Research Article

Cloud Service Scheduling Algorithm Research and Optimization

Hongyan Cui,^{1,2} Xiaofei Liu,¹ Tao Yu,³ Honggang Zhang,⁴ Yajun Fang,⁵ and Zongguo Xia⁴

¹State Key Lab of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China

²Key Lab of Network System Architecture and Convergence, Beijing 100876, China

³Institute of Network Science and Cyberspace, Tsinghua University, Beijing, China

⁴UMass Boston, 100 William T Morrissey Boulevard, Boston, MA 02125, USA

⁵MIT CSAIL Lab, Cambridge, MA 02139, USA

Correspondence should be addressed to Hongyan Cui; yan555cui@163.com

Received 6 July 2016; Accepted 20 October 2016; Published 11 January 2017

Academic Editor: Jong-Hyouk Lee

Copyright © 2017 Hongyan Cui et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

We propose a cloud service scheduling model that is referred to as the Task Scheduling System (TSS). In the user module, the process time of each task is in accordance with a general distribution. In the task scheduling module, we take a weighted sum of makespan and flowtime as the objective function and use an Ant Colony Optimization (ACO) and a Genetic Algorithm (GA) to solve the problem of cloud task scheduling. Simulation results show that the convergence speed and output performance of our Genetic Algorithm-Chaos Ant Colony Optimization (GA-CACO) are optimal.

1. Introduction

Cloud service is a service that users obtain through the network, which features instant response and scalability. Nowadays, there are mainly three service modules for cloud computing [1]: IaaS, SaaS, and PaaS. Infrastructure as a Service (IaaS) enables customers to acquire services from updated and powerful computer infrastructures via the internet. Software as a Service (SaaS) is a software mode also provided through internet. Users can lease a web-based application to meet their business-level requirements without permanently buying the whole software package. Platform as a Service (PaaS) provides a platform allowing customers to develop, run, and manage applications without the complexity of building and maintaining the infrastructure typically associated with developing and launching an application. And to efficiently complete various tasks submitted by clients, a task scheduling method is absolutely needed. Some works have been done in this area like adaptive scheduling strategies [2], and the cloud task scheduling method introduced in this paper is an application of PaaS.

Owing to the features [3] like on-demand resource allocation and varieties of flexible and high-quality services, cloud computing is widely used for processing large-scale tasks.

How to schedule large quantities of tasks reasonably is thus of vital importance for allocating resources on the cloud, necessitating research into cloud computing task scheduling strategy.

As cloud computing services have grown in popularity, the amount of data and tasks to be dealt with have also sharply increased, requiring a lot of system resources and sometimes resulting in severe resource waste. We therefore aim to schedule these data and tasks more efficiently. In order to analyze the problem at a somewhat detailed level, we have built a module for cloud task scheduling, as shown in Figure 1. The Task Scheduling System (TSS) consists of three modules. One is a user module, which provides an interface for users to submit requests that will be pushed into a Queue of Tasks (QOT) later.

In order to facilitate analysis of request processing, we assume that requests sent by users are arriving in a Poisson distribution [4]. The second module is a data center [5] (DC), which is an infrastructure-level service provided by the cloud provider. It encapsulates a series of hosts whose resources are allocated at the same time. These in turn host a series of virtual machines which form the Virtual Cloud Network (VCN); each virtual machine is called a Virtual Cloud Network node, with properties including accessible memory, processors, and

FIGURE 1: Cloud scheduling system model.

storage capacity that are mainly responsible for task processing. Finally, the third module is a task scheduling module, the core of TSS. TSS moves tasks from QOT into memory, allocating the necessary resources from the scheduler module based on a certain job scheduling algorithm in order to meet customer's requirements.

The rest of the paper is organized as follows: Section 2 explains our objective function and describes our assumptions. In Section 3, we will talk about three algorithms, namely, the GA, CAO, and GA-CACO, in detail and describe how to use them to solve the cloud task scheduling issue in TSS. Section 4 presents and analyzes the simulation results to test our algorithms. Finally, we will conclude our work in Section 5.

