Available online at www.ijpe-online.com

International Journal of Performability Engineering

vol. 15, no. 10, October 2019, pp. 2772-2782
DOI: 10.23940/ijpe.19.10.p24.27722782

Improved Grid Task Scheduling Model Algorithm

Feng Liu*

Department of Electrical and Information Engineering, Heilongjiang University of Technology, Jixi, 158100, China

Abstract

On the basis of analyzing the current status and the key technology of grid workflow scheduling, in-depth research on the grid workflow scheduling algorithm under the restraint of time QOS and trust QOS is conducted in this paper. A grid workflow task scheduling algorithm (GWTS) based on critical tasks under the constraints of trust is designed. Firstly, backward depth of tasks is calculated in GWTS, and critical tasks are ascertained according to the execution time on candidate resources. Secondly, the trust of grid resources is computed based on direct experience and recommendation experience synthetically. Finally, tasks are scheduled by decreasing backward depth, and resources are closed to meet the integrated function of execution time and trust and are allocated for critical tasks as a priority. Experiments show that the workflow completion time is reduced, the success rate of task execution is increased by 6-15%, and the GWTS algorithm can effectively guarantee grid scheduling resource optimization and improve the scheduling efficiency.

Keywords: grid; workflow scheduling; trust QOS

(Submitted on September 15, 2019; Revised on September 29, 2019; Accepted on October 24, 2019)

1. Introduction

In grid computing, task management, task scheduling, and resource management are three necessary basic functions of grids. Effective job scheduling of workflows is key and also an issue for grid systems [1]. Due to the dynamism and autonomy of grids, many unreliable resources exist in grid environments. Choosing proper resources to meet user demand is a significant and complicated job, and thus the scheduling of dependable resources has become very important [2]. Grid workflow is an important part of grid computing environments [3]. So far, it has mainly been applied in the scientific research field. Workflow task scheduling not only distributes relative sub tasks to suitable grid resources, but also considers dependence relations between different tasks and the execution sequence of them among different resources. As the limited degree of grid workflow ending time is varied, workflow tasks are divided into two types [4]. A workflow with a highly restricted time range should ensure that the whole workflow is completed by a certain deadline. During the scheduling of workflow tasks, the completion time of key tasks determines the operational time of the entire workflow. If the time of the workflow needs to be reduced, key jobs should be preferentially scheduled [5-6].

Grid dynamism and self-governing allows for many unreliable resources in the grid system. Selecting appropriate resources to complete task scheduling becomes a complex and important job. If key jobs in the workflow are allocated to unreliable resources, it would lead to rescheduling; as a consequence [7], it would give rise to extra expenditures, waste of system resources, worse scheduling efficiency, and unnecessary losses. Under this circumstance, it is rather important to introduce trust to the study of grid scheduling mechanisms [8-9].

To solve the above problem, this paper proposes a grid workflow task scheduling algorithm based on trust constraint. The algorithm firstly determines the reverse depth of task to decide the execution order of the workflow by a processor. Then, with reference to the trust degree of resources and time of the sub tasks executed on the resource, it decides resources for key tasks. Finally, it allocates resources to non-key tasks.

* Corresponding author.
E-mail address: 3140025490@qq.com

2. Grid Workflow

Workflow is a kind of operating process that can be carried out completely or partially automatically. Following a series of rules, documents, tasks, or information can be passed and executed among different executors, making the process work automatically and collaboratively to improve working efficiency [10-11]. The concept of grid workflow was first proposed on the foundation of business workflow, an automatic execution process of grid tasks. The workflow management system can wholly define the administrative workflow and implement workflows in proper order according to the workflow logic predefined in the computer. Grid workflows are an automatic execution process of grid tasks.

2.1. Architecture and Execution of Grid Workflow

The main functions of the grid workflow management system are divided into two stages. Firstly, the workflow task is defined, and the workflow is modelled. Secondly, the workflow application is processed, including the implementation of the management workflow and the grid resource interaction. The system is shown in Figure 1.

