Vol. 10 No. 2 June 2003
Article ID: 1005-9784(2003)02-0140-05

J. CENT. SOUTH UNIV. TECHNOL.

EZDCP: A new static task scheduling algorithm with edge-zeroing based on dynamic critical paths

CHEN Zhi-gang(陈志刚), HUA Qiang-sheng(华强胜)
(College of Information Science&Engineering, Central South University, Changsha 410083, China)

Abstract: A new static task scheduling algorithm named edge-zeroing based on dynamic critical paths is proposed. The main ideas of the algorithm are as follows: firstly suppose that all of the tasks are in different clusters; secondly, select one of the critical paths of the partially clustered directed acyclic graph; thirdly, try to zero one of graph communication edges; fourthly, repeat above three processes until all edges are zeroed; finally, check the generated clusters to see if some of them can be further merged without increasing the parallel time. Comparisons of the previous algorithms with edge-zeroing based on dynamic critical paths show that the new algorithm has not only a low complexity but also a desired performance comparable or even better on average to much higher complexity heuristic algorithms.

Key words: EZDCP; directed acyclic graph; dynamic critical path; task scheduling algorithm
CLC number: TP316.4 **Document code**: A

1 INTRODUCTION

The efficient execution of a program on a parallel and distributed system highly depends on the methods taken for scheduling the tasks represented by a directed acyclic graph onto a multiprocessor system. Aiming to achieve better performance by using these systems, lots of scheduling algorithms are used, including branch-and-bound, graph-theory, randomization, genetic algorithms and evolutionary methods[1]. The objectives of these scheduling algorithms are to allocate tasks onto processors and to determine the order of their execution so that data dependencies are satisfied and the length of the produced schedule are minimized. However, it is proved that such scheduling problem is NP-complete[2]. Hence, heuristic approach of task scheduling is becoming an active research topic until now and many heuristic scheduling algorithms have been proposed, such as EZ, LC, DSC and DCP[3-6]. However, most of these algorithms have their one drawback or the other which lead to poor performance. In this paper, a new static task schduling algorithm with edge-zeroing based on dynamic critical paths EZDCP is proposed, which provides better performance than the previous algorithms in many ways and still posses a low complexity.

2 SOME DEFINITIONS

Definition 1 A weighted directed acyclic graph(DAG) is a tuple $G=(V,E,W,C)$, where $V=(V_1,V_2,\cdots,V_{|v|})$ is the set of nodes and $|V|$ is the number of the nodes; $E=\{e_{ij}\mid v_i,v_j\in V\}\subseteq V\times V$, is the set of communication edges and $|E|$ is the number of the edges. The set C is the set of edge communication cost and W is the set of node computation cost. The value $C_{ij}\in C$ is the communication cost incurred along the edge $e_{ij}\in E$, which is zero if both nodes are mapped in the same processor. The value $W_i\in W$ is the computation cost for node $V_i\in V$. Call $PRED(V_i)$ the set of immediate predecessors of V_i and $SUCC(V_i)$ the set of immediate successors of V_i. If $PRED(V_i)=\phi$, then node V_i is an entry node, and symmetrically if $SUCC(V_i)=\phi$, then node V_i is an exit node. Two nodes are called independent if there are no dependence paths between them.

Definition 2 A dynamic critical path(DCP) is the current critical path of the partially clustered DAG. Note that the communication cost between the nodes in a cluster is zero and it may contain the independent tasks. The scheduling length of a

Foundation item: Foundation for University Key Teachers by the Ministry of Education(No. 2002-143)
Biography of the first author: CHEN Zhi-gang, PhD, professor, born in May 1964, majoring in network computing and database technology.
Received date: June 13, 2002

DAG is determined by the DCP. The length of the scheduled graph(parallel time which is abbreviated as PT) is determined with the following formula:

$$PT=\max_{v_i \in V}\{t\text{-}level(V_i)+b\text{-}level(V_i)\}.$$

If all edges on the DCP are examined, then a sub-DCP is defined, and it refers to the longest path of the DAG which has at least one unexamined edge.

Definition 3 $t\text{-}level(V_i)$ is the length of the longest path from an entry node to V_i excluding the computation cost of V_i in a DAG. $b\text{-}level(V_i)$ is the length of the longest path from V_i to an exit node. Because the cost of the communication edge on the path may be zeroed during the clustering, both $t\text{-}level(V_i)$ and $b\text{-}level(V_i)$ are dynamically changed. If the calculation of $b\text{-}level(V_i)$ doesnot take the communication cost into account, then it is called static $b\text{-}level$ of task which is abbreviated as sbl.

