An ACO-LB Algorithm for Task Scheduling in the Cloud Environment

Shengjun Xue, Mengying Li, Xiaolong Xu, and Jingyi Chen
Nanjing University of Information Science & Technology, School of Computer and Software, Nanjing, China
Email: {sjxue@163.com, lmy3612@163.com, xlxu1988@gmail.com, cjy_1225@163.com}

Shengjun Xue
Nanjing University of Information Science & Technology, Jiangsu Engineering Center of Network Monitoring, Nanjing, China

Abstract—In the face of a large number of task requests which are submitted by users, the cloud data centers need not only to finish these massive tasks but also to satisfy the user's service demand. How to allocate virtual machine reasonably and schedule the tasks efficiently becomes a key problem to be solved in the cloud environment. This paper proposes a ACO-LB(Load balancing optimization algorithm based on ant colony algorithm) algorithm to solve the load imbalance of virtual machine in the process of task scheduling. The ACO-LB algorithm can adapt to the dynamic cloud environment. It will not only shorten the makespan of task scheduling, but also maintain the load balance of virtual machines in the data center. In this paper, the workflow scheduling is simulated in CloudSim. The results show that the proposed ACO-LB algorithm has better performance and load balancing ability.

Index Terms—cloud computing, task scheduling, ACO (Ant colony optimization), ACO-LB, Load Balancing

I. INTRODUCTION

With the rapid changing of business model, the grid computing which emphasizes to solve scientific computing problems has been difficult to solve many problems in the actual business. So far there has not been a commercial product in the grid computing. In 2006, Google proposed the concept of cloud computing for the first time. This computing mode is dedicated to solve business problems, and it just need be equipped with some cheap PC and servers instead of high performance computers. In addition, all of the resources in the cloud are from the same institution, so it will save a lot of unnecessary problems and reduce the difficulty of realization. Therefore grid computing was eventually replaced by this emerging business computing mode. Cloud computing has now become a hot technology in today’s computer industry, it is the development and commercial implementations of Grid computing, parallel computing and distributed computing at the same time.

Cloud computing can use virtualization technology to centralize storage capacity, processing capacity and communication capability in the data center as a shared resource pool which can be configured dynamically. Users just need to pay the corresponding fees and can obtain the corresponding services according to their needs through the Internet, therefore they will no longer need some specific terminal devices. Because cloud computing has a large number of users, the virtual machines in the cloud data center will almost need to handle massive tasks all the time. How to allocate the massive tasks to the virtual machines leased by users and schedule the tasks efficiently has become an important problem need to be solved in the cloud data center.

In general, First Input First Output (FIFO) scheduling policy and its Capacity Scheduler can be used to solve the problem of task scheduling in cloud environment[3]. With the development of cloud computing, there have been many heuristic algorithms to solve the problem of task scheduling in cloud computing, such as genetic algorithm, particle swarm algorithm and immune evolutionary algorithm. These algorithms usually have poor convergence performance and are easy to fall into premature state. Some researchers have used the ant colony optimization algorithm to solve the task scheduling problem, but the study found that the ant colony optimization algorithm is easily trapped into local optimal state and the algorithm has high randomness. Besides most current algorithms ignore the different calculated performance of each virtual machine in cloud data center and the calculation amount of tasks submitted by users also have the difference. In that case, it is easily to cause the load imbalance of virtual machine leased by users, accordingly the overall performance of the cloud data center will be under the influence. Aiming at the shortages of the existing scheduling algorithms, this paper proposes a ACO-LB(Load balancing optimization algorithm based on ant colony algorithm) algorithm. The ACO-LB algorithm will not only shorten the makespan of task scheduling, but also maintain the load balance of virtual machines leased by users in the process of task scheduling.

In addition, considering that most researches are focusing on scheduling independent tasks and ignore the workflow model with priority constraints users may submit, so this paper uses DAG (Directed Acyclic Graph) to research workflow scheduling. Through considering
the temporal and causal constraint for each task to choose the best resources and coordinate the execution of various tasks to obtain the final results.

This paper uses CloudSim [2] as the simulation platform and compares the ACO-LB algorithm with the FIFO scheduling policy, the ACO algorithm and a combination algorithm (ACO-RE) of ACO and roulette wheel algorithm through the simulation experiment on CloudSim to verify the superiority of ACO-LB algorithm.

