An Upper Bound on the Improvement of Asynchronous versus Synchronous Distributed Processing*

Robert E. Felderman and Leonard Kleinrock
UCLA Computer Science Department
3732L Boelter Hall
Los Angeles, CA 90024-1596

Abstract

We use simple models of two distributed processing methods, one asynchronous, the other synchronous, to calculate the maximum potential performance gain of the former over the latter. We show, in the limit as the number of tasks grows and the number of processors increases, that the asynchronous method has an expected potential speedup over the synchronous method of no more than $\ln P$ where P is the number of processors used by each strategy.

1 Introduction

We compare two synchronization methods used in distributed processing systems and determine how much better one performs than the other. Our motivation comes from the area of Parallel Discrete Event Simulation (PDES) which has received much attention recently [Misra 1986] [Jefferson 1985]. There are several algorithms used for PDES and this paper demonstrates the potential improvement by using an asynchronous approach (e.g. Time Warp), over a synchronous technique, (e.g. time-stepped simulation). We first give an introduction to PDES, discuss briefly the two methods chosen for comparison, and then follow with our models and analysis. Readers who are unfamiliar with Discrete Event Simulation and techniques used to parallelize it are referred to [Misra 1986] [Jefferson 1985] [Peacock et al. 1979] for more details.

2 Parallel Discrete Event Simulation

Parallel Discrete Event Simulation is generally accomplished by partitioning the simulation into logical processes (LP) or *entities* each of which simulates some physical process in the system. An example

*This work was supported by the Defense Advanced Research Projects Agency under Contract MDA 903-87-C0663, Parallel Systems Laboratory.

Figure 1: Example queueing network

is the simple queueing network shown in Figure 1. Entities in our system are the customer arrival process(A), the servers(B,C,D,E) and a final sink process(F) to collect departing customers. Each process receives messages, performs internal computations and sends messages to other processes. Each LP maintains a local clock which indicates the current time of the simulation at that entity, and a process terminates once its local or logical clock (the simulation time of the message currently being processed) has reached T_{max}, the total time of the simulation (a user specified duration). One can think of each logical process as residing on a separate processor, but this is not necessary. In fact, all the logical processes may reside on a single processor. LPs operate independently and communicate with each other only if the physical processes being simulated by the LPs are connected. For example, logical process A (LP_A) connects to LP_B which is in turn connected to LP_C and LP_D etc. Every path which can be traversed by a customer in the physical system must correspond to a logical communication path in the simulation system. Messages passed between LPs in our queueing example are the actual customers flowing through the system.

Each logical process could be placed on its own processor, and one might hope that we could then gain speedup proportional to the number of processors used. Unfortunately, this is often not the case as the system being simulated may have only limited parallelism. Also, the PDES algorithms themselves limit parallelism in their attempt to prevent the simulation from deadlocking and to ensure correctness.

Several competing techniques have been developed to address deadlocking and correctness. One [Peacock et al. 1979] is described as a synchronous approach which keeps logical process clocks in synchronization while another [Jefferson 1985] is an asynchronous strategy which uses a rollback mechanism which is invoked only when needed for synchronization.

2.1 Time-Stepped Simulation

Distributed time-stepped simulation [Peacock et al. 1979] is accomplished by keeping all the local clocks in strict synchronization. At any point in real time each LP's local clock will have the same value as any other LP's clock. As the simulation runs, the local clocks take on a sequence of discrete values $(t_0, t_1, t_2, \ldots)$ each differing by an amount Δ. All processors must complete execution of events up to t_i before any processor begins processing at t_{i+1}. Each processor may have a different amount of work to do at each time step or some may operate at different speeds so that many processors may have to wait for the slowest one to complete execution of the i^{th} step. Time-stepped simulation is attractive due to its simplicity of implementation. By keeping all the LPs processing at the same simulation time, deadlocks cannot occur and no further effort needs to be expended in guaranteeing the correctness of the simulation. Time-stepped simulation is an example of the synchronous approach.

