International Journal of Hybrid Information Technology
Vol. 9, No.12 (2016), pp. 367-386
http://dx.doi.org/10.14257/ijhit.2016.9.12.33

Delay-Bounded Associated Tasks Scheduling Based on Hierarchical Graph Model in the Cloud

Yingchi Mao[1], Haishing Zhong[1], Longbao Wang[1] and Xiaofang Li[2]

[1] College of Computer and Information Engineering, Hohai University, Nanjing, 211100, China
[2] College of Computer and Information Engineering, Changzhou Institute of Technology, Changzhou, China
maoyingchi@gmail.com, wlb620@163.com, lixf@czu.cn

Abstract

Cloud computing can provide the dynamic and elastic virtual resources for the users to execute the large-scale computing tasks. It has become the hot spot in the academic and industry fields. Task scheduling is one of the most important issues in the Cloud. In the Cloud systems, the goal of the tasks scheduling is to spread the workload among the computing nodes and maximize the utilization while the total execution time is within the specific delay bound. At present, almost scheduling algorithms focus on the single task dispatch in the Cloud. Unfortunately, there is little research on the associate tasks scheduling considering the deadline bound. In this paper, two hierarchical task models were discussed and the corresponding associated task scheduling algorithms based on delay-bound constraint (ATS-DB and SAH-DB) were proposed. The associated tasks and the task execution order were represented by one directed acyclic graph (DAG). The proposed hierarchical task models can improve the task execution concurrency. Extensive experimental results demonstrated that the proposed scheduling algorithms, ATS-DB and SAH-DB, can reduce the execution cost and improve the resource utilization within the user-expected delay bound.

***Keywords**: Cloud computing, delay bound, associated task scheduling, hierarchical graph model*

1. Introduction

Cloud computing is emerging as a new paradigm of large-scale distributed computing, which support convenient and on-demand network access to a shared pool of computing resources. Cloud computing can many advantages, including transparency of resources, flexibility, location independence, reliability, and so on [1]. To provide these facilities, the tasks should be scheduled to the appropriate resources in order to achieve maximum performance in minimum time.

Task scheduling is one of most important issues in the cloud computing environments. In the cloud systems, the goal of the scheduling algorithms is to spread the workload among the computing nodes and maximize the utilization while minimizing the total task execution time. The task scheduling process is performed in two stages. In the first stage, the scheduler allocates resources to the cloud as requested by an application. In the second stage, the incoming tasks are assigned to the appropriate virtual nodes in an effort to balance loads within the virtual nodes [2]. To minimize the total execution time of all incoming tasks, the scheduling algorithm should reduce the amount of transferred data and ensure the load balance [3].

At present, the existing task scheduling algorithms have been proposed to meet the different goals considering the different constraints. For example, First Come First Served algorithm (FCFS), Round Robin algorithm (RR), Min-Min algorithm and Max-Min algorithm can achieve good scheduling performance for non-associated tasks. Concerning the delay of the associated tasks scheduling in cloud computing, two hierarchical task models were discussed and the two corresponding associated task scheduling algorithms based on delay-bound constraint were proposed.

The contribution of this paper are as follows:

1) At present, almost research on task scheduling in cloud computing focus on the single and independent task scheduling in order to reduce the complexity of scheduling problem. In this paper, we proposed two associated task models for the associated task scheduling considering the real application requirements in cloud computing.

2) Based on the dependency of associated tasks, a directed acyclic graph (DAG) was presented to describe the execution order for the associated tasks. Also, considering the parallel structure of sub-DAG, we proposed the hierarchical task graph to decompose the associated tasks, which can improve the tasks execution concurrency and reduce the execution cost.

3) In order to execute all of the associated tasks in the specific delay-bound, we proposed the concept of tasks processing capacity and the corresponding calculation method, and further established the mapping between the task processing capacity and execution time.

The rest of the paper is organized as follows: Section 2 addresses the existing task scheduling algorithms. In section 3, the associated tasks scheduling model is defined. The CPM-based (Critical Path Method-based) scheduling method is presented in Section 4. The details of proposed associated tasks scheduling algorithm, ATS-DB, is discussed in Section 5. Structured-based tasks hierarchy method and the corresponding scheduling algorithm, SAH-DB, were proposed in Section 6. Extensive experiments results are presented in Section 7. Section 8 concludes the paper and discussed some future work.

2. Related Work

2.1. Independent Tasks Scheduling

Independent task scheduling has been investigated both in theory and practice with applications [19]. The most commonly deployed scheduling algorithm is First-Come-First-Served (FCFS) or variations. FCFS suffers from head of queue blocking and starvation issues. Two typical variations are backfilling and gang scheduling. Capability scheduler [12] supports multiple task queues which are allocated a fraction of the total resource capability. Fair scheduler [13] allow jobs to obtain the resource fairly with the passage of time. The dynamic priority parallel task scheduler for Hadoop was presented [14]. It allows users to control their allocated capacity by adjusting their spending over time. But their work don't consider the user's QoS requirements. Next, Dong *et al*. proposed a mixed real-time task scheduler [15]. They try to meet users' QoS demands, but only consider the time factor. Another model used to estimate the individual task completion times given a particular resource allocation was presented in [16]. They uses these estimated values as the task's performance evaluation. K. Kc presented a scheduler to meet the deadlines of tasks in Hadoop [17]. A load-aware scheduler for heterogeneous environments with dynamic loading was proposed, which can modify slot number based on the task execution environment [18].

In fact, the real application, there exist a large of number of associated tasks. These tasks have dependency and the execution order. If one of the associated tasks has delayed beyond the execution time bound, the execution of its successors will result in the delay affected by its predecessors [4].

2.2. Associated Task Scheduling

Because the cloud computing environment is a heterogeneous system, and the computing performance of each node are quite different, the online scheduling algorithms are more appropriate for the cloud computing. However, the existing research online mode algorithms pay little attention on the associated tasks scheduling. There has been some recent research on the time-bound constraint for various scheduling strategies for parallel tasks. This work fall in two categories. In decomposition-based strategies, the parallel tasks is decomposed into a set of sequential tasks and they are scheduled using existing sequential scheduling algorithms. In general, decomposition-based strategies require explicit knowledge of the structure of the DAG off-line. In non-decomposition based strategies, the program can unfold dynamically since no advent knowledge is required. Du *et al*. adopted the directed Acyclic Graph to represent the dependency among the associated tasks, and applying cluster algorithm to reduce the resource searching cost, and furthermore, shorten the execution time for all of the tasks [5]. Wilmer *et al*. presented the scheduling algorithm to select the appropriate resource, by setting the completion time in each tasks layer with the decomposition-based method [6].

In this paper, a structure-analysis hierarchical task models was discussed and the associated task scheduling algorithms based on delay-bound constraint (SAH-DB) was proposed. The proposed hierarchical task models can improve the task execution concurrency, and the proposed scheduling algorithms can reduce the execution cost within the user-expected delay bound.

