*Send Orders for Reprints to reprints@benthamscience.ae*





# Dynamic Replication of Fault-Tolerant Scheduling Algorithm

Wang Hongxia* Fang Haoran and Qiu Xin

*School of Computer Science and Engineering, Shenyang Ligong University, Shenyang, 110159, P.R. China*

**Abstract:** In order to reduce the impact on the grid scheduling caused by the resource error, for complex dependent tasks, designed a dynamic replication of fault-tolerant scheduling algorithm (DRFT). This algorithm build task model by applying hypergraph theory, using primary-backup task as backup mode, take the active execution mode combined with the passive execution mode to perform the backup task, set up dynamic backup level according to the level of importance and resources security condition of dependent task, pursuit time of the task execution is minimized, the simulation result shows that the algorithm can still be accomplished scheduling quickly in case of error caused by the resource, to make up the shortage of common algorithms in terms of fault-tolerant.

**Keywords:** Dependent task, dynamic replication, fault-tolerant scheduling, hypergraph.

## 1. INTRODUCTION

In order to improve the reliability of grid system, commonly used error prevention, error verification, error prediction and fault-tolerant four methods [1-3]. Among them, the fault-tolerant is a common and efficient solution. Fault-tolerant method includes retry, N version of the program design, the recovery block, primary-backup scheduling technology, the primary-backup scheduling technique is the most commonly used of Fault-tolerant method [4, 5].

The primary-backup scheduling technique is backup task on other resource, if the main task execution error, began to perform the backup task. The execute method of backup task can also be divided into active backup method, passive backup method and the combination of active and passive backup method [6-8]. Active backup method, refers to the resource without fault are also required to perform the backup task, this will undoubtedly increase the burden on resource; passive backup method is a method of execution only after the main task execute error, the fault-tolerant of such a system is not very high; the combination of active and passive backup method is a kind of backup task execution mode combine with active execution and passive execution, can achieve a relatively small cost but also improve the fault-tolerant of the system, so this paper uses the combination of active and passive backup method.

The researches apply primary-backup method to improve fault-tolerant on grid system are as follows, literature [9] were designed meta-task and dependent task of primary-backup fault-tolerant scheduling algorithm, though effectively achieve fault-tolerant, but without considering the size of the burden the backup to resource; literature [10] in order to minimize backup costs as the main target, designed a differed backup- copy of primary/backup copy, by delay backup-copy start time as far as possible , to achieve the purpose of minimize the duty cost, but the article does not consider the importance of the task; literature [11] designed a primary/backup-copy scheduling algorithm based on dependent task, to seek the balance between the minimal cost and the earliest completion of the backup task, but it can not achieve the backup cost and the completion time are all small. In a word, the above article are not consider the backup issues according to the importance of the task, the effect of minimize the backup level and execution time is not very obvious.

Hypergraph theory [12] was first proposed by C. Berge in 1970, study on the multiple relationship. Application hypergraph theory to build a model, provides a lot of convenience in describing the problem, conducive to solving practical problems, therefore, this paper use hypergraph model to express dependencies between tasks, not only can express the basic information of task clearly, but also can express the interdependence between tasks.

Therefore, this paper aimed at the problem of task scheduling in grid fault-tolerant, designed a dynamic replication of fault-tolerant scheduling algorithm (DRFT), this method takes into account the complexity of dependent task, based on the application of hypergraph theory build task model, using primary-backup task as backup mode, take the active execution mode combined with the passive execution mode to execute the backup tasks, set up dynamic backup level according to the level of importance and resources security condition of dependent task, pursuit time of the task execution is minimized. Compared with the classical algorithm GS-Min-min algorithm, even in the case of task execution error, this method is still able to quickly

1874-110X/15 

complete the scheduling, reduce the impact of resource failure for task scheduling.

## 2. MODEL CONSTITUTION

In this paper, the backup task execution mode using a combination of active and passive execution mode. In order to reduce the backup excessive use of resources, there is only one backup task, the backup task is split into active execution part and passive execution part by calculation unit, execution mode is the combination of active execution and passive execution, that is the active execution initiative implement, passive execution passive implement. The active part of backup tasks initiative implement, only when the main task execution fails, another part of the backup task, that is passive part, will be activated to execute. So to a certain extent can reduce backup costs, and if the main task execution fails, backup task can be completed quickly. Fig. (**1**) shows the primary-backup task.

