IOSR Journal of Engineering (IOSRJEN)
ISSN: 2250-3021 Volume 2, Issue 7(July 2012), PP 141-147
www.iosrjen.org

Advanced Task Scheduling for Cloud Service Provider Using Genetic Algorithm

[1]Sourav Banerjee, [2]Mainak Adhikari, [3]Utpal Biswas
[1]*Department of Computer Science and Engineering Kalyani Government Engineering College Kalyani,Nadia, West Bengal*
[2]*Department of Computer Science and Engineering University of Kalyani Kalyani,Nadia, West Bengal*
[3]*Department of Computer Science and Engineering University of Kalyani Kalyani,Nadia, West Bengal*

Abstract: - *Cloud computing is one the upcoming latest technology which is developing drastically. Today lots of business organizations and educational institutions using Cloud environment. But one of the most important things is to increase the Quality of Service (QoS) of the system. To improve the QoS in a system one must need to reduce the waiting time of the system. Genetic Algorithm (GA) is a heuristic search technique which produces the optimal solution of the tasks. This work produces one scheduling algorithm based on GA to optimize the waiting time of overall system. The cloud environment is divided into two parts mainly, one is Cloud User (CU) and another is Cloud Service Provider (CSP). CU sends service requests to the CSP and all the requests are stored in a Request Queue (RQ) inside CSP which directly communicates with GA Module Queue Sequencer (GAQS). GAQS perform background operation, like daemon, with extreme dedication and selects the best sequence of jobs to be executed which minimize the Waiting time (WT) of the tasks using Round Robin (RR) scheduling Algorithm and store them into Buffer Queue (BQ). Then the jobs must be scheduled by the Job Scheduler (JS) and select the particular resource from resource pool (RP) which it needs for execution.*

Keywords: - *Cloud computing, Quality of Service, Cloud User, Cloud Service Provider, Request Queue, GA Module Queue Sequencer, Buffer Queue, Waiting Time, Round Robin Scheduling Algorithm, Genetic Algorithm, Resource Pool.*

I. INTRODUCTION

Cloud computing [1,2,3,4] is the process of delivering computing as a service rather than a product, whereby shared resources, software, and information are provided to users and other devices as a utility over the network. Cloud computing environment is highly dynamic; the system load and computing resource utilization exhibit a rapidly changing characteristic over time. Therefore Cloud service provider normally over-position computing resources to accommodate the peak load and computing resources are typically left under-utilize in nonpeak time. Cloud environment allows users to use applications without installation and access their personal files at any computer with Internet access. End users access cloud based applications through a web browser or a light weight desktop or mobile app while the business software and data are stored inside CSP at a remote location. Cloud application providers [6, 7, 8, 9] strive to give the better service and performance than if the software programs were installed locally on end-user machines. Cloud environment [6, 8, 9] is used in lot of fields like in IT industries, educational institute as well as in other industries. In this paper we have proposed Cloud Service Provider, figure 1, which includes mainly three parts- GA Module Queue Sequencer, Job Scheduler (JS) and Resource Pool (RP). All service requests which are coming from Cloud Users domain are stored in RQ which is in GAQS. Now the requested processes must communicate with GAQS processor (GAP) and the processor finds out the appropriate sequence of tasks which reduce the waiting time of the tasks. GAQS processor then communicate directly with JS which schedules the tasks using Round Robin scheduling algorithm and communicate with RP and tries to assign each of these jobs as per their requirement to the resources. But the main problem here is that to find out the best sequence of the tasks from all possible sequences of tasks and JS schedules those tasks and optimize total Waiting time of those jobs.

The jobs assignment task is done by JS. So JS must need to assign the task such a way that assignments of the jobs to the resources must be fruitful as per as CU requests and the total execution time must be optimal of the whole operations. In next two sections discuss about our proposed model of CSP and one Genetic based scheduling an algorithm which assigns the task to the resource as per the CU's demand and also to optimize the total waiting time of those tasks.

II. PROPOSED MODEL

Before starting to discuss about Cloud queueing model first we discuss about the Genetic algorithm and then site our proposed scheduling algorithm based on Genetic algorithm.

Genetic algorithms (GA) [5, 10, 11, 12] were first proposed by the John Holland in the 1960s [5, 10, 11, 12]. The GA [5, 10, 11, 12] is a heuristic search technique that simulates the processes of natural selection and evolution. Genetic algorithm (GA) is a promising global optimization technique. It works by emulating the natural process of evolution as a means of progressing towards the optimal solution. A genetic algorithm has the capability to find out the optimal job sequence which is to be allocated to the processor.
General Algorithm perform its general operations using the following steps-

A. Select the fixed size chromosomes from the from the population set.
B. Perform any one type encoding operation on the chromosomes of the chromosome sets.
C. Select the best two chromosomes from the chromosome set using their fitness value.
D. Perform the crossover between two chromosomes and get two different offspring.
E. Perform the mutation operation on those offspring just interchanging the bit positions.
F. Continue the steps A to B until get the best solution of the population.
G. Finally perform the elitism operation of the chromosomes means store the best chromosomes in to the system for future use.

