International Journal of Computer Applications Technology and Research
Volume 6–Issue 4, 194-198, 2017, ISSN:-2319–8656

Cost-Efficient Task Scheduling with Ant Colony Algorithm for Executing Large Programs In Cloud Computing

Fatemeh Imani
Department of Computer Engineering, Ardabil Branch, Islamic Azad University, Ardabil, Iran

Shiva Razzaghzadeh
Department of Computer Engineering, Ardabil Branch, Islamic Azad University, Ardabil, Iran

Masoud Bekravi
Department of Computer Engineering, Ardabil Branch, Islamic Azad University, Ardabil, Iran

Abstract: The aim of cloud computing is to share a large number of resources and pieces of equipment to compute and store knowledge and information for great scientific sources. Therefore, the scheduling algorithm is regarded as one of the most important challenges and problems in the cloud. To solve the task scheduling problem in this study, the ant colony optimization (ACO) algorithm was adapted from social theories with a fair and accurate resource allocation approach based on machine performance and capacity. This study was intended to decrease the runtime and executive costs. It was also meant to optimize the use of machines and reduce their idle time. Finally, the proposed method was compared with Berger and greedy algorithms. The simulation results indicate that the proposed algorithm reduced the makespan and executive cost when tasks were added. It also increased fairness and load balancing. Moreover, it made the optimal use of machines possible and increased user satisfaction. According to evaluations, the proposed algorithm improved the makespan by 80%.

Keywords: cloud computing; formatting; task scheduling; makespan; load balancing

1. INTRODUCTION

With the development of information technology, it is necessary to do computing tasks everywhere at every time. It is also essential that people be able to perform their heavy computing tasks through some services without having expensive hardware and software requirements. Cloud computing is the latest solution provided by technology for these needs. The National Institute of Standards and Technology (NIST) defines cloud computing as a model to provide easy access based on user demand through the network for a group of modifiable and configurable computing resources such as networks, servers, storage space, applications and services. This access should be able to offer quickly-provided or free services without needing resource management or direct intervention. In this definition, the cloud can be described with five essential features including virtual computing resource sharing, WAN access, quick flexibility, requested services (based on order or demand), and measured services. The cloud environment is based on requests, and users can increase or decrease the use of resources. In other words, the results are related to usage in the cloud environment. Sharing the usable computing power among some tenants can improve the productivity rate because servers are not idle for nothing in this method. One reason is that computers are used more because cloud computing customers do not need to calculate and determine the maximum load [1]. Virtual machine scheduling problem has been investigated in many studies conducted on cloud computing environments. The main aim of scheduling in the cloud is to shorten the makespan, increase the system throughput, and establish load balancing in resource [2]. Running a program can be seen as the execution of different tasks in it. Different tasks can be executed simultaneously with several virtual machines. In this thesis, ACO was used with a fair and accurate resource allocation approach based on machine performance. As the number of natural resources should be proportionate to the performances of ants, the tasks performed by resources should be proportionate to the tasks assigned to them. An appropriate path for resource allocation is very important to perform tasks. Therefore, machines can process tasks at the maximum power and on the shortest path. As a result, resources are used optimally. In the cloud environment, providers want to make the most of their resources, and users want to minimize their costs [4]. However, they want to achieve their intended performance. The appropriate and optimal use of resources such as memory, processor and bandwidth is a challenge; therefore, the quality of task scheduling is regarded an important problem which has a great effect on the performance of cloud service providers. All of the scheduling algorithms are intended to minimize the makespan [4]. None of the previous studies dealt with task distribution in a cloud environment by considering the costs and optimal use of machines and increasing their idle times. Considering task scheduling algorithms, the ACO was adapted from social theories in this thesis to optimize the use of machines, reduce machine idle time, and decrease the makespan. It was also used to minimize executive costs by modifying the idle time (vms) to perform insensitive tasks.

