Cloud Task scheduling based on Load Balancing Ant Colony Optimization

Kun Li, Gaochao Xu, Guangyu Zhao, Yushuang Dong, Dan Wang
College of Computer Science and Technology
Jilin University
ChangChun, China
e-mail: xugc@jlu.edu.cn

***Abstract*—The cloud computing is the development of distributed computing, parallel computing and grid computing, or defined as the commercial implementation of these computer science concepts. One of the fundamental issues in this environment is related to task scheduling. Cloud task scheduling is an *NP*-hard optimization problem, and many meta-heuristic algorithms have been proposed to solve it. A good task scheduler should adapt its scheduling strategy to the changing environment and the types of tasks. This paper proposes a cloud task scheduling policy based on Load Balancing Ant Colony Optimization (LBACO) algorithm. The main contribution of our work is to balance the entire system load while trying to minimizing the makespan of a given tasks set. The new scheduling strategy was simulated using the CloudSim toolkit package. Experiments results showed the proposed LBACO algorithm outperformed FCFS (First Come First Serve) and the basic ACO (Ant Colony Optimization).**

Keywords-task scheduling; cloud computing; Load Balancing; Ant Colony Optimization; CloudSim;

I. INTRODUCTION

Cloud computing is experiencing a rapid development both in academia and industry; it is promoted by the business rather than academic which determines its focus on user applications. This technology aims to offer distributed, virtualized, and elastic resources as utilities to end users. It has the potential to support full realization of 'computing as a utility' in the near future[1]. With the support of virtualization technology[2, 3], cloud platforms enable enterprises to lease computing power in the form of virtual machines to users. Because these users may use hundreds of thousands of virtual machines (VMs)[4], it is difficult to manually assign tasks to computing resources in clouds[5, 6]. So we need an efficient algorithm for task scheduling in the cloud environment[7].

A good task scheduler should adapt its scheduling strategy to the changing environment and the types of tasks. Therefore, a dynamic task scheduling algorithm, such as Ant Colony Optimization (ACO)[8, 9], is appropriate for clouds.

ACO algorithm is a random search algorithm, like other evolutionary algorithms[10]. It imitates the behavior of real ant colonies in nature to search for food and to connect to each other by pheromone laid on paths traveled. Many researchers used ACO to solve NP-hard problems such as traveling salesman problem[10], graph coloring problem[11], vehicle routing problem[12], and so on.

In this paper, we proposed a Load Balancing Ant Colony Optimization (LBACO) algorithm to find the optimal resource allocation for each task in the dynamic cloud system. Not only does it minimize the makespan of a given tasks set but it also adapts to the dynamic cloud computing system and balance the entire system load. Then, this new scheduling strategy was simulated using the CloudSim version 2.1 toolkit package[13,14]. Experiments results showed the proposed LBACO algorithm satisfies expectation. The experiment considers:

- First Come First Served (FCFS)
- Ant Colony Optimization (ACO)
- Load Balancing Ant Colony Optimization (LBACO)

The rest of paper is organized as follows. Section Ⅱ introduces the CloudSim toolkit. Section Ⅲ introduces the basic Ant Colony algorithm. Section Ⅳ details the proposed LBACO algorithm. Section Ⅴ presents the simulation results. Finally, Section Ⅵ concludes this paper.

II. CLOUDSIM TOOLKIT

A. Characteristics of Cloud Simulator

Several Grid simulators, such as GridSim, SimGrid, and GangSim have been developed. These toolkits are capable of modeling and simulating the Grid application in a distributed environment, but none of these are able to support the infrastructure and application-level requirements arising from Cloud computing paradigm, such as modeling of on-demand virtualization enabled resource[15]. Hence, Cloud infrastructure modeling and simulation toolkits must support for real-time trading of services between customers and providers. Among the currently developed simulators, only GridSim offers support for economic-driven resource management and application scheduling simulation. So CloudSim framework is built based on GridSim toolkit[16].

CloudSim allows simulation of scenarios modeling IaaS (Infrastructure as a Service), PaaS (Platform as a Service), and SaaS (Software as a Service), because it offers basic components such as Hosts, Virtual Machines, and applications that model the three types of services.