2.2 Time Warp

Our asynchronous example comes from one of the more recent developments in the area of PDES; the so-called optimistic strategies. One such strategy is called Time Warp and was developed by Jefferson [Jefferson 1985]. The basic idea is that the requirement of keeping each LP in strict synchronization (keeping local clocks the same), even when it isn't necessary, may lead to a degradation in performance. The Time Warp mechanism allows LPs to race forward as quickly as possible. If a message arrives which has a lower timestamp than the value of the LP's clock, indicating the LP proceeded with incomplete information, the LP is "rolled back" to the time of the incoming message. This can be accomplished because the system periodically saves the state of the LP. Any effects of having advanced too far (i.e. erroneous messages) are canceled through an elegant technique using anti-messages [Jefferson 1985].

3 The Models

We have opted to use very simple models of the two approaches in order to assess the potential improvement of the asynchronous versus the synchronous strategy. Our model of the time-stepped (synchronous) strategy will provide us with an accurate estimate of its time to complete a simulation, while the model for the asynchronous strategy will provide us with an overly low estimate of its expected completion time. Therefore, we establish an upper bound on the potential improvement of the asynchronous strategy over the time-stepped method.

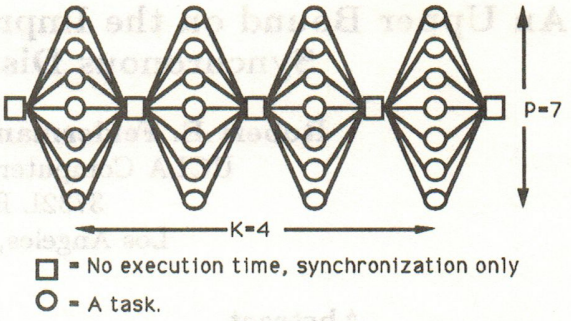

Figure 2: Synchronous Task Graph

To analyze the two techniques, we propose the following model: P processors each execute K tasks (events) sequentially. Each processor p must perform tasks $T_{p1} \cdots T_{pk} \cdots T_{pK}$ in sequential order. K determines the "size" of the simulation. A task will take a random amount of time to complete execution on any processor.

Our model of the synchronous approach is based on the idea that an LP must wait for all other LPs to complete a step before continuing. Each processor must wait until *every* processor has completed its i^{th} task prior to beginning execution of the $(i+1)^{st}$ task. This is essentially a "staged" execution with K stages where each stage takes as long as the slowest processor. This task graph is shown in Figure 2 for $K = 4$ and $P = 7$.

The asynchronous strategy has no such "staging" restriction, and, moreover, in the best possible circumstance no rollbacks will occur. We allow each processor to execute its tasks in order as fast as it can, without waiting for the other processors to finish. The total time to finish is simply the time that the slowest processor takes to complete its K tasks. To keep the model simple we are assuming no rollbacks; it is as if each processor never has to wait for any messages from other processors, and that all messages arrive in timestamp order. The asynchronous task graph is shown in Figure 3.

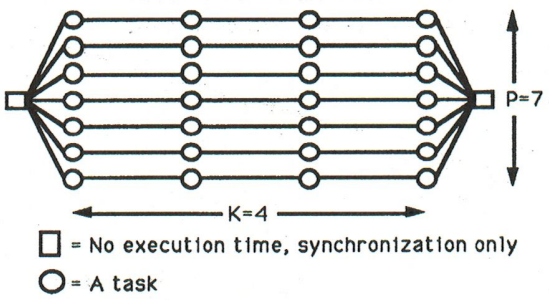

Figure 3: Asynchronous Task Graph

4 Space of Synchronization Methods

Though the models we are using are extremely simple, we believe they provide us with very important information. Our time-stepped model (Figure 2) requires the most synchronization, and therefore will take the longest time to complete execution of any system which exhibits full parallelism in each stage. Our asynchronous model (Figure 3) shows the least amount of internal synchronization (none) and should complete execution in less time than any other method. Therefore, we believe that these two models span the range of possibilities and give a good indication of the maximum performance improvement that could be gained by using the asynchronous strategy.

5 Exponentially Distributed Task Times

If we model each task execution time with an exponential distribution and treat the processors as identical, the expected time for the synchronous strategy is K times the maximum of P exponentials, while the expected time for the asynchronous strategy is the maximum of P K-stage Erlangs. We now proceed to calculate the ratio R_e of the expected completion times for exponentially distributed task times.

