Int. J. Communications, Network and System Sciences, **2011, 4, 227-231**
doi:10.4236/ijcns.2011.44027 Published Online April 2011 (http://www.SciRP.org/journal/ijcns)

An Improved Task Scheduling Algorithm in Grid Computing Environment

Liang Yu, Gang Zhou, Yifei Pu
College of Computer Science, *Sichuan University*, *Chengdu*, *China*
E-mail: *yuliang@scu.edu.cn*
Received January 21, 2011; *revised February* 19, 2011; *accepted February* 21, 2011

Abstract

Algorithm research of task scheduling is one of the key techniques in grid computing. This paper firstly describes a DAG task scheduling model used in grid computing environment, secondly discusses generational scheduling (GS) and communication inclusion generational scheduling (CIGS) algorithms. Finally, an improved CIGS algorithm is proposed to use in grid computing environment, and it has been proved effectively.

Keywords: Grid Computing, Model of Task Scheduling, Heuristics Algorithm, Dependent Task Scheduling Algorithm

1. Introduction

How to effectively dispatch the task of Scheduling Computing is one of the most important factors in the success of grid computing. Users can share grid resources through submitting computing tasks to grid system. According to some strategy, grid scheduling process allocates those tasks to appropriate resources. Efficient scheduling algorithm can make good use of processing capacity of the grid system, thereby improving application performance. Grid system with the goal of optimizing throughput for task scheduling has been proved to be NP complete problem, so this has often using heuristic method to search approximate optimal scheduling program. Heuristic method which often based on visual inspiration is a approximation algorithm. The method that has gradually optimized on the basis of feasible solution searches for a similar algorithm at a low polynomial time operation.

2. The Mathematical Description of Direct Acyclic Graph

Applications of grid environment can be described by set which is composed of numbers of subtasks. A grid application procedures can be expressed as a DAG (Direct Acyclic Graph), $G = (T, E)$, T is a node set and E is a set of direct edge. A node t denotes a task in DAG, and a direct edge is denoted by a pair of node (*ti*, *tj*). The first node is called father and the latter is called children node. The node with no father is called entrance, *i.e.*, the beginning of application. The node with no children is called export, *i.e.*, the end of application. **Figure 1** gives an example of DAG task. Among them, nx is a serial number of task and the numbers in brackets are parameters for calculating. The scheduling which is in a processing m unit and a task graph $G = (T, E)$ is a function *f*. Each of task is mapped on certain unit with a specific beginning time by *f*. A scheduling can be described as $f: T \to \{1,2,\cdots,m\}\times[0,\infty]$ in form. If it exits $v \in T$, $f(v) = (i,t)$, Scheduling tasks *V* should be scheduled on the processing unit *Pi* and it starts from time *t*.

3. Dependent Task Scheduling Algorithm GS and CIGS

Dependent task scheduling is an important scheduling

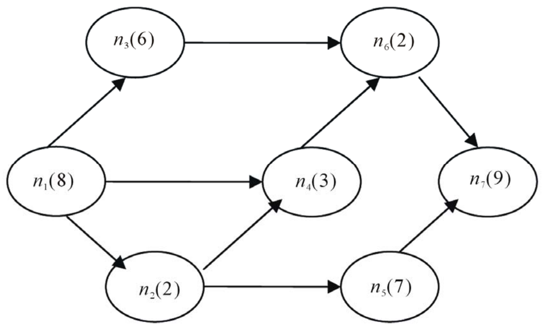

Figure 1. DAG task illustration.

problem, so many research papers have all expounded. Most of the existing heuristic algorithm is based on an independent task, and dependent task is researched in a homogeneous environment with the same structure, which composed of uniformity machine. For example, Shang [1] and others adopt a new list of scheduling technology to control dependence task scheduling in environment with the same structure. Wu [2] and others based on topological sort have proposed a fast local search algorithm to enhance scheduling of quality and efficiency. Oliveira [3] and others have proposed a task of scheduling algorithm with fixed priority under a distributed computer system. Above of algorithms are too many constrains, and not the general, Carter [4] and others have developed Generation Scheduling (GS) to control dependent task in heterogeneous system. The main steps of GS algorithm can be described as follows:

1) Create scheduling problems. The task of interdependent make some tasks start after the completion of other tasks and each task is not set up the initial scheduling. By tracking and analyzing the task of interdependent relationship, when a pilot task of entire mission has been completed, then the task can be set up to scheduling.
2) Filtering mandate. When scheduling events occur at any one time, its not scheduling task can be filtered out. Then the remaining tasks will be scheduled as the "independence" by the formation of a new set *SI* (Set of Independent tasks). Obviously, all of tasks in the *SI* have no predecessors mandate for immediate implementation.
3) Locally scheduling.Although these setting algorithms that can be selected by users should be scheduled, preemptive scheduling model can not be used should be paid attention.
4) Update scheduling problem.When incidents of new scheduling cycle started or repeated have perceived, relevant tasks are collected. The problem of a new dependent task scheduling is formed and the time has been set for the machines. If all the tasks scheduling have finished, then it should be terminated. Otherwise repeat the steps 1-4 (See **Figure 2**).

Carter and others proposed a CIGS (Communication-Inclusive of GS) to consider communication of GS algorithm, which support dependent task scheduling as a typical algorithm in heterogeneous distributed computer system. The algorithm is simple and easy achieving. In CIGS, through analyzing the dependence of task, the tasks that these have been to meet the mandate of current ancestors constrain should be filtered out from all those that have not yet scheduling tasks, *i.e.* current "task independence". The actual scheduling of sequence and matching scheduling are limited to "independence mandate". With these

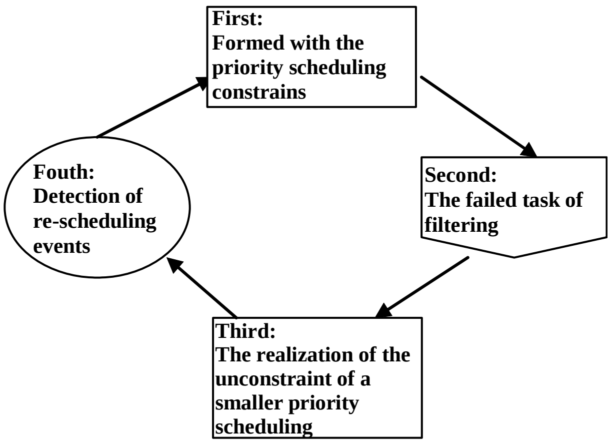
First:
Formed with the priority scheduling constrains
Second:
The failed task of filtering
Third:
The realization of the unconstraint of a smaller priority scheduling
Fouth:
Detection of re-scheduling events

Figure 2. GS overview.

new task scheduling and execution, the dependence tasks possibly become the current "independence mandate", which are scheduled until all tasks are assigned. The number of frequency filters has directly influence on all tasks of expected completion time. The task of makespan firstly filtered out as preparation time of all machine, priority will be ensured with an orderly implementation. However, overall makespan have increased. Its performance could be improved by reducing the number of filters and makespan can be reduced by optimizing the preparation time of machinery. The task of scheduling at any time or has been or is being implemented in the pro- cess of scheduling. When CIGS map tasks to a certain resources node, it could adopt an option heuristics algorithm. Including all relevant data transmission to the host node and the time cost of implementation in the host nodes will be considered in every mapping.

Efficiency of node in the dimension of time or space in grid computing is dynamic change. To allow CIGS can effectively deal with grid environment, the need of related improvement should be done. Otherwise, it would not be appropriate to improve its performance, such as lowering span, raising the utilization rate of resources, etc.

4. The Improvement of Task Scheduling Algorithm CIGS

In order to solve the above problem, the paper presents an improved algorithm CIGS to adapt computational grid environment. In the scheduling process, by making a judgment, if there is a 0 outdegree task in *SI*, it will be filtered to form a new set of *BSI*. Each of *BSI* tasks is not allowed to schedule in the current scheduling cycle, that is to say, *BSI*, such as a free buffer zone. Similarly, its forecast by the data transmission time, the time of expectant execution for all ancestors task and preparation time of machines will be determinant factors of scheduling, thus better performance can be acquired.

