An Ant Algorithm for Cloud Task Scheduling

Medhat A. Tawfeek, Ashraf El-Sisi
Department of Computer Science
Faculty of Computers and Information
Menoufia University
{medhattaw@yahoo.com, ashrafelsisim@yahoo.com}

Arabi E. keshk, Fawzy A. Torkey
Department of Computer Science
Faculty of Computers and Information
Menoufia University
{arabikeshk@yahoo.com, torkey1951@yahoo.com}

***Abstract*—Cloud computing is a type of parallel and distributed system consisting of a collection of interconnected and virtual computers. 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. In this paper, modified ant colony optimization for cloud task scheduling is proposed. The goal of modification is to enhance the performance of the basic ant colony optimization algorithm and optimize the task execution time in view of minimizing the makespan of a given tasks set. Our approach proposes self-adapting criteria for the basic ant colony optimization control parameters. Ant colony optimization algorithm and modified algorithm have been simulated using Cloudsim toolkit package. Experimental results showed that modified ant colony optimization outperformed the ant colony optimization algorithm.**

***Keywords*— Cloud computing; task scheduling; makespan; ant colony optimization; CloudSim**

I. Introduction

Cloud computing is one of the rapidly improving technologies. Users can host different kinds of applications on the cloud ranging from simple web applications to scientific workloads [1]. Cloud platforms enable enterprises to lease computing power in the form of virtual machines. Because hundreds of thousands of virtual machines (VMs) are used, it is difficult to manually assign tasks to computing resources in clouds [2]. So we need an efficient algorithm for task scheduling in the cloud environment. Many meta-heuristic algorithms have been proposed such as ant colony optimization (ACO) algorithm which is appropriate for dynamic cloud task scheduling [3]. In this paper, a Modified Ant Colony Optimization (MACO) for cloud task scheduling is proposed. The main goal of MACO is to enhance the performance of ACO algorithm. The organization of this paper is as follows. Section 2 presents some of the related work in this direction. Section 3 describes the ACO for cloud task allocation. In section 4, the details about our modifications of ACO are presented. The implementation and simulation results are seen in section 5. Finally, section 6 concludes this paper.

II. RELATED WORK

Millions of user share cloud resources by submitting their computing task to the cloud system. Scheduling these millions of task is a challenge to cloud computing environment. Cloud service scheduling is categorized at user level and system level [2]. At user level scheduling deals with problems raised by service provision between providers and customers [4 and 5]. The system level scheduling handles resource management within datacenter [2, 6, 7 and 8]. A novel approach of heuristic-based request scheduling at each server, in each of the geographically distributed data centers, to globally minimize the penalty charged to the cloud computing system is proposed in [9]. Scheduling based genetic algorithm is proposed in [10]. Ant algorithms are one of the most popular examples of swarm intelligence systems. It has already been applied to solve a number of complex problems, such as task allocation in grid environment [11 and 12]. In this paper, cloud task scheduling based on MACO approach has been proposed for allocation of incoming batch jobs to virtual machines (VMs) by providing self-adapting criteria for the ACO control parameters to increase the performance of ACO.

III. Cloud Task Scheduling Based ACO

The basic idea of ACO is to simulate the foraging behavior of ant colonies. When an ants group tries to search for the food, they use a special kind of chemical pheromone to communicate with each other [13]. Task scheduling based ACO algorithm is used to decrease the computation time of tasks. In ACO, all ants are placed at the starting VMs randomly. During an iteration ants build solutions to the cloud scheduling problem by moving from one VM to another for next task until they complete a tour (all tasks has been allocated). Iterations are indexed by t, $1 < t < t_{max}$, where t_{max} is the maximum number of iterations allowed. The pseudo code of cloud task scheduling based on basic ACO is shown in fig. 1. The main operations of the basic ACO are initializing pheromone, choosing VM for next task and phenomenon updating as following:

A. *Initializing Pheromone*

The amount of virtual pheromone trail $\tau_{ij}(t)$ on the edge connects task i to VM j. The initial amount of pheromone on edges is assumed to be a small positive constant τ_0 (homogeneous distribution of pheromone at time t = 0).