2. Problem Formulation

The goal of cloud scheduling is to optimize the process of tasks and VCN. An excellent task scheduling algorithm leads to better system performance. We can consider our task scheduling problem as an NP problem. A lot of algorithms have been suggested to tackle the NP problem [6–9]. For example, Genetic Algorithms and Ant Colony Algorithms, as well as some other heuristic algorithms, are characterized by fast convergence and robustness. Previous studies on cloud scheduling mostly focus on a single scheduling algorithm [10–12]. [13] solves the task scheduling problem using the Genetic Algorithm (GA) and [14] adopts Ant Colony Optimization (ACO). The search performance of the Genetic Algorithm is fast globally, but the efficiency of searching

decreases when searching scale increases because of the redundant iterations at the start of the algorithm. Besides, due to its poor stability, it is difficult for the algorithm to converge to a stable solution. The Ant Colony Algorithm, on the other hand, is less likely to be able to solve problems as it is unaware of the whole picture at the beginning phase, resulting in its slow convergence. However, as the pheromone grows during the process, the capability of searching using the Ant Colony Algorithm has also been enhanced thanks to the positive feedback mechanism. Some research like [15, 16] has also proposed Chaos Ant Colony Optimization (CACO) to overcome the slow convergence of ACO, but these still have local optimum defects. In this paper, we propose the Genetic Algorithm-Chaos Ant Colony Optimization (GA-CACO) Algorithm, which inherits the chaotic motion characteristics of CACO to overcome the deficiency of pheromone in early phase and also utilizes the crossover as well as mutation feature of the Genetic Algorithm to deal with the problem of local optimum in ACO.

In this paper, we will take makespan and flowtime as the criteria of algorithm's performance in cloud task scheduling. Makespan represents the time when TSS finishes the latest task. Flowtime equals the sum of execution time of all tasks [17]. We propose GA, ACO, and GA-CACO algorithms to study the cloud task scheduling problem and compare their performance later in the paper.

2.1. Assumptions. In order to perform a quantitative analysis of the TSS system modules, we will make some assumptions as follows.

- (1) Tasks can be expressed within the range of code instruction length, which also is the size of the task.
- (2) The arrival of tasks follows the Poisson distribution and the length of the queue is unbounded.
- (3) Each node is independent of all others and each node can only execute one task at a time.
- (4) When nodes fail, they do so independently of one another.

2.2. Objective Function. As previously mentioned, tasks are stored in QOT and then scheduled by TSS. TSS's task processing can be divided into three phases [16]: waiting, scheduling and execution, and feedback. We do not discuss feedback as it is not relevant to this research.

First, we assumed that tasks follow a M/G/S queuing model [17]: specifically, tasks reach TSS following a Poisson distribution and service times have a general distribution. Suppose that the arrival rate of task i is γ_i ($i = 1, 2, \dots, n$), where n is the number of tasks and the service rate is μ_i ($i = 1, 2, \dots, n$). According to Atencia and Moreno [18], we derive the average waiting time for a task:

$$W_i = \frac{R}{1 - \rho} \quad i = 1, 2, 3, \dots, n, \quad (1)$$

where R stands for residual service time and ρ represents system utilization.

$$R = \sum_{i=1}^n \frac{\rho_i}{\mu_i} \quad i = 1, 2, 3, \dots, n \quad (2)$$

$$\rho = \frac{\gamma_i}{s * \mu_i} \quad i = 1, 2, 3, \dots, n,$$

where s is the number of network nodes.

During the TSS operation, QOT pops n tasks, the i th task is denoted as J_i ($i = 1, 2, 3, \dots, n$), and submits them to VCN (as shown in Figure 2) to be executed. We can then get a task execution time matrix [19], PTM:

$$Q = \begin{bmatrix} pt_{11} & pt_{12} & pt_{13} & \cdots & pt_{1m} \\ pt_{21} & pt_{22} & pt_{23} & \cdots & pt_{2m} \\ & & \vdots & & \\ pt_{i1} & \cdots & pt_{ij} & \cdots & pt_{im} \\ & & \vdots & & \\ pt_{n1} & pt_{n2} & pt_{n3} & \cdots & pt_{nm} \end{bmatrix}, \quad (3)$$

where $pt_{i,j}$ represents the time consumed by node j to process task i . We acquire the total time T_i of task i by

$$T_i = pt_{ij} + W_i. \quad (4)$$

Thus

$$\text{Makespan} = \max T_i$$

$$\text{Flowtime} = \sum_{i=1}^n T_i. \quad (5)$$

FIGURE 2: QOT and VCN.