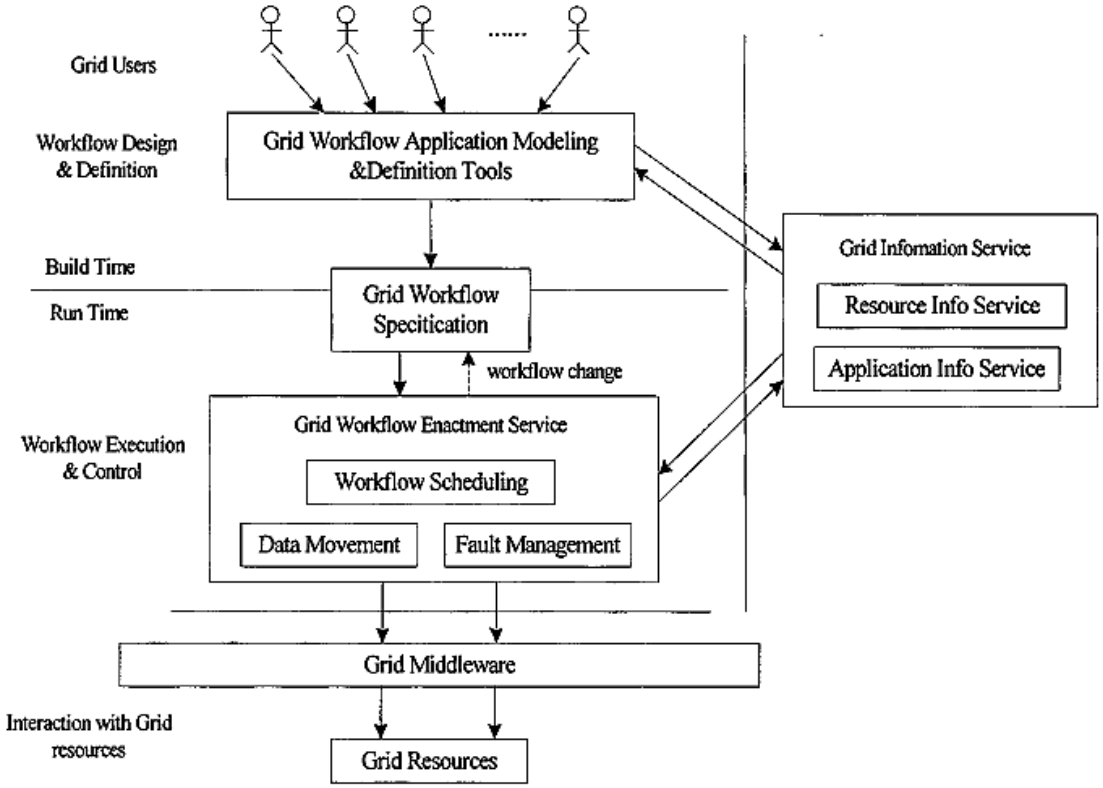


Figure 1. Grid workflow management system

The basic process of grid workflow execution includes:

(1) Grid users submit service requests and related requirements.

(2) The grid workflow engine and workflow system use the appropriate grid workflow description language to convert the service request into metadata.

(3) The task is defined to decompose the service request from top to bottom, including the function of the task and the input and output. Related data files are generated.

(4) According to the data file, service components are selected, and a logical workflow is formed.

(5) According to the current environment and task requirements, the appropriate resources for the task are selected to determine the order of the task.

(6) A task is performed, and the final results or data are passed to the user.

2.2. Task Type of Workflow

According to the definition of the Workflow Management Coalition (WMFC), the execution structure of the workflow task is mainly composed of the following basic types:

(1) Order relation

Tasks v_1, v_2, v_3, V1, and V2 are followed by sequential execution. Task v_1 performs after the execution of task v_2, as shown in Figure 2(a).

(2) Parallel relation

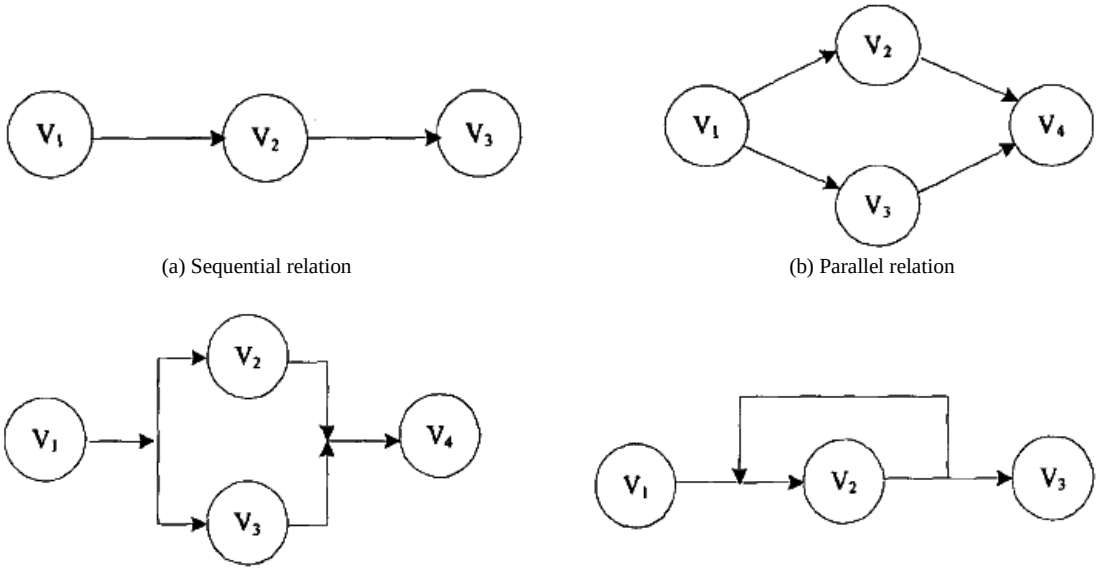
Tasks v_2 and v_3 follow parallel execution. That is, these two tasks can be executed simultaneously or in any order, as shown in Figure 2(b).