The contents of this paper are as follows: Section 2 represents related works. In Section 3, we introduce the workflow model. Section 4 mainly introduces the principle and basic steps of the ACO algorithm. In Section 5, we present the proposed ACO-LB algorithm detailedly. In Section 6, we provide a simulation experiment of the proposed algorithm. Section 7 concludes the paper and discusses the direction of future researches.

II. RELATED WORKS

It is well known that the mapping of jobs to the compute resources is an NP-complete problem in the general form [1]. So we need an efficient algorithm for task scheduling in the cloud environment [4]. There are many researches on task scheduling, this section contains a brief overview of some related works which have been done for tasks scheduling.

Most studies in the field of task scheduling aim at reducing the executing time of tasks. R.G. Babukarthik and others [5] presented a Hybrid algorithm based on ACO and Cuckoo search to solve the task scheduling problem and the results shows that the algorithm can reduce the total executing time. S.Z. Zhan [6] proposed an improved Particle Swarm Optimization (PSO) algorithm to reduce the task average running time and raise the rate availability of resources. S. Kaur [7] proposed a meta-heuristic algorithm which minimizes execution time and execution cost, the improved genetic algorithm is developed by merging two existing algorithms for scheduling tasks.

Besides some researches on task scheduling took into considerations users’ constraints and the system load. An Adaptive Hybrid Heuristic based on users’ Qos constraints was proposed by M. Rahman [8] to schedule Data Analytics workflow applications in Hybrid Cloud Environment. An optimized algorithm based on the Fuzzy-GA optimization which can achieve a better balanced load across all the nodes in the cloud was proposed by S. Tayal [9].

For the workflow applications in the cloud, A.K. Bardsiri and S.M. Hashemi [10] have reviewed different types of workflow scheduling algorithms to study various issues and types of them for cloud workflows. S. Pandey [11] presents a particle swarm optimization (PSO) based heuristic to schedule workflow applications to cloud resources which takes into count both computation cost and data transmission cost. Z.J. Wu and others [12] proposed a market-oriented hierarchical scheduling strategy with three representative meta-heuristic in cloud workflow systems and the experimental results show that the over performance of ACO based scheduling algorithm is better than others.

X.T. Wen [13] proposed an improved algorithm based on ACO and PSO to solve the resource scheduling problem which first used ACO algorithm to find out several groups of solutions and then got more effective solutions using PSO algorithm. W. Lin and L.H. Ai [14] proposed a task scheduling policy based on Ant Colony Optimization to minimize the makespan of the tasks submitted to the cloud system. And H. Liu [15] proposed an improved algorithm based on ACO-LB algorithm with various quality of service (QoS) requirements in cloud computing.

Most researches are dedicated to reduce the makespan and executing cost of task scheduling, but they don’t consider the load balance of virtual machines in the cloud data center.

III. PRELIMINARY KNOWLEDGE OF SCHEDULING MODEL

For some tasks submitted by users may exist interdependent relationship, this paper defines workflow as the object of study to solve the problem of task scheduling in cloud computing. Usually the workflow can be described as a Directed Acyclic Graph (DAG) \( G = (T, E) \). In this model: The set of nodes \( T = \{T_1, T_2, T_3, \ldots, T_n\} \) represents the tasks of the workflow, the set of edges \( E = \{(T_i, T_j) \mid T_i, T_j \in T\} \) represents precedence constraints between two tasks in the workflow. If task \( T_i \) has a directed edge pointing to task \( T_j \), \( T_j \) is called the parent task of \( T_i \) and \( T_i \) is called the child task of \( T_j \). In this case \( T_j \) can only be executed until its entire parent tasks have been completed [16]. Figure 1 depicts the basic structure of a set of workflow, it contains ten tasks to be processed and these tasks are labeled from \( T_0 \) to \( T_9 \).

![Fig.1. A workflow model](image)

In this paper, we use \( VM = \{VM_1, VM_2, VM_3, \ldots, VM_m\} \) to indicate the set of virtual machines users leased, \( m \) represents the number of these virtual machines. MIPS denotes the processing speed of \( VM_i \), where MIPS represents millions of instructions per second. In addition, we define a communication matrix \( com = \{c_{ij} \mid c_{ij} \geq 0, 1 \leq i \leq n, 1 \leq j \leq m\} \), \( n \) denotes the number of task nodes, \( m \) denotes the number of processors and \( c_{ij} \) represents the transfer cost of \( T_i \) assigned to \( VM_j \). We also
define a computing matrix \( e_{ij} = \frac{\text{output size}_i}{\text{bandwidth}_j} \) (1) and \( e_{ij} = \frac{\text{Length}_i}{\text{Mips}_j} \) (2).

