A Heuristic Scheduling on Heterogeneous Tree Network

Kun Huang[1] Zhiyan Wang[1] Xiaoxiong Weng[2]

[1]Department of Computer Science and Engineering, South China University of Technology, Guangzhou 510640, China

[2]Department of Traffic Engineering, South China University of Technology, Guangzhou 510640, China

Abstract

This paper firstly discusses the problem of independent tasks scheduling on tree network, where resources have different speeds of computation and communication. And then analyzes the property of tree-shaped logical network topologies, presented an integer linear programming for this problem, and a heuristic scheduling is also proposed. At last, a demand-driven and dynamic heuristic algorithms: TreeGrid is developed. The experimental results show that the algorithms for the scheduling problem obtain better performance than other algorithms.

Keywords: Grid computing, Task scheduling, Integer linear programming;

1. Introduction

The popularity of the Internet and the availability of powerful computers and high-speed networks as low-cost commodity components are changing the way we use computers today. These technical opportunities have led to the possibility of using geographically distributed and multi-owner resources to solve large-scale problems in science, engineering, and commerce. Recent research on these topics has led to the emergence of a new paradigm known as Grid computing. The general problem of scheduling tasks to machines has been shown to be NP-hard. Scheduling the tasks of a parallel application on the resources of a distributed computing platform efficiently is critical for achieving high performance. The scheduling problem has been studied for a variety of application models, many heuristic scheduling have been developed, such as Min-Min, Max-Min, FCFS,GA [2] etc.

There are some discussions about tasks scheduling in tree-based platform. Paper [1] prove that the problem of tasks scheduling on tree network is NP-hard, In paper [11] , A realization and some algorithms on tree-based Grid are offered. Paper [7] discusses the problem of divisible load in tree network. The state of the tree and star computational environments is surveyed and some open problems are discussed in paper [3].

The remainder of this paper is organized as follows. In Section 2, we detail our platform and cost models. In Section 3, we analyze the tree network and propose a linear programming about this model. A dynamic heuristic algorithm is offered in Section 4. And the simulation and experimentation about this algorithm is reviewed in Section 5. At last, a summary and some issues worthy of further exploration are proposed.

2. The platform and cost models

In this paper, we limit our discussion to tree-shaped logical network topologies. In this tree network, the network and the processors have different speeds. Because of the topologies, it is easy to implement master-worker computations, RPC (remote procedure call) etc.

For communications, the one-port model is used: The master can only communicate with a single worker at a given time-step. We assume that communications can overlap computations on the workers: A worker can compute a load fraction while receiving the data necessary for the execution of the next load fraction. And a worker (processor) can at the same time receive one task while sending another one to one of its children.

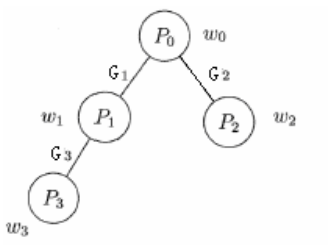


Fig. 1: An example of a tree.

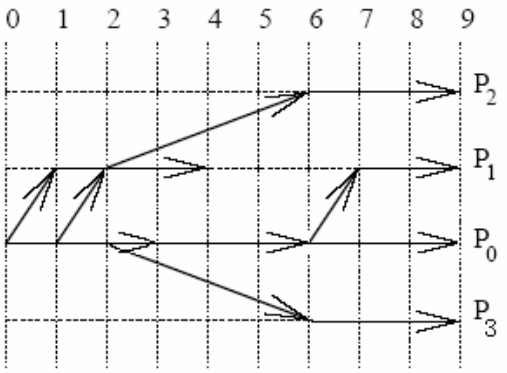


Fig. 2: An example of a schedule.

The example provided in figure 1, 2 shows how a schedule can be executed. The tree is shown on figure 1. The numbers in the circles are the computation times associated with the nodes. It is the time it takes for a task to be executed on the node. The numbers on the edges are the communication time. They represent the time needed to send one task using the labeled link. On the diagram, vertical dotted lines are time units, horizontal dashed lines are related to nodes. The horizontal arrows are the execution of the tasks, and the oblique ones are the communications. This kind of settings will be used in the diagrams describing the scheduling in the following sections.