5.1 Time-Stepped (Synchronous) Model

Let T = the maximum of P exponentials with mean $\frac{1}{\mu}$. The cumulative distribution (PDF) of T is

$$F_T(x) = (1 - e^{-\mu x})^P, \tag{1}$$

with density function

$$f_T(x) = P(1 - e^{-\mu x})^{P-1} \mu e^{-\mu x}. \tag{2}$$

Using the PDF we can calculate the expected value of T [Kleinrock 1975].

$$\begin{aligned} E[T] &= \int_0^\infty (1 - F_T(x))dx \\ &= \int_0^\infty \left[1 - (1 - e^{-\mu x})^P\right] dx \\ &= \int_0^\infty \left[1 - \sum_{i=0}^{P} \binom{P}{i} 1^{P-i}(-e^{-\mu x})^i\right] dx \\ &= \sum_{i=1}^{P} \binom{P}{i} (-1)^{i+1} \int_0^\infty (e^{-\mu i x})dx \end{aligned}$$

Since [Graham et al. 1989]

$$\sum_{i=1}^{P} \binom{P}{i} \frac{1}{i}(-1)^{i+1} = \sum_{i=1}^{P} \frac{1}{i},$$

$$E[T] = \frac{1}{\mu} \sum_{i=1}^{P} \frac{1}{i}. \tag{3}$$

We now define T_s as the time for all K stages to complete. Clearly, $E[T_s] = KE[T]$. So the final equation for $E[T_s]$ where P is the number of processors and K is the number of steps is:

$$E[T_s] = \frac{K}{\mu} \sum_{i=1}^{P} \frac{1}{i}. \tag{4}$$

An excellent approximation for this is [Jolley 1961]:

$$E[T_s] \approx \frac{K}{\mu}\left(E + \ln P + \frac{1}{2P} - \frac{1/12}{P(P+1)}\right). \tag{5}$$

where E = Euler's Constant ≈ 0.57722.

5.2 Time Warp (Asynchronous) Model

We define T_a as the maximum of P K-stage Erlangs where each stage has mean $\frac{1}{\mu}$. The probability density function of a single K-stage Erlang is

$$f_t(x) = \frac{\mu e^{-\mu x}(\mu x)^{K-1}}{(K-1)!}. \tag{6}$$

The PDF can be found either by direct integration of the density function, or by realizing that the probability that a K-stage Erlang takes time less than or equal to t is one minus the probability that it takes time greater than t, which is simply one minus the probability that there are less than K arrivals in the interval $[0 - t]$ from a Poisson process at rate μ. Therefore

$$F_t(x) = \int_0^x f_t(x)dx = 1 - e^{-\mu x} \sum_{i=0}^{K-1} \frac{(\mu x)^i}{i!}. \tag{7}$$

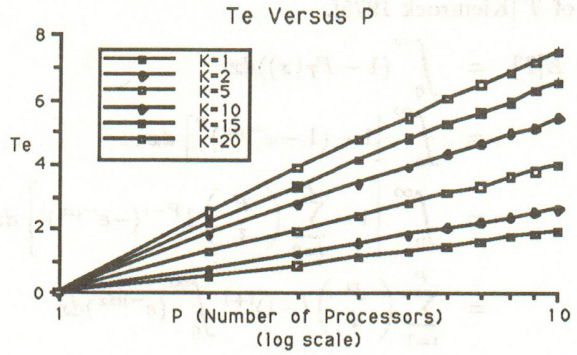

Figure 4: T_e versus $\ln P$.

