Activity Based Cost Scheduling Using Priority Approach in Cloud Computing

Monika*, Pardeep Kumar* and Sanjay Tyagi*

ABSTRACT

Since cost is the most effective factor in cloud computing, so in this paper, an approach is being proposed based on cost applied on datacenters using basic approach of ABC (Activity-based Cost) scheduling algorithm. The proposed approach works in two phases. In the first phase, we categorized three datacenters' based on cost. Tasks were assigned to datacenters based on cost. In the second phase, three different priority queues were implemented and tasks were assigned to VMs' according to their priorities.

Keywords: Activity Based Cost approach, Cloud Computing, Datacenter, Datacenter Based Approach, Scheduling Algorithm

I. INTRODUCTION

Before cloud computing, resources were used in a limited way and the cost of hardware and their maintenance were also high. When cloud computing came in existence, cost of maintaining resources became less. That's why researchers are taking more attention in this field. Cloud computing's on-demand network access, resource pooling, elasticity, and computing resources availability, e.g., networks, servers, and storage had made it so popular. Every organization is thinking of using these services. As cost is the most impacting factor for an organization, researchers started paying more attention in cost effective scheduling algorithms. ABC is one of the cost effective approaches used in cloud computing.

II. RELATED WORK

Cost is the most considerable part of an organization not only in recent years, but also since when organization came into existence. To reduce, cost Gary Cokins introduced Activity Based Cost approach in management and named "ABC/M" in 1988 [1]. Further, in 1993, it was also implemented by Yair M. Babad et. al. by grouping the activities into single drive to calculate or trace the cost of product or service [2]. This was the starting age of ABC algorithm. Carsten Homburg (2001) proposed algorithm based on ABC [3]. He suggested that with the existing cost drivers (cost of activities that are interconnected or that affect the other activities), a new combination of cost driver should be replaced. With this approach, system complexity with ABC will be same but cost allocation will be accurate.

When cloud computing technology came in existence, researchers also tried to implement ABC approach in scheduling. For this, QI CAO et. al. (2009) implement optimized ABC algorithm [4]. The proposed algorithm used priority in ABC that is calculated on the basis of task cost, individual resource cost and earned profit from individual tasks. Requested tasks were consigned to VMs according to defined priorities. Instead of using three queues, Shachee Parikh et. al. (2010) used double level priority queues in ABC approach of scheduling [5]. In their proposed methodology, all tasks were sorted according to their processing power and provide the resources according to tasks' turnaround time. Priority in ABC was used by Mrs. S.

^{*} Department of Computer Science & Applications, Kurukshetra University, Kurukshetra, India, *E-mail: monika.banasthali@gmail.com; mittalkuk@gmail.com; tyagikuk@gmail.com*

Selvarani et. al. (2010) in a different way [6]. Their proposed work was based on new arriving tasks. New tasks were arranged into priority queues according to their size and priority. Ashutosh Ingole et. al. (2011) suggested another cost effective algorithm based on ABC [7]. In their proposed algorithm, firstly highest priority tasks were allotted to the resources and then medium & low priority tasks were shifted and consigned to the resources according to their priorities. Except, ABC approach in cloud computing Zhi Yang et. al. (2011) proposed another cost effective scheduling algorithm [8]. In their proposed algorithm maximum profit was achieved through ordering supplier's need. First supplier's tasks with the highest profit were assigned resources before second highest profit tasks. The process was continued until all needs got the resources.

III. ABC (ACTIVITY BASED COST) APPROACH

Activity based cost approach is used not only for finding the cost of resource, but also performance cost [4] [6] [7]. Activity based approach means each activity is different and cost occurring from these activities should be different. In this approach, cost is measured in terms of CPU, memory and time required while fulfilling the individual request. After calculating the cost, priority is calculated of the individual task and divide these tasks into three categories HIGH, MID, and LOW. The following equation calculates individual task's priority [4] [6] [7]:

$$L_{k} = \sum_{i=0}^{n} R_{i,k} * C_{i,k} / P_{k}$$
⁽¹⁾

Here n is the no. of resources. Other parameters are explained below:

- 1) R_{ik} : The ith individual use of resources by the kth task.
- 2) C_{ik} : The cost of the ith individual use of resources by the kth task.
- 3) P_{μ} : The profit earned from the kth task.
- 4) L_{k} : The priority level of the kth task.