CloudSim offers the following novel features: (i) support model and instantiation of large scale Cloud computing

978-0-7695-4472-4/11 $26.00 © 2011 IEEE
DOI 10.1109/ChinaGrid.2011.17

infrastructure, including data centers on a single physical computing node and java virtual machine; (ii) a self-contained platform for modeling data centers, service brokers, scheduling, and allocations policies; (iii) availability of virtualization engine, which aids in creation and management of multiple, independent, and co-hosted virtualized services on a data center node; (iv) flexibility to switch between space-shared and time-shared allocation of processing cores to virtualized services[14].

B. CloudSim Work Style

CloudSim work style is shown as Fig. 1.

In general, the tasks from different users are relatively independent; we consider there are m users, as $User_1$, $User_2$, ……$User_m$, n independent tasks, as T_1, T_2 …T_n, n VMs, as VM_1, VM_2 ……VM_n and p datacenters, as $Datacenter_1$, $Datacenter_2$ … $Datacenter_p$.

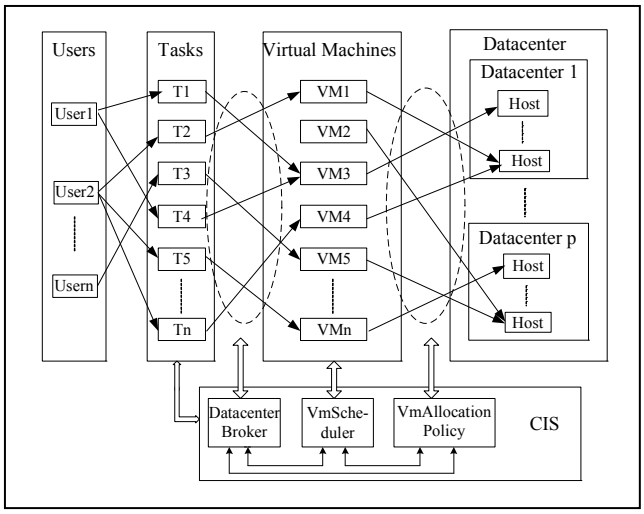

Figure 1. CloudSim Work Style

CIS: The CIS (Cloud Information Service) provides database level match-making services; it maps user requests to suitable cloud providers. CIS and Data-CenterBroker of CloudSim realized resource discovery and information interaction, it is the core of simulated scheduling[14, 15, 16].

DatacenterBroker: This class models a broker, which is responsible for mediating between users and service providers depending on users' QoS requirements. And the broker deploys service tasks across clouds. User-developed scheduling algorithms are implemented in DataCenterBroker method. Hence, the researchers and system developers must extend this class[14, 15, 16].

VmScheduler: This is an abstract class implemented by a Host component; it represents the policies (space-shared, time-shared) required for allocating processing power to VMs. The functionalities of this class can easily be overridden to accommodate specific processor sharing policies[14, 15, 16].

VmAllocationPolicy: This abstract class represents the provisioning policy that a VM Monitor utilizes for allocating VMs to Hosts. The chief functionality of the VmAllocationPolicy is to select available host in a datacenter, which meets the memory, storage, and availability requirement for a VM deployment[14, 15, 16].

It can be seen from Fig. 1, in the cloud computing environment, we carry out the task scheduling in the virtual machines. But in the grid computing environment, the task scheduling is carried out in the idle hardware resources directly.

C. Communication among Entities

Figure 2 depicts the flow of communication among core CloudSim entities. In the beginning of the simulation, each Datacenter entity registers itself with the CIS (Cloud Information Service) Registry, and DatacenterBroker manages information interaction among entities[14, 15, 16].