The cumulative distribution of the maximum of P K-stage Erlangs is:

$$F_{T_a}(x) = \left(1 - e^{-\mu x} \sum_{i=0}^{K-1} \frac{(\mu x)^i}{i!}\right)^P. \quad (8)$$

Using $F_{T_a}(x)$ we can calculate the expectation of T_a.

$$E[T_a] = \int_0^\infty [1 - F_{T_a}(x)]\, dx$$

Thus,

$$E[T_a] = \int_0^\infty 1 - \left(1 - e^{-\mu x} \sum_{i=0}^{K-1} \frac{(\mu x)^i}{i!}\right)^P dx$$

$$= \int_0^\infty \sum_{j=1}^{P} \binom{P}{j} (-1)^{j+1} \left(e^{-\mu x} \sum_{i=0}^{K-1} \frac{(\mu x)^i}{i!}\right)^j dx$$

Unfortunately, this equation has no closed form expression for the integral. By decomposing $E[T_a]$ into two components: the mean of a K-stage Erlang and $T_e \triangleq$ the difference between the mean and the expected value of the max of P K-stage Erlangs, we can approximate $E[T_a]$.

$$\begin{aligned} E[T_a] &= \text{Mean of } K\text{-stage Erlang} + T_e \\ &= \frac{K}{\mu} + T_e \end{aligned}$$

An excellent approximation for T_e when $K > 1$ and $P > 1$ is

$$T_e \approx \frac{1}{\mu}\left((C\ln^2 K + D)\ln P + AK + B\right). \quad (9)$$

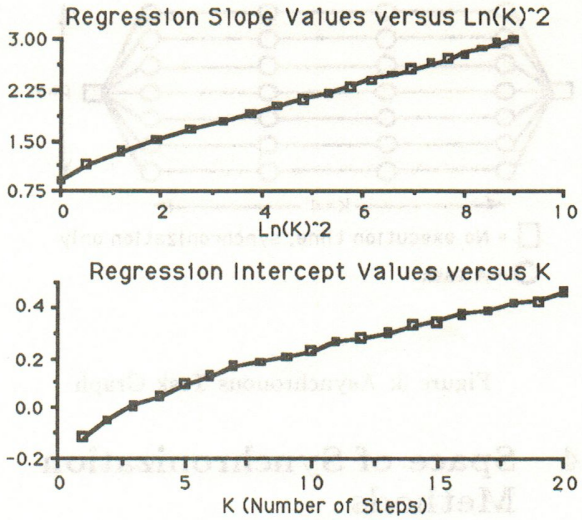

Figure 5: Regression Values

Where

$$A = 0.02244 \approx 0.02 \quad B = 1.14571 \approx 1.15$$
$$C = 0.22278 \approx 0.22 \quad D = 0.55957 \approx 0.56.$$

This approximation was developed by using least squares regression techniques three times. It was first noticed that for a fixed K, T_e seemed to be directly related to $\ln P$. This is clearly seen in Figure 4. For each value of $k \leq K$ we performed a linear regression for T_e such that $(T_e)_k = m_k(\ln P) + b_k$, thus generating K slopes and intercepts, one set for each value of k. Then, it appeared that the slopes for each k approximation were linearly related to $\ln^2 k$, while the intercepts seemed linearly related to k. This can be seen in Figure 5. Therefore, a second regression was performed on the slope values versus $\ln^2 k$ (generates the values for the constants C and D), while a third regression was performed on the intercept values versus k (generates the values for the constants A and B). Figure 6 shows the approximation compared to simulation for values of K and P between one and ten and Figure 7 shows the comparison for $K, P >= 100$.

5.3 Relative Performance

Let us look at the ratio, R_e of the two approximations.

$$E[T_s] \approx \frac{K}{\mu}\left(E + \ln P + \frac{1}{2P} - \frac{1/12}{P(P+1)}\right)$$

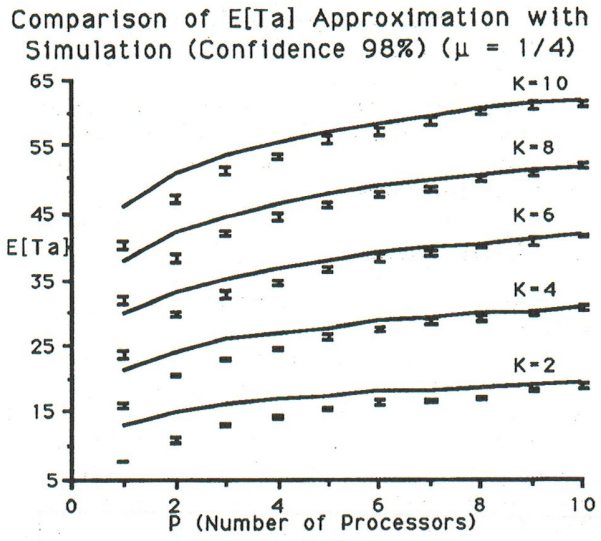

Figure 6: Comparison of Approximation and Simulation for $K \leq 10$.

