Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.12785/amis/070636

Rescheduling Oriented Dependent Tasks Spread Domain Computing

Tingwei Chen[1,*], *Hongning Zhu*[1], *Yu Dai*[2] *and Xianwen Hao*[3]

[1] College of Information, Liaoning University, Shenyang 110036, China
[2] College of Software, Northeastern University, Shenyang 110819, China
[3] China Mobile Group Liaoning Company Limited, Shenyang 110179, China

Received: 28 Mar. 2013, Revised: 30 Jul. 2013, Accepted: 31 Jul. 2013
Published online: 1 Nov. 2013

Abstract: To narrow the scope of rescheduling tasks is one of the effective ways to improve the grid dependent tasks rescheduling efficiency. For how to determine the scope which should improve the efficiency of rescheduling problem as far as possible without affecting the application performance, this paper proposed the rescheduling tasks spread domain concept and its method of computation. Beginning with a minimum tasks set to rescheduling, the computing process is oriented by resource share conflict and data transmission dependent of tasks, and limited by the degree of task to optimize the performance of the whole application. Experimentation results show that static scheduling strategy could maintain the performance advantages compare with the dynamic strategy, thus the efficiency of proposed rescheduling tasks spread domain is validated.

Keywords: Grid, dependent tasks, rescheduling, rescheduling tasks spread domain.

1 Introduction

Because the existence of dependent relation between tasks, the resources assignment will have an impact on the other tasks in the grid dependent task scheduling process. Therefore, in order to improve the performance of grid application, grid dependent tasks scheduling generally adopts static scheduling strategy, this strategy is generate a global scheduling plan before runtime, and rescheduling at the time of resources changing [1–5].

In order to achieve the scheduling goal of optimal application performance, it needs global optimization method, the grid dependent tasks rescheduling will take all unfinished tasks as scheduling object, so the grid dependent tasks rescheduling problem is a NP complete problem [6]. The solution to this problem will influenced by quantity of the gird application resources and tasks. If the quantity of the gird application resources and tasks is too large, the solution to this problem will requires a long time. At present, the study in this filed is using heuristic algorithm or AI optimization algorithm to improve the efficiency of the algorithm. However, rescheduling happens at application operating time, the executing efficiency of rescheduling algorithm is not only influence the cost of rescheduling, but also affect the performance of application, therefore, compared with the initial scheduling, rescheduling efficiency demanded more strict. In rescheduling triggered frequently cases, only improve the efficiency of rescheduling optimization algorithm is hard to ensure efficiency of the grid dependent tasks rescheduling, apart from this, also need to consider to reduce the size of the resources and tasks involved in rescheduling. However, in general sense, to reduce the size of the resources and tasks will influence the application performance [6]. Therefore, how to determine the involved scope of rescheduling tasks is a key problem in the gird dependent tasks rescheduling.

In view of this problem, this paper analyses the degree of data transmission dependent between tasks, the degree of resource sharing conflicts between tasks and the degree of task support to performance optimization in dynamic environment, put forward rescheduling tasks spread domain computation method which initial scope is the set of rescheduling tasks as minimum required, oriented as the sharing conflicts and data transmission dependent between tasks, bordered by the degree of task support to performance optimization, defined with gradually expand the scope of the rescheduling task. To calculate by this

[*] Corresponding author e-mail: t.w.chen@163.com

method, suitable rescheduling tasks scope can be obtained, strongly influence each other within the scope, weakly influence each other without the scope, thus, the rescheduling of the tasks scope will improve the efficiency, without affecting the gird application optimization as much as possible.

Section 2 of this paper introduces related work, Section 3 describes the concept of rescheduling task spread domain and computation methods, Section 4 is a experiment part, in the end is conclusions.

2 Related work

Different researchers have different ideas to deal with static scheduling problems in grid environment.

One type take increase the prediction accuracy, reduce dynamic and multi-resources backup as target. Such as Plan Switching method, which constructing a series of activity graphs before application executed, each activity graph representing a scheduling plan, if an activity graph failure at runtime, other activity graphs will be selected. Actually it is a method that uses a lot of alternative scheme to increase the feasible and optimization of static scheduling plan. The problem is that it cannot get real operating tasks, resource information. This paper does not care of such solutions.

The other takes reduce the dependent of prediction accuracy, adapt to dynamic as target. It is mainly based on rescheduling plan adjusting strategy as means.

To a certain extent, SIL and MQD algorithm [2] are able to solve the dependent problem about the static scheduling algorithm of performance prediction accuracy, and have good performance for data-intensive and compute-intensive applications, only the algorithm orients the independent grid application type of Bag-of-Task, rather than the application of dependent tasks workflow.

Low-Cost rescheduling strategy [3] orients the application of dependent tasks workflow, consider rescheduling at the special key points when the application running. Its main purpose is to solve contract conflict when static global scheduling predictions about tasks are not accurate, without considering resources change.

DAG - Man [4] is the scheduling system of Condor - G project, which support dependent task scheduling and rescheduling, considering the resources change, but only as a fault-tolerant technique, passive processing reserved resources exit events, without considering other change of resources affect scheduling plan in the maintenance optimization aspect.

AHEFT algorithm [5] meanwhile consider various resources changes and the scheduling optimization influenced by performance prediction accuracy, and realized the adjustment strategy of scheduling plan based on HEFT heuristic algorithm, have good performance in high parallelism degree data-intensive applications. But this method in the adaptation degree of resources isomerism is completely consistent with the original HEFT algorithm, and also without considering the rescheduling frequent trigger problems.

