IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 2 Issue 4, April 2015.
www.ijiset.com
ISSN 2348 – 7968

Weight based Budget Distribution (WBD) , a budget constrained scheduling algorithm for workflows in cloud

Mr. Bhushan M. Bhalerao, Mr. Shailendra W. Shende

Dept. of Information Technology, Yeshwantrao Chavan College of engineering, Nagpur, India.
Dept. of Information Technology, Yeshwantrao Chavan college of engineering , Nagpur, India.

Abstract— One of the advantage of Cloud Computing is its "Pay-as-you go" model where one of its service, Software as a Service (SaaS), is delivering access of applications to the end users over the Internet without prior investment in infrastructure and software. To serve, the customers are provided by resources of internal data centers or rent resources from a public Infrastructure as a Service (IaaS) provider. Almost every Cloud Service provider (CSP) is imposing same cost and same deadline for every customer. But in fact not every customer is always interested in same Quality of Service (QoS) as they may have budget problem or may they wants work to be done earlier than usual. To overcome these limitations, we propose Weight based Budget Distribution (WBD) budget constrained scheduling algorithm for SaaS providers to effectively utilize Cloud resources to avail service within customer's budget with reducing execution time of service. We used approximately same approach to use deadline as constrained. Furthermore, we conduct an extensive evaluation study to analyze which solution suits best in which scenario to schedule workflow of application within given budget. Simulations results show that our proposed algorithms provide substantial improvement over reference ones across all ranges of variation in QoS parameters.

Keywords—Cloud Computing; Workflow scheduling; Budget constrained; Deadline constrained; Greedy algorithm.

1. INTRODUCTION

Nowadays, Cloud computing is becoming current popular computing paradigms. Over the Internet, it is allocating computing services to users. Resources of cloud computing is often manages by third parties that could be either hardware or software. At any time, users of cloud computing can access cloud services. It is available for all 24 hours in a day. Cloud computing helps user to reduce their cost in purchasing, operating, and maintaining physical computing devices. There are many high performance computing physical machines, which are then configured to allow numbers of virtual machines (VMs) to enhance flexible sharing and higher utilization of physical resources. Key technology to achieve such in cloud computing is called Virtualization. Because of this virtualization cost of having many physical machines reduces.

Basically Cloud computing delivers its services in following three forms: 1] Software as a Service (SaaS), where the consumer simply uses an application and he does not need to control the host environment. It allows the number of common users to use software remotely. Salesforce.com and Google Apps are known examples of this model. It is the advantage of user to not install the software on their service 2] Platform as Service (PaaS), where cloud provides hosting environment to consumer for their applications. Amazon Web Services and The Google App Engine are known PaaS examples. Here the platform is actually an application framework. 3] Infrastructure as a Service (IaaS), where the consumer have been delivered computing resources e.g. processing power and storage in the form of virtual machines(VM) which is deployed on physical machine. Here consumers are allowed to deploy their application as well as are allowed to control the environment. Eucalyptus, Globus Nimbus and Amazon Elastic Compute Cloud are well known IaaS providers [7].

Many complex applications of scientific domain, e-science and e-business can be modeled as workflows. A well-known way to represent workflow application is the Directed Acyclic Graph (DAG), where nodes are individual application tasks, while the directed edges between the tasks represent inter-task data dependencies. Workflow scheduling is a well known NP-complete problem; many heuristic and meta-heuristics methods have been introduced for distributed systems [9]. In a cloud computing, pricing is dependent mainly on Quality of Service (QoS) offered. IaaS provider charges higher prices for higher QoS. It is seen that users are not always keen to complete workflows earlier than they require. Instead, they wish to save price and agreed for reduced QoS.

The requirements of workflows of scientific applications are usually fulfilled by IaaS clouds. In a IaaS cloud user have aim to minimize expenses while meeting their performance requirements. The economical success of application user achieves only when SaaS provider will

ISSN 2348 – 7968

forecast the accurate resources needed to run application which is under the user given budget. Thus, workflow scheduling is becoming a hot topic. It deals with the allocation of tasks to suitable resources while satisfying user's QoS requirements. Here we are concern about budget and deadline of workflow as a QoS requirements.

