Moon *et al. Hum. Cent. Comput. Inf. Sci.* (2017) 7:28
DOI 10.1186/s13673-017-0109-2

Human-centric Computing and Information Sciences

RESEARCH Open Access

A slave ants based ant colony optimization algorithm for task scheduling in cloud computing environments

YoungJu Moon[1†], HeonChang Yu[1†], Joon-Min Gil[2†] and JongBeom Lim[3*†]

*Correspondence:
jblim@kpu.ac.kr
†YoungJu Moon, HeonChang Yu, Joon-Min Gil and JongBeom Lim contributed equally to this work
[3] Department of Game & Multimedia Engineering, Korea Polytechnic University, Siheung, Gyeonggi-do, South Korea
Full list of author information is available at the end of the article

Abstract

Since cloud computing provides computing resources on a pay per use basis, a task scheduling algorithm directly affects the cost for users. In this paper, we propose a novel cloud task scheduling algorithm based on ant colony optimization that allocates tasks of cloud users to virtual machines in cloud computing environments in an efficient manner. To enhance the performance of the task scheduler in cloud computing environments with ant colony optimization, we adapt diversification and reinforcement strategies with slave ants. The proposed algorithm solves the global optimization problem with slave ants by avoiding long paths whose pheromones are wrongly accumulated by leading ants.

Keywords: Task scheduling, Ant colony system, Optimization algorithm, Cloud computing

Background

As cloud computing and its applications have come into wide use such as security [1], IoT [2], and vehicular ad hoc networks [3], it is important for cloud users to provide an efficient task scheduling technique since cloud computing is based on the pay as you go pricing model [4]. For a cloud user, it is important to finish the cloud user's tasks as quickly as possible in cloud computing environments [5].

For instance, suppose there are two cloud task schedulers (schedA and schedB) for cloud users. The cloud task schedulers deploy the tasks of the users according to the specification of their algorithm. If the expected makespan of schedA is 151 s and that of schedB is 130 s, then the user will choose schedB for the tasks deployment since the user will pay more as much as schedA takes longer than schedB (for 21 s) if the user chooses schedB.

Ant colony optimization (ACO) is one of nature-inspired optimization algorithms and is based on the population of the ant movement. It transcribes the cooperative behavior of the ant colony system for solving a particular optimization problem. When ants travel in search of food, the ants secrete a chemical trail called pheromone and the ants like to travel along the trails that have the strongest pheromone scent. For ACO, the role of the trail of pheromone is to share their experience about the journey for solving an optimization problem efficiently.

In this paper, we propose a novel ACO based algorithm called slave ants based ant colony optimization (SACO) that schedules tasks of cloud users to virtual machines (VMs) in cloud computing environments in an efficient manner with optimized parameter mapping. The proposed algorithm solves the global optimization problem with slave ants by avoiding long paths whose pheromones are wrongly accumulated by leading ants.

The rest of this paper is organized as follows. The next section discusses about related work for the ant colony optimization and cloud task scheduling techniques. Then, we present details of system model for SACO and the proposed task scheduling algorithm in "Slave ants based ant colony optimization algorithm for task scheduling". We give our results highlighting the efficiency of our proposed algorithm with scalable settings in "Performance evaluation". Finally, we conclude the paper in "Conclusion".

Preliminaries and related work

Ant colony optimization

Ant colony optimization is an evolutionary computational technique proposed by Dorigo et al. [6]. The ACO is composed of three steps for a cycle:

- *explore()* Release ants for finding the destination.
- *pheromoneUpdate()* Update pheromones the ants travel across the paths.
- *iteration()* Compare paths the ants found and find the best path if a predefined condition is met. Otherwise, iterate the process of finding the destination.

Figure 1 illustrates an example of finding the shortest path of the journey. If an ant encounters three-forked paths, it selects one of the three paths with probability parameters. After these processes iterate, ants will follow the shortcut because pheromones are most accumulated on the shortcut.

