Influence of overhead on processor allocation for multiple loops*

M. D. Gubitoso S. W. Song

Universidade de São Paulo
Instituto de Matemática e Estatística
R. Matão, 1010, CEP 05508-900 São Paulo, SP, Brazil
e-mail: {gubi,song}@ime.usp.br

Abstract

We consider two consecutive and independent *forall* loops and the strategy to allocate processors for their execution. One strategy is to execute each of the two loops consecutively, each time with all the available processors. Another strategy is to execute both loops simultaneously, each with a fraction of the available processors. We verify that the presence of overhead can influence this strategy, since the second strategy implies the use of a smaller number of processors for each individual loop, reducing thus the effect of the overhead. We establish conditions under which the second strategy is better. Finally we consider the special case when there is a single forall loop. We show conditions under which it is more advantageous to split it into two smaller loops and execute them simultaneously, each with a fraction of the available processors.

1 Introduction

In this paper we address the problem of allocating a total of n processors to the execution of two consecutive and independent *forall* loops, so that the execution time is minimized. This seems to be a simple problem. However, in the presence of overhead, it is not clear whether we should simply execute one loop after another, using all the processors for each of the loops, or partition the processors into two parts to execute both loops simultaneously. These two strategies will be referred to as the consecutive execution and the simultaneous execution strategies.

We show that the presence of overhead can influence the choice of the more suitable strategy. This is so because the simultaneous execution of the two loops implies the use of a smaller number of processors in each individual loop, reducing thus the efect of the

*Supported by FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo) - Proc. No. 93/0603-1, 94/4544-2, 95/0767-0 and CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico) - Proc. No. 523112/94-7 and PROTEM II, and Commission of the European Communities through Project ITDC-207.

overhead. We will establish conditions under which the simultaneous execution strategy is better.

We also consider the special case when there is a single forall loop. We show conditions under which it is more advantageous to split it into two smaller loops and execute them simultaneously, each with a fraction of the available processors.

Our work is motivated by the works of Flatt and Kennedy[5, 6]. Note that the use of an increasingly larger amount of processors could initially decrease the time to solve a problem. However, after some point, the use of too many processors could even take longer time than that required by a smaller number of processors[1, 5, 6, 8, 10]. This led Flatt and Kennedy to examine the problem of finding the number of processors that would result in the smallest execution time. Other related works concerning the performance of parallel systems and number of processors include [3, 4, 7].

In the next section we state the problem to be considered. In Section 3 we reproduce some preliminary results. In Section 4 we present the two execution strategies and establish conditions to choose the one with the smaller execution time. We consider cases without and with overhead. In Section 5 we consider the particular case of one large single forall loop. Again we give conditions to choose the better strategy. Finally in Section 6 we give concluding remarks.

2 Statement of the problem

We consider the problem of allocation of n processors to execute two consecutive independent forall loops.

forall $i = 1$ to m_1 do
begin
    statement S_1;
    statement S_2;
    $\vdots$
    statement S_s
end;
forall $i = 1$ to m_2 do
begin
    statement R_1;
    statement R_2;
    $\vdots$
    statement R_r
end

We assume that the two forall loops are independent. Therefore any of the two loops can be in principle be executed first, or both loops can be run simultaneously.

One way is to allocate all the n processors to execute the first loop and then the same n processors to execute the second loop.

Alternatively, we can divide the n processors into two groups of n_1 and n_2 processors each,

$$n = n_1 + n_2,$$

and run the first loop with n_1 processors and the second loop with n_2 processors, simultaneously.

One particular case of this problem is the following, with a single forall loop.

```
forall i = 1 to m1 + m2 do
begin
      statement S1;
      statement S2;
      ⋮
      statement Sk
end
```

As far as semantics is concerned, any single forall loop can be viewed as two consecutive and independent forall loops. Thus a solution to the original problem will be useful to this particular case of one single forall loop. As we shall see, it is sometimes more advantageous to consider a single forall loop as two smaller forall loops and run both simultaneously with n_1 and n_2 processors, respectively.