3 EZDCP ALGORITHM

As discussed in the introduction, both the EZ algorithm and LC algorithm have their disadvantages. And generally speaking, the parallel time is determined by the DCP of the scheduled DAG. Note that the DCP here may include the independent nodes. It is because they belong to the same cluster and the tasks have to be executed on one processor sequentially. For example, in Fig. 1(a), if (n, n_3) has been zeroed, and next choose (n_1, n_2) to zero, then the DCP now is not (n, n_2, n_4) but (n_1, n_3, n_2, n_4) and the PT is 44. Thus if the algorithm can guarantee to make the largest reduction in the length of the DCP at each step under some constraints, it is possible to minimize the parallel time in the end. In a word, the goal of the proposed algorithm is to make the best effort to reduce the length of the DCP and to reduce the numbers of the DCP at each step.

3.1 Description of the EZDCP

Based on the ideas put forward above, the EZDCP algorithm is given below:

1) Initially all edges are unexamined.

Repeat

2) Traverse the partially scheduled DAG to find out one of the DCP which has at least one unexamined edge, otherwise, find out a sub-DCP which also has at least one unexamined edge.

3) Sort the edges of the DCP in a descending order of edge weights.

4) Pick an unexamined edge on the selected DCP which has the largest edge weight. If there are more than one of this kind of edges, then select the higher level edge, that is, if $C(V_i, V_j)=C(V_k, V_l)$ and $t\text{-}level(V_i)<t\text{-}level(V_k)$ then select the edge e_{ij}. Mark the edge as examined. Zero the highest edge weight if this zeroing can satisfy the following constraints, that is, if this zeroing will not produce another different critical path whose length is equal to or even larger than the current critical paths. Merge the two clusters. Repeat this step until all the edges on the DCP are examined or if the edge zeroing is successful.

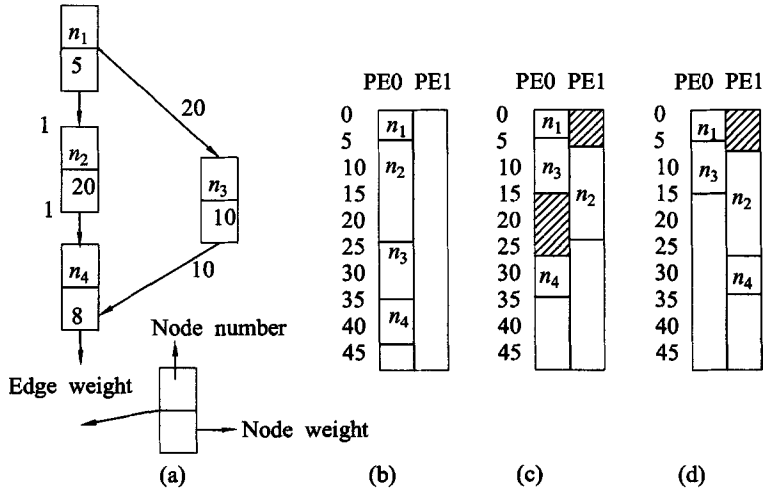
Until all of the edges on the DAG are examined.

5) Check the different clusters to see if some of them can be further merged without increasing the parallel time so that the number of processors used can be minimized.

In this algorithm, the constraints should be highlighted. It firstly ensures that the parallel time will not be increased at each step. Secondly, if the parallel time of scheduled DAG is equal to the current length of the DCP, then the parallel time will be only determined by the current DCP but not the new generated DCP, otherwise, the parallel time will be reduced to the greatest extent at each step.

3.2 Merging clusters

Suppose the DAG be divided into p clusters $(C_1, C_2, \cdots, C_p)$. $t\text{-}level(C_i)$ is the $t\text{-}level$ value of V_i which has the smallest $t\text{-}level$ value in C_i. $W(C_i)$ is the sum the $W(V_i)$. $b\text{-}level(C_i)$ is the $b\text{-}level$ value of V_i which has the largest $b\text{-}level$ value in cluster C_i. Suppose $t\text{-}level(C_i)\leqslant t\text{-}level(C_j)$, if $t\text{-}level(C_i)+W(C_i)+b\text{-}level(C_j)\leqslant PT$, then C_i and C_j can be further merged. For example, in Table 2, at the 10th step, the task n_3 and n_5 are originally in different clusters. At the 11th step, check if n_3 and n_5 can be merged together. Because $t\text{-}level(n_3)+W(n_3)+b\text{-}level(n_5)=3+3+5=11<17$ (PT), so n_3 and n_5 can be further merged. Thus the processors used are reduced from 4 to 3, and the efficiency of the algorithm is greatly improved.