We try to minimize the makespan and flowtime, so the objective function can be expressed as

$$f = \min [a \times \text{makespan} + b \times \text{flowtime}], \quad (6)$$

where a and b are weight coefficients, with $a \geq 0$ and $b \geq 0$ and $a + b = 1$. Here, we want to consider makespan and flowtime to be equally important; thus we assign $a = b = 0.5$, and this objective function will be seen as the criteria for computing individual fitness in the later section.

3. Algorithm Application

3.1. Genetic Algorithm (GA). For an optimization problem, a number of candidate solutions that are called individuals and whose abstract representations are named chromosomes, are improved in each iteration and eventually evolve to the optimum solution. The Genetic Algorithm (GA) uses a selection operation to determine the survival for each individual. That is, the fitness of the whole population is evaluated. Multiple individuals are chosen randomly from the current population based on their fitness, and individuals with high fitness are inherited in the next generation with a higher probability; on the contrary, for low fitness individuals, the probability of being inherited in the next generation is smaller. For each generation, a new population is generated according to the natural selection process described above and mutation.

3.1.1. Selection Operation. The roulette selection [20, 21] method is the most commonly used method for selecting individuals in the natural selection process just outlined above. As shown in Figure 3, the basic idea of a selection operator, also known as proportional selection, is to set the probability of an individual being selected in proportion to its fitness function value. If we imagine that the population size is N and the fitness of an individual x_i is $f(x_i)$, then probability x_i that will be selected by our operator is

$$P(x_i) = \frac{f(x_i)}{\sum_{j=1}^N f(x_j)}. \quad (7)$$

FIGURE 3: Roulette selection.

Assuming that every individual's chromosome represents a small sector on a wheel and its size is proportional to the fitness of the chromosome, then individuals with better adaptability are more likely to be chosen.

3.1.2. Genetic Operations. In order to promote evolution of populations and speed up convergence towards the optimal solutions, as well as to increase the diversity of the population, we need to perform genetic operations such as crossover and mutation on the individuals selected according to the roulette selection discussed previously.

(a) Crossover operation

Two chromosomes interchange part of their genes with each other based on the crossover probability, forming two new individuals.

Before crossover:

```
01000|0100001000
11000|1111000010
```

After crossover:

```
01000|1111000010
11000|0100001000
```

(b) Mutation operation

Mutation changes the code string of a chromosome of an individual to a new value of certain gene. The mutation operation is a good method to maintain population diversity.

Before mutation:

```
01000|1111000010
```

After mutation

```
01000|1111001010
```

3.1.3. GA Algorithm Framework for Cloud Task Scheduling. See Algorithm 1.

3.2. Ant Colony Optimization (ACO). An Ant Colony Optimization algorithm is a random probability model made to search for an optimal path, which is inspired by how ants discover a path to food. Ants will release a certain amount of pheromones as they explore. Gradually, the pheromone concentration on the shortest path becomes higher and attracts more ants, thus forming a positive feedback which leads to the discovery of the best path. Initially, the amount of pheromone on edges is assumed to be a small positive constant τ_0 ; in this paper, we set $\tau_0 = 1$. For each iteration of ACO, each ant gradually establishes a path to the "food." When an ant arrives at a certain node, it will make a decision on the next one it is going to visit. Imagine an ant k is located at node i ; then the probability that the ant will be at node j one step later is

$$p_{ij}^k(t) = \begin{cases} \frac{\tau_{ij}^\alpha(t)}{\sum_{j \in N_t^k} \tau_{ij}^\alpha(t)}, & j \in N_t^k \\ 0, & j \notin N_t^k, \end{cases} \quad (8)$$

where N_t^k are the adjacent nodes of node i . α is a factor used to amplify the impact of pheromone concentration. If α is too large, it will overpower the impact of the pheromone, leading the algorithm to converge to a suboptimal path. From experience, we assign $\alpha = 2$. Finally, we assume that the amount of pheromones released by ants walking along a path is $\Delta\tau_{ij}^k$:

$$\Delta\tau_{ij}^k \propto \frac{1}{f^k(t)}, \quad (9)$$

where $f^k(t)$ represents the fitness value of the VCN node path that the ant k is at iteration t . When it is in the $t + 1$ iteration, the pheromone concentration for each path becomes

$$\tau_{ij}(t+1) = (1 - \beta) \times \tau_{ij}(t) + \sum_{k=1}^{n_k} \Delta\tau_{ij}^k, \quad (10)$$

where n_k is the number of ants and β is the pheromone evaporation parameter.

3.2.1. CAO Algorithm Framework for Cloud Task Scheduling. See Algorithm 2.

3.3. Genetic Algorithm-Chaos Ant Colony Optimization (GA-CACO). When the ant colony system initializes, the pheromones of all paths take the same value, so the selecting probability of all the paths is the same. It is difficult to find an optimal path under this condition, resulting in slow convergence time for Ant Colony Optimization. To overcome this problem, some researchers have proposed the Chaotic Ant Colony Algorithm (CACO) [22], which combines the chaotic behavior of a single ant and the intelligent organization actions of an ant colony. CACO introduces an organization variable which is used to control self-organization process of the ant

```

Step 1. Initialization (configure parameters, initialize VCN)
Step 2. While iteration t < maximum iterations do:
    Calculate fitness value of every individual;
    Select individuals using roulette rotation method;
    Crossover and mutate individuals which has been selected to produce new individuals;
Step 3. End while
Step 4. Return best solution

```

ALGORITHM 1: GA procedure.

```

Step 1. Initialization (configure parameters, initialize QOT, VCN, and initialize the initial position of
all ants)
Step 2. While iteration t < maximum iterations do:
    Calculate the next position of ants according to the equation (8).
    Calculate the fitness value of every candidate solution.
    Update pheromone matrix according to equation (10).
Step 3. End while
Step 4. Select the optimal solution and return.

```

ALGORITHM 2: CAO procedure.

colony. Initially, the influence of the organization variable on the behavior of a single ant is small. As the organizational variable changes, its impact on the behavior of a single ant grows stronger. When the effect is large enough, the chaotic behavior of ants will disappear. Then, ants can move in a more regular way. Through the whole process, every ant exchanges information with its neighbors. However, the behavior is partial: a Chaotic Ant Colony Algorithm may easily fall into a local minimum, although this scenario is less likely to occur than for an ACO. In this paper, GA-CACO is proposed, merging GA and CACO. Due to the global search of GA, CACO can effectively avoid local minimum values.

Here we will introduce the mathematical model of CACO. Formula (11) describes the changes of organizational variables:

$$y_i(t) = y_i(t-1)^{(i+r_i)}. \quad (11)$$

The position of ants depends on the following:

$$\begin{aligned}
 x_{id}(t) = & \left(x_{id}(t-1) + \frac{7.5}{\Phi_d} \times v_i \right) \\
 & \times \exp \left((1 - \exp(-a \times y_i(t))) \right) \\
 & \times \left(3 - \Phi_d \times \left(x_{id}(t-1) + \frac{7.5}{\Phi_d} \times v_i \right) \right) + (p_{id}(t-1) - x_{id}(t-1)) \times \exp(-2a \times y_i(t) + b) - \frac{7.5}{\Phi_d} \\
 & \times v_i,
 \end{aligned} \quad (12)$$

where $y_i(t)$ represents the organizational variable in the iteration time t for the ant i . r_i is the organization factor, which

TABLE 1: Environment configuration.

Type	Specification	Value
Data center	Hardware resources of cloud computing	1
Cloud network nodes	Cloud computing nodes, similar to VM	20–100
Tasks	Tasks submitted remotely	10–100
	The length of task	500–1000

affects the convergence speed of the optimization process. Generally, $y_i(0) = 0.99$ and $0 \leq r_i \leq 0.5$. $x_{id}(t)$ is the position of ant i in d -dimensional space, with $d = 1$ in our case. v_i decides the attractive factor of ants' chaotic dynamical behavior in phase space. Φ_d is a constant adjusting search scope of x_{id} . $p_{id}(t)$ is the best position that ant i and its neighbors found within t steps; a and b are two positive constants which can be selected as $a = 200$ and $0 \leq b \leq 2/3$.

