Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 46 (2015) 660 – 667

Procedia Computer Science

International Conference on Information and Communication Technologies (ICICT 2014)

Customer Facilitated Cost-based Scheduling (CFCSC) in Cloud

D.I. George Amalarethinam[a*], T. Lucia Agnes Beena[b]

[a]*Associate Professor & Director – MCA, Jamal Mohamed College, Trichy-620020, Tamil Nadu, India.*
[b]*Assistant Professor in Information Technology, St. Joseph's College, Trichy,620002, Tamil Nadu, India*

Abstract

Cloud computing is a prototype which takes the new form of utility computing. The on-demand flexible service and pay-per-use schemes provided by the Cloud providers attract the customers to move towards Cloud computing environment. Both the service providers and the customers will receive the economic benefits only when the available resources are correctly scheduled. Due to commercialization, the cloud environment emphasizes the need for the development of new algorithms for better economic factors. This is done by the task and data scheduler in the workflow systems. In this paper, an algorithm CustomerFacilitated Cost-based Scheduling (CFCSC) algorithmis proposed to favor the Cloud customers with economic cost. The CFCSC algorithm outstrips the HEFT in terms of load balance, minimum cost and less complex cost function.Also, both CFCSC and HEFT algorithms have the same makespan.

Peer-review under responsibility of organizing committee of the International Conference on Information and Communication Technologies (ICICT 2014)

*Keywords:*Task scheduling; Directed Acyclic Graph; HEFT; Cloud computing; Makespan; Monetary cost

1. Introduction

Cloud computing, the new form of Utility computing gains its popularity in the last few years. Cloud Infrastructure as a Service (IaaS) enables the user to use services in a flexible environment. IaaS provide the illusion of unlimited resource allocation for most users[1].

*D. I. George Amalarethinam; M-+919443179535; di_george@ymail.com;

Peer-review under responsibility of organizing committee of the International Conference on Information and Communication Technologies (ICICT 2014)
doi:10.1016/j.procs.2015.02.119

The construction and operation of a medium sized datacenter is a time consuming process and depends on the economic strength of the company which is interested to build the datacenter. Cloud computing offers services with minimum cost compared to the setting up of the datacenter. Cloud Service Providers (CSPs) permits the users to select the machine hours based on their requirementregardless of the costs without paying a premium for large scale. This resource elasticity is unique in the history of Information Technology2.

The popular cloud service providers are Amazon EC24 and Google Computed Engine5. They have different schemes which vary in pricing as well as in computing capacities. For example, Amazon EC2 has micro, small, medium, large and extra large On-Demand instances prices. While Google Compute Engine charges for usage on a monthly basis depending on the type of machine needed. The users of the Cloud services will be content, if the service is affordable to their budget. Once the user submits the collection of tasks (Application) to the cloud environment, it will take care of scheduling and monitors the execution of the tasks. The important issue here is the scheduling (matching tasks to the resources). While preserving the logical sequence of the task, the scheduler must minimize the total execution time of all the tasks and the monetary cost. This can be achieved only with the aid of the efficient scheduling algorithms.

The user application is commonly represented by Directed Acyclic Graph (DAG). DAG structures have the ability to simulate the real life scenarios. DAG scheduling is a NP-complete problem6. Heuristic methods and Approximation algorithms are used to solve these kinds of problems to get optimal results. Hence, heuristic based scheduling algorithm can be proposed to achieve optimal solution for DAG structure. One such initiative is the proposed CFCSC algorithm. This CFCSC algorithm concentrates on load balance and monetary cost to improve the quality of scheduling.

2. Related work and Motivation

Workflows are used to represent a variety of scientific and engineering applications which involve high processing and storage. To satisfy these applications, the Cloud Computing Environment has emerged as a new paradigm. Workflows for these applications are represented by the Directed Acyclic Graph (DAG) model. Literature reveals lots of research work has been conducted to solve workflow scheduling. They mostly focus on minimizing the scheduling length. In Cloud Computing Environment, economic cost in scheduling plays a vital role. Existing research on scheduling algorithms concentrates on the economic cost factors. ScaleStar algorithm proposed by Lingfanget al[3]balances the makespan and monetary cost by allotting the appropriate task to a appropriate virtual machine which meets the user budget. Bittencourt[7] proposed a HCOC algorithm in Hybrid Clouds. HCOCchoosesthe resources that are to be leased from the public cloud and the private cloud. The combined resources are used to provide the needed processing power to execute a workflow within a given execution time. HCOC reduced the monetary cost with the established desired execution time. Workflow scheduling algorithm proposed by Mengxia[9], minimizes the cloud overhead within a user-specified execution time bound. Ranjitet al[10]designed a score based deadline constrained workflow scheduling algorithm that executes workflow within manageable cost while meeting user defined deadline constraint.