To conclude, by now about the research of dependent task rescheduling, the solution to the low efficiency of rescheduling problem is limited to improve the efficiency of rescheduling algorithm itself. This paper puts forward the rescheduling task spread domain conception and its method of computation, besides rescheduling algorithm, by reasonably narrow the involved scope of rescheduling tasks to further improve the execution efficiency of rescheduling.

3 Rescheduling tasks spread domain and its method of computation

In order to improve execution efficiency of the grid dependent tasks rescheduling, this paper proposed the rescheduling tasks spread domain and its method of computation in the aspect of narrowing the scope of rescheduling tasks. The description of rescheduling tasks spread domain as follows.

3.1 Rescheduling tasks spread domain

Rescheduling tasks spread domain is a new concept proposed in this paper, which is the scope that rescheduling tasks considered, for this scope appears the change process of gradually expanding, which take minimal scope as center (as shown in figure 1), it can be visually called spread process of scope, so we define this scope as rescheduling tasks spread domain.

Definition 1: rescheduling tasks spread domain as to the application expressed by DAG $= \langle V,E\rangle$, rescheduling tasks spread domain is the task and the relation between tasks which consider in rescheduling process, in short of spread domain. It can be also defined as formal as $RTSD = \langle V_1,E_1\rangle$, among it, V_1 is a task node, the nodes in v_1 are all in V, the nodes that need rescheduling are in V_1, E_1 is the edge between tasks, for two nodes in V_1, if there is a transferring relation of data in E, then there is also the relation in the two nodes, that is $\forall v_1, v_2 \in V_1 \wedge \langle v_1, v_2\rangle \in E, \exists\langle v_1, v_2\rangle \in E_1$, for the two nodes in V_1, which are in transferring relation of data, they have the point spread relation, or they have dependent spread relation, or they have connective spread relation. That is $\forall v_1$, $v_2 \in V_1$, $\exists$ $(v_1 R_{point} v_2) \vee (v_1 R_{depen} v_2) \vee (v_1 R_{conn} v_2)$ (the three relations will be described in the back of this section)

This paper considers rescheduling tasks spread domain from the three respects.

(1) In order to shorter the application completing time, assign the effective resource or increase parallelism for task. The performance of task is different in various resources. So this paper establishes the scope considered

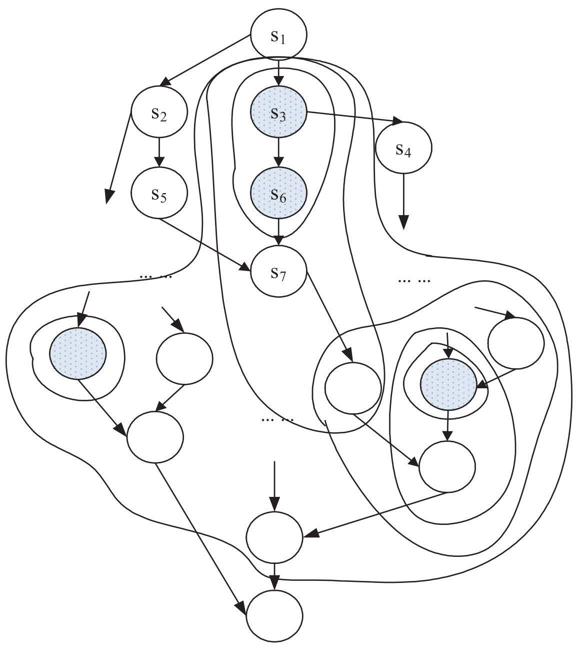

Fig. 1 The graph of rescheduling tasks spread domain

commonly in the rescheduling process according to the degree of competition to resource for tasks. Because the expanding process of spread domain shows in the task chart of DAG, it is a expanding process that rescheduling task discrete, non-continuous, therefore, this paper call this spread process as Point spread process, which is also called for computing process of point spread domain. The relation between tasks which the point spread process refer to is called for relation of point spread, recorded as R_{point}.

(2) For the view of improving the performance of dependent task, in the process of task redistribution, the dependent relation between tasks is supposed to consider, by reason of considering rescheduling tasks in the non-global scope, the dependent relation between the boundary nodes in the scope and nodes outside is supposed to consider. The greater dependence is, the greater requirement that scheduling tasks considered is. To a large extent, the advantages and disadvantages of performance optimization dependent on the advantages and disadvantages of dealing method, which uses in dependent relation between tasks. Therefore, in the view of this article, add dependent relation between tasks and extraterritorial tasks which the dependence degree between tasks satisfy a certain threshold on the spread domain boundary. This paper calls this spread process as dependent spread process, also computing process of dependent spread domain. The relation between tasks which dependent spread process refer to is called as relation of dependent spread recorded as R_{depen}.

(3) The dependent relation between tasks that within spread domain, which we get through the above two spread process have not totally embodied. Assuming that tasks rescheduling would be done on this domain, the phenomenon of time distribute deadlock will appear. According to this, if there is the predecessor and successor relation between the domain tasks, all paths between two task nodes should be added to the domain, including all the tasks and the dependent relation on the path. However, the spread domain handled by above process may not be connected, only lots of connected sub-graph, but there is no relation between connective sub-graphs, so in order to establish connected spread domain, pseudo-entrance nodes also should be added to ahead of many entrance nodes in each sub-graph, and the pseudo-export nodes should be added to back of many export nodes in each sub-graph to achieve connectivity, furthermore, the reasonable node and the pseudo-edge weight should be set for achieving the assumptions in the task graph of DAG, preparing the conditions for solving the rescheduling. New tasks also added to this process, so this process is a domain spreading process, this article call it as connective spread domain process; also call it as computing process of connective spread domain. The relation between tasks which connective domain spread process considered is called as connective spread relation, recorded as $R_{conn.}$.