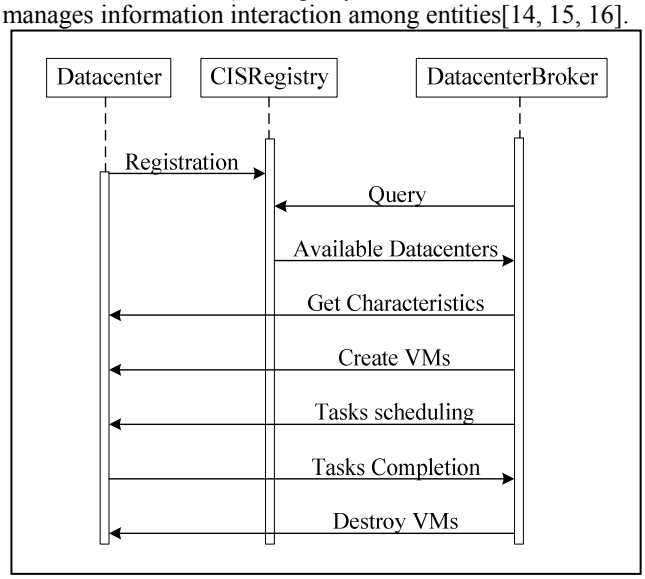

Figure 2. Simulation data flow.

III. THE BASIC ANT COLONY ALGORITHM

Dorigo M. introduced the ant algorithm based on the behavior of real ants in 1996[2], it is a new heuristic algorithm for the solution of combinatorial optimization problems. Investigations show that: Ant has the ability of finding an optimal path from nest to food[17,18,19].On the way of ants moving, they lay some pheromone on the ground; while an isolated ant encounter a previously laid trail, this ant can detect it and decide with high probability to follow it. Hence, the trail is reinforced with its own pheromone. The probability of ant chooses a way is proportion to the concentration of a way's pheromone. To a way, the more ants choose, the way has denser pheromone, and the denser pheromone attracts more ants. Through this positive feedback mechanism, ant can find an optimal way finally[20, 21, 22].

At time zero, ants are positioned on different towns, the initial values τ_{ij} (0) for trail intensity are set on edge (i, j). The first element of each ant's tabu list is set to be equal to its starting town[24, 25]. Thereafter the k-ant moves from town i to town j with a probability that is defined as:

$$p_{ij}^{k}(t)=\begin{cases}\dfrac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^{\beta}}{\sum_{k\in allowed_k}[\tau_{ik}(t)]^{\alpha}[\eta_{ik}(t)]^{\beta}} & if \quad j\in allowed_k \\ 0 & otherwise\end{cases} \quad (1)$$

Where $allowed_k = \{N\text{-}tabu_k\}$, $tabu_k$ is the tabu list of k-th ant, $\tau_{ij}(t)$ is the pheromone value on edge (i, j), η_{ij} is the value of the heuristic value, and $\eta_{ij}(t) = 1/d_{ij}$. Where d_{ij} is the distance between node i and node j. α, β are two parameters that control the relative weight of the pheromone trail and heuristic value. Finally the most optimal and effective path is selected and globally updated.

Figure 3 depicts programming steps of the basic ACO.

Procedure ACO

```
begin
    Initialize the pheromone
    while (stopping criterion not satisfied) do
        Position each ant in a starting VM
        while (stopping when every ant has
              build a solution) do
            for each ant do
                Chose VM for next task by
                pheromone trail intensity
            end for
        end while
        Update the pheromone
    end while
end
```

Figure 3. Programming steps of the basic ACO

IV. THE PROPOSED LBACO ALGORITHM

We utilize the characteristics of ant algorithms mentioned above to schedule task[26, 27]. We can carry out new task scheduling depending on the result in the past task scheduling. It is very helpful in the cloud environment.

In contrast to other ACO algorithm, the LBACO algorithm inherits the basic ideas from ACO algorithm to decrease the computation time of tasks executing, it also considers the loading of each VM. We can carry out new task scheduling depending on the result in the past task scheduling. It is very helpful in the cloud environment.

A. Initialize pheromone of VM_j

At the beginning, ants are distributed on VMs randomly, and then it will initialize the VM_j pheromone value based on:

$$\tau_j(0) = pe\_num_j \times pe\_mips_j + vm\_bw_j \quad (2)$$

Where pe_num_j is the number of VM_j processor, pe_mips_j is the MIPS (Million Instructions Per Second) of each processor of VM_j and the parameter VM_bw_j that is related to the communication bandwidth ability of the VM_j.

