Task Scheduling Algorithm in Grid Environment Based on Duplication and Insertion

Lijun Cao
Hebei Normal University of Science & Technology
Email: misscao6666@163.com

Xiyin Liu, Torkel Hans-Georg, Zhongping Zhang
Hebei Normal University of Science & Technology,China
Vocational College of Technology and computer science (BTI), Hammfelddamm 2, 41460 Neuss,Germany
College of Information Science and Engineering Yanshan University, China
Email: liuxiyin2003@sina.com,torkel.hansgeorg@googlemail.com, zpzhang@ysu.edu.cn

***Abstract*—Grid resource scheduling theory involving basic theoretical knowledge for grid scheduling was discussed in this paper. Then an intensive study of the scheduling strategy was made. According to the heterogeneous characteristic of grid environment, an improved algorithm DIBS for task scheduling were proposed. In this algorithm, the entire scheduling process was divided into three steps: layering, task priority, and task replication. In the layering stage, according to the characteristics of the DAG, the simultaneous distribution strategy for the multiple DAG images was adopted. In the task priority stage, an improved decision path strategy was proposed. In the replication stage, the previous key path nodes were replaced by the best precursor replication nodes. The effectiveness of this algorithm was verified by Gantt chart. In this paper, the relevant scheduling algorithm simulation was successfully realized by using the basic framework and functions provided by SimGrid and combining with the proposed scheduling algorithm. The availability, validity and stability of the DIBS scheduling algorithm were verified by comparison and analysis of simulation results.**

***Index Terms*—Task scheduling; scheduling algorithm, DIBS, SimGrid**

I. INTRODUCTION

The task scheduling is an essential component of the high-performance computing. However, with the emergence of grid computing, the task scheduling also faces new challenges. As the grid environment is composed of a large number of heterogeneous resources, many problems for grid system caused by the different structures and categories of the resources must be resolved, such as communication between resources, rational resource allocation, and efficient task scheduling. It is the resources heterogeneity in grid environment that places higher demand on the design of grid software. These problems should be well solved in order to give full play to the role of the grid technology.

Manuscript received January 1, 2013; revised May 1, 2013
Accepted June 1, 2013

The task scheduling issues in grid environment could be divided into "grid resource oriented scheduling" and "grid application oriented scheduling". The grid resource oriented scheduling is proposed from the provider's perspective, whose principle is to maximize the efficiency of resources application in the grid environment. The grid application oriented scheduling is proposed from the grid user's perspective, whose principle is to make the application execution best adapt the dynamic nature of grid resources performance, and thus to ensure the requirements of the application performance.

In general, the grid resources are provided by a number of resource providers, and each provider has unique scheduling strategy and mechanism for local resources. This "local resource scheduling" is the specific implementation for each grid resource oriented scheduling. On the other hand, the grid application oriented scheduling should be realized in collaboration with the local resource scheduler. The grid resource oriented scheduler usually does not provide the necessary interfaces for the grid application oriented scheduling to help implement an application oriented scheduling process, while the grid application oriented scheduler often has no authority to intervene the operating mode of the local grid resource scheduler, which makes the implementation of application oriented scheduling become very difficult. At present, some corresponding mechanisms have been proposed to address these problems, such as the resource reservation mechanism, dynamic resource performance prediction mechanism, resource scheduling information disclosure mechanisms, synchronous resource rationing mechanism, calculating migration mechanism, etc. However, the research and implementation of these mechanisms are still far from meeting requirements of the actual application. Therefore, a good task scheduling strategy will greatly improve the utilization of grid resources and further promote the development of grid technology.

The task scheduling based on task duplication is a new method that has been proved to have great performance. By far some existing task duplication-based scheduling algorithms have managed to generate optimal scheduling

doi:10.4304/jsw.8.10.2447-2454

when the tasks fulfill certain conditions, but they still show some inadequacies.

This paper made some developments and innovations in the following aspects:

(1)In this study, noticing that in grid environment, the target systems of task scheduling are usually randomly linked network, and the applications are mostly intensive parallel distributed applications, we proposed a new heuristic task scheduling algorithm based on task duplication and insertion – Duplication and Insertion Based Scheduling (DIBS). The proposed algorithm could reduce the time of repeatedly searching for the most suitable processor during the execution of tasks, meanwhile it allows simultaneous execution of multiple applications, thus shortening the total execution time of applications and balancing the load of processors..