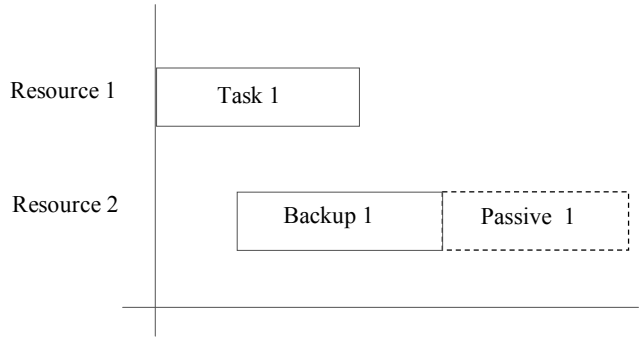


**Fig. (1).** The Primary-backup task.

### 2.1. The Main Task of The Hypergraph Model

Because there are interdependencies between tasks, therefore fusion hypergraph theory building the main task hypergraph model, task hypergraph model consist of task note and hyperedge, hyperedge can detailed express dependencies between tasks. The task mainly consider the dependent task, and all are the computing task, dependent task have interdependencies between tasks, only the current task all predecessor task is completed, this task can execute. Because of the interdependencies between tasks, hypergraph model is directed hypergraph. Hypothesis resource transmission speed very quickly, ignore the traffic load problems between tasks. Tasks node consists of a three-element-array, that is the task ID, task computation and the state of the task, the state of the task includes five states: idle, have been matched, waiting, execute, finish.

The primary task of directed hypergraph model is described as follows:

RH= (RX,RE) is the task model of the hypergraph representation.

a) RX ={$rv_1$,$rv_2$,……,$rv_{rn}$} is the set of node task, where $rv_i$={*rID*, *rC*a, *rS*} is a attribute set of task node, i [1,rn].

1) *rID* is the task ID.

2) *rCa* is the task computation.

3) *rS* is the state of the task, the state of the task is divided into idle state, has been matched state, wait state, execution state and finish state, the state of the task changes with scheduling process. Expressed as follows:

rS={rfree, rmatch, rwait, rwork, rdone}

b) RE= {$re_1$, $re_2$, ……,$re_m$} is a set of hyperedge, m=|RE| is the number of hyperedge, hyperedge $re_i$ composed by a three-element-array (*Pre*, *Des*, $rv_i$). *Pre* is all the predecessor task of the task set $rv_i$, predecessor task is that have a direct dependencies of task $rv_i$, if and only if after all tasks finished in the set *Pre*, task $rv_i$ can execute.

*Des* is a collection of all post-task of task $rv_i$, post-task is that have a direct dependencies of task $rv_i$, task $rv_i$ finish executing is a prerequisite for any of the tasks can be executed of *Des*.

Any task node has its corresponding hyperedge, the weight $D_i$ of hyperedge $re_i$ is the number of steps of task node $rv_i$. The number of steps is the maximum number of steps task node to the exit node. The task node has no post-task is the exit node.

Directed hypergraph task model is shown in Fig. (**2**).

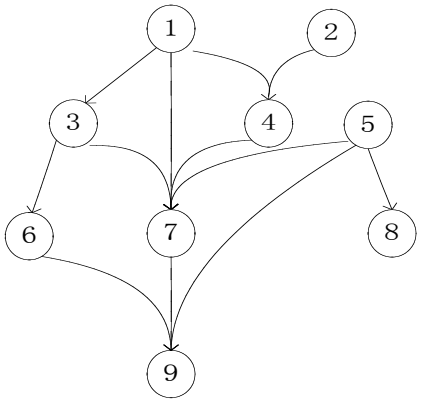


**Fig. (2).** Directed hypergraph task model.

As shown in Fig. (**2**), task $rv_1$,$rv_2$,……,$rv_{rn}$, hyperedge $re_1$, $re_2$, ……,$re_m$, e.g. $re_3$ ({$rv_1$}, {$rv_6$, $rv_7$}, $rv_3$), the weight of the $re_3$ is $D_3$=2; $re_7$ ({$rv_1$ ,$rv_3$, $rv_4$, $rv_5$,}, {$rv_9$}, $rv_7$), the weight of the $re_7$ is $D_7$=1.

