Enhanced SLA-Tree Algorithm to Support Incremental Tree Building

Rohit Gupta[1], Tushar Champaneria[2]

Abstract— **Cloud computing is revolutionize technology which make it possible to deliver computing service over the network. The unique concept of cloud computing allow individual to have business of providing hardware and software service to user from remote location. Job scheduling is an essential requirement in cloud computing environment with the given constraints. Some intensive surveys and researches have been done in the area of job scheduling in cloud computing. The scheduling algorithms schedule the jobs with different perspectives and constrain to achieve efficiency, performance and quality of service. SLA-Tree provides a new data structure based on profit gain through provisioning of services, to efficiently support profit-oriented decision making. But it requires a collection of existing queries to build SLA-Tree. An enhanced version of SLA-Tree is proposed based on incremental tree building approach which doesn't require collection of existing queries and update existing SLA-Tree as query arrives.**

Index Terms—**Cloud computing, Job Scheduling, SLA-Tree.**

I. INTRODUCTION

Extensive uses of cloud services introduce new opportunities and challenges in computing area. Various cloud service providers provide software and hardware services over the network with possibly less cost and high performance. They may have to serve much more diverse clients than other traditional network services. While serving various types of clients, Service providers have main consideration to optimize their profits. Generally, Profit gained by service provider is decided by Service Level Agreements.

Fig 1. shows the general cloud framework based on service provisioning which consists of three layers, namely, the cloud service provider, the internet and the connected clients. Clients use the required cloud services provided by cloud service provider over the internet based on "Pay as you go" method. Service provisioning is carried out as per Service Level Agreement between client and service provider. So it's a major challenge for a service provider to make intuitive profit oriented decision to be successful and gain maximum profit.

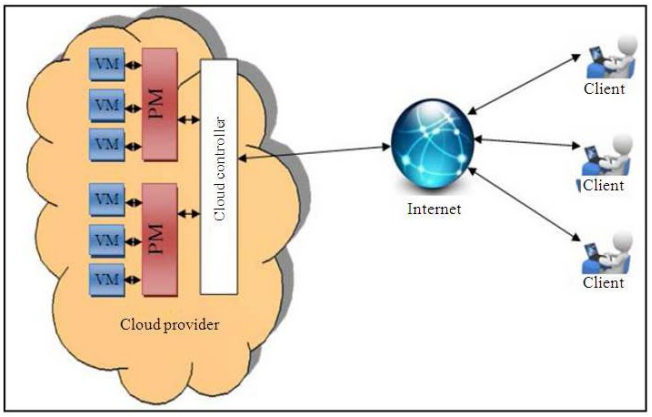

Fig 1. The General Cloud framework [11]

II. RELATED WORKS

2.1 LITERATURE SURVEY

Poonam Devi "Short Job scheduling" [3], proposed a job scheduling algorithm capable to handle the over load condition by the load distribution scheme. Some priority is assigned to each cloud resource to perform the efficient allocation.

Jing Liu, Xing-Guo Luo, Xing-Ming Zhang, Fan Zhang and Bai-Nan Li "Job Scheduling Model based on Multi-Objective Genetic Algorithm" [4], presented a scheduling model based on MO-GA algorithm in cloud environment to maximize the profit and minimize energy consumption of services under the constraint of deadlines. This scheduling model provides a dynamic job selection mechanism of the most suitable scheduling scheme for users based on real-time requirements.

Upendra Bhoi "Enhanced Max-min task scheduling algorithm" [6], offers an improved task scheduling algorithm based on Max-min. Unlike Max-min algorithm, in which the task with the maximum completion time has been selected and assigned to the resource on which it achieve minimum execution time, Enhanced Max-min select the task with average execution time (average or Nearest greater than average Task) and assign it to resource produces Minimum completion time (Slowest Resource). In situations when the largest task has long execution time compared to other tasks in Meta-task, other tasks are executed by faster resource while the large task gets executed by slowest resource first. It increases the overall makespan. Therefore if average or nearest greater is selected in place of largest task than

ISSN: 2278 – 1323

average task then overall it reduce overall makespan and balance load across resources too.

2.2 EXISITING WORK

Yun Chi, Hyun Jin Moon Hakan Hacıgumus, Junichi Tatemura [5] propose 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 Fig 2. In "SLA-Tree", The SLA metric is used on query response time, which is the time between query is presented to the system and the time when the query execution is completed.

