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

CHEN Zhi-gang, HUA Qiang-sheng
(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

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. 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. 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. However, most of these algorithms have their one drawback or the other which lead to poor performance. In this paper, a new static task scheduling 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 $V = (V_1, V_2, \ldots, V_{|V|})$ is the set of nodes and $|V|$ is the number of the nodes; $E = \{e_{ij} | v_i, v_j \in V\} \subseteq V \times V$, 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 $PRED(V_i) = \emptyset$, then node $V_i$ is an entry node, and symmetrically if $SUCC(V_i) = \emptyset$, 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
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:

$$PT = \max \{ t\text{-level}(V_i) + b\text{-level}(V_i) \}.$$ 

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 $b\text{-level}(V_i)$ does not 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.
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 $C(V_i, V_j) = C(V_k, V_j)$ 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.

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 $(C_1, C_2, \ldots, C_p)$. $t\text{-level}(C_i)$ 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 $t\text{-level}(C_i) \leq t\text{-level}(C_j)$, if $t\text{-level}(C_i) + W(C_i) + b\text{-level}(C_j) \leq PT$, 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 $n_1$ can be merged together. Because $t\text{-level}(n_3) + W(n_3) + b\text{-level}(n_1) = 3 + 3 + 5 = 11 < 17$ ($PT$), so $n_3$ and $n_1$ can be further merged. Thus the processors used are reduced from 4 to 3, and the efficiency of the algorithm is greatly improved.