### 2.2. The Backup Task Model

Backup task are copy from the main task, so the backup task inherits its main task *ID*, computation and dependencies between tasks, the backup task is one to one corresponding relation to main task. In addition, the backup task should also have the properties main task do not have, that is the active part and the passive part.

Backup task model and the main task model are the same, as follows:

**RH′**= (***RX′***, ***RE′***) is the hypergraph representation of backup task.

a) ***RX'***= {$rv_1'$, $rv_2'$, ……, $rv_{rn}'$} is the backup task node set, where $rv_i'$= {*rID'*, *vrCa'*, *rCa'*, *rS'*}, $i \in [1, r_n]$.

1) *rCa'* is backup task computation of active execution, *vrCa'* is backup task computation of passive execution.

2) ***rS'***={*rfree, rmatch, rwait, rwork, vrwork, rdone*} is the states of the backup task, *rwork* is active execution working state of backup task, *vrwork* is passive execution working state of backup task.

b) ***RE'***= {$re_1'$ $re_2'$ …… $re_m'$} is a set of hyperedge, *m*=|RE| is the number of hyperedge, hyperedge $re_i'$ (*Pre*, *Des*, $rv_i'$), here consistent with the main task model.

### 2.3. The Resource Model

Resource node constitute by resource *ID*, resource handling, resource load and the safety factor of resource, resources are described as follow:

**RE**= {$v_1$, $v_2$, ……, $v_n$} is a set of resource node, where $v_i$ is the *i-th* resource, $\boldsymbol{v_i}$= {*ID*, *P*, *Load SA*} is a resource properties collection, $i \in [1 \quad n]$.

*a) ID* is the resource *ID*.

*b) P* is the ability of resource processing.

*c)* Load is a resource load, resource load include the actual load and the virtual load. The actual load is the computation of backup task active execution part, the virtual load is the computation of backup task passive execution part. Resource load value equal to the actual load value, that is the load backup tasks actually executed.

*d) SA* is the safety factor of resource, set the safety factor for the resource, can dynamically view the security of resource, has a direct effect on the scheduling success rate.

$$SA_i = \frac{\text{Number of mission success}}{\text{Number of mission accept}} \tag{1}$$

## 3. SCHEDULING STRATEGY

### 3.1. Backup Level Calculation

Fault-tolerant backup for dependent tasks, with the influence extent of task failed to post-task execution, dynamically set of fault-tolerant backup level, if the task error is larger, the more it is necessary to increase the level of backup and there is a direct relationship between the level of importance of task and the number of step, the number of step is the maximum number of steps task node to the exit node.

The larger the number of steps of the task, the greater degree of influence on the post- task, therefore, the extent of their backup should be more, and the resource security situation the main task of backup task also need to be considered, so we can get the formula of the level of backup:

$$B(i) = \begin{cases} \dfrac{\alpha \quad -SA_{rv_i}}{D_{max}+1-D_i} * rCa_i & D_i \neq 0 \\ 0 & D_i = 0 \end{cases} \tag{2}$$

$SA_{rvi}$ ----- The safety factor of the main task of $rv_i$ resource;

α ----- Adjustment coefficient, can adjust the level of backup;

*rCa* ----- The calculation of task;

*D* ----- The number of step;

When the number of step is not zero, the backup level have a relationship with task step, computation and its main task the safety factor of resources; when the step number is zero, task is dependent task of exit node task, this task has no great influence on the post-task, so is no need to backup, that is backup level is zero.