In this paper, we propose Weight based Budget Distribution Greedy workflow scheduling algorithm for cloud environment. Our objective is to schedule workflow such that it lessen the execution time while meeting the price constraints and execute the workflow within constrained deadline but with reduced price of application, given by the user.

In section 2 and 3 we have explained about workflows in cloud and our system model. Then in section 4 we made some important literature survey. Section 5 covers Our proposed algorithm. Subsequently section 6 present Performance evaluations.

2. WORKFLOWS IN CLOUD

The DAG composed of an individual task represented by a node and its dependencies are represented by its edges as shown in fig 1 [10]. A dependency guarantees that a child node cannot be executed till its entire parent tasks finish successfully and transfer the input data required by the child. A DAG can be modeled by a tuple G (N,E), where N is the set of n nodes, each node $n_i \in N$ represents an application task, and E is the set of communication edges between tasks. Each edge e(i, j) ∈ E represents a task-dependency constraint such that task ni should complete its execution before task nj can start.

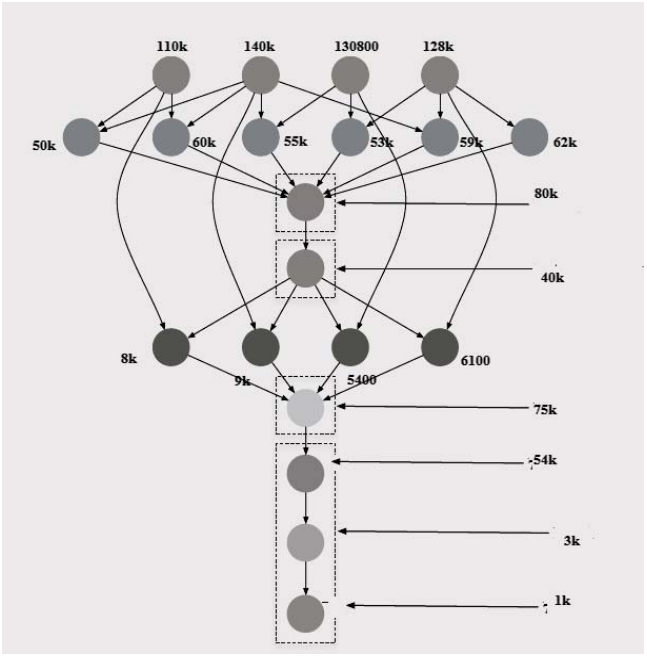

Fig 1: Montage Workflow

3. SYSTEM MODEL

We have shown a system model in fig 2 [1].In system model, we set up a model of SaaS provider, which consists of users, SaaS providers, and IaaS providers. Users are nothing but the end users of software, while SaaS provider are responsible for creation of application and availing it to end users while IaaS have responsibilities of executing the application on VM they have. Users will request for the application at SaaS provider using internet by submitting their QoS requirements. SaaS uses admission control to interpret and analyze the user's QoS requirements and decides whether to accept the request based on the availability, capability and cost of VMs. Then, the scheduling component is in charge for allocating resources based on admission control decision. Following are the actors involved in the model.

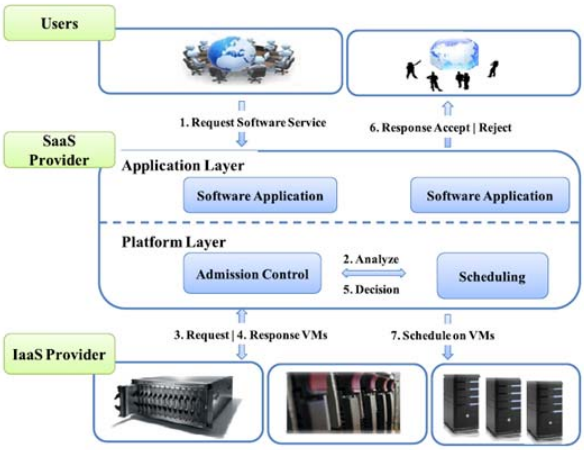

Fig 2: A high level system model for application service scalability using multiple IaaS providers in Cloud

1] *User* who request for accessing application to a SaaS provider with QoS constraints like budget or deadline. Then, the SaaS layer makes use of admission control and scheduling algorithms to admit or not admit this request. If the request gets accepted, a formal Service Layer Agreement (SLA) is confirmed between both parties to ensure the QoS requirements such as budget constrained.