(2) To further verify the effectiveness of the algorithm, simulation model tests of resource discovery MLON-RSA algorithm were carried out by the use of GridSim simulation toolkit and simulation analysis of DIBS algorithm were done by using SimGrid toolkit. The simulations of LHCNF algorithm and DIBS scheduling algorithm were achieved respectively in the tests. The analysis to the obtained data and the comparison with related algorithms proved the feasibility, efficiency and stability of DIBS algorithm..

The central idea of task duplication- based scheduling algorithms is to redundantly map some of the tasks in the task plan to some processors in order to reduce the communication between processors. This means that by making use of the processors' idle time to duplicate predecessor tasks, the communication data between some predecessor tasks does not need to be transmitted, and thereby the waiting time of processors could be saved. Based on different strategies of selecting the tasks to be duplicated, different scheduling algorithms could form, some of which duplicate the immediate predecessor tasks only while some others duplicate every possible predecessor tasks. Comparing with other scheduling techniques, this type of algorithms have higher time complexity, and in most situations require unlimited number of processors, however they are most likely to produce the optimal solution.

Three stages are included in Levelized Heavily Communicating Node First (LHCNF) algorithm: hierarchization, task prioritization, and task duplication. In the first stage, the tasks of each level are independent, which means the tasks in the same level have no data dependence on each other and could be executed in parallel. Assume a task graph G=(T, E), then level 0 contains the entry node, and level i includes all tasks with task ID number t_j , all edges (t_k, t_j) , task t_k is on the upper level of level i, leve i-1, containing at least one edge (t_k, t_j) the last level contains the exit node. In the second stage, priorities are assigned to each Directed Acyclic Graph (DAG) task including the exit node according to their average communication time. Tasks with the longest communication time are assigned to the highest priority, and for tasks with the same communication time, they are distinguished by execution time; the priority of the exit node is calculated by the average execution time. In the third stage, the Critical Immediate Parent (CIP) nodes and the Critical Immediate Grand Parent (CIGP) nodes or only the CIP nodes of a node are duplicated in order to move up the EFT (Earliest Finish Time) of the task. If the EST of a task could not be moved up, no duplication should be performed.

For existing algorithms like TDS[1,2], STDS[3,4], LDBS[6], and LHCNF[5], they require the processors be fully linked between each other; however in practice processors are only randomly linked. Targeting at this shortcoming, this article proposes DIBS algorithm.

II. Task Duplication Based Scheduling

Task duplication is to execute the same task on different processors, where a task can be distributed to more than one processor at one time, so that the communication time between tasks could be reduced, and the start time and finish time of tasks could be moved up. While retaining the original parallelism of applications, task duplication can also help reduce the time cost of the communication between tasks. Thus, task duplication is a very effective way to eliminate communication cost between tasks.

The central idea of task duplication based scheduling algorithms is to redundantly mapping some of the tasks in the task plan to some processors in order to reduce the communication between processors. This means by making use of the processors' idle time to duplicate predecessor tasks, the communication data between some predecessor tasks does not need to be transmitted, and thereby the waiting time of processors could be saved. Based on different strategies of selecting the tasks to be duplicated, different scheduling algorithms could form, some of which duplicate the immediate predecessor tasks only while some others duplicate every possible predecessor tasks. Comparing with other scheduling techniques, this type of algorithms have higher time complexity, and in most situations require unlimited number of processors, however they are most likely to produce the optimal solution.

III. DIBS Algorithm

A Establishment of Target System

The heterogeneous system referred to in this article means processors with different computing capacities and processors with different communication capacities between each other. The target system of task scheduling is composed of Processor Elements (PEs) networks with certain degree of topology, and each PE consists of one processor and local memory. Target systems could be indicated by array P= (Vp, Ep) , where Vp is the collection of processors, Ep is the collection of the links between processors, and ep(pi, pj) is the link between processors pi and pj, as shown in Figure 1. In this article we made the following assumptions about the target system:

(1) Each processor could carry out computation and communication at the same time.

(2) If a task is transmitted through several links, its communication time is the sum of the communication time costs of each link. The communication time of link Ci,k could be represented by Formula (1).

$$C_{i,k} = \frac{D_{i,k}}{R[p(i), p(k)]} \quad (1)$$

where Di,k is the data volume transmitted from task ti to task tk, R([p(i), p(k)] is the bandwidth between processors p(i) and p(j).

(3) The topology of the target grid system is relatively stable in a period of time.

