A Priority-based Task Scheduling Algorithm in Grid

Weifeng Sun, Yudan Zhu, Zhiyuan Su, Dong Jiao, Mingchu Li[1]
School of Software, Dalian University of Technology, Dalian 116621
Wfsun@dlut.edu.cn, zhu.yudan1205@gmail.com, ssdutsu@qq.com, jiaodong0418@yahoo.com.cn, mingchul@dlut.edu.cn

Abstract

Grid is proposed to solve large scale and complicated problems, and it is a form of parallel computing on Internet. Task scheduling in grid is one of the most important technologies in grid system, and it is a NP complete problem, which is used to schedule a task on an appropriate grid node. Task scheduling algorithms are used to improve the grid performance by minimizing the scheduling length. A priority-based task scheduling algorithm (P-TSA) in grid is proposed in this paper. In this kind of priority-based algorithm, tasks are scheduled according to the priority order firstly. And then assign processors Comparing P-TSA with existed grid scheduling algorithms on scheduling length and resource utilization rates. Simulations are done on examples of DAG, and simulation results shown that the performance of P-TSA is better than other scheduling algorithms such as Min-min and Max-min.

1. Introduction

Grid [1] has become a most important way to solve large-scale and complicate problems since Ian Foster proposed the concept of it. Grid computing is the most distributed form of parallel computing, and it makes use of computers communicating over the Internet to work on a given problem. Grid is used to connect the most kinds of widespread resources in the Internet (including computing resources, storage resources, bandwidth resources, software resources, data resources, information resources, knowledge resources, etc.) to logic holistic. Grid can provide users with integrated information and application services (computing, storage, accessing etc) to realize resource sharing and collaboration in this virtual environment. Grid scheduling, which studies how tasks are scheduled on resource computers, is one of the key point of improving performance of grid. Efficient task scheduling strategies and algorithms in grid can fully utilize the processing power of grid system and improve the performance of it. Grid applications are usually divided into many interdependent subtasks in real applications. Every single subtask is processed dependently and the subtasks should process concurrently in order to reduce the task running time, which is one of the most important problems in parallel computing. Grid scheduling objective function is to map the independent tasks to the processor and decide the order of task execution to minimize the total running time. Directed acyclic graph (DAG) is usually used to illustrate the data dependency among subtasks, and it can be used to solve task scheduling problems. A novel grid task scheduling model based on DAG is proposed in this paper, which can get better performance.

The rest of paper is organized as follows: section 2 is the related work; section 3 proposes the P-TSA algorithm; section 4 elaborates the simulation and result analysis is done. Section 5 draws the conclusion and discusses about the future work.

2. Related work

Many static scheduling algorithms in grid and in parallel computing have been proposed in the last few years. List scheduling methods maintain a priority queue according to the priority order of task graph, which can be divided into two parts. One is the part of task priority, which selects the high-priority waiting tasks to be scheduled. The other part is the processor-selecting part, which selects the appropriate processor to minimize the payment function. Min-min, Max-min [2, 3], HPS [4], DCP (Dynamic Critical Path) [5] are some kinds of list scheduling methods. Min-Min algorithm begins with the set MT of all unassigned

[1] Corresponding author

tasks, and it also has two phases. In the first phase, the set of minimum expected completion time (such that task has the earliest expected completion time on the corresponding machine) for each task in MT is found. In the second phase, the task with the overall minimum expected completion time from MT is chosen and assigned to the corresponding resource. Then this task is removed from MT and the process is repeated until all tasks in the MT are mapped. Max-Min is very similar to Min-Min, except in phase 2. Max-Min assigns task with maximum expected completion time to the corresponding resource, in phase 2.