3.2 Spread domain computing method

Spread domain computing is based on minimum task collection which needs rescheduling, forming single entry finally, and the task sub-graph with a single export. Its basic steps of the computing are as follows:

1. Get the minimum task set which needs rescheduling from rescheduling trigger case, combine with the task graph of DAG, forming the minimum spread domain;
2. Compute point spread domain on the basis of minimum spread domain. Sort as task which needs rescheduling and expected completion time of successor task that conflict with it on each resource (the compute method is shown in formula 1), add the resource which complete time earliest to the spread domain; And remove tasks from the spread domain, whose average expected completing time outside the average change cycle of resources environment;
3. Then based on the scheduling scheme, compute the interval time (the computation method is shown in formula 2) between tasks on each resource and sorting, add resource tasks in the largest interval time to spread domain, then according to the rule 1, remove the tasks of average expected finishing time (computation method is shown in formula 4) out of the average change cycle of resources environment from spread domain;
4. Dependent spread domain computing. On the basis of task graph DAG, compute the degree of dependence

between tasks through formula 3, mark the task graph DAG. Compute the degree of dependence of external task on the spread domain boundary; add extra-territorial tasks with its degree of dependence more than threshold and dependent relation to the spread domain, until there is no dependent relation which is more than threshold. And according to rule 1, remove the tasks of average expected finishing time (compute method is shown in formula 4) out of the average change cycle of resources environment from spread domain;

5. Connective spread domain computing. Add the all paths between region nodes which have dependent relation to the spread domain. For the nodes we get whose in-degree is 0 in spread domain, add pseudo-entrance task nodes to the head of it. Assuming that s_i is a node with its in-degree is 0, then the weight c_i of pseudo-edge between pseudo-task node s_{entry} and task node with its original in-degree is 0, is the maximum weight of s_i which is in graph DAG and direct precursor original node(that means suppose that the resource of pseudo-entrance nodes is assigned as the resource which is assigned by direct precursor original node corresponded with its weight), and supposed that the pseudo-edge weight is 0, which is connected with pseudo-exit node s_{exit}.

The computing process of rescheduling tasks spread domain is shown as follows:

In the example, according to the results of performance prediction in rescheduling task s_5, the optimal resource ar$_1$ in performance can be achieved, which assigned for it are the resources s_4 and s_9, and define the dependent relation as powerful dependent relation, whose dependence degree is more than 0.8.

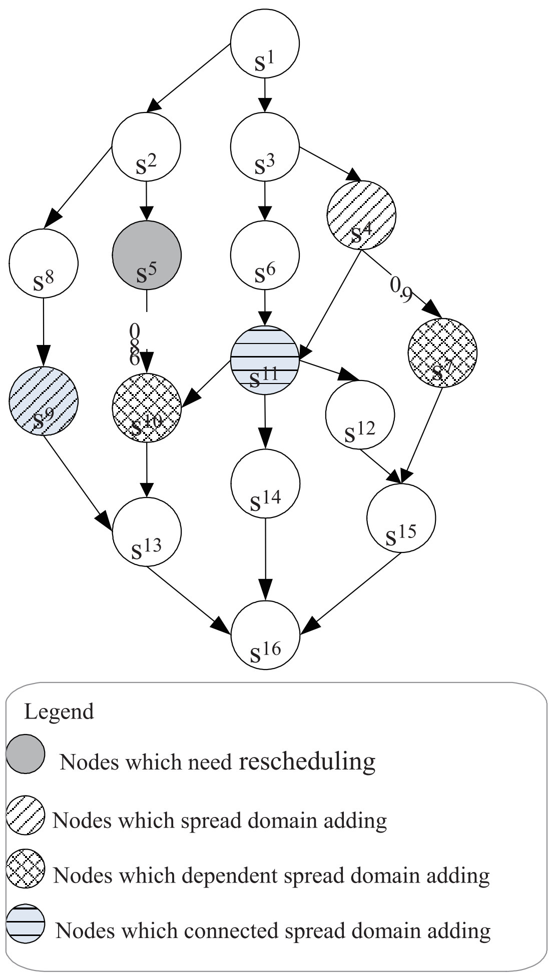
s1
s2
s3
s4
s5
s6
s7
s8
s9
s10
s11
s12
s13
s14
s15
s16
0.9
0.8
Legend
Nodes which need rescheduling
Nodes which spread domain adding
Nodes which dependent spread domain adding
Nodes which connected spread domain adding

Fig. 2 The computing process schematic of rescheduling tasks spread domain

3.2.1 Computation method of nodes in spread domain

In the size of resources under certain circumstances, we should not delay application completion time on it as far as possible, which is mean increasing parallelism on tasks. In addition, in terms of requirements on the performance of the replaced resources, sort as completing time of tasks which need rescheduling and the successor tasks on each resource, looking for better resource; On the other hand, in order to improve the parallelism, we should consider the internal time on between tasks on the allocation of resources, the possibility on solving this problem is bigger in the bigger internal time. So this paper adds the tasks on two kinds of resources to the spread domain. According to this, the computation method of point spread domain is shown as follows.

In the computation method of point spread domain, we need to compute the completion time of tasks which need rescheduling on each resource currently. Assuming that si is a task which need rescheduling currently, the computation method of expected completing time on resource mj as follows

$$s_i.EstiET = \begin{cases} NOW + \frac{w_{ij}\times m_j\cdot p}{m_j\cdot p_{now}}, s_i\cdot EstiET \prec t_2 \\ t_2 + \frac{w_{ij}\times m_j\cdot p - m_j\cdot p_{now}(t_2-NOW)}{m_j\cdot \overline{p}}, other \end{cases} \quad (1)$$

Among it, t_2 shows the changing time of next resource, which resource m_j get on the basis of its average change cycle of resources environment, if it can be done before t_2, compute only with the current performance of resource, instead, if more than t_2, with the average performance computing on resource.