In grid environments, the links between processors are random, thus finding the path with the minimal communication time cost between two processors is crucially related to the efficiency of data transmission. Here we use Dijkstra's algorithm to find the minimum weight between any two nodes in an undirected graph, and create a list to store the nodes that the shortest path goes through.

The fundamental idea of Dijkstra's algorithm is to start from start point s and gradually search for the shortest path outward. During the execution, for each node, record the number (called the notation of this node) which is either the weight of the shortest path from s to this node (called notation P), or the upper bound of this weight (called notation T). Specifically, after each extension, change the node of notation T to the node of notation P, so that the vertices in directed graph D with notation P have one more member. Repeat these steps and the shortest path from s to each node could be obtained. The minimum communication between processors could be represented by the array showed in Figure 2. Listed in Table 1 are the nodes that the shortest path between two processors goes through, where "---" indicates that the two processors are directly reachable. The data in Table 1 could help to reduce the transmission time, and this table is updated regularly.

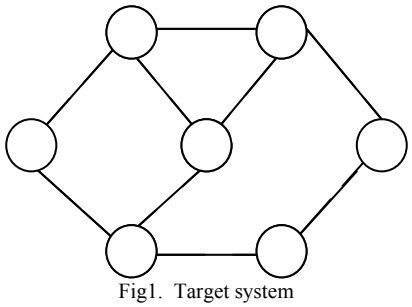

Fig1. Target system

$$\begin{pmatrix} 0 & 6 & 3 & 8 & 10 & 10 & 14 \\ 6 & 0 & 8 & 3 & 4 & 14 & 10 \\ 3 & 8 & 0 & 5 & 7 & 7 & 11 \\ 8 & 3 & 5 & 0 & 2 & 12 & 8 \\ 10 & 4 & 7 & 2 & 0 & 10 & 6 \\ 10 & 14 & 7 & 12 & 10 & 0 & 4 \\ 14 & 10 & 11 & 8 & 6 & 4 & 0 \end{pmatrix}$$

Fig2. Minimum communication time among processors

B Definitions

The model of the scheduling system consists of applications, target system, and performance indicators. Each application could be indicated by one DAG, which could be represented by array G, and G= (V, E, P, T, C), where V is the node collection, E is the edge collection, P is the processor collection, E(vi, vj) is the edge between nodes vi and vj, T(vi, pj) is the execution time on processor pj for task vi, and C(vi, vj) is the communication time of the data transmission between tasks vi and vj. When two tasks vi and vj are executed on the same processor, C(vi, vj)=0. We define pred(vi) as the collection of all predecessor nodes of task vi, and succ(vi) as the collection of all the successor nodes of it. Avail (pj) indicates the earliest available time of processor pj, est(vi, pj) indicates the earliest execution start time of task vi on processor pj, eft(vi, pj) indicates the earliest finish time of task vi on processor pj, EST(vi) and EFT(vi) indicate respectively the earliest start time and finish time of task vi on any processor, AFT(vi) indicates the actual finish time of task vi, and Time(pi) indicates the spare time between the finish time of the last task and the start time of the next task for processor pi, as presented by Formula (2).

$$Time(p_i) = EST(t_k, p_i) - Avail(p_i) \quad (2)$$

TABLE 1.

MINIMUM ACCESS AMONG PROCESSORS

	P_1	P_2	P_3	P_4	P_5	P_6	P_7
P_1	---	---	---	P_2	P_2	P_3	P_3,P_6
P_2	---	---	P_4	---	---	P_5,P_7	P_5
P_3	---	P_4	---	---	P_4	---	
P_4	P_2	---	---	---	---	P_3	P_5
P_5	P_2	---	P_4	---	---	P_7	---
P_6	P_3	P_5,P_7	---	P_3	P_7	---	---
P_7	P_3,P_6	P_5	P_6	P_5	---	---	---

Definition 1 (Critical Path) The critical path of an application means the longest path from the entry node (the node with no predecessor node) to the exit node (the node with no successor node); the length of this path is the sum of the weights of the nodes and edges on this path.

In this paper we calculate the critical path with the average execution costs of a task on all processors. A node's top distance is the longest distance from this node to the entry node, excluding the computing cost of the node. A node's bottom distance is the longest distance from this node to the exit node, including the computing cost of the node.

Definition 2 (Decisive Path) Decisive Path (DP) is the sum of the top distance and bottom distance of a node.

Each DAG node has its decisive distance, and the critical path is the maximum decisive distance of the exit node.