4.1. CIGS Algorithm with Improved Constrain Conditions and Definition of Terms

Improved constrain condition includes:

1) An application (the overall mission) can be divided into many tasks. Dependent relationship may exist between these tasks. These restrictions that ancestors meet the conditions are not time-varying, but they would be able to describe by graph DAG.
2) A node machine operates only one task at the same time and the task will not be seized by others.
3) Heuristics would be the same in every cycle. When data transmit, the communicated cost of transmission will be used as one of decisions for scheduling.
4) Although the performance and status of network and nodes are dynamic changes in grid environment, they can be accurately predicted at a time in scheduling. Execution of tasks and the communicated cost of transmission also can be predicted.
5) All tasks scheduled to nodes usually can be implemented. If a node failure at any time, its task will automatically enter a new scheduling problem sets in next cycle. Checkpoint mechanisms, movement and error recovery mechanisms are not be supported in scheduling.

The improved definition of terms:

Improved CIGS algorithm for the main definition of terms described. Decomposition has been described here is the application which is divided into *N* tasks that there are part or all dependent relationship between.

1) *N* is the total number of all tasks in grid scheduling application, set *S* is scheduling tasks. $S=\{S[0],S[1],\cdots,S[N-1]\}$.
2) *M* involved in matching scheduling in grid is the number of node machines, machines based scheduling for the pool *H*, $H=\{H[0],H[1],\cdots,H[M-1]\},M\geq 2$.
3) *SI* is a task set, which all the elements of can be executed when its all ancestors have completely implemented. It is dynamic set in scheduling.
4) *BSI* is a subset of *SI*, and exists in scheduling cycle. Every element is not predecessor for any unexecuted mission. It is filtering through from the current *SI* that outdegrees of all tasks is 0.
5) *R* is expectation time for node machine in a grid environment. $R=\{R[0],R[1],\cdots,R[M-1]\}$, $R[J]$ is node or machine $H[j]$ expectations for period.
6) $E=\{E[0][0],E[0][1],\cdots,E[N-1][M-1]\}$, $E[k][j]$ is expressed as expected execution time of $S[k]$ task in machinery $H[j]$.
7) *D* is data-transmission capacity of the matrix size in nodes operating for task. $D=\{D[0][0],D[0][1],\cdots,D[N-1][M-1]\}$, $D[k][j]$ said for the implementation of mandate $S[k]$ in machine, which are essential for the quantity of data transmission. If $S[k]$ execute in their own data sets $H[j]$, the $D[k][j]=0$.
8) *TC* execute in expected data transmission time matrix of nodes or tasks, $TC=\{TC[0][0][0],TC[0][0][1],\cdots,TC[N-1][M-1][M-1]\}$, $TC[k][i][j]$ stands the necessary expected time of data transmission for task $S[k]$ between node $H[i]$ and node $H[j]$.
9) C is the time matrix expected to completion of task, $C=\{C[0][0],C[0][1],\cdots,C[N-1][M-1]\}$, $C[k][j]$ is expressed as expected completion time of $S[k]$ task in node or machinery $H[j]$.
10) *LC* and *LM* stand for complete time of expected execution in a machine for task, and they are a temporary set. $LC=\{LC[0],LC[1],\cdots,LC[N-1]\}$, and $LM=\{LM[0],LM[1],\cdots,LM[M-1]\}$.
11) Makespan is complete time for entire tasks. It means that all the biggest complete time of task can be expressed as following:

$$Makespan=\max_{k=0}^{N-1}\left(LC[k]\right)$$

12) *TR* is dependent matrix for the task. Assuming the dependence $TR[k][t]$ of task between $S[k]$ and $S[t]$. If $TR[k][t]=1$, $S[k]$ depend on $S[t]$. Only if $S[k]$ is completely executed, $S[t]$ will be executed. Symbol "→" is defined as dependence, *i.e.* if $S[t]\to S[k]$, $S[k]$ depend on $S[t]$. $S[k]$ is successor (or son) of $S[t]$, $S[t]$ is pilot (or father) of $S[k]$.
13) DAG $=\langle V,U\rangle$ (Direct Acyclic Graph, DAG), the node $V[k]$ stands for task $S[k]$, the direct edge $U[k,t]$ that is from $V[k]$ to $V[t]$ said for dependence of $S[k]$ and $S[t]$.
14) *ID* is an indegree set of all node tasks, *OD* is an outdegree set. $ID=\{ID[0],ID[1],\cdots,ID[N-1]\}$, $OD=\{OD[0],OD[1],\cdots,OD[N-1]\}$.