Priority-based schemes [6-8] assume a priority for each task that is utilized to assign the tasks to the different processors. Priorities based scheduling algorithms, such as Heterogeneous Earliest Finish Time (HEFT) and Critical-Path-On a Processor (CPOP). The heterogeneous earliest finish time (HEFT) [9] algorithm has two phases: the task prioritizing phase and the processor selection phase. In the task prioritizing phase, the upward rank attribute is computed for each task and a task list is generated by sorting the tasks in decreasing order of the upward rank. In the processor selection phase, tasks are selected in order of their priorities and scheduled to the best processor that minimizes the task's finish time. The CPOP [9] algorithm has the same two phases as the HEFT algorithm. In the first phase, the upward and downward rank values for all tasks are computed using mean computation and communication costs. The priority of each task is assigned with the summation of the upward and downward ranks. A priority queue is used to maintain the ready task, and the ready task with the highest priority is selected for the processor assignment. In the second phase, it defines the critical path processor as the processor that minimizes the cumulative computation cost of the tasks on the critical path. If the selected task is on the critical path, it is assigned to the critical path processor; otherwise, it is assigned to a processor that minimizes the earliest execution finish time of this task.

Max-min, Min-min and such algorithms don't consider the load balancing. The HEFT and CPOP have no idea about the topology structure and resource availability. The paper studies the minimum scheduling time, task topology structure and the load balancing, namely resource utilization.

3. P-TSA

3.1 Task scheduling problems

A grid application consists of many dependent tasks, which are represented by a Directed Acyclic Graph (DAG).

G=(V, E), where V={v_i, i=1…n) is the set of n tasks. The v_i represents the node i.

E =(v_i,v_j) {i,j=1……n} is the set of directed edges , which represents a partial order on V.

For any two tasks v_i, $v_j \in$ V, the v_j cannot be scheduled until task v_i has been completed, hence v_i is a predecessor of v_j and v_j is a successor of v_i. The tasks executions of a given application are assumed to be non-preemptive.

$Data_{nn}$ is a $n \times n$ matrix of communication data, where $Data_{ij}$ is the amount of data required to be transmitted from task v_i to task v_j.

In a given task graph, a task without any parent is called an entry task and a task without any child is called exit task.

In an actual implementation, we can create a PseudoEntry task and PseudoExit task with zero computation time and communication time.

The DAG of Fast Fourier Transform is shown in Fig.1. The weight of edge denotes the processing and transmitting time from one node to the next node.

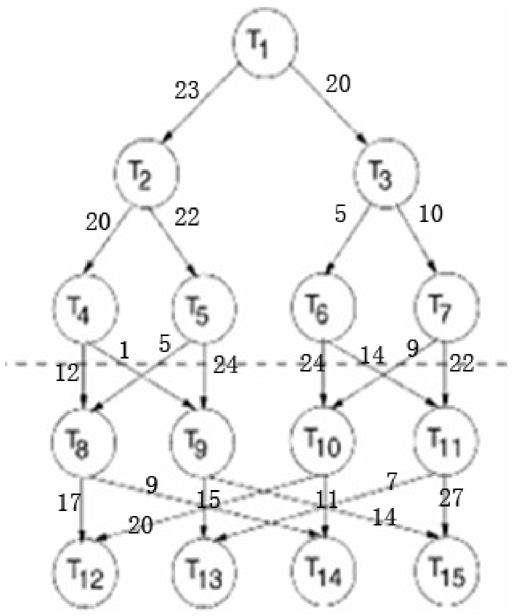
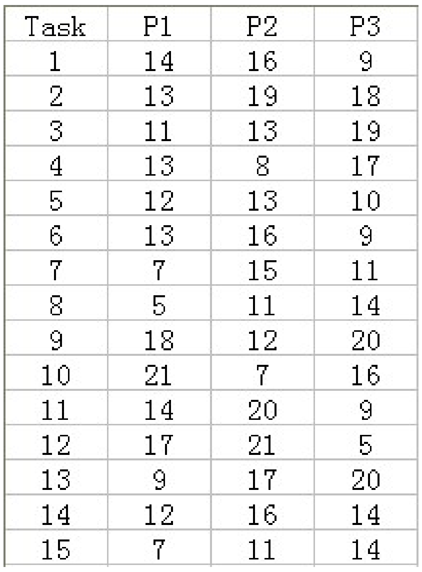


Figure 1. **A DAG of fast fourier transform**

Assume grid system consists of m independent different types of processors of a set P = { p_j: j =0,…, m-1}. The estimate accomplish time that a set of tasks execute on the 3 processors is shown in Tab.1.