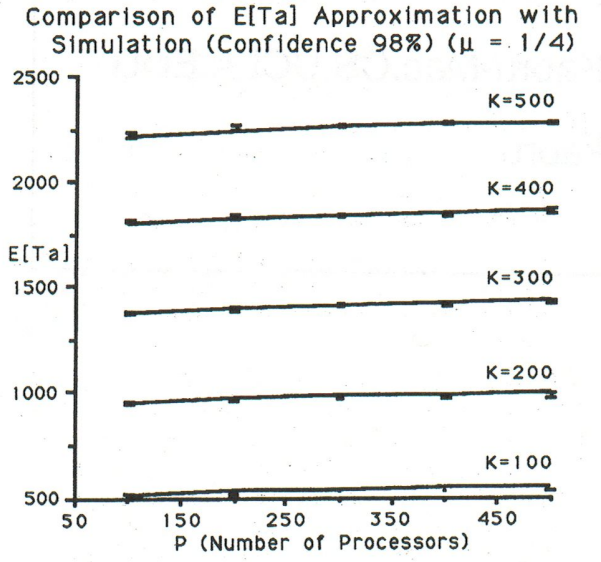

Figure 7: Comparison of Approximation and Simulation for $K \geq 100$.

$$E[T_a] \approx \frac{K}{\mu} + \frac{1}{\mu}\left((C\ln^2 K + D)\ln P + AK + B\right)$$

$$\begin{aligned} R_e &= \frac{E[T_s]}{E[T_a]} \\ &= \frac{\frac{K}{\mu}\left(E + \ln(P) + \frac{1}{2P} - \frac{1/12}{P(P+1)}\right)}{\frac{1}{\mu}\left((C\ln^2(K) + D)\ln(P) + (1+A)K + B\right)} \\ &= \frac{E + \ln P + \frac{1}{2P} - \frac{1/12}{P(P+1)}}{\frac{(C\ln^2 K + D)\ln P}{K} + (1+A) + \frac{B}{K}} \end{aligned}$$

Taking the limit as the size of the simulation increases ($K \rightarrow \infty$) and assuming that $K >> \ln P$ we get

$$\lim_{K\to\infty} R_e = \frac{E + \ln P + \frac{1}{2P} - \frac{1/12}{P(P+1)}}{(1+A)}.$$

Finally, for large P

$$\lim_{K\to\infty, P\to\infty} R_e \approx \frac{\ln P}{(1+A)} \approx \frac{\ln P}{1.02} \approx \ln P. \quad (10)$$

Thus, in the limit as the size of the simulation increases to infinity, the asynchronous approach could, at most, complete $\ln P$ times as fast as the time-stepped method on average. We can derive this result by appealing to intuition. We have exponential task times where each task takes, on average, $\frac{1}{\mu}$ seconds to complete. For synchronized execution, basic principles (Section 5.1) tell us that each stage will take time proportional to $\frac{1}{\mu}\ln P$ on the average for a total expected time of $\frac{K}{\mu}\ln P$. The asynchronous execution on average takes time equal to $\frac{K}{\mu}$ plus T_e a term which is small compared to $\frac{K}{\mu}$ for large K. Therefore, the ratio of the two times should be $\ln P$. It should be noted that a trivial lower bound on T_a (thus an upper bound on R_e) is found by simply using the mean of P K-stage Erlangs. Our approximation (Equation 9) confirms this result since A,B,C and D are all non-negative.

Additionally, if we believe that no method could achieve a speedup greater than P for P processors over execution on a single processor, then any time-stepped method is limited to a maximum speedup of $\frac{P}{\ln P}$. These results depend on the assumption of an exponential distribution for task times. The next section uses a uniform [0-X] distribution for task execution times.