In this model, it is usually assumed that every node has an unlimited buffer capacity. However the proof is still valid if all the nodes have a buffer of size one.

3. The tasks scheduling on heterogeneous tree network

As shown in Figure 1, in this article, we assume:
(1) P= {P_0, P_1… P_{k-1}} represent the workers in the tree network. There are k nodes in this network. P_0 is the root node, and $P_{1...}$ P_{k-1} are its children nodes.
(2) X_i is the sum of the tasks that the node P_i executed.
(3) Each worker P_i has a computing power W_i: It takes X_iW_i time units to execute X_i units of load on worker P_i.
(4) Each worker P_i has a communicational power G_i: it takes X_iG_i time units to send X_i units of load from root to worker P_i.
(5) M is the sum of all tasks.
(6) C_i represent the list of the node P_i's children.

According to the property of tree-shaped logical network topologies, some equalities and inequalities can be proposed from the model.

$$\sum_{i=0}^{k-1} X_i = M \qquad (1)$$

Since X_i represents the sum of the tasks that the node executes, the sum of all X_i is M.

$$1 \leqslant X_i \leqslant M \qquad 1 \leqslant i \leqslant k\text{-}1 \qquad (2)$$

The sum of the tasks of arbitrary node is within [1, M].

$$X_0W_0 \leqslant T \qquad (3)$$

The time executed by root node is less than the time of all tasks finished.

$$X_i(G_i+W_i) \leqslant T \qquad (4)$$

For each node P_i, the time spends in computing and communication is always less than the time of all tasks finished.

$$\sum_{i=0}^{k-1} X_iG_i \leqslant T \qquad 1 \leqslant i \leqslant k\text{-}1 \qquad (5)$$

The sum of time of all nodes spend in communication is always less than the time of all tasks finished.

$$G_i\,(X_i + \sum_{p \in Ci} X_p) \leqslant T \qquad 1 \leqslant i \leqslant k\text{-}1 \qquad (6)$$

For each non-leaf node, the time spend in communication is always less than the time of all task finished. Note that, the content in bracket is the sum of the tasks executed by P_i and its children.

From above, the optimal solution is given by the following linear program:
Minimize T,
Subject to:

$$\begin{cases} (1)\ \sum_{i=0}^{k-1} X_i = M \\ (2)\ 1 \le X_i \le M \qquad 0 \le i \le k-1 \\ (3)\ X_0W_0 \le T \\ (4)\ X_i(G_i + W_i) \le T \qquad 0 \le i \le k-1 \\ (5)\ \sum_{i=1}^{k-1} X_iG_i \le T \qquad 0 \le i \le k-1 \\ (6)\ G_i\left(X_i + \sum_{p \in C_i} X_p\right) \le T \qquad 0 \le i \le k-1 \end{cases}$$

M, T, i, k, W_i ,G_i ,C_i is positive integer, G_i ,W_i , C_i is already known, X_i is variable, T is the time all tasks finished. Minimize T is target function.

The solution of the linear programming can get by the polynomial-time algorithms [8] presented by Karmarker. The algorithmic complexity is O (n^3).

If all tasks are identical independent tasks, the linear programming can change as follows:
Maximize M,
Subject to:

$$\begin{cases} (1)\ \sum_{i=0}^{k-1} X_i = M \\ (2)\ 1 \le X_i \le M \qquad 0 \le i \le k-1 \\ (3)\ X_0 W_0 \le 1E \\ (4)\ X_i (G_i + W_i) \le 1E \quad 0 \le i \le k-1 \\ (5)\ \sum_{i=1}^{k-1} X_i G_i \le 1E \qquad 0 \le i \le k-1 \\ (6)\ G_i \left(X_i + \sum_{p \in C_i} X_p \right) \le 1E \quad 0 \le i \le k-1 \end{cases}$$

1E is one time-unit. Maximize M is target function. The inequalities aim at determining the maximum amount of the workers can process in one time-unit. Note that the only impact factor is the order of the communicational power in optimal scheduling. So we infer: there is an optimal communicational order, the model will reach maximal tasks in one time-unit by using the order.

Proposition 1. if all tasks are identical independent tasks and the loads are large enough, in optimal scheduling, all the nodes will participate in order of increasing communicational power.