Main task sequencing and backup level calculation process:

//: sequencing

for (all main tasks) {

Breadth-first traversal dependent task graph to get the number of step for each task;

Consolidated the same step of hyperedge task and in descending order;

hyperedge within the same task, in descending order by priority value formula;

}

//: Backup level calculation

for (all backup tasks) {

Inherit their main task dependencies and the number of step;

Calculate the level of the backup;

}

### 3.2. Match Scheduling

Breadth-first traversal task graph, in descending order based on the mission maximum number of step, and more than one task at each step, so every step needed to sort. Because of dependencies between tasks, only a predecessor task completes, all post-task can be executed, therefore, the execution time of a predecessor task determines the start time of the post-task, if not considered the resource processing capability, the execution time of the task is only concerned with the computational task. So the priority value to each task is equal to the maximum value of the predecessor task computation and the current task to calculate together, see the formula (3).

$$PR_i = \max\{rCa_j + rCa_i\} \quad , \quad rv_j \in \text{Pred}(rv_i) \tag{3}$$

In descending order according to the priority value of the task, therefore, the task of sort result determine task priority matching. Main task and backup task can not be on the same resource, because there are different matching requirements between main task and backup task, so the main task and backup task using different matching mode

a) The main task of matching

According to the task of sort result, match scheduling start sequentially, matching with the resources of smallest SL.

$$SL_i = \alpha * \frac{Load_i - Load_{min}}{Load_{max} - Load_{min}} + \beta * \frac{SA_{max} - SA_i}{SA_{max} - SA_{min}} + \gamma * \frac{P_{max} - P_i}{P_{max} - P_{min}} \quad (4)$$

$Load_i$ ----- Load of resource $i$;

$SA_i$ ----- The safety factor of Resource $i$;

$P_i$ ----- Processing capacity of resource $i$.

By equation (4) shows, a small value of SL indicates that the resource load is small, high safety coefficient and strong processing ability, hoping task to scheduling to such resources.

b) Matching backup task

Backup task match resource , requires matching high safety coefficient and small load resource, the most important is that this resource can not be consistent with the main task resource.

The backup task scheduling sequence, followed by the main task. That is $rv_1$, $rv_1'$, $rv_2$, $rv_2'$, ……, $rv_{rn}$, $rv_{rn}'$.

If the main task and the backup task execution error, need to return to the task waiting queue, reallocation of execution. If the backup task has not been executed or being executed, and its main task has been finished, stop execute backup task, this can reduce backup costs to a certain extent.

Dynamic replication of fault-tolerant scheduling algorithm (DRFT) scheduling process:

Let T be a task waiting queue, copy of each task;

```
//: sequencing
//: Backup level calculation
for (each task rv_i of T) {
    main task rv_i match the minimum value resources of SL;
    Backup tasks rv_i '() {
        Looking for a high safety coefficient resource;
        Exclude resource of the main task;
        Matching resource to which the load is minimal;
        Scheduling tasks rvi to resources v_j;
    }
    Delete tasks rv_i, rv_i ' in T;
    Updated resource load Load, update task state;
}
if (task execution success) {
   if (main task rv_i execution success) {
       Output scheduling result;
   }
   else if (backup tasks rv_i 'execution success) {
       Output scheduling result;
   }
}
else {
   Task returns to the waiting queue T, reassigned;
}
```

Backup task the active execution part is the actual load, the passive execution part is the virtual load. Resource load is the actual load, but on condition when the main task execution error, passive part is activated to execute, at this point the passive part of the load, which is the virtual load will be transformed into the actual load, so the actual load and the virtual load is constantly changing.

## 4. PERFORMANCE ANALYSIS

The backup task is an implementation of fault-tolerant technology, there is a direct relationship between the success of the backup task execution and backup task position, that is for a task the main task and its backup task are not on the same resource, as shown in Fig. (**3**). Main task and backup task of different tasks can be on the same resource, as shown in Fig. (**4**).