2. LITERATURE

In this section, previous research works on network processing are reviewed. First, preliminary methods such as Dynamic Level Scheduling are described, then the most recent methods are reviewed.

2.1 Dynamic Level Scheduling

For this purpose, a specific model has been proposed. The main aim of this method is to decrease the processing time. In network processing environments, other scheduling algorithms do not emphasize the subtasks of an application which is run in the computing host or the virtual organization. The main aim is to perform scheduling in a way that all the input applications can use the available throughput. In the paper presented by Cathody and Caratasa, the heuristic technique was added to the abovementioned method to increase the system efficiency [5].

2.2 Allocating the Fast Process to the Largest Task

The FPLTF scheduling algorithm (Xoa et al.) determine the tasks based on the available resources in the system [6]. This method depends on speed of processor and resources and the size of tasks. In this method, the largest task is allocated to the fastest resource. If there are many large tasks, this method will not be efficient enough. The dynamic FPLTF (Chang et al.) algorithm was developed with respect to the static FPLTF algorithm. In this method, the highest priority is allocated to the largest task. It is also necessary to estimate the data which are required for processing [7].

2.3 WQR (Queue with Repeat)

This method is based on WQ. In this method, faster processors are allocated to large tasks (Young et al.) by using the FCFS and random scheduling methods. WQR iterates the tasks to transfer them to available resources. The iteration of tasks can be selected by the user. When one of these tasks is finished, the scheduling algorithm stops the iteration of other tasks. One of the problems of this method is that it spends too much time allocating resources to the iteration operations.

2.4 Balanced Ant Colony Optimization (BACO)

The main idea of this method is taken from ACO (Xoa et al.), and it is mainly intended to decrease the processing time and load balancing of each resource. This method changes the density of pheromone based on the positions of resources, something which can be possible by updating the pheromone locally and globally. In this method, the makespans are shortened at the same time as the system is kept in balance. In the architecture of this network process scheduling, there are four components: portal, information server, task scheduling algorithm, and resources required for processing. The portal is used as an interface for users [6].

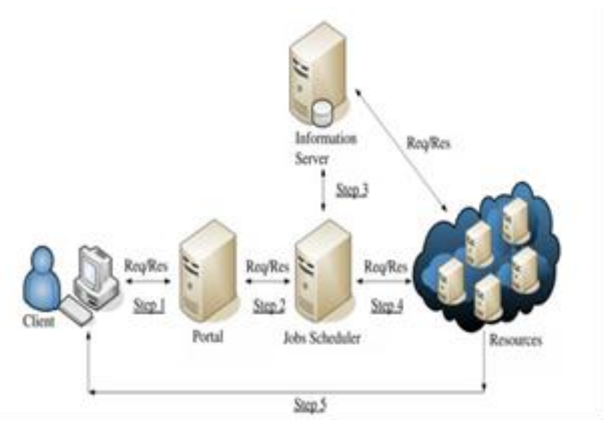

Figure 1. Structure of the system

2.5 Load Balancing

Load evaluation is usually mechanized for the continuity of a service when one or some components of the system fail. The components are constantly monitored. When one component does not respond, the load balancer comes up and prevents the traffic from being sent to it. With an appropriate load evaluation where resources are used, problems can usually be mitigated. Not only does this decrease costs and creates green computing, but it also keeps the pressure low on unique circuits whose lifetime will be potentially elongated. In fact, it can be stated that the load balancing is meant to find an appropriate map of tasks on the available processors in the system in a way that each processor runs an equal number of tasks until the total makespan is minimized as much as possible.

2.6 The Importance of Load Balancing

With load balancing, the load can be balanced through the dynamic transfer of local tasks from one machine to another one in a remote node or a machine which is used less often. This solution maximizes user satisfaction, minimizes response time, increases the exploitation of resources, increases the failure times and improves the system efficiency. Load balancing is also needed to achieve green computing in the clouds [8].