In addition, the tasks time of a resource need to be computed in the computation method, to arrange for free time. Assuming that s_{exit} is ending task nodes of application, the computation method of task time arranging internal on a resource as follows

$$SpareTime(m_j) = \sum_{i=2}^{n} (s_i.ST - s_{i-1}.ET) + (s_1.ST - NOW) + (s_{exit}.ET - s_n.ET) \quad (2)$$

Among it, $\{s_1, s_2, \ldots, s_n\}$ is the task collection which assigns on the resource mj.

Get the tasks which need rescheduling from its trigger case, combined with DAG-task-graph to generate the smallest spread domain, sort according to the expected finishing time of the tasks which need rescheduling and their successor tasks which conflict with them, on every resource (see the calculation method as formula 1). Add the waiting tasks to spread domain, which have the earliest finishing time on resource and the starting executing time in the average change cycle of resources environment; and then based on the scheduling plan, calculate the tasks time interval on each resource, add the tasks to spread domain, which have the largest time interval on resource and have finished in the average change cycle resources environment.

3.2.2 Dependent Spread Domain calculation methods

For improving the dependent tasks' performance, it's necessary to take the dependent between tasks in the process of rescheduling into account, as there is re-allocation of tasks in a non-global scope of tasks to consider, so the existence of the dependent relations between boundary nodes within the scope and the external. The greater dependent, the larger demand needs to consider on the distribution of tasks at the same time. To a large extent, performance optimization depends on the processing of the dependent between tasks. Therefore, from this point of view, this paper will add extra-territorial tasks which satisfy certain threshold and the dependent relations between tasks on the boundary of the spread domain into spread domain.

Based on the analysis of the dependent relations between tasks, this paper gives the quantitative methods of task dependence degree.

In the DAG-based model of the grid application, the task has multiple inputs and multiple outputs. If multiple inputs trigger the task at the same time, after the execution of tasks they will transfer data to the successor task at the same time, during the execution period they will not accept external parameters or export any parameters neither, so the dependent relations between tasks totally embodied in the data transmission between tasks. Under the premise of using the available resources which generally support all the tasks, the task dependence degree can be measured by the amount of transmission between tasks.

Definition 2 Task Dependence Degree Mapping all the value intervals of data transmission amount between tasks to [0,1] interval, then get the score of data transmission amount, this paper define the data transmission amount between tasks as task dependence degree, to express the intensity of dependent relations between tasks. The computation method of task dependence degree as follows:

$$TaskDependenceDegree(s_i, s_j) = \frac{c_{ij} - \operatorname{Min}_{c_{ij}\in \mathbf{C}}(c_{ij})}{\operatorname{Max}_{cr_{ij}\in \mathbf{C}}(c_{ij}) - \operatorname{Min}_{cr_{ij}\in \mathbf{C}}(c_{ij})} \quad (3)$$

At the determination of initial range of task allocation, we will take the task which exceeds the threshold as a strong dependent task by setting the threshold. In dependent spread domain algorithms, it needs to gradually calculate the external dependent degree of boundary tasks in spread domain, to add the external tasks which have strong dependent relation to spread domain until there is not dependent relation which exceed threshold.

First of all, we compute the task dependence degree, and mark DAG-task-graph. In view of all the tasks, which have been calculated within the point spread domain and have dependent relations with the external domain tasks, to compare task dependence degree with the setting threshold: $0 \leqslant \lambda \leqslant 1$ one by one, then add the tasks which are bigger than the threshold and the task dependent relations to the spread domain until there is no bigger dependent relations exceed the threshold.

The basic calculation steps as follows:

(1) Find the task in rescheduling set, which has data transmission relation with the external, and form a boundary task set;

(2) Compare the threshold with the dependence degree of each task in boundary set and external tasks, add the task which is bigger than threshold to the task set needs rescheduling, and determine whether it is the boundary task, if it is, add it to the boundary task set too;

(3) Repeat step 2 until all the boundary tasks have been traversal.

3.2.3 Connected spread domain calculation method

The dependent relations between tasks that within spread domain, which we get through the above two spread processes have not totally embodied. The task re-distribution result by these spread processes will lead to time arrangement deadlock. Figure 3 is a local task graph, task s_1, s_3, s_6 will in spread domain after dependent spread calculation. If rescheduling begin at this time, do not guarantee the distribution time of s_2 is not diverted for other uses, then, after rescheduling s_2 maybe need to postpone backwards in the adjusting process of scheduling plan, and the following task s_5 is the predecessor task of s_6, whose time arrangement needs to remain unchanged, then the deadlock will occur.

Therefore, if there is predecessor and successor relation between two tasks in domain, then all paths

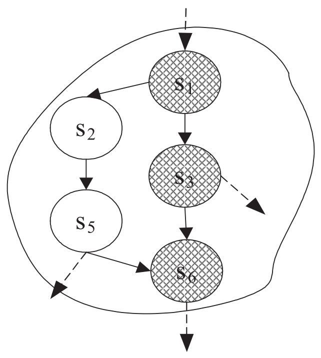

Fig. 3 The deadlock graph of task

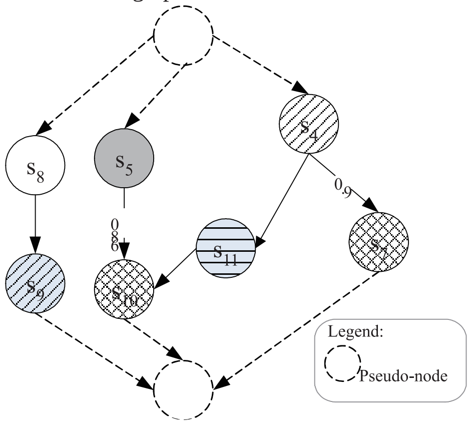

Fig. 4 The calculating results of rescheduling arrangement time tasks spread domain

