Effect of Granularity on the Time Performance of the Multiprocessor Systems

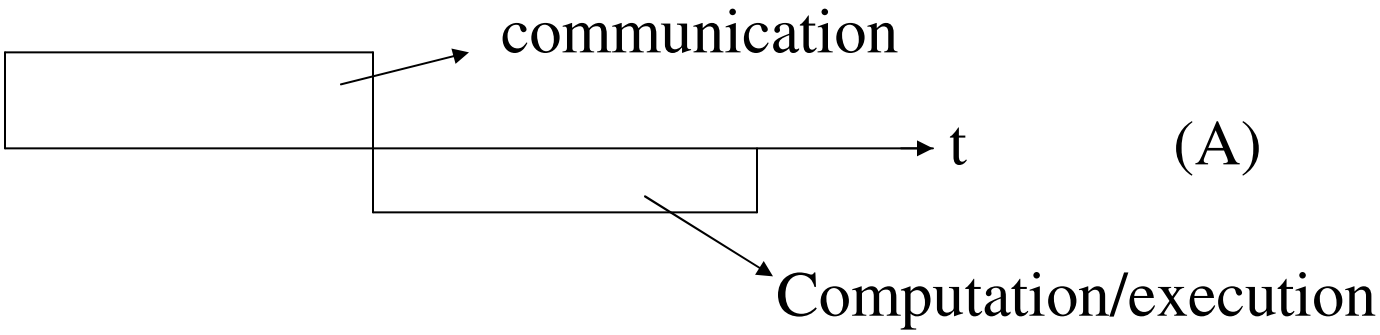
R/C is a measure of the granularity
R : length of the run time quantum **C**: Overhead associated with the communication for that quantum

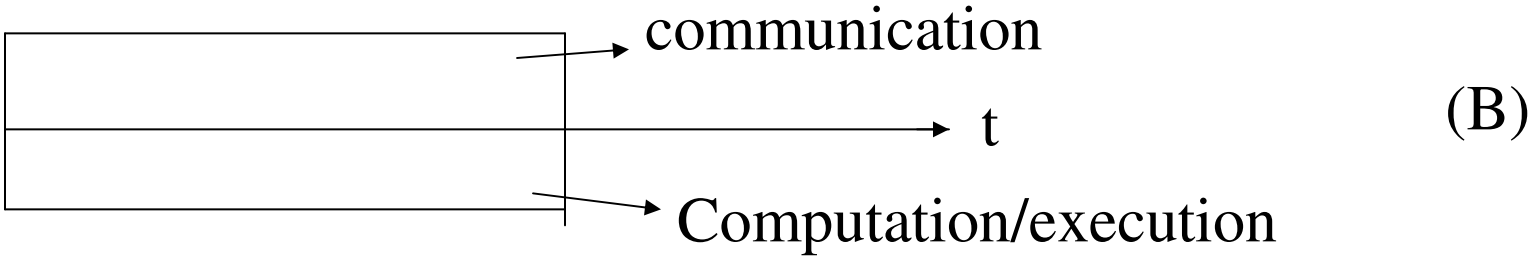
Coarse grain : **R/C** is high Fine grain : **R/C** is low

We will present two different models to analyze the performance of the multiprocessor systems including the granularity of the tasks

Assumptions : (1). Each task executes in R units of time
(2). Each task communicates with every other task with an overhead of C units, whenever the communicating tasks reside on different processors

Communication and Computation can be in a time overlapped fashion. We will discuss on (A). Non-overlapped case and (B). Overlapped case. We consider a two processor system first.





(A). Non-Overlapped Case: The performance of this case can be described by the following equation.

$$\text{Execution Time} : T(k) = R\ \mathbf{Max}\{M-k,k\} + C\ (M-k)k \quad (1)$$

Equation (1) describes the total time to complete M tasks on a system with two processors. The first term describes the computation time when **k** tasks are scheduled on one processor and the rest of the (**M-k**) are scheduled on a different processor. Note that the first term is a linear function of **k**. The second term describes the effect of communication between the processors. Note that this is a quadratic term due to the fact that every task communicates with every other task, i.e., accounting the number of pair-wise communications.