C DIBS Algorithm

In this article we propose a heuristic scheduling algorithm based on duplication and insertion. The core idea of the algorithm is to try to shorten the total execution time of tasks, move up the earliest finish time of tasks .The fundamental method is to put the entry node (the node with no predecessor nodes) of the DAGs of multiple applications into the ready queue SCH according to their priorities. Applications are given different priorities, while within the same application the nodes with higher DP values are assigned higher priorities because the nodes with higher DP values are more likely to be on the longest path. Nodes with higher priorities are then distributed to more appropriate processors in order to reduce the total execution time. If the duplication of the predecessor node of a node could move up the earliest finish time of the task, duplicate the predecessor task to the processor for execution, otherwise no duplication is needed. After the nodes of the SCH queue are distributed, if none of the predecessor nodes of this node's successor nodes are in the SCH queue, put the successor nodes of this node to the REA queue.

Algorithm 1 Task Scheduling Algorithm DIBS

Inputs： Directed Acyclic Graph sequence DAGs

Outputs： Task Scheduling Queue SCH

DIBS(DAGs)
Begin
Calculate the DP values for all nodes
Put the head nodes of all DAGs into the SCH queue by their priorities
While (SCH!=$\varnothing$)
Extract the first node vmn from SCH queue, and remove this node from the queue; // vmn indicates the n-th task node in the m-th DAG
For all pj$\in$ P do
pk=pj; eft(vmn, pj) ;
For vmi$\in$ npred(vmn) do
eft1=max{Avail[pj],AFT(vmn)+c(vmn, vmi)}
+T(vmn, pj) ;
If eft1==eft(vmn, pj) then if Time(pj)>Time(pk) then {U=vmi; pk=pj;}
If eft1<eft(vmn, pj) then {U=vmi; pk=pj; eft(vmn, pk)=eft1;}
Endfor
Endfor
If (U$\neq\varnothing$) then duplicate the tasks into processor pk for execution
Else no duplication is needed
endif
If (SCH!=$\varnothing$) then
For vmk $\in$ succ(vmn) do
If all pred(vmk)$\notin$ SCH then Put vmn into the REA queue
Endfor
Else Sort the tasks in REA queue by their priorities and put them into SCH queue
Endif
End while
End

D Analysis of Algorithm

Comparing with LHCNF algorithm, DIBS algorithm has the following improvements.

LHCNF algorithm determines the priority of a task based on its average communication time, however the tasks with long communication time are not necessarily on the critical path, and assigning higher priority to them does not guarantee the reduction the total execution time. The proposed DIBS algorithm determines the priorities of tasks based on their DP values, because the nodes with higher DP values are more likely to be on the longest path. By assigning tasks with higher priorities to the most appropriate processors, the total execution time is reduced.

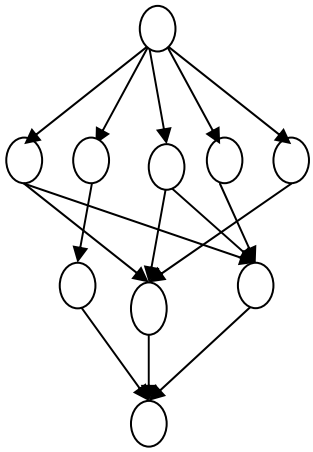

Fig 3. DAG G1 with average intertask communication times

(2) LHCNF algorithm duplicates the critical immediate parent nodes and critical immediate grand nodes or only the CIP of a task to move up the earliest finish time of the task. However, CIP is the immediate predecessor task of the task with the latest start time, as illustrated in Figure 4 and 5 which are generated from Figure 3 and Table 2, where CIP nodes are duplicated in Figure 4 and the most appropriate nodes in the predecessor nodes are duplicated in Figure 5, it can be seen that duplicating the critical immediate parent node and critical immediate grant nodes or only the CIP of the task does not necessarily move up the earliest finish time of the task. Algorithm 1 duplicates the most suitable

predecessor nodes to move up the earliest finish time of a task.

TABLE 2.

EXECUTION TIMES OF TASKS IN G1 ON THREE DIFFERENT PROCESSORS

Task	*P1*	*P2*	*P3*
1	14	16	9
2	13	19	18
3	11	13	19
4	13	8	17
5	12	13	10
6	13	16	9
7	7	15	11
8	5	11	14
9	18	12	20
10	21	7	16

(3) LHCN algorithm uses depth-first traversing in the hierarchization stage, and its time complexity is O(v+e). The proposed algorithm adopts the approach that assigns one node first, if the predecessor nodes of this node's successor nodes are not in the SCH queue, put the successor nodes of this node to the REA queue; only when the SCH queue is empty, assign the nodes in REA queue to it for distribution; and the time complexity is O(v).