Vermaet al[11] proposed a Deadline and Budget distribution-based Cost-Time Optimization (DBD-CTO) workflow scheduling algorithm. This algorithm minimizes the computation cost as well asdelivers the results at the appropriate time.The compromised-time-cost scheduling algorithm[12],favors the cloud computing to accommodate instance-intensive cost-constrained workflows at the expense of the execution time and cost. This algorithm meets the user designed deadline with reduced mean execution time.The Improved cost-based scheduling algorithm by Selvaraniet al[13] focuses both on resource cost and computation performance. It groups the user tasks that belong to a particular cloud resources' capacity to reduce the computation/communication ratio and assigns the grouped tasks to the resources.

Li et al[8] designed a cost-conscious scheduling algorithm (CCSH) for the cloud environment with a goal of minimizing both the makespan and monetary cost. CCSH is able to achieve minimum cost at a reasonable increase in the makespan but the cost function is complex. This paper is analogous to these studies with the aim of minimizing the monetary cost and to balance the load. The proposed CFCSC algorithm is a simple algorithm with a less complex cost function. The CFCSC reduces the monetary cost with the same makespan as HEFT. The CFCSC outperforms the popular HEFT algorithm in load balancing and monetary cost.

3. DAG Model

Efficient scheduling of user applications submitted to the cloud environment is critical due to its heterogeneity, precedence constraints and communication between the tasks. An application is usually denoted by the DAG model[14]. It includes the execution time of tasks, data to be communicated between the tasks and the task dependencies of an application.

The formal way of representing a DAG is G = (V, E). Here V denotes the set of n tasks and E denotes the set of weighted directed edges. A node is non-preemptive task that is to be executed sequentially on a same processor. The weight associated with the node denotes the computation time needed for the completion of the particular task. The weight associated with the edge denotes the data to be transferred between the nodes and their precedence relation. Thus, e(i, j) ∈points the direction and the volume of data sent between nodes $v_i$andv_j. Thus v_i is called the parent node and $v_j$is called the child node. A node is called the entry node, if it doesn't have a parent. A node is called an exit node, if it doesn't have a child. Figure.1 shows an example of a Task Graph.

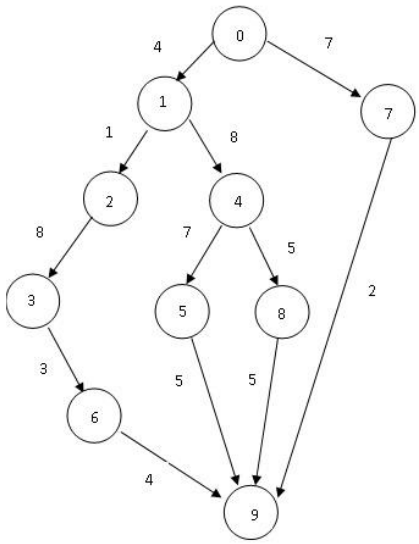

Fig. 1 Sample Task Graph

The matrix w has v rows and vmcolumns denoting the computation cost of tasks. The v rows represent the computation cost of a particular task in various virtual machines. vm denotes the number of virtual machines used in the system. Each w(i, j) gives the estimated execution time to complete task v_i on processor vm_j.The average execution costs of tasks are applied in the task priority equations. The average execution time of a node v_i is given by

$$W_i' = \frac{\sum_j^{vm} \boldsymbol{w(i,j)}}{vm} \quad (1)$$

The communication cost of the edge e(i, j), which is for data transfer from task v_i (scheduled on vm_k) to task v_j (scheduled on vm_l), is denoted by c(i, j). When both v_i and v_j are scheduled on the same processor, $vm_k = vm_l$, then c(i, j) becomes zero, since the intra-processor communication cost is negligible compared with the inter processor communication cost. The proposed algorithm follows the upward rank to prioritize the task. The upward rank of the task $v_i$is recursively defined by