Figure 1: Datacenter Based ABC algorithm

IV. PROPOSED WORK

Earlier researches were performed at the single datacenter, which causes slow execution speed. All tasks priorities were calculated and divided into different categories according to their priorities. After categorizing, tasks are grouped in three different queues (LOW, MID, HIGH) and are assigned resources simultaneously for the fast execution. In this paper, a new approach is proposed in two phases, based on priority. In the first phase, datacenters were distributed based on their cost. Service provider divides three datacenters and assigns the tasks as per the cost requirement. In the second phase, basic ABC algorithm has been applied on individual host based on task's priority using equation (1). Flowchart of proposed algorithm is as under:

4.1. Proposed Algorithm

r = resource size, t = task size

- 1) Calculate ETM (Execution Time Matrix) of each task with respect to each resource by using $\sum_{i=0}^{t} \sum_{j=0}^{r} t.\text{length} / r.\text{mips.}$
- 2) Calculate CM (Cost Matrix) of each task with respect to resource by using $\sum_{i=0}^{t} \sum_{j=0}^{r} \operatorname{etm}[i][j] * r.mips.$
- 3) Find out minimum cost of each task w.r.t. resource.
- 4) Catagorise datacenters according to cost.
- 5) Calculate priority of each task in each datacenter by using task.priority = cost/profit
- 6) Assign VMs according to their cost and priority.

In proposed approach, the execution time of tasks is calculated based on their MIPS.

4.2. Experimental parameters

For this proposed work, following configuration has been taken in cloudsim simulator:

-							
Datacenter							
No. of host per datacenter	3						
P.E. (Process Element) per datacenter / Speed	3 / 21000 MIPS						
RAM	10192 MB						
Storage	1000000 MB						
Bandwidth (BW)	10000 MIPS						
Scheduling policy	Timeshared						

Table 1

Datacenter configuration

Table 2VM Configuration for high cost datacenter

VM Configuration (High Cost Datacenter)						
No. of VM	3					
No. of P.E. (process element) per VM.	1					
Storage size	10000 MB					
RAM	4096 MB					
Bandwidth (BW)	1000 MIPS					
Virtual Machine Manager (VMM)	Xen					
MIPS	21000(high priority) 20000(mid priority) 19000(low priority)					

VM Configuration for me	dium cost datacenter					
Vm Configuration (Media	um Cost Datacenter)					
No. of VM	No. of VM 3					
No. of P.E. process element) per VM	1					
Storage size	10000 MB					
RAM	4096 MB					
Bandwidth (BW)	1000 MIPS					
Virtual Machine Manager (VMM)	Xen					
MIPS	18000(high priority) 17000(mid priority) 16000(low priority)					

Table 3

Table 4 VM Configuration for low cost detecentor				
VM	Configuration for low cost datacenter			

Vm Configuration (Lov	v Cost Datacenter)			
No. of VM	3			
No. of P.E. process element) per VM	1			
Storage size	10000 MB			
RAM	4096 MB			
Bandwidth (BW)	1000 MIPS			
Virtual Machine Manager (VMM)	Xen			
MIPS	15000(high priority) 14000(mid priority) 13000(low priority)			

4.3. Result and Analysis

After implementing proposed algorithm, tasks were divided among three datacenters' based on cost and then tasks were provided resource or VMs based their priorities. Three VMs with different MIPS were implemented on each datacenter. Our result shows less execution time as compared to basic ABC scheduling policy. Table 5 shows execution time taken by various tasks on basic ABC scheduling and proposed algorithm:

 Table 5

 Comparison of ABC scheduling & Datacenter Based ABC approach

	Datac	enter Based	I ABC App	proach	Basic ABC Approach				
No. of cloudlets	Н	М	L	A	Н	М	L	А	
25	2.73	4.77	4.1	3.23	3.14	14.34	36.03	1.7	
50	8.09	9.61	7.71	4.6	37.64	30.25	73.03	53.45	
100	25.85	29. 9 2	27. 69	12.03	49.96	85.77	99.47	88.48	
150	38.6	43.51	40.86	19.96	102.67	125.65	140.87	130.64	
2,00	43 2.7	43 79	45 42	21.52	132.43	176 41	184 67	178-38	

Table 5 shows execution time taken by various tasks using ABC approach and using proposed algorithm. In the above table, variables H, M, L, A show time taken by high, mid, low and all (high, mid and low) queues to execute all tasks.

Graph representation of above results:



Figure 2: Execution of all tasks in High Priority Queue.