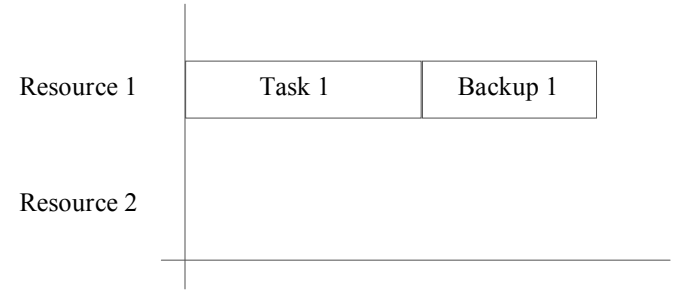


**Fig. (3).** Error backup task pattern.

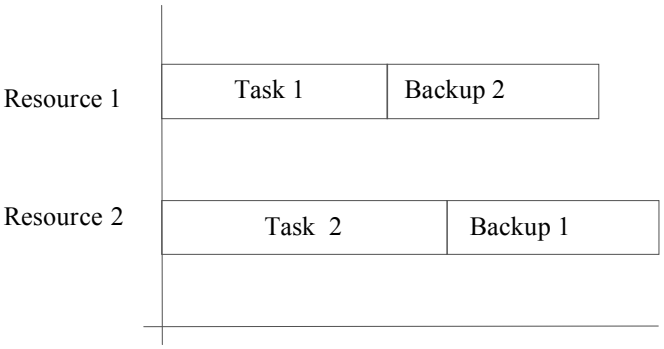


**Fig. (4).** Correct backup task pattern.

Variable level of fault-tolerant backup scheduling is a fault-tolerant algorithm based on dependent task, only under the condition of the predecessor task is completed, to be able to execute post-task. If the task error, and there are other task on the backup task resource, as shown in Fig. (**5**), if this task is being executed, as shown in Figure 6, stops the execution of the task, return to the task queue waiting to reallocate.

Fig. (**5**) and Fig. (**6**) shows that at t time, resource 1 is broken, at this time the main task of task 1 is not finished yet, the active part of the backup task of task 1 is being executed, at this time need to activate the passive part of the backup task, but the resource is not idle, the passive part is urgent for execution, thus, the task 2 will return to the task queue waiting to reallocate.

If the main task is completed in execution, it has not yet begin to execute backup tasks as shown in Fig. (**7**), or being

executed as shown in Fig. (**8**), at this time the backup task should stop execution, to a certain extent, can reduce the cost of backup, save the scheduling time.

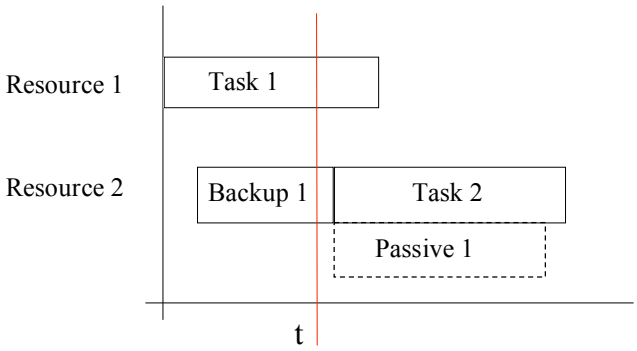


**Fig. (5).** Failure occurs when the backup task execution.

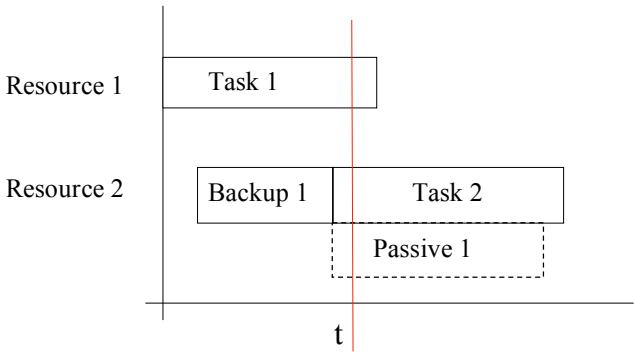


**Fig. (6).** Failure occurred after the backup task is completed.

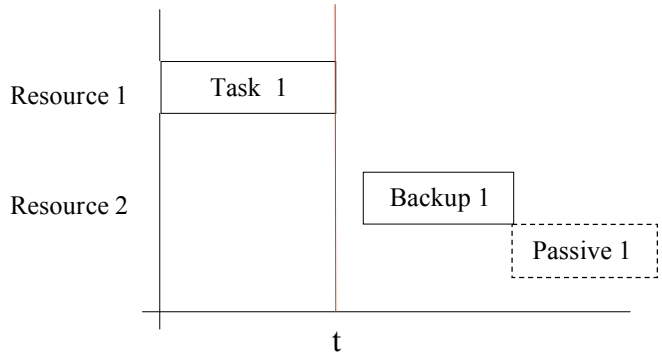


**Fig. (7).** Main task is complete, backup is not performed.

If the backup task execution error, at this time its main task is executing state, as shown in Fig. (**9**), if the main task can be executed successfully, will not affect the post-task execution; If the main task fails, this task will return to the task queue waiting to reallocate.