B. *Vm Choosing Rule for Next Task*

During an iteration of the ACO algorithm each ant k, k = 1, ..., m (m is the number of the ants), builds a tour executing n (n is number of tasks) steps in which a probabilistic transition rule is applied. The k-ant chooses

VM j for next task i with a probability that is computed by Eq. (1).

$$P_{ij}^{k}(t) = \begin{cases} \frac{[\tau_{ij}(t)]^{\alpha}.[\eta_{ij}]^{\beta}}{\sum_{s\in allowed_{k}}[\tau_{is}(t)]^{\alpha}.[\eta_{is}]^{\beta}} & if\ j \in allowed_{k} \\ 0, & otherwise \end{cases}$$
(1)

```
Input: List of Cloudlet (Tasks) and List of VMs
Output: the best solution for tsaks alloction on VMs
Steps:
 1. Initialize:
        Set Current_iteration_t=1.
        Set Current_optimal_solution=null.
        Set an initial value $\tau_{ij}(t)$=c for each path between
            tasks and VMs.
 2. Place the m ants on the starting VMs randomly.
 3. For k :=1 to m do
        Place the starting VM of the k-th ant in $tabu_k$.
        Do ants_trip while all ants don't end their trips
           Every ant chooses the VM for the next task
              according to formula (1).
          Insert the selected VM to $tabu_k$.
        End Do
 4. For k :=1 to m do
        Compute the length $L_k$ of the tour described by the k-
            th ant according to formula (4).
        Update the current_optimal_solution with the best
            founded solution.
 5. For every edge (i,j), apply the local pheromone
        according to formula (5).
 6. Apply global pheromone update according to formula
    (7).
 7. Increment Current_iteration_t by one.
 8. If (Current_iteration_t < $t_{max}$)
       Empty all tabu lists.
       Goto step 2
    Else
       Print current_optimal_solution.
    End If
 Stop
```

Fig. 1. Pseudo code of basic ACO procedure

Where

- $\tau_{ij}(t)$ shows the pheromone concentration at the t time on the path between task i and VM j.
- $allowed_k = \{0,1,\ldots,n-1\}-tabu_k$ express the allowed VMs for ant k in next step and $tabu_k$ records the traversed VM by ant k.
- $\eta_{ij}=1/d_{ij}$ is the visibility for the t moment, calculated with heuristic algorithm and d_{ij} which expresses the expected execution time and transfer time of the task i on VM j can be computed with Eq. (2).

$$d_{ij} = \frac{TL\_Task_{i}}{Pe\_num_{j} * Pe\_mips_{j}\_VM_{j}} + \frac{InputFileSize}{VM\_bw_{j}} \quad (2)$$

Where TL_Task_i is the total length of the task that has been submitted to VM_j, Pe_num_j is the number of VM_j processors, Pe_mips_j is the MIPS of each processor of VM_j, InputFileSize is the length of the task before execution and VM_bw_j is the communication bandwidth ability of the VM_j.

- Finally the two parameters α and β control the relative weight of the pheromone trail and the visibility information respectively.

C. Pheromone Updating

After the completion of a tour, each ant k lays a quantity of pheromone $\Delta\tau_{ij}^{k}(t)$ computed by Eq. (3) on each edge (i,j) that it has used.

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \frac{Q}{L^{k}(t)} & if (i,j) \in T^{k}(t) \\ 0 & if (i,j) \notin T^{k}(t) \end{cases} \quad (3)$$

Where $T^k(t)$ is the tour done by ant k at iteration t, $L^k(t)$ is its length (the expected makespan of this tour) that is computed by Eq. (4), and Q is a adaptive parameter.

$$L^{k}(t) = arg\ max_{j\in J}\{sum_{i\in IJ}(d_{ij})\} \quad (4)$$

Where, IJ is the set of tasks that assigned to the VM_j.