Figure 3: Execution of all tasks in Medium Priority Queue



Figure 4: Execution of all the tasks in Low Priority Queue



Figure 5: Execution of all the tasks in all queues





Figure 7: Datacenter Based ABC algorithm (Proposed)

Fig. 6 shows the execution of tasks by Basic ABC algorithm and fig. 7 shows the execution of various tasks by proposed algorithms. Graphs and tables discussed above show that proposed algorithm executes tasks according to cost in less time as compared to basic ABC approach.

4.4. Output screenshot

Edit View Navigate Source Refactor Run Debug Profil	Team Tools Window Help	Q - Search (Ctrl-I)
1 2 2 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	î ĭi ▷·B;·O·II ● ● ▲ ★ ▲ 4 5 m	
ices Projects # Files	🗐 🖄 cht5.java 🕫 Start Page 🗴 🖄 DatacenterCharacteristics.java 🕫 🗷 Datacenter.java 🕫	a PriorityCalc.java 🗴 🗟 Resource.java 🛪 🗟 Task.java 💷 💌
doud_analyst	· .	
JavaApplication1		
Source Packages		
🖞 🔠 «default package»	Output - JavaApplication1 (run)	
CETaskScheduler.java	IVm2cheduler.vmCreatel Allocation of VH #7 to Host #7 failed by HIFS	
Holpon jeve	[VmScheduler.vmCreate] Allocation of VM #7 to Host #8 failed by MIPS	
- 20 Helpert Java	(VaScheduler.vaCreate) Allocation of VM \$7 to Most \$0 failed by MIPS	
- dd PritarityCakjava	(Valebadeler, Valebale) Allocation of Valebales of Social by MTR	
- Id8. PriorityCalc1.tava	23 (TheCabadular unConsent Silanseins of Th 45 on New 41 doiled by MTDC	
Resource iava	(Westedular unGreatel Allocation of UN 12 to Host 14 failed by MIRS	
Besource1.java	(Vmfcheduler.vmCreate) Allocation of VM #8 to Host #5 failed by MIPS	
Task tava	(VELCREQUIET.VEUTeste) Allocation of VE EU to Host F0 failed by Hist	
	(Unfekadular.veCreate) Allocation of VM #0 to Hore #7 failed by MIDE	1
E. Test Course in ca	[VmScheduler.vmCreate] Allocation of VM #8 to Host #8 failed by MIPS	1
Tak Crown Linux	(VmScheduler.vmCreate) Allocation of VM #8 to Host #0 failed by MIPS	1
and the state of t	(VmScheduler.vmSreate) Allocation of VM #0 to Host #1 failed by MIPS	(
Life shaft to see	(Vafchedelor.VElleste) Allocation of WK #8 to Nowt #2 dealed by MIPS	l
- Ing. cbcs.java	0.1: Broker: VM 20 has been created in Datacenter 22, Eost 20	
ED CDC1, Java	0.1. Broker: VM #1 has been created in Datacenter #2, Host #1)
Egg cots.java	0.1: broker: VN ** has been created in Datacenter *2, host *2	
- [0] vmistLJava	O.I. Cleation of the point of the point of the point of the	
exemples.org.couldus.coucsm.examples	0.1: Broker: Creation of VM #5 failed in Datacenter #2	
	0.1: Broker: Greation of VM #6 failed in Datacenter #2	
examples.org.coudbus.coudsm.examples.netvork.da	ence 0.1: Eroker: Creation of VM #7 failed in Datacenter #2	
evanoies are claudhus claudsin evanoies power	bit Backer, Consider of Vid (C Cathod in December (C	
Image:	^b 0.1: Droker: Trying to Create VM #3 in Datacenter_1	
examples.org.cloudbus.cloudsim.examples.power.rand interview.examples.power.rand interview.e	0.1: Ercker: Trying to Create VM #4 in Datacenter 1	
examples.workload.planetlab.20110303	0.1: Broker: Trying to Create VM #5 in Datacenter 1	
B-III examples.workload.planetlab.20110306	0.1: Broker: Trying to Create VM f6 in Detecenter_1	
examples.workload.planetlab.20110309	0.1: Lesses Toyang to Consta UN 47 an Detaemter 1	
E- E energies. vo Noviglarellab. 20110022	U.1: progen: inving to create VM #8 in Datacenter 1 [UmSchedular umCreate] Silocation of UM #6 to Heat #8 failed by MIDS	
Pi- E examples. workload.planetlab.20110325	lymcchequier.ymCreatel Allocation of VM #6 to Host #6 failed by MIFS	
(i) III examples.workload.planetlab.20110403	" (Vmfcheduler.vmCreate) Allocation of VH #6 to Host #5 failed by HIPS	3
18	IVmScheduler.vmCreatel Allocation of VM #A to Most #A failed by MIPS	