(3) Choice relation

When task v_1 is executed, it selects v_2 or v_3 to perform the task, as shown in Figure 2(c).

(4) Cyclic relation

The cyclic relation indicates that a task is executed several times. Task v_2 in Figure 2 can be executed one or more times, as shown in Figure 2(d).



(a) Sequential relation (b) Parallel relation

(c) Choice relation (d) Circular relation

Figure 2. Four kinds of execution structure of workflow

3. Description of Problem

3.1. Task Model

Workflow task TG is expressed with V, E, W, and C, where V is the collection of task nodes v_j, E is the collection of relationship between tasks, e_{ij} is the dependence relation between v_i and v_j, v_i is the precursor of v_j, $v_i \in pre(v_j)$, $succ(v_i)$ means the precursor's task collection of v_i, $v_j \in succ(v_i)$ means task v_i's subsequent task collection, W is the collection task computing expense, weight w_j is the computing expense of task v_j, C is the collection of communication

time, and c_{ij} is the time of data transfer from task vi to task v_j. When v_i and v_j are at the same resource node, c_{ij} =0.

Entry task V_{entry} means a task with no precursor, and exit task V_{exit} means a task with no successor. Here, we discuss only tasks of the computation-intensive type. For them, the communication time has little calculations, which can be neglected.

3.2. Resource Model

The distributed heterogeneous resources in grid environments, such as the processor's computing resource and storage memory resource, belong to different grid nodes, for which management strategies are diversified. However, standard and uniform interfaces are provided. Resources in the grid environment include the collection of different static attributes and dynamic attributes. Static attributes include the CPU, number of processors, geographical position of resources, memory size, memory capacity, and network bandwidth. Dynamic attributes include the available number of processing units, current CPU loading, available memory, and current storage space. Suppose the grid contains resource nodes $m_1, m_2, \cdots, m_z$ of *z* grids. Each node mi includes a computing resource, storage resource, etc. One resource performs higher, indicating that its execution time on the resource becomes shorter; otherwise, the execution time is longer. GR is the collection of grid resource node m_i.

3.3. Scheduling Principle

Grid scheduling distributes tasks to the most suitable resource nodes for execution by certain strategies in the case of meeting QOS, such as time and trust constraints. Grid workflow scheduling considers not only a task's dependence relationship but also resourceful reliability. The proposed grid workflow task scheduling algorithm based on trust restriction has the following principles.

3.3.1. Subtask Priority Principle

Owing to the time sequence dependence relationship between subtasks, the scheduling and execution will affect the completion of the following tasks. Non-branch tasks have little effect on the execution of subsequent tasks. Even if there is a delay, it will not impact the execution of the whole grid workflow. After the completion of sub task v_1, branch v_2 will affect v_4, v_5 tasks and scheduling, so give priority to the allocation of resources for v_2 and then for task v_3. In doing so, the task of v_3, v_4, v_5 execution parallelism is improved, the scheduling time is shortened. In the specific scheduling algorithm, the inverse depth of the task is reflected by the reverse depth; the greater the reverse depth, the more tasks to participate in the branch, and the higher the priority in scheduling. This is shown in Figure 3.

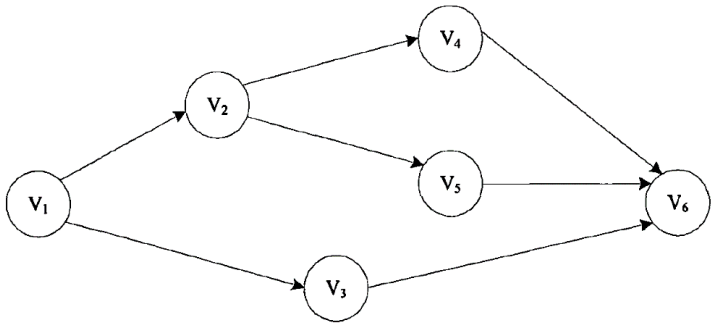


Figure 3. Grid workflow application example

3.3.2. Key Path Minimization Principles

The key path is the workflow path with the biggest time consumption in the grid workflow task chart. From workflow entry task to exit task, the biggest path of the summed task computing expense and communication cost is named CP (critical path). The task on the key path is called the key task. The execution time of the task on the key path plays a decisive role in the scheduling time of the entire grid workflow. Shortening the completion time of the whole application requires reducing the execution time of the key path.