Now we discuss our algorithm based on Genetic Algorithm step by step-.

a) Cloud users send the request to the Cloud service provider for the resource or resources.
b) CSP stores the request initially into request queue of GAQS.
c) GAQS processor then select set of tasks from the RQ and rearrange them until it gets the best arrangement of the tasks.
d) GAQS store the final task set into the buffer queue.
e) Then job scheduler execute the tasks one by one using round robin scheduling algorithm and select the resource or resources to the cloud users.

Figure 1 describes the architecture of the GA guided scheduling mechanism and the execution steps also shown by the numbering.

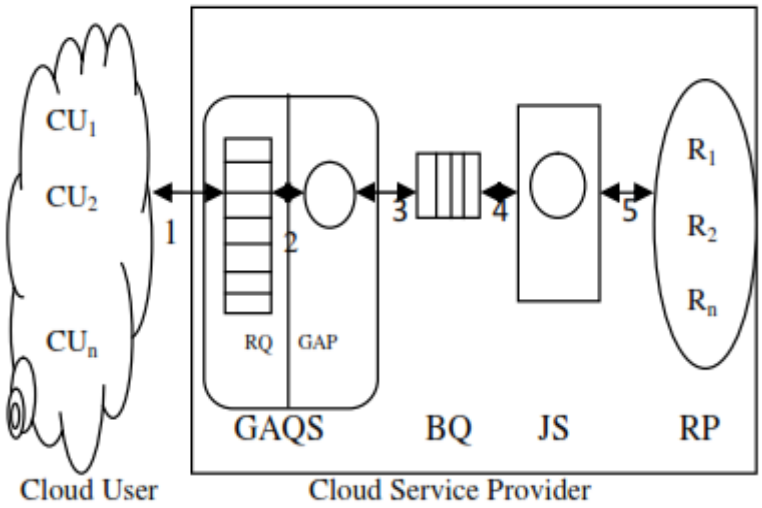

Figure 1: Architecture of Cloud queuing model using Genetic Algorithm

Now we discuss the details of GAQS operation step by step-

1) Request initially comes from the CU for the resources and store into the resource pool.
2) Now GA processor execute the following steps until the best sequence of tasks are produced-

   a) Select the suitable number of tasks from the request pool using their fitness value (means the tasks those who are ready to execute has the higher fitness value compare to the no ready tasks) and create one chromosome.
   b) Now perform the mutation operation on the tasks, just to interchange the positions of them and find out the waiting time of that sequence using round robin scheduling algorithm individually.
   c) Choose the best sequence of tasks from the task sets which have least waiting time, this step is known as elitism.

3) Finally the tasks that produced by the GAQS must be stored in the buffer queue and latter the JS execute the operation using those tasks.

In Figure 2 diagrammatically shown the basic architecture of GAQS which is already discuss in previous part.

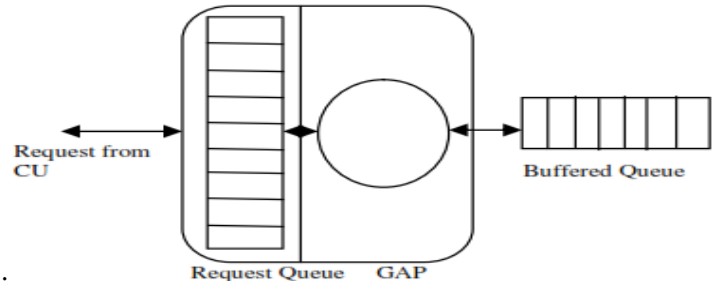

Figure 2: Diagram of GA Module Queue Sequencer

In the next section we discuss our propose algorithm using one example.

III. PERFORMANCE ANALYSIS

In this section we elaborately discuss our propose algorithm using one example. Suppose Cloud users sends n number request for the resources and those resources initially store into the request queue in GAQS like P_1, P_2... P_n as the request come from CU. Now GA processor of GAQS select the tasks from the RP those are ready to execute. Suppose first time tasks P_1, P_2, P_3 are ready to execute and their burst times are 10, 15 and 5 respectively. Now GAP executes all possible sequences of task one by one using Round Robin scheduling. If there are n number of tasks are ready to execute, so the number of possible ways are n!. Here three tasks are ready to execute, so the possible way to execute of the tasks into JS are 3! Or 6 way. We discuss all of them one by one. Here we mention the time quantum of the tasks is 5 for Round Robin scheduling operation.