It is interesting to investigate on the behavior of the execution time as a function of **k** and also to determine an **optimal k** that gives the minimum execution time. Solution to this problem can be obtained by analyzing the time performance as shown in the figures.

From the figures, we observe the following. In Fig. 1 (next slide) we see that the *communication overheads* are much more severe than in Fig.2.

First term in (1) is linear function of **k**, and is symmetrical about the point **k=M/2**. When this linear term is added to the quadratic term, the resulting curve has a minimum at **M/2** in Fig. 2 and has a minimum at **k=0** (k=M)in Fig. 1.

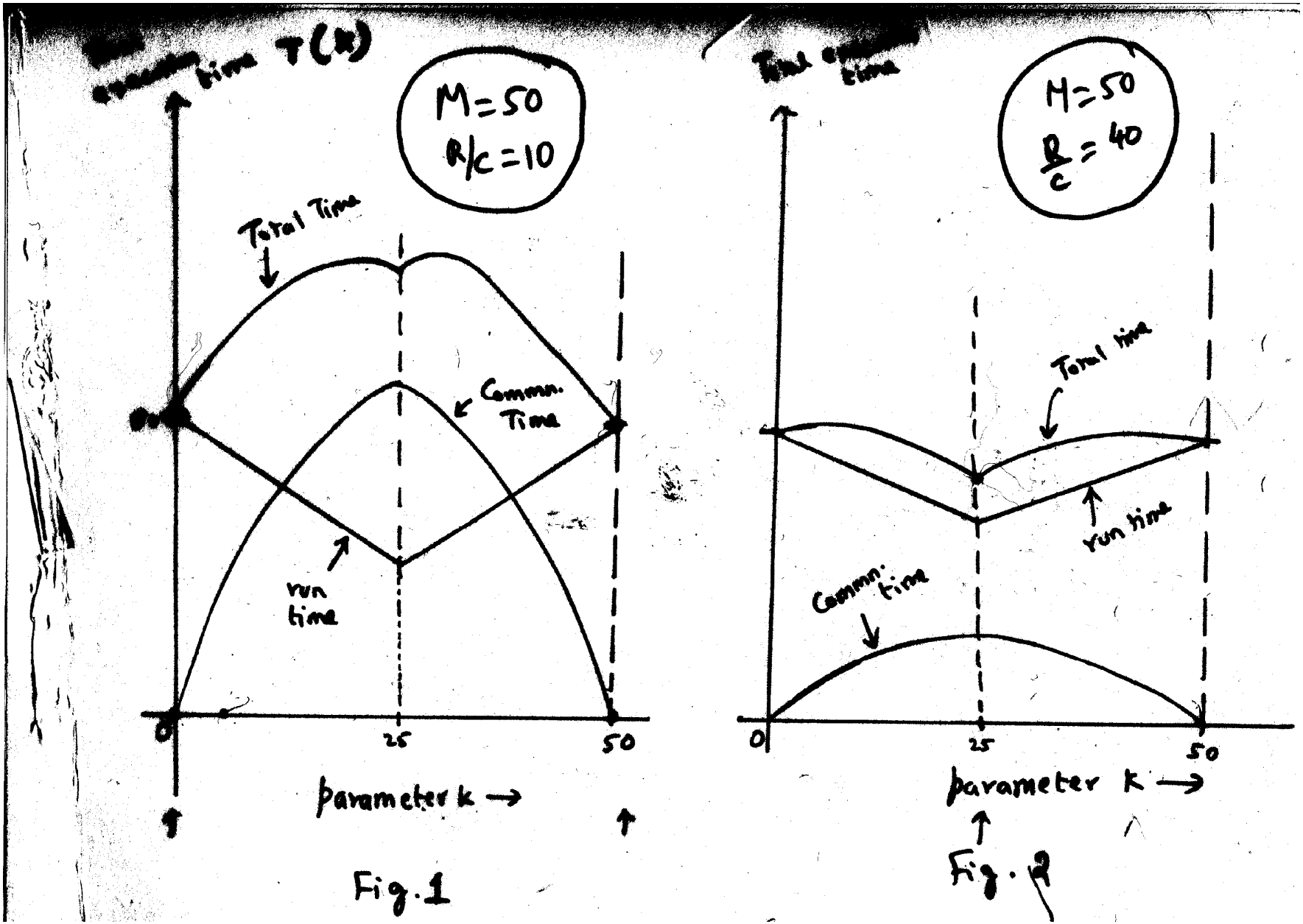


Fig.1

Fig. 2

What we see here is that, whenever **R/C < M/2**, the minimum occurs at the value **k=0**, implies that it is *optimal* to schedule all the M tasks on a single processor. On the other hand, whenever **R/C >= M/2**, the minimum occurs at **k=M/2**, implies that assigning equal loads to the processors is *optimal*.

The same line of argument can be applied for the ***N*** processors case as follows.

When there are N processors in the system, we have the following equation describing the behavior of the execution time.

$$\text{Execution time } T(k) = R \max_{i} \{k[i]\} + (C/2) \left(\sum_{i} k[i](M-k[i])\right) \quad (2)$$

where the maximization is over all i=1,…,M. Simplifying, we have,

$$T(k) = R \max_{i}(k[i]) + (C/2)\,(M^2 - \sum_{i} k[i]k[i]) \qquad (3)$$

where we allow only distinct pair-wise communications between tasks

Note that the first term accounts for the maximum computation time among all the N processors.

The second term accounts for a distinct pair-wise communication interaction among the tasks, each contributing an overhead of C units.

We observe that the same rationale for the case of two processors also hold in this case.

What is more interesting is the case when **R/C** value is more than **M/2**. Note that the case when **R/C < M/2** is trivial and needs a single processor for optimality. Now, considering the difference in the total execution times with N processors and a single processor, for **R/C >= M/**2 and for very large **N**, we have,

$$T_{diff} = [\ R*M/N + C*M^2/2 - C*M^2/(2*N)\ ] - R*M \qquad (4)$$