2] *SaaS provider* rents required resources from IaaS providers and leases software as services to users. Their aim is to minimize operational cost by using resources from IaaS providers, and improving the execution time but with satisfying SLA.

3] An *IaaS provider* makes resources available in form of VMs to SaaS providers and is responsible for dispatching VM images to run on their physical resources. It is important for resource provider to maintain SLA because it guarantees service quality [1].

ISSN 2348 – 7968

4. LITERATURE REVIEW

We perform a survey of related work on workflow scheduling over cloud environment under budget constrained [2-6], and believe that greedy approach would be more beneficial to solve the 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.

Amandeep and Sakshi [11] propose Deadline and Budget distribution-based Cost-Time Optimization (DBD-CTO) workflow scheduling algorithm that minimizes execution cost while meeting timeframe for delivering results using a greedy approach.

But we have special interest in work done by Yang Wang [8]. He proposed a greedy approach to select candidate machine from available set of candidate machines. MapReduce framework was his targeted application. He proposed an idea of in-stage greedy algorithm and global greedy algorithm to optimize the makespan of workflow in given budget constrained. We are also proposing an idea of greedy for in stage as well as for global optimization.

5. PROPOSED ALGORITHM

5.1 *Cost or Budget constrained greedy algorithm*

It is studied that initially many of heuristics algorithm always distribute whole budget into all stages or at all tasks in a stage equally, and after successful iterations budget gets distributed. We believe that instead of distributing budget equally, we can distribute the budget according to their weights. After obtaining budget a task will search for available candidate machine within a set of candidate machines under given budget constrained and return its remaining budget to the task next to it. Every stage will also do same thing like nodes.

For given stage (i) budget will be distributed according to equation 1and for a task (j) of stage (i) budget will be distributed according to equation 2. Here weight is termed as no. of instruction node has to be executed in cloud

Budget for stage (i) = (Number of tasks in stage (i))/(all Task of workflow) * User given Budget ---------- [1]

Budget for node (j) = (Weight of node(j))/(weight of all nodes in stage (i)) * Budget for stage (i) ---------- [2]

Following are the steps of algorithm:
Step 1: Distribute budget into all given stages of workflow as per equation 1.
Step 2 Distribute budget of a stage into all its nodes as per equation 2.
Step 3 Highest amount obtained node will first search for available VM. If VM of price below constrained amount is available algorithm will work continue otherwise it will ask user to increase the budget.
Step 4: If obtained VM is under given budget then it will search for highest price available machine and few of amount is remaining then it will pass to the next higher amount obtained machine.
Step 5: Continue process 4 until all tasks of a current stage gets free available VM for execution. At the last if few of amount is still remaining will be pass to the next Stage.
Step 6: process 5 will continue until all the stages of workflow is not computed.

5.2 *Deadline constrained greedy algorithm*

In time constrained we are using same approach we used for budget constrained. We are dividing the whole deadline of application, given by user according to weight as in eq. 3 into all stages of workflow. But here we are not dividing the deadline given to stage into all its tasks; instead we are arranging the task according to their rank given on basis of weight the posses.