(4) When several applications are executed at the same time, LHCNF algorithm can only execute each application in sequence, whereas the proposed algorithm could execute them in parallel.

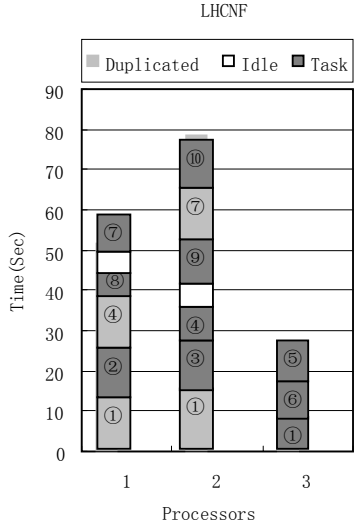

Fig 4. Gantt chart for G1 using LHCNF

(5) When the earliest finish time of a task on two or more processors are the same, this algorithm assigns the task to the processor with the longest Time(pj).

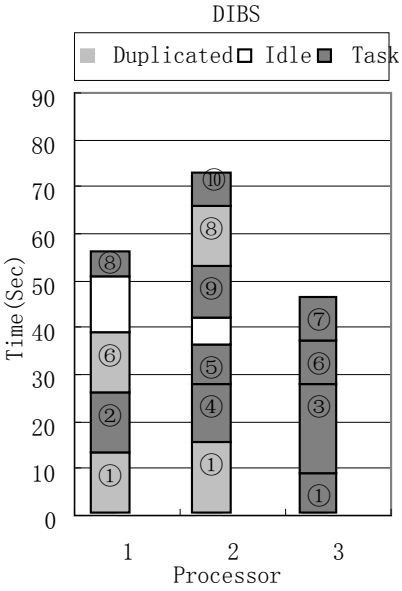

Fig 5. Gantt chart for G1 using DIBS

IV THE SIMULATION TEST BASED ON SIMGRID

A SimGrid Simulation Package and Its Structure

The role of grid simulator is to imitate a grid environment, where various issues can be studied, such as the feasibility and the performance of the algorithm. A well configuration parameter can contribute to a more real simulation environment and more reliable results. In addition, the algorithm can be constantly improved and optimized by analyzing the results of the test in the simulator.

In this paper,SimGrid toolkit is used for the simulation analysis of the algorithm proposed in chapter 3. SimGrid was dominantly developed by the Grid Research and Innovation Laboratory in the University of California San Diego and its goal is to provide an appropriate model for distributed parallel application under the grid environment, abstract and generate correct analog result. In fact, SimGrid simulation package is a simulation toolkit, a simulator providing a series of core functions to build specific computing environments and application fields. The distributed environment here can be a simple network composed of workstations, or a complicated grid environment made up of workstations, PC and other nodes. In SimGrid review, resource model comprises processor and network connection while task model includes computation tasks consuming processor resources, data transmission resources consuming network connection and execution order created by dependency between tasks. In addition, some functions like scheduling, predication, resource trace, time and simulation are provided. These API functions can rapidly build and evaluate the studied scheduling algorithm. As SimGrid execution is an event-driven simulation, whose

process only requires one host and whose results are given in the form of virtual time, the simulation results refrain from

the impact of the performance of the host. The most remarkable characteristic of SimGrid simulation package is that the simulation is available to the complex platform in consistent with the real situation and is rapid in simulating. In general, the simulation lasts about 6s to 10s.

B The Structure of SimGrid Simulation Package

SimGrid Simulation Package can be divided into three layers

(1) Programmation Environments Layer

SimGrid provides several programming environments in simulation kernel. MSG is more prone to build real multi-agent simulations. The main target of the environment is not the reality, but to build many real platforms. GRAS(Grid Reality And Simulation) contributes to the development of a real distributed application.

SimGrid simulation toolkit has two API, one on the top of SURF, which allows to develop and test your application program in an appropriate simulator; the other is suitable for real platform and is very efficient.

SimDag is to provide framework for DAGs parallel tasks. The chapter focuses on the simulation test of DIBS algorithm in SimDag programming environment.

(2) Simulation Kernel Layer

The role of Kernel Layer is to simulate a virtual platform by SURF model. It lies in the low layer, is rejected by terminal user and serves as the base for high-level users. The main feature of SURF is a resolver with maximum celerity and minimum linear, which can easily change the model to describe the platform. It is very easy to compare the models in the references.

