Improved Particle Swarm Optimization based Workflow Scheduling in Cloud-fog Environment

Rongbin Xu[1,2], Yeguo Wang[1], Yongliang Cheng[1], Yuanwei Zhu[1], Ying Xie[1,2], Dong Yuan[3]

[1]School of Computer Science and Technology, Anhui University, Hefei, 230601, China
[2] Co-Innovation Center for Information Supply & Assurance Technology, Anhui University, Hefei, 230601, China
[3] School of Electrical and Information Engineering, University of Sydney, Sydney, Australia
xurb_910@ahu.edu.cn; e16201061@stu.ahu.edu.cn; 1399603339,575072641@qq.com; xieying@ahu.edu.cn; dong.yuan@sydney.edu.au

Abstract. Mobile edge devices with high requirements typically need to obtain faster response on local network services. Fog computing is an emerging computing paradigm motivated by this need, which currently is viewed as an extension of cloud computing. This computing paradigm is presented to provide low commutation latency service for workflow applications. However, how to schedule workflow applications for seeking the tradeoff between makespan and cost in cloud-fog environment is facing huge challenge. To address this issue, in current paper, we propose a workflow scheduling algorithm based on improved particle swarm optimization (IPSO), where a nonlinear decreasing function of inertia weight in PSO is designed for promoting PSO to gain the optimal solution. Finally, comprehensive simulation experiment results show that our proposed scheduling algorithm is more cost-effective and can obtain better performance than baseline approach.

Keywords: cloud computing, fog computing, workflow scheduling, PSO.

1 Introduction

The explosive growth of information and data in the Internet age has brought more attention to cloud computing. In recent years, with the continuous popularity of Internet of Things (IoT) technology, traditional network architecture of cloud computing framework is facing great challenge. Currently, a large number of smart IoT devices located at the edge of network require data processing with low latency, location-aware and mobility requirements. To cope with huge number of end-user IoT devices and big data volumes for real-time low-latency applications, Bonomi F et al. [1] first proposed the concept of "fog computing" aiming to extend the application scenarios of computing. Fog computing is a new computing paradigm that maintains the advantages of cloud computing, which can be introduced as an "edge network cloud". Fog resources can provide local computing, storage and network for end-user applications. The location of fog devices is close to end-user applications so that it can process tasks with low-latency in high response. On the other hand, the flexibility and scalability of cloud computing can help fog computing to cope with the growing demands for large-scale computation-intensive business applications when the processing capacity of fog computing are insufficient. It is obvious that fog computing can compensate the shortage of cloud computing [2].

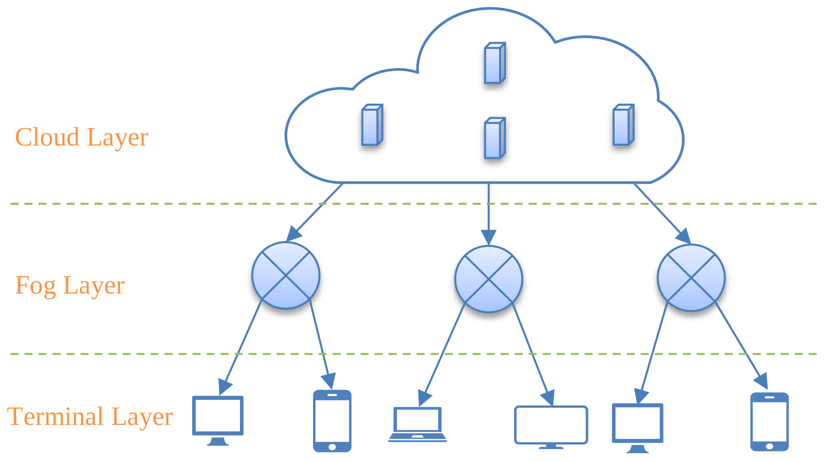


Fig. 1. Architecture of fog computing

In general, the architecture of fog computing is composed of three layers of network structure. As shown in Fig.1, the bottom layer is regarded as terminal layer, which mainly consists of IoT devices such as smartphone, smart wearable devices, smart home appliances and so on. These terminal devices send service requests to upper layer. The middle layer is viewed as fog layer, which contains fog devices with the capabilities of computing and storage such as routers, switches and gateways. These fog devices can be employed by near end-user devices so as to quickly process those time-sensitive tasks. Meanwhile, fog devices also need to connect to the cloud layer for offloading those latency-tolerant and computation-intensive tasks on the demand of users. The upper layer is called as cloud layer, which hosts a large number of heterogeneous virtual machines that provide abundant resources to process the task requests dispatched from fog layer.

