An ECG Parallel Scheduling Algorithm for the Distributed Systems

Maoyuan ZHANG [1,3], Ruixuan LI [1], Zhengding LU [1], Chunyan ZOU [2]
[1]*Department of Computer Science and Technology, HuaZhong University of Science and Technology, Wuhan, P.R.CHINA, 430074*
[2]*School of Foreign Language, HuaZhong Normal University, Wuhan P.R.CHINA, 430079*
[3]*School of Management, HuaZhong University of Science and Technology, Wuhan, P.R.CHINA, 430074*
zmydragon@163.com

Abstract

The problem is that the parallel scheduling strategies are minimally supported in distributed environments. The well-known Coffman-Graham algorithm is highly effective but applied only for the sample case when all tasks have the same execution time and not applied for the distributed systems. In this paper, an expanding task graph and an expanding Coffman-Graham algorithm, which orients to the distributed systems and can be applied for the case when the tasks have the different execution time, are proposed. Like the well-known Coffman-Graham algorithm, the ECG algorithm is proved to be an efficient scheduling algorithm for the distributed systems, through presenting and deducting one theorem and two corollaries.

1. Introduction

Parallel computing is helpful to a wide sector of users and application programmers with minimal extra cost, such as for feature selection of Web pages [1], and for the segmentation method of Chinese words [2]. The problem is that task scheduling may be minimally supported in distributed environments, in which both the execution time and communication time should be taken into account.

On one hand, the execution time of one task is dependent on the computer. According to the scheduling problem with unrelated parallel machines, if task j is assigned to computer k, the time needed to process this task depends on both task j and computer k [3]. Hence, the local computer may not know the execution time of some tasks in other computers.

On the other hand, the problem of scheduling with communication is much harder than that without communication [4]. The communication between two tasks in different computers is not easily evaluated because of routing issues and aggregated delays. Therefore, the local computer may not know the communication time of some tasks in other computers.

For distributed systems, some parallel methods have been recently researched. The backfilling algorithm [5][6][7] allows small tasks to jump ahead, but more backfills may affect the performance of the scheduling. The gang scheduling or co-scheduling algorithm [8][9][10][11] slices the time axis into multiple virtual machines and is thus quite effective in reducing the waiting time at the expense of increasing the apparent task execution time. Moreover, backfilling method, gang method and integrating method [12] do not take the communication time into account, while CAs-based scheduling algorithm [13] assumes the communication time has been known before scheduling.

In the section 2, an expanding task graph and an ECG scheduling algorithm are presented for distributed systems. In the section 3, through presenting and deducting one theorem and two corollaries, the ECG algorithm is proved to be an optimal schedule if the number of processors in a computer is two.

2. Expanding Coffman-Graham scheduling algorithm

In the following part of the paper, as to the local system, the execution time of tasks in the exterior systems denotes the sum of the execution time and communication time, for the convenience of the following analysis.

· This work was partially supported by National Natural Science Foundation of China under Grant 60403027.

Proceedings of the Seventh International Conference on
Parallel and Distributed Computing,Applications and Technologies (PDCAT'06)
0-7695-2736-1/06 $20.00 © 2006 IEEE

IEEE COMPUTER SOCIETY

2.1. Expanding task graph

On one hand, the local system probably messages other system to trigger the processing of some sequential tasks, after processing a task. On the other hand, the local system may need to wait for the completion of some preceding tasks in other systems, before processing a task.

To describe this, an expanding task graph is presented, to which the deferrable relationship is added. In the expanding task graph, two tasks, which have deferrable relationship, are scheduled in two different systems. As to a couple of deferrable tasks, the local task is called, by the local system, as a deferrable local task, while the other task is called, by the local system, as a deferrable exterior task. As shown in Fig.1, the relationship between T_{13} and T_{22} is a deferrable relationship, which is labeled with character D and dashed. T_{13} is called as a deferrable local task by system 1, while T_{22} is called as a deferrable exterior task by system 1.

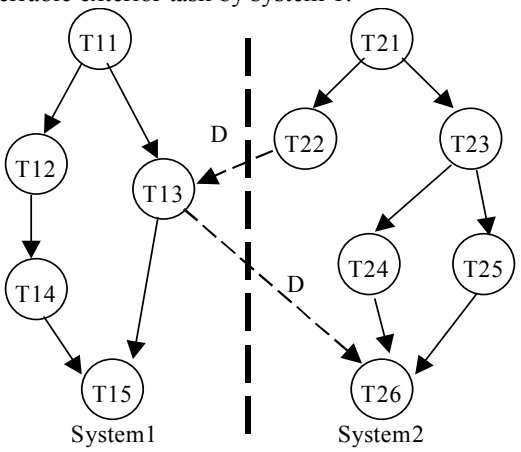

Figure 1. An expanding task graph

A pre-processing algorithm is needed, in which each deferrable exterior task is replaced by a virtual task. Because the deferrable exterior task may take uncertain execution time, which may be a long time, the time of the virtual task is set as the maximum time of all local tasks.