$$Ranku\ (v_i) = w_i' + max_{vi \in succ\ (vj)}(\ c(i,j)\ +\ Ranku\ (vj)) \quad (2)$$

wheresucc(v_i) is the set of immediate successors of task v_i. Since it is calculated recursively beginning from the exit node till the entry node, it is called as an upward rank. The EST(v_i,vm_j) and EFT(v_i,vm_j) are the earliest execution start time (EST) and the earliest execution finish time (EFT) of node v_i on processor vm_j, respectively. They are defined by

$$EST(v_i,vm_j) = max\{T\_Available[j], max_{vm \in pred\ (vmk)}(\ (EFT(vm,vmk) + c(m,i))\} \quad (3)$$

$$EFT(v_i,vm_j) = w_{i,j} + EST(v_i,vm_j) \quad (4)$$

wherepred(vm_k) is the set of immediate predecessors of task v_i, and the task execution start time is represented as T_ Available[j]. Also the execution of the task is initiated only after the necessary data are sent by the v_i has reached at the host vm_jwhich is denoted by the inner max block of equation (3). The resource allocation is maintained in a two-dimensional matrix.

4. The Proposed Algorithm (CFCSC)

HEFT is a well-known List scheduling algorithm for heterogeneous computing systems and it aims at makespan minimization[14]. It is also a simple and most commonly used list scheduling heuristic. Since it was developed before the emergence of Cloud computing and utility grids, it does not include monetary costs[15]. Thus the proposed algorithm CFCSC, considers the monetary cost in the HEFT algorithm.

The proposed algorithm CFCSCmakes certain assumptions :

- A set of heterogeneous virtual machines (VMs) denoted by M are considered for creating cloud environment.
- The communication network is always connected.
- Tasks are executed normally and there are no failures.
- Tasks are non-preemptive.

The objectives of the CFCSC algorithm are:

- To minimize the total monetary cost.
- To balance the load.

4.1. The Cost Function

In Cloud computing environment price of using the resource is a vital parameter in scheduling. There are various pricing models proposed by Amazon[4] and Google[5]. In the proposed Cost-effective Task Scheduling algorithm (CFCSC), the cost of using the virtual machine is based on the CPU speed. Li et al[8] proposed a cost-conscious scheduling algorithm (CCSH) by computing a cost function using CPU speed. The cost function is complex. So the CFCSC uses simple cost function to reduce the computation complexity. The virtual machine with high CPU speed is costlier than the virtual machine with the low CPU speed. In CFCSC, the price P is calculated using the definition

$$P(i,j) = (W(i,j)\ *\ \text{cost of vmj}\ )/\text{minute} \quad (5)$$

where P(i,j) is the cost of executing task v_i in the virtual machine vm_j.The total monetary cost is given by

$$P = \sum_{j \in \text{selected vm}} p(i,j) \quad (6)$$

The CFCSC algorithm incorporates the monetary cost component in the final phase of the algorithm by calculating the Modified Earliest Finish time (MEFT) for the task using the following equation

$$MEFT(v_i, vm_j) = EST(v_i, vm_j) + w_i + p(i,j) \quad (7)$$

The list scheduling algorithms have two phases: the prioritizing phase assigns a priority to each task and a processor selection phase allocates a suitable processor that reduces the heuristic cost function. If more than one task has equal priority, then it is solved by selecting a task randomly.The proposed algorithm CFCSC, in the prioritizing phase weights are assigned to the nodes and the edges in the first step. In the next step, the upward rank calculation is made. Then the nodes are sorted in the descending order using upward rank (Ranku) for prioritization.At the same time, the virtual machines are sorted in the ascending order based on their prices. As one of the objectives is to minimize the monetary cost, the virtual machine with the least cost is given the highest priority. The virtual machine with lower speed is priced low and the virtual machine with higher speed is priced high. This is done to balance the load and to minimize the total monetary cost, which includes only the virtual machine cost. However, it does not include the cost for memory size and storage space. As a final step the algorithm calculates the Modified Earliest Finish Time (MEFT) instead of EFT using equation 7, for the task v_i for every virtualmachine and selects the virtual machine with minimum MEFT value, preserving the precedence constraint.