We take Figure 3 as an example. From task v_2 to v_6, there are two paths. Assume v_4's execution time is far less than

that of v_5. There are two candidate resources, m_1 and m_2, which have different performances: m_1's performance is higher than that of m_2. To minimize v_5's execution time, dispatching resource m_1 to execute v_5 and m_2 to execute v_4 takes less time than m_1 matching v_4 and m_2 matching v_5; thus, the scheduling efficiency is increased.

3.3.3. Reliable Resource Prior Scheduling Principle

If the key path is scheduled to resources with better performance but lower trust, the grid task may not be executed in a timely manner as a result of resource failures or dynamic quits of resources, thus causing re-scheduling, extending the scheduling time of the key task, and raising the workflow completion time. It is necessary to choose the resource that has the highest comprehensive beneficial function of execution time and credibility for scheduling.

3.3.4. Minimum Time and Load Balancing Principle of Sequential Task

In the premise of meeting the grid workflow task QOS requirement, choose for the key task the resource that has the best performance and trust for treatment. For the non-key task, choose a dependable idle resource to enhance system resource utilization and achieve load balancing.

4. Grid Workflow Task Scheduling Algorithm

At present, there are many workflow task scheduling algorithms based on grid systems. MCP and DLS are list scheduling methods in an isomorphic computing environment, while HEFT and CPOP are list scheduling methods based on the earliest completion time and key paths in an isomerous computing environment. Those techniques only consider a task's time QOS and do not account for uncertainties or fraudulence of resource nodes. They are unable to cater to a user's service quality requirements in terms of scheduling length and resource confidence. Hence, this paper proposes a grid workflow task scheduling algorithm (GWTS). The main idea of this algorithm is to schedule workflow tasks from large to small based on the tasks' inverse depth to conform to task dependence relationships and distribute the resources that satisfy the comprehensive function of completion time and trust, until all tasks fulfill resource allocation.

The algorithm is investigated on the basis of the following grid environment and prerequisites:

(1) When the GR description of the resource model is met; each resource is mutually connected and can inter-communicate;

(2) Schedule the computation-intensive type workflow task, formally describe the workflow task being met, and assume that the task computing time is much greater than the communication time. Therefore, the communication time is ignored;

(3) Only when the precursor task is completely finished and the relative program data transferring is over can the following task start;

(4) One node can execute one subtask at a time. The grid resource is not reliable and may be invalid. A higher resource trust suggests the failure rate is lower.

Step 1 For the application of one workflow, from exit task V_{exit} to entry task V_{entry}, calculate the inverse depth of each task. The specific calculation method is defined as follows:

Definition 1: The task forward depth $TD(j)$ means the greatest path length from entry task V_{entry} to task v_j. It is given by Equation (1).

$$TD(j) = \begin{cases} 0, & \text{if } (v_j \in V_{entry}) \\ \max\limits_{v_i \in pre_{(v_j)}} \{TD(i)\} + 1, & otherwise \end{cases} \tag{1}$$

Definition 2: The task backward depth $BD(i)$ means the greatest path length from task v_i to exit task V_{exit}. It is expressed as Equation (2).

$$BD(j) = \begin{cases} 0, & \text{if } (v_i \in V_{exit}) \\ \max\limits_{v_j \in succ_{(v_i)}} \{BD(j)\} + 1, & \text{otherwise} \end{cases} \quad (2)$$

The task depth of the grid workflow instance in Figure 3 is shown in Table 1.

Table 1. Calculation results of two kinds of hierarchical lists in an example

TD	Top Level	BD	Bottom level
0	v_1	3	v_1
1	v_2, v_3	4	v_2
2	v_4, v_5	1	v_3, v_4, v_5
3	v_6	0	v_6

Step 2 Use Equation (3) to calculate the average execution time according to the candidate resource for each workflow subtask. According to the definition and computing method in Equation (5), determine the task on the path with the longest completion time from entry node to exit node, and decide the key task, i.e., CT.