Table 1. **The execution time matrix**



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
11	14	20	9
12	17	21	5
13	9	17	20
14	12	16	14
15	7	11	14

R_{mm} is an $m \times m$ matrix of data transmission rate, so that the transmission time between task v_i (execute on the p_x) and task v_j(execute on the p_y) is shown in formula 1. If the execution machine of the v_i is the same with the v_j, then the transmission time is zero. If not, the time is the amount of transmission data (The transmission rate defined is 1).

3.2 Base algorithm

First of all, every task's priority (TP) is computed by formula 1, where depth(i) represents the depth of task v_i . Depth of node(depth) is used to describe the depth of task v_i and we suppose that depth(entry)=1.And v_j is is the next node of v_i.

All of the tasks will be sorted by their priority values. The ECT(estimate complete time) is the time which every processors executes the task. The ULC (Up Link Cost) of a task is the maximum data transmission time with its immediate predecessors. The DLC (Down Link Cost) of a task is the data transmission time with its immediate successors. The target of our scheduling algorithm is minimizing scheduling length, called makespan, which is the measurement of grid computing system's throughput.

$$\mathrm{TP(i)}=\frac{\sum_{j \in succ(i)} \frac{1}{depth(j)} * \frac{1}{DLC(i)}}{ULC(i) * average(ECT(i))} \quad (1)$$

The notes which have the same depth are in the same team, and the lower depth team has a higher priority. Then we will get the executing order by computing the priority value of tasks in every team. So we can get the ordered tasks from figure 1, which is illustrated in figure 2.

T1 T3 T2 T6 T7 T5 T4 T8 T11 T10
T9 T14 T13 T15 T12

Figure 2. **The ordered tasks**

At last, we assign the tasks with established order to executing machines. Before doing that, the resource machines will be scanned and the tasks' executing time of machines (m_i) that cannot provide the resource will be changed into ∞. With regard to every task, each of them will travel across all the resource machines to record the completion time when assigning tasks. The resource machine having a minimum completion time will be chosen for scheduling. We can minimize the total completion time by this way.

Figure 3 shows a scheduling result created by P-TSA.

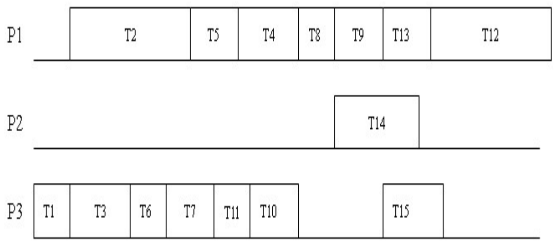


Figure 3. **The scheduling result generated by P-TSA**

The three processers should process the subtask at the proper time an all the processing orders are based on the priorities of DAG.

3.3 Algorithm description

To assess the performance of the P-TSA, we make a series of simulation experiments using the Matlab. P-TSA algorithm pseudo-code is as follows:

1. **Read** the DAG
2. **Divide** the tasks into groups
3. **Initialize** the Estimated Completion Time (ECT) of each task, the amount of data between tasks, and the transmission rate between the machines (define the value is 1). And **initialize** the Machine Available Time (MAT).
4. **For** each group do

```
5.   Begin
6.     compute the average ECT of every task in this group
7.     Sort the tasks in the group
8.     For each task in this group do
9.       Begin
10.        For each machine do
11.          Begin
12.            $ACT(v_i)=Max\{ MAT,EST(v_i)\}+TET(v_i)$
13.            Select the minimum ACT
14.          End
15.        Delete the task from the group
16.        Update the value of MAT 、ACT and EST of the tasks
17.      End
18.  End
```

4. Simulations and performance analysis

Scheduling length and resource utilization are the parameters to be compared in the simulations. Scheduling length of grid task, an indicator of performance of the grid system, is important in task scheduling system. While the resource utilization reflects the load of resource machine which is also important to the grid system. So the two parameters are selected to compare the effect of P-TSA.