Process	Burst Time
P_1	10
P_2	15
P_3	5

Table 1. Process table

The table 1 is depicting here a sample snapshot of three processes whose CPU burst are as follows 10, 15 and 5. Here we have mentioned the burst table according to the present scenario of system state.

P_1	P_2	P_3	P_1	P_2	P_2

0 5 10 15 20 25 30

Table 1.1. Burst table for the above processes

The estimated waiting time of the several processes and the average waiting time of the cloud service provider system will be as follows

Process name	Calculation of waiting time	Output
P_1	(0+(15-5))	10
P_2	(5+(20-10))	15
P_3		10

Table 1.2. Waiting time of several processes

Here the average Waiting time = (10+15+10)/3 = 35/3 = 11.6. The second option is being explained in the next table of process pool. Here a slight variation can be seen using different notion of process shuffling inside CSP.

Process	Burst time
P_2	15
P_1	10
P_3	5

Table 2. Process table

Table 2 in the above is defining the process pool with several Burst Time 15, 10 and 5. Here we have mentioned the burst table according to the present scenario of system state.

P_2	P_1	P_3	P_2	$P_{1`}$	P_2

0 5 10 15 20 25 30

Table 2.1. Burst table for the above processes

The estimated waiting time of the several processes and the average waiting time of the cloud service provider system will be as follows

Process name	Calculation of waiting time	Output
P_1	(5+(20-10))	15
P_2	(0+(15-5)+(25-20)	15
P_3		10

Table 2.2. Waiting time of several processes

Here the average Waiting time = (15+15+10)/3 = 40/3 = 13.3. The third option is being explained in the next table of process pool. Here a slight variation can be seen using different notion of process shuffling inside CSP.

Process	Burst Time
P_1	10
P_3	5
P_2	15

Table 3. Process table

Table 3 in the above is defining the process pool with several Burst Time 10, 5 and 15. Here we have mentioned the burst table according to the present scenario of system state.

P_1	P_3	P_2	P_1	P_2	P_2

0 5 10 15 20 25 30

Table 3.1 Burst table for the above processes

The estimated waiting time of the several processes and the average waiting time of the cloud service provider system will be as follows

Process name	Calculation of waiting time	Output
P_1	(0+(15-5))	10
P_2	(10+(20-15))	15
P_3		5

Table 3.2 Waiting time for several processes

Here the average Waiting time = (10+15+5)/3 = 30/3 = 10. The fourth option is being explained in the next table of process pool. Here a slight variation can be seen using different notion of process shuffling inside CSP.

Process	Burst Time
P_2	10
P_3	5
P_1	15

Table 4. Process table

Table 4 in the above is defining the process pool with several Burst Time 15, 5 and 10. Here we have mentioned the burst table according to the present scenario of system state.

P_2	P_3	P_1	P_2	P_1	P_2

0 5 10 15 20 25 30

Table 4.1 Burst Table for the above processes

The estimated waiting time of the several processes and the average waiting time of the cloud service provider system will be as follows

Process name	Calculation of waiting time	Output
P_1	(10+(20-15))	15
P_2	(0+(15-5)+(25-20))	15
P_3		5

Table 4.2 Waiting time for several processes

Here the average Waiting time = (15+15+5)/3 = 35/3 = 11.6. The fifth option is being explained in the next table of process pool. Here a slight variation can be seen using different notion of process shuffling inside CSP

Process	Burst Time
P_3	5
P_2	15
P_1	10

Table 5. Process table

Table 5 in the above is defining the process pool with several Burst Time 5, 15 and 10. Here we have mentioned the burst table according to the present scenario of system state.

P_3	P_2	P_1	P_2	P_1	P_2

0 5 10 15 20 25 30

Table 5.1 Burst table for the above processes

The estimated waiting time of the several processes and the average waiting time of the cloud service provider system will be as follows

Process name	Calculation of waiting time	Output
P_1	(10+(20-15))	15
P_2	(0+(15-5)+(25-20))	15
P_3		0

Table 5.2 Waiting time for several processes

Here the average Waiting time = (15+15+0)/3 = 30/3 = 10. The sixth option is being explained in the next table of process pool. Here a slight variation can be seen using different notion of process shuffling inside CSP

Process	Burst Time
P_3	10
P_1	5
P_2	15

Table 6. Process table

Table 6 in the above is defining the process pool with several Burst Time 5, 10 and 15. Here we have mentioned the burst table according to the present scenario of system state.