Proof. Assume that there are two different systems A and B, and the communicational power $G_A<G_B$, so in one time-unit, the tasks transferred $X_A>X_B$, obviously, in common system, the communicational power G<<W, the computing power (if G>W, the communicational power will bottleneck the system), therefore, the finished tasks $M_A > M_B$.

4. A heuristic algorithms based on linear programming

The solutions of the linear programming above are approximating optimal, include the time of all tasks finished, the lists of tasks assigned to every node. From proposition 1, we know in some condition, all the nodes will participate in order of increasing communicational power G, so we propose a heuristic algorithms based on linear programming: TreeGrid.

We describe the algorithm as following:

Procedure algorithm TreeGrid ( )
Initialization ();
Resource_Discovery ();
//system gets the node's capacities of communication and computation.
Get_Tasks();
Soluting ();
//The root node get tasks, solute the linear programming, the result put in Queue (Task).
Reorder_G ();
//all nodes queue in order of increasing link capacities, the result put in Queue (g).
Do While Queue (Task) not empty
P = Select_Computer(Queue(g));
X = Matching (P, Queue (Task));
//get a node P_i from Queue (g), find matching tasks from queue(Task).
Trans_Task();
Execute_Task()
// send tasks to P_i execute.
End While
Trans_Result()
//All nodes transfer the result to root node, the algorithm end
End

Let's explain the TreeGrid algorithms briefly. Firstly the system checks error. Since the grid system is always heterogeneous, dynamic, we must examine the system's resource carefully. We can get the power of communication and computation by run some loads in every worker; exclude the invalid nodes from system before all tasks start. Secondly according the tasks and the resource, we solute the linear programming, the results put in Queue (Task). Thirdly we reorder the workers so that $G_1 \leqslant G_2 \leqslant \ldots \leqslant G_{k-1}$, the result put in Queue (g). At last do following steps until Queue (g) empty: get a node from Queue (g), find matching tasks from Queue (Task), and execute.

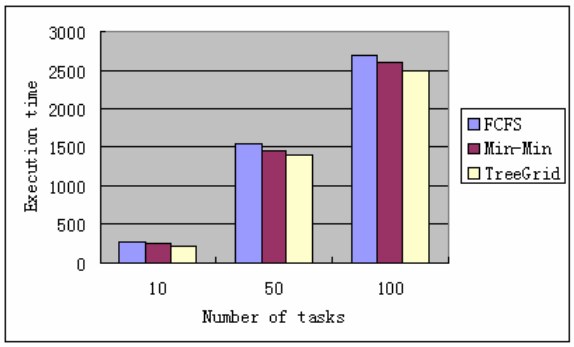
5. Experimental results

In order to prove the effectiveness of the algorithm, we compare TreeGrid with Min-Min [2], FCFS [2] algorithms. The main idea of algorithm Min-Min (Max -Min) is: compute the shortest time of every task, select the shortest (longest) task to matching node execute, then delete the task. Repeat the steps until all tasks finished. The main idea of algorithm FCFS is: Let the first task run first. The two algorithms have better performance in general grid task scheduling, always selected to comparative benchmark.

The experiment use GridSim to simulate the three algorithms above. GridSim provide a grid simulation environment. In this simulation environment, users can easily add various different scheduling policies into the task scheduler and don't need to encode for other parts of the environment repeatedly.

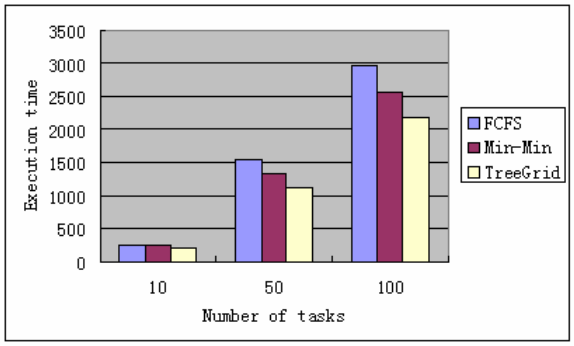
In this experiment, we randomly product 5 different two-level trees networks and 5 multi-level trees networks. The GridSim use virtual time to express time. This makes the results that made by different speed machines become comparable. Take account the computing power of the machine, we limit sum of the nodes within [3, 15], the node's computation power within [30, 60], the edge's

communicational power within [1, 9], the size of tasks within [10, 20], then we run 10, 50, 100 tasks respectively in GridSim, applying the three different algorithms. The results are shown as follows:



	10 tasks	50 tasks	100 tasks
FCFS	258	1597	2967
Min-Min	248	1462	2597
TreeGrid	247	1376	2484
Improvement	0.58%	4.64%	4.36%

Fig. 3: The results of task scheduling on the two-level tree grid computing platform.