P-TSA algorithm is compared with Min-min and Max-min algorithm in the simulation by using the DAG map.

4.1. Simulation environment

Application maps of three typical problems are used in the simulation, Fast Fourier Transform (Fig.1), the random DAG (Fig.4) and Molecular Dynamics Code (Fig.5). Fast Fourier Transform algorithm is the example in this paper. We focus on the analysis and comparison of random generated DAG and Molecular Dynamics Code in this simulation. In this paper, we design the random parameter generator to get the scheduling result. The experimental parameters are as follows:

1. The number of hosts *m*,
2. $m \in (10,20,30,40,60)$ for random generated DAG;
   $m \in (2,4,8,16,32,64)$ for Molecular Dynamics Code;
3. The eliminate completion time of task on the resource host ECT;
4. The data transmission between spots DT (Data Transmission);
5. The ratio between average of DT and average of ECT, set to be 1.

According to the above parameters, the paper runs the Max-min, Min-min and P-TSA algorithm based on the two DAGs.

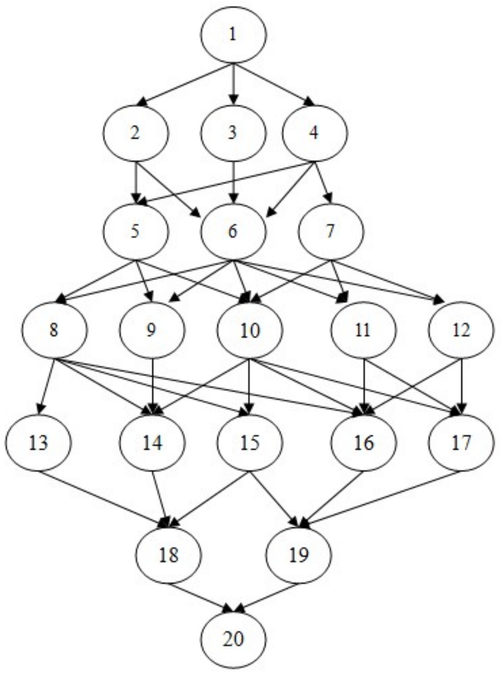


Figure 4. **The random DAG**

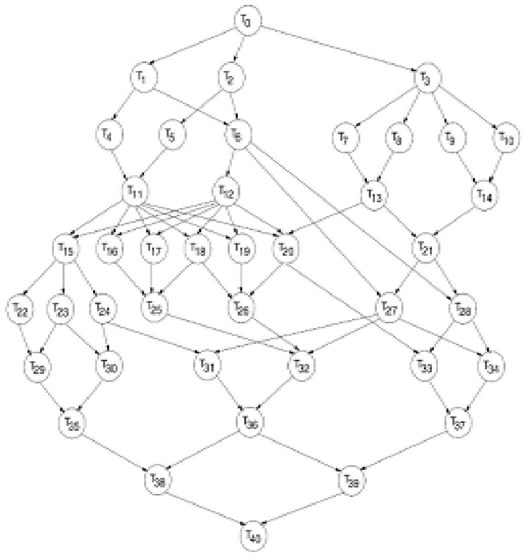


Figure 5. **DAG of molecular dynamics code**

4.2 Experiment results and analysis

Using the two DAGs, the makespan based on P-TSA and the other two algorithms show as the fig.6 and fig.7 while the resource utilization show as the fig.8 and fig.9.