B. The rule of choosing VM for next task

The k-ant chooses VM_j for next task with a probability that is defined as:

$$p_j^k(t)=\begin{cases}\dfrac{[\tau_j(t)]^{\alpha}[EV_j]^{\beta}[LB_j]^{\gamma}}{\sum[\tau_k(t)]^{\alpha}[EV_k]^{\beta}[LB_k]^{\gamma}} & if \quad j\in 1\cdots n \\ 0 & otherwise\end{cases} \quad (3)$$

Where

- $\tau_j(t)$ is the VM_j pheromone value at time t.
- EV_j is the computing capacity of VM_j, it is defined as follows:

$$EV_j = pe\_num_j \times pe\_mips_j + vm\_bw_j \quad (4)$$

  Where pe_num_j is the number of VM_j processor, pe_mips_j is the MIPS of each processor of VM_j and the parameter VM_bw_j that is related to the communication bandwidth ability of the VM_j.
- LB_j is the load balancing factor of VM_j, to minimize the degree of imbalance, which is defined as follows:

$$LB_j = 1 - \frac{res_j - lastAver\_res}{res_j + lastAver\_res} \quad (5)$$

  Where *lastAver_res* is the average execution time of the virtual machines in the last iteration of the optimal path, and res_j is the expected execution time of the task in the VM_j, which is defined as follows:

$$res_j = \frac{total\_tasklength}{EV_j} + \frac{InputFilesize}{vm\_bw_j} \quad (6)$$

  Where *total_tasklength* is the total length of the tasks that have been submitted to VM_j, and *InputFilesize* is the length of the task before execution.
- α, β and γ are three parameters that control the relative weight of the pheromone trail, the computing capacity of VMs and the load balancing factor of VMs.

Once some VMs are loading heavy, it becomes a bottleneck in the cloud and it influences the makespan of a given tasks set. Therefore we define the load balancing factor LB_j in the ant algorithm to improve the load balancing capability, and the bigger LB_j of VM_j should be chosen with high probability, that means the comprehensive ability of VM_j is power now, and then it is high desirable.

C. Phenomenon Updating

Let $\tau_j(t)$ be the intensity of VM_j pheromone at time t. The pheromone update is given by (7):

$$\tau_j(t+1) = (1-\rho)\times\tau_j(t) + \Delta\tau_j \qquad (7)$$

Where $\rho \in (0, 1]$ is the pheromone trail decay coefficient. The greater the value of ρ is, the less the impact of past solution is. The value of $\Delta\tau_j$ is defined as follows:

When an ant completes its tour, the local pheromone updating is applied on the visited VMs, and the value of $\Delta\tau_j$ is given by (8).

$$\Delta\tau_j = 1/T_{ik} \qquad (8)$$

Where T_{ik} is the shortest path length that searched by k-ant at i-th iteration.

When an ant completes its tour, if it finds the current optimal solution, it can lay a larger intensity of the pheromone on its tour[20, 21], and the global pheromone updating is applied on the visited VMs, and the value of $\Delta\tau_j$ is given by (9).

$$\Delta\tau_j = D/T_{op} \qquad (9)$$

Where T_{op} is the current optimal solution, and D is the encouragement coefficient.

D. Programming Steps of the proposed LBACO

The programming steps of the proposed LBACO algorithm in searching for the minimum makespan path can be described as follows:

- Step1 Initialize the pheromones of all VMs.
- Step2 Place all ants at the starting VMs randomly.
- Step3 Every ant chooses the VM for the next task according to formula (3) ~ (6).
- Step4 When an ant completes its tour, update the pheromone according to formula (7) ~ (9).
- Step5 If all the ants end their trip, continue to Step6; otherwise, repeat Step3
- Step6 Nc = Nc + 1, calculate the makespan of each ant and reserve the current optimal solution.
- Step7 Judge if it satisfies the iterative condition Nc > Ncmax, If it satisfies, end the iteration and output the best solution, else return to Step2 until satisfy the iterative condition.

The flowchart in Fig. 4 describes the above-mentioned procedure.