Deadline for Stage (i) = (Number of tasks in stage (i))/(all Task of workflow)* User given Deadline ---------- [3]

Following are the steps of algorithm:
Step 1: Distribute deadline into all given stages of workflow as per equation.
Step 2: Arrange all tasks in descending order of their weights, and give them rank from 1 as highest to onwards.
Step 3: Lowest task will be first assign lowest budget available machine, and then procedure will continue up to highest rank task.
Step 4: Estimate the execution of all tasks is done within obtained deadline. And if it is not then assign tasks to next higher budget available machine.
Step 5: Repeat step 4 until estimated time not come under obtained deadline for given stage and in the case some of time is left forward it to next stage.
Step 6: After step 5 it is still found that workflow is not executing within deadline, ask user to raise deadline.
Step 7: Repeat the procedure from step 2 to step 6 until all tasks not get assigned within deadline imposed by user.

6. PERFORMANCE EVALUATION

In this section, we will find performance evaluation results, which includes comparison with reference algorithms [8] & [11] to our proposed algorithms. We are considering Montage workflow [10] for our experimental evaluation; here every node is weighted as a number of instructions it has to execute in cloud as shown in fig 1.

ISSN 2348 – 7968

5.1. Experimental methodology

We refer CloudSim [12] and make our own simulation program as a Cloud environment simulator and implement our algorithms within this environment. As our aim is to prove substantiality of our algorithm, we are assuming that the set of candidate machine is nothing but the set of all available machines at the time of user request, and we are not considering the network or data transfer charges as it is same for almost all algorithms.

5.2 Impact of Number of VM's

We first evaluated the impact of the number of VM's as search space on the total scheduling length of the workflow with respect to different budget constraints. To this end at first instant, we fix 10 VM's to be available free as a searching space, and then search space is raised to 20 no. of VM's. The results of proposed algorithm for taken workflow are shown in Fig. 3. It is found that with the no. of VM's increasing, the scheduling lengths are decreased.

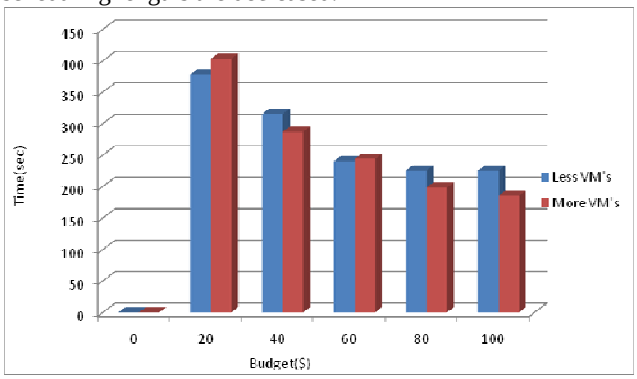

Fig. 3: Impact of Number of VM's

5.3 Comparison between Algorithms's:

From the fig 4, it is clearly reveal that WBD budget constrained algorithm starts the execution at even lower budget. It also outperforms others with respect to execution time at various budgets.

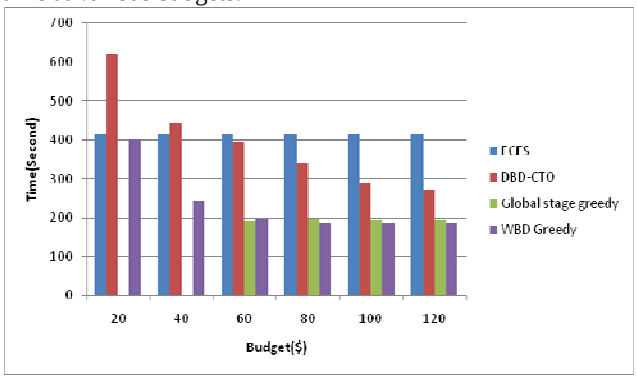

Fig 4: Comparison between Algorithm's under budget constrained

5.4 Impact of deadline constrained

We perform the experiments under time constrained and obtained the following results shown in fig 5. As we will increase constrained of deadline, total cost for workflow will decrease linearly. And hence we achieved our objective of time constrained.

