Evaluation of parallel execution of program tree structures

Philippe Mussi, Philippe Nain

INRIA

CENTRE DE ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P. 105
78153 Le Chesnay Cedex
France
Tél.: (3) 954 90 20

Rapports de Recherche

N° 318

EVALUATION OF PARALLEL EXECUTION OF PROGRAM TREE STRUCTURES

Philippe MUSSI
Philippe NAIN

Juillet 1984

EVALUATION OF PARALLEL EXECUTION OF PROGRAM

TREE STRUCTURES

Philippe Mussi - Philippe Nain
INRIA
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
France

RESUME

Nous nous intéressons à l'exécution parallèle d'un ensemble de tâches auxquelles les contraintes de synchronisation assignent une structure d'arbre.

Nous définissons deux politiques pour l'exécution -ascendante- de ce type de structures. Ces politiques se différencient par l'existence (NA) ou non (A) de synchronisations supplémentaires à la fin de l'exécution de chaque niveau.

Sous l'hypothèse que les temps d'exécution des différentes tâches constituent des variables aléatoires indépendantes de même loi exponentielle, nous obtenons, pour la politique NA, la transformée de Laplace-Stieltjes de la distribution du temps total d'exécution pour un nombre arbitraire de processeurs. Cette même transformée est obtenue, pour la politique A, dans le cas de deux processeurs, moyennant la résolution numérique d'un système linéaire.

Nous comparons ensuite les temps d'exécution moyens d'arbres binaires pleins sous les deux politiques et nous sommes amenés à conjecturer le comportement asymptotique du temps moyen d'exécution sous la politique A, en fonction du nombre de tâches.

EVALUATION OF PARALLEL EXECUTION OF PROGRAM TREE STRUCTURES

Ph. MUSSI - Ph. NAIN
INRIA
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
France

ABSTRACT

We define and evaluate two policies (NA-policy, A-policy) for parallel execution of program tree structures. Via a probabilistic model we analytically determine, for each policy, the Laplace-Stieltjes transform for the tree processing time distribution. The acceleration of the program execution time achieved when adding processors to a single processor environment, is computed and plotted for each policy.

1 - INTRODUCTION

With the growing development of multiprocessor systems, various algorithms for parallel computation recently appeared in the literature. Let us quote, for instance, the parallel parser of arithmetic expressions [1,2], the parallel multiplication of two binary N-bit numbers in VLSI architectures [6], the parallel computation of partial differential equations in numerical analysis [4].

More formally, many program executions can be represented as a set **T** of tasks and a set **R** of precedence relations between these tasks, such that (**R** , **T**) be a tree. Such tree structures are particularly appropriate for a parallel execution. Given a tree structure and a multiprocessor system, it is clear that two different policies for the execution of the tree will generally yield different performances (e.g. execution time of the tree, processor availability,...). Consequently, various scheduling policies have to be evaluated and compared in order to determine the more efficient. Most studies related to this subject deal with a number of available processors equal to the number of tasks to be processed, which is referred in the literature to as the case with "infinite available processors" [1], [2], [3].

Theoretical tools involved in these works in view of performance evaluation (e.g. branching processes [1], [2], convolution equations [3]) cannot be used if we consider a small number of available processors compared to the number of tasks to be executed. An illustration of this last case arises in the evaluation of arithmetic expressions and also in many problems with vector and matrix computations. For instance, let us consider the scalar product of two m-vectors X,Y, where x_i[resp y_i] is the ith component of vector X [resp Y], i = 1,...,m. This scalar product can be computed using the tree structure given in Fig. 1, where in practical situations the number of available processors will be always smaller than 2m for large values of m.

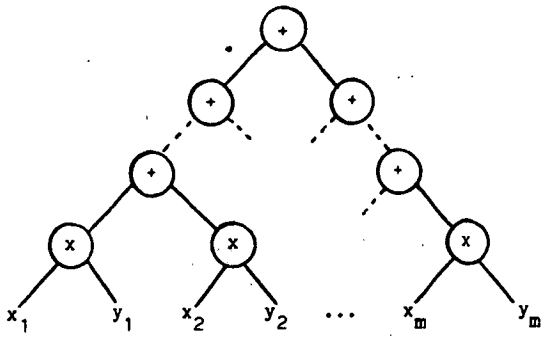


Figure 1 : Scalar product of two m-vectors

In this paper we consider a tree **A**, with M (M is deterministic) the total number of nodes and leaves (the tasks) and a fixed number of processors N, with $N < M$.