between two task nodes should be added to the spread domain, including all the tasks and all dependent relations on the path. In the above graph, s_2 and s_5 will not add to the spread domain. And the spread domain after above mentioned treatment maybe still unconnected, but just multiplies of connected sub-graphs, and do not exist dependent relation between sub-graphs. Therefore, in order to establish connected spread domain, it needs to add pseudo- entrance nodes before the entrance nodes of each sub-graph, and add pseudo- exit nodes before the exit nodes of each sub-graph, then achieve the connectivity. And by setting reasonable nodes and pseudo-edge weight, it realizes the assumptions of the DAG task graph, and it also prepares for the following re-allocation algorithm. Also adds new tasks into this process, so this process is a spread process.

In this paper, we add all paths which have dependent between nodes in the domain to the spread domain. For the nodes whose in-degree is 0 in spread domain, it needs to add pseudo- entrance nodes whose executing cost is 0 before them. Suppose the node s_i's in-degree is 0, then the pseudo-edge weight c_i is the weight summation of s_i's original the direct precursor node in DAG graph, which is between the pseudo-node s_{start} and the node whose original in-degree is 0. While the setting principles of pseudo-exit node's weight and the pseudo-edge weight is as the same.

The example of rescheduling tasks spread domain calculating results is shown in Figure 4.

3.2.4 The spread domain limitation method based on dynamic measurement of resource

The grid resources are dynamic, for the grid application, the resource may be change again and trigger rescheduling in the executing process which we get the global scheduling plan through a rescheduling, that is, the previous rescheduling influence on the optimization scheduling results of the current unfinished tasks will failure, it needs rescheduling to update the optimal results. Thus, this rescheduling towards such tasks did not support the performance optimization of grid application, that is, we can ignore the rescheduling to such tasks in the previous rescheduling process, and this narrowing task domain method will not affect the grid application performance optimization. While in a rescheduling decision to identify and filter these tasks needs to capture the dynamic change discipline of resources. Therefore, dynamic measures for grid resource are needed, to reasonably narrow the scope of the rescheduling, to improve the efficiency of rescheduling while that does not affect performance optimization of grid application at the same time.

Above we described the spread process of spread domain, in the point spread process and dependent spread process, we make limitation to the spread process by taking the average change cycle of resources environment as constraint condition, and the task based on the current performance of resources, the average performance of resources, and the average change cycle of resources, when it is in the calculation execution time of resource. However, in the third spread process, dynamic is not be considered, because the nodes we added into this spread process or between paths within domain, while the end node on path has been in the average change cycle of resources environment, the nodes on path will certainly satisfy it, and as the added pseudo-nodes just realize the convenience of scheduling algorithm, therefore they are not taken into account. To calculate the executing cost with the current performance of resources, the average performance of resources, and the average change cycle of resources will improve the reliability of the expectation, The usage of the average change cycle of resources environment limits the scope of the affected domain and improves efficiency by a certain dynamic conditions and a reasonable way to meet the performance.

In the calculation process of the point spread domain and the dependence spread domain, it needs to calculate the average expected finishing time of a task, its calculating method as follows:

$$s_i.\overline{EstiET} = s_i.ST + \frac{\sum_{j=1}^{k}(w_{ij} \times m_j.p)}{k \times \overline{\overline{p}}} \quad (4)$$

Among it, $\overline{\overline{p}}$ is the average performance of resources environment, that is, take this task's beginning time of scheduling plan as the starting executing time, and calculate the average expected finishing time on the application of available resources set in according to the task's average executing time on all the available resources.

IF($s_i.\overline{EstiET} > NOW + \overline{\overline{T_f}}$) THEN shouldBeRemoved ($s_i$) $==$ TURE

[Rules 1]The limitation rules of spread domain based on the dynamic measurement of the resources environment

IF($s_i.\overline{EstiET} > NOW + \overline{\overline{T_f}}$) THEN shouldBeRemoved ($s_i$) $==$ TURE

That is, if a task' s average expected finishing time is larger than the average change cycle of resources environment, then the task should be removed from the spread domain.

4 Experimentation Analysis

In order to validate the efficiency of proposed rescheduling tasks spread domain, validation should be done with the compared result of rescheduling, so this paper achieve the solution of rescheduling problem based on the HEFT (Heterogeneous Earliest-Finish-Time) algorithm.

4.1 The solution of rescheduling problem on the basis of HEFT algorithm

HEFT algorithm is proposed by Haluk, it is a classic static scheduling heuristic algorithm under the heterogeneous environment. The basic idea of the algorithm is based on the calculation of tasks and relationship of tasks constraints, which compute the weight of the task node and generate a list, and then repeats the following two steps until a feasible scheduling program is achieved: (1) Select a highest priority node from the list; (2) Select an appropriate resources for the node. Before the start of the scheduling process, first calculate the weights of each node, in the scheduling process to select the highest priority node for scheduling, in the second step, select the resources which the task can execute earliest for the task. Grid project ASKALON [8] use HEFT algorithm, and prove greater than the performance of the HEFT algorithm [9]. In order to put it into the runtime rescheduling, this paper has improved the HEFT algorithm, named IHEFT algorithm.