Although fog computing has many advantages, it also faces enormous challenges. One of challenges is the allocation of different resources for the scheduling of business tasks [3], especially workflow applications scheduling in fog environment where any workflow application contains many tasks with communication constraint or temporal dependency [4]. In recent years, with fast development of big data, cloud, fog and edge computing, many business workflow applications have been emerged and efficient scheduling of workflow applications has become a hot spot for scholars at home and abroad. For example, the video intelligent surveillance application typically is a low latency workflow application which is composed of five modules [5], including motion detector, object detector, object tracker, user interface and pan, tilting, and zoom control (PTZ). In such application, the motion detector module and PTZ control module are normally executed in the fog nodes, and the user interface is always in the remote cloud. The remaining two modules are executed in the fog nodes or cloud nodes according to the decision-making policies. Therefore, how to design an efficient workflow scheduling algorithm in a cloud and fog environment is crucial, which can highly improve the quality of service (QoS) for providing better user experience.

In this paper, we mainly focus on workflow scheduling in cloud-fog environment and present our scheduling method based on improved particle swarm optimization (IPSO) for workflow applications. The update way of the inertia weight in original PSO is changed by a novel nonlinear decreasing function, which can balance and adjust the search capability of particles in search process. Then a scheduling algorithm is designed by considering the actual problem of workflow applications in cloud-fog environment.

The remainder of this paper is organized as follows. Section 2 presents the related work. Section 3 describes the scheduling problem in cloud-fog environment and proposes our solution based on IPSO. Section 4 elaborates the detail experimental settings and results. Section 5 concludes the paper and puts forward some future work.

2 Related Work

There are many existing works regarding the task or workflow scheduling in distributed environment, especially task scheduling under cloud computing platform [6-8]. The authors in [6] present a market-oriented hierarchical scheduling algorithm in cloud workflow. The scheduling algorithm contains two levels, include the service-level scheduling which deals with the Task-to-Service assignment and the task-level scheduling which deals with the optimization of the Task-to-VM assignment. The authors in [7] propose a near-optimal dynamic priority scheduling (DPS) strategy for instance-intensive business workflows that have a larger number of parallel workflow instances. The authors in [8] propose a hybrid PSO algorithm based on non-dominance sort for handling the workflow scheduling problem with multiple objective in the cloud.

Meanwhile, there is a small amount of studies about task scheduling in the cloud and fog environment [9-12]. Hoang et al. [9] first design a fog-based region architecture for providing nearby computing resources. They investigate efficient scheduling algorithms to allocate tasks among regions and remote clouds. Pham et al. [10] propose a cost-makespan aware workflow scheduling for achieving the balance between performance of application execution and monetary cost for using cloud resources. Besides, an efficient task reassignment strategy based on critical path is also presented to satisfy the user-defined QoS constraints. Tang et al. [11] present a mobile cloud-based scheduling method for the industrial internet of things, which views energy consumption as the main optimization objective of the task scheduling problem while taking into account task dependency, data transmission and other constraint conditions. Zeng et al. [12] consider fog computing as support software-defined embedded system. The authors mainly address three issues: 1) how to balance the task workload and computation servers; 2) how to place task images on storage servers; 3) how to balance the I/O interrupt requests from storage servers.

All the above researches mainly focus on task scheduling problem which have not considered task scheduling with temporal dependency and workflow task scheduling based on traditional method in hybrid cloud and fog environment. As we know, task or workflow scheduling in distributed computing environment is viewed as an NP-hard problem. PSO, as one of intelligent meta-heuristic algorithms, has been applied in addressing the scheduling problem in many fields [13]. However, there are only very few researches about workflow application scheduling problem based on PSO in cloud-fog environment. Thus, this paper will consider PSO-based scheduling algorithm as our basic model.

3 Problem Formalization and Solution

Workflow application scheduling in cloud-fog environment is defined as the formulated problem of assigning computing resources with different processing abilities to the tasks of workflow application, which can minimize the makspan and cost of workflow application scheduling. To formulate this issue, we apply directed acyclic graph (DAG) to represent a workflow application. In addition, a PSO based scheduling algorithm is proposed for solving the mapping process between tasks and computing resources. Then we also present the solution for the proposed issue in this section.