Definition 3: The task execution time is consumed by task t_j for relative candidate resource m_k in the process from start to end, and it is expressed as t_{kj}. The execution time is related to the resource type and performance, which can be predicted before scheduling and is known. If the resource does not have a problem or becomes ineffective, the execution time is relatively fixed.

Definition 4: The task average execution time $\overline{t_j}$ is the mean value of execution time t_{kj} by tasks in the candidate resource collection. It is shown in Equation (3).

$$t_j = \frac{\sum t_{jk}}{N(qi)} \quad (3)$$

Definition 5: The resource earliest available time avai $[m_k]$ is the earliest idle usable time by resource m_k for scheduling tasks to resources, which is also the earliest time for submitting tasks before a local task is completed.

Definition 6: The task beginning time est(v_j) is the earliest starting time of task v_j on processing machine m_k. The time is the biggest value of summation of the earliest available time avai $[m_k]$ of the task's node and completion time and the communication time c_{ij} of the precursor task. It is shown in Equation (4).

$$est_{(v_j)} = \max\{avail[m_k], \max_{v_i \in pre_{(v_i)}} (etc(v_i) + c_{ij})\} \quad (4)$$

Definition 7: The task completion time $est(v_j)$ refers to the sum of the task starting time $est(v_j)$ and the task's execution time t_{kj} on the resource. Combine Equations (4) and (5) to get Equation (6).

$$est_{(v_j)} = est_{(v_j)} + t_{kj} \quad (5)$$

$$est_{(v_j)} = max\{avail[m_k], \max_{v_i \in pre_{(v_i)}} \{etc(v_i) + c_{ij}\} + est_{(v_j)} = est_{(v_j)} + t_{kj} \quad (6)$$

Step 3 Compute the resource credibility, which includes direct confidence and recommendation confidence. Direct confidence is associated with resource successful execution times, failure times, delay times, and a time decaying function, which is calculated using Equation (5). Recommendation confidence is the direct confidence of the recommendation resource itself as well as the recommendation factor connected with the time decaying function, which is acquired using Equation (6). Let the direct confidence impact factor α mean the weight taken up by the direct confidence degree. Generally speaking, the influence of the direct trust degree is greater than that of the recommendation confidence. In different grid environments, α varies. It is shown in Equation (7).

$$\Omega(m,t,c) = \alpha \times D\big(m_j,t,c\big) + (1-\alpha) \times R(m_j,t,c) \tag{7}$$

Step 4 According to the task backward depth from large to small, distribute the key task, with the task execution time and resource credibility as comprehensive beneficial functions to allocate resources for all workflow tasks.

Here, the computing-intensive task is mainly discussed. Its communication time is less than the computing time and thus can be neglected. The objective of scheduling in this paper is to choose $\min(f(v_j))$ resources by using time and trust, and the target function $f(v_j)$ is calculated using Equation (7). λ means the weight of task scheduling; the bigger the value of λ, the greater influence the trust has on task scheduling. trs_j indicates loss caused by the trust, which is acquired using Equation (8). $\{1-\Omega(m_j,t,c)\}$ represents the risk probability of resources; a bigger trust value means $\{1-\Omega(m_j,t,c)\}$ is smaller, implying that failures have a lower impact on task scheduling. The GWTS algorithm firstly selects for the key task the resource that has a minimum comprehensive function of execution time and trust. Then, according to the scale of task $BD(j)$ from big to small, it selects for the non-key task the resource that has minimum comprehensive function of completion time and trust. The calculations are presented in Equation (9).

$$trs_j = t_{jk} \times \{1-\Omega\big(m_j,t,c\big)\} \tag{8}$$

$$f\big(v_j\big) = \begin{cases} \lambda \times trs_j + (1-\lambda) \times t_{jk}, if\big(v_j \in CT\big) \\ \lambda \times trs_j + (1-\lambda) \times ect(v_j), if\big(v_j \notin CT\big) \end{cases} \tag{9}$$

To sum up, the grid workflow task scheduling algorithm based on critical paths under trust constraints is as follows:

Algorithm of scheduling tasks on resources using GWTS

Input: TG, GR
Output: Allocated m_n for each task v_j
Begin
1. For each task v_j from V_{entry} to V_{exit} in TG
2. Calculate TD(j) using (1)
3. End for
4. For each resource m_j
5. Calculate $\Omega(m_j)$ using (7)
6. End for
7. From V_{entry} to V_{exit} in TG
8. Calculate $ect(v_j)$ for each task using (2)
9. Find CP according to $ect(v_{exit})$ and add tasks to CT
10. Rank task according to TD(j)
11. While there is task not allocated
12. Select v_j which has the minimum TD(j)
13. Calculate trs_j using (9) for each resource in Q(j)
14. Calculate $f(v_j)$ using (10)
15. Select m_n for task v_j having min $f(v_j)$
16. End while
17. End

5. Experiment Design and Discussion

5.1. Grid Simulation Tool

Currently, the commonly-used grid simulator mainly has SimGrid, GridSim, MicroGrid, Bricks, and other simulation tools.