For two different tree execution policies and if the execution times of tasks are independent random variables, we determine the Laplace-Stieltjes transform (L.S.T.) of the tree processing time distribution (total program execution time).

In Section 2 we introduce the two tree execution policies, which will be refer to as the Non-Anticipatory policy (NA-policy) and the Anticipatory policy (A-policy). The existence or not of the anticipation is related to the existence (A-policy) or not (NA-policy) of synchronization constraints between processors at the entry of each execution level. A more thorough description will be given in Section 2.

Under probabilistic assumptions characterizing the program tree structure (Section 3) we explicitly obtain using pure death process and for arbitrary N ($1 \leq N < M$), the L.S.T. of the tree processing time distribution under the NA-policy.

Similarly an analytic expression is derived for the L.S.T. of the tree execution time distribution under the A-policy with $N = 2$ (Section 4), up to a fixed number of unknown constants.

It is shown that these constants satisfy linear recurrence relations, whose solutions can be easily determined using an exact numerical procedure.

Mean tree processing times obtained under the two policies under consideration are compared for N=2 and $N=\infty$ to the corresponding result with a single processor (Section 5). A kind of speeding up is defined and estimated. Numerical results found for the A-policy and N=2 lead us to conjecture an asymptotic result for the mean tree processing time.

2 - BASIC ASSUMPTIONS

We consider a tree **A**, consisting in M nodes, with D its depth.

All the nodes of **A** represent tasks.

The execution time of an arbitrary task of level i ($i = 0,1,\ldots, D$) is an exponentially distributed random variable with parameter μ_i.

These M random variables are supposed to be mutually independent.

A task can be executed iff. all its sons have been executed and no other conflicts can occur between tasks. We assume that the time required for allocation of a task to a processor is negligible regarding to its execution time.

Let M_i and T be respectively the number of tasks at level i and the total execution time of the tree ($M = M_0 + M_1 + \ldots + M_D$, $M_0 = 1$). Note that M_i is a known constant and T is a random variable. $\overline{T}$ will denote the expected total execution time and $T^*(s) = E[e^{-sT}]$ for $\mathrm{Re}\, s \geq 0$, the L.S.T. of the distribution of T. We now extensively define the two policies under consideration.

Let T_i be the set of tasks of level i ($i = 0,1,\ldots,D$). Under the NA-policy, each of the N available processors will behave as follows, where initial values of (E,E',i) are (T_D,T_D, D) :

```
Step 1 If E' = ∅ go to Step 6
Step 2 Choose arbitrarily  e  ∈  E'
Step 3 Remove e from E'
Step 4 Execute e
Step 5 Remove e from E
Step 6 If E = ∅ then
         if i ≠ 0  then
          (E,E',i) ← ( T_{i-1}, T_{i-1},i-1)
          go to Step 1
                 else END
              else
         if E' ≠ ∅  then go to Step 2
                    else
              if i ≠ 0 then WAIT until E' ≠ ∅
                            go to Step 1
                       else END
```

At any instant of the tree execution, the set E indicates the total number of tasks of the current level which are either being executed or non-executed. Similarly, the set E' represents the total number of tasks of the current level which are non-executed ($E' \subset E$). Under this algorithm, the D+1 subsets of tasks T_D, T_{D-1},..., T_0 are therefore successively processed in that order.

Instant of end of execution of each subset is a synchronization point for the N processors.

Thus, the M_i tasks of T_i are executed by the N processors with maximal parallelism.

Under the A-policy each of the N processors behaves as follows (we assume that all the nodes of **A** are initially in the state "non-marked") :

Step 1 — If there exists non-marked leaves, then choose arbitrarily one leaf L among those with maximal depth.
Else END

Step 2 — Mark L

Step 3 — Execute L

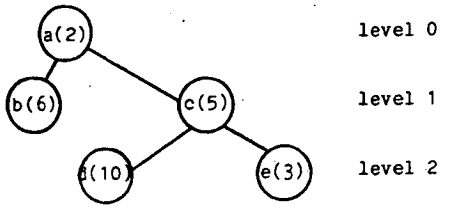
Step 4 — Remove L from **A**
Go to Step 1

Hence, it is seen that a processor executing tasks under the A-policy do not wait for the other processors for synchronization constraints. When there is no more executable tasks, it becomes free.

Example :

N = 2
D = 2
T_0 = {a} ; T_1 = {b,c} ; T_2 = {d,e}.

Execution times of a,b,c,d,e are respectively 2,6,5,10,3.



