Using duplication for the multiprocessor scheduling problem with hierarchical communications

Evripidis Bampis[1], Rodolphe Giroudeau[1] and Jean-Claude König[2]

[1] La.M.I., C.N.R.S. EP 738, Université d'Evry Val d'Essonne, Boulevard Mitterrand, 91025 Evry Cedex, France
[2] LIRMM, 161 rue Ada, 34392 Montpellier Cedex 5, France

Abstract. We propose a two-step algorithm that efficiently constructs a schedule of minimum makespan for the precedence multiprocessor constrained scheduling problem in the presence of hierarchical communications and task duplication for UET-UCT graphs.

1 Introduction.

Many works deal with the multiprocessor scheduling problem with communication delays ([4],[5],[7]). Usually the parallel program is represented by a directed acyclic graph where each vertex corresponds to a task and each arc to a (potential) communication between a predecessor-task and a successor-task. The objective is to find a feasible schedule minimizing the completion time (makespan).

Last years the notion of hierarchical parallel architecture becomes a reality with the advance of many parallel machines which are based on this concept. Parallel architectures of this type include parallel machines constituted by different multiprocessors, bi-processors connected by myrinet switches, architectures where the processors are connected by hierarchical busses, or point-to-point architectures where each vertex of the topology is a cluster of processors. In this type of architectures, it is clear that there are two levels of communications: the intracluster-communications between two processors of a same machine, and the interprocessor-communications between two processors of different machines, that we call in what follows intercluster-communications. Hence, there is a need for an extension of the classical scheduling model with homogeneous communication delays in order to take into account this important technological evolution.

We study here the simplest case where the tasks have unit execution times, the intercluster-communication delay takes also a unit of time ($c_{ij} = 1$) while the intracluster-communication delay is equal to zero ($\epsilon_{ij} = 0$). We focus on the case where the number of clusters is unrestricted, the number of processors within each cluster is equal to two, and task duplication is allowed. Extending the standard three-field notation of [6], our problem can be denoted as $\bar{P}, 2|prec; (c_{ij}, \epsilon_{ij}) = (1,0); p_i = 1; dup|C_{max}$. This problem has already been studied in the case where the duplication of tasks is not allowed ([1],[2]). In

P. Amestoy et al. (Eds.): Euro-Par'99, LNCS 1685, pp. 369-372, 1999.

[2], the authors proved that the problem of deciding whether an instance of $\bar{P}, 2|prec; (c_{ij}, \epsilon_{ij}) = (1,0); p_i = 1|C_{max}$ has a schedule of length at most four is NP-complete. As a corollary, no polynomial-time algorithm exists with performance bound smaller than $\frac{5}{4}$ unless $P = NP$. In [1], a heuristic with a relative performance guarantee of $\frac{8}{5}$ has been proposed.

In this paper, we allow the duplication of tasks and we propose an extension of the approach proposed in [4]. Colin and Chrétienne in [4] defined an optimal algorithm for the so-called small (homogeneous) communication times case (SCT), where the minimum execution time of any task is greater than or equal to the biggest communication delay. We focus on the hierarchical communication-case and more precisely on the $\bar{P}, 2|prec; (c_{ij}, \epsilon_{ij}) = (1,0); p_i = 1; dup|C_{max}$ problem, and we propose an optimal scheduling algorithm. Our algorithm is in two steps: we first compute for each task an earliest starting time of any of its copies, we construct a critical graph, and then we build up an earliest optimal schedule. The proofs have been omitted because of space limitations.

2 Problem definition.

We consider a directed acyclic graph $G = (V; E)$ which models the precedence constraints. Given a graph G and a task i, we denote by $Pred_G(i)$ (resp. $Succ_G(i)$) the set of immediate predecessors (resp. successors) of the task i in G. More generally, we define $Pred_G^l(i) = Pred_G^{l-1}(Pred_G(i))$, where integer $l > 1$, and $Pred_G^1(i) = Pred_G(i)$. $Pred_G^*(i)$ denotes the transitive closure of $Pred_G(i)$. In the following, if no confusion arises, we will use the notation $Pred(i)$ instead of $Pred_G(i)$.

We may assume w.l.o.g. that all the copies of any task $i \in V$ start their execution at the same time, denoted t_i. Let us denote by Π_i the set of clusters on which we execute the copies of task i. In order to simplify the presentation whenever two tasks i, j (or their copies) such that $(i, j) \in E$, are not executed in consecutive time units, we consider w.l.o.g. that they are executed on different clusters. Any feasible schedule must satisfy the following constraints:

1. at any time, a cluster executes at most two copies;
2. if (i, j) is an arc of G then:

$$t_j \geq t_i + 2 \text{ if } \Pi_i \bigcap \Pi_j = \varnothing$$
$$t_i \geq t_j + 1 \text{ if } \Pi_i \bigcap \Pi_j \neq \varnothing$$

Our objective is to find a feasible schedule with a minimum makespan.

3 An optimal schedule

3.1 Finding the release times of the tasks

The algorithm works in two steps; the first step is the procedure RT (Release Times), which computes a lower bound of the starting time of any copy of task i, denoted by b_i, and a critical graph, denoted in what follows as $G_c = (V, E_c)$. The

second step of the algorithm is the procedure ES (Earliest Scheduling), which uses the critical graph to build up a schedule in which any copy of a task starts at its lower bound.

Procedure RT can be stated as follows:

Let $G_c = (V, E_c)$ be the critical graph of $G = (V, E)$. Initially, $E_c = \varnothing$.

For every task $i \in V$ without predecessor, let $b_i = 0$.

While there is a task i which has not been assigned a lower bound and whose all predecessors have been assigned their release times, do the following:

Let $b_{P(i)} = \max\{b_k \backslash k \in Pred(i)\}$, and $S_i = \{k \backslash b_k = b_{P(i)}\}$.

- If $|S_i| = 1$ then $b_i := b_{P(i)} + 1$, and if $S_i = \{k\}$, then $E_c = E_c \bigcup \{(k, i)\}$
- else if $|S_i| \geq 3$ then $b_i := b_{P(i)} + 2$;
- else if $|S_i| = 2$ then w.l.o.g., let $S_i = \{j, k\}$
  - if for every l, $|Pred^l_{G_c}(j) \bigcup Pred^l_{G_c}(k)| \leq 2$, then $b_i := b_{P(i)} + 1$, and $E_c = E_c \bigcup \{(k, i), (j, i)\}$
  - else $b_i := b_{P(i)} + 2$;

We proved in the full version of the paper [3] that RT computes the lower bounds of the starting times of any copy of each task.

Lemma 1. *Let b_i be the starting time computed by procedure RT. For any feasible schedule of G the starting time of any copy of task i is not less than b_i.*

3.2 Construction of the earliest schedule.

The second step of the algorithm consists in building up an earliest schedule, that is, a schedule such that each copy is scheduled at its lower bound.

We define an arc (i, j) of E to be a *critical arc*, if $b_i + 1 = b_j$. From this definition, it is clear that if (i, j) is a critical arc, then in any earliest schedule every copy of task j must be preceded by a copy of task i executed on the same cluster. The critical graph $G_c = (V; E_c)$ is defined as the partial graph of G induced by the critical arcs. According to the construction, it is easy to see that the following lemma is true.

Lemma 2. *Every vertex in the critical graph has at most two critical predecessors.*

In the following, for every terminal task i in the critical graph G_c (i.e. a vertex such that $Succ_{G_c}(i) = \varnothing$), we denote by C^i the *critical subgraph* of $G_c = (V; E_c)$ defined as follows:

$$C^i = \{[i], [Pred_{G_c}(i)], [Pred^2_{G_c}(i)], \dots\} = Pred^*_{G_c}(i).$$

The tasks inside the brackets ([ ]) can be (potentially) executed simultaneously. According to the construction of $G_c = (V; E_c)$, it is clear that for every i and l, we have $|Pred^l_{G_c}(i)| \leq 2$. Notice also that a critical subgraph cannot be

a proper subgraph of another critical subgraph. So, we have at most n critical subgraphs where n is the number of tasks, and in addition every task in the initial graph G is an element of at least one critical subgraph.

It is obvious that the critical subgraphs are in one-to-one correspondence with the terminal tasks of G_c, a property that will be used in the second step of the algorithm. This is done by allocating one cluster to each critical subgraph and processing all the corresponding copies at their release times.

The second step of the algorithm is the procedure ES which can be stated simply as follows:

- Assign each critical subgraph of $G_c = (V; E_c)$ to a distinct cluster;
- Start executing each copy of the tasks at its lower bound.

According to the construction, it is not difficult to verify that the obtained schedule is a feasible optimal schedule (see [3] for the proof).

Theorem 1. *The scheduling given above is feasible.*

4 Conclusion and future works.

An optimal $O(n^2)$ algorithm for the multiprocessor scheduling in the presence of hierarchical communications and task duplication has been presented. We have considered here the unit-execution-time unit-communication-time problem. We are now working to extend our study in the case of small hierarchical communication delays.

References

1. E. Bampis, R. Giroudeau, and J.C. König. A heuristic for the precedence constrained multiprocessor scheduling problem with hierarchical communications. Technical Report 35, Université d'Evry Val d'Essonne, 1999.
2. E. Bampis, R. Giroudeau, and J.C. König. On the hardness of approximationg the precedence constrained multiprocessor scheduling problem with hierarchical communications. Technical Report 34, Université d'Evry Val d'Essonne, 1999.
3. E. Bampis, R. Giroudeau, and J.C. König. Using duplication for the precedence constrained multiprocessor scheduling problem with hierarchical communications. Technical Report 36, Université d'Evry Val d'Essonne, 1999.
4. P. Chrétienne and J.Y. Colin. C.P.M. scheduling with small interprocessor communication delays. *Operations Research*, 39(3), 1991.
5. P. Chrétienne, E.J. Coffman Jr, J.K. Lenstra, and Z. Liu. *Scheduling Theory and its Applications*. Wiley, 1995.
6. R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnooy Kan. Optimization and approximation in determinics sequencing and scheduling theory : a survey. *Ann. Discrete Math.*, 5:287–326, 1979.
7. A. Munier and C. Hanen. Using duplication for scheduling unitary tasks on m processors with communication delays. *Theoretical Computer Science*, 178:119–127, 1997.