3. OPTIMIZED ACO

Cloud computing is the extended version of network computing which is done in a parallel and distributed way. It is also a new model for business computing. Compared with network computing, the new features of cloud computing include heterogeneous resources distributed and dispersed in large scale to include the datacenter. Moreover, the virtualization technology creates latent heterogeneous resources in cloud computing. Network computing is generally used in scientific computing to solve the limited domain problem. Cloud computing provides a user-oriented plan which offers various services to meet the needs of users. In cloud computing, resources are converted into virtual resources by using the virtualization packaging technology. This makes the resource allocation and interaction process be different from user tasks and network computing [9,10].

3.1 Designing the Optimized ACO

To design the optimized ACO, user tasks are allocated to resources which are the same as the machine output so that the machine can do the processing with full power, and there should not be any loads on resources. This increases the efficiency of resources. Now tasks should be classified and prioritized for the fair allocation of resources. This action was not possible in the normal ACO. Users, resource providers and the scheduler system are intended in cloud computing. The main part of scheduling computations includes user tasks and resources so that the fair distribution of resource allocation can be possible in cloud computing by using the optimized algorithm. The scheduling algorithm is optimized includes two main steps. Figure 2 shows the architecture of the optimized algorithm.

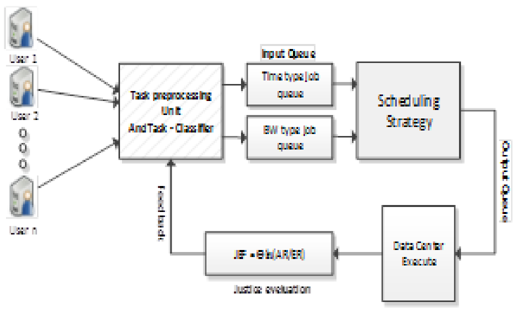

Figure 2. Architecture of the optimized algorithm

3.2 The Strategy of the Proposed Algorithm (Optimized Scheduling)

Figure 2 shows main mechanism of the algorithm in this thesis. It uses the local optimal method to allocate resources to a group of tasks and virtual machines. Now the algorithm is briefly described.

1. The tasks sent by the user are classified and prioritized with respect to the quality parameters in the computing unit.
2. Given the type of tasks, two lists are created. One of them is meant for processing tasks (runtime), and the other one is meant for the tasks needing bandwidth.
3. A group of virtual machines named VMlist is given to the system based on their normalized performances.
4. Given the number of virtual machine processors and the expected waiting time of each task, the virtual machine is selected.
5. Given the real bandwidth of virtual machine and the expected bandwidth of tasks, the desired machine is selected.
6. If tasks are equal to the performances of resources, they are assigned to them. In other words, fairness is considered in the allocation of tasks.
7. Finally, the virtual machine is freed after task processing is done.