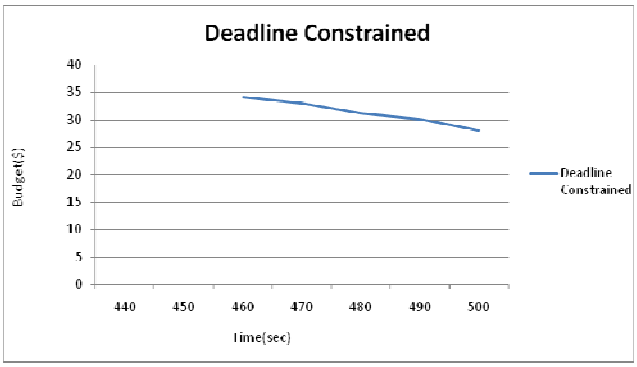

Fig 5: Time vs Budget (under Deadline constrained)

7. CONCLUSION AND FUTURE SCOPE

In this paper, we studied constraints of budget and deadline for the scheduling of a workflow in the Cloud. We designed an algorithm which is distributing budget according to weight of node and also forwarding remaining budget to next node which ultimately gives it numbers of advantages like Execution in even lower budget, reduce workflow makespan. We also successfully designed deadline constrained algorithm which will reduce total cost of workflow execution as cost constrained is increases. Admittedly, our proposed for the budget-constrained scheduling of workflows is comparatively simple, which may not fully be a sign of some advanced features in reality such as redundant computing for fault tolerance, dynamic pricing and so on. Though in any case it makes a reasonable use case as a baseline to reveal how cost-effective scheduling of the workflows could be available in Cloud computing. Validating this model in real cloud and scheduling with time constrained is our future work.

References

1] S.K. Garg, R. Buyya, H.J. Siegel, Time And Cost Trade-Off Management For Scheduling Parallel Applications On Utility Grids, Future Gener. Comput.Syst. 26 (8) (2009) 1344–1355.

2] Bhushan Bhalerao and Shailendra Shende, "A Review on Different Scheduling Algorithms for Workflows in Cloud environment" *in International Journal on Recent and Innovation Trends in Computing and Communication*, 2015, Volume: 3 Issue: 2.

3] Hamid Mohammadi Fard and et al., "Budget-Constrained Resource Provisioning for Scientific Applications in Clouds"

ISSN 2348 – 7968

In *IEEE International Conference on Cloud Computing Technology and Science*, 2013.
4] Xiangyu Lin, Chase Qishi Wu, "On Scientific Workflow Scheduling in Clouds under Budget Constraint" 42nd International Conference on Parallel Processing, 2013.
5] E Chase Qishi Wu and et.al, "End to-end Delay Minimization for Scientific Workflows in Clouds under Budget Constraint" *IEEE Transactions on Cloud Computing*, 2014.
6] Gunho Lee, "Resource Allocation and Scheduling in Heterogeneous Cloud Environments" Electrical Engineering and Computer Sciences University of California at Berkeley, UCB/EECS-2012-78, May 10, 2012.
7] L Bittencourt and E Mauro, "HCOC: A Cost Optimization Algorithm for Workflow Scheduling in Hybrid Clouds" University of Campinas.
8] Yang Wang and Wei Shi, "Budget-Driven Scheduling Algorithms for Batches of MapReduce Jobs" *IEEE transactions on cloud computing*, 2014.
9] Yu, J., and Buyya, R. "Workflow Scheduling Algorithms for Grid Computing", In: Xhafa F, Abraham A (eds) Metaheuristics for scheduling in distributed computing environments. ISBN: 978-3-540-69260-7. Springer, Berlin.
10] Shishir Bharathi and et al., "Characterization of Scientific Workflows" In USC Information Sciences Institute Marina del Rey, CA.
11] Amandeep Verma and Sakshi Kaushal. "Deadline and Budget Distribution based Cost- Time Optimization Workflow Scheduling Algorithm for Cloud." *In International Conference on Recent Advances and Future Trends in Information Technology*, 2012.
12] R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A.F. De Rose, R. Buyya, CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Software: Practice and Experience (ISSN 0038-0644) 1 (41) (2011) 23–50, Wiley Press, New York, USA.