2.2. ECG scheduling algorithm

Let T={T1, T2, …, Tn} be a set of n tasks to be executed on p processors, and let $\prec$ be a partial order on T, which specifies some tasks must complete before other tasks begin. If Ti $\prec$ Tj, then task Ti is an immediate predecessor of task Tj, and task Tj is an immediate successor of task Ti.

Define 1: S(Ti) ={Tj| Tj $\prec$ Ti} denote the set of immediate successor of Ti.

Define 2: $N_C(Ti)=time(Tx)+N_C(Tx)$, where Tx is the task in S(Ti) such that $N_C(Tx)$ is greater than $N_C(T)$ for all T in S(Ti).

Define 3: $N_{EC}(Ti)=time(Ti)+N_{EC}(Ty)$, where Ty is the task in S(Ti) such that $N_{EC}(Tx)$ is greater than $N_{EC}(T)$ for all T in S(Ti).

$N_C(Ti)$ can be applied to calculate N(T) for the Coffman-Graham scheduling algorithm, while $N_{EC}(Ti)$ is used in the ECG scheduling algorithm.

Let α(Ti) be an integer label assigned to Ti, and according to the define 3, $N_{EC}(Ti)$ can be calculated by $N_{EC}(Ti)=time(Ti)+ \max\{N_{EC}(Tj)|\ Tj \prec Ti\}$. The ECG scheduling algorithm is shown as follows.

1)Let R be the set of tasks without successors.

2)For i ←1 to n do

a)Let T* be the task in R such that $N_{EC}(T^*)$ is smaller than N_{EC} (T) for all T in R.

b)If there are several such tasks, select virtual tasks firstly, and select arbitrary one secondly.

c)α(T*)←i.

d)Let R be the set of unlabeled tasks without unlabeled successors.

End for

3)Construct a list of tasks $L'=(u_n,u_{n-1},\cdots,u_2,u_1)$ such that α(ui)=i, where 1≤i≤n.

4)Construct a list of tasks L=L', and then remove all virtual tasks from L.

5)Given (T, $\prec$, L), use the Graham's list scheduling algorithm to schedule the tasks in L.

3. The analysis of parallel effect

Theorem 1: As to a task graph without deferrable relationship, the scheduling effect of the expanding Coffman-Graham algorithm is the same as that of the Coffman-Graham algorithm.

Corollary 1: As to an expanding task graph with deferrable relationship, the scheduling effect of the expanding Coffman-Graham algorithm is the same as that of the Coffman-Graham algorithm.

Lemma 1 [14]: The Coffman-Graham algorithm generates an optimal schedule if the number of processors is two.

Corollary 2: The expanding Coffman-Graham algorithm generates an optimal schedule if the number of processors is two.

3.1. Proof of the theorem 1

Because the Coffman-Graham scheduling algorithm is only applied to unit-time tasks, it needs to transform a task graph to the unit-time task graph firstly. Suppose Ti is a random task and its execution time is m.

IEEE COMPUTER SOCIETY

Construct a task sequence, which consists of m unit-time tasks, and then replace the task Ti with the task sequence in the graph. As shown in Fig.2, Ti and Tj, which are two random tasks in an original task graph, are transformed to two task sequences in its unit-time transformed task graph.

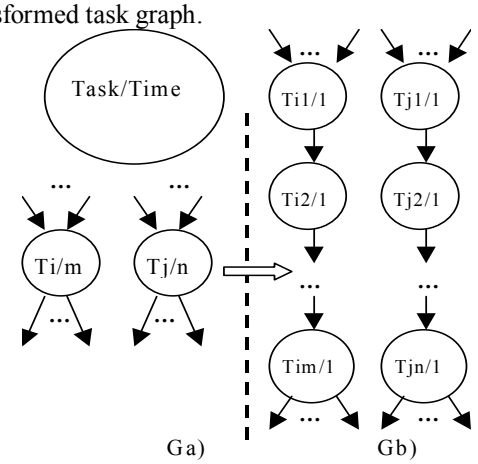

Figure 2**. Unit-time transformation of a task graph**

The original task graph Ga is scheduled with ECG algorithm, while its unit-time transformed task graph Gb is scheduled with the Coffman-Graham algorithm. Suppose two schedules are denoted as L_{EC} and L_C respectively, and define L(Ti) as the position of the task Ti in the schedule L. According to define 2, 3, $N_{EC}(Ti)$ can be calculated by $N_{EC}(Ti)=time(Ti)+\max\{N_{EC}(Tj)|\ Tj \prec Ti\}$ while $N_C(Ti)$ can be calculated by $N_C(Ti)=\max\{N_C(Tj)|\ Tj \prec Ti\}+time(Tj)$.