(3) Base Layer

Base Layer is compose of XBT(eXtended Bundle of Tools) and provides some basic features like Logging support, Exception support and Configuration support.

C Sage of Simulation Package

SimGrid package usually runs on the Linux or Unix platform with gcc as the compiler. The simulation package can be achieved by static library and dynamic linking library. The format using static library is Gcc libSimgrid.a –o MainProgram MainProgram.c. Under such circumstances, all the SimGrid functions are included in MainProgram, thus resulting in a large binary file, while by using dynamic linking library, the format is Gcc –lsimgrid –o MainProgram MainProgram.c, where all SimGrid functions are not included in MainProgram directly. Libsimgrid.so can be found by setting the environment variable: export LD_LIBRARY_PATH=$HOME/lib/:$LD_LIBRARY_PA-TH. The latter method is more common.

In compiling simulated program, makefile should be corrected by setting the variable of top_srcdir into the catalog of installing SimGrid and the variable of top_builddir into the path to be compiled; if the program includes other libraries, the libraries should be added after the variable of LIBS in Makefile

V MULATION AND EVALUATION

A Simulation

In this study, SimGrid[7] toolkit is used for the simulation analysis of the proposed DIBS algorithm. The simulation program runs on Redhat Linux 9.0 platform with gcc as the compiler. The SimDag module of the SimGrid toolkit is the main module used in this study. The flow chart of the algorithm is presented in Figure 6.

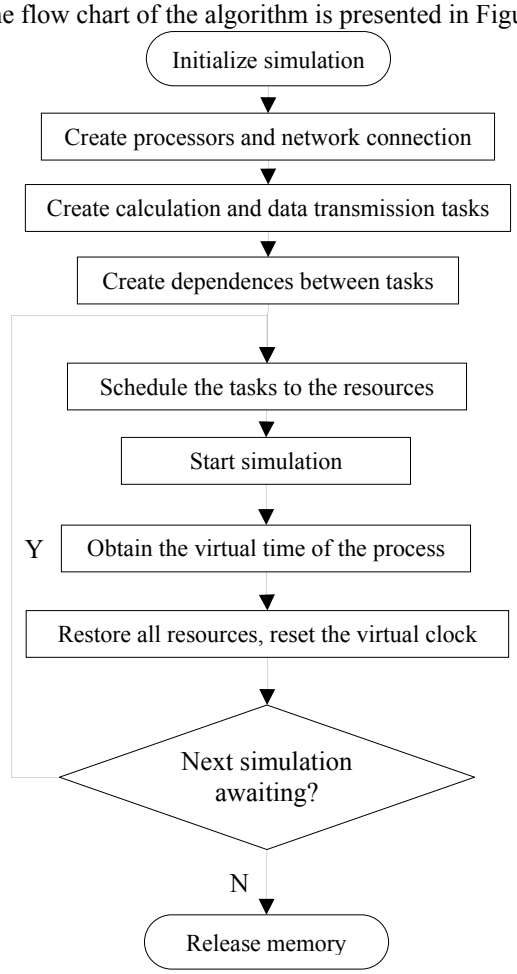

Fig. 6 Simulation process flow chart

Below are the explanations for the core codes of the program.

(1) SimGrid Initialization function SG_init(). This function must be called before calling other simulation functions of SimGrid.

(2) Processor creation function Host[i]=SG_new Host(buffer, 1.0, SG_SEQUENTIAL_I-N_ORDER, NULL, 0.0,1.0, NULL,NULL,0.0,NULL). This function creates the host resource. The first parameter is a string that defines the name of the processor, the second indicates that the relative processing speed of the host is 1, and the ninth parameter indicates that the possibility of host failure is 0.0.

(3) Network connection creation function SG_newLink(buffer, SG_TIME_SLICED, NULL, 0.0,

1.0, NULL, 0.0, 100.0,NULL). The first parameter of this function is the name of one of the connections.

(4) Computing task creation function SG_newTask(SG_COMPUTATION, buffer, cost, NULL). This function creates a commutating task, the second parameter is the name of the task to be created, and the third parameter sets the execution time of the task.

(5) Data transmission task creation function SG_newTask(SG_TRANSFER, buffer, 0.0, NULL). This function creates a data transmission task, the second parameter is the name of the task to be created, and the third parameter indicates the volume of the data that needs to be transmitted.

(6) Inter-task dependency setup function SG_addDependency(task[i], compution[j]). This function adds data dependency between two tasks.