## 5. SIMULATION ANALYSIS

This article is designed a dynamic replication of fault-tolerant scheduling algorithm (DRFT), according to the importance level of task, the task is split into two parts, active execution and passive execution, with the flexibility to adjust the cost of backup, pursuit of the execution time is minimized. Compared with the classical dependent task scheduling algorithm GS-Min-min algorithm, in the case of resource error, DRFT algorithm scheduled for execution can still be completed in a short time.

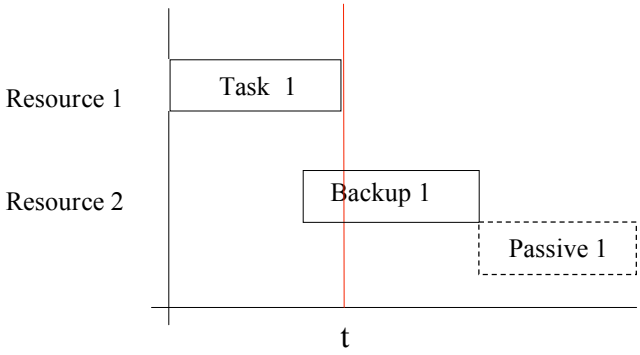


**Fig. (8).** Main task is complete, backup is being performed.

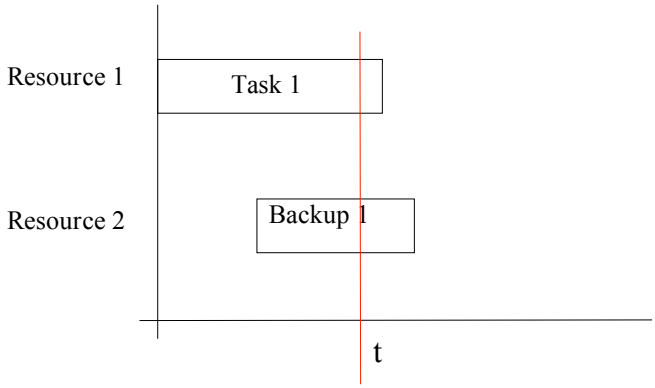


**Fig. (9).** Backup task error.

GS-Min-min algorithm is based on the dependent task scheduling, if this algorithm is applied in the fault-tolerant system, when task error occurred will return to the task queue waiting to reallocate.

Compared the scheduling simulation experiment of DRFT algorithm and GS-Min-min algorithm, according to the task dependency of Fig. (**2**), build different properties of the task and the different attributes of the resource, in the case of the probability of error resource, comparing the maximum completion time Makespan.

Assuming computational task in the range of 250 to 350, resource processing capacity in the range of 30 to 60, resource safety factor between 0.4 and 0.7, at the moment t1=9S, resource 4 fails, at the moment t2=17S, resource 6 fails, assuming resource does not recover after the failure. Fig. (**10**) is DRFT algorithm Gantt chart, Fig. (**11**) is GS-Min-min algorithm Gantt chart, from the two figures it is clear that even after the resource fails, DRFT algorithm still complete the execution of 23S, but GS-Min-min algorithm finished executing using 29S.