We assume, for instance, that at time t = 0, tasks d and e are respectively allocated to processors P_1 and P_2 ($P_1 \leftarrow$ d, $P_2 \leftarrow$ e).

We then have the following sequences :

NA-policy

```
0  P1 ← d, P2 ← e ;
3  end of e (P2) ;
10 end of d (P1) ,
   P1 ← c ; P2 ← b (for instance) ;
15 end of c (P1) ;
16 end of b (P2) ,
   P1 ← a ; P2 idle (for instance) ;
18 end of A
time
```

A-policy

```
0  P1 ← d, P2 ← e ;
3  end of e (P2) ;
9  end of b (P2) ;
   P2 becomes free ;
10 end of d (P1) ;
15 end of c (P1) ;
17 end of A .
```

3 - PERFORMANCE EVALUATION OF THE NA-POLICY

3.1 - Notations

For level i of the tree (i = 0,1,...,D), let us define,

- $N_i = \min(N, M_i)$ the number of available processors,

- $\nu_i(t)$ the number of non-executed tasks (including the one(s) being executed) at time t ($t \geq 0$),

- $Q_i(t;n) = P(\nu_i(t) = n)$, $n = 0,1,\ldots, M_i$

- T_i the execution time of level i, $\overline{T}_i$ its expectation,

- $T_i(t) = P(T_i \leq t)$, and $T_i^*(s) = E[e^{-sT_i}]$, Re $s \geq 0$.

From model assumptions, it is readily seen that

$$T^*(s) = \prod_{i=0}^{D} T_i^*(s) \qquad (3.1)$$

and $$T_i(t) = Q_i(t;0). \qquad (3.2)$$

In the following of this section, we are then concerned, for given i (i = 0,1,...,D), with the determination of $T_i^*(s)$ (Re $s \geq 0$), the L.S.T. of $T_i(t)$. For sake of clearness and without loss of generality, we assume that execution of level i begins at time t = 0.

3.2 - Determination of the distribution of T

From model assumptions, the process $\{\nu_i(t), t>0\}$ is a pure death process with death rate $\mu_i \min(N_i,n)$, which is the rate at which deaths occur when the population (non-executed tasks) is of size n (n=0,1,...M_i).

Consequently, we have the resulting equations,

$$\frac{d}{dt} Q_i(t;n) = -\mu_i n\, Q_i(t;n) + \mu_i(n+1)\, Q_i(t;n+1),$$

for n=0,1,...N_i-1 (3.3)

$$\frac{d}{dt} Q_i(t;n) = -\mu_i N_i\, Q_i(t;n) + \mu_i N_i\, Q_i(t;n+1),$$

for n=N_i,...,M_i-1 (3.4)

$$Q_i(t;M_i) = e^{-\mu_i N_i t}, \qquad (3.5)$$

$$Q_i(0;n) = 1_{\{n=M_i\}}, \qquad (3.6)$$

where $1_{\{.\}}$ is the indicator function of the event $\{.\}$.

Solving for $Q(t;N_i)$, we easily have from eqs. (3.4) and (3.5),

$$Q_i(t;N_i) = \frac{(\mu_i N_i t)^{M_i - N_i}}{(M_i - N_i)!}\, e^{-\mu_i N_i t}. \qquad (3.7)$$

Interpretation of this result is quite obvious, since when all the N_i processors are busy (which occurs when the number of non-executed tasks is at least equal to N_i), then the death rate of the pure death process is $\mu_i N_i$.

Solving now for $Q_i(t;n)$ for n = 0,1,...,N_i-1 we obtain from eq. (3.3),

$$Q_i(t;n) = \mu_i(n+1) \int_0^t e^{-\mu_i n(t-x)} Q_i(x;n+1)\, dx. \qquad (3.8)$$

Taking Laplace transform at point s (Re $s \geq 0$) in both sides of eq.(3.8), we get,
for n = 1,2,... $N_i - 1$,

$$\tilde{Q}_i(s;n) = \frac{\mu_i(n+1)}{\mu_i n + s}\, \tilde{Q}_i(s;n+1), \qquad (3.9)$$

where $\tilde{Q}_i(s;k) \overset{\text{def}}{=} \int_0^\infty e^{-st} Q_i(t;k)dt$,
k=1,..., N_i-1, Re $s \geq 0$.