3.1 Problem formalization

In general, a workflow application is composed of a set of tasks, which has similar structure to DAG. So we employ the workflow application by DAG as shown in Fig.2. We denote a DAG as $G = (T, E)$, where T indicates a set of tasks and E represents the set of temporal dependency or communication constraint between pairs of tasks. Specifically, each task $t_s \in T(T = \{t_1, t_2, \cdots, t_n\})$ has its own computation workload cw_s. Accordingly, each directed edge $e_{ij} =< t_i, t_j >\in E$ means that t_j cannot be executed until t_i is completed and e_{ij} has its nonnegative weight value cv_{ij} which represents the communication data transferring t_i to t_j. The task t_i without direct predecessors is denoted as T_{start} and the task t_j without direct successors is denoted as T_{end}. Here, we assume that a task cannot be implemented until all the direct precedent tasks have been completed.

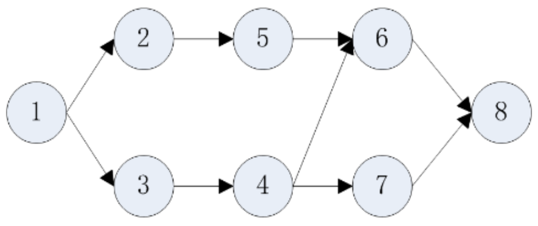


Fig. 2. A simple DAG example of workflow application

A. Makespan

In addition, computing resources in cloud-fog environment are divided into two types, i.e., fog servers and cloud servers. For a workflow task, it is either processed by fog servers or cloud servers. If the task t_s is submitted to a server, the execution time of t_s can be calculated as follows:

$$T_{t_s}^{l} = \frac{cw_s}{\delta_l} \tag{1}$$

where δ_l represents the processing rate of the computing server l. Let $\mathrm{ct}\left(e_{ij}^{l}\right)$ be the commutation time for transferring data from t_i to t_j, which is defined by

$$\mathrm{ct}\left(e_{ij}^{l}\right) = \frac{cv_{ij}}{B} \tag{2}$$

where B in equation (4) is the network bandwidth between two servers. Duo to task t_s will be allocated to one server, this task has its start time ST_{t_s} and finish time FT_{t_s}, which are represented by equation (3) and equation (4), respectively.

$$ST_{t_s} = \max\{ET_{t_p} + \mathrm{ct}\left(e_{ps}^{l}\right),\ t_p \in pre(t_s)\} \tag{3}$$

$$ET_{t_s} = ST_{t_s} + T_{t_s}^{l'} \tag{4}$$

where $pre(t_s)$ denotes the set of direct predecessors of task t_s.
The time period from the start of the first task to the completion of the last task is called the makespan of the entire workflow, which is calculated by equation (5).

$$T_{total} = \max\{ET_{t_s},\ t_s \in \mathrm{T}\} \tag{5}$$

B. Economic cost

The corresponding computation cost of server l is

$$C_{t_s}^{l} = p_l * (ET_{t_s} - ST_{t_s}) \tag{6}$$

where p_l is the unit price of server l. In this paper, we only consider the computing cost of renting servers. So the total cost can be denoted as C_{total}, which is the computing cost of servers. That is

$$C_{total} = \sum_{l=1}^{m} \sum_{s=1}^{n} C_{t_s}^{l} \tag{7}$$

where m and n indicate the number of servers and the number of tasks, respectively.

C. Objective function

Based on the total makespan and cost which have been determined above, the objective function of this paper can be defined as follows:

$$f = 0.5 * T_{total} + 0.5 * C_{total} \tag{8}$$

Equation (8) takes two main factors into consideration, which can help to maintain the balance between makespan and economic cost.

3.2 Solution

In this subsection, we further discuss IPSO and IPSO-based workflow scheduling algorithm in cloud-fog environment.