We have assumed that M is an integral multiple of **N**. Now, to solve for the threshold value of R/C, we equate (4) to zero, and obtain, **R/C = M/2**. This shows that, if **R/C >= M/2**, then an even distribution of tasks to as many processors available will fetch optimal finish time. On the other hand, if **R/C < M/2**, no matter how large **N** is, scheduling all the **M** tasks on a single processor will fetch the optimal finish time.

As the performance is usually determined by the ***speed-up*** factor, in our case, we have,

Divide the Nr & Dr by C & bring N to the Nr;

$$\underline{Speed\text{-}up} = R*M \,/\, (RM/N + C*M^2/2 - CM^2/2N)$$
$$= (RN/C) \,/\, (R/C + M(N-1)/2) \qquad (5)$$

(Hint : Use the definition of speed-up and the fact that N is very large)

Discussion

Now, if **R/C >> M*(N-1)/2**, then we achieve a speed-up proportional to N. This means that we need **M** and **N** to be as small as possible for **R/C** to be too large. Suppose, if we need more parallelism with a large **N**, the second term in the denominator is larger than the first term, the speed-up is proportional to **R/(C*M),** which does not depend on the number of processors. Hence, as **N** increases, the speed-up asymptotically approaches this value. At this point, each processor added to the system is of no benefit!

Thus, the so far analysis has demonstrated the effect of granularity and the overhead factor on the time performance. It shows the importance of choosing the right granularity which minimizes the total 'cost' of the system. One can use a variety of models to analyze the multiprocessor performance. We will now consider the case when we have a time overlapped computation and communication operations.

(B). Overlapped Case : This is an *optimistic* model in which the communication phase is completely masked by the computation. To realize this, we have the communication term to be overlapped with the computation term as much as possible. We describe such a behavior as,

$$\text{Execution Time} = \textbf{Max}_{\mathbf{i}} \{ R \max(k[i]), (C/2)\ k[i](M-k[i]) \} \qquad (6)$$

Worth considering assignments

•Analyze the performance for a two-processor system for the overlapped case

• Suppose if have a model in which the cost of communication is proportional to the number of processors, and not to the number of tasks assigned remotely. How will be the performance?

Execution time = R Max $\{ k_i \}$ + C N

Describe the performance when more processors are added to the system.

Good luck!