(a)—a DAG; (b)—scheduling length (43); (c)—scheduling length (35); (d)—scheduling length (34)

Fig. 1 Different gantt graphs generated by different algorithms

4 PROPERTIES AND PERFORMANCE OF EZDCP

4.1 EZDCP properties

Theorem 1 For each step of EZDCP, $PT_{i+1} \leqslant PT_i$, if $PT_{i+1} = PT_i$ then there must exist more than one of this kind of DCP or this zeroing is not successful.

Proof Suppose at step i, the parallel time is PT_i, (1) If there is only one DCP, then if the incoming zeroing makes the parallel time increase or if this zeroing makes another DCP , this zeroing is not successful according to the predefined constraints, so the $PT_{i+1} = PT_i$, otherwise $PT_{i+1} < PT_i$; (2) If there are more than one DCP, then if the incoming zeroing is successful, $PT_{i+1} = PT_i$, otherwise PT_{i+1} also equals to PT_i. So with the above mentioned, the proof is completed.

Theorem 2 The time complexity of EZDCP is $O(|E| * (|E|+|V|))$ and the space complexity is $O(|E|+|V|)$.

Proof For each step i, the algorithm needs to find the current DCP, so it has to traverse the scheduled DAG in $O(|E|+|V|)$ time. Since there are almost $|E|$ steps, the overall time complexity is $O(|E| * (|E|+|V|))$. Because the space needed for EZDCP is to store the DAG, i. e. to store the $|V|$ tasks and the $|E|$ edges, the space complexity is $O(|E|+|V|)$.

4.2 Performance comparisons

4.2.1 EZ algorithm

For the DAG shown in Fig. 1(a), the EZ algorithm is shown in Fig. 1(c). It is apparent from this example that the criterion used by the EZ algorithm to select node for scheduling does not properly identify the most important node at each scheduling step. Furthermore, the ordering of the node execution is based on the *sbl*, which cannot guarantee the optimal schedule at each step. After the scheduling of n_1 and n_3, the highest cost edge is (n_3, n_4). Thus, n_4 is scheduled to PE0. However, n_2 cannot be scheduled to PE0 afterwards so as not to increase the schedule length. The result is an inefficient schedule compared with the result of Fig. 1(d) which generates a length of 34.

4.2.2 LC algorithm

For the DAG shown in Fig. 1(a), the LC algorithm generates the schedule shown in Fig. 1(c). The result is also an inefficient schedule compared with the result of Fig. 1(d). In fact, if there are some paths whose lengths are similar to the length of the CP, the result will be worse than other algorithms generated including EZ.

4.2.3 Other scheduling algorithms

There are many other scheduling algorithms besides the EZ and LC, of which the DSC[5] algorithm and the DCP[6] algorithm are well-known. The DCP algorithm can produce a shorter schedule length than the EZDCP, but it is very complex and it is at the cost of its complexity. Other algorithms such as HLFET, MCP, ETF, DLS and MD are described in Ref. [3]. Fig. 1(b) shows that these algorithms generate a much longer scheduling length than the proposed algorithm EZDCP.

5 EXPERIMENTAL RESULTS

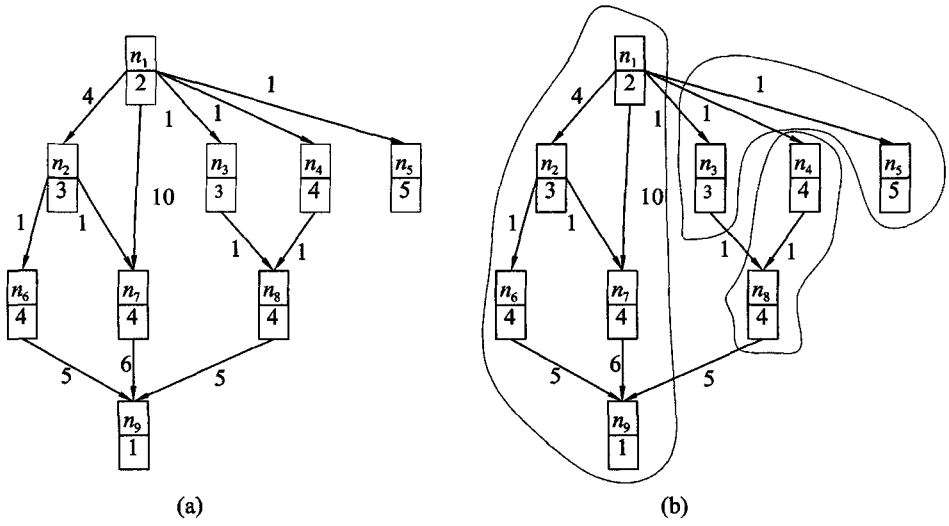
A given DAG is shown in Fig. 2(a). Because all directed acyclic graphs can be decomposed into forks and joins, and they are very effective in pro-