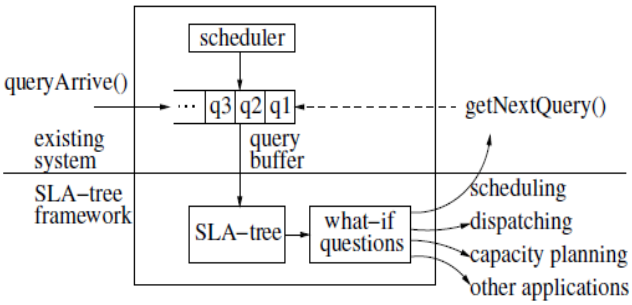

Fig 2. The SLA-tree Framework

In SLA-Tree algorithm two key questions are used in 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". Bottom-Up Complete Binary Tree structure is used to build S+ and S- Tree.
Slack time is calculated as follow: there are three parameters: t_s,t_e and t_d where

t_s is the scheduled starting time for query.

t_e is the execution time for qi in the database and

t_d is the deadline for query q. That is, query must be finished before td.

So slack time $s = t_d - t_s - t_e$;

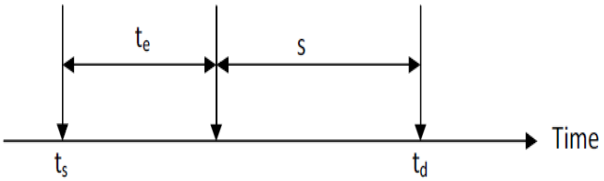

Fig 3. Slack Time

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.

Following are some application of SLA-Tree:

1. SLA-Tree for Scheduling
2. SLA-Tree for Dispatching
3. SLA-Tree for Capacity Planning

Following are some issues related to implementation and use of existing SLA-Tree:

1) Number of queries required to build SLA-Tree must be in power of 2.
2) If new queries arrive, SLA-Tree must be rebuilt to add these new queries in SLA-Tree.
3) How to handle queries which have same slack values?

III. PROPOSED SCHEMA

As explained, SLA-Tree provide a well defined and well structured framework, based on it a profit oriented job scheduling decision can be made. But requirement of collection of Query with Execution Order, specifically in power of 2 in exiting scheme is a big limitation of SLA-Tree which bounds its usage in cloud computing environment where the jobs not come in collections.

To tackle these limitations, an incremental tree construction scheme is proposed which is an enhancement in the existing SLA-Tree framework.

Unlike Existing SLA-Tree algorithm which uses Bottom-Up Complete Binary Tree structure, Incremental SLA-Tree uses Top-Down AVL Tree structure. So that instead of using a collection of scheduled jobs to build an SLA-Tree each time, Incremental approach build the SLA-Tree based on available jobs and update the tree as new job or query arrives. Though the querying time of SLA-Tree O(log(NK)) remain unchanged, This new scheme significantly reduce the space complexity and may reduce

time complexity of building the SLA-Tree in long run.

It also require to change the nodes structure in SLA-Tree because the propose scheme require the intermediate nodes to be scheduled jobs which is not in the existing SLA-Tree framework.

Algorithm:

Step 1: Get the new query and schedule it using some job scheduling algorithms.

Step 2: Calculate the slack value of the new query.

Step 3: Attach the new query based on its slack value in existing SLA-Tree.

Step 4: Update the required intermediate nodes dependent on the new queries.

Step 5: Perform rotation operation over the tree to balance it if required.

Take example data of following queries with certain ids and slack values as follows:

Table I. Query Example

Query ID	Slack value
7	90
6	60
9	60
8	130
3	40
4	70
1	110
2	140
5	80

Fig 4. illustrates the structure and information stored in the incremental version of SLA-tree which is built based on given example data. For simplicity we take K=1 where K is the level of SLA used in SLA-Tree.

In this example, we assume there are 8 queries which have positive slack values. In the figure, for each node n in the slack tree, its slack value is dT which given by a number at centre of the circle. List of descendant nodes is shown at left of the node and the list of node ids below the node is query ids which have same slack values as of node. For example, look at node with slack values 60: it has list of three descendants with ids 3, 4 and 5. And has list of queries id with slack values 60 are 6 and 9, respectively.