Related work

There are three categories for task scheduling strategies in cloud computing environments: focusing on performance by improving of cloud task schedulers (Category 1), considering multi-objective (e.g., QoS, energy consumption, financial cost, and SLA)

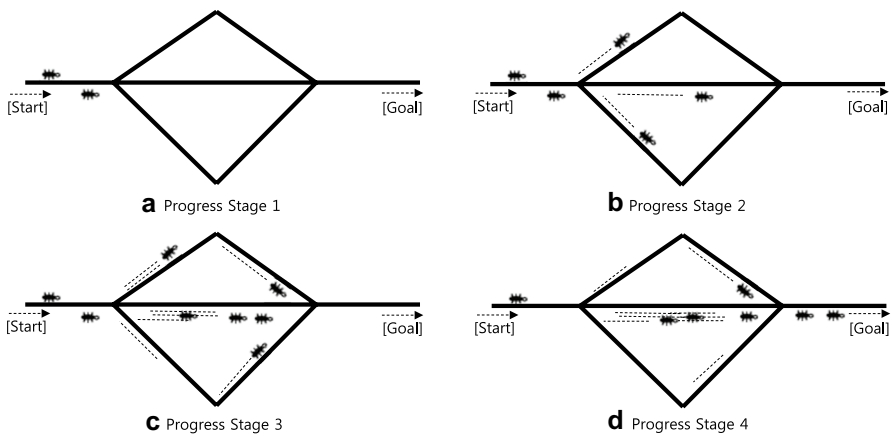

Fig. 1 An example of finding the shortest path

for finding optimal solutions (Category 2), and using nature-inspired optimization algorithms to solve cloud task scheduling problems (Category 3).

Category 1

The total processing time of cloud tasks is one of most important factors for schedulers. To reduce the total processing time, several research studies have been proposed with heuristic algorithms. In [7], the authors divided the scheduling problem into two parts: the diversity detection and improvement detection. Then, it dynamically selects one of the heuristic algorithms to solve the scheduling problem. In [8], the authors considered information of tasks (i.e., task completion time and size of tasks) and the information is used in the self-adaptive scheduling technique to reduce total processing time and average response time. To solve dependency issues of cloud tasks, the scheduling problem can be translated to the directed acyclic graph (DAG) [9]. To address the performance issue, the authors of [10] proposed a prioritizing scheme of the DAG of tasks and the authors of [11] proposed an AREA-Oriented scheduling (AO-scheduling) to compensate the defects of cloud based scheduling problems. The authors of [12] proposed a workflow scheduling based on directed search optimization (DSO) with artificial neural network (ANN) and radial basis function neural network (RBFNN).

Category 2

In cloud computing environments, there are additional factors to be considered for scheduling tasks (e.g., QoS, energy consumption, financial cost, and SLA). To incorporate these influencing factors, a multi-objective optimization method has been proposed with four performance metrics (makespan, cost, deadline violation rate, and resource utilization) [13]. As energy consumption becomes the key issue for the operation and maintenance of cloud datacenters [14], the authors of [15] introduced the concept of skewness to measure the unevenness of servers' resource utilization.

Category 3

As one of the nature-inspired optimization algorithms, ACO has been widely used for solving cloud task scheduling problems. In [16], the authors employed the concept of lazy ants for finding solutions. However, it introduces additional preprocessing time due to travel iterations. The authors of [17] used ACO to schedule cloud tasks with a random approach. Our slave ants based ant colony optimization algorithm for task scheduling differs from previous work in that we adapt diversification and reinforcement strategies with slave ants and the proposed ACO algorithm solves the global optimization problem with slave ants by avoiding long paths whose pheromones are wrongly accumulated by leading ants.

Although security issues are also of concern for using cloud computing environments, we focus on task scheduling mechanisms to improve the performance and server utilization. Mitigating threats [18] and detecting attacks [19] in cloud computing environments are beyond the scope of this paper. Nonetheless, our approach can be applied in various cloud computing architectures since the scheduling component can be developed and easily integrated to the cloud system without dependency.

Slave ants based ant colony optimization algorithm for task scheduling

System model

Figure 2 shows the architecture framework model for SACO. When a user submits a set of cloud tasks, the task manager accepts the tasks and manages them with the task scheduler. The task scheduler, on which the proposed task scheduling algorithm resides, is responsible for deciding the task allocation to resources. The task scheduler interacts with the resource manager, which is able to monitor the physical/logical resources and the flow of task allocation from the task scheduler.