5.1.1. SimGrid

SimGrid is the research and development of Santiago Grid Research and the Innovation Laboratory of California

University in the United States. The distributed parallel scheduling is mainly provided with suitable models and abstractions, and accurate simulation results are given. SimGrid includes two versions. One is SG, which is suitable for simulation based on DAG centralized scheduling and provides an underlying API to build simulation environments. The other is MSG, which is based on SG, provides the application-oriented API, and is suitable for simulating the scheduling of multiple independent tasks.

5.1.2. GridSim

Gridsim is a Java based grid simulation tool developed at Melbourne University in Australia, Rajkumar. It achieves resource allocation of the computational economy model. It also provides rich library functions as well as different parameter configurations for the heterogeneous distribution of grid entities, such as resources, applications, users, schedulers, and user agents. GridSim can simulate global computing and network resources and provide tasks of virtual processing and other functions. It uses a layered approach to simulate grid task scheduling, focusing on one aspect of the function of each layer. GridSim tools include multiple entities, such as grid resources, networks, and information services, and the simulation of these entities with the aid of the interface is provided by SimJava. Communication is facilitated by sending and receiving event objects to simulate the interaction behavior of the grid bottom layer. This layer includes Gridlet and the GridSimRandom grid system model, and it uses the model to achieve a scheduler model. The upper layer is the application layer, which mainly includes the different application scenarios, including the establishment of the simulation experiment, the establishment of the user model, and so on. Resource entities are mainly composed of GridResource and a series of auxiliary classes in the GridSim toolkit. They contain many distributed and heterogeneous resources. The heterogeneity of resources is mainly reflected in the processing speed, number of processors, processing cost, number of machines, local load parameters, internal scheduling strategy, and so on. Different parameters can be established as SMP, PC, cluster, and other different resource models. The layer also includes the behavior of different entities, such as task submission, resource registration, resource query, and task processing.

5.1.3. MicroGrid

MicroGrid is the leading development of the California University Santiago Parallel System Architecture group. Existing physical resources are used to simulate the virtual grid environment. The important part of MicroGrid is the simulation engine, which is built on the basis of the parallel distributed simulation engine. By implementing a completely controllable virtual grid environment, we can design a grid system and conduct a performance evaluation of the system.

5.1.4. GridNet

GridNet mainly simulates the dynamic data replication strategies in data grids. A GridNet replica decision based on the cost estimation model is proposed. GridNet is based on the modular design. The GridNet simulation environment mainly includes the following three types of nodes: clients, servers, and cache nodes. The client is responsible for generating the data access request. The server side represents the storage node. The cache node represents an intermediate storage node, which is used to copy the data stored on the server.

5.1.5. Bricks

Bricks is the research and development of the Tokyo Institute of Science in Japan. The main simulation provides remote access to the scientific computing library and client server types of computing systems. Bricks is composed of a wide area computing environment and a scheduling unit, which can be used to test various scheduling algorithms. However, centralized global scheduling has certain limitations. Here, the Gridsim tool is utilized to construct the simulation environment. Next, the proposed GWTS algorithm is compared with the HEFT algorithm and CPOP.

5.2. Introduction of Related Algorithms

The HEFT algorithm [12] is a classical scheduling algorithm in heterogeneous environments. It divides the scheduling process into two phases: weight distribution and task allocation. First, according to the task's execution time and dependence relationship, assign different weights to tasks from entry task to exit task. Then, in the other phase, allocate tasks to the nodes of resources with the earliest ending time from smallest to largest weight. The CPOP algorithm [13] pre-emptively considers key tasks. Assign different weights in accordance to the length. Later, in the task allocation period, start from the entry task. Assign priority to the task on the key path, and assign the key task to the nodes of resources with the shortest execution time. If it is a non-key task, distribute it to the nodes of resources with the earliest ending time.

5.3. Experimental Environment Configuration

The experiment uses the GridSim packet as a foundation to create a grid simulation environment [14], a heterogeneous multi-cluster grid system model. The model contains ten resource sites (CE_1-CE_10). The configuration of each resource site is listed in detail in Table 2. The experimental environment randomly generates a workflow, which contains different subtasks.