(7) Task to resource scheduling function SG_scheduleTaskOnResource(task[i], host[i]). This function schedules a task task[i] to a host resource host[i].

(8) Simulation function SG_simulate(-1.0, SG_ALL_TASKS, SG_SOME). This function runs the simulation until all the tasks are completed.

(9) Simulation ending function SG_clear(). This function ends the simulation program and releases the memories occupied by the simulation tool.

B Aalysis to the Simulation Result

First, the algorithm was tested with different degrees of heterogeneity of the hosts, 100 times of simulation tests were performed with 200 randomly generated tasks and given execution time of each task used in each test, the average time cost of the tests are calculated. The results are as shown in Figure 7, 8 and 9.

Different scheduling algorithms have different prerequisites. the DIBS algorithm proposed in this paper focuses on multiple applications. As shown in Figure 7, when there is only one application in DIBS, the total execution time of DIBS when the number of tasks increase is longer than that of LHCNF algorithm. However, as shown in Figure 8 when there are multiple applications, the LNCNF algorithm executes them by repeatedly calling itself, thereby its total execution time when the number of task increases is significantly longer than that of DIBS. It can be seen from Figure 9 that the performance of DIBS algorithm is slightly better than LHCNF algorithm, where performance refers to the ratio of accelerated speed to the number of processors, and the accelerated speed is the ratio of the time cost of sequential execution to the time cost of parallel execution.

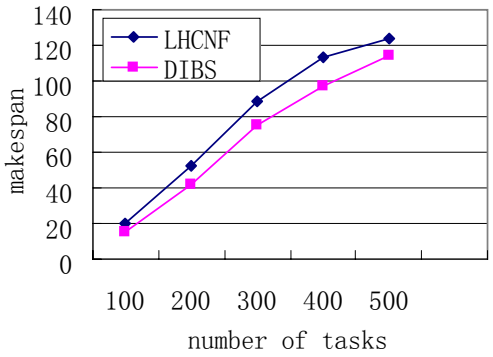

Fig. 7 The comparision in makespan between DIBS and LHCNF, When only an application is executed

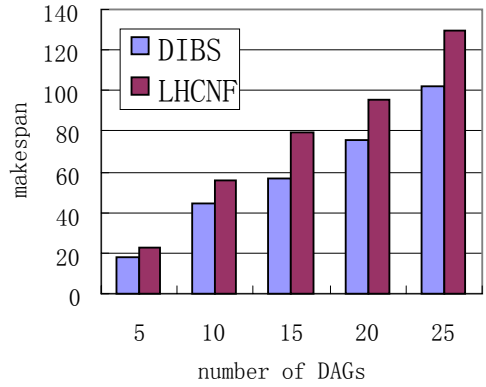

Fig.8 The comparision in makespan between DIBS and LHCNF, When only an application is execute

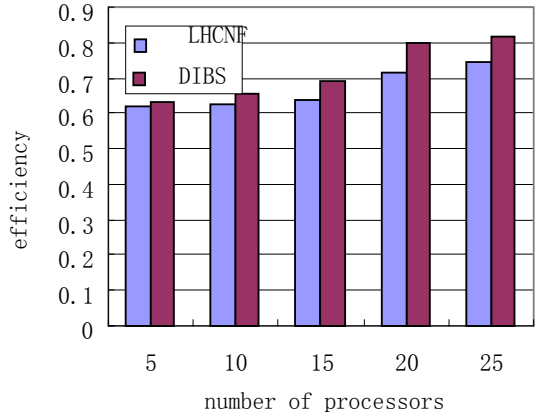

Fig. 9 The performance cooperation of between DIBS and LHCNF

Through the simulation of the LHCNF algorithm and DIBS algorithm, the analysis to the obtained data, and the comparison with related algorithms, the proposed DIBS algorithm is proved feasible, effective, and stable.

CONCLUSIONS

The goal of grid computing is to provide a convenient and effective way to make use of widely spread resources [8,9,10,11]. Scheduling strategy, as a means of the effective utilization of grid resources, is one of the core functions that the underlying software of grid systems ought to provide. In this study, discussion and in-depth research on the scheduling strategy are conducted, the DIBS algorithm is proposed. This algorithm determines the priorities of tasks based on the decisive paths of them, and duplicates the predecessor nodes that can move up

the earliest finish time of tasks, thereby shortening the total execution time.