4.2. The Computing of Indegree and Outdegree for DAG

DAG node indegree is the number of precursor to nodes and node indegree can be acquired by computing the precursor of a node. The method of computing precursors is simple. According to the relationship matrix between tasks, the precursor of a node is corresponding to the number of line elements "1" in matrix. The indegree of a node is sum of the line elements.

The method is computing the number of successors. According to the relationship matrix between tasks, the successor of a node is corresponding to the number of array elements "1" in matrix. The outdegree of a node is sum of the array elements. Indegree $ID[k]$ and outdegree $OD[k]$ of node $V[k]$ can be expressed by formula as follows:

$$ID[k] = \sum_{i=0}^{N-1} TR[k][i]$$

$$OD[k] = \sum_{i=0}^{N-1} TR[i][k]$$

4.3. The Assigned Priority of DAG Node

Filtering out non-reliance tasks from set *S* of dependence tasks can be used in various ways. The paper adopts filter method by priority. Based on the filtering method of priority can guarantee the same priority tasks with non-reliance relationship. The priority of node $V[k]$ and task $S[k]$ can be acquired by formula as follows:

$$TP[k] = \max(TP[i]) + 1$$

$$S[i] \in PS[k]$$

There are main steps for assigned priority number of DAG node:

1) Labeling all nodes as Unassigned and putting all nodes in an unassigned set US.
2) The source node is assigned as the highest priority level 0.
3) Labeling nodes that have been assigned priority as Assigned. Deleting it from US, the nodes will be put into an assigned node set AS. Is US empty? If it is empty, go to step 7. Otherwise, go to step 4.
4) Taking a node form unassigned priority nodes.
5) If pilots have been assigned priority number, the highest priority number *P* should be found in pilot node. Otherwise, go to step 4.
6) The priority number should be assigned as *P* + 1, go to step 3.
7) Getting the largest priority number P_{max} of all tasks and assigning each individual task to priority number $P_{max.}$
8) Algorithm end.

4.4. The Key Steps of Improving Algorithm

After completing above computing work, CIGS scheduling algorithm is improved in accordance with the following order. Scheduling summarized as follows:

1) Finding out all nodes that priority number is 0 in AS (Filtering out all source nodes from DAG), and these tasks will be set into the tasks set *SI*. Searching each task $SI[k]$ that belongs to *SI*, only if $OD[k] = 0$. This task will be filtered out and put into the task set *BSI*. To be known or to identify the distribution of data sets, nodes and network status information can be acquired by grid tools. Updating *H* and *R*, if scheduled task is $S[k]$ to every machine, situation of operation will be forecasted. Obtaining the corresponding expected com- pletion time and it is simultaneously recorded in $LM[j]$ and $LC[k]$.
2) According to validity machine situation of set *H*, the mandate of task set *SI* is scheduled by using certain heuristics algorithm (Min-Min or Max-Min algorithm). But all mandates of task set *BSI* will be overlooked for the time being.
3) All task that its priority number is 1 will be filtered out from set AS and these tasks are put into *SI*. The mandate $SI[k]$ which its $OD[k] = 0$ is filtered out from *SI* and these are put into set *BSI*. Assuming each task $S[k]$ which can be scheduled and executed in machine $H[i]$, algorithm will calculate its expected completion time in accordance with the following formula. Updating *H* and *R*, all tasks are scheduled by expected completion time matrix *C* in set *SI*. And use the same algorithm as step (2).
4) Updating set *H* and *R*, number of validity machines (expressed as N2) is counted in set *H*. All key tasks of set *i.e.* $S[x]$ are statistical calculated and $TP[x] = P = 1$. Only if $N1 \le N2$, the algorithm will go to step (6).
5) According to (3) the method of calculating its expected completion time for each task of set *BSI*, all tasks will be scheduled in set *BSI* in accordance with time matrix of expected completion. The completed scheduling task will be taken out from *BSI*, which used (2) the same algorithm.
6) Executing *P* = *P* + 1, algorithm is executed by cycle from (3) to (5) until all tasks have been scheduled in the pool *SI* and *BSI*.