3. Task Hierarchical Model

3.1. Tasks DAG Model

In this paper, all of the mentioned tasks are referred to the computing tasks. The association of the tasks are reflected on the execution order of tasks. That is to say, each one task has its own previous related tasks and/or the successive related tasks. For each one task, it is ready to be executed just after all of its predecessors have been executed. In order to explicitly describe the correlation of the associated tasks, we consider a general model for associated tasks, namely the DAG model. Each task is characterized by its execution pattern, defined by a directed acyclic graph (DAG). In a DAG, each node represents an associated task and each directed edge represents dependency between tasks. As shown in Figure 1, the directed edge connected node 1 and 3. It represents that the task t_1 is the predecessors of task t_2. Task t_2 should execute after the tasks t_1 has finished.

To order explicitly describe a group of associate tasks, their dependency, and communication cost, a three tuple is applied to represent it.

Definition 1: $G=\{T,V\}$ is defined to describe a group of the associate tasks and their correlations.

1) Given n associated tasks, set $T=\{t_1,t_2,...,t_n\}$ is all of associated task in a set T. $\forall t_i \in T,\ i \in [1,\mathrm{n}]$, $t_i=(\tau_{i\_id},\tau_{i\_length})$. τ_{i_id} and τ_{i_length} represent the id and the length of task t_i, respectively. In the same condition, the tasks with longer length will cost the more execution time.

2) V represents the set of execution order for the adjacent tasks $< t_i, t_j >$, in which t_i is the predecessors of task τ_j

To clear illustrate the Definition 1, Figure 1 shows a DAG containing 23 associated tasks. In this DAG, the first executed task is t_1, and the last task is t_{23}. The label of the edge represents the communication cost between two adjacent tasks.

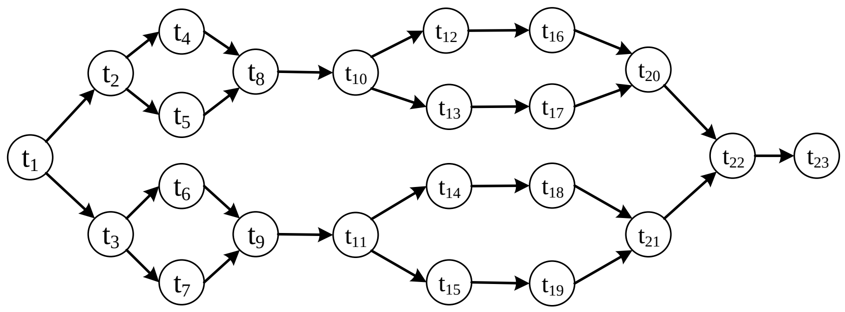

Figure 1. An Example of the Associated Tasks (DAG)

In the associated tasks model, each task is the atomic task, the basic execution unit in the cloud computing. The execution of the associated tasks has dependency order. Only the predecessors have been finished, the successors just are executed.

In the Cloud computing system, each node has different capacities in the computing, storage, and network. For the same task, due to the different capacities for the computing nodes, they provide the different execution time. On the other hand, the different tasks consume the different resources in the same computing node. Therefore, it should give the unified resource description for the different resources in the different computing nodes.

Definition 2: Resources Set. In the Cloud computing system, there are m resources, including computing, storage, and networks. The resource set is represented as $R = \{r_1, r_2, ..., r_j, ..., r_m\}$, in which $r_j \, (j \in [1, m])$ denotes the resource in the j^{th} computing node within the set R.

For $\forall r_j \in R$, $r_j = [r_{j\_id}, r_{j\_cpu}, r_{j\_memory}, r_{j\_disk}, r_{j\_bw}]$, which represents the *id, computing capacity, memory, disk storage capacity, and network bandwidth*, respectively.

Definition 3: The associated tasks scheduling set TS is defined as a set of all possible tasks scheduling. TS can be described as $TS = \{TS_1, ..., TS_i, ..., TS_p\}$, TS_i refers to one of possible tasks scheduling. p means the number of possible tasks scheduling.

For one task scheduling $TS_j \in TS$ can be denoted as a three tuple, $TS_j = \{< id_i^k, \tau_i^k, c_i^k > | 1 \le i \le n, 1 \le k \le m\}$. For one task $t_i, (1 \le i \le n)$, id_i^k means the task t_i utilizing the resource k during the task scheduling. τ_i^k denotes the execution time of the task t_i executed on the resource k. c_i^k is the cost of the resource k executing the task t_i.

3.2. Problem Statement

Given one set of the associated tasks $T = \{t_1, t_2, ..., t_n\}$, and the corresponding tasks scheduling $TS_j \in TS$. To solve the associated task scheduling problem based on delay-bound constraint, it should propose an effective scheduling scheme to minimize the execution cost and satisfy the execution delay bound constraint. That is to say,

$$\begin{cases} Cost_{ts} = \min\left(\sum_{i=1}^{|V|}\sum_{k=1}^{m} id_i^k C_i^k\right) \\ T_{exe\_ts} \le T_{deadline} \\ id_i^k = \{0,1\} \\ \sum_{k=1}^{m} id_i^k = 1 \end{cases} \quad (1)$$

$Cost_{ts}$ is the total execution cost of all associated tasks, T_{exe_ts} is the total execution time and $T_{deadline}$ is the user-specific finish time of all associated tasks. From the formula (1), for one task $t_i\ (1 \le i \le n)$, it just chooses only and only one appropriate resource to execute the corresponding task. However, multiple tasks can be run in the same resource simultaneously. All of the associated tasks should be finished before the deadline.

3.3. Example

Figure 1 illustrates 23 associated tasks in a DAG. In that DAG, the first executed task is t_1, and the last task is t_{23}. The label of the edge represents the communication cost between two adjacent tasks. Assume that there are many resources are appropriate for the computing in the cloud. For each associated task, the corresponding available resources are listed in Table I.

Table 1. Available Resources List

Task	Tasks execution time and cost in the available resources	Task	Tasks execution time and cost in the available resources
t_1	<4, 56>	t_{13}	<5, 57>, <7, 46>, <8, 54>
t_2	<6, 68>, <7, 52>, <15, 100>	t_{14}	<10, 47>, <13, 55>, <18, 45>
t_3	<7, 60>, <8, 58>, <10, 90>	t_{15}	<13, 56>, <14, 32>
t_4	<5, 87>, <6, 50>, <15, 86>, <20, 105>	t_{16}	<6, 65>, <8, 60>, <10, 82>
t_5	<5, 68>, <6, 55>, <13, 79>	t_{17}	<5, 70>, <9, 85>, <13, 60>
t_6	<6, 78>, <8, 96>, <10, 106>, <13, 97>	t_{18}	<4, 65>, <6, 45>, <10, 40>
t_7	<3, 109>, <4, 65>, <8, 79>	t_{19}	<5, 60>, <9, 75>, <13, 68>
t_8	<7, 65>, <13, 102>, <4, 98>	t_{20}	<2, 100>, <3, 90>, <10, 87>, <13, 95>
t_9	<5, 76>, <8, 64>, <10, 78>	t_{21}	<5, 64>, <9, 75>
t_{10}	<7, 85>, <9, 60>, <13, 50>, <15, 65>	t_{22}	<4, 32>, <7, 65>, <9, 54>
t_{11}	<5, 98>, <7, 65>, <11, 78>	t_{23}	<3, 87>, <4, 49>, <11, 50>
t_{12}	<8, 56>, <10, 40>, <16, 78>, <4, 65>		

4. CPM Scheduling Alogrithm

In order to minimize the execution time of all associated tasks without considering the execution cost, the naïve approach is to adopt the Critical Path Method (CPM) [12] to compute the minimal execution time $T_{\min}$. Adopting CPM scheduling algorithm, it can get the critical path of the DAG based on the available resources in Table I, as shown in Figure 2.

