ACADEMIC
World Academic Union

ISSN 1746-7659, England, UK
Journal of Information and Computing Science
Vol. 6, No. 2, 2011, *pp*. 083-096

An Extension of Edge Zeroing Heuristic for Scheduling Precedence Constrained Task Graphs on Parallel Systems Using Cluster Dependent Priority Scheme

Abhishek Mishra and Anil Kumar Tripathi

Deptartment of Computer Engineering, Institute of Technology, Banaras Hindu University, Varanasi, India, 221005

(*Received November 3, 2010, accepted December 20, 2010*)

(*An extended abstract of this paper appears in the Proceedings of 2010 IEEE International Conference on Computer & Communication Technology (ICCCT-2010), pages 647-651, ISBN: 978-1-4244-9034-9.*)

Abstract. Sarkar's edge zeroing heuristic for scheduling precedence constrained task graphs on parallel systems can be viewed as a priority based algorithm in which the priority is assigned to edges. In this algorithm, the priority is taken as the edge weight. This can also be viewed as a task dependent priority function that is defined for pairs of tasks. We have extended this idea in which the priority is a cluster dependent function of pairs of clusters (of tasks). Using this idea we propose an algorithm of complexity $O(|V||E|(|V|+|E|))$ and compare it with some well known algorithms.

Keywords: clustering, homogeneous systems, parallel processing, scheduling, task allocation.

1. Introduction

A parallel system is designed so that it can execute the applications faster than a sequential system. For this we need to parallelize the program. There are three steps involved in the parallelization of a program (Sinnen [27]). The first step is called *task decomposition* in which the application is divided into tasks. The *degree of concurrency* is the number of tasks that can be executed simultaneously (Grama et al. [10]).

The tasks generated may have interdependencies between them that will decide the partial execution order of tasks. The determination of precedence constraints between the tasks is the second step of parallelization and is called *dependence analysis* (Banerjee et al. [2], Wolfe [29]).

A dependence relation among the tasks is represented as a directed acyclic graph, also known as the *task graph*. Nodes in the task graph represent the tasks and have a weight associated with them that represents the execution time of the task. Edges in the task graph represent the dependence relation between the tasks and have a weight associated with them that represents the communication time between the tasks.

The final step of parallelization is the *scheduling* of tasks to the processors. By scheduling we mean both the spatial assignment (task allocation), and the temporal assignment (assigning start time) of tasks to the processors.

The problem of finding a scheduling for a given task graph on a given set of processors that takes minimum time is *NP-Complete* (Sarkar [26], Papadimitriou and Yannakakis [24]). Therefore several heuristics are applied for solving this problem in polynomial time (Yang and Gerasoulis [32], Gerasoulis and Yang [8], Dikaiakos et al. [7], Kim and Browne [16], Kwok and Ahmed [17], [18], Lo [19], Malloy et al. [22], Wu and Gajski [30], Kadamuddi and Tsai [14], Sarkar [26], Yang and Gerasoulis [33], Wu et al. [31], Sih and Lee [28]). The solutions generated by using these algorithms are generally suboptimal.

Our heuristic is an extension of Sarkar's edge zeroing heuristic [26] for scheduling precedence constrained task graphs on parallel systems. Sarkar's algorithm can be viewed as a task dependent priority based algorithm in which the priority (in this case edge weight) is a function of pairs of *tasks*. We extend this concept and define the priority as a cluster dependent function of pairs of *clusters*. Using this concept we propose an algorithm of complexity $O(|V||E|(|V|+|E|))$ and compare it with some well known algorithms.

The remainder of the paper is organized in the following manner. Section 2 presents an overview of the

related literature. Section 3 defines a cluster dependent priority scheme that is dependent on cluster-pairs and also presents the proposed algorithm. Section 4 presents a detailed description of the algorithms used. Section 5 gives a sample run of the algorithm. Section 6 presents some experimental results. And finally in section 7 we conclude our work.

2. Literature Overview

Most scheduling algorithms for parallel systems in the literature are based on an idealized model of the target parallel system also referred to as the *classic model* (Sinnen [27]). It is a set of identical processors with fully connected dedicated communication subsystem. Local communications are cost-free and we also have concurrent inter-processor communications.