The following structure of Incremental SLA-Tree handles the mentioned 3 issues of existing SLA-tree in following manner:

1) There can be any number of queries in new Incremental SLA-Tree.
2) There are two lists of node ids instead of one.
   i. List of descendant node ids
   ii. List of query ids with same slack values
3) If a new query arrives, it can easily be added to existing tree with small number of node updating. So that it doesn't need to create whole tree from scratch.

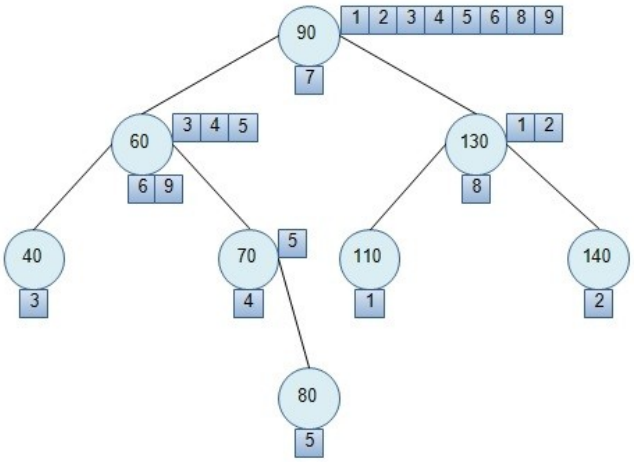

Fig 4. The simple version of a slack tree in the Incremental SLA-Tree, with 1/0 profit model

In existing SLA-Tree, the tree size is 2M-1 where M is number of queries. So if the existing SLA-Tree has been used for mentioned example (by neglecting duplicate slack values), the tree size will be 15. But here as presented in Fig 4, Tree size of incremental SLA-Tree is 8 which is less than number of queries due to equal slack values. It provides a clear view of reduced space complexity of incremental SLA-Tree as compared to existing SLA-Tree.

Now have a look at how Incremental SLA-Tree is updated when new query arrives. Suppose a new query arrives with id: 11 and slack value: 170. Fig 5. Illustrates the new SLA-Tree after adding the new query and highlight the places where update had been performed.

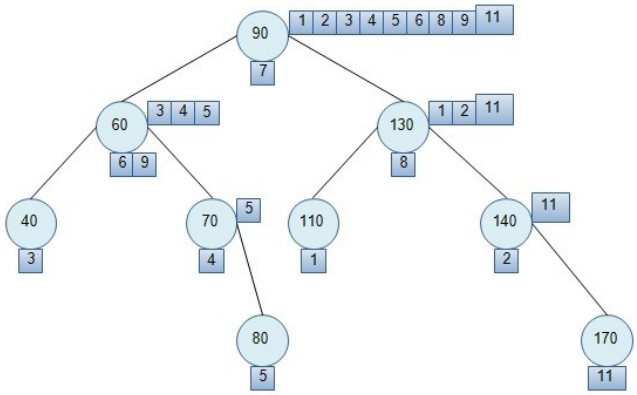

Fig 5. Updated SLA-Tree after adding the new query

As illustrated in Fig 5, new query has been added as new node in tree based on its slack value as right child of node "140". And it require only list of three nodes to be updated to add this new query.

In the same way, a query when removed from execution

query list can also be easily removed from Incremental SLA-Tree with a few numbers of modifications which is not supported by existing SLA-Tree.

There are some situation arise during update of Incremental SLA-Tree when the tree become "imbalance". So certain balancing steps are required to rebalance the Incremental SLA-Tree. It increase the overall time complexity of building the Incremental SLA-Tree.

IV. Result Analysis

We have implemented the existing and proposed algorithm in Cloudsim simulator and following are the implementation output, analysis data and plotted result of existing and new (proposed) SLA-Tree algorithm.