After each iteration pheromone updating which is applied to all edges is refreshed by Eq. (5).

$$\tau_{ij}(t) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t) \quad (5)$$

Where ρ is the trail decay, $0 < \rho < 1$ and $\Delta\tau_{ij}(t)$ is computed by Eq. (6).

$$\Delta\tau_{ij}(t) = \sum_{k=1}^{m}\Delta\tau_{ij}^{k} \quad (6)$$

When all ants complete a traverse, an elitist is an ant which reinforces pheromone on the edges belonging to the best tour found from the beginning of the trial (T^+), by a quantity Q/L^+, where L^+ is the length of the best tour (T^+). This reinforcement is called global pheromone update and computed by Eq. (7).

$$\tau_{ij}(t) = \tau_{ij}(t) + \frac{Q}{L^{+}}\ \ if\ (i,j) \in T^{+} \quad (7)$$

IV. Cloud Task Scheduling Based on proposed MACO

The MACO algorithm inherits the basic ideas from ACO algorithm to decrease the computation time of tasks executing. In this section we introduce MACO algorithm to improve the performance of the scheduling problems in cloud computing. It is similar to ACO but has four major differences as following:

A. Vm Choosing Rule for Next Task

The Rule of Choosing Vm for Next Task is modified to Eq. (8).

$$P_{ij}^{k}(t) = \begin{cases} \arg\ max_{s\in allowe\ d_{k}}\{(\alpha * \tau_{is}(t)) + ((1-\propto) * \eta_{is})\} & if\ q \le q_{0} \\ J & if\ q > q_{0} \end{cases} \quad (8)$$

Where α is a parameter that allow a user to control the relative importance of pheromone trail and q is a random number uniformly distributed in [0, 1]. If q is greater than q_0, this process is called exploration; otherwise it is called exploitation [14]. Our principle to progressively adapt the system by tuning q_0 goes from 0 to 1, in order to favor exploration in the initial part of the algorithm and then favor exploitation. J is a random variable selected according to the following random-proportional rule probability distribution (Eq. (9)) which is the probability that ant k chooses to assign VM j to task i.

$$J = \begin{cases} \dfrac{\left(\alpha * \tau_{ij}(t)\right) + ((1-\alpha) * \eta_{ij})}{\sum_{s \in allowed_k}\left(\alpha * \tau_{is}(t)\right) + ((1-\alpha) * \eta_{is})} & if\ j \in allowed_k \\ 0, & otherwise \end{cases} \quad (9)$$

There are two reasons for adopting the above method to calculate the selection probability. The first is the simplicity as only one control parameter, i.e. α, is used to map the relative importance of quantity of pheromone and the desirability of each movement [15]. The second reason is the computational efficiency of this method as multiplication operations are used instead of exponentiations.

B. Local Pheromone Updating

The local pheromone update rule, which is applied to all edges, is replaced by Eq. (10).

$$\tau_{ij}(t) = (1-\rho_l)\tau_{ij}(t) + \Delta\tau_{ij}(t) \quad (10)$$

Where ρ_l is the trail decay, $0 < \rho_l < 1$

The following adaptive formula is proposed to compute ρ_l:

$$\rho_l = \frac{|L^K(t) - L^+|}{L^K(t) + L^+} + .1 \quad (11)$$

This formula removes pheromone from edges that belong to the worst tours. This has the effect of making the visited edges of worst tours less and less attractive. So, giving negative reinforcement to bad tours increases the convergence speed toward good solutions.

C. Global Pheromone Update.

The global pheromone update rule, which is applied to all edges belonging to the best tour (T+), is modified by the Eq. (12). The modification of global pheromone update is applied but in different shape in [6].

$$\tau_{ij}(t) = \left(1-\rho_g\right)\tau_{ij}(t) + \frac{\rho_g * \lambda}{L^+} \quad (12)$$

Where λ is an adaptive coefficient and ρ_g ($0 < \rho_g < 1$) is the pheromone evaporation parameter of global updating, they computed by Eq. (13) and Eq. (14) respectively:

$$\lambda = \frac{m}{t - ni_s + 1} \quad (13) \qquad \text{and} \qquad \rho_g = \frac{\rho_l}{2} \quad (14)$$

The λ coefficient used to control how a solution s contributes to pheromone information over time, m is the number of ants, ni_s represents the number of iterations that the best solution not changed. This global updating rule tries to increasing the learning of ants.