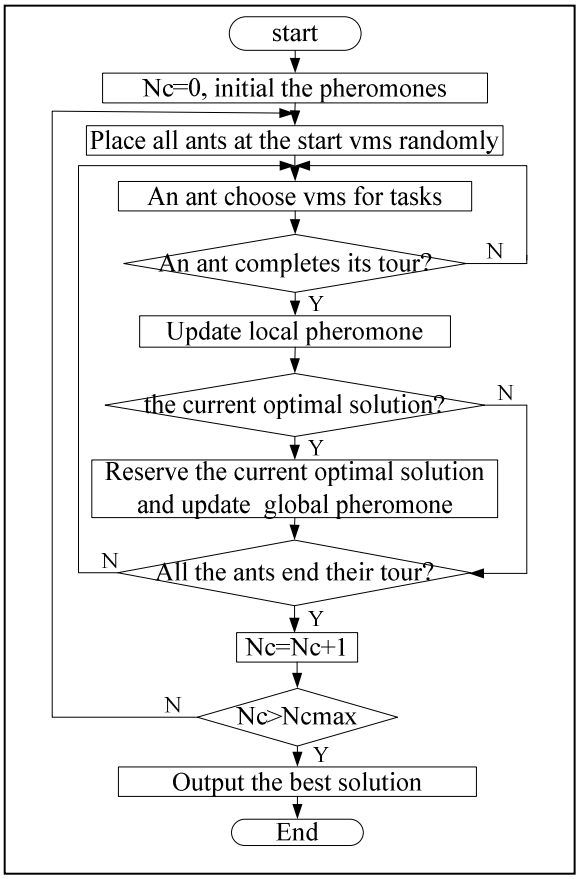

Figure 4. Flowchart of LBACO

V. SIMULATIONS.

The experiment is implemented on the CloudSim platform. The scheduling algorithms of the experiment include the LBACO, the basic ACO[6] and FCFS (First Come First Service).

A. Assumptions

Adopting the application model introduced in [15], we assume that

- Tasks are mutually independent, i.e., there is no precedence constraint between tasks.
- Tasks are computationally intensive.
- Tasks are not preemptive and they cannot be interrupted or moved to another processor during their execution.

The scheduling problem aims to minimize the total execution time of tasks as well as to achieve a well-balanced load across all VMs in Cloud. That is, there are two factors considered here. One is the minimization of the tasks completion time. The other is to distribute workload evenly among virtual machines[28, 29].

B. Define the degree of imbalance

Moreover, we define the degree of imbalance (DI for short) to measure the imbalance among VMs, which is defined as follows:

$$T_i = \frac{total\_tasklength_j}{pe\_num_j \times pe\_mips_j} \quad (10)$$

Where *total_tasklength* is the total length of tasks which are submitted to the VM_j, pe_num_j is the number of processor of VM_j and pe_mips_j is the MIPS of each processor of VM_j.

$$DI = \frac{T_{\max} - T_{\min}}{T_{avg}} \quad (11)$$

Where T_{max} and T_{min} are the maximum and minimum T_i among all VMs, T_{avg} is the average T_i of VMs. Thus, the scheduling problem also aims to minimize the degree of imbalance. The consideration of *DI* during the allocation would help to avoid unbalanced workload of VMs.

C. Implementation environment

The experiment is implemented with 10 Datacenters and 100-500 tasks under the simulation platform. The resource situation is shown in Table 1. The computation workload of the task is from 5000 MI (Million Instruction) to 15000 MI, and the manager type of 10 datacenters both have space_shared and time_shared policy for VMs, but, to the manager type of 50 VMs, we only set time_shared for tasks. The parameters' setting of cloud simulator is shown in Table 2.

TABLE 1. PARAMETERS SETTING OF CLOUD SIMULATOR

Type	Parameters	Value
Datacenter	Number of Datacenter	10
	Number of Host	2-6
	Type of Manager	Space_shared Time_shared
	Datacenter Cost	1-15
Virtual Machine (VM)	Total number of VMs	50
	MIPS of PE (processing element)	250-2000 (MIPS)
	Number of PE per VM	2-8
	VM memory(RAM)	512-2048(MB)
	Bandwidth	500-1000 bit
	Type of Manager	Time_shared
Task	Total number of task	100-500
	Length of task	5000-15000MI
	Number of PEs requirement	1-4

D. Parameters Setting of the basic ACO and LBACO

The performance evaluation of our proposed LBACO algorithm and the comparison study with other algorithms for task scheduling have been implemented on the CloudSim platform[13, 14], the CloudSim is developed to support simulation of heterogeneous cloud resources and application models. The parameters' setting is shown in Table 2.