Where \( \text{output size}_i \) be the output file size of \( T_i \) and \( \text{bandwidth}_j \) be the bandwidth of links between virtual machines. If two tasks are executed on the same virtual machine, then there is no data transfer cost [17].

The makespan we need to process all the tasks on the \( VM_f \) can be expressed as \( E_f (3) \). The total makespan of the whole workflow is the finishing time of the last task in the entire workflow and it can be defined as \( E_{\text{total}} \).

\[
E_f = \sum_{i \in \text{Task}_f} e_{ij} + \sum_{z \in \text{Fiask}} c_{ij}
\]

In (3), Task\(_f\) represents all tasks executed on \( VM_f \) and Fiask represents the father tasks of Task\(_f\). Only when Fiask are not executed at the \( VM_f \), the formula will be workable.

In this paper, we define that
- The ready time of \( VM_f \) is defined as the completed time for all the tasks assigned to the virtual machine before processing the task \( T_i \).
- The ready time of \( T_i \) processed on \( VM_f \) is defined as the maximum value of the ready time of \( VM_f \) and the sum of maximum completed time for \( T_i \)'s all precursor nodes and the communication time for all precursor nodes reaching \( VM_f \).
- The completed time of \( T_i \) processed on \( VM_f \) is defined as the sum of ready time and execution time spent on \( VM_f \).

IV. WORKFLOW SCHEDULING BASED ON ACO-LB

A. Ant Colony Optimization Algorithm

Ant Colony Optimization (ACO) [18, 19] algorithm is a new kind of simulated evolutionary algorithm [20] and it has been successfully applied to several NP-hard combinatorial optimization problems [21]. The ACO algorithm was proposed by Italian scholar M. Dorigo according to food-seeking behavior by ants in 1996 [22]. The ants will release some pheromone on the way they moved, when these ants reach a crossing they never walked, they will choose a path randomly and release some pheromone proportional to the length of path. The follows will follow the trail of the other ants to the food source by sensing the pheromone on the path. As this process continues, most of the ant will be more likely to choose the shortest path with a huge amount of pheromones, on this occasion a mechanism of positive feedback will be formed, this mechanism ensures that the good information can be preserved and ants can find an optimal way finally [23, 24]. In this case, there will be more and more pheromones on this path. As the time on, the amount of information on other paths will be gradually reduced, eventually the path most ants moved to will be the optimal path.

Firstly we set any ants into some randomly selected cities, the number of ants is \( m \) and the number of cities is \( n \). Ant \( k \) \((k=1, 2, 3, \ldots, m)\) will determine the transfer direction according to the concentration of the pheromone on each path in the searching for target city. At first, the ants will randomly select a path because of the minor difference of the pheromone quantity among paths. The tabu list \( \text{tabu}_k \) \((k=1, 2, 3, \ldots, m)\) is to record the path which ant \( k \) has walked and it will adjust dynamically along with the changing movement of the ant.

Let \( P^{k+1}_j (t) \) (4) be the state transition probability of ant \( k \) selecting city \( j \) as the target city at the moment.

\[
P^{k+1}_j (t) = \begin{cases} \dfrac{[\tau_k(t)]^{\alpha}[\eta_k(t)]^{\beta}}{\sum_{j \in \text{allowed}}[\tau_k(t)]^{\alpha}[\eta_k(t)]^{\beta}}, & \text{if } j \in \text{allowed} \\ 0, & \text{else} \end{cases} (4)
\]

In (4), Allowed\(_j\) represents the allowed cities of ant \( k \) in the next step. \( \tau_k(t) \) represents the pheromone of the path between city \( i \) and city \( j \) at time \( t \). At the initial time, the pheromone of each path is equal. \( \eta_k(t) \) is a heuristic function and defined as \( \eta_k(t) = 1/d_{ij} \) \((d_{ij} \) is the distance between nodes\)) to represent the desired transfer degree of ants from the city \( i \) to the city \( j \). \( \alpha \) is the heuristic factor of pheromone, which indicates the relative importance of the track. It reflects the guiding effect of the information accumulated in the track on the ant’s movement. The greater the values is, the more likely the ant chooses the track which other ants passed by. \( \beta \) is expected heuristic factor, which indicates the relative weight of calculation ability. It reflects the importance degree of heuristic information in ant’s choice. The greater the value is, the closer the state transition probability will be to greedy rule.