A. Improved Particle swarm optimization algorithm

The original particle swarm optimization (PSO) algorithm is derived by the study of predation behaviors of flock birds, which is an intelligence evolutionary computing technology. The basic idea of PSO is to search the optimal solution through the cooperation and information sharing among individuals in a group. Suppose that one population has N particles and the searching space is D dimensional. For a particle $P_i(i = 1,2,\cdots,N)$, it has two typical parameters, i.e., the position $X_i = (x_{i1}, x_{i2}, \cdots, x_{iD})$ and the velocity $V_i = (v_{i1}, v_{i2}, \cdots, v_{iD})$. The optimal position of particle individual is represented as $pbest$ and the global optimal position of the whole population is denoted as $gbest$. In k^{th} iteration of PSO, the velocity and position of particle will be updated by the following two equations:

$$V_i^k = w^{(k)} \cdot V_i^{k-1} + c_1 \cdot r_1 \cdot \left(pbest_i - X_i^{k-1}\right) + c_2 \cdot r_2 \cdot \left(gbest - X_i^{k-1}\right) \tag{9}$$

$$X_i^k = X_i^{k-1} + V_i^k \tag{10}$$

where c_1 and c_2 denote to learning factor. r_1 and r_2 are random numbers from the range of [0,1]. $w^{(k)}$ is called inertia weight that influences search capability of particles. The inertia weight w in the original PSO is a liner decreasing function, which is not conducive to balance the global and local search capabilities of particles. Thus, a novel update method in this paper is designed as follows:

$$w^{(k)} = w_{end} + (w_{ini} - w_{end}) \cdot \sin\left(\frac{\pi}{2}\sqrt{\left(1 - \frac{k}{Tmax}\right)^3}\right) \tag{11}$$

where w_{ini} and w_{end} is the initial value and ending value of inertia weight w, respectively. $Tmax$ indicates the maximum iteration times in PSO. We can know that w is a nonlinear decreasing function from equation (11). In the early period of IPSO algorithm search, the value of w is large, which facilitates to enhance the global search capability of particles. In the later period of the search, the value of w is small, which is beneficial to the improvement of the local search ability of particles.

B. IPSO-based workflow scheduling algorithm

Here, the solution space of a particle in PSO is employed to represent a task scheduling plan. Each task node in DAG corresponding to a dimension of one particle. In other words, the dimension of a particle encoding is equal to the number of workflow tasks. The solution for each dimension of one particle indicates a service mapping between task and server.

As shown in Fig.2, the DAG of a workflow application contains 8 tasks, thus the corresponding dimension of a particle is also 8. As shown in Table 1, one of the particles is constructed according to Fig.2. The value of this particle in each dimension indicates the ID of server (cloud or fog server) which each task will be assigned to. Thus, the encoding of position for a particle can be seen in Table 2.

Table 1. The mapping between tasks and servers

Task	1	2	3	4	5	6	7	8
Server	4	2	3	1	2	1	4	2

Table 2. The position of each particle encoding

4	2	3	1	2	1	4	2

Table 3. IPSO-based workflow scheduling algorithm

Input: number of tasks $\boldsymbol{TN}$, maximum numbers of iteration $\boldsymbol{Tmax}$, size of particle swarm $\boldsymbol{K}$, a set of cloud and fog servers.
Output: the optimal scheduling solution $\boldsymbol{S_{best}}$

```
1.  for $i$=1 to $K$ do
2.      Randomize the initialization of scheduling plan $S_i$ and search velocity $V_i$;
3.  end for
4.  for $i$=1 to $K$ do
5.      Compute the fitness of the scheduling plan $S_i$;
6.  end for
7.  Select the global optimal scheduling plan of minimum fitness from $K$ scheduling plan;
8.  for $i$=1 to $Tmax$ do
9.      Update the velocity of particle;
10.     Update the scheduling plan according to the velocity;
11.     Redistribute servers for tasks in the scheduling plan;
12.     for $i$=1 to $K$ do
13.         Compute the fitness value of the scheduling plan $S_i$;
14.     end for
15.     Select the global optimal scheduling plan of minimum fitness from $K$ scheduling plan;
16. end for
17. return $S_{best}$.
```

Furthermore, equation (8) is regarded as the fitness function in PSO. Our object is to minimize the fitness value. Based on the above particle encoding and the fitness function, we design an IPSO-based scheduling algorithm for workflow application as shown in Table 3. This algorithm first initializes randomly position (i.e., the scheduling plan) and velocity of all particles, and some other necessary parameters (see lines 1-7). Next, update scheduling plan according to the velocity of current generation and allocate tasks to servers again (see lines 8-11). After that, compute the fitness value for each scheduling plan and update the global optimal scheduling plan (see lines 12-16). Finally, the algorithm returns the optimal scheduling plan (see line 17).