ACKNOWLEDGEMENT

This work was financially supported by hall of Hebei province science and technology research (12270140),

REFERENCES

[1] Y. Kwok, I. Ahmad. FASTEST, "a practical low-complexity algorithm for compile-time assignment of parallel programs to multiprocessors", IEEE Trans. Parallel Distrib. Comput., 10(2) ,pp.147-159,1999
[2] H. Casanova, A. Legrand, and L, Marchal. "Scheduling Distributed Applications: the SimGrid Simulation Framework", 3rd IEEE Int'l Symposium on Cluster Computing and the Grid (CCGrid'03), 2003
[3] E. Ilavarasan, P. Thambidurai, "Levelized Scheduling of Directed A-cyclic Precedence Constrained Task Graphs onto Heterogeneous Computing System",pp. 262-269,2005
[4] Atakan Dogan and Fusun Ozguner, "LDBS: A Duplication Based Scheduling Algorithm for Heterogeneous Computing Systems",Proc. of Int'l Parallel Processing (ICPP'02), 2002
[5] E. Ilavarasan, P. Thambidurai, "Low Complexity Performance Effective Task Scheduling Algorithm for Heterogeneous Computing Environments", Journal of Computer Sciences, 3(2): pp.94-103, 2007
[6] Z.G.Chen, Q.S.Hua, EZDCP, "A new static task scheduling algorithm with edge-zeroing based on dynamic critical paths", Journal of Central South University of Technology, 10(2), pp.140-144, 2003
[7] Henri Casanova, "SimGrid: A Toolkit for the Simulation of Application Scheduling", First IEEE/ACM International Symposium on Cluster Computing and the Grid, pp.430-437,2001
[8] Hatice Tekiner-Mogulkoc, David W. Coit, Frank A. Felder, "Electric power system generation expansion plans considering the impact of Smart Grid technologies", International Journal of Electrical Power & Energy Systems, Vol. 42, no. 1, November, pp. 229-239,2012
[9] Ahmad Usman, Sajjad Haider Shami , "Evolution of Communication Technologies for Smart Grid applications ",Review Article.Renewable and Sustainable Energy Reviews, Vol.19, March, pp. 191-199, 2013
[10] A.P, Malozemoff, "New Material Requirements for Superconductor Grid Technology", Original Research Article.Physics Procedia, Vol. 36, pp. 1429-1433, 2012
[11] Vincenzo Giordano, Gianluca Fulli, "A business case for Smart Grid technologies",A systemic perspective Original Research Article.Energy Policy, Vol. 40, January , pp.252-259,2012
[12] Microsoft.Windows Malicious Software Removal Tool[EB/OLI].[2007-12-20] http://www.microsoft.corn/seeurity/malwareremoge/.
[13] The adore-ng Rootkit[EB/OLI].[2012-02-23]. http://stealth.openwall.net/rootkits/.
[14] Wanglina, gaohanjn, liuwei, pengyang. Detection and management of virtual machine monitor. Research and development process of [J]. computer. 2011. pp:1534-1541
[15] Rising computer virus statistics. [EB/OLI]. [2008-03-05]. http://www.rising.com.cn/2007/annual/index.htm
[16] Yinghua Xue, Hongpeng Liu ,Intelligent Storage and Retrieval Systems Based on RFID and Vision in Automated Warehouse. Journal of Network.Vol 7, No 2 (2012),pp：365-369
[17] Haipeng Qu, Lili Wen, Yanfei Xu, Ning Wang, LCCWS: Lightweight Copyfree Cross-layer Web Server, Journal of Network Vol 8, No 1 (2013),pp：165-173
[18] Huan Zhao, Kai Zhao, He Liu, Fei Yu, Improved MFCC Feature Extraction Combining Symmetric ICA Algorithm for Robust Speech Recognition, Journal of Network multimedia ,Vol 7, No 1.2012.pp:7
[19] Luo Y., Li L., Zhang B.S., Yang H.M. Video Hand Tracking Algorithm Based on Hybrid CamShift and Kalman Filter. Application Research of Computers, Vol.26, No.3, pp.1163-1165, 2009.

Lijun Cao, Born in 1971 , master, associate professor, college of mathematics and information science of Hebei Normal University of Science and Technology, main research directions are: data mining, grid technology.

Xiyin Liu, Born in 1970 , master, associate professor, College of mechanical and electrical engineering, of Hebei Normal University of Science and Technology, main research directions are: data mining, grid technology.

Hans-Georg Torkel, Born in 1960, Master engineers, patent engineer, Principal Of Vocational College of Technology and computer science(BTI).