The theorem 1 is proved as foolows.

1) $\because N_{EC}(Ti1)= N_{EC}(Tim)+(m-1)$,
$N_{EC}(Tim)= N_C(Tim)+1$
$\therefore N_{EC}(Ti1)= N_C(Tim)+m$ (1).
$\because N_{EC}(Ti)= N_C(Ti)+m$, $N_C(Ti)= N_C(Tim)$
$\therefore N_{EC}(Ti)= N_C(Tim)+m$ (2).
Substitute (1) into (2), so that
$N_{EC}(Ti)= N_{EC}(Ti1)$ (3).
Similarly it is obtained that
$N_{EC}(Tj)= N_{EC}(Tj1)$ (4).
2) if $L_C(Ti1)< L_C(Tj1)$, then $N_C(Ti1)> N_C(Tj1)$.
$\because N_C(Ti1)= N_{EC}(Ti2)$, $N_C(Tj1)= N_{EC}(Tj2)$
$\therefore N_{EC}(Ti2)> N_{EC}(Tj2)$.
$\therefore N_{EC}(Ti2)+1> N_{EC}(Tj2)+1$,
That is $N_{EC}(Ti1)> N_{EC}(Tj1)$ (5).
Substitute (3) and (4) into (5), so that
$N_{EC}(Ti)> N_{EC}(Tj)$,
So $L_{EC}(Ti)< L_{EC}(Tj)$.

If $L_C(Ti1)> L_C(Tj1)$, it is similarly deduced that $L_{EC}(Ti)> L_{EC}(Tj)$. Hence the position order between Ti1 and Tj1 in L_C, is consistent with that order between Ti and Tj in L_{EC}. Ti1 and Tj1 in the unit-time transformed task graph correspond to Ti and Tj in the original task graph respectively. Therefore, the ECG scheduling algorithm has the same scheduling effect as the Coffman-Graham scheduling algorithm for a task graph without deferrable relationship.

The proof of the theorem 1 is now completed.

3.2. Proof of the corollary 1

As to an expanding task graph with deferrable relationship, its local expanding task graph can be obtained with the pre-processing algorithm. As a result, there may be several end-points in the graph, such as a virtual task that is the immediate successor of the deferrable local task, and then the partial order between the deferrable local task and another local task is uncertain.

Construct a virtual end-point task, whose execution time is unit-time, and let it act as the immediate successor of all end-point tasks. Now the graph has a single end-point, and the partial order between the deferrable local task and another local task is certain. Hence the deferrable relationship can be removed.

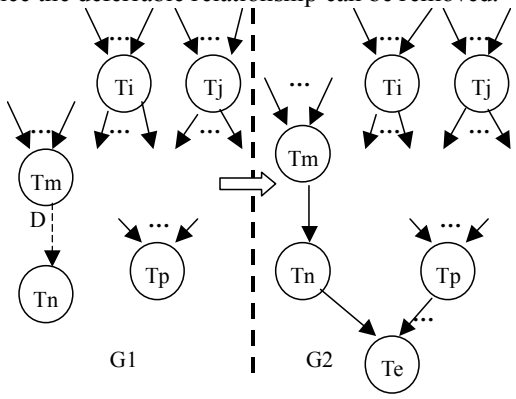

Figure 3. **Deferral removing transformation of a task graph**

As shown in Fig.3, task Te is a virtual end-point task. Suppose the task graph with deferrable relationship is named as G1, and its transformed graph is named as G2. Let Ti and Tj be two random tasks, which are in both G1 and G2, and neither of them is task Te. Applied to the Expanding Coffman-Graham scheduling algorithm, schedule $L1_{EC}$ and schedule $L2_{EC}$ can be obtained from G1 and G2 respectively.

Moreover, $N1_{EC}(Ti)$ is defined as $N_{EC}(Ti)$ for G1, while $N2_{EC}(Ti)$ denotes $N_{EC}(Ti)$ for G2, where $N_{EC}(Ti)$ is calculated by $N_{EC}(Ti)=time(Ti)+\max\{N_{EC}(Tj)|\ Tj \prec Ti\}$.

The corollary 1 is proved as foolows.

If $L2_{EC}(Ti)<L2_{EC}(Tj)$, then $N2_{EC}(Ti)>N2_{EC}(Tj)$.
$\because N2_{EC}(Ti) =N1_{EC}(Ti)+1$, $N2_{EC}(Tj) =N1_{EC}(Tj)+1$.
$\therefore N1_{EC}(Ti)>N1_{EC}(Tj)$.
So that $L1_{EC}(Ti)<L1_{EC}(Tj)$.