3 Preliminary results

The motivation of our study is the papers by Flatt and Kennedy[5, 6]. They present a kind of updated version of Amdahl's model or "law" [2], by including overheads. Their model views the sequential time (t_{seq}) to solve a problem as composed of two parts: one part must be executed on a single processor (t_s) and another part *could* be executed in parallel with two or more processors (t_p).

$$t_{seq} = t_s + t_p.$$

Flatt assumes that, in general, a parallel application can have its execution time viewed as the sum of three parts. One part is purely sequential (to be called sequential component t_s), one part is perfectly parallel that varies inversely with the number of processors (t_p/n) and a third part is the overhead that depends on n ($t_o(n)$). The total time, as function of the number of processors n, is therefore,

$$t(n) = t_s + \frac{t_p}{n} + t_o(n).$$

The speedup is defined as

$$S_n = \frac{t_{seq}}{t(n)}$$

If we ignore overhead, S_n is monotonically increasing, and for large n its growth is linear. However, taking into account the typical assumption that $t_o(n)$ increases with n, S_n should have a maximum.

Through a study of performance/cost, Flatt and Kennedy suggest an "ideal" value for the number of processors to be utilized. This number is, in general, much smaller than the one that gives the maximum speedup. The idea is that after a certain point, a great increase in the number of processors is needed to get a marginal gain on the speedup.

This fact of using a subset of processors for a certain task instead of all of the processors suggests a solution to our proposed problem, which we will now elaborate.

In the next section we consider the case of two consecutive and independent forall loops. We present conditions under which they should execute simultaneously, each one with a fraction of the available processors. This study will be done in two cases: with or without overheads.

4 Execution strategies and times

Given the problem of Section 2, we can consider two possibilities.

(a) Execute each of the two loops consecutively, each time with all the available processors (to be referred to as the *consecutive execution* of the two loops), or

(b) execute both loops at the same time, each with a fraction of the available processors (to referred to as the *simultaneous execution* of the two loops).

We consider each situation with or without overhead. Thus we have four distinct situations. For each of the four situations, we present the condition of minimum execution time.

We assume that the machine possesses n processors and we wish to make use of all of them. In the case of simultaneous execution of both loops, we assume the following.

αn processors will be used in the first loop and
βn processors in the second loop, with $\beta = 1 - \alpha$.

The time of each loop is given by

$$t_i(n) = t_{s_i} + t_{o_i}(n) + \frac{t_{p_i}}{n}, i \in \{1, 2\}.$$

In a "pure" forall loop, we ignore the sequential component ($t_{s_i} = 0$). In general, we assume the existence of a sequential component for our analysis.

4.1 Cases without overhead

In this case the execution time is

$$t(n) = t_s + \frac{t_p}{n}$$

(a) Consecutive execution of the two loops, each time with all the n processors.

This case is the usual option[9]. The total time when the loops are executed one after the other is given by:

$$\begin{aligned} T_I &= t_{s_1} + \frac{t_{p_1}}{n} + t_{s_2} + \frac{t_{p2}}{n} \\ &= t_{s_1} + t_{s_2} + \frac{t_{p_1} + t_{p_2}}{n} \end{aligned} \quad (1)$$

(b) Simultaneous execution of the two loops, each with a fraction of the available processors.

We have the following[1]:

$$\begin{aligned} T_{II} &= \max\left(t_{s_1} + \frac{t_{p_1}}{\alpha n},\, t_{s_2} + \frac{t_{p_2}}{\beta n}\right) \\ &= \max\left(t_1(\alpha n),\, t_2(\beta n)\right) \end{aligned} \quad (2)$$

Without loss of generality, we consider

$$t_1(\alpha n) \geq t_2(\beta n). \quad (3)$$

The condition for the simultaneous execution (case (b)) to be more advantageous than case (a) can be obtained by imposing $T_I \geq T_{II}$:

$$t_{s_1} + t_{s_2} + \frac{t_{p_1} + t_{p_2}}{n} \geq t_1(\alpha n)$$

that is,

$$t_{s_2} + \frac{t_{p_1} + t_{p_2}}{n} \geq \frac{t_{p_1}}{\alpha n} \quad (4)$$

$$t_{s_2} \geq \frac{\beta t_{p_1} - \alpha t_{p_2}}{\alpha n} \quad (5)$$

Assuming $t_{s_2} \ll \frac{t_{p_1} + t_{p_2}}{n}$, we can disregard the left hand side of (5), and we have

$$\begin{aligned} \alpha t_{p_2} &\geq \beta t_{p_1} \\ \frac{\alpha}{\beta} &\geq \frac{t_{p_1}}{t_{p_2}} \end{aligned}$$

[1] We can even have a better time, se we use $n-1$ processors during $|t_{s_1} - t_{s_2}|$ and n processors after $\min(t_1(n), t_2(n))$. We will not, however, consider this situation.

