack values are used to make S+ tree and negative slack values (tardiness) are used for S- tree.

APPLICATIONS OF SLA-TREE:

1) SLA-Tree for Scheduling

Through SLA-Tree, the profit gain of rushing each job qi to the front of the buffer can be efficiently computed. By expediting qi, we change the total profit in two ways. First, there will be a potential profit gain because qi is executed earlier than it was originally scheduled. Second, there will be a potential profit loss because queries q1 through qi−1 will be postponed by t, the execution time of qi, from their original schedule. That is, the potential profit gain of rushing qi is

$pi = pgain(qi) - ploss(q1, \ldots, qi-1).$

The first part can be easily obtained by looking at qi and the second part is exactly postpone (1, i-1, t), where t is the execution time of qi.

2) SLA-Tree for Dispatching

Profit-aware dispatching policy can be made by using SLA-Tree by asking "what if" question to each server Si like "What will be Si's expected profit change if q is dispatched to Si?"

The profit change here depends on the profit brought by q itself as well as the potential profit loss brought by inserting q to the query buffer of Si, which affects other queries in the buffer. After obtaining the answers to these "what if" questions, a profit-aware dispatcher can then dispatch the query q to the server that results in the maximal profit gain, or drop q if profit impacts for all the servers are negative.

3) SLA-Tree for Capacity Planning

Capacity planning can be done together with dispatching process by introducing a virtual server in computation process which has infinite processing power and queue.

During dispatching process with considering virtual server, Workstation will collect Meta data which provide profit that can be gained by adding another server.

LIMITATIONS:

- SLA-tree Based Scheduling is A Greedy Algorithm
- SLA-tree Requires An Existing Query Execution Order

Both theoretical analysis and experimental studies demonstrated that SLA-tree could efficiently provide valuable and accurate information that can be used by cloud database service providers for profit-oriented decisions such as scheduling, dispatching, and capacity planning.

E. Enhanced Max-min Task Scheduling Algorithm

Upendra Bhoi [8] proposed an Enhanced Max-min task scheduling algorithm that offers an improved task scheduling algorithm based on Max-min. The Max-min algorithm selects the task with the maximum completion time and assigns it to the resource on which achieve minimum execution time while Enhanced Max-min assign task with average execution time (average or Nearest greater than average Task) to resource produces Minimum completion time (Slowest Resource).

There are some situations when largest task has long execution time compared to other tasks in Meta-task, in such case overall makespan is increased because this large task is executed by slowest resource first while other tasks are executed by faster resource. Therefore, instead of selecting largest task if we select Average or nearest greater than average task then overall makespan is reduced and also balance load across resources.

Fig. 8 illustrated the detailed steps of Enhanced Max-min Task Scheduling Algorithm.

Algorithm:

1. For all submitted tasks in Meta-task; T_i
   1.1. For all resources; R_j
      1.1.1. $C_{ij} = E_{ij} + r_j$
2. Find task T_k costs Average or nearest Greater than Average execution time.
3. Assign task T_k to resource R_j which gives minimum completion time (Slowest resource).
4. Remove task T_k from Meta-tasks set.
5. Update r_j for selected R_j.
6. Update C_{ij} for all j.
7. While Meta-task not Empty
   7.1. Find task T_k costs maximum completion time.
   7.2. Assign task T_k to resource R_j which gives minimum execution time (Faster Resource).
   7.3. Remove Task T_k form Meta-tasks set.
   7.4. Update r_j for Selected R_j.
   7.5. Update C_{ij} for all j.

Figure 8: Enhanced Max-min Task Scheduling Algorithm

Algorithm take set of tasks need to be executed as input. Than for all submitted tasks in meta-tasks, Execution and Completion time is calculated. After that task Tk having cost average or nearest greater than average execution time is selected and assigned to resource Rj which gives minimum completion time. Then this task is removed from Meat-tasks set and scheduled time of tasks in resource Rj is updated. Simultaneously completion times of tasks in j are also updated. In next step, until the Meta-task set not get empty, Task Tk having maximum completion time is selected and assigned to resource Rj which gives minimum execution time and then removed from the Meta-task and as above explained, the scheduled time and completion time of task in j is updated. So in Enhanced Max-min, task selection scenario is changed, it is stated as "Select task with Average or nearest greater than average execution time (Average or nearest greater than average task) then assign to be executed by resource with minimum completion time (Slowest resource)".

IV. CONCLUSIONS

In this paper various scheduling algorithms, namely Short Job Scheduling, Job Scheduling Model based on Multi-Objective Genetic Algorithm, Priority based Job Scheduling Algorithm, SLA-Tree and Enhanced Max-min Task Scheduling Algorithm have been studied and analyzed. Based on their own experimental result, it is shown that some of the scheduling algorithms are beneficial to be used in Cloud computing. And there is not a single scheduling algorithm which can solve problem of various types of quality services. Because selection of job scheduling algorithms depends on personal reference and on problem need to be solved.

ACKNOWLEDGMENT

I am thankful to my guide Prof. Tushar Champaneria, Assistant Professor in Department of Computer Engineering for their valuable support. Also I am thankful to my Department for the technical support.

REFERENCES

[1] Mohammad Hamdaqa and Ladan Tahvildari , "Cloud Computing Uncovered: A Research Landscape". Elsevier Press. pp. 41–85. ISBN 0-12-396535-7.
[2] Isam Azawi Mohialdeen, "Comparative Study Of Scheduling Algorithms In Cloud Computing Environment", Journal of Computer Science ISSN 1549-3636
[3] The NIST definition of cloud computing, NIST special publication 800-145
[4] Poonam Devi, "Implementation of Cloud Computing By Using Short Job scheduling", International Journal of Advanced Research in Computer Science and Software Engineering, July – 2013, ISSN: 2277-128X

[5] Jing Liu, Xing-Guo Luo, Xing-Ming Zhang, Fan Zhang and Bai-Nan Li, "Job Scheduling Model for Cloud Computing Based on Multi-Objective Genetic Algorithm" , IJCSI International Journal of Computer Science Issues, January 2013,ISSN (Print): 1694-0784 | ISSN (Online): 1694-0814

[6] Shamsollah Ghanbari, Mohamed Othman "A Priority based Job Scheduling Algorithm in Cloud Computing", International Conference on Advances Science and Contemporary Engineering 2012 (ICASCE 2012)

[7] Yun Chi, Hyun Jin Moon, Hakan Hacıgümüs, Junichi Tatemura, "SLA-Tree: A Framework for Efficiently Supporting SLA-based Decisions in Cloud Computing", EDBT/ICDT'2011 Proceedings of the 14th International Conference on Extending Database Technology, ACM 978-1-4503-0528-0/11/0003

[8] Upendra Bhoi, "Enhanced Max-min Task Scheduling Algorithm in Cloud Computing", International Journal of Application or Innovation in Engineering and Management(IJAIEM), ISSN 2319-4847