Figure 8: Creation of VMs in high-cost datacenter

JavaApplication1 - NetBeans IDE 8.0.2		
File Edit View Navigate Source Refactor Run Debug Profile Team 1	Tools Window Help	Q Search (Ctrl+I)
😷 🛍 🎴 🍓 🦈 🥐 🖉 🔤 config> 💽 🏹 🎉	3 D · 15 · (7) · 1 0 0 1 1 1 2 1 4 1 1	
Services Projects # Files	B det.java x Szet Page x E Detscence/Characteristics.java x ∅ Detscenter java x ∞ PriorityCac.java	20 🕅 Resource.jøva 22 🖉 Tasc.jøva 🤄 🖭 💌
Source Passage Source P	Output-JewaApplications (int) # Transmissional (int) # TWDSchedular.wCreased Allocation of W # 7 to Rost # 6 failed by MTDS WDSchedular.wCreased Allocation of W # 7 to Rost # 6 failed by MTDS WDSchedular.wCreased Allocation of W # 7 to Rost # 6 failed by MTDS WDSchedular.wCreased Allocation of W # 7 to Rost # 1 failed by MTDS WDSchedular.wCreased Allocation of W # 7 to Rost # 0 failed by MTDS WDSchedular.wCreased Allocation of W # 7 to Rost # 0 failed by MTDS WDSchedular.wCreased Allocation of W # 7 to Rost # 0 failed by MTDS WDSchedular.wCreased Allocation of W # 7 to Rost # 0 failed by MTDS VDSchedular.wCreased Allocation of W # 7 to Rost # 2 failed by MTDS VDSchedular.wCreased Allocation of W # 5 to Rost # 2 failed by MTDS VDSchedular.wCreased Allocation of W # 5 to Rost # 2 failed by MTDS VDSchedular.wCreased Allocation of W # 5 to Rost # 2 failed by MTDS VDSchedular.wCreased Allocation of W # 5 to Rost # 2 failed by MTDS VDSchedular.wCreased Allocation of W # 5 to Rost # 2 failed by MTDS VDSchedular.wCreased Allocation of W # 5 to Rost # 2 failed by MTDS VDSchedular.wCreased Allocation of W # 5 to Rost # 2 failed by MTDS VDSchedular.wCreased Allocation of W # 5 to Rost # 5 failed by MTDS VDSchedular.wCreased Allocation of W # 5 to Rost # 5 failed by MTDS VDSchedular.wCreased Al	
example: org.cloudbus.cloudsin.examples.power.planetlab	0.2: broker: irying to treate VM #0 in Datacenter_2 0.2: broker: Trying to Greate VM #7 in Datacenter 2	