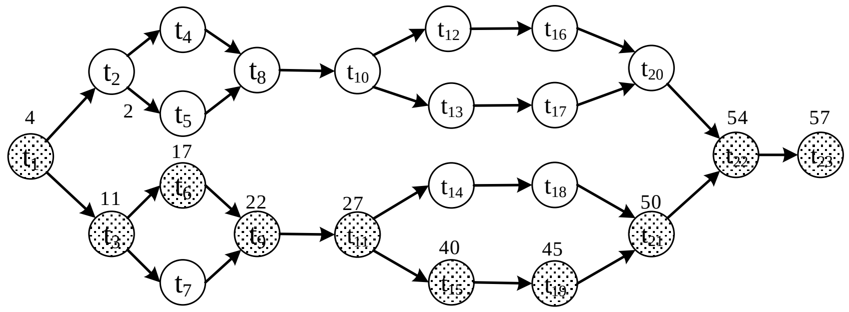

Figure 2. CPM for the Associated Tasks

According the results of CPM algorithm, the nodes in the critical path is $cp=\{t_1,t_3,t_6,t_9,t_{11},t_{15},t_{19},t_{21},t_{22},t_{23}\}$. The minimal execution time of all associated tasks is: $T_{\min}=\sum_{i=1}^{n} id_i^k \tau_i^k = 57$, $n=|V|$, $id_i^k=\begin{cases}1, & if \;\; i\in cp\\ 0, & esle\end{cases}$. The corresponding cost for execution tasks is $Cost_{CPM}=\min\left(\sum_{i=1}^{|V|}\sum_{k=1}^{m} id_i^k c_i^k\right)=1638$, $id_i^k=\begin{cases}1, & if \;\; i\in cp\\ 0, & esle\end{cases}$. The CPM scheduling results are shown in Table II.

Table 2. CPM Scheduling Results

Task	Selected resource	Execution time range	Task	Selected resources	Execution time range
t_1	<4, 56>	[0, 4]	t_{13}	<5, 57>	[31, 36]
t_2	<6, 68>	[4, 10]	t_{14}	<10, 47>	[27, 37]
t_3	<7, 60>	[4, 11]	t_{15}	<13, 56>	[27, 40]
t_4	<5, 87>	[10, 20]	t_{16}	<6, 65>	[35, 41]
t_5	<6, 55>	[10, 15]	t_{17}	<5, 70>	[36, 41]
t_6	<6, 78>	[11, 17]	t_{18}	<4, 65>	[37, 41]
t_7	<3, 109>	[11, 14]	t_{19}	<5, 60>	[40, 45]
t_8	<4, 98>	[20, 24]	t_{20}	<2, 100>	[41, 43]
t_9	<5, 76>	[17, 22]	t_{21}	<5, 64>	[45, 50]
t_{10}	<7, 85>	[24, 31]	t_{22}	<4, 32>	[50, 54]
t_{11}	<5, 98>	[22, 27]	t_{23}	<3, 87>	[54, 57]
t_{12}	<4, 65>	[31, 35]			

According to the given group of associated tasks in Figure 1, and the corresponding execution time and cost in the different resources listed in Table 1, the execution time for those associated tasks is 57 and corresponding cost is 1638 via CPM scheduling algorithm. Without adopting the tasks hierarchy method in the CPM scheduling, the execution time by CPM is the minimal execution time.

On the other hand, there have several concurrent tasks in the group of associated tasks, as shown in Figure 1. In order to improve the efficiency of the scheduling algorithm, we can adopt the tasks hierarchy methods to schedule the associated tasks, which can reduce the cost of tasks execution within the user-specific delay bound.

5. ATS-DB Scheduling Algorithm

In this section, a hierarchical task model was presented to improve the non-associated tasks concurrency. An associated task scheduling algorithm based on delay-bound constraint (ATS-DB) was proposed to dispatch the tasks to the appropriate nodes in order to minimize the execution cost within the delay bound.

5.1. Hierarchical Decomposition Method

In order to improve the concurrency of the computing tasks, and reduce the execution time delay, the hierarchical decomposition method is adopted. We decompose the associate tasks into different tasks layer based on the correlations among the associated tasks. In each hierarchical layer, the tasks are independent without any the correlation in the execution order. Thus, the tasks in the same hierarchical layer will be grouped into one tasks set. These tasks in the same layer can be concurrently scheduled. The details of the hierarchical decomposition steps are as follows:

1) The dispatcher receivers a task set T of n associated tasks $\{t_1,t_2,...,t_n\}$;

2) Based on the dependency order of tasks' execution, these associated tasks are used to establish the corresponding DAG;

3) Adopting the DAG traversal method from the starting task to the last task, the structure of task hierarchy is also constructed. In each one task hierarchy, there is no dependency among these tasks. That is to say, all of the tasks are concurrent tasks in the same layer. If two tasks are in the adjacent layers, the two tasks are considered as the associated tasks;

4) Based on the tasks hierarchy, the tasks sets are established. The tasks in the same layer are grouped into one set.