Table 1.The CFCSC Algorithm

1. Compute *Ranku* b-level(the length of a longest path from a node to the exit node) of all the nodes.
2. Arrange the nodes in a list by decreasing order of *Ranku* values.
3. Arrange the virtual machines list by pricing, in ascending order.
4. Calculate MEFT value for all the nodes.
5. Repeat the steps 7, 8 and 9
6. **begin**
7. The first task v_i in the list is removed.
8. Find the MEFT value of task v_i for all virtual machines.
9. Find the vm_j which has minimum MEFT value for task v_i and assign it to vm_j.
10.**until** all the nodes in the list are scheduled

5. Experiments and Results

The sample DAG in Figure 1 is used to illustrate the CFCSC algorithm,. There areten tasks and three virtual machines with different CPU speed. In the prioritizing phase, Ranku is calculated for all the tasks and sorted in the descending order. Thus the priority of execution of the tasks is 0, 1, 4, 2, 3, 8, 7, 5, 6 and 9. The execution time of the task in different virtual machines is given in Table 2. The cost of using the virtual machines/minute is considered as $0.03, $0.01 and $0.16 for VM1, VM2, VM3 respectively. Based on the price, the virtual machines are sorted as vm2, vm1 and vm3 and the corresponding cost matrix for the tasks is given in Table 3.

The modified earliest finish time (MEFT) which includes the cost factor is also calculated for all the tasks. The mapping of resources using HEFT and CFCSC are shown in Figure 2 and Figure 3 respectively. The CFCSC algorithm balances the load when compared to the HEFT. By applying HEFT algorithm, four tasks (V2, V4, V6, and V9) are assigned to VM1, task (V5) is assigned to VM2 and VM3 is assigned (V0, V1, V3, V7 and V8) with remaining five tasks.

On the other hand, in CFCSC algorithm the virtual machines are sorted by their price. So the order of the virtual machines is changed to VM2, VM1 and VM3. When the CFCSC algorithm is applied, VM2 is assigned with four tasks (V2, V4, V6 and V9), VM1 is assigned with three tasks (V0, V1, V5) and VM3 is assigned with three tasks(V3, V7 and V8). Two tasks V0 and V1 are assigned to VM2 instead of VM3. Hence, the load is shared by all the three virtual machines in CFCSC algorithm as compared to the HEFT algorithm.

Table 2.Computation Time and Priority of Tasks

TASK	Computation Time	VM1	VM2	VM3	*Ranku* (Priority)
T0	4	3.2	4	2.285714	61
T1	2	1.6	2	1.142857	53
T2	1	0.8	1	0.571429	38
T3	18	14.4	18	10.28571	29
T4	11	8.8	11	6.285714	43
T5	3	2.4	3	1.714286	11
T6	1	0.8	1	0.571429	8
T7	15	12	15	8.571429	20
T8	19	15.2	19	10.85714	27
T9	3	2.4	3	1.714286	3

Table 3.Monetary Cost

TASK	VM1	VM2	VM3
T0	0.12	0.04	0.37
T1	0.06	0.02	0.18
T2	0.03	0.01	0.09
T3	0.54	0.18	1.65
T4	0.33	0.11	1.01
T5	0.09	0.03	0.27
T6	0.03	0.01	0.09
T7	0.45	0.15	1.37
T8	0.57	0.19	1.74
T9	0.09	0.03	0.27

Table 4. Total Monetary Cost of Scheduling Algorithms

Tasks	Total Cost	
	CFCSC	HEFT
20	5	5.5
40	17.2	18.2
60	36.6	37.4
80	20.6	21
100	28	29
200	117.7	119
400	158	159.4
600	263.4	264.5
800	253.4	254.7
1000	433.7	435.7

Since the virtual machines are sorted by their price, most of the tasks are assigned to the low price virtual machines resulting in the reduction of total monetary cost in CFCSC when compared to that of the HEFT algorithm. The HEFT algorithm completes all the tasks with the makespan of 43.49secs.at the cost of $5.72. The CFCSC algorithm's makespan is also 43.49secs. But it costs only $5.2.