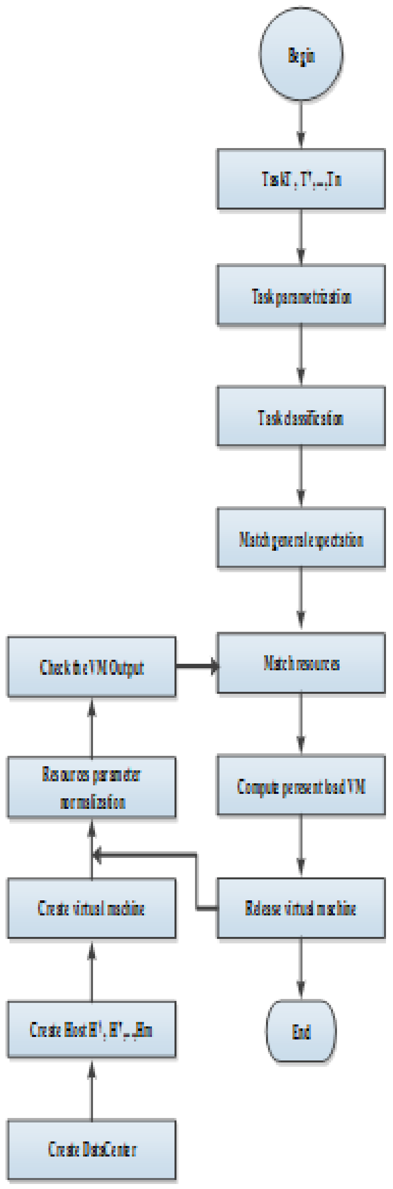

Figure 3. ACO algorithm

4. SIMULATION ENVIRONMENT

The experiments were implemented in the Cloud simulator using library functions including CloudSim, SimJava, and GridSim. This test is run in the CloudSim environment, and the application is run at the user layer code [10,11].

4.1 Evaluation Criteria

In the next sections, three tests are conducted in the form of different parameters such as makespan, the number of processors, bandwidth, and user satisfaction. Then the optimized algorithm is compared with different algorithms. In this evaluation, the optimized ACO algorithm is compared with other scheduling algorithms based on Berger's model and greedy model. In all the experiments, virtual machine parameters including machine ID, the number of processors, available memory, and bandwidth in Table (2-4) were used along with task parameters such as task ID, class ID, length, file size, output size, the expected time, and the expected bandwidth according to Table (1-4). Then the proposed algorithm is compared with other algorithms [12, 13].

Table 1. Jobs Parameter

Task Id	Class type	Length	File_size	Output_size	Expectation time	Expectation BW
0	**1**	**4000**	**2500**	**500**	**400**	**-**
1	**1**	**3000**	**2000**	**400**	**200**	**-**
2	**1**	**2000**	**800**	**300**	**150**	**-**
3	**1**	**5000**	**5000**	**2000**	**500**	**-**
4	**2**	**2000**	**800**	**300**	**-**	**2000**
5	**2**	**3000**	**2000**	**400**	**-**	**3000**
6	**2**	**800**	**300**	**300**	**-**	**1200**
7	**2**	**2500**	**1000**	**500**	**-**	**2000**

Table 2. Vm Parameter

VM ID	CPU	Memory	Bandwidth
0	4	2048	1200
1	2	1024	3000
2	2	1024	1000
3	1	512	1200

4.2 Comparing the Results of Simulating the Proposed ACO Method with Berger and Greedy

4.2.1 Makespan in Different Algorithm

In this experiment, the optimized algorithm was compared with the scheduling Berger algorithm and greedy base by considering the makespan. The tests were conducted in a highly heterogeneous environment. Figure 4 and Table 3 show the runtime of tasks in an analytic comparison.

Table 3. Vm Parameter

Task Id	0	1	2	3	4	5	6	7
ACo-O	400	500	200	1000	200	600	80	250
Berger	400	500	200	520	250	520	140	500
Greedy Base	400	500	200	600	200	300	90	500

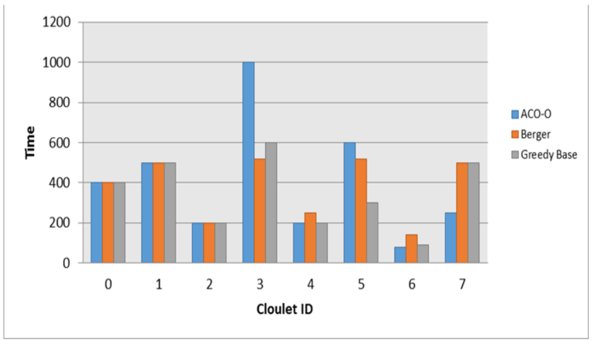

Figure 3. Comparing completing time

4.2.2 Comparing the Number of Processors in the Machines Allocated to Tasks

In this section, the proposed algorithm is compared to different scheduling algorithms by using improvement strategies with respect to quality. The tests are run in this range. The main research goal and necessity is to select resources and to achieve the best time as well as the appropriate cost. According to the datasets shown in Tables 2 and 3, the number of processors of machines used in the Berger and Greedy Base algorithms can be compared with the proposed algorithm.