A fundamental scheduling heuristic is called the *list scheduling* heuristic. In list scheduling, first we assign a priority scheme to the tasks. Then we sort the tasks according to the priority scheme, while respecting the precedence constraints of the tasks. Finally each task is successively scheduled on a processor chosen for it. Some examples of list scheduling algorithms are: Adam et al. [1], Coffman and Graham [3], Graham [9], Hu [13], Kasahara and Nartia [15], Lee et al. [20], Liu et al. [21], Wu and Gajski [30], Yang and Gerasoulis [34].

Another fundamental scheduling heuristic is called *clustering*. Basically it is a scheduling technique for an unlimited number of processors. It is often proposed as an initial step in scheduling for a limited number of processors. A cluster is a set of tasks that are scheduled on the same processor. Clustering based scheduling algorithms generally consist of three steps. The first step finds a clustering of the task graph. The second step finds an allocation of clusters to the processors. The last step finds a scheduling of the tasks. Some examples of clustering based scheduling algorithms are: Mishra et al. [23], Yang and Gerasoulis [32], Kim and Browne [16], Kadamuddi and Tsai [14], Sarkar [26], Hanen and Munier [11].

3. The Cluster Dependent Priority Scheduling Algorithm

3.1. Notation

Let $\boldsymbol{N}$ denote the set of natural numbers: *{1, 2, 3, ...}*. Let $\boldsymbol{R}$ denote the set of real numbers, and let $\boldsymbol{R}^+$ denote the set of non-negative real numbers. For $(1 \leq i \leq n)$, let there be n tasks M_i. Let

$$\mathbf{M} = \{\mathrm{M_i} \mid 1 \leq \mathrm{i} \leq \mathrm{n}\} \tag{1}$$

be the set of tasks. Then for $(1 \leq i \leq n)$, the clusters $C_i \in \boldsymbol{M}$ are such that for $i \neq j$ and $(1 \leq i \leq n, 1 \leq j \leq n)$:

$$C_i \cap_{i \neq j} C_j = \phi, \tag{2}$$

and

$$\cup^{n}_{i=1} C_i = \boldsymbol{M}. \tag{3}$$

Let

$$\boldsymbol{C} = \{C_j \mid C_j \in \boldsymbol{M}, 1 \leq i \leq n\} \tag{4}$$

be a decomposition of $\boldsymbol{M}$ into clusters. Note that some of the C_j's may be empty. Let

$$V = \{i \mid 1 \leq i \leq n\} \tag{5}$$

denote the set of vertices of the task graph. Let the directed edge from i to j be denoted as $(i \rightarrow j)$. Let

$$E = \{(i, j) \mid i \in V, j \in V, \exists (i \rightarrow j)\} \tag{6}$$

denote the set of edges of the task graph. Let $m_i \in \boldsymbol{R}^+$ be the execution time of the task M_i. If $(i, j) \in E$, then let $w_{ij} \in \boldsymbol{R}^+$ be the communication time from M_i to M_j. Let T be the adjacency list representation of the task graph.

Let $cluster : \boldsymbol{N} \rightarrow \boldsymbol{N}$ be a function such that:

$$cluster(i) = j \leftrightarrow M_i \in C_j. \tag{7}$$

For $C_j \in \boldsymbol{C}$, let $comp : \boldsymbol{C} \rightarrow \boldsymbol{R}^+$ be a function that gives the total computation time of a cluster:

$$comp(C_j) = \Sigma_{Mi \in Cj} w_i. \tag{8}$$

For $C_i \in \boldsymbol{C}$, and $C_j \in \boldsymbol{C}$, let $comm : \boldsymbol{C}\ X\ \boldsymbol{C} \rightarrow \boldsymbol{R}^+$ be a function that gives the total communication time from the first cluster to the second cluster:

$$comm(C_i, C_j) = \Sigma_{Mp \in Ci, Mq \in Cj, (p, q) \in E} w_{pq}. \tag{9}$$