	10 tasks	50 tasks	100 tasks
FCFS	269	1547	2962
Min-Min	258	1314	2557
TreeGrid	255	1224	2376
Improvement	1.12%	6.84%	7.06%

Fig. 4: The results of task scheduling on the multi-level tree grid computing platform.

From the figure 3, 4, we can see that the TreeGrid algorithm can increase performance by approximately 4%~7% as compared to the Min-Min case. In two-level tree, the communicational delay is not obviously, the results of the three algorithms are almost same. In multi-level tree, as the communicational delay become more visible, the TreeGrid algorithm become more effective. This proves the TreeGrid algorithm is fit for multi-level tree environment. Note that when the system runs a small amount of tasks, the improvement of the TreeGrid is slight.

The algorithmic complexity of TreeGrid is $O(n^3)$. That is bigger than Min-Min ($O(n^2)$). But the algorithm run only once, in the condition of the loads is large enough, the impact is not great.

6. Summary

An efficient Grid scheduling system is an essential part of the Grid. This paper analyzes the property of tree-shaped logical network topologies, presented an integer linear programming for this problem, at last, a heuristic scheduling is also proposed. The experimental results show that the algorithms for the scheduling problem obtain better performance than other algorithms. On the other hand, task scheduling on tree network is NP-hard also, there are many issues worthy of further exploration, such as the algorithmic complexity, the order problem of the tasks matching to the nodes etc. This need research in future work.

Acknowledgement

This work is partially supported by National Nature Science Foundation of GuangDong Province of China (Grant No. 05011896).

References

[1] P. Dutot, Complexity of master-slave tasking on heterogeneous trees. *European Journal on Operational Research*, 164(3): 690-695, 2005.

[2] T.D. Braun, H.J Siegel., Beck N, 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(6): 810-837, 2001.

[3] O. Beaumont, H. Casanova, A. Legrand and Y. Robert, et.al., Scheduling divisible loads on star and tree networks: Results and open problems. *IEEE Trans. on Parallel and Distributed Systems (TPDS)*, 16(3): 207-218, 2005.

[4] P.F. Dutot, Master-slave tasking on heterogeneous processors. *International Parallel and Distributed Processing Symposium*, IEEE Computer Society Press, 2003.

[5] R. Bajaj, D.P. Agrawal, Improving Scheduling of Tasks in a Heterogeneous Environment. *IEEE Transactions on Parallel and Distributed Systems*, 15(2): 107-118, 2004.

[6] M, Arora, S.K. Das, R. Biswas, A Decentralized Scheduling and Load Balancing Algorithm for Heterogeneous Grid Environments. *Proc. of*

International Conference on Parallel Processing Workshops (ICPPW'02), pp. 499-505, 2002.

[7] S. Bataineh, T. Hsiung, T.G. Robertazzi, Closed Form Solutions for Bus and Tree Networks of Processors Load Sharing a Divisible Job. *IEEE Trans. Computers*, 43(10): 1184-1196, 1994.

[8] N. Karmarkar, A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4): 373-39, 1984.

[9] T. Robertazzi, *Divisible Load Scheduling*, http://www.ece.sunysb.edu/tom/dlt.html, 2004.

[10] V. Bharadwaj, G. Barlas, Scheduling Divisible Loads with Processor Release Times and Finite Size Buffer Capacity Constraints in Bus Networks. *Cluster Computing*, 6(1): 63-74, 2003.

[11] W.W. LIN, Independent Tasks Scheduling on Tree-Based Grid Computing Platforms. *Journal of software*, 17(11): 2352-2361, 2006.