4 Evaluation

In this section, we first describe the experimental environment and settings. Then, the experimental results are discussed and elaborated.

4.1 Experimental Environment and Settings

In this experiment, first, we test our proposed scheduling method using Matlab 2016a software based on the running environment of Intel core i5 3.0 GHz CPU and 8G RAM.

Next, some details about experimental settings are described as shown in Table 3. We specify the performance parameters regarding cloud servers and fog servers where various servers have different processing capabilities. The number of cloud servers and fog servers are 6 and 4, respectively. For tasks in workflow, the computation workload of each task is ranging from 500 to 15000 MI and the I/O data of a task has a size from 100 to 500 MB. In addition, the population size of IPSO is set as 30. The dimension of particle is equal to the number of workflow tasks in a DAG. The learning factor $c_1 = c_2 = 2$. The initial value and ending value of inertia weight $w_{ini} = 0.9$ and $w_{end} = 0.4$, respectively. The value of maximum iteration times $Tmax$ in PSO is set as 100. In order to reduce the impact of experimental uncertainties, finally, each round of experiment running is set to 30 times repeatedly to get average results.

Table 4. Parameter list of cloud and fog servers

Parameter	Processing rate (MIPS)	Processing cost per unit time	Bandwidth (Mbps)	Number of servers
Cloud servers	6000	0.8	1000	2
	4500	0.6	1000	2
	3500	0.4	1000	2
Fog servers	2500	0.3	2000	2
	2500	0.2	2000	2

4.2 Discussion on Experimental Results

To test the influence of different number of tasks on experimental results, workflow is randomly generated by DAG generator, where the number of tasks in a DAG varies from 50 to 250. The results for the total makespan of workflow and economic cost of server rental under different number of tasks are shown in Table 5. We can see that the total economic costs of the two methods are close with each other in the same scale. With regards to the total makespan of workflow, the difference between results of the two methods is obvious. For example, when the number of tasks is 200, the total economic costs of PSO-based algorithm and IPSO-based algorithm are 162.6287 and 160.7947, respectively, while the total makespan of workflow about the two methods are 71.2534 and 63.1079, respectively. However, as a whole, the value of the result of IPSO-based algorithm is always smaller than the value of the result of PSO-based algorithm, regardless of the total economic cost or makespan.

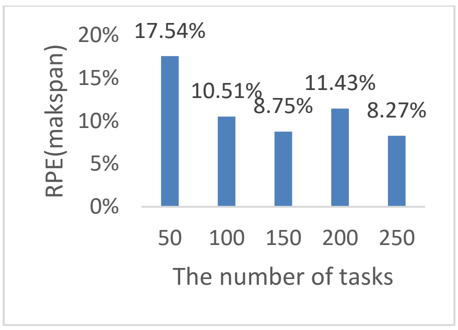
To compare the experimental results regarding the two methods more vividly, relative percentage error is defined as follows:

$$RPE(var) = \left|\frac{RV_1 - RV_2}{RV_2}\right| \times 100\% \tag{12}$$

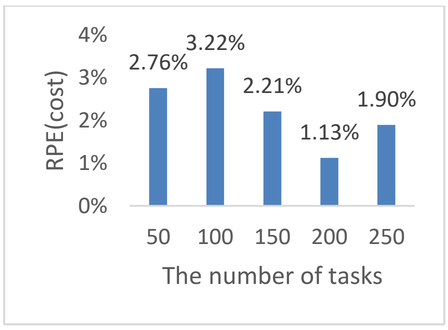
where RV_1 and RV_2 indicate the values of the total makespan or cost of IPSO-based algorithm and PSO-based algorithm, respectively. We can see the differences between results of the two methods from Fig.3 clearly. For one thing, the bar chart in Fig.3(b) shows that the values of $RPE(cost)$ are always small among all the different number of tasks. This phenomenon indicates that our method can guarantee that the cost does not increase compared with the baseline method. Although it cannot significantly reduce the economic cost of renting servers. For another thing, the differences between results of the two methods are easy to be seen in Fig.3(a). For instance, the value of $RPE(makespan)$ is 10.51% when the number of tasks is 100, which shows that our method reduces the 10.51% of the total makespan of workflow application than PSO-based method.