4.1 Space Complexity

Table II. Implementation result based on Space complexity

No. of Queries	Existing Algorithm	New Algorithm
1	0	0
2	2	1
4	8	4
8	24	13
16	64	39
32	160	108
64	384	273
128	896	668
256	2048	1591
512	4608	3695
1024	10240	8395
2048	22528	18962
4096	49152	42020
8192	106496	92078
16384	229376	203322

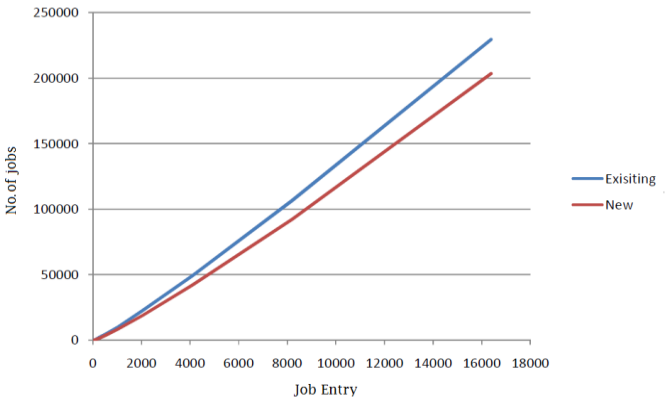

Fig 6. Graph of implementation result (Space Complexity)

As illustrated in Fig 6. Implementation result, Number of query entries required to store in Incremental SLA-Tree algorithm is much lower than the existing SLA-Tree algorithm. And the difference between the entries of queries in existing and proposed algorithm increase exponentially as number of queries increase.

4.2 Time Complexity

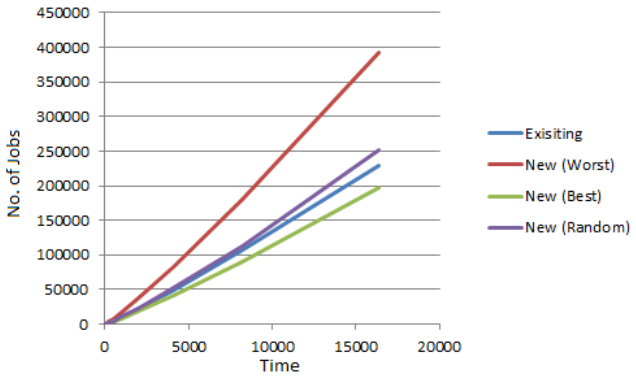

Fig 7. Graph of implementation result (Time Complexity)

As illustrated in Fig 7. Implementation result, Time required to add new node in Incremental SLA-Tree algorithm is better than the existing SLA-Tree algorithm in best case. In average (random) case, both algorithms have almost same time complexity. In worst case, Incremental SLA-Tree requires high number of balancing while adding new nodes so that it's time complexity is much higher than existing SLA-Tree algorithm.

V. Conclusion and Future Work

Both theoretical analysis and experimental studies demonstrated that SLA-tree could efficiently provide valuable and accurate information that can be used by cloud service providers for profit-oriented decisions such as scheduling, dispatching and capacity planning. And this new schema will further enhance the speed and capability of SLA-Tree framework. It resolves the issues of existing algorithm simultaneously as explained.

As a future work, some update is required in proposed structure of Incremental SLA-Tree to reduce the balancing operation in tree in worst case and support multilevel of SLA.

REFERENCES

[1] Gordon Haff, Introduction To Cloud Computing, Red Hat Cloud.
[2] The NIST definition of cloud computing, NIST special publication 800-145
[3] 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
[4] 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
[5] 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

[6] 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
[7] "Cloud Computing For Dummies", Author:Judith Hurwitz, Robin Bloor, Marcia Kaufman, Fern Halper
[8] Bhushan Lal Sahu, Rajesh Tiwari, "A Comprehensive Study on Cloud Computing", IJARCSSE, Sep-2010
[9] Mohsin Nazir,"Cloud Computing: Overview & Current Research Challenges",IOSR JCE, Dec-2012
[10] Rohit O. Gupta, Tushar Champaneria, "A Survey of Proposed Job Scheduling Algorithms in Cloud Computing Environment", International Journal of Advanced Research in Computer Science and Software Engineering, November-2013, ISSN: 2277-128X
[11] Isam Azawi Mohialdeen, "Comparative Study Of Scheduling Algorithms In Cloud Computing Environment",Journal of Computer Science ISSN 1549-3636

AUTHORS REFERENCES

[1]**Rohit Gupta** received degree of B.E. Computer Engineering from Kalol Institute of Technology in 2012. Now pursuing M.E. Computer Science and Engineering in L.D. College of Engineering, Ahmedabad.

[2] **Tushar Champaneria** is currently working as an Assistant professor at Computer Engineering Department, L. D. College of Engineering, Ahmedabad.