6 Uniformly Distributed Task Times

If we make the assumption that the task times are uniformly distributed between 0 and X, we calculate

a different limiting value for the ratio of completion times. It is easy to show that the maximum of P uniformly distributed random variables is $X\frac{P}{P+1}$. We immediately find that

$$E[T_s] = KX\frac{P}{P+1}. \quad (11)$$

Fortuitously, we can use the same regression technique used with the exponential distribution in Section 5.2 to develop an accurate approximation for $E[T_a]$. Therefore

$$E[T_a] = \frac{KX}{2} + T_e$$

$$\approx \frac{KX}{2} + X\left((C\ln^2 K + D)\ln P + AK + B\right). \quad (12)$$

Where

$$A = 0.012384 \approx 0.01 \quad B = 0.330691 \approx 0.33$$
$$C = 0.053147 \approx 0.05 \quad D = 0.125102 \approx 0.13.$$

Finally, we look at the ratio of the expected completion times.

$$R_u = \frac{E[T_s]}{E[T_a]}$$

$$= \frac{XK\frac{P}{P+1}}{X\left((C\ln^2(K) + D)\ln(P) + (A+1/2)K + B\right)}$$

$$= \frac{\frac{P}{P+1}}{\frac{(C\ln^2 K + D)\ln P}{K} + (A+1/2) + \frac{B}{K}}$$

Taking the limit as the size of the simulation increases ($K \to \infty$) and assuming that $K >> \ln P$ we get

$$\lim_{K\to\infty} R_u = \frac{P}{(A+1/2)(P+1)}.$$

Finally, for large P

$$\lim_{K\to\infty, P\to\infty} R_u = \frac{1}{(A+1/2)} \approx \frac{1}{0.51} \approx 2. \quad (13)$$

Thus, when using a uniform distribution, the asynchronous strategy is only able to complete in roughly half the time, regardless of the number of processors used (as compared to our previous case where the task execution times had exponential tails and the asynchronous strategy was able to gain its $\ln P$ performance improvement). This result should apply to any distribution with finite support since the maximum of many such random variables will invariably approach the upper limit.

Again, we can appeal to intuition to find the speedup ratio. For large P the synchronized execution will take X seconds per stage (the max) on average for a total time of KX. The asynchronous system, on average, will take time equal to $\frac{KX}{2} + T_e$. Since T_e is small compared to $\frac{KX}{2}$ for large K, the speedup ratio should be 2.

As before, if we assume that no method can achieve speedup greater than P over a sequential execution, then the synchronous strategy could possibly have speedup proportional to P when the task times are uniformly distributed.

7 Conclusions

We have shown that an asynchronous distributed simulation strategy can have at most a $\ln P$ performance improvement over a time-stepped method, in the case where task times are exponentials. We conjecture that this result is due to the infinite tail on the exponential distribution and may therefore be applicable to distributions with exponential tails. The improvement when using a distribution with finite support (e.g. uniform) is reduced to a constant amount independent of P.

References

[1] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics.* Addison-Wesley Publishing Co., 1989.

[2] David R. Jefferson. Virtual time. *ACM Transactions on Programming Languages and Systems*, 7(3), July 1985.

[3] L.B.W. Jolley. *Summation of Series.* Dover Publications, Inc., second revised edition, 1961.

[4] Leonard Kleinrock. *Queueing Systems: Volume 1: Theory.* John Wiley and Sons, Inc., 1975.

[5] Jayadev Misra. Distributed discrete-event simulation. *Computing Surveys*, 18(1), March 1986.

[6] J. Kent Peacock, J.W. Wong, and Eric G. Manning. Distributed simulation using a network of processors. *Computer Networks*, 3(1):44–56, 1979.

Bob Felderman was born in Chicago, Illinois in 1962. He graduated Magna Cum Laude from Princeton University in 1984 with a double major in Electrical Engineering & Computer Science and Systems Engineering. After spending a year at Hughes Aircraft Company working on guidance systems for torpedos, he returned to the good life of academia, completed his Master's degree in Computer Science at UCLA in 1986 and is currently pursuing a Ph.D. in Computer Science specializing in distributed systems. In his spare time he can be found in the great outdoors usually with frisbee in hand.