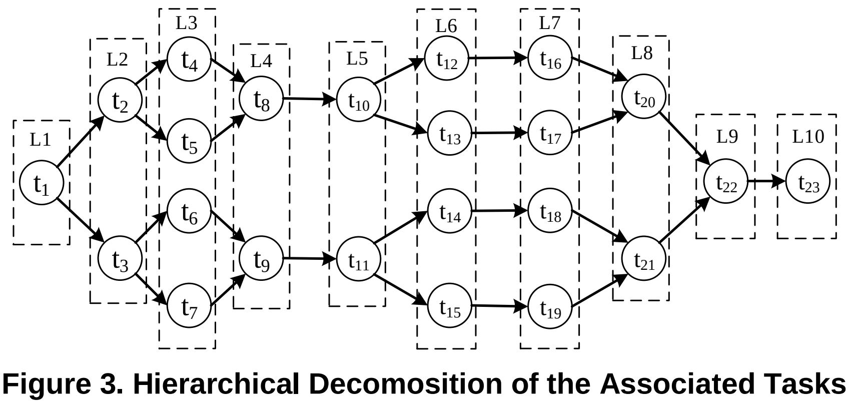

Figure 3. Hierarchical Decomosition of the Associated Tasks

Adopting the hierarchical decomposition method for the associated tasks, it can greatly improve the concurrency performance and fully utilize the system's resources when scheduling the associated tasks. Thus, it should ensure all of the tasks in one set finished within the delay-bound. In order to demonstrate the hierarchical decomposition method, Figure 3 and Table II give an illustrative example. From Figure 3, the DAG of the associated tasks can be divided into ten layers. The first layer just has one node t_1 . $\{t_2,t_3\}$ and $\{t_4,t_5,t_6,t_7\}$ belong to the second and third layer, respectively. t_2, t_3 are the successor of t_1, and t_4, t_5 the predecessors of t_2, respectively. The detail is shown in Table III.

Table 3. An Example of Tasks Hierarchy with ATS-DB

layer	Task sets	layer	Task sets
L_1	$\{t_1\}$	L_6	$\{t_{12}, t_{13}, t_{14}, t_{15}\}$
L_2	$\{t_2, t_3\}$	L_7	$\{t_{16}, t_{17}, t_{18}, t_{19}\}$
L_3	$\{t_4, t_5, t_6, t_7\}$	L_8	$\{t_{20}, t_{21}\}$
L_4	$\{t_8, t_9\}$	L_9	$\{t_{22}\}$
L_5	$\{t_{10}, t_{11}\}$	L_{10}	$\{t_{23}\}$

5.2. Calculation Delay Bound

As to the task scheduling in Cloud computing system, all of resources referring to the computing, storage, and bandwidth capacities, are quite different. When one task is dispatched to the different resources, the execution time may be much different due to the different computing capacities. Therefore, it should compute the execution time based on the computing capacities allocated before executing the scheduling algorithm. It can ensure the total execution time within the delay-bound. In order to accurately compute the execution time for the specific task, some definitions are given as follows.

Definition 4: Processing capacity P_{ij}. P_{ij} is denoted as the processing capacity of resource r_j for the task t_i. The greater P_{ij}, the shorter execution time. Considering the heterogeneous resources, P_{ij} is defined as the ratio of the length of task $t_{i.length}$ to the computing capacity of resource $r_{j.computing}$, and can be calculated as Formula (2).

$$P_{ij} = \frac{t_{i.length}}{r_{j.computing}} \tag{2}$$

Definition 5: Average processing capacity of task t_i, $\overline{P_i}$. If there are m resources used to process the task t_i, $\overline{P_i}$ can be denoted as the estimation of every processing capacity for the task t_i. Average processing capacity of task t_i, $\overline{P_i}$ is calculated as:

$$\overline{P_i} = \frac{\sum_{j=1}^{m} P_{ij}}{m} \tag{3}$$

Definition 6: Average processing capacity of all associated tasks $\overline{P}$. If there are n tasks for scheduling, $\overline{P}$ can defined as the average processing capacity for all associated tasks. For every task, it can calculated from Formula (3). $\overline{P}$ is calculated as:

$$\overline{P} = \sum_{j=1}^{n} \overline{P_i} \tag{4}$$

According to the above definitions, for the specific task t_i, if the average processing capacity $\overline{P_i}$ is greater, task t_i can obtain the shorter execution time. Thus, the processing capacity of the task can indirectly represent the execution time delay.

Definition 7: Delay-bound of the k^{th} task layer Γ_k. If the set of associate tasks has n tasks, the corresponding DAG is divided into K task layers. The average number of the tasks in every layer z_k, the average processing capacity $\overline{P}$, the delay bound in the k^{th} task layer Γ_k can be calculated as:

$$\Gamma_k = \frac{z_k}{n}\overline{P} \tag{5}$$

5.3. Detail of ATS-DB Algorithm

To schedule the associate tasks to meet the delay bound, the associate scheduling algorithm based on the delay-bound constraint (ATS-DB) was proposed in this section. The main idea of ATS-DB algorithm is as follows:

1) When the system received all of the task scheduling requests, based on their dependency in the execution order, all of the associated tasks construct the corresponding task association graph, namely DAG.

2) Two resource queues, resource ready queue and resource waiting queue, are established. All of the resources (computing, storage, and network) are sorted in descending order based on the resources processing capacity. Meanwhile, all of the resources are grouped into the ready queue.

3) Adopting the hierarchical decomposition method, the DAG is divided into multiple tasks layers. The tasks in the same layer are group into the identical task set.

4) For every task layer, the corresponding delay-bound Γ_k can be calculated. Due to no dependency among the tasks in the same layer, they can be concurrently scheduled.

5) The dispatcher will search the appropriate resource from the ready queue to execute those tasks.

6) After those tasks have been completed, the occupied resources are released and put into the waiting queue.

The pseudo-code of ATS-DB scheduling algorithm is shown in Figure 4.

5.4. Example of ATS-DB Scheduling

Based on the CPM scheduling algorithm, the minimal execution time, $T_{\min}$, is 57. If the user-specific execution time (user-expected deadline, $T_{deadline}$) is greater than $T_{\min}$, there is no reasonable scheduling to execution all of associated tasks before the $T_{deadline}$. Assume the user-specific execution time, $T_{deadline}$, is 75, the redundancy time of scheduling is $T_{redundancy} = T_{deadline} - T_{\min} = 18$. According to Figure 3, the redundancy execution time doesn't be allocated to the tasks t_1. If the redundancy time is uniformly allocated to each layer, it can obtain 2 extra time for execution tasks in each layer from L_2 to L_{10}. Thus, the execution time ranges from L_2 to L_{10} are listed in Table IV.

```
Input: $G=\{T,V\}$;
Divide the DAG to $L$ layers
For each task $t_i$ in $L$ layers
Task set $T_k \leftarrow t_i$
Compute $\Gamma_k$
        Calculate the priority of Task set $T_k$
While ($T_k$ is not empty) do
$t_i$ $\leftarrow$tasks from Task set $T_j$
$S_j \leftarrow$ Resources from resource ready queue
If ($d_k < \Gamma_k$ )
dispatch $t_i$ to $S_j$
else
Search resource from resource waiting queue
dispatch $t_i$ to resource
End
Allocate the resources to the resource waiting queue
```

Figure 4. The Pseudo-Code of ATS-DB Scheduling Algorithm

Table 4. Execution Time Range of ATS-DB Scheduling

Layer	Time range	Layer	Time range
L_1	[0, 4]	L_6	[35, 50]
L_2	[4, 13]	L_7	[50, 57]
L_3	[13, 21]	L_8	[57, 64]
L_4	[21, 28]	L_9	[64, 70]
L_5	[28, 35]	L_{10}	[70, 75]

ATS-DB scheduling algorithm will dispatch the tasks to the appropriate resource based on the execution redundancy time range of each layer in Table IV. ATS-DB scheduling can ensure to reduce the total cost while all of tasks finished before the deadline. The details of resource allocation are shown in Table V.

Table 5. ATS-DB Scheduling Results

Task	Selected resource	Execution time range	Task	Selected resources	Execution time range
t_1	<4, 56>	[0, 4]	t_{13}	<7, 46>	[35, 42]
t_2	<7, 52>	[4, 11]	t_{14}	<10, 47>	[35, 45]
t_3	<8, 58>	[4, 12]	t_{15}	<14, 32>	[35, 49]
t_4	<6, 50>	[13, 19]	t_{16}	<6, 65>	[50, 56]
t_5	<6, 55>	[13, 19]	t_{17}	<5, 70>	[50, 55]
t_6	<6, 78>	[13, 19]	t_{18}	<6, 45>	[50, 56]
t_7	<4, 65>	[13, 17]	t_{19}	<5, 60>	[50, 55]
t_8	<7, 65>	[21, 28]	t_{20}	<3, 90>	[57, 60]
t_9	<5, 76>	[21, 26]	t_{21}	<5, 64>	[57, 62]
t_{10}	<7, 85>	[28, 35]	t_{22}	<4, 32>	[64, 70]
t_{11}	<7, 65>	[28, 35]	t_{23}	<4, 49>	[70, 74]
t_{12}	<10, 40>	[35, 45]			

Based on the results of ATS-DB scheduling algorithm, the real execution time is 74 within the expected delay bound. The corresponding cost for execution tasks is

$$Cost_{ATS-DB} = \min\left(\sum_{i=1}^{|V|}\sum_{k=1}^{m} id_i^k c_i^k\right) = 1345.$$

It is obvious that ATS-DB scheduling algorithm can reduce the execution cost by utilizing the redundant time. All of the associated tasks can be finished within the user-expected delay-bound.

5.5. Analysis of ATS-DB Algorithm

Assume that there are *n* associated tasks formed in a DAG. The DAG is divided into *k* layers. In each layer, there are m tasks. To decompose all of the associated tasks into layers, it should traverse all DAG. Therefore, the time complexity of tasks decomposition is $O(k \times m)$. When there are p available resource in the system, the tasks scheduling complexity in each layer is $O(m/p)$.

However, ATS-DB scheduling algorithm have shortcomings. There is no serial tasks in each layer, resulting in too small time slot of redundant execution. The redundant time can be fully utilized. Meanwhile, all of the tasks in the same layer should be executed at the same starting time slot, which reduces the optimization performance in redundant time allocation. Therefore, it should further analyze the structure of the associated tasks to improve the concurrency of tasks, and present a better redundant time allocation scheme.

6. SAH-DB Scheduling Algorithm

In this section, a structure-based hierarchical task model was presented to improve the concurrency of tasks. Moreover, a structure analysis-based tasks scheduling algorithm based on delay-bound constraint (SAH-DB) was proposed to further improve the scheduling performance while meeting the delay bound requirements.

6.1. SAH-based Hierarchical Decomposition Method

To improve the execution concurrency of non-associated tasks, it should further analyze the structure of the DAG based on the dependency order of the associated tasks. From the Figure 1, we can find that there are two basic structures in the graph: serial and parallel structure, as shown in Figure 5 and 6. It is easy to find that the tasks hierarchies with different structure have the different number of the paths for executing the tasks. Considering the tasks' structure, the non-associated tasks in the different parallel execution paths can be concurrently executed. Thus, all of the non-associated tasks in the different parallel paths are grouped into the same layer. These tasks in the same layer can be concurrently scheduled. The details of the structure-based hierarchical decomposition steps are as follows:

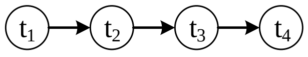

Figure 5. Serail Structure in the DAG

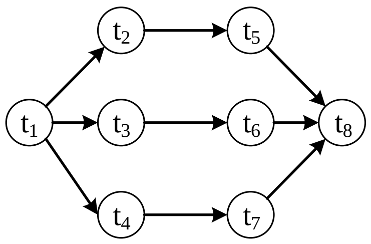

Figure 6. Parallel Structure in the DAG

1) The dispatcher receivers a task set T of n associated tasks $\{t_1, t_2, ..., t_n\}$;

2) Based on the dependency order of tasks' execution, these associated tasks are used to establish the corresponding DAG;

3) Adopting the DAG traversal method from the starting task to the last task, the structure of task hierarchy is also constructed.

4) Traversing the DAG, it can locate the parallel structure in the DAG. For each parallel structure, all of the tasks in the parallel sub-paths are merged into one layer. That is to say, the tasks in the different sub-paths belonging to the same layer can be concurrently executed.

5) Based on the analysis results of graph structure, the tasks hierarchy sets are established. The tasks in the same layer are grouped into one set.

Table 6. An example of Tasks Hierarchy with SAH-DB

layer	Task sets	layer	Task sets
L_1	$\{t_1\}$	L_5	$\{t_{12}, t_{13}, t_{14}, t_{15}, t_{16}, t_{17}, t_{18}, t_{19}\}$
L_2	$\{t_2, t_3\}$	L_6	$\{t_{20}, t_{21}\}$
L_3	$\{t_4, t_5, t_6, t_7\}$	L_7	$\{t_{22}, t_{23}\}$
L_4	$\{t_8, t_9, t_{10}, t_{11}\}$		

After analysis of the structure of associated tasks shown in Figure 1, they can be decomposed into 7 layers. Adopting the structure-based hierarchical decomposition method for the associated tasks, it can further improve the concurrency performance when scheduling the set of associated tasks. To clearly illustrate the structure-based hierarchical decomposition method, Figure 7 and Table VI give an instance based on the DAG in the Figure 1. From the Figure 7, the tasks $\{t_8, t_{10}\}$ and $\{t_9, t_{11}\}$ are the parallel structures in the layer L_4. The tasks $\{t_{12}, t_{16}\}$, $\{t_{13}, t_{17}\}$, $\{t_{14}, t_{18}\}$, and $\{t_{15}, t_{19}\}$ are the parallel structures in the layer L_5. Compared with the tasks decomposition method in ATS-DB, the concurrency of non-associated tasks can be further improved.