If $t_{p_1} = t_{p_2}$, we have equality with $\alpha = \beta$.

We should be careful with the above analysis because (3) depends on the value of α. Let us therefore calculate the minimum value of T_{II} for any α, satisfying (3).

According to (2) we have to consider the maximum of t_1 and t_2. t_1 is a monotonically decreasing function in α, with minimum in $\alpha = 1$, on the other hand, t_2 is monotonically increasing in α. Ignoring the limit cases where $t_1(n) > t_2(1)$ and $t_2(n) > t_1(1)$, the minimum is found through the equality of the two times:

$$\begin{aligned} t_1(\alpha n) &= t_2(\beta n) \\ t_{s_1} + \frac{t_{p_1}}{\alpha n} &= t_{s_2} + \frac{t_{p_2}}{\beta n}. \end{aligned}$$

In the particular case where $t_{s_1} = t_{s_2}$, α for the minimum time takes a simple value:

$$\alpha = \frac{t_{p_1}}{t_{p_1} + t_{p_2}}$$

and we can verify easily that the minimum time is equal to the sequential component. Under this situation it is indifferent to use one strategy or another. This is true for the case for the perfectly parallelizable forall ($t_{s_i} = 0$). This is not true, however, when there is a difference between the sequential components.

4.2 Case with overhead

We use the same kind of analysis for the case with overhead. The consecutive execution time is

$$\begin{aligned} T_I &= t_{s_1} + t_{o_1}(n) + \frac{t_{p_1}}{n} + t_{s_2} + t_{o_2}(n) + \frac{t_{p_2}}{n} \\ &= t_{s_1} + t_{s_2} + t_{o_1}(n) + t_{o_2}(n) + \frac{t_{p_1} + t_{p_2}}{n} \end{aligned} \tag{6}$$

To calculate the simultaneous execution time, we consider that all the processors are used ($\alpha + \beta = 1$), though in some cases it might be better to use a subset of the total, due to the overhead.

Thus we have:

$$\begin{aligned} T_{II} &= \max\left(t_{s_1} + t_{o_1}(\alpha n) + \frac{t_{p_1}}{\alpha n},\ t_{s_2} + t_{o_2}(\beta n) + \frac{t_{p_2}}{\beta n}\right) \\ &= \max\left(t_1(\alpha n),\ t_2(\beta n)\right) \end{aligned} \tag{7}$$

We can assume, without loss of generality, that

$$t_1(\alpha n) \geq t_2(\beta n).$$

Simultaneous execution will be better when $T_I \geq T_{II}$:

$$
\begin{aligned}
t_{s_2} + t_{o_1}(n) + t_{o_2}(n) + \frac{t_{p_1} + t_{p_2}}{n} &\geq t_{o_1}(\alpha n) + \frac{t_{p_1}}{\alpha n} \\
t_{s_2} + t_{o_1}(n) + t_{o_2}(n) - t_{o_1}(\alpha n) &\geq \frac{t_{p_1} - \alpha(t_{p_1} + t_{p_2})}{\alpha n} \qquad (8)
\end{aligned}
$$

Let us make the typical assumption that the overhead increases with n. Observe immediately that the left side of the inequality (8) increases with n while the right side decreases, for any α. The conclusion is that, if overhead is increasing, inequality (8) is always verified for a sufficiently large number of processors. Therefore, under such conditions, simultaneous execution of the two loops each with a fraction of the processors is always better than the usual consecutive execution.

5 Particular cases

It is interesting to analyze the inequality (8) with particular overhead models and verify the case of one large forall loop (as seen in Section 2). Let us examine some overhead models.

5.1 Constant model

This is the simplest overhead model. Overhead does not vary with the number of processors. Notice that in this case the observation at the end of last section does not apply.

Let us rewrite (8), using $t_{o_1}(n) = t_{o_1}(\alpha n) = t_{o_1}$ and $t_{o_2}(n) = t_{o_2}$, we have the condition that favors simultaneous execution.

$$
t_{o_2} \geq \frac{t_{p_1} - \alpha(t_{p_1} + t_{p_2})}{\alpha n} - t_{s_2} \qquad (9)
$$

Recall that we are assuming $T_I \geq T_{II}$. We have found a lower bound for the overhead of the faster loop with the chosen distribution of processors. Obviously we can consider the inverse problem and find α such as to satisfy (9), in an analogous way as in Section 4.1.