As randomly generated DAG experiment, the resource utilization of Max-min and Min-min are almost the same. When the number of resource machines is selected to be {10, 20,30,40,60} based on the random DAG and {2, 4, 8, 16, 32, 64} based on the real application DAG, the scheduling length curve of P-TSA algorithm is under the curve of Max-min and Min-min algorithm.

The makespan based on P-TSA and the other two algorithms show as the fig.6 and fig.7.

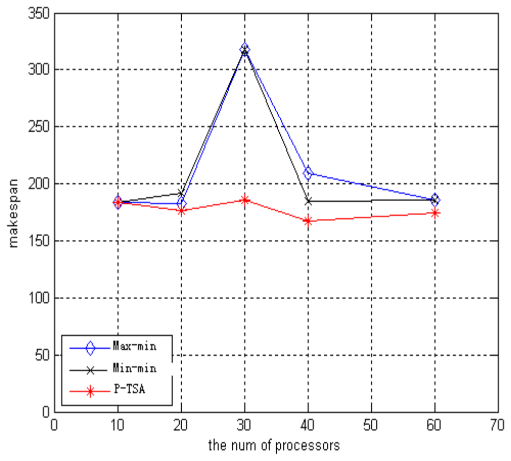


Figure 6. **The makespan contrast with the random DAG**

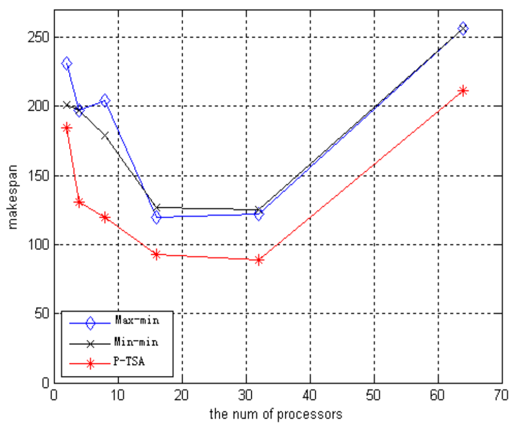


Figure 7. **The makespan contrast with molecular dynamics code**

The simulation results show that the average of scheduling length of P-TSA is smaller than the Max-min and Min-min. The average of scheduling length of P-TSA based on a random generated DAG is 82.2% of Max-min algorithm and 83.4% of Min-min algorithm, and the scheduling length can save 41.6% time at most. The minimal makespan occurred when the number of processors is 16 or 32 in molecular dynamics code scenario. But the makespan is high when the number of processors reaching 30 in random DAG scenario. The nodes in the two scenarios are different, while the changing of makespans in different scenarios is quite different.

The resource utilization of three algorithms in the simulation is shown as fig.8 and fig.9.