The resource manager periodically monitors resources (physical and virtual machines) and stores information about CPU utilization and memory usage. Then, the stored information can be used for the task scheduler. Therefore, the task scheduler maintains up-to-date resource information by collecting and updating the system's information from the resource manager.

The basic rules for scheduling cloud tasks are as follows: (1) it collects information about cloud tasks and resources from the resource manager, (2) it checks whether the resource VM_j meets the requirement of the task T_i. Note that the requirement is dependent on the specification of the scheduling algorithm, and (3) the scheduler allocates the task T_i to resource VM_j with the help of the resource manager.

The proposed task scheduling algorithm

In ACO, there are rules for state transitions and pheromone updates. For the state transition rule, the probability that an ant transits from node i to j is determined by Eq. 1.

$$p_{i,j} = \tau_{i,j}^{\alpha} \cdot \eta_{i,j}^{\beta} / \Sigma l \notin S \tau_{i,l}^{\alpha} \cdot \eta_{i,l}^{\beta}, \tag{1}$$

where α and β are parameters that determine the amount of pheromones and pheromone reinforcement. The symbols used in our ACO based algorithm are listed in Table 1.

To enhance the performance of the task scheduler in cloud computing environments, we adapt a diversification strategy with slave ants. The basic idea of the diversification strategy

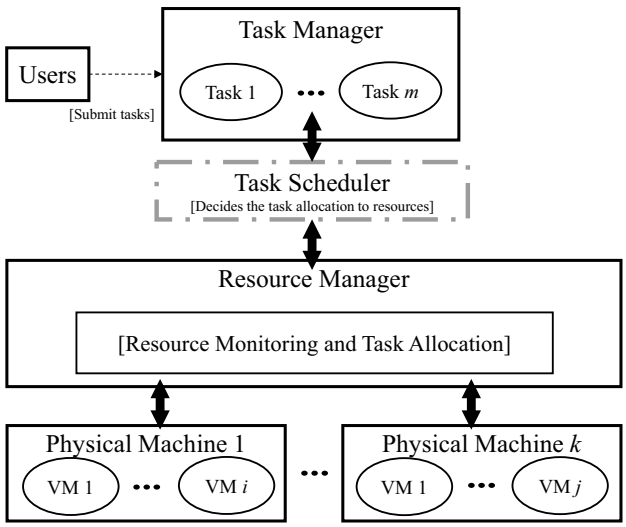

Fig. 2 The system framework model for SACO

Table 1 The list of symbols used in the algorithm

Symbol	Definition
T_i	Task, $\forall i \in \{1, 2, \ldots, n\}$
R_j	Resource, $\forall j \in \{1, 2, \ldots, m\}$
S	A set of nodes an ant traveled
$\eta_{i,j}$	The importance between node i and j
$p_{i,j}$	The probability that an ant transits from node i to j
$\tau_{i,j}$	The amount of pheromones between T_i and R_j
q_0	The probability parameter for host ants
s_n	The probability parameter for slave ants, $\forall n \in \{1, 2, \ldots, k\}$
k	The number of slave ants of a host ant
α, β	The parameters that determine pheromones and reinforcement
ϕ	The parameter for local pheromones evaporation
d	The parameter for acceleration of local pheromones evaporation
p	The parameter for global pheromones evaporation

is to have different probability parameters for slave ants to solve the task scheduling optimization problem. The probability parameter for slave ants is determined by Eq. 2.

$$s_n = q_0 \times n/k, \tag{2}$$

where $\forall n \in \{1, 2, \ldots, k\}$, $s_n \leq q_0$, and $0 < q_0 < 1$.

With Eq. 2, each slave ant of the colony system has a different probability parameter and less dependency with other ants.

To reduce makespan of cloud tasks, an effective pheromone update scheme is essential. For the reinforcement strategy, we employ the acceleration parameter of local pheromones evaporation. The local pheromone update rule used in our algorithm is based on Eq. 3.

$$\tau_{i,j} = (1 - \phi - d)\tau_{i,j} + \phi\tau_0, \tag{3}$$

where ϕ is the parameter for local pheromones evaporation and τ_0 is the initial value of pheromone.