B. ACO-LB Algorithm

At the moment, the ant colony algorithm which is used to solve the problem of task scheduling in cloud computing only consider how to minimize the time overhead, but it often ignore the load of virtual machine. In the cloud data center, the computer power of virtual machines varies very much, so there will be significant difference among these virtual machines when they perform the same task. In addition there are obvious differences among tasks, so while allocating tasks to the virtual machines, most tasks will be assigned to the virtual machine which has better performance than others. In that case the virtual machine will be overloaded and others will be in idle state. As a result, it will cause resource wastes. The load imbalance of virtual machines
will reduce efficiency of the cloud data center, so in this paper we consider the average load for each virtual machine based on the ant colony algorithm to increase the utilization rate of resources.

The step of ACO-LB algorithm is described as follows:

- **Step 1. Initializing the pheromone and heuristic factor**

  When ant colony algorithm is used for solving some basic problems, the pheromone and desired degree of transfer between two nodes are usually associated with distance etc. But because of the particularity of the cloud, this paper lets the pheromone \( \tau \) and desired degree \( \eta \) proportional to the computing capability of nodes.

\[
\tau_{i,j} = \eta_{i,j} = \frac{MIPS_j}{N} \quad (5)
\]

In (5), \( \tau \) and \( \eta \) represent the pheromone and desired degree of \( T_i \) allocated to \( VM_j \). \( N \) is a constant as coordinate coefficient.

- **Step 2. Calculation of transition probability**

\[
P_{i,j} = \frac{[\tau_{i,j}]^\alpha [\eta_{i,j}]^\beta}{\sum_{x=1}^{n} [\tau_{i,x}]^\alpha [\eta_{i,x}]^\beta} \quad (6)
\]

The possibility of \( T_i \) being allocated to \( VM_j \) is expressed here formulaically; \( n \) represents the number of available virtual machines of the cloud data center.

- **Step 3. Selecting virtual machine by roulette wheel algorithm**

  In this paper, the roulette algorithm is used for solving the transfer probability problem of ants. For each task of workflow, when the probability for each virtual machine being selected is determined, a random number will be emerged in the range of 0 and 1. Then subtracting the probability of first virtual machine being selected from the number, if their difference is less than zero, the virtual machine will be selected, otherwise subtracting the probability of the next virtual machine being selected from the difference again until the subtracted result is less than or equal to 0. Therefore, the virtual machine whose probability is subtracted at last will be selected by the task.

- **Step 4. Updating the pheromone**

  In order to avoid the overmuch residual pheromone submerging the heuristic information, this paper uses (7) to update the residual pheromone when each ant has completed one step or gone through all cities.

\[
\tau_{i,j}(t+1) = (1-\rho)\tau_{i,j}(t) + \Delta \tau_{i,j}(t) \quad (7)
\]

Where \( \rho \in (0,1] \) is the pheromone trail decay coefficient to prevent infinite accumulation of pheromone.

This paper uses Ant-Cycle model which is raised by M.Dorigo, this model will adjust pheromone by global information. The pheromone updating is applied on the visited \( VM_i \) and can be defined as \( \Delta \tau_{i,j}(t) \). At the beginning, the value of \( \Delta \tau_{i,j}(t) \) is set to zero. When the ants complete the scheduling for all tasks, the value of \( \Delta \tau_{i,j}(t) \) will be determined by the following conditions.

1) Updating local pheromone

When an ant completes the scheduling for all tasks, the local pheromone updating is applied on all visited VMs and the value of \( \Delta \tau_{i,j}(t) \) is given by (8).

\[
\Delta \tau_{i,j}(t) = \frac{D}{clock_{i,j}} \quad (8)
\]

Where \( D \) is a constant and \( clock_{i,j} \) represents the completion time of \( T_i \) being allocated to \( VM_j \).

2) Updating global pheromone

When all ants complete the traversal, then find out the best scheduling in this iteration and the value of \( \Delta \tau_{i,j}(t) \) in this scheduling is given by equation (9).

\[
\Delta \tau_{i,j}(t) = \frac{D}{bestclock_{i,j}} \quad (9)
\]

Where \( D \) is a constant and \( bestclock_{i,j} \) represents the completion time of \( T_i \) being allocated to \( VM_j \) in the best scheduling.