ving whether a task scheduling algorithm is efficient or not, the DAG can satisfy our needs[3,7-10].

Table 1 and Table 2 show the detailed steps of scheduling the DAG onto the multiprocessors with EZ and EZDCP algorithms respectively. By using the EZ algorithm, the DAG is divided into 4 clusters, $\{(n_1, n_2, n_4, n_6, n_7, n_9), (n_3), (n_5), (n_8)\}$. But if select the edge$(n_1, n_3)$ at step 6 instead of edge (n_1, n_4), then the algorithm will yield to another clusters $\{(n_1, n_2, n_3, n_6, n_7, n_9), (n_4, n_8), (n_5)\}$. It is a better clustering for this DAG, but the EZ algorithm cannot guarantee the better division. The EZDCP can guarantee the optimal selection via the dynamic critical path, because it guarantees to reduce the length of the DCP and the number of the DCP at each step under the given constraints discussed above. As shown in Fig. 3(c), the scheduling length of the proposed algorithm is 17 and it divides the DAG into 3 clusters $\{(n_1, n_2, n_6, n_7, n_9), (n_4, n_8), (n_3, n_5)\}$.

Table 1 Detailed steps of the given DAG clustering generated by the EZ algorithm

Step	Edge examined	PT if zeroed	Zeroing	PT_i
1	(n_1, n_7)	21	yes	21
2	(n_7, n_9)	20	yes	20
3	(n_6, n_9)	19	yes	19
4	(n_8, n_9)	22	no	19
5	(n_1, n_2)	18	yes	18
6	(n_1, n_4)	18	yes	18
7	(n_4, n_8)	22	no	18
8	(n_1, n_3)	21	no	18
9	(n_3, n_8)	20	no	18
10	(n_2, n_5)	23	no	18
11	(n_2, n_6)	18	yes	18
12	(n_2, n_7)	18	yes	18



(a)—a typical DAG; (b)—three clusters generated by EZDCP

Fig. 2 DAG clustering generated by the EZDCP algorithm

Table 2 Detailed steps of the given DAG clustering generated by the EZDCP algorithm

Step	DCP or sub-DCP which hs at least one unexamined edge	Edge examined	PT if zeroed	Zeroing	PT_i
1	(n_1, n_7, n_9)	(n_1, n_7)	21	yes	21
2	(n_1, n_2, n_7, n_9)	(n_7, n_9)	20	yes	20
3	(n_1, n_2, n_6, n_9)	(n_6, n_9)	19	yes	19
4	$(n_1, n_2, n_6, n7, n_9)$	(n_1, n_2)	18	yes	18
5	(n_1, n_4, n_8, n_9)	(n_8, n_9)	18	no	18
6	(n_1, n_4, n_8, n_9)	(n_1, n_4)	18	no	18
7	(n_1, n_4, n_8, n_9)	(n_4, n_8)	17	yes	17
8	(n_1, n_3, n_8, n_9)	(n_1, n_3)	17	no	17
9	(n_1, n_3, n_8, n_9)	(n_3, n_8)	20	no	17
10	(n_1, n_5)	(n_1, n_5)	19	no	17
11		(n_3, n_5)	17	yes	17