P_3	P_1	P_2	P_1	P_2	P_2

0 5 10 15 20 25 30

Table 6.1 Burst Table for the above processes

The estimated waiting time of the several processes and the average waiting time of the cloud service provider system will be as follows

Process name	Calculation of waiting time	Output
P_1	(5+(15-10))	10
P_2	(10+(20-15))	15
P_3		0

Table 6.2 Waiting time for several processes

Here the average Waiting time = (10+15+0)/3 = 25/3 = 8.3. Here the average waiting time of all possible arrangement of three processes those are ready to execute. From that we see that if we arrange the processes using sixth table and then execute them the average waiting time of the tasks must be optimal. So this particular sequence must be initially store into the buffer queue and JS execute the tasks using the sequence which is stored into BQ and select the resources as CUs demands. So, using GA we reduce the waiting time of the overall system.

3.1 ISSUES ARISING ON THE ABOVE PERFORMANCE

Assumption related to GA based Algorithm:-

i. The algorithm must operate on multi-resource based CSP system. It must not operate on accessory system because it creates an overhead of the whole system.
ii. The GA processor must operate on the tasks those are ready in RQ to execute. The tasks those are recently come to RQ or the tasks those are come after the execution start by the GA processor must not execute at that time.

Advantages of GA based Algorithm:-

i. The algorithm must be executed in background. So it must not hamper the execution of the selected tasks on the other processors.
ii. First time it requires some time to find out the sequence of the tasks those are executed in other processors. But from next step it must not take any extra time because GAP find out the exact sequence of the task in the background and other processor must execute in foreground.
iii. This algorithm reduces the overall average waiting time of the system.
iv. This algorithm increases the throughput of the system.

Disadvantages of GA based Algorithm:-

i. The algorithm is not applicable in single resource based CSP system because the throughput of the

system must be degraded a lot.

ii. The algorithm is not applicable in online application, means when any new process arrives at RQ the GAP must not use it in current operation which is already started. So the algorithm works as static mode.

IV. CONCLUSION AND FUTURE WORK

Previous two sections we describe the GA based scheduling algorithm. Genetic algorithm is a heuristic search algorithm and it always find out the solution which takes minimum time to execute or find optimal solution from the set of possible solutions. This algorithm must increase the throughput of the system. Here the tasks those are ready to execute are operate in all possible way and find the best sequence of the tasks which average waiting time must be optical one. So the waiting time of the system must be optimum.

In future we are trying to reduce the problems of that algorithm and try to create another algorithm which also produces an optimal solution. In future we also create a simulator of that algorithm and implement it in our environment.

REFERENCES

[1] "Service Performance and Analysis in Cloud Computing" by Kaiqi Xiong, Harry Perros *978-0-7695-3708-5/09 $25.00 © 2009 IEEE page- 693-700*

[2] "Virtual Infrastructure Management in Private and Hybrid Clouds" by Borja Sotomayor, Rubén S. Montero and Ignacio M. Llorente, Ian *Foster 1089-7801/09/$26.00 © 2009 IEEE*

[3] "Research on Distributed Architecture Based on SOA" by Hongqi Li, Zhuang *Wu 978-0-7695-3522-7/09 $25.00 © 2009 IEEE 670-674*

[4] "A Berkeley View of Cloud computing". M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, M. Zaharia. Above the Clouds: Technical Report No. *UCB/EECS-2009-28, University of California at Berkley, USA, Feb. 10, 2009*

[5] "Handbook of Genetic Algorithms", Davis, L. *1991 Van,Nostrand Reinhold.*

[6] "An Approach to a Cloud Computing Network" by Francesco Maria Aymerich, Gianni Fenu1, Simone Surcis *978-1-4244-2624-9/08/$25.00 ©2008 IEEE 113 page 113-118*

[7] "Cloud Computing and Services Platform Construction of Telecom Operator" by Xu Lei, Xin Zhe, Ma Shaowu, Tang Xiongyan. *Broadband Network & Multimedia Technology, 2009. IC-BNMT '09. 2nd IEEE International Conference on Digital Object Identifier, pp. 864 – 867.*

[8] "Service Performance and Analysis in Cloud Computing", Kaiqi Xiong and Harry Perros, *2009 Congress on Services –I*

[9] "An Optimistic Differentiated Service Job Scheduling System for Cloud Computing Service Users and Providers" by Luqun *Li 2009 Third International Conference on Multimedia and Ubiquitous Engineering page-295-299*

[10] "Multiprocessor Tasks with Genetic Algorithms", M Golub, S Kasapovic *Scheduling - Applied Informatics- Proceedings- 2002.*

[11] "Parallel Genetic Algorithms to Find Near Optimal Schedules for Tasks on Multiprocessor Architectures, Communicating Process Architectures", Moore, M., *IOS Press, 2001, pgs, 27-36*

[12] "Process Scheduling with Genetic Algorithms", Pai-Chou Wang, W. Korfhage, Proceedings *of the 7th IEEE Symposium on Parallel and Distributed Processing, Page: 638, ISBN: 0-8186- 7195-5, October 2005, IEEE Computer Society Washington, DC, USA.*