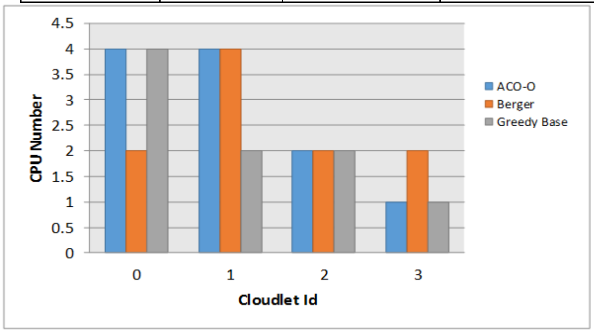

Figure 4. Comparing the Number of Processors in the Machines Allocated to Tasks

5. SUGGESTIONS

More studies can be conducted on resource scheduling. Regarding future works, some suggestions are made as follows:

• A fuzzy neural network of the service quality of tasks and resources can be used with an ACO approach in the future.

• The methods based on the genetic algorithm can be used to make significant improvements in the proposed algorithm because they are very helpful in optimization problems.

6. REFERENCES

[1] Danielson K. 2008. Distinguishing Cloud Computing from Utility Computing . Available from http:/ / www. ebizq. net/ blogs/ saasweek/ 2008/ 03/

distinguishing_cloud_computing/ . [Accessed 26 March 2008]

[2] Sran N. and Kaur N. 2013. Comparative Analysis of Existing Load Balancing Techniques in Cloud Computing, International Journal of Engineering Science Invention, Vol.2, Issue 1.

[3] Hamo A. and Saeed A. 2013. Towards a Reference Model for Surveying a Load Balancing, International Journal of Computer Science and Network Security , Vol. 13.

[4] Peng L. 2009. the definition of cloud computing and characteristics. http:// www.china cloud .cn/ [Accessed 25 February 2009]

[5] Gkoutioudi, H. D. Karatza, "Task cluster scheduling in a grid system", Simulation Modeling Practice and Theory, Vol. 18, pp. 1242-1252, 2010.

[6] Chang and et al, "An ant algorithm for balanced job scheduling in grids", Future Generation Computer Systems, Vol. 25, pp. 20-27, 2009.

[7] Goa Y. and et al, "Adaptive grid job scheduling with genetic algorithms", Future Generation Computer Systems, Vol. 21, pp. 151-161, 2005.

[8] Begum S and Prashanth C S R. 2013. Review of Load Balancing in Cloud Computing. International Journal of Computer Science Issues , Vol.10, Issue 1.

[9] Khetan A, Bhushan V and Chand Gupta S. 2013.A Novel Survey on Load Balancing in Cloud Computing International Journal of Engineering Research & Technology

[10] Sundararajan S, Raj B. 2011. IBM Center for High Performance On Demand Solutions. IBM Cloud Computing White Paper http://www.ibm.com/developerworks/Websphere /zones/hipods/.[Accessed2011].

[11] Lia J, Fenga L, Fang S. 2014. An Greedy-Based Job Scheduling Algorithm in Cloud Computing. JOURNAL OF SOFTWARE, VOL. 9, NO. 4, APRIL.

[12] Buyya R, Ranjan R and Rodrigo N. 2009. Calheiros, Modeling and simulation of scalable cloud c computing environments and the CloudSim Toolkit: challenges and opportunities. In: Proceedings of the seventh high performance computing and simulation conference (HPCS 2009, ISBN:978-1- 4244-49071), Leipzig, Germany. New York, USA: IEEE Press; June 21-24.

[13] Baomin X U C Z . 2011. Job scheduling algorithm based on berger model in cloud environment. Advances in Engineering Software, vol. 42, pp. 419–425.