Figure 9: Creation of VMs in medium-cost datacenter







Figure 11: Sending tasks or cloudlets to the particular VMs

e Edit View Navigate Source Refactor Run Debug Profile Team	Tools Win	fow Help							(SLT. Search (CH+E)	
2 2 2 4 5 6 (mint only)	留D・	部・③	· 🗐 🔘	◎ 曲 Ⅲ 单	1 曲 1	古開				
ervices Projects II Files	Bass	not il Sta	tPage H	Ostacenter/Characteria	dica.java	n 🗟 Delaca	nter java 🛛 🖁	PriorityCalc.java II 🗃 A	insource.jeve — H 🗟 Task.jeve	
🖢 doud_analyst 🔹										
terestopication1	1									
🕀 🚡 Source Packages										
🖟 🛄 -defailtpadage>	Output -	JavaApplicati	M (nun) Ino							
	ED :	and a second second second	and the second							
- R Helper, java	100	THE R PLACE	comparents.							
- 🗃 Helper1.java	- IN		OUTPUT							
 PriorityCalc.ava 	1 H c	loudies ID	STATUS	Data center 10	V06 2.0	71.04	Start Time	Finish Time		
- RiorityCalc1java	182	3	SOCCESS	2	1	0.59	0.8	0.09		
- 😥 Resource.java		2.2	soccass	3	4	0.66	0.3	0.97		
Resource 1.jeva		212	SOCCESS.	4	7	0.77	Q.9	8.07		
- E Takjava		2	SOCCESS	2	0	1.45	0.3	1.79		
- R Teklava			SOCCESS	2		5.54	0.3	1.14		
- Bit Leakerpup Java		- î.	SUCCESS	· · ·	°.	1.64	0.3	1.04		
- 🖉 TaskGroup Ljeve		3	\$00CE44			5.99	0.3	2.07		
- B. dd2ave		20	SUCCESS	- a		3.77	0.3	8.07		
- iš ddava		2.0	STOCKESS	4	6	2	0.3	2.3		
- R. detpra		* *	400709-04		4	0.11	A 8	0.41		
- init state for		19	SOCCESS	4	4	2.11	0.3	2.41		
ill velett.seva		2.4	BACCERS.	4		2.2	9.3	5.6		
8 III exercise on diadha diador exercises		22	success	4		2.41	0.3	2.71		
iii iiii examples org. cloudbus. cloudsin. examples. network		23	soccas	4		2.41	0.3	2.72		
iii) iiii) extension on doubte doubte supervise options detacente			POCCESS	-	-	2.63	0.3	2.99		
examples oro, doudous, doudein, examples, power		-	anyone as			2.74	0.5	5.04		
(i) [1] examples are cloudes, double, examples nowe size tab		ē.	EDCCERR	;	-	2.74	0.3	3.04		
(2) III) exercises and cloudless clouders exercises oncer candies		÷	SOCCESS.	2	2	2.74	0.0	8.04		
iii iiii examples workload planetab. 20110303		27	accessa.	3		3.1.4	0.3	3.42		
 III exercise workload planetab. 2010/206 		1.3	soccast	3	6	3.23	0.3	3.53		
R-III examples workload planettab. 20110309		1.4	SOCCESS	3	6	3.23	0.3	3.63		
 III examples undited should be \$110,000 		2.5	success	3	5	3.25	0.3	3.63		
		24	SOCCESS	3		3.23	0.0	3.53		
examples, workload, planetab, 20110-403		LOUGSLE FL	NUMBER OF STREET	street dis second.	- 1					
	1.1.1	case division	entral (1996)	AND AN PROVING						
	d have been									

Figure 12: The result, if all tasks are executed by the low priority queue

Figures 8, 9,10,11,12 shown above show the flow of proposed approach. Fig. 8, 9, 10 shows the creation of VMs in different datacenters. Fig. 11 shows the binding to VMS to the clouds and finally fig. 12 shows the execution of tasks.

V. CONCLUSION AND FUTURE WORK

In this paper, a basic framework for scheduling in cloud computing based on cost has been proposed. Cloud was divided into three categories of datacenters based on cost. Tasks with high cost were assigned with high processing power units in the first datacenter, while the medium cost tasks were assigned to the medium processing power unit. In the last, tasks with low cost were assigned to the third datacenter with the low processing power unit. This paper calculates the expected cost of the tasks submitted and accordingly the task is sent to the datacenter. Proposed approach is based on the expected cost of tasks before their execution. The results of the proposed approach are found to be better as compared to basic ABC approach i.e. the time taken for completion of a task is less as compared to ABC. In future, the scheduling policies which will prove beneficial in terms of cost/profit to both service provider as well as client.

REFERENCES

- [1] Gary Cokins, "Activity Based Cost Management", John Wiley & Sons, New York, 2001.
- [2] Yair M. Babad and Bala V. Balachandran, "Cost Driver Optimization in Activity-Based Costing", *American Accounting Association*, July, 1993, pp. 563-575.
- [3] Carsten Homburg, "A note on optimal cost driver selection in ABC", Management Accounting Research, 2001, pp. 197–205.
- [4] Qi Cao, Zhi-Bo Wei, Wen-Mao Gong, "An Optimized Algorithm for Task Scheduling Based On Activity Based Costing in Cloud Computing", IEEE, 2009.
- [5] Shachee Parikh, Richa Sinha, "Double Level Priority based Optimization Algorithm for Task Scheduling in Cloud Computing", *International Journal of Computer Applications*, January 2013.
- [6] Mrs.S.Selvarani, Dr.G.Sudha Sadhasivam, "Improved Cost-Based Algorithm For Task Scheduling In Cloud Computing", IEEE, 2010.
- [7] Ashutosh Ingole, Sumit Chavan, Utkarsh Pawde, "An Optimized Algorithm for Task Scheduling based on Activity-based Costing in Cloud Computing", 2nd National Conference on Information and Communication Technology, 2011.
- [8] Zhi Yang, Changqin Yin, Yan Liu, "A Cost-based Resource Scheduling Paradigm in Cloud Computing", *12th International Conference on Parallel and Distributed Computing, Applications and Technologies*, 2011, IEEE, pp. 41.