Based on the local pheromone update rule, the proposed global pheromone update rule can be applied with Eq. 4.

$$\tau_{i,j} = (1 - p)\tau_{i,j} + \Delta\tau_{i,j}^{\text{best}}. \tag{4}$$

Algorithm 1 shows the proposed task scheduling algorithm based on ACO with slave ants. The input of the algorithm is a set of cloud tasks and resources (VMs) and the output is the best mapping information between the tasks and resources. At the initial stage of a cycle, it releases all the ants of the colony system for finding solutions.

The assignment of a task to a resource is dependent on the probability parameter determined by Eq. 1. For an ant group (including a normal ant and its slave ants), makespan information is updated (Line 10). With updated makespan information, one of ants will become a normal ant whose makespan is best for the group. Other ants except the elected normal ant become slave ants and one of slave ants whose make space is worst will be designated as a weak ant (Line 11–12). Then, the local pheromone update procedure is performed (Line 13).

After all the ants of the colony system finish the journey, the makespan information is recoded. Among normal ants, the one that has the best mapping information between cloud tasks and resources will be elected as a queen ant. Next, the global pheromone update procedure is performed. Then, the queen ant's makespan is better than the existing map $(T_i, R_j)_{best}$, the queen ant's mapping is assigned to map $(T_i, R_j)_{best}$ (Line 17–19). Finally, the algorithm returns map $(T_i, R_j)_{best}$. The next cycle will be performed again when the predefined condition is not met. Otherwise, the cloud scheduler deploys the cloud tasks to the resources with the returned mapping information.

Algorithm 1. The task scheduling algorithm based on ACO with slave ants

Input: T_i, R_j, where $\forall i \in \{1, 2, \ldots, n\}$ and $\forall j \in \{1, 2, \ldots, m\}$

Output: Optimized map $(T_i, R_j)_{best}$

```
1:  begin
2:      for each ant ∈ antList
3:          for each slaveAnt ∈ slaveAntList_ant
4:              for each T_i
5:                  for each R_j
6:                      Assign T_i to R_j with probability parameter;
7:                  end for
8:              end for
9:          end for
10:         Call updateMakespanInformation ();
11:         Elect slaveAnt_best as a normal ant based on makespan;
12:         Elect slaveAnt_worst as a weak ant based on makespan;
13:         Call localPheromoneUpdate ();
14:     end for
15:     Elect ant_best as a queen ant based on makespan;
16:     Call globalpheromoneUpdate ();
17:     if a solution of ant_best is better than existing map (T_i, R_j)_best then
18:         map (T_i, R_j)_best ← map (T_i, R_j) of ant_best;
19:     end if
20:     return map (T_i, R_j)_best ;
21: end
```

Performance evaluation

In this section, we provide experimental results to show the efficiency of the proposed cloud task scheduling algorithm in terms of makespan. Since cloud computing environments adapt virtualization technology [20], scalability should be considered enabling large-scale distribution [21]. Table 2 shows the experimental settings. To show the scalability of the algorithm in terms of the number of tasks, we vary the number of cloud tasks from 100 to 700 in comparison with default ACO and IACO algorithms [17].

Figure 3 shows the performance results of makespan for ACO, IACO, and SACO. When the number of cloud tasks is 100 and 200, ACO outperforms IACO since IACO

Table 2 Experimental settings

Parameter	Value
The number of data centers	6
The number of hosts	6
Host memory	4096 MB
Host bandwidth	2660 MB/s
The number of CPUs	[2, 4]
The number of VMs	50
The number of vCPU	[1, 4]
vCPU capacity	[500, 2000] MIPS
vRAM	[256, 2048] MB
Bandwidth	[500, 1000] MB/s
The number of cloud tasks	[100, 700]
The required MIPS for tasks	[100, 20,000]
File size of tasks	[200, 400] MB
Output file size	[20, 40] MB