TABLE 2. PARAMETERS OF LBACO

Parameters	values
Number of tasks	100-500
Number of ants in colony	8
Number of iterations	50
ρ	0.01
α	3
β	2
γ	8

E. Experimental results

We compared our LBACO algorithm with the First-Come-First-Served (FCFS) and the basic Ant Colony System (ACO)[10]. The FCFS algorithm aims to find the earliest completion time of each task individually. The basic ACO algorithm aims to minimize the makespan of a given set of tasks. The LBACO algorithm chooses optimal resources to perform tasks according to resources status and the size of given task in the Cloud environment. Not only does it minimize the makespan of a given set of tasks but it also balances the entire system load.

In the following experiments, we compared the average makespan of the basic ACO and LBACO algorithm with different iterations; we also compared the average makespan of 100-500 tasks set, and the average degree of imbalance (DI) of each algorithm in the following experiments.

The average makespan of the basic ACO and LBACO algorithm with different iterations is shown in Fig. 5. In this experiment, we used 300 tasks set to compare the average performance of the basic ACO and the LBACO algorithm, and we recorded the makespan using the time in the CloudSim (ms).

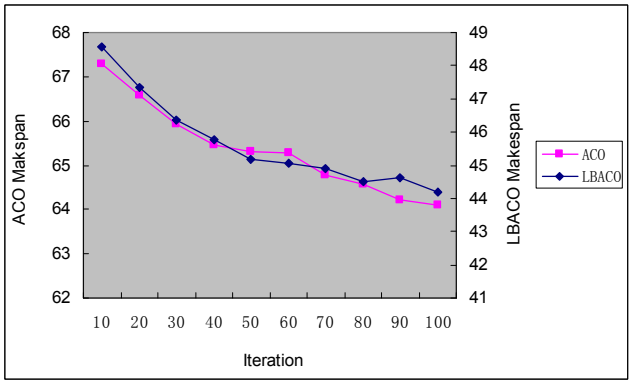

Figure 5. The average makespan of 300 tasks set

Fig. 5 shows that the average makespan of the basic ACO and LBACO algorithm reduced roughly with the number of iterations increased. But for the basic ACO and LBACO algorithm, this change became slow after 50 iterations. Hence, we used 50 iterations for other experiments in this paper.

The average makespan of each algorithm with the number of tasks varying from 100 to 500 is shown in Fig. 6. In this experiment, we also use the time in the CloudSim (ms) to record the makespan.

At last the average degree of imbalance (DI) of each algorithm with the number of tasks varying from 100 to 500 is shown in Fig. 7.

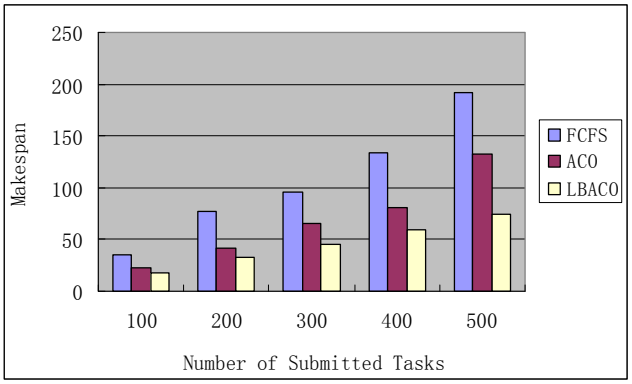

Figure 6. The average makespan of 100-500 tasks set

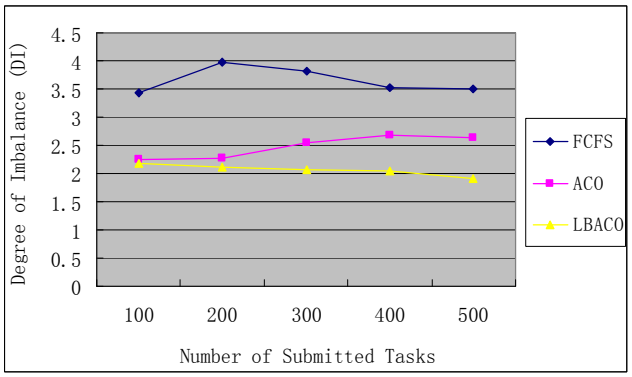

Figure 7. The average DI of each algorithn

It can be seen from the Figure 6 and Figure 7, the average performance of the LBACO algorithm is better than the FCFS algorithm and ACO algorithm. It means that the LBACO can achieve good system load balance in any situation and take less time to execute tasks. In other words, these results demonstrated the effectiveness of the LBACO algorithm.