D. The Control Parameter α

The Parameter α controls the relative weight of the pheromone trail and the visibility information, and computed by the following adaptive formulas:

$$\alpha = \frac{\rho_l * m * L^k(t)}{t_{max}} \quad (15)$$

This adaptive rule tries to enhance the selection of weight pheromone trail and the visibility information. When the variation of pheromone concentration is high we give the visibility information high weight over pheromone trail and vice versa.

V. IMPLEMENTATION AND EXPERIMENTAL RESULTS

The proposed MACO and ACO algorithms are developed and simulated using Cloudsim. The comparison of MACO and ACO is done using various ranges of task length and resource capabilities. Cloudsim can be used to model data centers, host, service brokers, scheduling and allocation policies of a large scaled cloud platform. Hence, the researcher has used Cloudsim to model datacenters, hosts, VMs for experimenting in simulated cloud environment [16]. The parameters setting of cloud simulator are shown in Table 1. Our experiments are implemented with 10 Datacenters with 50 VMs and (100-1000) tasks under the simulation platform. The length of the task is from 1000 MI (Million Instructions) to 20000 MI. The parameters (α, β, ρ, t_{max}, m the number of ants and Q) considered here are those that affect directly or indirectly the computation of the algorithm. We tested several values for each parameter while all the others were held constant on 100 tasks. The ACO performance for different values of parameters (α, β, ρ, t_{max} ,m the number of ants and Q) has been evaluated. The selected best parameters of ACO are shown in table 2. The MACO parameters are calculated using the proposed self-adaptive formulas. The m and tmax parameters are the same in both algorithms.

TABLE I. PARAMETERS SETTING OF CLOUD SIMULATOR

Entity Type	Parameters	Value
Task (cloudlet)	Length of task	1000-20000
	Total number of task	100-1000
Virtual Machine	Total number of VMs	50
	MIPS	500-2000
	VM memory(RAM)	256-2048
	Bandwidth	500-1000
	cloudlet Scheduler	Space_shared and Time_shared
	Number of PEs requirement	1-4
Datacenter	Number of Datacenter	10
	Number of Host	2-6
	VmScheduler	Space_shared and Time_shared

TABLE II. SELECTED PARAMETERS OF ACO

Parameter	α	β	ρ	Q	m	t_{max}
Value	. 3	1	.4	100	10	100

In the following experiments, we assume that tasks are mutually independent i.e. there is no precedence constraint between tasks and tasks are not preemptive and they cannot be interrupted or moved to another processor during their execution. We compared the average makespan with different tasks set. The average makespan of the MACO, ACO algorithm is shown in Fig. 2. It can be seen from the figure, with the increase of the quantity task, the MACO takes the time less than ACO algorithm.

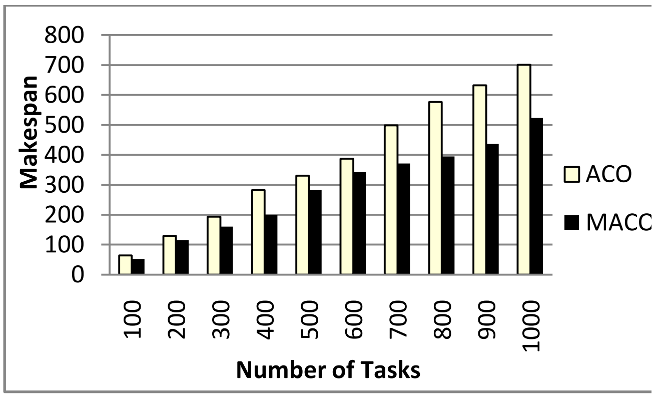

Fig. 2. Average makespan of MACO and ACO

The degree of imbalance measures the imbalance among VMs, which is computed by Eq. (16).

$$Di = \frac{T_{max} + T_{min}}{T_{avg}} \quad (16)$$

Where, T_{max}, T_{min} and T_{avg} are the maximum, minimum and average execution time of all VMs respectively. The average degree of imbalance (DI) of each algorithm with the number of tasks varying from (100) to (1000) is shown in Fig. 3. It can be seen that the MACO can achieve better system load balance than ACO.