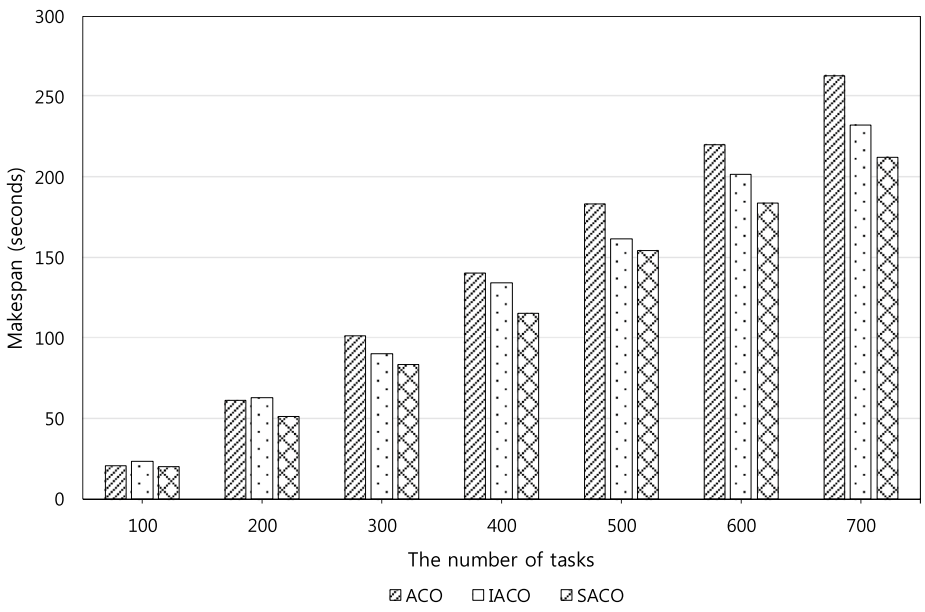

Fig. 3 Performance results of ACO, IACO, and SACO as the number of cloud tasks increases (makespan)

relies solely on uncertainty based on probability parameters. In other words, in IACO, the leading ants affect the global optimization and it is very prone to local minima.

The performance gaps among the three algorithms increases as the number of cloud tasks increases. When the number of cloud tasks is 200, the difference of the makespan between ACO and SACO is about 15 s, and that between IACO and SACO is about 5 s. However, when the number of cloud tasks is 700, the difference of makespan between ACO and SACO is over 50 s, and that between IACO and SACO is over 20 s. This signifies that SACO is cost-efficient, that is, a cloud user using our proposed task scheduling algorithm will pay less when there are many tasks to submit to cloud computing environments.

Figure 4 shows the performance results for ACO, IACO, and SACO (the preprocessing time + makespan). It is interesting to note that SACO does not outperform ACO and IACO when the number of cloud tasks is 100. The reason why SACO results in longer makespan is that SACO involves the preprocessing time for slave ants. However, this preprocessing overhead is negligible and it incurs once at the initial stage of task scheduling. As the number of cloud tasks increases, the makespan of SACO is always less than ACO and IACO. This shows the efficiency of the proposed task scheduling algorithm and the effectiveness of the pheromone update scheme based on slave ants.

Figure 5 depicts the performance results for ACO, IACO, and SACO with different experiments. ACO shows the worst performance among the three algorithms. The reason why ACO results in the worst performance is that the leading ants affect the global optimization. In other words, if the leading ants travel long paths, it is hard to reach the shortcut among the paths. The performance results of IACO fluctuate with different experiments since it is based on a random approach. Therefore, IACO sometimes outperforms SACO. However, SACO outperforms ACO and IACO on average. The average makespan of SACO is about 180 s, but that of ACO and IACO is about 220 and 200 s,