- **Step 5. The realization of load balancing**

  Because of the particularity of the workflow model, if the tasks in the same layer are assigned to the same virtual machine, other tasks in the same layer will be into long waiting period while one task is executing. In that case, it will lead other virtual machines which have accomplished all task on them to be in the idle state and cause the waste of resources. In order to solve this problem, once ACO-LB algorithm detects that the tasks in the same layer have selected the same virtual machine and been into a waiting period, it will select other idle virtual machines again according to equation (6).

In addition, in order to prevent all tasks in different layer being assigned to the virtual machine with good computing power and making this virtual machine being overloaded. So this paper will come up with a pheromone adjustment factor \( PC \), and the value of \( PC \) is given by (10).

\[
PC = 1 - \frac{(E_j - E_{avg})}{\sum_{j=VM} E_j} \quad (10)
\]

\[
\tau_{i,j}(t+1) = ((1-\rho)\tau_{i,j}(t) + \Delta \tau_{i,j}(t)) \times PC \quad (11)
\]

\[
\tau_{i,k}(t+1) = \tau_{i,k}(t) \times PC \quad (12)
\]
So if \( VM_j \) is overloaded in the previous iteration, \( E_j \) will be larger than other virtual machines and the adjusting factor \( PC \) of \( VM_j \) will be smaller than others. So in the next iteration the probability of \( T_i \) assigned to \( VM_j \) will be relatively low. After multiple iterations, the algorithm can guarantee load balancing of each virtual machine and improve the execution efficiency of system.

**Algorithm: ACO-LB**

<table>
<thead>
<tr>
<th>Input: ten tasks in workflow model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output: the makespan of the workflow model, optimum solution</td>
</tr>
</tbody>
</table>

1. initialize the \( \alpha, \beta, \rho \), the number of ants and iterations
2. While \( NC < NC_{max} \)
3. do for each ant
4. for each task
5. do for each virtual machine
6. do calculate the transition probability by Equation (6)
7. do select \( VM \) by roulette wheel algorithm
8. End for
9. do if the tasks in the same layer are assigned to the same virtual machine
10. then select another \( VM \) again by Equation (6) and roulette wheel algorithm
11. Update local pheromone by Equation (8);
12. do for each virtual machine
13. do if the virtual machine is selected by the task
14. Then adjust local pheromone by Equation (11)
15. else
16. adjust local pheromone by Equation (12)
17. end for
18. end for
19. end for
20. Find the best solution in this iteration
21. \( NC++ \)
22. Update global pheromone by Equation (9)
23. End while
24. Find optimum solution and bind the tasks in workflow to virtual machines with the optimum solution

**Fig. 2** describes the implementation steps of ACO-LB algorithm

**V. EXPERIMENT AND ANALYSIS**

**A. Experimental Environment and Parameter Settings**

We compare our ACO-LB algorithm with the First-Come-First-Served (FCFS) scheduling strategy, the basic ant colony algorithm (ACO) and a combination algorithm (ACO-RE) of ACO and roulette wheel algorithm to test whether the ACO-LB algorithm has more superior performance and load balancing ability than others. This paper uses CloudSim to simulate a cloud data center and overrides the DatacenterBroker class and the Cloudlet class to realize the simulation of these algorithms above. Besides, this paper designs a workflow model to test the validity of ACO-LB algorithm, table 1 shows that the parameter values of ten tasks in the workflow model. The ACO-LB parameter are setting as: \( \alpha = 0.7, \beta = 0.7, \rho = 0.3, \ m = 50, NC_{max} = 50 \).

<table>
<thead>
<tr>
<th>Cloudlet Id</th>
<th>Length</th>
<th>Output size</th>
<th>Required Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>63572</td>
<td>300</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>32830</td>
<td>300</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>50274</td>
<td>300</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>23532</td>
<td>300</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>74356</td>
<td>300</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>63443</td>
<td>300</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>34345</td>
<td>300</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>64353</td>
<td>300</td>
<td>3,4</td>
</tr>
<tr>
<td>8</td>
<td>76433</td>
<td>300</td>
<td>5,6</td>
</tr>
<tr>
<td>9</td>
<td>46435</td>
<td>300</td>
<td>7,8</td>
</tr>
</tbody>
</table>

© 2014 ACADEMY PUBLISHER
This paper simulates a data center in CoudSim and creates four virtual machines; these virtual machines are labeled from $VM_0$ to $VM_3$. The parameters of these virtual machines are set as in Table 2. We assume that other properties of these virtual machines are the same and the bandwidth of links between virtual machines are equal.