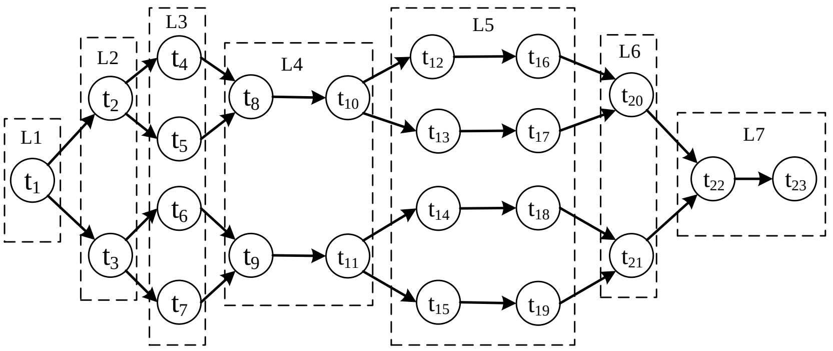

Figure 7. Structure-Based Hierarchical Decomposition of the Associated Tasks

6.2. Detail of SAH-DB Algorithm

In the ATS-DB, the different structures adopt the same allocation scheme of the redundant time, which will result in the poor execution performance of parallel tasks. To fully utilize the redundant time, SAH-DB scheduling algorithm will allocate the redundant time based on the number of tasks and parallel paths in each layer. The redundant time allocation in each layer can be computed as follows:

$$T_{j\_redundant}^{SAH} = T_{redundant} \times |N_j| / N \tag{6}$$

where $T_{j\_redundant}^{SAH}$ is the redundant time of the j^{th} layer, $T_{redundant}$ is the total redundant time, $|N_j|$ is the number of tasks in the j^{th} layer, and N is the total number of tasks.

```
Input: $G=\{T,V\}$, expected time $T_{deadline}$;
Output: the tasks scheduling list
Divide the DAG to $L$ layers
If $T_{deadline} > T_{min}$ then
  For each task $t_i$ in the $j^{th}$ layer
    task set $T_j \leftarrow t_i$;
    compute $T_{j\_redundant}^{SAH}$;
    choose the appropriate resource for task $t_i$ based on $T_{j\_redundant}^{SAH}$
    While ($T_j$ is not empty) do
      $t_i$ $\leftarrow$tasks from Task set $T_j$
      $s_j$ $\leftarrow$ Resources from resource ready queue
     dispatch $t_i$ to resource
    End
    Allocate the resources to the resource waiting queue
Else
    return false; // the expected time is greater than the deadline
```

Figure 8. The Pseudo-Code of SAH-DB Scheduling Algorithm

To schedule the associate tasks to meet the delay bound, the structure-based scheduling algorithm based on the delay-bound constraint (SAH-DB) was proposed in this section. The pseudo-code of ASH-DB scheduling algorithm is shown in Figure 8. The main idea of SAH-DB algorithm is as follows.

1) When the system received all of the task scheduling requests, based on their dependency in the execution order, all of the associated tasks construct the corresponding task association graph, namely DAG.

2) Two resource queues, resource ready queue and resource waiting queue, are established. All of the resources (computing, storage, and network) are sorted in descending order based on the resources processing capacity. Meanwhile, all of the resources are grouped into the ready queue.

3) Adopting the structure-based hierarchical decomposition method, the DAG is divided into multiple tasks layers. The tasks in the same layer are included as the corresponding task set. In the same layer, there may exist parallel paths of tasks execution.

4) For each task layer, the corresponding redundant time $T_{j\_redundant}^{SAH}$ can be calculated according to Formula (6).

5) The dispatcher will choose the appropriate resource from the ready queue for each task based on its redundant time.

6) After those tasks have been completed, the occupied resources are released and put into the waiting queue.

Based on the structure-based hierarchy method, the number of layers with SAH-DB algorithm is smaller than that with ATS-DB algorithm. The smaller number of layers can alleviate the time slots fragments and improve the utilization of the redundant execution time.

6.3. Example of SAH-DB Scheduling

Based on the CPM scheduling algorithm, the minimal execution time, $T_{\min}$, is 57. As the same deadline of ATS-DB scheduling algorithm, the deadline of execution time is 75. So, the redundant time $T_{redundancy}$ is 18. According to Formula (6), it can compute the corresponding the execution time range of each layer, as shown in Table VII. From the Table VII, it is obvious that the redundant time can be much reasonably allocated for all of the associated tasks. Thus it can improve the tasks' concurrency.

Table 7. Execution Time Range of SAH-DB Scheduling

Layer	Time range	Layer	Time range
L_1	[0, 4.8]	L_5	[35.6, 59.9]
L_2	[4.8, 13.4]	L_6	[59.9, 66.5]
L_3	[13.4, 21.5]	L_7	[66.5, 75]
L_4	[21.5, 35.6]		

SAH-DB scheduling algorithm will dispatch the tasks to the appropriate resource based on the execution redundant time range of each layer in Table VII. SAH-DB scheduling can ensure to reduce the total cost while all of tasks finished before the delay bound. The details of SAH-DB scheduling results are shown in Table VIII.

Table 8. SAH-DB Scheduling Results

Task	Selected resource	Execution time range	Task	Selected resources	Execution time range
t_1	<4, 56>	[0, 4]	t_{13}	<7, 46>	[35.6, 42.6]
t_2	<7, 52>	[4.8, 11.8]	t_{14}	<10, 47>	[35.6, 45.6]
t_3	<8, 58>	[4.8, 12.8]	t_{15}	<14, 32>	[35.6, 49.6]
t_4	<6, 50>	[13.4, 19.4]	t_{16}	<8, 60>	[45.6, 53.6]
t_5	<6, 55>	[13.4, 19.4]	t_{17}	<5, 70>	[42.6, 47.6]
t_6	<6, 78>	[13.4, 19.4]	t_{18}	<10, 40>	[45.6, 55.6]
t_7	<4, 65>	[13.4, 17.4]	t_{19}	<5, 60>	[49.6, 54.6]

t_8	<7, 65>	[21.5, 28.5]	t_{20}	<3, 90>	[59.9, 62.9]
t_9	<5, 76>	[21.5, 26.5]	t_{21}	<5, 64>	[59.9, 64.9]
t_{10}	<7, 85>	[28.5, 35.5]	t_{22}	<4, 32>	[66.5, 70.5]
t_{11}	<7, 65>	[26.5, 33.5]	t_{23}	<4, 49>	[70.5, 74.5]
t_{12}	<10, 40>	[35.6, 45.6]			

Based on the results of SAH-DB scheduling algorithm, the real execution time is 74.5, within the expected delay bound. The corresponding cost for execution tasks is $Cost_{SAH-DB} = \min\left(\sum_{i=1}^{|V|}\sum_{k=1}^{m} id_i^k c_i^k\right) = 1323$.

It clearly demonstrates that ATS-DB scheduling algorithm can improve the concurrency of non-associated tasks and further reduce the execution cost by fully utilizing the redundant time. Meanwhile, all of the associated tasks can be executed within the user-expected delay-bound.

7. Performance Evaluation

To evaluate the better performance on tasks' executing time for the proposed ATS-DB, SA-DB, and CPM, in this section, we compare ATS-DB, SA-DB with CPM in terms of the time span and time delay for all of the associated tasks.

7.1. Experiments Settings & Methodology

The simulation tool, CloudSim, is used to simulate the procedure of task scheduling. In the CloudSim, each virtual node represents one resource and has one processor. All of the requested tasks are executed on the virtual nodes. The computing capacity of virtual nodes varies from 25 to 100. The storage capacity is set from 100,000 to 200,000, and the bandwidth of the network is set between 1,000 and 2,000.

For the associated tasks were established by DAGs. The number of tasks set is $\{20, 40, 60, 80, 100, 120, 140\}$. All the DAGs are generated by the DAG graph random generator. The computer resources pool can be randomly generated from the interval [5, 10], while the task executing time is randomly generated from the interval [5, 30] and the corresponding cost is inversely proportional to the time. In the experiments, we set 50 virtual nodes to establish a heterogeneous computing environment. Adopting ATS-DB, ASH-DB, and CPM algorithm to evaluate the execution performance with different number of the associated tasks.

To evaluate the scheduling performance of the proposed ATS-DB and SAH-DB, the experiments uses two merits to indicate the scheduling performance. One is the optimization ratio of the cost, denoted α_{cost}. α_{cost} is defined as the ratio of the execution cost with scheduling to that with CPM.

$$\alpha_{cost} = (cost_{CMP} - cost_{ATS-DB}) / cost_{CMP} \tag{7}$$

The larger is the optimization ratio of cost α_{cost}, the better concurrency performance has the scheduling algorithm.

The second merit is to evaluate the execution cost with the different deadline of tasks scheduling. In the simulation, the deadline is set to $T_{deadline} = T_{\min} \times (1+\theta)$, $\theta = \{0.05, 0.10, 0.15, ..., 0.50\}$.

7.2. Execution Cost with the Same Deadline

We evaluate the scheduling efficiency in terms of the execution cost under a varying number of associated tasks, ranging from 20 to 140. Figure 9, 10, 11, and 12 illustrate the execution cost by applying the proposed ATS-DB, SAH-DB, and CPM scheduling algorithm with 25, 50, 75, and 100 virtual nodes, respectively. As Figure 9-12 shown, the proposed ATS-DB and SAH-DB scheduling algorithms can comsume smaller cost, compared with CPM algorithm in the different number of associated tasks. Meanwhile, with the increase of the number of resources, the total cost of scheduling decreseas. Moreover, the cost of scheduling increases with the number of associated tasks.

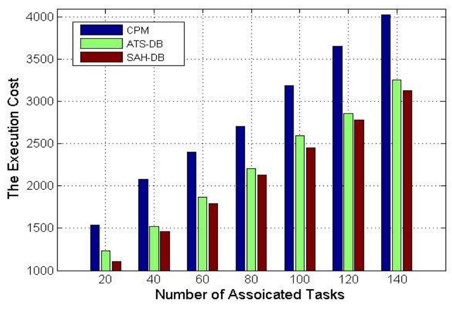

Figure 9. The Execution Cost With Resources = 25

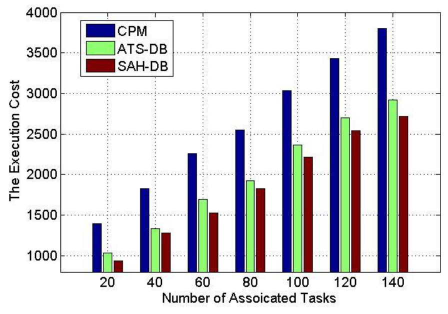

Figure 10. The Execution Cost With Resources = 50

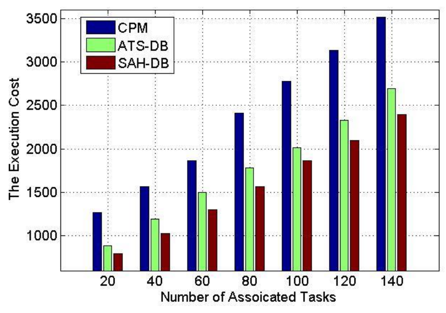

Figure 11. The Execution Cost With Resources = 75

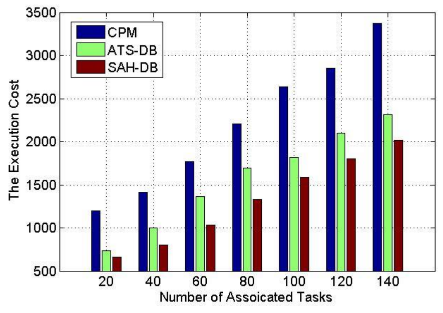

Figure 12. The Execution Cost With Resources = 100

On the other hand, Figure 13 shows the optimization ratio of execution cost with ATS-DB and SAH-DB scheduling algorithms. From the results of Figure 13, SAH-DB can present better scheduling performance under the same resources and execution deadline when the number of tasks varies from 20 to 140. The reason is that SAH-DB scheduling algorithm can obtain better concurrency by adopting the structure-based tasks hierarchical method. Furthermore, the redundant time of scheduling is allocated considering the structure in each layer.

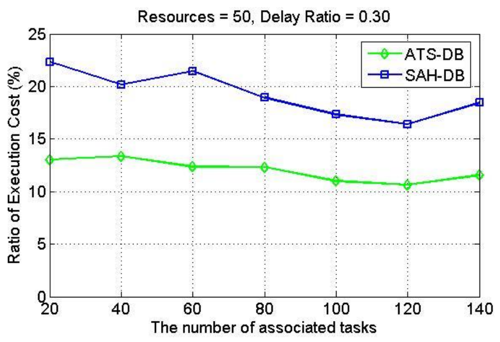

Figure 13. The Ratio of Execution Cost

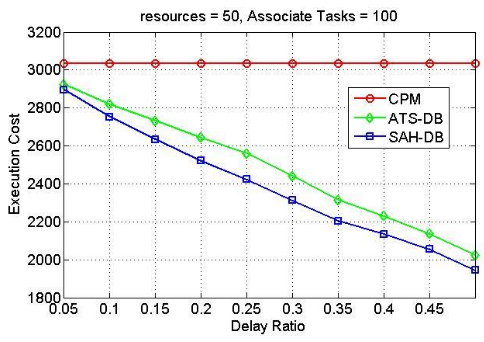

Figure 14. The Delay Ratio with Associate Tasks = 100

7.3. Optimization Ratio with Different Deadline

To evaluate the execution efficiency, we compare ATS-DB, SAH-DB with CPM in terms of the optimization ratio of cost in different deadline parameters, from 0.05 to 0.50, when the virtual resources is 50, and the number of associated tasks is 100. Figure 14 illustrates the optimization ratio of ATS-DB and SAH-DB. From the results in Figure 14 shown, SAH-DB can obtain better optimization ratio of cost than that of ATS-DB when the deadline of execution time is the same. SAH-DB can reduce the execution cost by 10% to 40%. On the other hand, with the increase in the deadline of the execution time from 0.05 to 0.5, the total execution cost of two scheduling algorithms decrease. The reason is that when the deadline increases, the appropriate resources can be allocated to the corresponding tasks, which results in the decrease of the total cost of scheduling associated tasks.

From the simulation results, we can obviously find that the proposed ATS-DB and SAH-DB algorithm can get a better performance that CPM algorithm. Furthermore, SAH-DB can have better concurrency performance than ATS-DB due to adopting the structure-based hierarchy and redundant time allocation method.

8. Conclusion and Future Work

Task scheduling is one of the most important issues in the cloud computing environments. In the cloud systems, the main goal of the task scheduling algorithms is to balance the workload among the computing nodes and maximize the utilization while meeting the bound of the total execution time. Concerning the delay of the associated tasks scheduling in cloud computing, two hierarchical task models were discussed and the associated task scheduling algorithms based on delay-bound constraint (ATS-DB and SAH-DB) were proposed. The associated tasks and the task execution order were represented by one directed acyclic graph (DAG). The proposed hierarchical task models can improve the task execution concurrency. The independent tasks in each layer was grouped into the corresponding task set belonging to the task layer. Through the calculation of the total tasks execution time-bound in each task layer, the associated task was dispatched to the resources with the minimum execution time. Extensive experimental results demonstrated that the proposed ATS-DB and SAH-DB algorithms can achieve better performance than CPM algorithm in the terms of the total execution cost and resource utilization. In this paper, the communication costs among the associate tasks are ignored. In the future work, the communication costs will be considered to meet the requirements of real applications.

Acknowledgements

This research is partially supported by the National Key Technology Research and Development Program of the Ministry of Science and Technology of China under Grant No. 2013BAB06B04; Key Technology Project of China Huaneng Group under Grant No.HNKJ13-H17-04; Science and Technology Program of Yunnan Province under Grant No. 2014GA007; the Fundamental Research Funds for the Central Universities under Grant No. 2015B22214; Nature Science Foundation of Jiangsu Province under Grant No. BK20130852; NSF-China and Guangdong Province Joint Project: U1301252.

References

[1] A. Delavar, M. Javanmard, M. Shabestari and M Talebi, "Reliable Scheduling Distributed in Cloud Computing", in International Journal of Computer Science, Engineering and Applications (IJCSEA), vol. 2, no. 3, **(2012)**.
[2] R. Patel and S. Patel, "Survey on Resource Allocation Strategies in Cloud Computing", International Journal of Engineering Research & Technology (IJERT), vol.2, no. 2, **(2013)**, pp. 1-5.
[3] L. Guo, S. Zhao, S. Shen and C. Jiang, "Task Scheduling Optimization in Cloud Computing Based on Heuristic Algorithm", Journal of Networks, vol. 7, no. 3, **(2012)**, pp. 547-553.
[4] G. Yan, J. Yu and X. Yang, "Workflow scheduling strategy based on the reliability in Cloud computing", Computer Applications, vol.34, no.3, **(2014)**, pp. 673-677.
[5] X. Du, C. Jiang, G. Xu and Z. Ding, "A Grid DAG Scheduling Algorithm Based on Fuzzy Clustering", Journal of Software, vol. 17, no. 11, **(2006)**, pp. 2277-2288.
[6] D. Wilmer, T. Klos and M. Wilson, "Distributing Flexibility to Enhance Robustness in Task Scheduling Problems", Proceedings BNAIC, **(2013)**, pp. 344-351.
[7] J. Huang, "The Workflow Task Scheduling Algorithm Based on GA Model in the Cloud Computing Environment", Journal of Software, vol. 9, no. 4, **(2014)**, pp. 873-880.
[8] H. Tabatabaee, T. M. R. Akbarzadeh and N. Pariz, "Dynamic task scheduling modeling in unstructured heterogeneous multiprocessor systems", **(2014)**.
[9] J. Li, A. Saifullah and K. Agrawal, "Capacity Augmentation Bound of Federated Scheduling for Parallel DAG Tasks", Tech. Rep. WUCSE-2014-44, Washington University in St Louis, USA, **(2014)**.
[10] Z. Liu, W. Qu and W. Liu, "Resource preprocessing and optimal task scheduling in cloud computing environments", Concurrency and Computation: Practice and Experience, **(2014)**.
[11] http://en.wikipedia.org/wiki/Critical_Path.
[12] Hadoop Capacity Scheduler, http://hadoop.apache.org/common/docs/current/capacity_scheduler.html.
[13] Hadoop Fair Scheduler http://hadoop.apache.org/common/docs/current/fair_scheduler.html.
[14] S. Thomas and L. Kevin, "Dynamic Proportional Share Scheduling in Hadoop", Lecture Notes in Computer Science, vol. 6253, **(2010)**, pp. 110-131.
[15] D. Xi, Y. Wang and H. Li, "Scheduling Mixed Real-time and Non-real-time Application in MapReduce Environment", In Procs. of 17th IEEE Int'l on Parallel and Distributed Systems, **(2011)**, pp.9-16,Tainan.
[16] J. Polo, "Performance-Driven Task Co-Scheduling for MapReduce Environments", in Proc. of 2010 IEEE/IFIP Network Operations and Management Symposium (NOMS), **(2010)**, pp. 373-380.
[17] K. Kc and K. Anyanwu, "Scheduling Hadoop Jobs to Meet Deadlines", in Procs. of the 2010 IEEE 2nd Int'l Conf. on Cloud Computing Technology and Science (CloudComm 2010), **(2010)**, pp. 388-392.
[18] H. You, C. Yang and J. Huang, "A Load-Aware Scheduler for MapReduce Framework in Heterogeneous Cloud Environments", in Procs. of the ACM Symp. on Applied Computing, **(2011)**, pp. 127-132.
[19] M. Prinedo, "Scheduling: Theory, Algorithms, and Systems", 3rd edition, Springer Science, **(2008)**.

Authors

Yingchi Mao, she is an associate professor at College of Computer and Information Engineering of Hohai University, China. She received her B.Sc. and M.Sc. degrees in computer science and technology from Hohai University in 1999 and 2003, respectively. She received her Ph.D. degree in computer science and technology from Nanjing University, China in 2007. Her research areas include distributed data management, distributed computing system.

Haishi Zhong, he is a master candidate at College of Computer and Information, Hohai University, China. He was received his B.Sc. degree in computer science and technology from Hohai University in June, 2015. His research interests include data storage, cloud computing and big data.

Longbao Wang, he is an assistant profeesor in the College of Computer and Information Engineering at Hohai University. He received her B.Sc. degree and M. Sc degree in computer science and technology from Hohai University in 2001 and in 2005, respectively. His research areas include Cloud computing, big data analysis and processing.

Xiaofang Li, she is an associate professor in the College of Computer and Information Engineering at Changzhou Institute of Technology, Changzhou, China. She received her B.Sc. degree in computer science and technology from Hohai University in 1995 and her M.Sc. degree in power system automation from Hohai University in 2004. She is the Secretary General of the IEEE SMC Nanjing Chapter. Her research areas include information acquisition and processing in wireless sensor networks.