A DAG generation program developed in Java[16] is used to generate random DAGs. This program generates the needed resources(virtual machines) with various speeds randomly. Given the number of tasks to be generated and the number of virtual machines, the program generates the Random DAG. Thus the generated DAG is fed to the HEFT and CFCSC algorithm (developed in Java) for observation. The makespan and the monetary cost are given as output. The experiment is conducted by varying the tasks from 20 to 1000 and the respective makespan and the monetary costs are observed and tabulated. Table 4 gives the detail of the total monetary cost of HEFT and CFCSC algorithms. Their graphical representation is shown in Figure 4.It was observed that the proposed CFCSC algorithm balances the load and minimizes the total monetary cost. Both HEFT and CFCSC algorithms have the same makespan.

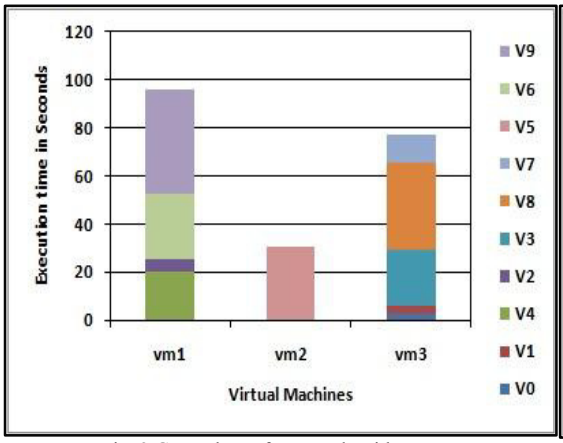

Fig. 2 Gantt chart of HEFT algorithm

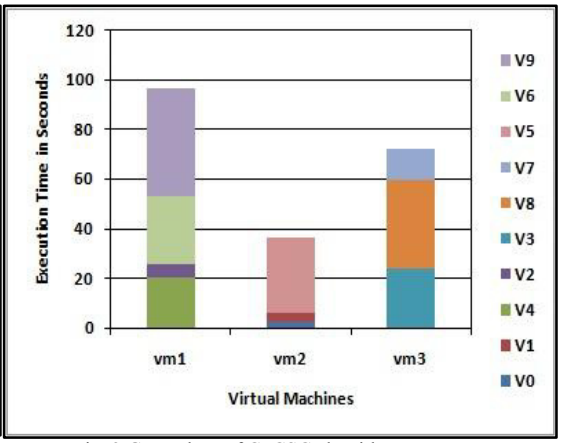

Fig. 3 Gantt chart of CFCSC algorithm

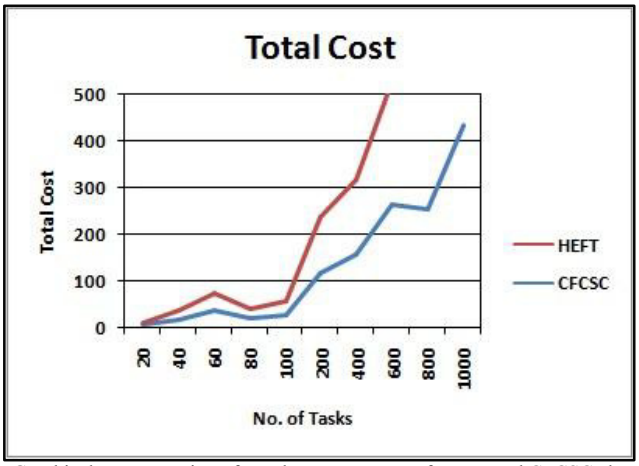

Fig. 4 Graphical representation of Total monetary cost of HEFT and CFCSC algorithms

6. Conclusion

Cloud computing gained the attention of large number of users through its various services. The pay-per-use basis increased the cloud customer population tremendously in few years. Studies show that there is a huge market awaiting due to the consequence of a business model that offers high performance with low cost[15].This paper proposes a CFCSC algorithm that balances the load and minimizes the total monetary cost as compared to the HEFT algorithm. The CFCSC algorithm saves cost by using easier cost function. In future, the CFCSC algorithm can be simulated in Cloudsim tool for various real world applications, namely, Montage, LIGO-1 and LIGO-2. The CFCSC algorithm includes computation cost alone in the cost function. As an extension of the CFCSC algorithm, factors like the storage cost for input and output data, monetary cost for system initialization and memory size can also be incorporated to meet the user's economic constraints. Using the meta-heuristic optimization techniques the makespan can be minimized.