Iterating eq.(3.9), we finally have,

$$\tilde{Q}_i(s;1) = \frac{\mu_i^{N_i-1}\, N_i!}{\prod_{k=1}^{N_i-1} (\mu_i k + s)}\, \tilde{Q}_i(s;N_i), \quad \text{Re } s \geq 0. \qquad (3.10)$$

From the wellknown identity,

$$\int_0^{+\infty} x^k e^{-\lambda x}\, dx = \frac{k!}{\lambda^{k+1}} \quad (\lambda > 0,\ k \in \mathbb{N})$$

and eq.(3.7), it comes

$$\tilde{Q}_i(s;N_i) = \frac{1}{\mu_i N_i}\left(\frac{\mu_i N_i}{\mu_i N_i + s}\right)^{M_i - N_i + 1}, \quad \text{Re } s \geq 0.$$

So, using eq.(3.10),

$$\tilde{Q}_i(s;1) = \frac{\mu_i^{N_i - 1}\, N_i\,!}{\prod_{k=1}^{N_i}(\mu_i k + s)}\left(\frac{\mu_i N_i}{\mu_i N_i + s}\right)^{M_i - N_i}, \quad \text{Re } s \geq 0. \quad (3.11)$$

Since from eqs.(3.2), (3.3),

$$T_i^*(s) = \mu_i \tilde{Q}_i(s;1), \quad \text{Re } s \geq 0$$

then (cf. eqs.(3.1), (3.11))

$$T_i^*(s) = \frac{\mu_i^{N_i}\, N_i\,!}{\prod_{k=1}^{N_i}(\mu_i k + s)}\left(\frac{\mu_i N_i}{\mu_i N_i + s}\right)^{M_i - N_i}, \quad (3.12)$$

and

$$T^*(s) = \prod_{i=0}^{D} \frac{\mu_i^{N_i}\, N_i\,!}{\prod_{k=1}^{N_i}(\mu_i k + s)}\left(\frac{\mu_i N_i}{\mu_i N_i + s}\right)^{M_i - N_i}, \quad \text{Re } s \geq 0. \quad (3.13)$$

Differentiating eq.(3.12), then making $s = 0$, we have

$$\bar{T}_i = \frac{M_i - N_i + 1}{\mu_i N_i} + \frac{1}{\mu_i}\sum_{k=1}^{N_i - 1}\frac{1}{k}, \quad (3.14)$$

which is the mean execution time of level i.

The mean execution time of the entire tree under the NA-policy is then given by (cf. eq(3.1)),

$$\bar{T} = \sum_{i=0}^{D}\left(\frac{M_i - N_i + 1}{\mu_i N_i} + \frac{1}{\mu_i}\sum_{k=1}^{N_i - 1}\frac{1}{k}\right). \quad (3.15)$$

The first term $\frac{M_i - N_i + 1}{\mu_i N_i}$ of the right side of eq.(3.14) is the mean execution time of the $(M_i - N_i + 1)$st tasks of level i, since for these tasks the death rate is $\mu_i N_i$.

Similarly, the second term $\frac{1}{\mu_i}\sum_{k=1}^{N_i - 1}\frac{1}{k}$ gives the mean execution time of the $N_i - 1$ last tasks, since for these tasks the death rate decreases from $\mu_i(N_i - 1)$ (for the $(M_i - N_i + 2)$th task) to μ_i (for the last task to be executed at level i).

4 - PERFORMANCE EVALUATION OF THE A-POLICY

In this section, we compute the L.S.T. of the execution time distribution of a given tree **A** of M tasks, when two processors work under the A-policy [see Section 2 for details on the A-policy].

4.1 - Assumptions and Notation

Let us recall the main notations introduced in Sections 1 and 2. T is the total execution time of a given tree **A** with D levels ($D \geq 2$) and M nodes. At level i, the M_i tasks have exponentially distributed execution times, with parameter μ_i. We denote as n_i the number of tasks of level i, which are neither executed nor being executed.

In the following, we call a state of the system (**A**, Processors) one of the objects :

- (i,j,n_i), $1 \leq i \leq j \leq D$, $1_{\{i \neq j\}} \leq n_i \leq M_i - 1 - 1_{\{i=j\}}$, which represents the event { the processors are executing 2 tasks of level i and j, $i \leq j$, and n_i tasks remain unprocessed at level i} ;

- (I,j), $0 \leq j \leq D$, which represents the event {one processor is idle, the other is executing a task of level j } ;

- F (the "final" state : all the tasks have been executed).

We denote by E the set of these states. We assume, without loss of generality, that the two processors begin working at time t = 0.

4.2 - Computation of the distribution of T

It is easy to see that, under the assumptions made above, the process $\{E(t),\ t > 0\}$ is a Markov process, where E(t) denotes the state of the system at time t.