VI. Conclusions and Future Work

In this paper we have proposed the LBACO algorithm for achieving tasks scheduling with load balancing, and we have experimentally evaluated the LBACO algorithm in applications with the number of tasks varying from 100 to 500. The experimental result shows that the LBACO balance the entire system load effectively. Weather the sizes of the tasks are the same or not, LBACO can handle all conditions, and outperforms FCFS and ACO algorithms in cloud computing environment.

As for the future work, there are two interesting points that deserve further investigation. First, in this work, we assume that all Tasks are mutually independent, i.e., there is no precedence constraint between tasks. Second, we assume that tasks are computationally intensive, which is not realistic for cloud systems. Moreover, as a future work, in order to accommodate the heterogeneous processing of the tasks, the availability vector should be extended to incorporate information about task requirements.

References

[1] A. Weiss, "Computing in the Clouds," netWorker on Cloud computing: PC functions move onto the web, vol. 11, Dec. 2007, DOI: 10.1145/1327512.1327513, pp. 16-25, 2007.

[2] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization" in SOSP '03: Proc. of 19th ACM symposium on Operating systems principles, 2003.

[3] J. Fisher-Ogden, Hardware support for efficient virtualization, http://cseweb.ucsd.edu/jfisherogden/hardwareVirt.pdf,April 2006.

[4] Fangzhe, C., Jennifer R., Ramesh, V.,"Optimal Resource Allocation in Clouds" in 2010 IEEE 3rd International Conference on Cloud Computing, pp. 189–196, 2010.

[5] F. Chang, J. Ren, and R. Viswanathan, "Optimal resource allocation for batch testing" in ICST, 2009 IEEE International Conference on Software Testing Verification and Validation, pp.91-100, 2009.

[6] F. Chang, J. Ren, and R. Viswanathan, "Optimal Resource Allocation in Clouds" in 2010 IEEE 3rd International Conference on Cloud Computing, pp.418-425, 2010.

[7] Qiyi, H., Tinglei, H., "An Optimistic Job Scheduling Strategy based on QoS for Cloud Computing" in 2010 IEEE International Conference on Intelligent Computing and Integrated Systems (ICISS), DOI: 10.1109/ICISS.2010.5655492, pp.673-675, 2010.

[8] M. Dorigo, C. Blum, "Ant colony optimization theory: A survey" in Theoretical Computer Science 344 (2–3) (2005), DOI: 10.1016/j.tcs.2005.05.020, pp.243–278, 2005.

[9] M. Dorigo, M. Birattari, T. Stutzel, "Ant colony optimization", in IEEE Computational Intelligence Magazine, DOI: 10.1109/MCI.2006.329691, pp.28-39, 2006.

[10] M. Dorigo, L.M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem", in IEEE Transactions on Evolutionary Computation (1997), DOI: 10.1109/4235.585892, pp.53–66, 1997.

[11] E. Salari, K. Eshghi, "An ACO algorithmfor graph coloring problem" in Congress on Computational Intelligence Methods and Applications, 2005, DOI: 10.1109/CIMA.2005.1662331, p. 5, 2005.

[12] Xiaoxia Zhang, Lixin Tang, "CT-ACO—hybridizing ant colony optimization with cycle transfer search for the vehicle routing problem" in Congress on Computational Intelligence Methods and Applications, 2005, DOI: 10.1109/CIMA.2005.1662313, p. 6, 2005.

[13] Buyya, R., Ranjan, R., Calheiros, R.N., "Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities" in Proceedings of the 7th High Performance Computing and Simulation (HPCS 2009) Conference, Leipzig, Germany, DOI: 10.1109/HPCSIM.2009.5192685, 2009 .

[14] Calheiros, R.N., Ranjan, R., De Rose, C.A.F., Buyya, R., " CloudSim: A Novel Framework for Modeling and Simulation of Cloud Computing Infrastructures and Services" in Technical Report, GRIDS-TR-2009-1, Grid Computing and Distributed Systems Laboratory, The University of Melbourne, Australia, 2009.