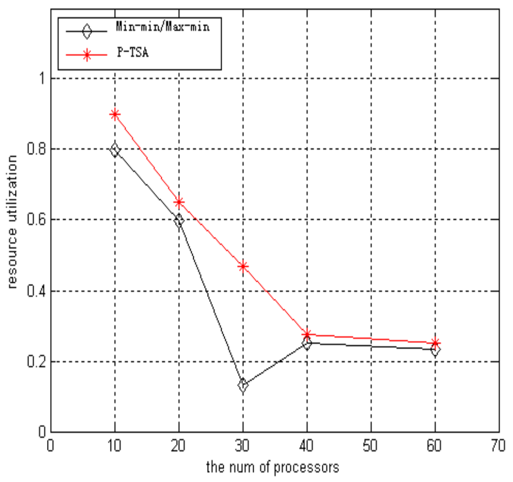


Figure 8. **The resource utilization contrast with the random DAG**

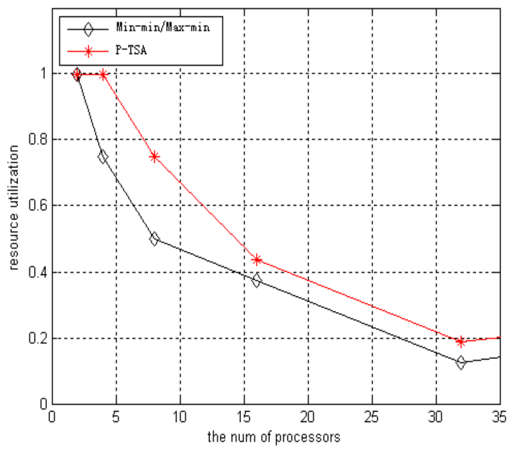


Figure 9. **The resource utilization contrast with molecular dynamics code**

The resource utilization of P-TSA is increased by 12.5 percentage points compared with the Max-min and Min-min algorithm. The average of scheduling length of P-TSA based on DAG of molecular dynamics code is 73.3% of Max-min algorithm and 76.3% of Min-min algorithm, and the scheduling length can save 41.2% time at most. The resource utilization of P-TSA is increased by 11.6 percentage points compared with the Max-min and Min-min algorithm. The changing of resource utilization in different scenarios is also quite different.

P-TSA shows better performance than Max-min and Min-min in the makespan and resource utilization based on the graphs of real applications and random DAG, especially in the environment with large num of the tasks and processors.

5. Conclusion

Grid computing is a special form of parallel computing, the scheduling algorithm in grid can solve the scheduling problems in parallel computing. A QoS-aware algorithm P-TSA is proposed in this paper to solve the priority-based task scheduling problem in grid. P-TSA divides tasks by priority and schedule grids according to the priority. Considering the service migration, the P-TSA algorithm can get smaller grid scheduling length in solving the dependent task scheduling problems. Compared with Min-min and Max-min algorithm by using the randomly generated DAG and DAG used in the real world, the P-TSA have had a better performance and got a distinct advantage on the aspect of scheduling length.

The scheduling length is the only factor to be considered in this paper and other QoS factors will be considered in the P-TSA algorithm in the future work.

6. Acknowledgement

This paper is partially supported by Supported by Nature Science Foundation of China under grant No.: 60673046，60903153.

7. References

[1] Ian Foster. What is the grid? A three point checklist, Grid Today, vol.1, pp.6-12, 2002.

[2] Ehsan Ullan Munir and Jian-Zhong Li. Performance Analysis of Task Scheduling Heuristics in grid. Proceedings of the Sixth International Conference on Machine Learning and Cybernetics, Hong Kong,19-22 August 2007.

[3] Ladislau L.boloni and Muthucumaru Maheswaran, A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto heterogeneous Distributed Computing Systems. Journal of Parallel and Distributed Computing 61, 810-837 ,2001.

[4]E.Ilavarasan,P.Thambidurai,R.Mahilmannan,”High Performance Task Scheduling Algorithm for Heterogeneous Computing System,” Lecture Notes in Computer Science3719,pp.193-203,2005.

[5] M.Wu and D.Gajski,”Hypertool:A programming Aid for Message Passing System” IEEE Trans.Parallel and Distirbuted Systems,vol.1,pp.330-343,July 1990.

[6] Marjan Abdeyazdan,Amir Masoud Rahmani,”Multiprocessor Task Scheduling using a new Prioritizing Genetic Algorithm based on number of Task Childern”,Distributed and Parallel System,Springer US,pp.105-114,2008.

[7] M.Iverson, F.Ozguner and G.Follen, “Parallelizing Existing Applications in a Distributed Heterogeneous Environments”, Proc. Heterogeneous Computing Workshop, pp.93-100, 1995.

[8] H. Topcuglou, S. Hariri and M.Y. Wu, “Performance Effective and Low-Complexity Task Scheduling for Heterogeneous Computing”, IEEE Trans. on Parallel and Distributed Systems, vol. 13, No.3, Feb’ 2002.

[9] M.Rahman,S.Venugopal,andR.Buyya.A dynamic critical path algorithm for scheduling scientific workflow applications on global grids. Proceedings of the 3rd IEEE International Conference on e-Science and Grid Computing,Bangalore,India,2007.