3.3.1. GA-CACO Framework for Cloud Task Scheduling. See Algorithm 3.

4. Simulation Results

In this section, we will show that simulation results verify the usefulness of the algorithm. In this experiment, we have compared the results of three options: GA, CAO, and GA-CACO. The GA-CACO algorithm indeed has the best performance. The experiment is performed under the hardware environment with "Intel Core-i3-2310 CPU-2.10 Ghz" processor and 10 G RAM. More details about resource allocation are shown in Table 1.


```

Step 1. Initialization, configure parameters.
Step 2. While iteration  $t < \text{maximum iterations}$  do:
    Calculate the distance between one ant and others, choose its neighbors and store its position and
    object value;
    Update organization factor and position of ants according to formulas (11), (12);
    Based on GA, modify genes of population, producing new individuals.
Step 3. End while
Step 4. Select the optimal solution and return.

```

ALGORITHM 3: GA-CACO procedure.

TABLE 2: Performance comparison of three algorithms.

	GA-CACO	CAO	GA
10	3.5229	3.6348	3.6720
20	5.8303	6.2052	6.4110
40	10.6139	11.4159	12.0167
60	18.7145	19.9727	20.5220
80	24.0400	25.5633	26.7745

FIGURE 4: Comparison among GA, ACO, and GA-CACO under different number of tasks.

Figure 4 shows the results of GA-CACO, ACO, and GA algorithm execution under different task sizes. The “Cost Value” is objective function; lower value is expected. The figure shows that GA-CACO algorithm is more efficient than other two algorithms when task amount is large. Specific values are shown in Table 2. Figure 5 compares the convergence rate of GA-CACO and ACO; the figure shows that GA-CACO and ACO algorithms are stable, while GA-CACO converges faster than ACO.

5. Conclusion

In this paper, we present a cloud service task scheduling model TSS, which consists of three modules: a user module, a data center module, and a task scheduling module. In the user module, we assume that the tasks model is a Poisson process. In the task scheduling module, because of the deficiencies

FIGURE 5: Convergence comparison between ACO and GA-CACO.

of GA and ACO, we have proposed a GA-CACO algorithm, which merges GA and CACO. And finally, we compared GA-CACO, GA, and ACO for several different task sizes. The results indicate that GA-CACO algorithm is optimal for the optimization of the objective function and also has acceptable convergence speed.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This work was supported by the Project 61201153 supported by National Natural Science Foundation of China, the Research Project of CCF-Qimingxingchen Hongyan (CCF-VenustechRP2016004), and the National 973 Program of China under Grant 2012CB315805.

References

- [1] D. Villegas, N. Bobroff, I. Rodero et al., “Cloud federation in a layered service model,” *Journal of Computer and System Sciences*, vol. 78, no. 5, pp. 1330–1344, 2012.
- [2] L. Deng, Q. Yu, and J. Peng, “Adaptive scheduling strategies for cloud-based resource infrastructures,” *Security and Communication Networks*, vol. 5, no. 10, pp. 1102–1111, 2012.