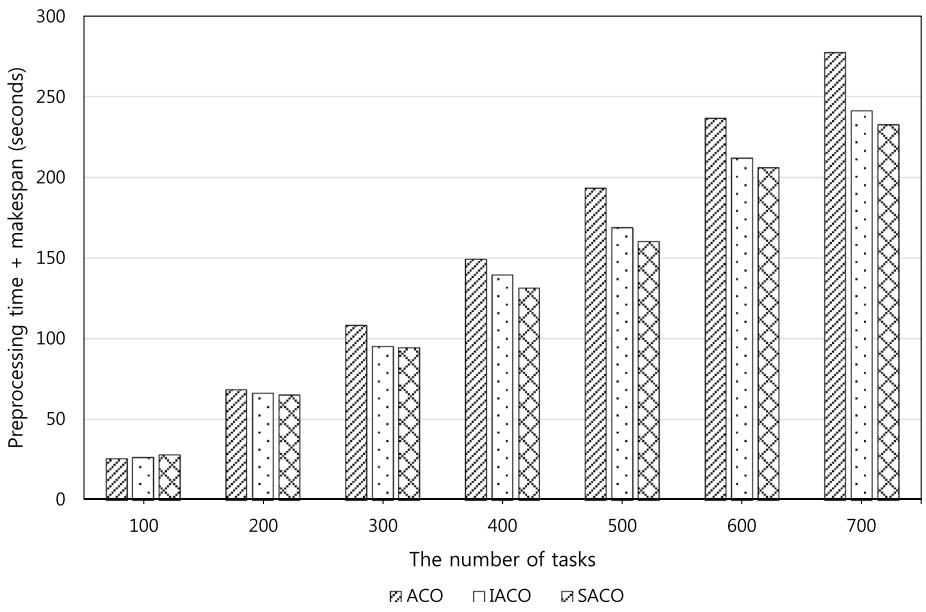

Fig. 4 Performance results of ACO, IACO, and SACO as the number of cloud tasks increases (preprocessing time + makespan)

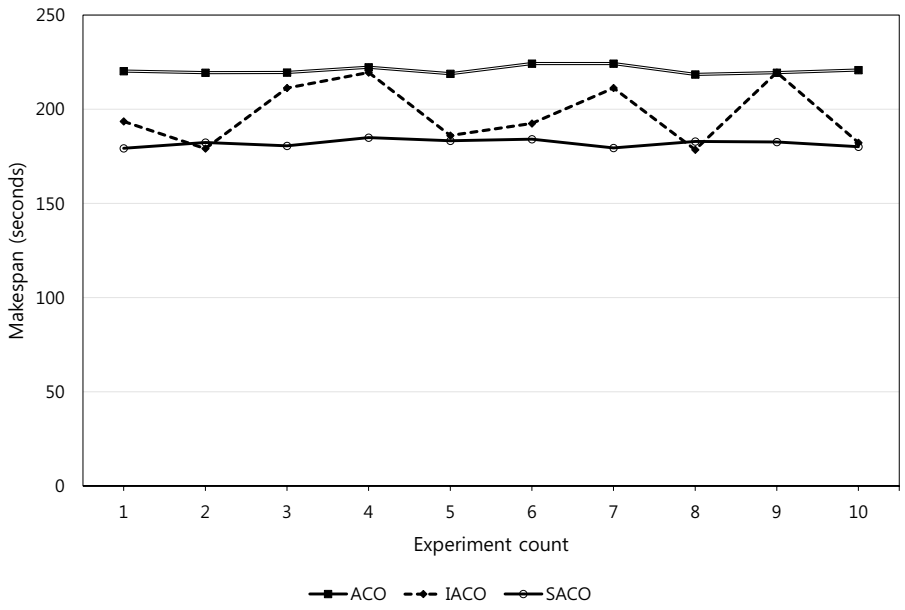

Fig. 5 Performance results of ACO, IACO, and SACO with different experiments

respectively. Overall, the proposed cloud task scheduling algorithm guarantees the globally optimal solution with negligible preprocessing overheads.

Conclusions

In this paper, we proposed a novel ACO algorithm called SACO with slave ants for scheduling tasks in cloud computing environments. We adapt diversification and reinforcement strategies with slave ants to avoid long paths whose pheromones are wrongly accumulated by leading ants. The proposed algorithm introduces minimal preprocessing overheads for slave ants and outperforms the existing ACO based cloud tasks scheduling strategies. Experimental results show that SACO solves the NP-hard problem in an efficient way, while maximizing utilization of cloud servers. Our future work is to consider heterogeneous clusters because the cost is determined also by the type of computing instances.

Abbreviations

ACO: ant colony optimization; VM: virtual machine; QoS: quality of service; SLA: service level agreement; DAG: directed acyclic graph; ANN: artificial neural network; RBFNN: radial basis function neural network.

Authors' contributions

All authors read and approved the final manuscript.

Author details

[1] Department of Computer Science and Engineering, Korea University, Seoul, South Korea. [2] School of Information Technology Engineering, Catholic University of Daegu, Gyeongbuk, South Korea. [3] Department of Game & Multimedia Engineering, Korea Polytechnic University, Siheung, Gyeonggi-do, South Korea.

Acknowledgements

Not applicable.