4.1.1 Basic thinking of algorithm

First, the formalization expression of HEFT algorithm that stems from the traditional parallel distributed environment. In order to fit in with expression of grid environment and keep consistent with the concept definition of former rescheduling model, this paper change a part of variable symbols of the HEFT algorithm, in addition as a result of dealing with the different problem, some function names of the original algorithm easily lead to be misunderstand, so they have also been made to adjust, in addition add the part of the increased variables. The predecessor part of this paper will describe meaning of symbols when first use the new symbols.

Task priority definition is as same as HEFT in IHEFT algorithm and, HEFT algorithm is achieved from expanding heterogeneous systems on the basis of the heterogeneous scheduling program, at the same time, the algorithm takes into account the execution time of tasks and the communication time among tasks. Task priority has been recursively defined as:

$$\begin{cases} rank_u(s_i) = \overline{w_i} + \max\limits_{n_j \in succ(n_i)} (\overline{c_{ij}} + rank_u(s_j)) \\ rank_u(s_{exit}) = \overline{w_{exit}} \end{cases} \quad (5)$$

Among it, s shows task, succ(s_i) shows that s_i direct successor set of task, c_{ij} shows that the transmission cost between resource fr_m and resource fr_n assigned by task s_i and task s_k, $c_{ij} = data_{ij}/cr_{mn}$. $\overline{c_{ij}}$ shows that the average communication cost between s_i and s_j for tasks, $\overline{w_i}$ is s_i average computing cost.

In this paper, the priority of resources continues to definite the earliest completion times of task, but its computing method improves as follows:

$$EFT(s_i, fr_j, NOW) = w_{ij} + EST(s_i, fr_j, NOW) \quad (6)$$

Compared with the original formula, computing method of the earliest start time EST function of task changes as follows:

$$\begin{cases} EST(s_i, fr_j, NOW) = \max\left\{avail[j], \right. \\ \qquad \left. \max\limits_{s_m \in pred(s_i)} (EDA(s_m, s_i, fr_j, NOW))\right\} \\ EST(s_{entry}, fr_j) = 0 \end{cases} \quad (7)$$

$$EDA(s_m, s_i, fr_j, NOW) = \begin{cases} PFT(s_m) + c_{mi}, & \text{case1} \\ AFT(s_m), & \text{case2} \\ \max(NOW, AFT(s_m) + c_{mi}), & \text{case3} \\ NOW + c_{mi}, & \text{case4} \end{cases} \quad (8)$$

Among it, $PFT(s_i)$ shows the time expected to complete after the task allocates resources, $AFT(s_i)$ shows the real completion time after implementation of the task, *EDA* (s_m, s_i, fr_j, NOW) show the time that at NOW moment the data of s_m earliest transmit to the resources fr_j which s_i locate.

Case1: When s_m is not completed;

Case2: When s_m is completed, and the resources that s_m is assigned is fr_j;

Case3: When s_m is completed,and the resources that s_m is assigned is not fr_j, however, transfer data that has been designated to fr_j;

Case4: When s_m is completed, and the resources that s_m is assigned is not fr_j, and transfer data that has not been designated to fr_j;

Changes in the second max function, which is shown as PFT(sm)+cmi in original formula, that is expected completion time which the nodes of pioneer tasks gain after the resources is assigned, this is because the original algorithm deals with the one-time static scheduling problems before the implementation, thus in the entire scheduling process, tasks have not been implemented.

4.1.2 The description of algorithm

The solving algorithm of improving HEFT rescheduling as follows:

Algorithm one: Improving solving algorithm of HEFT rescheduling

Input: the set of unfinished tasks is TaskSet in DAG, the set of applied available resources FR, the price matrix CC that task executed, the network bandwidth matrix CR of the available resources of application, NOW present moment

Output: New scheduling scheme NewSchedule;

```
Algorithm Improved HEFT(TaskSet, FR,CC,CR,NOW)
{
  using the formula 5 calculate rank for all task in TaskSet;
  Rank the tasks as the non-increasing sequence of ranku
  while there is the non-scheduling task in the an ordered list of task
    Select the first task of the ordered list;
    for each resource fr[k] in FR do
    using the formula 6 calculate EFT(s[i], fr[k], CC,CR, NOW)
    if(EFT(s[i],fr[j]) is the minimum){
     assign job s[i] to the resource fr[j]
     // Record the resources allocation of tasks and the results of time arrangement to new task scheduling program
     NewSchedule <-{s[i],fr[j],EST(s[i],fr[j]),EFT(s[i],fr[j])}
     }
  endwhile
  return NewSchedule;//Return to a new scheduling scheme
}
```

4.2 Experimentation Design

This paper adopts the simulative experimentation method proposed by literature [7], according to characteristic parameters, generated test cases randomly, including two parts of the task graph and resources environment. Furthermore, the method is improved further, on simulation of resources environment, to embody the dynamic nature.

In order to reduce the complexity of experiment and stress the core issues, the experiment assumed that in a reasonable scope: First of all, it is assumed that the prediction of performance is accurate; ignore the network delay of resources; don't consider the sharing conflict with other applications, reckon the available time quantum of resources in the interval $[0, \infty)$ It is assumed that the task graph has the single entrance and exit nodes.

The required feature parameters which are randomly generated by DAG task graph and the resources environment include:

v: The number of tasks in the task graph.

α: The parallelism factor.(the parameters which is controlled by task graph shape)Height of task graph (H) is randomly generated by the normal distribution whose mean is $\sqrt{v}/\alpha$, and integer H. the number of task on each level In the graph is randomly generated by the normal distribution whose mean is $\alpha \times \sqrt{v}$ and integer value. The larger α is, the higher parallelism is.