Proceedings of the Seventh International Conference on
Parallel and Distributed Computing,Applications and Technologies (PDCAT'06)
0-7695-2736-1/06 $20.00 © 2006 IEEE
IEEE COMPUTER SOCIETY

If $L2_{EC}(Ti)>L2_{EC}(Tj)$, it is similarly obtained that $L1_{EC}(Ti)>L1_{EC}(Tj)$. Hence the position order between Ti and Tj in $L1_{EC}$, is consistent with that in $L2_{EC}$, which indicates that the Expanding Coffman-Graham algorithm has the same scheduling effect in G1 as that in G2.

According to theorem1, as to G2, the scheduling effect of the ECG scheduling algorithm is the same as that of the Coffman-Graham scheduling algorithm. Therefore, as to G1, which has deferrable relationship, the scheduling effect of ECG scheduling algorithm is the same as that of the Coffman-Graham scheduling algorithm.

The proof of the corollary 1 is now completed.

3.3. Proof of the corollary 2

According to the corollary 2, the ECG algorithm has the same scheduling effect as the Coffman-Graham algorithm does. In addition, it is known from the lemma 1 that the Coffman-Graham algorithm generates an optimal schedule if the number of processors is two. Hence, the expanding Coffman-Graham algorithm generates an optimal schedule if the number of processors is two.

The proof of the corollary 2 is now completed.

In addition, the ECG algorithm has been applied to simulated schedules with many task graphs, and the scheduling results can show the corollary 2 is right.

4. Conclusion

The ECG scheduling algorithm, which is proposed in this paper, orients to the processing of the distributed systems. And it is proved to be an optimal schedule when the number of processors in a computer is two

5. References

[1]M. Zhang, and Z. Lu, "A Fuzzy Classification Based on Feature Selection for Web Pages", 2004 IEEE/WIC/ACM International Conference on Web intelligence, IEEE Computer Society Press, 2004, pp. 469-472.

[2]M. Zhang, Z. Lu, and C. Zou, "A Chinese Word Segmentation Based on Language Situation in Processing Ambiguous Words", *Information Sciences*, Elsevier Science, 2004, 162 (3-4), pp. 275-285.

[3]G.C. Anagnostopoulos, and G.A. Rabadi, "Simulated annealing algorithm for the unrelated parallel machine scheduling problem", Proceedings of the 5th Biannual World Automation Congress, 2002, pp.115 –120.

[4]V. Sarkar, *Partitioning and Scheduling Parallel Programs for Multiprocessors*, the MIT Press, 1989.

[5]D. Lifka, "The ANL/IBM SP scheduling system", Proc.of Int'l Parallel and Distributed Processing Symp., Workshop Job Scheduling Strategies for Parallel Processing, 1995, pp. 295-303.

[6]E.Shmueli, and D.G. Feitelson, "Backfilling with lookahead to optimize the packing of parallel jobs", *Journal of Parallel and Distributed Computing*, Elsevier Science, 2005, 65(9), pp.1090-1107.

[7]D. Talby, and D.G. Feitelson, "Improving and Stabilizing Parallel Computer Performance Using Adaptive Backfilling", 19th IEEE International Symposium on Parallel and Distributed Processing, IEEE Computer Society Press, 2005, pp.84 - 84.

[8]J.E. Moreira, H. Franke, W. Chan, L.L. Fong, M.A. Jette, and A. Yoo, "A gang-scheduling system for ASCI blue-pacific", Seventh Int'l Conf on High-Performance Computing and Networking, 1999, pp.831-840.

[9]H.D. Karatza, "Gang scheduling in a distributed system under processor failures and time-varying gang size", The Ninth IEEE Workshop on Distributed Computing Systems, IEEE Computer Society Press, 2003, pp.330 - 336.

[10]Y. Wiseman, and D.G. Feitelson, "Paired gang scheduling", *IEEE Transactions on Parallel and Distributed Systems*, IEEE Computer Society Press, 2003, 14(6), pp. 581 - 592.

[11]K.D. Ryu, N. Pachapurkar, and L.L. Fong, "Adaptive memory paging for efficient gang scheduling of parallel applications", 2004 Parallel and Distributed Processing Symposium, 2004, pp.30-40.

[12]Y. Zhang, H. Franke, J. Moreira, A. Sivasubramaniam, "An integrated approach to parallel scheduling using gang-scheduling, backfilling, and migration", IEEE Transactions on Parallel and Distributed Systems, 2003, 14(3), pp.236-247.

[13]A. Swiecicka, F. Seredynski, "Applying cellular automata in multiprocessor scheduling", Proceedings of 2002 International Conference on Parallel Computing in Electrical Engineering, 2002, pp. 177-182.

[14]M.J. Quinn, *Parallel Computing: Theory and Practice*, McGraw Hill Book Company, New York, 1994.