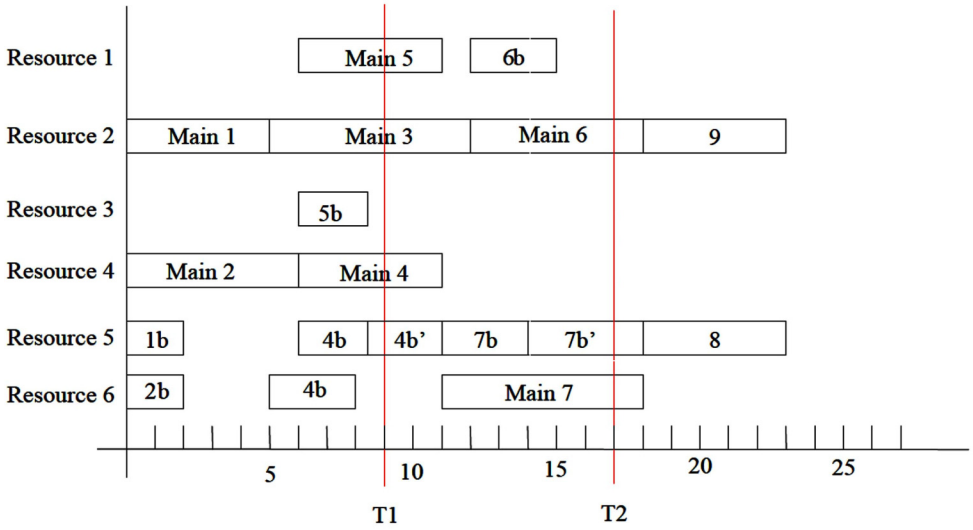


**Fig. (10).** DRFT algorithm gantt chart.

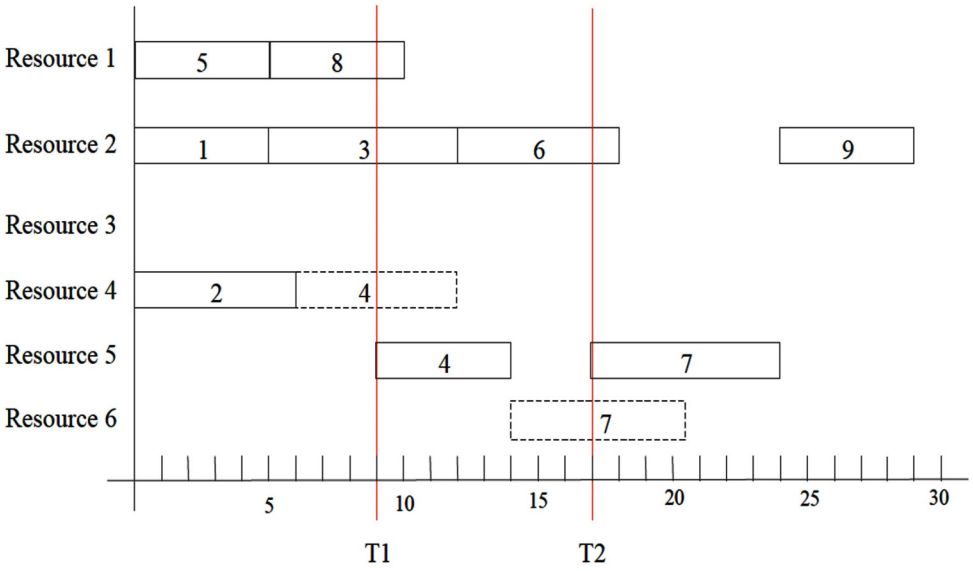


**Fig. (11).** GS-min-min algorithm gantt chart.

Resource error will greatly affect the operation of post-task, and can not predict the probability of resource error, Fig. (**12**). shows with the probability of resource error increasing, the maximum completion time of DRFT algorithm and GS-Min-min algorithm. It was concluded that the greater probability of resource error, DRFT algorithm is more superior.

In the grid environment, dynamic resource cannot accurately predict, the probability of resource error can only be based on the state of past to predict, method to deal with resource emergency is important, take dynamic backup level scheduling algorithm of primary-backup, can reduce or even does not affect the post-task performance. Analysis the simulation of DRFT algorithm and GS-Min-min algorithm, can be clearly seen the superior of the DRFT algorithm.

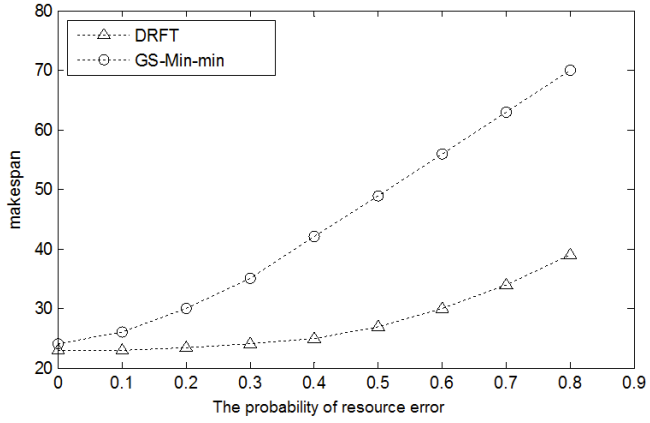


**Fig. (12).** The probability of resource error impact on Makespan.

## CONCLUSION

This article designed the dynamic replication of fault-tolerant scheduling algorithm (DRFT), use task relationship caused by the importance of the task and the security of resource of dependent task as parameters, constitute a variable level of backup and combine with active and passive of primary-backup scheduling algorithm, dynamic flexibility to adjust the level of backup tasks, pursuit of scheduling time is minimized, at the same time the fault-tolerant algorithm perform excellent. Compared with the classical algorithm GS-Min-min algorithm, even in the case of error resources, the algorithm can still finish scheduling quickly.

## CONFLICT OF INTEREST

The author confirms that this article content has no conflict of interest.

## ACKNOWLEDGEMENTS

This research was partially funded by the National Natural Science Foundation of China (Grant No. 61301256), supported by Program for Liaoning Excellent Talents in University (No. LJQ2013021).

## REFERENCES

[1] M. Amoon, "A fault-tolerant scheduling system for computational grids," *Computers & Electrical Engineering*, vol. 38, no. 2, pp. 399-412, 2012.

[2] K.J. Naik, and N. Satyanarayana, "A novel fault-tolerant task scheduling algorithm for computational grids," *In: 15th International Conference on Advanced Computing Technologies*, Rajampet , pp.1-6, 2013.

[3] S. Jain, and J. Chaudhary, "New fault tolerant scheduling algorithm implemented using check pointing in grid computing environment," In: *International Conference on, Optimization, Reliabilty, and Information Technology,* Faridabad, pp. 393-396, 2014.

[4] S. Xu, "Fault-tolerant computing systems," Hubei:Wuhan University Press, pp.113-115, 2010.

[5] W. Luo, " A Real-Time Fault-Tolerant Scheduling Algorithm of Periodic Tasks in Heterogeneous Distributed Systems," *Chinese Journal of Computers*, vol. 30, no. 10, pp. 1740-1749, 2007.

[6] C. Yang, G. Deconinck, and W. Gui, "Fault-tolerant scheduling for real-time embedded control systems," *Journal of Computer Science and Technology*, vol. 19, no. 2, pp. 191-202, 2004.

[7] Q. Zheng, B. Veeravalli, and C.K. Tham, "On the design of fault-tolerant scheduling strategies using primary-backup approach for computational girds with low replication costs," *IEEE Transactions on computers*, vol. 58, no. 3, pp. 380-393, 2009.

[8] A. Benoit, M. Hakem, and Y. Robert, "Contention awareness and fault-tolerant scheduling for precedence constrained tasks in heterogeneous systems," *Parallel Computing*, vol. 35, no. 2, pp. 83-108, 2009.

[9] H. Liu, "Research On Primary-Backup Based Fault-Tolerant Scheduling Algorithms For Cloud Computing," M.D. thesis. Zhejiang Gongshang University, On Hangzhou, 2010.

[10] W. Luo, "A Real-Time Fault-Tolerant Scheduling Algorithm for Distributed Systems Based on Deferred Active Backup-Copy", *Journal of Computer Research and Development*, vol. 44, no. 3, pp. 521-528, 2007.

[11] W. Jing, "Fault-tolerant scheduling algorithm for precedence constrained tasks", *Journal of Tsinghua University(Science and Technology)*, vol. 51, no. S1, pp. 1440-1444, 2011.

[12] C. Berge, "Graphs and Hypergraphs," North Holland, Amsterdam. 1973.

Received: June 10, 2015 Revised: July 29, 2015 Accepted: August 15, 2015