Competing interests

The authors declare that they have no competing interests.

Availability of data and materials
Data will not be shared because the data can be used with a specific simulation environment.

Consent for publication
Not applicable.

Ethics approval and consent to participate
Not applicable.

Funding
This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2015R1D1A1A01061373).

Publisher's Note

Received: 22 March 2017 Accepted: 1 August 2017
Published online: 09 October 2017

References

1. Zhu W, Lee C (2016) A security protection framework for cloud computing. J Inf Process Syst 12:538–547
2. Maity S, Park J-H (2016) Powering IoT devices: a novel design and analysis technique. J Converg 7:1–18
3. Lim J, Jeong YS, Park D-S, Lee H (2016) An efficient distributed mutual exclusion algorithm for intersection traffic control. J Supercomput. doi:10.1007/s11227-016-1799-3
4. Choi H, Lim J, Yu H, Lee E (2016) Task classification based energy-aware consolidation in clouds. Sci Program 2016:13
5. Motavaselalhagh F, Esfahani FS, Arabnia HR (2015) Knowledge-based adaptable scheduler for SaaS providers in cloud computing. Hum-centric Comput Inf Sci 5:16
6. Dorigo M, Maniezzo V, Colorni A (1996) Ant system: optimization by a colony of cooperating agents. IEEE Trans Syst Man Cybern Part B (Cybern) 26:29–41
7. Tsai CW, Huang WC, Chiang MH, Chiang MC, Yang CS (2014) A hyper-heuristic scheduling algorithm for cloud. IEEE Trans Cloud Comput 2:236–250
8. Tang Z, Jiang L, Zhou J, Li K, Li K (2015) A self-adaptive scheduling algorithm for reduce start time. Futur Gener Comput Syst 43–44:51–60
9. Zheng W, Tang L, Sakellariou R (2015) A priority-based scheduling heuristic to maximize parallelism of ready tasks for DAG applications. In: 2015 15th IEEE/ACM international symposium on cluster, cloud and grid computing, pp. 596–605
10. Malewicz G, Foster I, Rosenberg AL, Wilde M (2006) A tool for prioritizing DAG man jobs and its evaluation. In: 2006 15th IEEE international conference on high performance distributed computing, pp. 156–168
11. Cordasco G, De Chiara R, Rosenberg AL (2011) Assessing the computational benefits of area-oriented DAG-scheduling. In: Jeannot E, Namyst R, Roman J (eds.) Euro-Par 2011 Parallel Processing: 17th International Conference, Euro-Par 2011, Bordeaux, France, August 29–September 2, 2011, Proceedings, Part I, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 180–192
12. Tripathy B, Dash S, Padhy SK (2015) Dynamic task scheduling using a directed neural network. J Parallel Distrib Comput 75:101–106
13. Zuo L, Shu L, Dong S, Zhu C, Hara T (2015) A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing. IEEE Access 3:2687–2699
14. Agrawal P, Rao S (2014) Energy-aware scheduling of distributed systems. IEEE Trans Autom Sci Eng 11:1163–1175
15. Xiao Z, Song W, Chen Q (2013) Dynamic resource allocation using virtual machines for cloud computing environment. IEEE Trans Parallel Distrib Syst 24:1107–1117
16. Tiwari PK, Vidyarthi DP (2016) Improved auto control ant colony optimization using lazy ant approach for grid scheduling problem. Futur Gener Comput Syst 60:78–89
17. Tawfeek MA, El-Sisi A, Keshk AE, Torkey FA (2013) Cloud task scheduling based on ant colony optimization. In: 2013 8th international conference on computer engineering & systems (ICCES), pp. 64–69
18. Mishra JKR (2016) Mitigating threats and security metrics in cloud computing. J Inf Process Syst 12(2):226–233. doi:10.3745/JIPS.03.0049
19. Lim J, Yu H, Gil JM (2017) Detecting sybil attacks in cloud computing environments based on fail-stop signature. Symmetry 9:35
20. Huh J-H, Seo K (2016) Design and test bed experiments of server operation system using virtualization technology. Hum-centric Comput Inf Sci 6:1
21. Lim J, Suh T, Gil J, Yu H (2014) Scalable and leaderless Byzantine consensus in cloud computing environments. Inf Syst Front 16:19–34