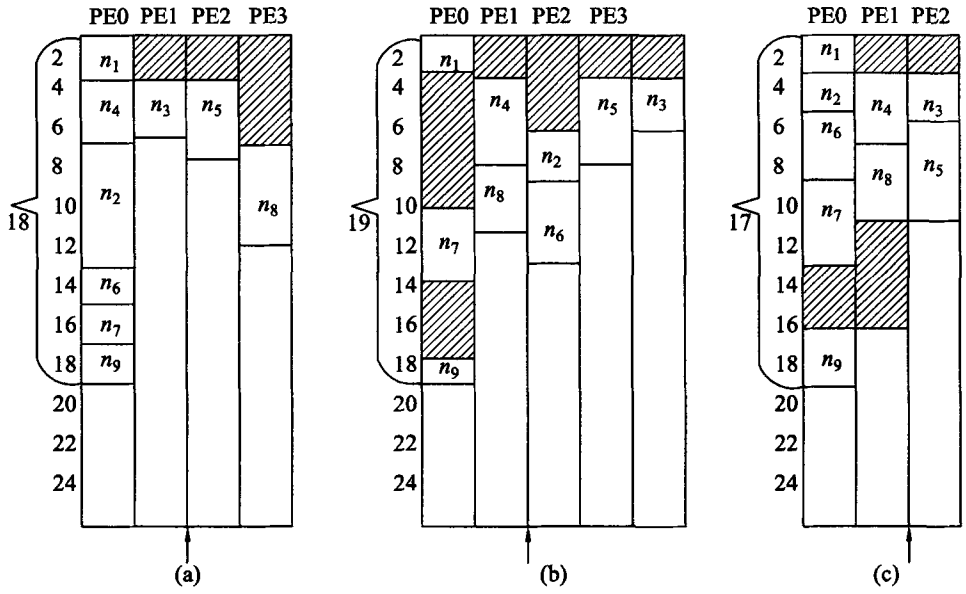
6 CONCLUSIONS

1) The EZDCP algorithm guarantees to reduce the length of the dynamic critical path at each step unless there are more than one DCP at that time.

2) The new algorithm determines the order of the execution time of the task based on its *t-level* and *b-level* values together other than its static *b-level* or exclusive *t-level* or *b-level* values.

3) The new algorithm can get better scheduling results than other algorithms which have the similar or even higher complexity. For example, though the MD algorithm has a time complexity $O(|V|^3)$, it still leads to a poor performance, though the EZ algorithm has the same complexity with EZDCP, its scheduling length is longer than EZDCP and it can't guarantee to minimize the processors either.

4) The new algorithm can minimize the processors without increasing the scheduling length, so the EZDCP improves the efficiency greatly.



(a)—EZ algorithm (18) 4 processors; (b)—LC algorithm (19) 5 processors; (c)—EZDCP algorithm (17) 3 processors

Fig. 3 Different gantt graphs generated by three scheduling algorithm

REFERENCES

[1] ZHONG Qiu-xi, XIE Tao. Task allocation and scheduling by computational model of coevolution[J]. Journal of Computer(in Chinese),2001,24(3):308-313.

[2] Coffman E G. Computer and job-shop scheduling theory[M]. New York: John Willey & Sons Publication, 1976.

[3] Kwok Yu-kwong, Ahmad I. Static scheduling algorithms for allocating directed task graphs to multiprocessors[J]. ACM Computing Surveys, 1999, 31(4): 406-471.

[4] Sarkar V. Partitioning and scheduling parallel programs for multiprocessors[M]. Cambridge: MIT Press,1989.

[5] Yang Tao, Gerasoulis A. Dsc: Scheduling parallel tasks on an unbounded number of processors[J]. IEEE Transactions on Parallel and Distributed Systems,1994,5(9): 951-967.

[6] Kwok Yu-kwong, Ahmad I. Dynamic critical-path scheduling: An effective technique for allocating task graphs to multiprocessors[J]. IEEE Transactions on Parallel and Distributed Systems, 1996,7(5):506-521.

[7] Gerasoulis A, Yang Tao. On the granularity and clustering of directed acyclic task graphs[J]. IEEE Transactions on Parallel and Distributed Systems, 1993,4(6): 686-701.

[8] YANG Tao, Gerasoulis Apostolos. Pyrros: Static task scheduling and code generation for message passing multiprocessors[A]. Proceedings of 6th ACM International Conference on Supercomputing(in Chinese)[C]. Washington D. C. :ACM Press ,1992.

[9] SHI Wei, ZHENG Wei-min. The balanced dynamic critical path scheduling algorithm of dependent task graphs [J]. Journal of Computer,2001,24(9):991-997.

[10] LIU Zhen-yin, FANG Bin-xing. Tsaot: An algorithm scheduling an outtree DAG[J]. Journal of Computer(in Chinese), 2001,24(4):390-394.