5. Simulation Test

We use a grid prototype system of computing visualization to test the performance of scheduling algorithm. LAN A and B bandwidth is 10 M and 100 M in this system, which are connected with 10 M bandwidth. We have selected a specific application tasks in the areas of computational fluid dynamics. All tasks exist dependence in applications and their size is random.

Three experiments have established.

1) Three machines are selected in LAN grid environment B, implementation of dependent ten tasks will be tested.
2) One hundred and forty tasks will be tested in seven machines, five machines are chosen in B LAN grid environment and two machines are chosen in A LAN grid environment.
3) One hundred and sixty-five tasks will be testes in 9 machines, seven machines are chosen in B LAN grid environment and two machines are chosen in A LAN grid environment.

Its scheduling algorithm has used by CIGS-Greedy, CIGS-Min-Min, improved CIGS-Greedy, and improved CIGS-Min-Min. According to the performance of span makespan, test results are shown below in bargraph:

According to the bargraph (**Figure 3**), when the number of dependent tasks has increased, the span reduces obviously. In contrast, the number of tasks is less, the span between them become small. When the core matching algorithm is adopted, the span of improved algorithm

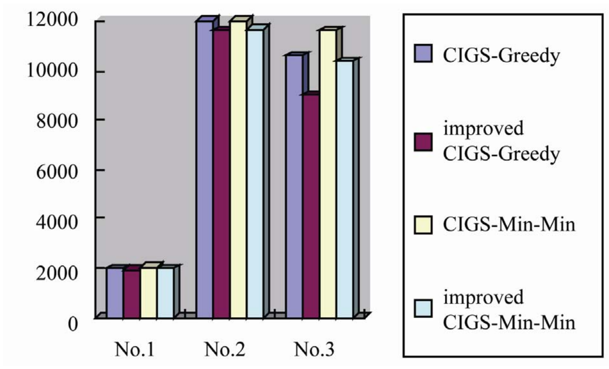

Figure 3. Scheduling performance comparison.

CIGS compared with CTGS algorithm is always small. In other word, improved CIGS algorithm can effectively scheduled for dependent task and management performance is improved in computing grid environment.

6. Conclusions

Algorithm research of task scheduling is one of the key techniques in grid computing. The paper has analyzed scheduling problem that composed of numbers of dependent tasks. There is in-depth analysis for the typical task scheduling algorithm GS in heterogeneous systems and the CIGS algorithm considered the cost of communication. The paper proposed an improved CIGS algorithm that is suitable for computing grid. The corresponding experiments show that improved CIGS algorithm can improve performance of execution.

7. References

[1] M. Shang, S. Sun, *et al*., “An Efficient Parallel Scheduling Algorithm of Dependent Task Graphs,” *Proceedings of the 4th International Conference on Parallel and Distributed Computing, Applications and Technologies*, Chengdu, 27-29 August 2003, pp. 595-598. doi:10.1109/PDCAT.2003.1236372

[2] M. Wu, W. Shu, *et al*., “Efficient Local Search for DAG Scheduling,” *IEEE Transactions on Parallel and Distributed Systems*, Vol. 12, No. 6, 2001, pp. 617-627.

[3] R. S. Oliveira and J. S. Fraga, “Fixed Priority Scheduling of Tasks with Arbitrary Precedence Constraints in Distributed Hard Real-Time Systems,” *Journal of Systems Architecture*, Vol. 46, No. 9, 2000, pp. 991-1004.

[4] B. R. Carter, D. W. Watson, *et al.*, “Generational Scheduling for Dynamic Task Management in Heterogeneous Computing Systems,” *Journal of Information Sciences*, Vol. 106, No. 1, 1998, pp. 219-236.