In order to ensure that the graph generated has one entrance and one exit node, the disposal of dealing with graph is shown as follows: If the number of nodes of which the in-degree is 0 is more than one, add a node to connect with all the nodes whose in-degree is 0, the node is recorded as the start node; If the number of nodes whose the out-degree is 0 is more than one, then add a node to connect with all the out-degree nodes whose out-degree is 0, the node is recorded as the end node.

Out-degree: The maximum value of out-boundary of the task node shows as a percentage of v. The out-boundary of the node is randomly generated in the interval [1, out-degree].

CCR: The ratio of the average communication cost to the average computing cost in the task graph indicates that the application is a data-intensive parameter or compute-intensive one.

β: The heterogeneous factor of resource capacity. The values of β change in the interval [0, 2). The higher value is, the greater the difference of disposing ability of resources is. $\beta = 0$ shows that resources isomorphism. $\overline{w_{DAG}}$ is the average computing cost of all tasks in the DAG. The average computing cost $\overline{w_i}$ of one task called s_i is randomly generated in the interval $[0, 2 \times \overline{w_{DAG}}]$. The average computing cost of task w_{ij} in the resource fr_j is chose in the interval $[\overline{w_i} \times (1-\beta/2), \overline{w_i} \times (1+\beta/2)]$ randomly.

γ: Resources dynamic factor. It is expressed by the times of average task computing cost $\overline{w_{DAG}}$, the time interval of resources changes is randomly selected in $[0, \gamma]$.

δ: Network performance isomer factor. The difference of network bandwidth embodied the changes scope of resource connecting with bandwidth in the grid, in this paper, the value of network transmition bandwidth between resources is any random numbers in the interval

$[\overline{B}/\delta, \overline{B}\delta]$. $\overline{B}$ is the average network bandwidth of the available resources of application. In the DAG the average data transmission quantity between tasks is $\overline{data_{DAG}} = \text{CCR} \times \overline{w_{DAG}} \times \overline{B}$. The transmission quantity between tasks data_{ij} randomly generated in the interval $[0, 2 \times \overline{data_{DAG}}]$.

m: The amount of available resources in the initial application.

INR: the proportion of events which join available resources to all the events. This paper studied join and exit event of available resources.

In the simulation experiment of this paper, the parameters are set in table 1 as follows.

Table 1 task graph, the parameter table of resource randomly generate

parameter	value
v	10,20,50
α	0.5,1,2
out_degree	0.1, 0.5,1
CCR	0.1,0.5,1,2,10
β	0.1,0.5,1,1.5
δ	2,3,5,8,10
γ	100,50,20,10,5
m	10,20,50
INR	0,0.2,0.5,0.8,1

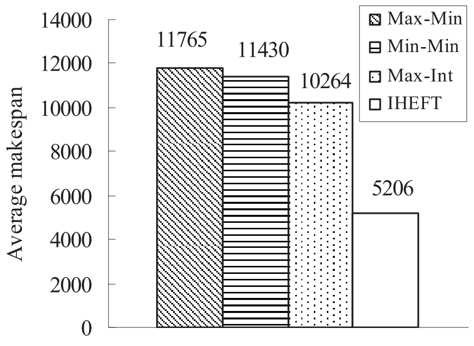

Fig. 5 The performance comparison of IHEFT algorithm and classical dynamic scheduling algorithm

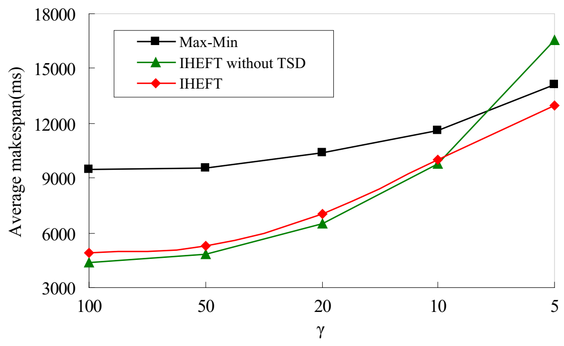

Fig. 6 The performance comparison of Max-Min and IHEFT under different dynamic without supporting of tasks spread domain

4.3 Experimentation results

In order to validate the supporting role of rescheduling tasks spread domain to the static scheduling strategy, this paper compare IHEFT algorithm with the classic dynamic scheduling algorithm, including Max-Min, Min-Min, Max-Int [10], the experimentation results shown in figure 5:

The experiment results shown, in the supporting of rescheduling tasks spread domain, the performance advantages of the static scheduling strategy is still very obvious, in the whole the efficiency of gird dependent tasks rescheduling mechanism of resource-based dynamic organizations is validated.

At the same time, we did the comparative experiment of performance optimization and efficiency in different dynamic, compared IHEFT (IHEFT) algorithm based on tasks spread domain, IHEFT (IHEFT without TSD) algorithm based on all the unfinished tasks with the dynamic the Max-Min algorithm.

Experimentation shown that under the low dynamic IHEFT algorithm of based on spread domain, the performance optimization effect of the application close to the IHEFT algorithm, under the strong dynamic, gradually close to the Max-Min algorithm, however under the strong dynamic the IHEFT without TSD algorithm performance decreased obviously, experimentation shown that spread domain proposed adapt to the variety of dynamic resources, under the low dynamic, the loss of performance optimization is very small, and under the strong dynamic, close to the degree of dynamic scheduling. Thus avoid the performance of global optimization under dependent rescheduling task and the strong dynamic decline obviously.

The comparison of three algorithms in the aspect of efficiency as follows:

As the number of tasks grew, the efficiency of IHEFT algorithm between the efficiency of IHEFT without TSD algorithm and Max-Min algorithm, and IHEFT algorithm still has an approximate linear relation with the amount of tasks, however, compared with IHEFT without TSD, it has obviously declined.