Table 2. Grid system model configuration of simulation experiment

Resource name	Number of CPU systems	MIPS
CE_1	128	450
CE_2	64	300
CE_3	256	410
CE_4	64	340
CE_5	512	230
CE_6	256	420
CE_7	128	230
CE_8	128	340
CE_9	64	450
CE_10	256	300

The resources attribute increases the reliability level (R, N, U, M), and the delay and failure rate of the reliability level are shown in Table 3. The performance of the resource is not stable, and the failure probability of the resource is expressed as the failure probability of the resources, which reflects the possibility of the unilateral revocation of the resource.

Table 3. Reliability level of resources

Reliability level	Delay rate	failure rate
Reliable (R)	[0, 5]	[0.01, 0.1]
Normal (N)	[5, 20]	[0.1, 1]
Unreliable (U)	[20, 50]	[1, 10]
Malicious (M)	[50, 100]	[10, 100]

The influence of direct trust and recommendation trust is equivalent in the initial time, so the confidence factor of Equation (7) is $\alpha = 5$. According to the reliability level defined in Table 2, the experiment sets two resource scenes, as shown in Table 4.

Table 4. Scene distribution of reliable resources and non-reliable resources

Scenario	*R*	*N*	*U*	*M*
SC1 (Reliable)	85%	7%	5%	3%
SC2 (Unreliable)	40%	30%	20%	10%

Two indicators are used to evaluate the performance of the algorithm.

(1) MakeSpan

Span means the time span from the beginning of the task to resource execution until all tasks are complete; the less time the completion takes, the better performance the algorithm and grid system can achieve.

(2) Task execution success rate

The workflow execution success rate analyzes the effect of trust QOS on task scheduling. S means the number of successful scheduling, and n is the total number of scheduled tasks. The calculation method is shown in Equation (10).

$$suc\_rate = \frac{\sum_{i=1}^{m} s}{n} \tag{10}$$

5.4. Experimental Result Analysis

Here, a simulation program is compiled with the GRIDSIM tool, which includes a relative resource description, task description, and relevant scheduling algorithm. The experiment sets SCI and SC2 scenes. Adjust the influence of trust on the scheduling according to different values of λ, which is experimentally implemented as below:

(1) In SC1, when $\lambda = 0.5$, the schedule contains 20, 50, 100, 200, and 500 workflow tasks. The completion time of the different algorithm is shown in Table 5.

Table 5. The completion time of the task of $\lambda = 0.5$ in SC1

Task number	HEFT	CPOP	GWTS
20	170	160	158
50	420	420	400
100	670	680	631
200	1567	1459	1412
500	2644	2645	2356

(2) Schedule different workflow tasks of $\lambda = 0.9$ in SC2. The completion time of the algorithm is shown in Table 6.

Table 6. The completion time of the task of $\lambda = 0.9$ in SC2

Task number	HEFT	CPOP	GWTS
20	233	190	160
50	520	500	456
100	850	812	617
200	1756	1765	1570
500	3200	3250	2611

(3) Schedule different workflow tasks of $\lambda = 0.9$ in SC2. The success rate of workflow execution is shown in Table 7 and Figure 4.

Table 7. The success rate of workflow execution of $\lambda = 0.9$ in SC2

Task number	HEFT	CPOP	GWTS
20	61	63	66
50	63	67	73
100	70	69	79
200	72	74	80
500	75	80	89

The major shortcoming of the HEFT algorithm is its inability to reduce scheduled MAKESPAN during each scheduling, because the ready task with the earliest and minimum starting time is not necessarily the key task. Tasks preferentially scheduled to the processing resource node take time; as a result, the key task cannot obtain the resource node with faster processing speed when the resource is selected, increasing the MAKESPAN of the whole workflow. However, the HEFT algorithm and CPOP algorithm do not take into account the trustability of resources. The proposed GWTS algorithm made improvements of the two methods by merging the influence of QOS trust and QOS time on scheduling.

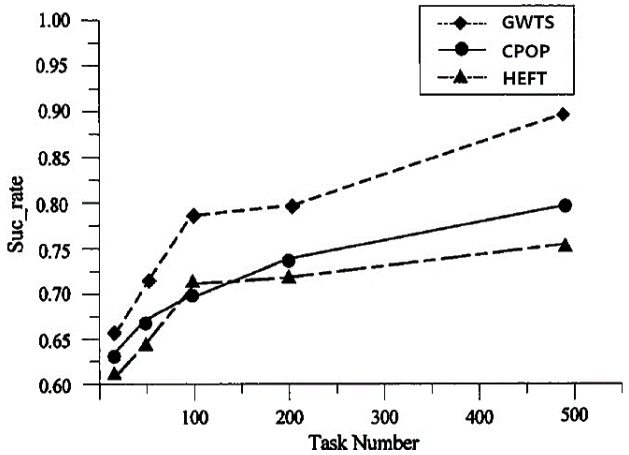


Figure 4. The success rate of workflow execution of $\lambda = 0.9$ in SC2

If tasks on the key path are distributed to the resource that has the shortest execution time and worst confidence degree, it would lead to re-scheduling due to resource failure. If at this moment tasks are assigned to other candidate resources, the key task needs to wait, extending the completion time of the whole workflow. The GWTS algorithm considers the task's average execution time on each candidate resource when deciding the key path. By integrating resource trust and execution time, it distributes the resource preferentially for the key task, shortening the execution time of the workflow. Table 7