Table 5. The total makespan and economic cost for different number of tasks

Number of tasks	Makespan		Cost	
	PSO-based	IPSO-based	PSO-based	IPSO-based
50	25.2029	20.7818	34.7404	33.7818
100	44.4373	39.7671	75.6929	73.2540
150	56.1236	51.2107	120.6615	117.9896
200	71.2534	63.1079	162.6287	160.7947
250	89.4044	82.0063	205.1188	201.2285



(a)



(b)

Fig. 3. Relative percentage error of the results of two algorithms

From what has been discussed above, we can make a clear conclusion that our proposed method can effectively reduce the total makespan of workflow applications with low latency in cloud-fog environment while the economic cost of renting cloud or fog servers is guaranteed under certain cost constraint.

5 Conclusion and Future Work

The total completion time of the workflow application and economic cost of cloud or fog server rental are two key issues of workflow scheduling in emerging and hybrid cloud-fog environment. In current study, we propose a scheduling algorithm for workflow application based on the improved PSO aiming to maintain the tradeoff between two objectives. In our IPSO, first, the novel update method of inertia weight is designed as a nonlinear decreasing function, which facilitates to balance and adjust the global and local abilities of particles. Next, each particle in PSO is encoded as a scheduling plan according to the number of workflow tasks. Finally, the experimental results show that our proposed scheduling algorithm reduces the overall completion time of workflow compared with the original PSO-based scheduling method while the economic costs of the two methods are close with each other.

In the future, we will further consider workflow scheduling algorithm based on multiple objectives optimization in hybrid cloud-fog environment.

References

1. Bonomi F., Milito R., Zhu J., et al: Fog computing and its role in the internet of things. In: the first edition of the MCC workshop on Mobile cloud computing on Proceedings, pp. 13-16, ACM, Helsinki (2012).
2. Lin Y., Shen H.: CloudFog: Leveraging Fog to Extend Cloud Gaming for Thin-Client MMOG with High Quality of Service. IEEE Transactions on Parallel and Distributed Systems, 28(2), 431-445(2017).
3. Puliafito C., Mingozzi E., Anastasi G.: Fog Computing for the Internet of Mobile Things: issues and challenges. In: 2017 IEEE International Conference on Smart Computing (SMARTCOMP). pp. 1-6, IEEE, Hong Kong (2017).
4. Xu R., Wang Y., Luo H., et al.: A sufficient and necessary temporal violation handling point selection strategy in cloud workflow. Future Generation Computer Systems, 2018. (Online: https://doi.org/10.1016/j.future.2018.03.056).
5. Bittencourt L F, Diaz-Montes J, Buyya R, et al.: Mobility-aware application scheduling in fog computing. IEEE Cloud Computing, 4(2): 26-35(2017).
6. Wu Z., Liu X., Ni Z., et al.: A market-oriented hierarchical scheduling strategy in cloud workflow systems. The Journal of Supercomputing, 63(1), 256-293(2013).
7. Xu R., Wang Y., Huang W., et al.: Near‐optimal dynamic priority scheduling strategy for instance‐intensive business workflows in cloud computing. Concurrency and Computation: Practice and Experience, 29(18), 1-12 (2017).
8. Verma A., Kaushal S.: A hybrid multi-objective Particle Swarm Optimization for scientific workflow scheduling. Parallel Computing, 62: 1-19(2017).
9. Hoang D., Dang T D.: FBRC: Optimization of task scheduling in Fog-based Region and Cloud. In: 2017 IEEE Trustcom/BigDataSE/ICESS, pp. 1109-1114, IEEE, Sydney (2017).

10. Pham X Q., Man N D., Tri N D T., et al.: A cost-and performance-effective approach for task scheduling based on collaboration between cloud and fog computing. International Journal of Distributed Sensor Networks, 13(11), 1550147717742073(2017).
11. Tang C., Wei X., Xiao S., et al.: A Mobile Cloud Based Scheduling Strategy for Industrial Internet of Things. IEEE Access, 6, 7262-7275(2018).
12. Zeng D., Gu L., Guo S., et al.: Joint optimization of task scheduling and image placement in fog computing supported software-defined embedded system. IEEE Transactions on Computers, 65(12), 3702-3712(2016).
13. Pandey S., Wu L., Guru S M., et al.：A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In: 2010 24th IEEE international conference on Advanced information networking and applications (AINA), pp. 400-407, IEEE, Perth (2010).