- [3] T. Muth, J. Peters, J. Blackburn, E. Rapp, and L. Martens, "ProteoCloud: a full-featured open source proteomics cloud computing pipeline," *Journal of Proteomics*, vol. 88, pp. 104–108, 2013.
- [4] A. C. Kotze, R. J. Dobson, D. Humphries, M. Wilson, and M. Cappello, "Application of a Poisson distribution quality control measure to the analysis of two human hookworm drug treatment studies in Ghana," *International Journal for Parasitology: Drugs and Drug Resistance*, vol. 4, no. 1, pp. 64–70, 2014.
- [5] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 68–73, 2008.
- [6] P. Wang, S. Huang, and Z.-Q. Zhu, "Swarm intelligence algorithms for circles packing problem with equilibrium constraints," in *Proceedings of the 12th International Symposium on Distributed Computing and Applications to Business, Engineering & Science (DCABES '13)*, pp. 55–60, IEEE, Los Alamitos, Calif, USA, September 2013.
- [7] X. Huang, G. Liu, W. Guo, and G. Chen, "Obstacle-avoiding octagonal steiner tree construction based on particle swarm optimization," in *Proceedings of the 9th International Conference on Natural Computation (ICNC '13)*, pp. 539–543, Shenyang, China, July 2013.
- [8] C. Wang, J. Zhang, J. Yang et al., "A modified particle swarm optimization algorithm and its application for solving traveling salesman problem," in *Proceedings of the International Conference on Neural Networks and Brain (ICNN&B '05)*, vol. 2, pp. 689–694, Beijing, China, October 2005.
- [9] Y. M. Wang and H. L. Yin, "A two-stage approach based on genetic algorithm for large size flow shop scheduling problem," in *Proceedings of the 10th IEEE International Conference on Mechatronics and Automation (IEEE ICMA '13)*, pp. 376–381, Takamatsu, Japan, August 2013.
- [10] J. Li and J. Peng, "Task scheduling algorithm based on improved genetic algorithm in cloud computing environment," *Journal of Computer Applications*, vol. 31, no. 1, pp. 184–186, 2011.
- [11] N. Srinivas and K. Deb, "Multiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary Computation*, vol. 2, no. 3, pp. 221–248, 1994.
- [12] J. P. Wang, Y. L. Zhu, and H. Y. Feng, "A multi-task scheduling method based on ant colony algorithm combined QoS in cloud computing," *Advances in Information Sciences and Service Sciences*, vol. 4, no. 11, pp. 185–192, 2012.
- [13] J. Cai, Q. Li, L. Li, H. Peng, and Y. Yang, "A fuzzy adaptive chaotic ant swarm optimization for economic dispatch," *International Journal of Electrical Power & Energy Systems*, vol. 34, no. 1, pp. 154–160, 2012.
- [14] N. Sridhar, N. Ramrao, and M. K. Singh, "Design of PID controller with auto tuned prefilter using chaotic optimization," in *Proceedings of the International Conference on Circuits, Communication, Control and Computing (I4C '14)*, pp. 141–145, IEEE, Bangalore, India, November 2014.
- [15] J. Xu, A. Y. S. Lam, and V. O. K. Li, "Chemical reaction optimization for task scheduling in grid computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 10, pp. 1624–1631, 2011.
- [16] Y. S. Dai, B. Yang, J. Dongarra et al., "Cloud service reliability: modeling and analysis," in *Proceedings of the 15th IEEE Pacific Rim International Symposium on Dependable Computing*, 2009.
- [17] X. Nan, Y. He, and L. Guan, "Optimal resource allocation for multimedia cloud based on queuing model," in *Proceedings of the 3rd IEEE International Workshop on Multimedia Signal Processing (MMSP '11)*, pp. 1–6, IEEE, Hangzhou, China, November 2011.
- [18] I. Atencia and P. Moreno, "A single-server retrial queue with general retrial times and Bernoulli schedule," *Applied Mathematics and Computation*, vol. 162, no. 2, pp. 855–880, 2005.
- [19] S. Ali, H. J. Siegel, M. Maheswaran et al., "Task execution time modeling for heterogeneous computing systems," in *Proceedings of the Heterogeneous Computing Workshop (HCW '00)*, pp. 185–199, IEEE, 2000.
- [20] A. Lipowski and D. Lipowska, "Roulette-wheel selection via stochastic acceptance," *Physica A: Statistical Mechanics and Its Applications*, vol. 391, no. 6, pp. 2193–2196, 2012.
- [21] W. Zhao, T. Tao, and E. Zio, "System reliability prediction by support vector regression with analytic selection and genetic algorithm parameters selection," *Applied Soft Computing*, vol. 30, pp. 792–802, 2015.
- [22] D. Yang, Z. Liu, and J. Zhou, "Chaos optimization algorithms based on chaotic maps with different probability distribution and search speed for global optimization," *Communications in Nonlinear Science and Numerical Simulation*, vol. 19, no. 4, pp. 1229–1246, 2014.