And, the efficiency of IHEFT increased gradually as the dynamic enhanced, close to the efficiency of Max-Min, however in the weak dynamic, due to spread computing of spread domain, the spread domain is not contained all the unfinished tasks, therefore, the efficiency is still higher

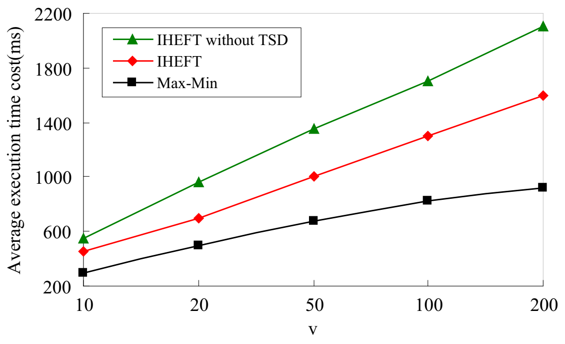

Fig. 7 The efficiency comparison of IHEFT, IHEFT without supporting of tasks spread domain and Max-Min under different applications scale

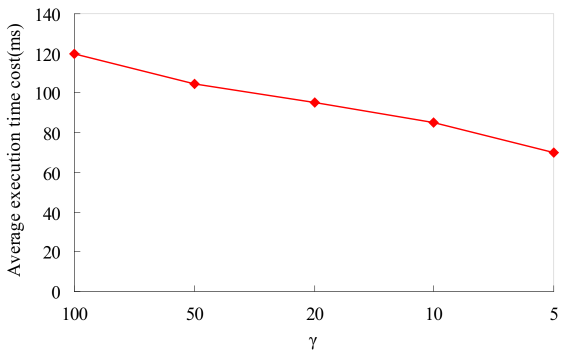

Fig. 8 The relation of the efficiency and dynamic of IHEFT algorithm

than IHEFT without TSD. Thus the efficiency of proposed rescheduling tasks spread domain is validated.

5 Conclusion

In order to resolve how to determine the tasks scope of the grid dependent tasks rescheduling, which should improve the efficiency of rescheduling problem as far as possible without affecting the application performance, this article proposed rescheduling tasks spread domain concept and computation method, which initial scope is the set of rescheduling tasks as minimum required, according to the degree of data transmission dependent between tasks, the degree of resource sharing conflicts between tasks, the degree of task support to performance optimization in dynamic environment, gradually expand and determine the scope of task, makes strong influence each other within the scope, weak influence each other without the scope. Experimentation results show that rescheduling tasks spread domain makes static scheduling strategy maintain the performance advantages compare with the dynamic strategy, improve the rescheduling efficiency without affecting the gird application optimization as much as possible, thus the efficiency of proposed rescheduling tasks spread domain and computation method is validated.

Acknowledgements

This work is supported by National Natural Science Foundation of China(No.60903008) and General scientific research project of the Education Department of Liaoning Province (No.L2011004)

References

[1] Yu H, Marinescu D C, and et al. Plan switching: an approach to plan execution in changing environments [A]. Proceedings of the 2006 International Parallel and Distributed Processing Symposium[C], 33–41 (2006).

[2] Lee Y C and Zomaya A Y. Practical Scheduling of Bag-of-Tasks Applications on Grids with Dynamic Resilience, IEEE TRANSACTIONS ON COMPUTERS, **56**, 815–825 (2007).

[3] Sakellariou R and Zhao H. A low-cost rescheduling policy for efficient mapping of workflows on grid systems[J]. Scientific Programming, **12**, 253–262 (2004).

[4] Imamagic E, Radic B, Dobrenic D. An approach to grid scheduling by using condor-G matchmaking mechanism. Information Technology Interfaces [A]. Proceedings of the 28th International Conference[C], 625–632 (2006).

[5] Yu Zhinfeng and Shi Weisong. An Adaptive Rescheduling Strategy for Grid Workflow Applications Parallel and Distributed Processing Symposium [A]. Proceedings of the 2007 International Parallel and Distributed Processing Symposium[C], 1–8 (2007).

[6] Ullman J. NP-Complete Schedulling Problems [J]. Journal of Computer and System Sciences, **10**, 384–394 (1975).

[7] Topcuoglu H, Harir S, and Wu M Y. Performance-effective and low-complexity task scheduling for heterogeneous computing [J]. IEEE Trans. on Parallel and Distributed Systems, **13**, 260–274 (2002).

[8] Fahringer T, Jugravu A, et al. ASKALON: a tool set for cluster and Grid computing [J]. Concurrency and Computation: Practice and Experience, **17**, 143–169 (2005).

[9] Wieczorek M, Prodan R, and Fahringer T. Scheduling of scientific workflows in the askalon grid environment [J]. SIGMOD, **34**, 56–62 (2005).

[10] Prodan and Fahringer T. Dynamic scheduling of scientific workflow applications on the grid: a case study [A]. Proceedings of the 2005 ACM symposium on Applied computing[C], 687–694 (2005).

Tingwei Chen received the PhD degree in computer science from Northeastern University in 2007. He is currently an associate professor in Liaoning University. His research interests are in the area of Distributed computing, Cloud computing and Big data.

Hongning Zhu received the PhD degree in computer science from Northeastern University in 2009. He is currently an associate professor in Liaoning University. His research interests are in the area of Services computing and Intelligent systems engineering.

Yu Dai received the PhD degree in computer science from Northeastern University in 2008. She is currently an associate professor in Northeastern University. Her research interests are in the area of Cloud computing and Performance management.

Xianwen Hao received the PhD degree in computer science from Northeastern University in 2008. He is currently an engineer in China Mobile Group Liaoning Company Limited. His research interests are in the area of Cloud computing, Artificial intelligence and Big data.