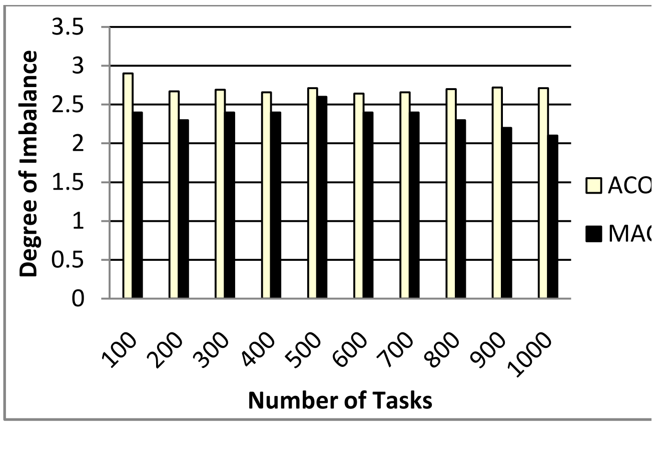

Fig. 3. Average degree of imbalance (DI) of MACO and ACO

VI. CONCLUSIONS AND FUTURE WORK

In this paper we have proposed MACO algorithm for improving the cloud tasks scheduling. MACO is used to find the optimal resource allocation for batch tasks in the dynamic cloud system and minimize the makespan of tasks on the entire system. It introduces self-adapting criteria for the ACO control parameters. We have experimentally evaluated the MACO and ACO algorithms in applications with the number of tasks varying from 100 to 1000 using Cloudsim. Simulation results demonstrate that MACO algorithm outperforms ACO algorithm. MACO algorithm can be extended with improvements to handle precedence between tasks and load balancing.

REFERENCES

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

[2] 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

[3] Paul, M., Sanyal, G., "Survey and analysis of optimal scheduling strategies in cloud environment", IEEE International Conference on Information and Communication Technologies (WICT), pp. 789 – 792, 2012

[4] 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), pp.673-675, 2010

[5] Meng Xu, Lizhen Cui, Haiyang Wang, Yanbing Bi, "A Multiple QoS Constrained Scheduling Strategy of Multiple Workflows for Cloud Computing", IEEE International Conference on Parallel and Distributed Processing with Applications, PP. 629 - 634, 2009

[6] Y. Gao et al., "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing" J. Comput. System Sci. (2013) http://dx.doi.org/10.1016/j.jcss.2013.02.004

[7] B. Rajkumar, B. Anton, and A. Jemal, "Energy efficient management of data center resources for computing: Vision, architectural elements and open challenges," in International Conference on Parallel and Distributed Processing Techniques and Applications, Jul. 2010

[8] M. Mazzucco, D. Dyachuk, and R. Deters, "Maximizing Cloud Providers' Revenues via Energy Aware Allocation Policies," in 2010 IEEE 3rd International Conference on Cloud Computing. IEEE, pp. 131–138, 2010

[9] Boloor, K., Chirkova, R., Salo, T., Viniotis, Y., "Heuristic-Based Request Scheduling Subject to a Percentile Response Time SLA in a Distributed Cloud". IEEE International Conference on Global Telecommunications Conference (GLOBECOM), PP.1-6 , 2010

[10] Chenhong Zhao, Shanshan Zhang, Qingfeng Liu, Jian Xie, Jicheng Hu, "Independent Tasks Scheduling Based on Genetic Algorithm in Cloud Computing", IEEE International Conference on Wireless Communications, Networking and Mobile Computing, PP. 1 – 4, 2009

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

[12] 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

[13] M. Dorigo, M. Birattari, T. Stutzel, "Ant colony optimization", in IEEE Computational Intelligence Magazine, pp.28-39, 2006.

[14] M. Dorigo, V. Maniezzo, A. Colorni, Ant system: optimization by a colony of cooperating agents, IEEE Trans. Syst. Man Cybernet. Part B: Cybernet-ics 26 (1) pp. 29–41, 1996

[15] V. Maniezzo, Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem, INFORMS J. Comput. 11 (4) pp. 358–369, 1999

[16] Ghalem, B., Fatima Zohra, T., and Wieme, Z. "Approaches to Improve the Resources Management in the Simulator CloudSim" in ICICA 2010, LNCS 6377, pp. 189–196, 2010.