[15] Ghalem, B., Fatima Zohra, T., and Wieme, Z. "Approaches to Improve the Resources Management in the Simulator CloudSim" in ICICA 2010, LNCS 6377, DOI: 10.1007/978-3-642-16167-4_25, pp. 189–196, 2010.

[16] Bhathiya, W., Buyya, R., Ranjan, R., "CloudAnalyst: A CloudSim-based Visual Modeller for Analysing Cloud Computing Environments and Applications" in 2010 24th IEEE International Conference on Advanced Information Networking and Applications, pp.446-452, 2010.

[17] Hui, Y., Xueqin, S., Xing, L.,Minghui, W,. "An improved ant algorithm for job scheduling in Grid " in Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, DOI: 10.1109/ICMLC.2005.1527448 , pp. 2957-2961, 2005.

[18] Manpreet Singh, "GRAAA: Grid Resource Allocation Based on Ant Algorithm" in 2010 Academy Publisher DOI: 10.4304/jait.1.3.133-135, 2010.

[19] Ajay, K., Arnesh, S., Sanchit, A., and Satish, C., “An ACO Approach to Job Scheduling in Grid Environment” in Springer-Verlag Berlin Heidelberg 2010, SEMCCO 2010, LNCS 6466, DOI: 10.1007/978-3-642-17563-3_35, pp. 286–295, 2010.

[20] Li, L., Yi, Y., Lian, L., and Wanbin, S.,“Using Ant Colony Optimization for SuperScheduling in Computational Grid” in 2006 IEEE Asia-Pacific Conference on Service Computing, ISBN: 0-7695-2751-5, 2006.

[21] Liang, B., Yanli, H., Songyang, L., Weiming, Z., “Task Scheduling with Load Balancing using Multiple Ant Colonies Optimization in Grid Computing” in 2010 Sixth International Conference on Natural Computation (ICNC 2010), DIO: 10.1109/ICNC.2010.5582599, pp.2715-2719, 2010.

[22] Bing, T., Yingying, Y., Quan, L., Zude, Z, “Research on the Application of Ant Colony Algorithm in Grid Resource Scheduling” in Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08. 4th International Conference, DOI: 10.1109/WiCom.2008.1354, pp.1-4, 2008.

[23] Jin, X., Lam, A.Y.S., Li, V.O.K., “Chemical Reaction Optimization for the Grid Scheduling Problem” in Communications (ICC), 2010 IEEE International Conference, DOI: 10.1109/ICC.2010.5502406 pp.1-5, 2010.

[24] Meihong, W., Wenhua, Z., “A comparison of four popular heuristics for task scheduling problem in computational grid” in Wireless Communications Networking and Mobile Computing (WiCOM), 2010 6th International Conference, DOI: 10.1109/WICOM.2010.5600872, 2010.

[25] Ku Ruhana Ku-Mahamud, Husna Jamal Abdul Nasir, "Ant Colony Algorithm for Job Scheduling in Grid Computing" in ams, 2010 Fourth Asia International Conference on Mathematical/Analytical Modelling and Computer Simulation, pp.40-45, 2010

[26] M. Bandieramonte, A. Di Stefano and G. Morana, “Grid jobs scheduling: The Alienated Ant Algorithm solution” in Multiagent and Grid Systems – An International Journal 6 (2010), DOI: 10.3233/MGS-2010-0149, pp.225–243, 2010.

[27] Fidanova, S., and Durchova, M., “Ant Algorithm for Grid Scheduling Problem” in Lecture Notes in Computer Science, 2006, Volume 3743/2006, DOI: 10.1007/11666806_46, pp.405-412, 2006.

[28] Bagherzadeh, J., MadadyarAdeh, M., “An Improved Ant Algorithm for Grid Scheduling Problem” in Computer Conference, 2009. CSICC 2009. 14th International CSI, DOI: 10.1109/CSICC.2009.5349368, pp.323-328, 2009.

[29] Lorpunmanee, S., Sap, M.N, Abdul Hanan Abdullah, A.H., “An Ant Colony Optimization for Dynamic Job Scheduling in Grid Environment” in Proceedings of World Academy of Science, English and Technology Volume 23 august 2007, ISSN 1307-6884, 2007.