### TABLE 2

<table>
<thead>
<tr>
<th>VM ID</th>
<th>MIPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>335</td>
</tr>
<tr>
<td>1</td>
<td>310</td>
</tr>
<tr>
<td>2</td>
<td>489</td>
</tr>
<tr>
<td>3</td>
<td>560</td>
</tr>
</tbody>
</table>

### B. Analysis of Experiment Results

In this experiment, we use the FCFS scheduling strategy, the ACO algorithm, the ACO-RE algorithm and our ACO-LB algorithm for task scheduling. The task allocation schemes of these four experiments are shown in Table 3, where ACO algorithm, ACO-RE algorithm and ACO-LB algorithm adopt the data of the global optimal solutions.

#### (c) Task Allocation Schemes of ACO-RE

<table>
<thead>
<tr>
<th>Cloudlet ID</th>
<th>VM ID</th>
<th>Finish time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>113.52</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>172.15</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>219.33</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>214.17</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>397.10</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>349.07</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>283.66</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>515.02</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>508.38</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>600.94</td>
</tr>
</tbody>
</table>

#### (d) Task Allocation Schemes of ACO-LB

<table>
<thead>
<tr>
<th>Cloudlet ID</th>
<th>VM ID</th>
<th>Finish time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>113.52</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>183.66</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>203.30</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>256.90</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>335.72</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>316.59</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>317.09</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>470.32</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>456.57</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>556.24</td>
</tr>
</tbody>
</table>

From table 3a we can see that in FIFO scheduling strategy the tasks are assigned to virtual machines in order. The efficiency of this method is very low, because it does not take into account the processing capacity of each virtual machine. In that case, the virtual machines with low performance will be heavily loaded, and there will take a long time to complete the whole process of task scheduling. Table 3b shows that all the tasks are assigned to $VM_3$ which has the best performance after several iterations in the ACO algorithm. Therefore $VM_3$ is under the heavy load and the makespan of tasks is too large. Table 3c shows that in the ACO-RE algorithm, there are a large number of tasks assigned to $VM_i$ and $VM_j$ which have better performance than others, besides $VM_i$ is in the idle state because there is no task on it. In that case, this can lead to resource wastage. From table 3d we can see that each virtual machine obtained some tasks according to the performance of them, ACO-LB algorithm not only doesn't lead to resource wastage, but also improve the efficiency of the system.
The finish time of each task in the workflow with four different algorithms is shown in Fig.3. From figure 3 we can see that the finish time of each task with FIFO and ACO algorithm is larger than ACO-RE and ACO-LB algorithm. Before executing the seventh task, the finish time of each task with ACO-RE and ACO-LB algorithm is rough, but while executing the last three tasks, the processing time with ACO-LB algorithm is significantly less than the ACO-RE algorithm.

In the future, there are several points that deserve further investigation. First, this paper only presents one workflow model with single structure and correlation to verify the advantages of the ACO-LB algorithm, so we will propose different types of workflow models in the future to verify the reliability of the ACO-LB algorithm. Second, this paper only considers minimizing the execution time of tasks and ignores the cost problem existing in reality, so we will realize high efficiency and low cost at the same time.

ACKNOWLEDGMENT
This work is partly supported by National Natural Science Foundation of China (Grand Nos. 41275116) and Jiangsu Economic and Information Technology Commission project of China (Grand Nos.{2011}1178).

REFERENCES


Shengjun Xue earned his Ph.D. and MS degrees in computer application technology at Wuhan University of Technology in 2002. From 2007, he is a postdoctoral researcher at Purdue University. His major field of study is computer network, intelligent transportation and cloud computing.

He has been worked in Nanjing University of Information Science&Technology since 2007. He has published more than 90papers in journals and conferences, and he has written 6 books. Now he is researching the cloud computing.

Prof. Xue is the senior member of High Performance Computing and has served as the president of the international conference, such as RSETE, CSSS.

Mengying Li was born in Nanjing, China, in 1992. She will be a master in Nanjing University of Information Science& Technology. Her research interests include cloud computing and cloud scheduling.

Xiaolong Xu was born in Nantong, China, in 1988. He will receive his master soon at Nanjing University of Information Science & Technology. And he will be a PH.D student at Nanjing University. His research interests include cloud computing, cloud scheduling and cloud storage.

Jingyi Chen was born in Dongtai, China in 1991. She will be a master in Nanjing University of Information Science& Technology. Her research interests include cloud computing and meteorology.