References

1.Kupferman J,Silverman J, Jara P, and Browne J. Scaling into the cloud,.*http://cs.ucsb.edu/~jkupferman/docs/ScalingIntoTheClouds.pdf*; 2009.

2.Armbrust M,Fox A, Griffith R, Joseph A. D, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, toica I, Zaharia M. A view of cloud computing.*Communications of the ACM*; 2010;Apr(53):4 p. 50–58.

3.Lingfang Zeng, BharadwajVeeravalli, Xiaorong Li. ScaleStar: Budget Conscious Scheduling Precedence-Constrained Many-task WorkflowApplications in Cloud.*26th IEEE International Conference on Advanced Information Networking and Applications*;2012.p. 534-541.

4. AmazonWebServices. Available from: URL:*http://aws.amazon.com/ec2*

5. Google AppEngine.Available from: URL:*http://code.google.com/appengine*

6.Ullman J. NP-complete scheduling problems.*Journal of Computer and System Sciences*; 1975;10:3 p. 384–393.

7.Bittencourt L. F, MadeiraE. R. M.HCOC: A cost optimization algorithm for workflow scheduling in hybrid clouds.*Journal of InternetServices and Applications* 2011; 2:3p. 207–227.

8. Li J, Su S, Cheng X, Huang Q, Zhang Z. Cost-conscious scheduling for large graph processing in the cloud.*inProc. of the 13th International Conference on High Performance Computing and Communications (HPCC)*;2011. p. 808–813.

9.Mengxia Zhu, Qishi Wu, Yang Zhao.A cost-effective scheduling algorithm for scientific workflows in clouds.*Performance Computing andCommunications Conference (IPCCC), IEEE 31st International Conference* 2012; p. 256 – 265.DOI: 10.1109/PCCC.2012.6407766.

10.Ranjit Singh, Sarbjeet Singh.Score based deadline constrained workflow scheduling algorithm for Cloud systems. *International Journal onCloud Computing: Services and Architecture (IJCCSA)*;2013; 3:6 p. 31-41.

11. AmandeepVerma, SakshiKaushal.Deadline and Budget Distribution based Cost- Time Optimization Workflow Scheduling Algorithm for Cloud.*International Conference on Recent Advances and Future Trends in Information Technology (iRAFIT2012) Proceedings published in International Journal of Computer Applications (IJCA)*;2012. p.1-4.

12.Ke Liu, Hai Jin, Jinjun Chen, Xiao Liu, Dong Yuan,Yun Yang. A Compromised-Time-Cost Scheduling Algorithm in SwinDeW-C for Instance-Intensive Cost-Constrained Workflows on a Cloud Computing Platform.*The International Journal of High Performance Computing Applications*; 2010;24:4p. 445–456. DOI: 10.1177/1094342010369114.

13.Selvarani, S., and G. SudhaSadhasivam. Improved cost-based algorithm for task scheduling in cloud computing.*In Computational Intelligence and Computing Research (ICCIC), 2010 IEEE International Conference*;2010; p. 1-5.

14.Topcuoglu H, Hariri S, Wu M.Y. Performance effective and low complexity task scheduling for heterogeneous computing.*IEEE Transactions on Parallel and Distributed Systems*;2002;13:3p. 260–274.

15. Luiz F. Bittencourt, Edmundo R. M. Madeira, Nelson L. S. da Fonseca.Scheduling in hybrid clouds. *IEEE Communications Magazine*;2012;50:9p. 41-47.

16.GeorgeAmalarethinamD.I, Joyce MaryG. J. DAGEN – A tool to generate arbitrary Directed Acyclic Graphs used for Multiprocessor Scheduling. *International Journal of Research and Reviews in Computer Science (IJRRCS)*; 2011;2:3 p. 782 – 787.