reveals that the proposed GWTS algorithm considers resource credibility on the key path and increased task execution successful rate. More tasks indicate that the GWTS algorithm achieves a higher task execution success rate. Through adjustment of the proportion of trust according to the value of λ, the influence of trust on scheduling is well-assured.

6. Conclusions

This paper proposes a grid workflow task scheduling algorithm in grid environments. The GWTS algorithm first calculates the inverse depth of the task and then determines the key task as well as the trust degree of the resources. Finally, task scheduling is performed on the execution time and trust degree. The performances of the GWTS algorithm and other algorithms are analyzed in different scenarios. Experiments show that, compared with the HEFT algorithm and CPOP algorithm, the GWTS algorithm reduces the running time of the whole workflow and improves the successful execution rate by 5-15%.

Acknowledgements

This work was supported by the project of the Nature Scientific Foundation of Heilongjiang Province (No. F2016038).

References

1. X. Li, Z. G. Hu, and Z. K. Yan, "Research and Development of Grid Workflow Scheduling Algorithm based on Deadline Satisfaction," *Computer Research and Development*, Vol. 5, pp. 877-884, 2011
2. X. N. Yang, F. Liu, and T. G. Tian, "A Grid Scheduling Algorithm based on Improved Genetic Algorithm," *Computer and Digital Engineering*, Vol. 46, No. 9, pp. 1786-1789, 2018
3. M. Chang and L. Qiu, "An Improved Grid Task Scheduling Algorithm," *Microcomputer Applications*, Vol. 33, No. 33, pp. 30-36, 2017
4. Y. L. Kang and J. A. Zhang, "Packet Scheduling Algorithm for Grid Tasks based on QoS," *Software*, Vol. 37, No. 10, pp. 25-28, 2016
5. L. Q. Pan, Y. Q. Zhang, and Z. J. Mao, "Research on Task Scheduling Algorithm in Grid Environment," *Fujian Computer*, Vol. 32, No. 8, pp. 1-12, 2016
6. J. Cao and S. Q. Wang, "Emergency Resource Grid Scheduling based on Game Strategy," *Logistics Science and Technology*, Vol. 39, No. 1, pp. 1-15, 2016
7. Y. C. Hua and G. K. Zhang, "Research and Analysis of Grid Computing Model based on Big Data," *Information Technology and Informatization*, Vol. 10, pp. 230-231, 2015
8. D. H. Liu, "Brief Analysis of Task Scheduling Algorithm in Grid Environment," *Information Security and Technology*, Vol. 5, No. 12, pp. 33-35, 2014
9. D. C. Wang and Y. Gong, "Task Scheduling Algorithm based on Multi-QoS Constraints in Grid Environment," *Journal of Changchun University of Technology* (*Natural Science Edition*), Vol. 37, No. 5, pp. 134-136, 2014
10. D. W. Wang and S. Jiang, "An Improved Workflow Scheduling Algorithm in Grid Computing," *Computer Technology and Development*, Vol. 2, pp. 71-75, 2014
11. X. H. Zhu, "A Grid Workflow Scheduling Algorithm for Communication Overhead," *Journal of Jianghan University* (*Natural Science Edition*), Vol. 3, pp. 278-282, 2015
12. P. Radu and W. Marek, "Bi-Criteria Scheduling of Scientific Grid Workflow," *IEEE Transactions on Automation Science and Engineering*, pp. 364-380, 2012
13. C. Li and Q. M. Zhu, "Review of Research on Grid Workflow Scheduling," *Computer Application and Software*, pp. 279-283, 2010
14. H. J. Cao and X. H. Shi, "Efficient and Dependable Scheduling of DAG Workflow Job in Grid," *High Performance Computing and Communication*, pp. 201-223, 2012

Feng Liu received his B.S degree in computer science from Heilongjiang University of Science and Technology. He received his M.S degree in computer science from Liaoning Technical University. He is currently an associate professor at Heilongjiang University of Technology. His research interests include databases and algorithms.