Hence, we can write the following evolution equations for the function $P(t;E) \overset{def}{=}$ Probability of state E at time t, $E \in \mathbf{E}$,

i) $1 \le i < j \le D$, $0 < n_i \le M_i - 1$

$$\frac{d}{dt} P(t;i,j,n_i) = -(\mu_i+\mu_j)\, P(t;i,j,n_i)$$
$$+ \mu_{j+1}\, 1_{\{j<D\}}\ P(t;i,j+1,n_i)$$
$$+ 1_{\{n_i \le M_i-2\}}\, \mu_i\, P(t;i,j,n_i+1)$$
$$+ 1_{\{n_i = M_i-1\}} \left[2\mu_j\, 1_{\{o_i=j\}}\, P(t;j,j,0)\right.$$
$$\left. + \mu_{o_i} 1_{\{o_i<j\}} P(t;o_i,j,1)\right], \qquad (4.1)$$

where $o_i \overset{def}{=} \min\{k \in [i+1,\dots,D] / M_k > 1\}$, $i = 1,\dots,D-1$;

ii) $1 \le i \le D$, $0 \le n_i \le M_i - 2$

$$\frac{d}{dt} P(t;i,i,n_i) = -2\mu_i\, P(t;i,i,n_i)$$
$$+ \mu_{i+1}\, 1_{\{i<D\}}\, P(t;i,i+1,n_i+1)$$
$$+ 2\mu_i\ 1_{\{n_i+1 \le M_i-2\}}\, P(t;i,i,n_i+1), \qquad (4.2)$$

iii) $2 \le i \le D$

$$\frac{d}{dt} P(t;I,i) = -\mu_i P(t;I,i)$$
$$+\mu_{i+1}\, 1_{\{i<D\}}\, P(t;I,i+1) + \mu_1 P(t;1,i,1), \qquad (4.3)$$

iv) $$\frac{d}{dt} P(t;I,1) = -\mu_1 P(t;I,1) + \mu_2 P(t;I,2)$$
$$+ 2\mu_1 P(t;1,1,0), \qquad (4.4)$$

v) $$\frac{d}{dt} P(t;I;0) = -\mu_0\, P(t\ ;\ I\ ;\ 0) + \mu_1 P(t;I,1)\ , \qquad (4.5)$$

vi) $$\frac{d}{dt} P(t\ ;\ F) = \mu_0 P(\ t\ ;\ I\ ,\ 0)\ , \qquad (4.6)$$

with, for instance and without loss of generality the following initial condition

$$P\ (0;D,D,M_D-2) = 1\ , \qquad (4.7)$$

(i.e. we assume that at level D there exists at least 2 leaves, which in turn ensures that o_i is well-defined for $i = 1,\dots,D-1$)

and the convention

$P(t\ ;\ i\ ,\ j\ ,\ n_i) = 0$ and $P(t\ ;\ I\ ,i) \overset{def}{=} 0$ if $(i\ ,\ j\ ,\ n_i)$ or (I,i) are not in E.

The last term between brackets in the right-hand side of equation (4.1) expresses that the "head processor" jumps from level j to level j-1 iff. $M_{j-1} > 1$ (cf. Section 2); otherwise it jumps to the nearer level i ($i<j-1$) that contains at least two tasks (i.e. $M_i>1$), if any.

In order to obtain tractable equations, we make the additional assumptions $\mu \overset{def}{=} \mu_i$, $i = 0,\dots, D$. Introducing $\tilde{P}(s;E)$, the Laplace transform of the function P(t ; E), we obtain from eqs. (4.1) to (4.6) the following equations, Re (s) ≥ 0,

i) $1 \le i < j \le D$, $0 < n_i \le M_i - 1$

$$\tilde{P}(s;i,j,n_i) = \frac{\mu}{s+2\mu} \left[1_{\{j<D\}}\, \tilde{P}(s;i,j+1,n_i)\right.$$
$$+1_{\{n_i \le M_i-2\}} \tilde{P}(s\ ;\ i\ ,\ j\ ,\ n_i+1)$$
$$+ 1_{\{n_i = M_i-1\}} [1_{\{o_i=j\}} 2\tilde{P}(s;j,j,0)$$
$$\left. + 1_{\{o_i<j\}} \tilde{P}(s;o_i,j,1)\right], \qquad (4.8)$$

