AN OPTIMAL ALGORITHM FOR SCHEDULING A COMPUTATIONAL TREE ON A PARALLEL SYSTEM WITH COMMUNICATION DELAYS

Pauline Markenscoff and Yong Yuan Li
Department of Electrical Engineering
University of Houston
Houston, Texas 77204-4793

ABSTRACT

We consider the problem of optimally scheduling the subtasks of a computational task modeled by a tree on parallel systems with identical processors. Execution of the subtasks must satisfy precedence constraints that are met via data exchanges among processors which introduce communication delays. The optimization criterion used is the minimization of the processing time and we assume that there is no restriction on the number of processors needed. We first show that the optimal scheduling problem can be solved in polynomial amount of time when the computational graph is a two-level tree. We then present an algorithm for the general tree that significantly reduces the search space and can work very fast in many cases. The number of processors needed for the optimal schedule is also computed.

1. INTRODUCTION

The optimal scheduling of the subtasks comprising a computational task to the individual processors of parallel systems with communication delays is an important design problem [1,2]. When fast real-time response is required, the optimal task scheduling must be determined by minimizing the processing time. Stone [3] used this criterion to study the allocation problem for a task with no parallelism and a non-homogeneous system of processors. In previous studies [4, 5], we considered the allocation problem for a task exhibiting parallelism. This task was modeled by a directed acyclic graph whose nodes (or subtasks) were executed on a system that had a fixed number of processors. Our approach involved the development of suboptimal algorithms using heuristics.

Here, we study the problem of optimally scheduling the subtasks of a computational task on parallel systems with identical processors, when there is no restriction in the number of processors. Execution of the subtasks must satisfy precedence constraints that are met via data exchanges among processors which introduce **communication delays.** We consider a task that can be modeled as a **tree** whose nodes correspond to the subtasks. A specific weight equal to the execution time of the corresponding subtask is assigned to each node. Communication costs are assigned to each arc and they are equal to the time required for the necessary data transmission when the two subtasks on this arc are allocated to different processors.

The communication costs are functions of the amount of data involved in a given transmission and depend on the particular parallel system. We assume, however, that the communication costs do not depend on the allocation scheme and do not vary when two tasks are allocated to different pairs of processors. Note that in some systems the communication costs will depend on the allocation scheme. In the original iPSC INTEL Hypercube implementation, for example, communication costs depended on the "distance" between processors and were smallest when the communicating processors were neighbors. This limitation was alleviated in the current iPSC-2 machines where data transmission costs are virtually independent of the "distance" between communicating processors.

We will develop the optimal algorithm for the general tree using induction. The 2-level tree is analyzed first and we show that the optimal schedule requires

$$O(N \log_2 N)$$

time. We then consider the 3-level tree case which requires a non-trivial extension of the 2-level tree results. Finally, we assume that we have computed the solution for a tree with (K-1) levels and develop the optimal algorithm for the K-level tree.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

© 1991 ACM 089791-382-5/91/0003/0289 $1.50

2. OPTIMAL SCHEDULING FOR A 2-LEVEL TREE

Figure 1 gives an example of a two-level tree. If ES_{1i} and EF_{1i} ($1 \le i \le N_1$) are the earliest starting and finish times respectively of the first level nodes, then

$$ES_{1i} = 0 \quad 1 \le i \le N_1$$
$$EF_{1i} = E_{1i} \quad 1 \le i \le N_1$$

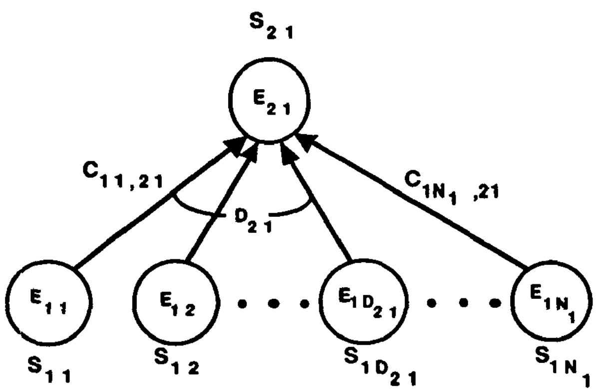


Figure 1: An example of a two-level tree

A first-level node may or may not be allocated to the processor executing node S_{21}. By not allocating a node S_{1j} to the same processor as node S_{21}, we add a communication overhead equal to $C_{1j,21}$. Also, if two first-level nodes S_{1j} and S_{1m} are not allocated to the same processor as node S_{21}, we will allocate each of these nodes to a different processor. If we allocated S_{1j} and S_{1m} to the same processor, we might delay the earliest starting time ES_{21} of node S_{21} since the execution of S_{21} would start only after both S_{1j} and S_{1m} were sequentially executed on the same processor and their results were transmitted.

With this assumption, an allocation of the two-level tree is completely defined by the boolean vector

$$\mathbf{B}_1(j) = [\, b_{11}, b_{12}, \ldots, b_{1N_1} \,]$$

where

$b_{1i} = 1$ if first-level node S_{1i} is allocated to the same processor as node S_{21} ($1 \le i \le N_1$).

and

$b_{1i} = 0$ if first-level node S_{1i} is **not** allocated to the same processor as node S_{21} ($1 \le i \le N_1$).

The cardinality of the set of all possible allocations is 2^{N-1}, that is $0 \le j \le 2^{N_1} - 1 \equiv 2^{N-1} - 1$.