5.2 Linear model

In the linear model the overhead is $t_o(n) = an + b$. This is a very used model, especially when the overhead is basically due to communication.

The left side of (8) becomes:

$$
t_{s_2} + t_{o_1}((1-\alpha)n) + t_{o_2}(n) - b
$$

Let us write $T_{o_i}(n)$ as $a_i n + b_i$. Since these parameters a_i and b_i in general depend on the machine, we can assume $a_1 = a_2 = a$ and $b_1 = b_2 = b$.

Then

$$
\begin{aligned}
t_{o_1}((1-\alpha)n) + t_{o_2}(n) &= \\
&(2-\alpha)an + 2b
\end{aligned}
$$

and (8) becomes

$$t_{s_2} + (2-\alpha)an + 2b \;\geq\; \frac{t_{p_1} - \alpha(t_{p_1} + t_{p_2})}{\alpha n}$$

5.3 One single loop

As mentioned in Section 2, we consider the case of one single forall loop. We present conditions under which it is better to split it into two smaller loops, of the same size, to be executed simultaneously.

This problem makes sense if there exists some kind of overhead, for instance, balancing, access of data outside the processor, etc.

For such a loop we have:

$$\begin{aligned} t_{s_1} &= t_{s_2} = 0 \\ t_{p_1} &= t_{p_2} \\ t_{o_1} &= t_{o_2} \end{aligned}$$

The inequality (8) becomes very simple:

$$2t_o(n) - t_o(\alpha n) \;\geq\; \frac{t_p(1-2\alpha)}{\alpha n}$$

We see that even under the constant model, the left side can be greater than the right side with a not too large number of processors.

6 Conclusion

We studied the case of two consecutive and indpendent forall loops and the strategy to allocate processors for their execution. The two strategies considered are: (a) The consecutive execution strategy – execute each of the two loops consecutively, each time with all the available processors, or (b) The simultaneous execution strategy – execute both loops at the same time, each with a fraction of the available processors.

We verified that the presence of overhead can influence the choice of the better strategy. This is so because the simultaneous execution of the two loops implies the use of a smaller number of processors for each individual loop, reducing thus the efect of the overhead.

When there is no overhead, there is no significant difference to choose one strategy or another. In particular, under the situation where the sequential components of the loops are equal, it is always possible to discover a partitioning of the processors to make the consecutive and simultaneous execution times equal.

In the case of existence of overhead, we established conditions under which the simultaneous execution strategy is better. For a sufficiently large number of processors, the simultaneous execution is better.

We then considered the special case when there is a single forall loop. We show conditions under which it is more advantageous to split it into two smaller loops and execute them simultaneously, each with a fraction of the available processors.

Note that the same approach can be used to analyze the more general case of k consecutive loops.

References

[1] Adams, L. M. and Crokett, T. W. Modelling algorithm execution time on processor arrays. *Computer* **17**, No. 7, 38-43 (1984).

[2] Amdahl, G. M. Validity of the single processor approach to achieving large scale computer capabilities. *Proc. AFIPS Computer Conference* **30**, 483 (1967).

[3] Cytron, R. Useful parallelism in a multiprocessing environment. *Proceedings of the International Conference on Parallel Processing*, IEEE Computer Society Press, 450-457 (1985).

[4] Eager, D. L., Zahorjan, J. and Lazowska, E. D. Speedup versus efficiency in parallel systems. *IEEE Trans. Computers* **38**, No. 3, 408-423 (1989).

[5] Flatt, H. P. Further results using the overdead model for parallel systems. *IBM Journal Res. Development* **35**, No. 5/6, 721-726 (1991).

[6] Flatt, H. P. and Kennedy, K. Performance of parallel processors. *Parallel Computing* **12**, No. 1, 1-20 (1989).

[7] Gustafson, J. L., Montry, G. R. and Benner, R. E. Development of parallel methods for a 1024-processor hypercube.

[8] M. Lehman. A survey of problems and preliminary results concerning parallel processing and parallel processors. *Proc. IEEE* **54**, 1889-1901 (1966).

[9] Polychronopoulos, C. D. *Parallel Programming and Compilers*, Kluwer Academic Publishers, Boston, 1988.

[10] Rosenfeld, J. L. A case study in programming for parallel processors. *Comm. ACM* **12**, No. 12, 645-655 (1969).