ii) $1 \le i \le D$, $0 \le n_i \le M_i - 2$

$$\tilde{P}(s;i,i,n_i) = \frac{\mu}{s+2\mu} \left[1_{\{i<D\}} \tilde{P}(s;i,i+1,n_i+1) + 1_{\{n_i+1 \le M_i-2\}} 2\tilde{P}(s;i,i,n_i+1) + 1_{\{i=D, n_i=M_i-2\}} \frac{C}{s+2\mu} \right], \qquad (4.9)$$

where C is a constant determined by initial condition (4.7),

iii) $2 \le i \le D$

$$\tilde{P}(s;I,i) = \frac{\mu}{s+\mu} \left[1_{\{i<D\}} \tilde{P}(s;I,i+1) + \tilde{P}(s;1,i,1) \right], \qquad (4.10)$$

iv) $$\tilde{P}(s;I,1) = \frac{\mu}{s+\mu} \left[\tilde{P}(s;I,2) + 2\tilde{P}(s;1,1,0) \right], \qquad (4.11)$$

v) $$\tilde{P}(s;I,0) = \frac{\mu}{s+\mu} \tilde{P}(s;I,1). \qquad (4.12)$$

From model assumptions, and eq.(4.6), it is readily seen that

$$T^*(s) = \mu P(s;I,0) , \quad \text{Re } s \ge 0, \qquad (4.13)$$

where $T^*(s)$ is the L.S.T. of the total tree execution time.

Since we have assumed that $P(0;D,D,M_D-2) = 1$ (cf(4.7)), then

$\tilde{P}(s;D,D,M_D-2) = \frac{1}{s+2\mu}$, Re $s \ge 0$, which in turn entails that $C = 1$ (cf.(4.9)).

We now remark that for all $(i,j,n_i) \in E$, $\tilde{P}(s;i,j,n_i)$ may be rewritten as :

$$\tilde{P}(s;i,j,n_i) = a(i,j,n_i) \frac{\mu^{N_1(i,j,n_i)}}{(s+2\mu)^{N_2(i,j,n_i)}} ,$$

with a, N_1, N_2 three integer state functions.

More precisely, we obtain from eqs.(4.8) and (4.9),

$$\tilde{P}(s;i,j,n_i) = \frac{a(i,j,n_i)}{\mu} \left(\frac{\mu}{s+2\mu} \right)^{N(i,j,n_i)} , \qquad (4.14)$$

where $N(i,j,n_i) \overset{\text{def}}{=} \sum_{k=i}^{D} M_k - n_i + i - j$, if $i < j$,

$\overset{\text{def}}{=} \sum_{k=i}^{D} M_k - n_i - 1$, if $i = j$,

and $a(i,j,n_i)$ is the integer state function defined by the following linear relations,

i) $1 \le i < j \le D$, $0 < n_i \le M_i - 1$

$$a(i,j,n_i) = 1_{\{j<D\}} a(i,j+1,n_i) + 1_{\{n_i \le M_i-2\}} a(i,j,n_i+1) + 1_{\{n_i=M_i-1\}} \left[1_{\{o_i=j\}} 2a(j,j,0) + 1_{\{o_i<j\}} a(o_i,j,1) \right], \qquad (4.15a)$$

ii) $1 \le i \le D$, $0 \le n_i \le M_i - 2$

$$a(i,i,n_i) = 1_{\{i<D\}} a(i,i+1,n_i+1) + 1_{\{n_i+1 \le M_i-2\}} 2a(i,i,n_i+1) + 1_{\{i=D, n_i=M_i-2\}} , \qquad (4.15b)$$

iii) $a(i,j,n_i) = 0$ for all other values of i,j,n_i. (4.15c)

On the other hand, iterating eq.(4.10), we easily derive for i=2,...,D,

$$\tilde{P}(s;I,i) = \sum_{k=i}^{D} \left(\frac{\mu}{s+\mu} \right)^{k-1} \tilde{P}(s;1,k,1). \qquad (4.16)$$

Then, using eqs.(4.11) to (4.13) we get

$$T^*(s) = \mu\tilde{P}(s;I,0) = \frac{\mu^2}{s+\mu}\ \tilde{P}(s;I,1)$$

$$- \frac{\mu^3}{(s+\mu)^2}\ [\tilde{P}(s;I,2)+2\tilde{P}(s;1,1,0)]. \qquad (4.17)$$

Finally from eqs.(4.14),(4.16),(4.17), we obtain that for Re s ≥ 0 the Laplace-Stieltjes transform of the total execution time distribution is given by :

$$T^*(s) = \frac{\mu^M}{(s+\mu)^2\ (s+2\mu)^{M-2}}\ [2a(1,1,0)$$

$$+ \sum_{k=2}^{D} a(1,k,1)\ \frac{(s+2\mu)^{k-1}}{(s+\mu)^{k-1}}\ ], \qquad (4.18)$$

where the $a(i,j,n_i)$'s are given by recurrence relations (4.15).