An allocation $\mathbf{B}_1(j)$ can also be uniquely defined by a vector $\mathbf{V}_1(j)$ whose components are the locations of the 1's of the boolean vector $\mathbf{B}_1(j)$. For example, if

$$B_1(j_1) = [\; 1 \;\ldots\; \underset{\substack{\uparrow\\ k-1}}{1} \; \underset{\substack{\uparrow\\ k}}{0} \;\ldots\; 0 \; \underset{\substack{\uparrow\\ m_1}}{1} \; 0 \;\ldots\; 0 \; \underset{\substack{\uparrow\\ m_s}}{1} \; 0 \;\ldots\;]$$

then

$$V_1(j_1) = [\, 11, 12, \ldots, 1(k\text{-}1), 1m_1, 1m_2, \ldots, 1m_s \,]$$

The two vectors $\mathbf{B}_1(j)$ and $\mathbf{V}_1(j)$ are **equivalent** and fully interchangeable descriptions of the same allocation.

Each allocation $\mathbf{B}_1(j) = [\, b_{11}, b_{12}, \ldots, b_{1N_1} \,]$ defines a schedule.

- $\mathbf{B}_1$ defines all the first level nodes S_{1j} that must be executed on the same processor with node S_{21} (these are all the nodes for which $b_{1j} = 1$). All these nodes S_{1j} can be executed in any order before node S_{21}.
- Each of the other nodes (for which $b_{1j} = 0$) is executed on a separate processor and the results are transmitted before the execution of S_{21} can start.

For each allocation $\mathbf{B}_1$ we can compute an earliest starting time for node S_{21} that we will denote by $ES_{21}(\mathbf{B}_1)$ or $ES_{21}(\mathbf{V}_1)$. Similarly, the earliest finish times $EF_{21}(\mathbf{B}_1)$ or $EF_{21}(\mathbf{V}_1)$ can be defined for the specific allocation. Note that $EF_{21}(\mathbf{B}_1) = ES_{21}(\mathbf{B}_1) + E_{21}$. We can now define the earliest starting time for S_{21} over all possible allocations of its predecessors by

$$ES_{21} = \min \{ES_{21}(\mathbf{B}_1(j))\} \quad \forall\, j = 0,1,2,\ldots,2^{N_1} - 1$$

Similarly

$$EF_{21} = \min \{EF_{21}(\mathbf{B}_1(j))\} \quad \forall\, j = 0,1,2,\ldots,2^{N_1} - 1$$

Finally, let $EF_{11,\,12,\,\ldots,\,1k}$ be the earliest finish time for the same-level nodes $S_{11}, S_{12}, \ldots, S_{1k}$ when these nodes are assigned to the same processor. For the first level nodes of a tree

$$EF_{11,\,12,\,\ldots,\,1k} = E_{11} + E_{12} + \ldots + E_{1k} \qquad (1)$$

where $k = 1, 2, \ldots, N_1$. The definition of $EF_{11,\,12,\,\ldots,\,1k}$ implies that

$$EF_{11} \le EF_{11,\,12} \le \ldots \le EF_{11,\,12,\,\ldots,\,1N_1} \qquad (2)$$

THEOREM 1: Consider a 2-level tree with N nodes and order the N_1 ($N_1 = N - 1$) first level nodes so that the following inequalities are satisfied

$$EF_{11}+C_{11,21} \geq EF_{12}+C_{12,21} \geq \dots \geq \geq EF_{1,N_1}+C_{1N_1,21} \quad (3)$$

Then:

(a) The optimal allocation (that is the allocation providing the minimum processing time) is obtained by assigning
- subtask S_{21} and D_{21} of its predecessors S_{11}, S_{12}, ..., $S_{1D_{21}}$ ($1 \leq D_{21} \leq N_1$) to one processor and
- each one of the remaining subtasks $S_{1(D_{21}+1)}$, $S_{1(D_{21}+1)}$, ..., S_{1N_1} to a separate processor.

  Nodes $S_{11}, S_{12}, \dots, S_{1D_{21}}$ will be referred to as the **critical predecessors** of node S_{21} and the optimal allocation requires N-D_{21} processors. The number D_{21} can be as small as 1 (i.e. S_{21} has only one critical predecessor) or as large as N_1. In the latter case, all nodes should be executed on the same processor.

(b) The search space is reduced to k* allocations, where $k^* \leq N_1$. Note that $k^* = D_{21}$ or $k^* = D_{21} + 1$ and k* may be small even when $N_1 \rightarrow \infty$.

(c) The cardinality k* of the search space, the number D_{21} of critical predecessors, and the optimal schedule can be determined in O(N $\log_2$N) amount of time.

Proof of Theorem 1:

CLAIM 1: The optimal schedule can be found by considering N-1 allocations.

The earliest starting time ES_{21} of node S_{21} for the allocation defined by V_1 = [11, 12, ..., 1(k-1), $1m_1$, $1m_2$, ..., $1m_j$] is given by the maximum of

(a) the earliest finish time

$$EF_{11,\ 12,\ \dots,\ 1(k-1),\ 1m1,\ 1m2,\ \dots,\ 1ms}$$

for the same-level nodes $S_{11}, S_{12}, \dots, S_{1(k-1)}$, $S_{1m1}, S_{1m2}, \dots, S_{1ms}$ when these nodes are allocated to the same processor and

(b) all the sums

$$EF_{1j} + C_{1j,21} \quad k \leq j \leq N_1, j \neq m_1, j \neq m_2, \dots, j \neq m_s$$

since the execution of node S_{21} will start only after its processor has received all the data from the predecessors allocated to different processors.

Thus

$$ES_{21}\left(11, 12, \dots, 1(k\text{-}1), 1m_1, 1m_2, \dots, 1m_j \right) = \max \left\{ EF_{11,\ 12,\ \dots,\ 1(k-1),\ 1m1,\ 1m2,\ \dots,\ 1ms},\ \max [\ EF_{1j} + C_{1j,21}\ ] \right\}$$

for $k \leq j \leq N_1$ and $j \neq m_1$, $j \neq m_2$, ..., $j \neq m_s$.

For the first level nodes $S_{11}, S_{12}, \dots, S_{1(k-1)}$, $S_{1m1}, S_{1m2}, \dots, S_{1ms}$ we have

$$EF_{11,\ 12,\ \dots,\ 1(k-1),\ 1m1,\ 1m2,\ \dots,\ 1ms} = E_{11} + E_{12} + \dots + E_{1(k-1)} + E_{1m_1} + E_{1m_2} + E_{1m_1s}$$

and from (3)

$$\max [\ EF_{1j} + C_{1j,21}\ ] = EF_{1k} + C_{1k,21}$$

for $k \leq j \leq N_1$ and $j \neq m_1$, $j \neq m_2$, ..., $j \neq m_s$. Thus,

$$\begin{aligned} ES_{21}\left(11, 12, \dots, 1(k\text{-}1), 1m_1, 1m_2, \dots, 1m_j \right) &= \\ = \max &\left\{ EF_{11,\ 12,\ \dots,\ 1(k-1),\ 1m1,\ 1m2,\ \dots,\ 1ms},\ EF_{1k} + C_{1k,21} \right\} = \\ = \max &\left\{ E_{11} + E_{12} + \dots + E_{1(k-1)} + E_{1m_1} + E_{1m_2} + E_{1m_1s},\ EF_{1k} + C_{1k,21} \right\} \end{aligned} \quad (4)$$

Consider now the allocation given by the vector

$$\begin{array}{cccccccc} B_1(j_2) = [& 1 & 1 \cdots & 1 & 0 \cdots 0 &] \\ & & & \uparrow & \uparrow & \\ & & & k\text{-}1 & k & \end{array}$$

where $b_{1i} = 0$ for $i > (k-1)$ and

$$V_1(j_2) = [\ 11, 12, \dots, 1(k\text{-}1)\ ]$$

Then $ES_{21}\left[\ V_1(j_2)\ \right] \equiv ES_{21}(\ 11, 12, \dots, 1(k\text{-}1))$ which is the earliest starting time of node S_{21} for allocation $V_1(j_2)$ is given by

$$\begin{aligned} ES_{21}(\ 11, 12, \dots, 1(k\text{-}1)\ ) &= \\ = \max &\left\{ EF_{11,\ 12,\ \dots,\ 1(k-1)},\ EF_{1k} + C_{1k,21} \right\} = \\ = \max &\left\{ E_{11} + E_{12} + \dots + E_{1(k-1)},\ EF_{1k} + C_{1k,21} \right\} \end{aligned} \quad (5)$$

By comparing (4) and (5) we obtain

$$ES_{21}\left(11, 12, \dots, 1(k\text{-}1), 1m_1, 1m_2, \dots, 1m_j \right) \geq \geq ES_{21}(\ 11, 12, \dots, 1(k\text{-}1)\ )$$

Therefore, the search space for the optimal schedule is limited to the following N_1 allocations:

$$
\begin{array}{l}
B_1(1) = [1\ 0\ 0\ \dots\ 0] \\
B_1(2) = [1\ 1\ 0\ \dots\ 0] \\
\dots\dots\dots\dots \\
B_1(k) = [\ 1\ 1 \cdots \underset{\substack{\uparrow \\ k}}{1}\ 0 \cdots 0\ ] \\
\dots\dots\dots\dots \\
B_1(N_1) = [1\ 1\ 1\ \dots\ 1]
\end{array}
\tag{6}
$$

where each of the above vectors has N_1 elements.

We can now compute the earliest starting time of node S_{21} over all possible schedules of its predecessors as follows:

$$ES_{21} = \min \left\{ ES_{21}(11), ES_{21}(11, 12), \dots, ES_{21}(11, 12, \dots, 1N_1) \right\} \tag{7}$$

where

$$
\begin{aligned}
ES_{21}(11) &= \max \left\{EF_{11},\ EF_{12} + C_{12,21}\right\} \\
ES_{21}(11, 12) &= \max \left\{EF_{11,12},\ EF_{13} + C_{13,21}\right\} \\
&\dots\dots \\
&\dots\dots \\
&\dots\dots \\
ES_{21}(11, 12, 1(N_1-1)) &= \max \left\{EF_{11,\ 12,\ \dots,\ 1(N_1-1)},\ EF_{1N_1} + C_{1N_1,21}\right\} \\
ES_{21}(11, 12, 1N_1) &= EF_{11,\ 12,\ \dots,\ 1N_1}
\end{aligned}
\tag{8}
$$

Excluding the time required for sorting, we need $O(N)$ amount of time to compute $ES_{21}(11)$, $ES_{21}(11, 12)$, ..., $ES_{21}(11, 12, \dots, 1N_1)$ according to Equations (8) and then compute the minimum of these $N_1 = N-1$ terms according to Equation (7).

This concludes the proof of Claim 1.

Critical Predecessors of the Root Node and Optimal Allocation

If

$$ES_{21}(11, 12, 1D_{21}) = \min \left\{ ES_{21}(11), ES_{21}(11, 12), \dots, ES_{21}(11, 12, 1N_1) \right.$$

for some integer D_{21} ($1 \le D_{21} \le N_1$), then we will refer to the subtasks $S_{11}, S_{12}, \dots, S_{1D_{21}}$ as the **critical predecessors** of S_{21}. Then the optimal allocation is given by

$$\mathbf{V_1^*} = [\ 11, 12, \dots, 1D_{21}\ ]$$

and it assigns

a) S_{21} and its critical predecessors $S_{11}, S_{12}, \dots, S_{1D_{21}}$ to the same processor and
b) each of the other predecessors to a separate processor.

We then define as the critical list of node S_{21} the ordered list of nodes $S_{11}, S_{12}, \dots, S_{1D_{21}}$ and denote it as critical-list(S_{21})= ($S_{11}, S_{12}, \dots, S_{1D_{21}}, S_{21}$)

CLAIM 2: The search space is reduced to k* allocations.

Equations (8) have the form

$$ES_{21}(11, 12, \dots, 1k) = \max \left\{ EF_{11,12,\ \dots,\ 1k},\ EF_{1(k+1)} + C_{1(k+1),21} \right\},$$

for k= 1, 2, ..., N_1-1. Let us introduce two cost functions

$$F(k) = EF_{11,12,\ \dots,\ 1k} \qquad 1 \le k \le N_1$$

and

$$G(k) = EF_{1(k+1)} + C_{1(k+1),21} \qquad 1 \le k \le N_1 - 1$$

where **k** is the **allocation index**. Equations (8) can now be written as

$$
\begin{aligned}
H(1) &\equiv ES_{21}(11) = \max \left\{F(1),\ G(1)\right\} \\
H(2) &\equiv ES_{21}(11, 12) = \max \left\{F(2),\ G(2)\right\} \\
&\dots \\
H(k) &\equiv ES_{21}(11, 12, \dots, 1k) = \max \left\{F(k),\ G(k)\right\} \\
&\dots \\
H(N_1-1) &\equiv ES_{21}(11, 12, \dots, 1(N_1-1)) = \max \left\{ F(N_1-1),\ G(N_1-1) \right\} \\
H(N_1) &\equiv ES_{21}(11, 12, \dots, 1N_1) = F(N_1)
\end{aligned}
\tag{8}
$$

while equation (7) becomes

$$ES_{21} = \min \{ H(1), H(2), \dots, H(N_1) \}$$

Inequalities (2) imply that F(k) is a **monotonically increasing function of k**, while the node ordering indicated by (3) implies that the cost function G(k) is a **monotonically decreasing function of k**.

If $F(1) \ge G(1)$, then $F(k) \ge G(k)$ for any $k > 1$ since F increases monotonically with k and G decreases with k. Therefore, $H(k) = F(k)$ for $1 \le k \le N_1$ and

$$ES_{21} = \min \{ H(1), H(2), \dots, H(N_1) \} = H(1) = F(1) = EF_{11}$$

If $F(1) \geq G(1)$, therefore, the optimal solution can be found by computing only the earliest starting time for one allocation (the one defined by $V_1(1) = [11]$).

If $F(1) \leq G(1)$ and $F(2) \geq G(2)$, then $F(k) \geq G(k)$ for any $k > 2$. Therefore

$$H(1) = G(1)$$
$$H(k) = F(k) \quad \text{for } 2 \leq k \leq N_1$$

and

$$ES_{21} = \min \{ H(1), H(2) \}$$

If $F(1) \leq G(1)$ and $F(2) \geq G(2)$, therefore, the optimal solution can be found by computing only the earliest starting times for the two allocations defined by $V_1(1) = [11]$ and $V_1(2) = [11, 12]$.

Since F increases monotonically with k and G decreases with k, we can determine a k^* $(1 \leq k^* \leq N_1)$ such that $F(k) < G(k)$ for $k < k^*$ and $F(k) \geq G(k)$ for any $k \geq k^*$. Then

$$H(k) = G(k) \quad \text{for } 0 \leq k < k^*$$
$$H(k) = F(k) \quad \text{for } k^* \leq k \leq N_1$$

Figure 2A shows one of the two possible cases for intersections of the cost curves F(k) and G(k) for $k^* > 1$, while Figure 2B shows the corresponding H(k) function. We observe that the minimum of the H(k) curve can occur either for $k = k^* - 1$ (case of Figure 2) or for $k = k^*$. Thus,

$$ES_{21} = \min \{ H(k^*-1), H(k^*) \} \tag{9}$$

or equivalently

$$ES_{21} = \min \{ ES_{21}(11, 12, ..., 1(k^*-1)), ES_{21}(11, 12, ..., 1k^*) \} \tag{10}$$

The above equation implies that if

$$ES_{21}(11, 12, ..., 1(k^*-1)) = \\ = \min \{ES_{21}(11,12, ...,1(k^*-1)), ES_{21}(11, 12, ...,1k^*) \}$$

then the number of critical predecessors is

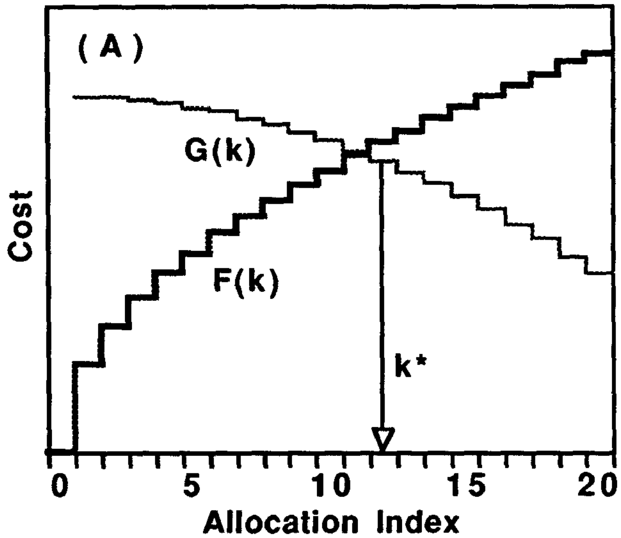
$$D_{21} = k^* - 1 \tag{11}$$

while if

$$ES_{21}(11, 12, ..., 1k^*) = \\ = \min \{ES_{21}(11,12, ..., 1(k^*-1)), ES_{21}(11,12, ...,1k^*) \}$$

then

$$D_{21} = k^* \tag{12}$$



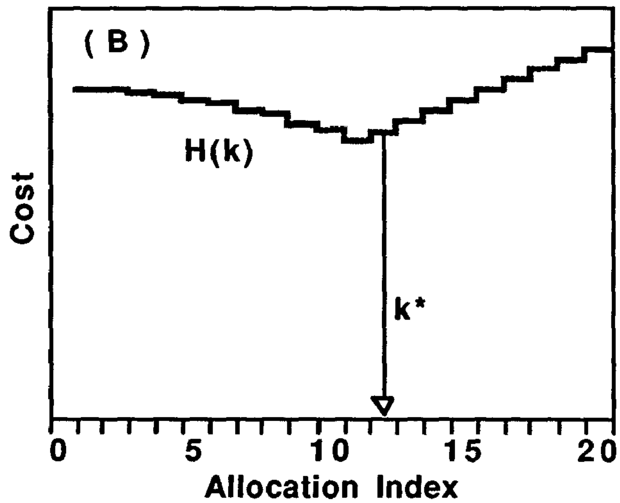


Figure 2: An example showing (A) the intersection of the cost functions F(k) and G(k) and (B) the corresponding H(k) function.

Computational Requirements

If we include the time required for sorting, the computation of the optimal solution from equation (9) for the two level tree requires $O(N \log_2 N)$ time.

Note that k^* could be as small as 1 and as large as N-1. We must also emphasize that k^* may in many cases be a small integer even when N is large. This depends on the initial values of F(1) and G(1) and on the magnitude of the "slopes" of the two curves $\Delta F(k) = F(k+1) - F(k)$ and $\Delta G(k) = G(k+1) - G(k)$. The savings in time provided by equation (9) over equation (7) depends on how small k^* is.

As $N_1 \to \infty$, k* remains finite if the two curves F(k) and G(k) intersect for some finite value of k. There are, however, some families of curves F(k) and G(k) for which a finite k* does not exist.

Consider first the family of F(k) curves that have a horizontal asymptote defined by $y = F_\infty$ where F_∞ is an arbitrary constant. Consider also the family of G(k) curves that have a horizontal asymptote defined by $y = G_\infty$ where G_∞ is another arbitrary constant. If the asymptotic value of the G(k) curve is greater than or equal to the asymptotic value of the F(k) curve, then the two curves will not intersect at a finite value of k. Due to the monotonicity of the F(k) and G(k) curves, it is easy to see that the conditions

$$F(k) \to F_\infty \text{ as } k \to \infty$$
$$G(k) \to G_\infty \text{ as } k \to \infty$$
and
$$G_\infty \geq F_\infty$$

are necessary and sufficient for not having a finite k*.

3. OPTIMAL SCHEDULING FOR A GENERAL K-LEVEL TREE

The optimal algorithm we developed for the K-level tree proceeds by levels .Starting with the second level,it continues to the K-th level as follows. For every 2-nd level node we compute its earliest starting time,earliest finish time as well as its critical list.Then for every node of the i-th (i=2,....,K) level we compute its earliest starting time ,earliest finish time and its critical list from the earliest finish times and the critical lists of its immediate predecessors.We have proved that to compute the earliest finish time and critical list of a node of any level we do not need to consider all the possible allocations of its predecessors and we have reduced significantly the search space for the optimal solution. Claims 3 and 4 establish the feasibility of this reduction of the search space for the 3-level tree. The proof that the algorithm is optimal is performed by induction.An algorithm is presented for the 3-level tree that is proved to be optimal .Then assuming that we have an optimal solution for the (K-1)th level tree we get the optimal solution for the K-th level tree.

3.1. A 3-LEVEL TREE

Figure 3 shows an example of a 3-level tree with N_1 nodes at the first level and N_2 nodes at the second level. An allocation for the 3-level tree is described by two boolean vectors: one vector giving the allocation of the first level nodes and a second vector giving the allocation of the second level nodes. The vectors $\mathbf{B}_2$ with the allocations of the second level nodes are given by:

$$\mathbf{B}_2\,(j) = [\, b_{21}, b_{22}, \dots, b_{2N_2} \,] \qquad 0 \leq j \leq 2^{N_2} - 1$$

where for $1 \leq i \leq N_2$

$b_{2i} = 1$ if S_{2i} is allocated to the same processor as its successor S_{31} on the third level

and

$b_{2i} = 0$ if S_{2i} is **not** allocated to the same processor as its successor S_{31} on the third level.

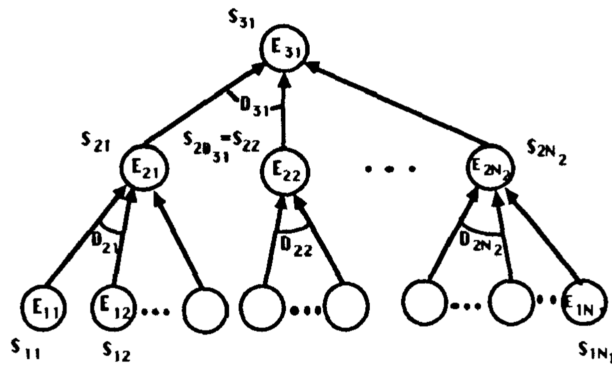


Figure 3: An example of a three-level tree.

The allocation vector $\mathbf{B}_1$ for the first level nodes has now a more complicated form. Each 2-nd level node with its first-level predecessors defines a subtree. The allocation vector $\mathbf{B}_1$ should thus provide for each subtree with root node S_{2i} $(1 \leq i \leq N_2)$ all the information about whether each predecessor of S_{2i} is allocated to the same processor with the root node or to a different one. In analogy to the notation we used in the previous section, this information will be provided by a boolean subvector

$$\beta_{1,2i}\,(j) = [\, \beta_1, \beta_2, \dots, \beta_{N_{2i}} \,]$$

with

$\beta_r = 1$ if S_{1r} is allocated to the same processor as its successor S_{2i} on the second level

and

$\beta_r = 0$ if S_{1r} is **not** allocated to the same processor as its successor S_{2i} on the third level

where N_{2i} is the number of predecessors of node S_{2i} and $1 \leq r \leq N_{2i}$. The allocation vectors $\mathbf{B}_1$ for the first level nodes can be expressed as follows:

$$B_1\,(j) = [\, \beta_{1,21}, \beta_{1,22}, \dots, \beta_{1,2N_2} \,] \quad 0 \leq j \leq 2^{N_1} - 1$$

and an exhaustive search would require us to consider $2^{N_1+N_2}$ allocations.

Each allocation of the 3-level tree, which we will denote by $\mathbf{B} \equiv (\mathbf{B}_1, \mathbf{B}_2)$ or $\mathbf{V} \equiv (\mathbf{V}_1, \mathbf{V}_2)$, defines a schedule as follows:

(a) The vector $\mathbf{B}_2$ gives the 2nd-level nodes that are allocated to the same processor as node S_{31}. These 2nd-level nodes are scheduled according to the order imposed by their earliest starting times which are computed for the particular allocation of the first level nodes indicated by the vector $\mathbf{B}_1$.

(b) Since the earliest starting times for the first level nodes are all the same (and equal to zero), these nodes may be scheduled in any order.

The following definitions are also necessary for the discussion of the three-level tree.

- $ES_{2i}(\beta_{1,2i})$ is the earliest starting time of node S_{2i} for the allocation $\beta_{1,2i}$ of its predecessors.

- ES_{2i} is the earliest starting time for second-level node S_{2i} over **all possible allocations** of its first-level predecessors.

- critical-list (S_{2i}) is the set of nodes consisting of node S_{2i} and all the first level nodes that have to be allocated on the same processor as node S_{2i} in the optimal allocation.

- $ES_{31}(\mathbf{B}_1, \mathbf{B}_2)$ is the earliest starting time of node S_{31} for the allocation of its first- and second-level predecessors defined by $(\mathbf{B}_1, \mathbf{B}_2)$.

- $ES_{31}(\mathbf{B}_2)$ is the earliest starting time of node S_{31} **over all possible allocations** of its first-level predecessors when the allocation of its second-level predecessors is defined by $\mathbf{B}_2$.

- ES_{31} is the earliest starting time of node S_{31} **over all possible allocations** of its first- and second-level predecessors.

- critical-list(S_{31}) is the set of all the nodes consisting of node S_{31} as well as the first and second level nodes that have to be allocated on the same processor as node S_{31} in the optimal allocation.

- $EF_{21,22,\ldots,2k}(\mathbf{B}_1(j))$ is the earliest finish time of the second level nodes $S_{21}, S_{22}, \ldots, S_{2k}$ when they are assigned to the same processor for a given allocation $\mathbf{B}_1(j)$ of their first-level predecessors.

- $EF_{21,22,\ldots,2k}$ is the earliest finish time of the second level nodes $S_{21}, S_{22}, \ldots, S_{2k}$ when they are assigned to the same processor **over all possible allocations** of their first-level predecessors.

- critical-list $(S_{21}, S_{22}, \ldots, S_{2k})$ is the set of nodes consisting of $S_{21}, S_{22}, \ldots, S_{2k}$ as well as all the first level nodes that have to be allocated on the same processor as nodes $S_{21}, S_{22}, \ldots, S_{2k}$ for an optimal $EF_{21,22,\ldots,2k}$.

Theorem 2 summarizes the main results we obtained for a 3-level tree.

THEOREM 2: Consider a 3-level tree with N nodes. For each node S_{2i}, $1 \le i \le N_2$ of the 2-nd level, we compute the earliest starting and finish times ES_{2i} and EF_{2i}, as well as the critical-list (S_{2i}). As before, we order the nodes on the 2-nd level so that they satisfy the inequalities

$$EF_{21} + C_{21,31} \ge EF_{22} + C_{22,31} \ge \\ \ge \ldots \ge EF_{2N_2} + C_{2N_2,31} \qquad (13)$$

Then the following statements are true:

(a) The optimal allocation and the **critical-list** of node S_{31} can be determined in

$$O((D_{21}+1)(D_{22}+1)\ldots(D_{2D_{31}}+1))$$

amount of time , ($1 \le D_{31} \le N_2$). Nodes that do not belong in the critical-list of node S_{31} should be allocated as follows:
Each 2-nd level node that is not an immediate critical predecessor of node S_{31},namely each of the nodes $S_{2(D_{31}+1)}, S_{2(D_{31}+2)}, \ldots, S_{2N_2}$ is allocated to a different processor. We consider each of the nodes $S_{2(D_{31}+1)}, S_{2(D_{31}+2)}, \ldots, S_{2N_2}$ as the **root of an independent 2-level tree** and assign their predecessors in an optimal way as indicated by Theorem 1.

(b) On a given processor, the nodes are scheduled in order of increasing earliest starting times.

(c) The optimal allocation requires $N - SD_1$ number of processors where SD_1 is the total number of critical nodes on the first level (i.e. $SD_1 = D_{21} + D_{22} + \ldots + D_{2D_{31}}$). The algorithm, however, does not guarantee this to be the minimum number of processors.

Proof of Theorem 2 : We have proved that the critical list of node S_{31} consists of

(a) D_{31} 2-nd level nodes ,namely nodes $S_{21}, S_{22}, ..., S_{2D_{31}}$ ($1 \le D_{31} \le N_2$) that will be referred to as the **immediate critical predecessors** of node S_{31}, and

(b) first level nodes that are critical predecessors of the immediate critical predecessors of node S_{31}. For each 2nd-level node S_{2k} ($1 \le k \le D_{31}$) that is an immediate critical predecessor of S_{31}, we determine which first-level nodes will be assigned to the same processor as the one that executes $S_{21}, S_{22}, ..., S_{2D_{31}}$ and S_{31}.

Claims 3 and 4 presented below establish that to compute ES_{31} and the critical list of node S_{31} we do not need to consider all the possible allocations of the 2-nd and 1-st level nodes but only

$$O((D_{21} + 1) (D_{22} + 1) ... (D_{2D_{31}} + 1))$$

possible allocations. Due to the limitations in the length of this communication, we will not present here the proof of these claims.

CLAIM 3: The earliest starting and finish time ES_{31} and EF_{31} of node S_{31} over all possible allocations of its predecessors can be computed by considering for the 2-nd level nodes only those k_{31}^* allocations ($k_{31}^* \le N_2$) for which

(a) node S_{31} and nodes $S_{21}, S_{22}, ..., S_{2k}$, $1 \le k \le k_{31}^*$ are allocated to the same processor and

(b) each remaining node $S_{2(k+1)}, S_{2(k+2)}, ..., S_{2N_2}$ is allocated to a different processor.

CLAIM 4: Let $B_2(j_1)$ be an allocation that assigns the second level nodes $S_{21}, S_{22}, ..., S_{2j}, ..., S_{2k}$ to the same processor as node S_{31}. Let also S_{2j} $1 \le j \le k$ $k = 2, ..., k_{31}^*$ be one of the 2-nd level nodes assigned to the same processor with node S_{31}. Then the following are true:

a) Only nodes of the critical list of node S_{2j} should be assigned to the same processor with S_{2j}.

b) If m_{2j} (immediate) critical predecessors of S_{2j} ($0 \le m_{2j} \le D_{2j}$) are also allocated to the same processor as S_{2j}, then these should be the first m_{2j} critical predecessors according to the ranking defined by the inequalities (3).

We say then that node S_{2j} has D_{2j} immediate critical predecessors from the second level point of view and denote it by $D_{2j}(2)$ and m_{2j} immediate critical predecessors from the third level point of view and denote it by $D_{2j}(3)$. We have

$$D_{2j}(2)= D_{2j} \text{ and } D_{2j}(3) = m_{2j} \le D_{2j}(2).$$

Algorithm for the 3-level tree

This procedure consisting of steps1-5 computes ES_{31} and critical-list(S_{31}) from the earliest finish times and the critical lists of all the second level nodes.

1. We first compute EF_{2i} , $i = 1, 2, ..., N_2$ for each second-level node according to Theorem 1 and order these nodes so that the following inequalities are satisfied

$$EF_{21} + C_{21,31} \ge EF_{22} + C_{22,31} \ge ... \ge EF_{2N_2} + C_{2N_2,31}$$

2. **CASE I:** If $EF_{21} \ge EF_{22} + C_{22,31}$, the search for the optimal solution stops. Then

$$ES_{31} = EF_{21} + E_{31}$$

and

$$\text{critical-list } (S_{31}) = \text{critical-list } (S_{21}) \cup S_{31}$$

CASE II: If $EF_{21} < EF_{22} + C_{22,31}$, we proceed to compute $EF_{21,22}$ and the critical-list (S_{21},S_{22}) as shown in Step 5 below.

3. **CASE I:** If $EF_{21,22} \ge EF_{23} + C_{23,31}$, the search for the optimal solution stops. Then

$$ES_{31} = \min \{ \max (EF_{21} , EF_{22}+C_{22,31}), \max (EF_{21,22}, EF_{23}+C_{23,31}) \} =$$

$$= \min \{ EF_{22}+C_{22,31} , EF_{21,22} \}$$

If

$$\min \{ \max (EF_{21} , EF_{22}+C_{22,31}), \max (EF_{21,22}, EF_{23} + C_{23,31}) \} = = EF_{22} + C_{22,31}$$

then

$$\text{critical-list } (S_{31}) = \text{critical-list } (S_{21}) \cup S_{31}$$

If

$$\min \{ \max (EF_{21} , EF_{22} + C_{22,31}), \max (EF_{21,22}, EF_{23} + C_{23,31}) \} = = EF_{21,22}$$

then

$$\text{critical-list } (S_{31}) = \text{critical-list } (S_{21},S_{22}) \cup S_{31}$$

CASE II: If $EF_{21,22} < EF_{23} + C_{23,31}$, however, we proceed to compute $EF_{21, 22, 23}$ and critical-list (S_{21},S_{22},S_{23}) as shown in Step 5 below.

4. This procedure is repeated until we find an integer k_{31}^* such that
$EF_{21,22,\ldots,2k_{31}^*} \geq EF_{2k_{31}^*+1} + C_{2k_{31}^*+1,31}$

Then the search for the optimal solution stops and

$$ES_{31} = \min \{ \max (EF_{21,22,\ldots,2k_{31}^*-1}, EF_{2k_{31}^*} + C_{2k_{31}^*,31}), \max (EF_{21,22,\ldots,2k_{31}^*}, EF_{2k_{31}^*+1} + C_{2k_{31}^*+1,31}) \} = \min \{ EF_{2k_{31}^*} + C_{2k_{31}^*,31}, EF_{21,22,\ldots,2k_{31}^*} \}$$

If $\min \{ \max (EF_{21,22,\ldots,2k_{31}^*-1}, EF_{2k_{31}^*} + C_{2k_{31}^*,31}), \max (EF_{21,22,\ldots,2k_{31}^*}, EF_{2k_{31}^*+1} + C_{k_{31}^*+1,,31}) \} = EF_{2k_{31}^*} + C_{2k_{31}^*,31}$
then
critical-list (S_{31}) =
= critical-list $(S_{21}, S_{22}, \ldots\ldots, S_{2k_{31}^*-1}) \cup S_{31}$

If $\min \{ \max (EF_{21,22,\ldots,2k_{31}^*-1}, EF_{2k_{31}^*} + C_{2k_{31}^*,31}), \max (EF_{21,22,\ldots,2k_{31}^*}, EF_{2k_{31}^*+1} + C_{k_{31}^*+1,,31}) \} = EF_{21,22,\ldots,2k_{31}^*}$
then
critical-list (S_{31}) =
= critical-list $(S_{21}, S_{22}, \ldots\ldots, S_{2k_{31}^*}) \cup S_{31}$

5. **Computation of $EF_{21,22,\ldots,2k}$ and of the critical-list $(S_{21}, S_{22}, \ldots\ldots, S_{2k})$, $k = 1, 2, \ldots, k_{31}^*$**

To compute $EF_{21,22,\ldots,2k}$ as well the critical-list $(S_{21}, S_{22}, \ldots\ldots, S_{2k})$ $k = 1, 2, \ldots, k_{31}^*$ we need to consider the following allocation for the 2-nd level nodes

$$B_2(j) = \begin{array}{c} [\ 1\ \ 1\ \ldots\ 1\ \ 0\ \ 0\ \ldots\ 0\ ] \\ \uparrow \\ k \end{array}$$

An exhaustive search would have required to consider all $2^{N_{21}+N_{22}+\ldots+N_{2K}}$ possible allocations

$$B_1(j) = [\ \beta_{1,21}, \beta_{1,22}, \ldots, \beta_{1,2N_2}\ ]$$

where $0 \leq j \leq 2^{N_{21}+N_{22}+\ldots+N_{2K}} - 1$ for the first level nodes. According to claim 4, however, we need only consider the following $(D_{2i} + 1)$ allocations for each of the boolean subvectors $\beta_{1,2i}$ $1 \leq i \leq k$

$$\begin{array}{c} [\ 0\ 0 \cdots\ 0\ \cdots 0\ ] \\ [\ 1\ 0 \cdots\ 0\ \cdots 0\ ] \\ [\ 1\ 1 \cdots\ 0\ \cdots 0\ ] \\ \cdots\ \ \cdots \\ [\ 1\ 1 \cdots\ 1\ \cdots 0\ ] \\ \uparrow \\ D_{2i} \end{array} \tag{16}$$

Thus for the derivation of $EF_{21,22,\ldots,2k}$ and the critical-list $(S_{21}, S_{22}, \ldots\ldots, S_{2k})$ we need to consider totally only
$(D_{21} + 1)(D_{22} + 1) \ldots (D_{2D_{31}} + 1)$
possible allocations of the predecessors of $S_{21}, S_{22}, \ldots, S_{2k}$ and
$EF_{21,22,\ldots,2k}$ =
= $\min \{ EF_{21,22,\ldots,2k}(\beta_{1,21}, \beta_{1,22}, \ldots, \beta_{1,2k}) \}$

over all possible allocations $\beta_{1,2i}$ given by (16).

3.2 A K-LEVEL TREE

Theorem 3: Consider a K-level tree with N nodes. For each node $S_{(K-1)i}$, $1 \leq i \leq N_{K-1}$ of the (K-1) level, we compute the earliest starting and finish times $ES_{(K-1)i}$ and $EF_{(K-1)i}$ $(1 \leq i \leq N_{K-1})$, as well as the critical list of nodes $S_{(K-1)i}$. As before, we order the nodes on the (K-1) level so that they satisfy the inequalities

$$EF_{(K-1)1} + C_{(K-1)1,K1} \geq EF_{(K-1)2} + C_{(K-1)2,K1} \geq \ldots \geq EF_{(K-1)N_{(K-1)},K1} + C_{(K-1)N_{(K-1)},K1}$$

Then the following statements are true:

(i) The optimal allocation and the critical list of node S_{K1} can be determined in

$$O((D_{21} + 1)(D_{22} + 1) \ldots (D_{2SD_2} + 1))$$

amount of time where SD_2 is the total number of critical nodes of the second level.

We should allocate all nodes that do not belong to the critical list of node S_{K1} in an optimal allocation as follows:

- Each node of the (K-1)st level which is not an immediate critical predecessor of node S_{K1}, namely each of the nodes $S_{(K-1)(D_{K1}+1)}, S_{(K-1)(D_{K1}+2)}, \ldots, S_{(K-1)N_{K-1}}$, is allocated to a different processor.

We consider each of the nodes $S_{(K-1)(D_{K1}+1)}$, $S_{(K-1)(D_{K1}+2)}$, ..., $S_{(K-1)N_{(K-1)}}$ as the **root of an independent tree with K-1 levels** and assign their predecessors in an optimal way.

(ii) On a given processor, the nodes are scheduled in order of increasing earliest starting times.

(iii) The number of processors needed for the optimal schedule is $N - SD_1$, where SD_1 is the total number of critical nodes on the first level. The algorithm, however, does not guarantee this to be the minimum number of processors.

We should emphasize that even when $N_1 \rightarrow \infty$ and $N_2 \rightarrow \infty$, the numbers $D_{21}, D_{22}, \ldots, D_{2SD_2}$ as well as SD_2 can be small and the algorithm very efficient.

4. CONCLUSIONS

An optimal algorithm was developed that minimizes the processing time for a computation that can be represented by a tree and executed on a parallel system with communication delays when there is no restriction on the number of processors used.The algorithm is efficient in terms of computation time as well the number of processors it requires for the optimal solution.

ACKNOWLEDGMENT

This work was supported through a grant by the Texas Advanced Research Program.

REFERENCES

[1] C.H. Papadimitriou and M. Yannakakis, "Towards an Architecture-Independent Analysis of Parallel Algorithms", Procs. STOC, 1988.

[2] H.Jung, L. Kiroussis and P. Spirakis, "Lower Bounds and Efficient Algorithms for Multiprocessor Scheduling of Dags with Communication Delays ", Procs. SPAA, 1989.

[3] H.S. Stone, "Multiprocessor Scheduling with the Aid of Network Flow Algorithms", IEEE Trans. on Software Eng., Vol. SE-3, No. 1, January 1977, pp. 85-93.

[4] P. Markenscoff, W. Liaw, "Task Allocation Problems in Distributed Computer Systems," Procs. International Conference on Parallel Processing, August 1986, pp. 953-960.

[5] P. Markenscoff, D. Joe "Computation of Tasks Modeled by Directed Acyclic Graphs - Allocation withoout Subtask Replication" Procs 1990 IEEE International Symposium on Circuits and Systems.