Differentiating $T^*(s)$ at point s = 0, we get the following expression for the expected total execution time

$$\overline{T} = \frac{1}{\mu}\ [a(1,1,0)\ \frac{M+2}{2^{M-2}}$$

$$+ \sum_{k=2}^{D} a(1,k,1)\ .\ \frac{(1+k+M)}{2^{M-k}}\ ]. \qquad (4.19)$$

5 - NUMERICAL RESULTS

We consider a full binary tree **A** of depth D (D≥1), where tasks execution times are exponentially distributed random variables with mean $1/\mu$ (μ>0). Let S(D) be the expected execution time of **A** with a single processor. We obviously have $S(D) = \frac{M}{\mu}$, where $M = 2^{D+1} - 1$ is the number of tasks of **A**.

Similarly let us define NA(D,N) [resp. A(D,N)] the expected execution time of **A** under the NA-policy [resp. A-policy] with N processors.

From Section 3, it is seen that (cf. 3.15),

$$NA(D,2) = \frac{1}{\mu}\ (2^D + \frac{D}{2})$$

and

$$NA(D,\infty) = \frac{1}{\mu}\ \Big(\sum_{i=0}^{D} (2^{-i} + \sum_{k=1}^{2^i-1} k^{-1})\Big).$$

On the other hand, A(D,2) is computed from eq.(4.19) where the $a(i,j,n_i)$'s are exactly determined using recursive relations (4.15).

We have found the following interesting results,

$$A(1,2) = \frac{1}{\mu}\ (2 + 0.5),$$

$$A(2,2) = \frac{1}{\mu}\ (2^2 + 0.75),$$

$$A(3,2) = \frac{1}{\mu}\ (2^3 + 0,7968750...),$$

$$A(4,2) = \frac{1}{\mu}\ (2^4 + 0,79785156...),$$

$$A(5,2) = \frac{1}{\mu}\ (2^5 + 0,79785172...),$$

$$A(6,2) = \frac{1}{\mu}\ (2^6 + 0,79785172...)\ \text{... etc.}$$

These results lead us to conjecture that

$$\lim_{D\to\infty} \Big(A(D,2) - \frac{2^D}{\mu}\Big) = \frac{\alpha}{\mu}$$ and that 0,79785172 is the eight digit decimal approximation of α.

In the case where an infinite number of processors is available (i.e. $N \geq 2^D$) let us define $F_i(x)$ the probability distribution of the execution time of a sub-tree rooted at level i, i = 0,1,..., D.

It is readily seen, from our assumptions, that

$$F_i(x) = \mu e^{-\mu x} * [F_{i+1}(s)]^2, \text{ for } i = 0,1,...,D-1,$$

$$F_D(x) = 1 - e^{-\mu x}\ , \qquad (n \geq 0)$$

where * denotes convolution.

We have found

$$A(2,\infty) = \frac{1}{\mu}\ (2^2 + 0{,}30555\ldots\ ),$$

$$A(3,\infty) = \frac{1}{\mu}\ (2^3 - 1{,}69247\ldots\ ),$$

$$A(4,\infty) = \frac{1}{\mu}\ (2^4 - 7{,}5582094\ldots\ ).$$

These three values have been exactly computed using the MACSYMA program for symbolic calculations.

For $D \geq 5$, we have used numerical methods based on Laguerre transforms (cf. [3], [5]).

For $\mu = 2$, we have plotted [Figure 2] $S(D)$, $NA(D,2)$, $NA(D,\infty)$, $A(D,2)$ and $A(D,\infty)$ for different values of D, the depth of the tree **A**. As already observed in [3], the mean processing time $A(D,\infty)$ grows almost linearly as the depth of the tree increases.

In order to estimate the gain achieved when adding processors to a single processor environment, we have also plotted [Figure 3] for different values of D, quantities $1 - \frac{X}{S(D)}$ with $X \in \{NA(D,2), NA(D,\infty), A(D,2), A(D,\infty)\}$, which can be interpreted as a kind of acceleration of the tree processing time.

From [Figure 3] we see that results obtained for the NA-policy are a good approximation of the corresponding results with the A-policy.

However the accuracy of this approximation will probably decrease for more general trees and for arbitrary distributions of task execution times.

6 - CONCLUSION

In a tentative to evaluate and to compare different policies for parallel execution of program tree structures, we have investigated two particular policies called the NA-policy and the A-policy.

In one case (NA-policy) results have been obtained for the distribution of the tree processing time for an arbitrary number of processors and for exponentially distributed task execution times.

In the other case (A-policy), similar results have been derived either for two available processors or for an infinite number of available processors.

Unfortunately less restrictive assumptions (eg. arbitrary non-lattice distributions for the execution times of tasks) yield theoretical difficulties and need further research investigations.

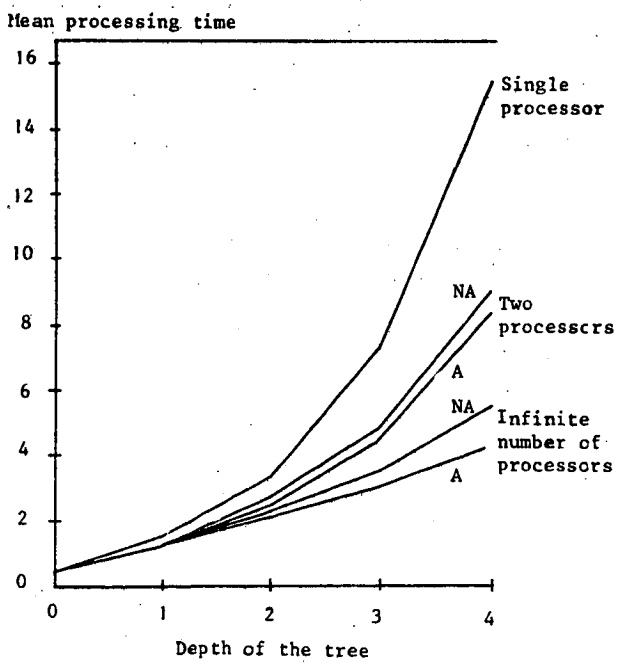


Figure 2 : Mean processing time

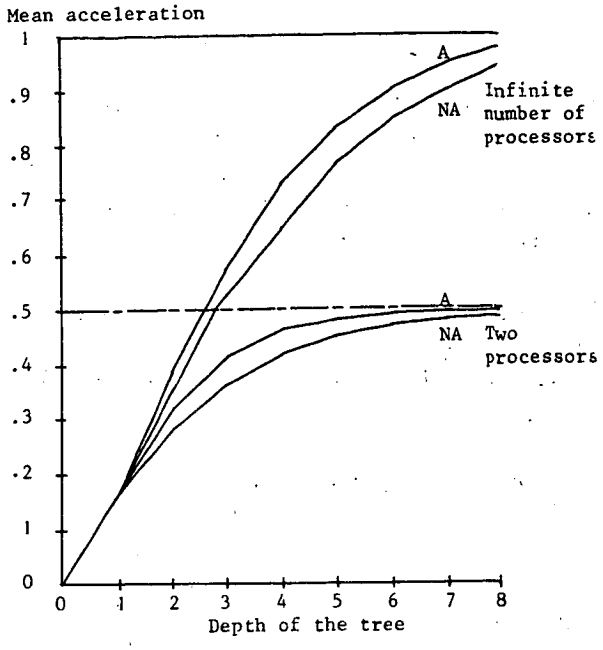


Figure 3 : Mean acceleration

REFERENCES

[1] Baccelli, F. and Fleury, T.,
"On parsing arithmetic expressions in a multiprocessing environment"
Acta Informatica, 17 (1982) pp. 287-310.

[2] Baccelli, F. and Mussi, P.,
"A decentralized control parallel interpreter for arithmetic expressions and its evaluation"
Proceedings of "Parallel Computing 83", Berlin, September 26,28 1983, North Holland.

[3] Fayolle, G., King, P.J.B. and Mitrani, I.,
"On the Execution of Programs by Many processors"
Proceedings of "Performance 83", Maryland, May 27,29 1983, North Holland.

[4] Gelenbe, E., Lichnewsky, A. and Staphylopatis, A.,
"Experience with the Parallel Solution of Partial Differential Equations on a Distributed Computing System".
IEEE Trans. on Computers, c-31, 12 (1982) pp.1157-1164.

[5] Keilson, J. and Nunn, W.R.,
"Laguerre transformation as a tool for the numerical solution of integral equations of convolution type"
Applied Mathematics and Computation, 5 (1979) pp. 313-359.

[6] Vuillemin, J.,
"A very fast multiplication algorithm for VLSI implementation"
INRIA Report, n° 183, 1983.

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique