Two Processor Time Warp Analysis: Some Results on a Unifying Approach*

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

Abstract

We present some results from an exact analysis of a new model for the problem of two processors running the Time Warp distributed simulation protocol. The model creates a unifying framework for previous work in this area and additionally provides some clear insight into the operation of systems synchronized by rollback.

1 Introduction

Distributed simulation has proven to be an important application for parallel processing. Accordingly, several algorithms have been developed to perform simulation with multiple processors. The most well known techniques are generally classified into two types; conservative [8] and optimistic [2]. Generally speaking, optimistic simulation on multiple processors is a technique which allows each processor to proceed with its portion of a simulation independent of the other processors (optimistically assuming that the others will not interact with that processor); if, at a later time, it finds that some other processor caused its earlier assumption to be false, it will roll back and proceed forward again. Our research focuses on the analysis of the average case behavior of Time Warp, the most well known optimistic technique.

Very little work has appeared in the literature which discusses average case behavior of Time Warp (TW). Lavenberg et al. [5] and Mitra and Mitrani [9] have examined models similar to ours, and we will address their relationship to this work in Section 5. Recently, Lin and Lazowska [6] have examined Time Warp and conservative methods by appealing to critical path analysis. Though their work provides important insights, it generates different types of results than ours. Finally, Madisetti [7] provides bounds on the performance of a two processor system where the processors have different speeds of processing and move at constant rates. Madisetti extends his model to multiple processors, something we do not address in this work.

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

The next section introduces our model for Time Warp. Section 3 provides its exact solution while in Section 4 we derive some performance measures. In Section 5 we examine the model as we take limits on various parameters and discuss the relationship of this work to that of Lavenberg et al. and Mitra and Mitrani. Section 6 discusses what we can learn from the model. Finally, in Section 7 we provide some concluding remarks and notes on future research directions.

2 A Model for Two Time Warp Processors

Assume we have a job which is partitioned into two processes, each of which is executed on a separate processor. As these processes are executed, we consider that they advance along the integers on the x-axis in discrete steps, each beginning at $x = 0$ at time $t = 0$. Each process independently makes jumps forward on the axis where the size of the jump is geometrically distributed with mean $1/\beta_i$ $(i = 1, 2)$ The amount of real time between jumps is a geometrically distributed number of time slots with parameter α_i $(i = 1, 2)$. After process i makes an advance along the axis, it will send a message to the other process with probability q_i $(i = 1, 2)$. Upon receiving a message from the other (sending) process, this (receiving) process will do the following:

Case 1: If its position along the x-axis is equal to or behind the sending process, it will ignore the message.

Case 2: If it is ahead of the sending process, it will immediately move back (i.e., "rollback") along the x-axis to the current position of the sending process.

Let $F(t)$= the position of the First process (process one) at time t and let $S(t)$= the position of the Second process (process two) at time t. Further, let

$$D(t) = F(t) - S(t).$$

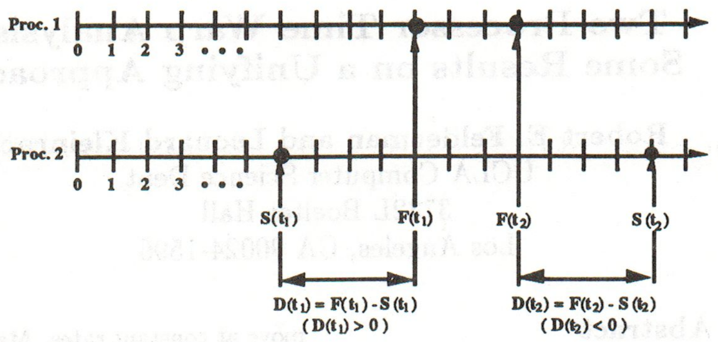

Figure 1: States of two processors at times t_1 and t_2.

$D(t) = 0$ whenever Case 2 occurs (i.e., a rollback). We are interested in studying the Markov process $D(t)$. From our assumptions that $F(0) = S(0) = 0$, we have $D(0) = 0$. Clearly, $D(t)$ can take on any integer value (i.e., it certainly can go negative, see Figure 1). We will solve for

$$\begin{aligned} p_k &= \lim_{t\to\infty} P[D(t) = k] \quad k = 0,1,2,\ldots \\ n_k &= \lim_{t\to\infty} P[D(t) = -k] \quad k = 1,2,3,\ldots \end{aligned}$$

namely, the equilibrium probability for the Markov chain $D(t)$. Moreover, we will find the speedup with which the computation proceeds when using two processors relative to the use of a single processor.

This is a simple model of the Time Warp distributed simulation algorithm where two processors are both working on a simulation job in an effort to speed it up. They both proceed independently until such time as one (behind) process transmits a message in the "past" of the other (ahead) process. This causes the faster process to "rollback" to the point where the slower process is located, after which they advance independently again until the next rollback, etc. The interpretation of the model is that the position of each process on the axis is the value of the local clock (or virtual time of the message being processed) of each process. The amount of real time to execute a particular event is modelled by the geometric distribution of time slots between jumps. The jumps in virtual time indicate the increase in the virtual timestamp from one event to the next. Messages passed between processors (with probability q_i) have virtual time stamps equal to the virtual time of the sending process. Our model assumes that states are stored after every event, otherwise a rollback would not necessarily send the processor back to the time of the tardy message; rather it might have to roll back to a much earlier time, namely, that of the last saved state. Another implicit assumption is that each process always schedules events for itself. Finally, the interaction between the processes is probabilistic.

3 Discrete Time, Discrete State Analysis

In this section we provide the exact solution for the discrete time, discrete state model introduced in Section 2. Although, as we proceed, the equations may look formidable, the analysis is quite straightforward. First, we provide some definitions.

$$\begin{aligned}
\alpha_i &= \text{P[}i^{th} \text{ processor advances in a time slot]} \\
\overline{\alpha_i} &= 1 - \alpha_i \\
A_1 &= \alpha_1\overline{\alpha}_2 \text{ (Only proc. 1 advances)} \\
A_2 &= \alpha_2\overline{\alpha}_1 \text{ (Only proc. 2 advances)} \\
A_3 &= \alpha_1\alpha_2 \text{ (Both advance)} \\
A_4 &= \overline{\alpha_1}\,\overline{\alpha_2} \text{ (Neither advance)} \\
g_j &= \text{P[processor 1 advances } j \text{ units]} \\
&= \beta_1\overline{\beta}_1^{\,j-1} (j > 0) \quad (\overline{\beta}_1 = 1 - \beta_1) \\
f_j &= \text{P[processor 2 advances } j \text{ units]} \\
&= \beta_2\overline{\beta}_2^{\,j-1} (j > 0) \quad (\overline{\beta}_2 = 1 - \beta_2) \\
\gamma &= \text{P[procs. 1 and 2 advance the same dist.]} \\
&= \frac{\beta_1\beta_2}{1 - \overline{\beta}_1\overline{\beta}_2} \\
q_i &= \text{P[} i^{th} \text{ proc. sends a message]} \\
\overline{q}_i &= 1 - q_i
\end{aligned}$$

Since the transitions in our system are quite complex (there are an infinite number of transitions into and out of each state) we choose to show the state diagram only for a simplified version of our system where $\beta_1 = \beta_2 = 1$ in Figure 2. This is the case where

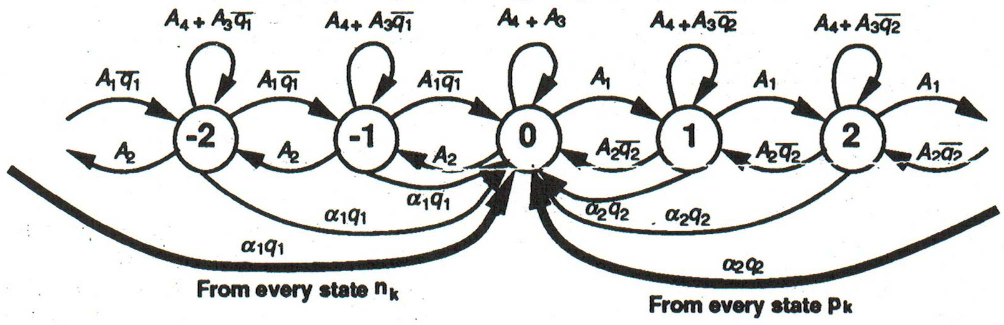

Figure 2: State Diagram for $\beta_1 = \beta_2 = 1$.

the processors only make jumps of a single step (from k to $k+1$).

The balance equations for our completely general system (no restrictions on β_i) are:

$$
\begin{aligned}
&(A_1 + A_2 + A_3((1-\gamma) + q_2\gamma))\, p_k = \\
&= A_1 \left(\sum_{i=0}^{k-1} p_i g_{k-i} + \sum_{i=1}^{\infty} n_i g_{k+i} \right) \\
&\quad + A_2\overline{q}_2 \sum_{i=k+1}^{\infty} p_i f_{i-k} \\
&\quad + A_3\overline{q}_2 \sum_{i=k+1}^{\infty} p_i \sum_{j=1}^{\infty} g_j f_{j+i-k} \\
&\quad + A_3\overline{q}_2 \sum_{i=0}^{k-1} p_i \sum_{j=1}^{\infty} g_{j+k-i} f_j \\
&\quad + A_3\overline{q}_2 \sum_{i=1}^{\infty} n_i \sum_{j=1}^{\infty} g_{j+k+i} f_j \qquad k \ge 1
\end{aligned}
$$

$$
\begin{aligned}
&(A_1 + A_2 + A_3((1-\gamma) + q_1\gamma))\, n_k = \\
&= A_2 \left(\sum_{i=0}^{k-1} n_i f_{k-i} + \sum_{i=1}^{\infty} p_i f_{k+i} \right) \\
&\quad + A_1\overline{q}_1 \sum_{i=k+1}^{\infty} n_i g_{i-k} \\
&\quad + A_3\overline{q}_1 \sum_{i=k+1}^{\infty} n_i \sum_{j=1}^{\infty} f_j g_{j+i-k} \\
&\quad + A_3\overline{q}_1 \sum_{i=0}^{k-1} n_i \sum_{j=1}^{\infty} f_{j+k-i} g_j \\
&\quad + A_3\overline{q}_1 \sum_{i=1}^{\infty} p_i \sum_{j=1}^{\infty} f_{j+k+i} g_j \qquad k \ge 1
\end{aligned}
$$

$$p_0 = 1 - \sum_{i=1}^{\infty} p_i - \sum_{i=1}^{\infty} n_i .$$

By using the technique of z-transforms we are able to solve explicitly for p_k and n_k. We only give the results here; the full analysis will be found in a forthcoming paper [4].

We obtain

$$p_k = C_p p_0 \left(\frac{1}{r_1}\right)^k \quad k \ge 1 \tag{1}$$

$$n_k = C_n p_0 \left(\frac{1}{s_1}\right)^k \quad k \ge 1 \tag{2}$$

$$p_0 = \frac{1}{1 + \left(\frac{C_p}{r_1 - 1}\right) + \left(\frac{C_n}{s_1 - 1}\right)}. \tag{3}$$

Where

$$
\begin{aligned}
C_p &= \frac{\beta_1 D_p (1 + K_n)}{(1 - K_p K_n)(1 - \overline{\beta}_1 r_2)(A_1 + \overline{\beta}_1(A_2 + A_3))} \\
C_n &= \frac{\beta_2 D_n (1 + K_p)}{(1 - K_p K_n)(1 - \overline{\beta}_2 s_2)(A_2 + \overline{\beta}_2(A_1 + A_3))}
\end{aligned}
$$

$$
\begin{aligned}
K_p &= \frac{\beta_1 \overline{\beta}_2 D_p}{(1 - \overline{\beta}_1 r_2)(A_1 + \overline{\beta}_1(A_2 + A_3))(r_1 - \overline{\beta}_2)} \\
K_n &= \frac{\beta_2 \overline{\beta}_1 D_n}{(1 - \overline{\beta}_2 s_2)(A_2 + \overline{\beta}_2(A_1 + A_3))(s_1 - \overline{\beta}_1)}
\end{aligned}
$$

$$
\begin{aligned}
D_p &= (A_3 \overline{\beta}_1 \beta_2 \overline{q}_2 + A_1(1 - \overline{\beta}_1 \overline{\beta}_2)) \\
D_n &= (A_3 \overline{\beta}_2 \beta_1 \overline{q}_1 + A_2(1 - \overline{\beta}_1 \overline{\beta}_2))
\end{aligned}
$$

$$(r_1, r_2) = \frac{a_p \pm \sqrt{{b_p}^2 - 4a_p c_p}}{2a_p}$$

$$\begin{aligned}
a_p &= A_1 + \overline{\beta}_1(A_2 + A_3) \\
b_p &= -\left((A_1 + A_2 + A_3)(1 + \overline{\beta}_1\overline{\beta}_2) + A_1\beta_1\overline{\beta}_2 \right. \\
&\quad \left. + \beta_2\overline{q}_2(A_2\overline{\beta}_1 - A_3\beta_1)\right) \\
c_p &= \overline{\beta}_2(A_1 + A_2 + A_3) + A_2\beta_2\overline{q}_2
\end{aligned}$$

$$(s_1, s_2) = \frac{a_n \pm \sqrt{{b_n}^2 - 4a_n c_n}}{2a_n}$$

$$\begin{aligned}
a_n &= A_2 + \overline{\beta}_2(A_1 + A_3) \\
b_n &= -\left((A_1 + A_2 + A_3)(1 + \overline{\beta}_1\overline{\beta}_2) + A_2\beta_2\overline{\beta}_1 \right. \\
&\quad \left. + \beta_1\overline{q}_1(A_1\overline{\beta}_2 - A_3\beta_2)\right) \\
c_n &= \overline{\beta}_1(A_1 + A_2 + A_3) + A_1\beta_1\overline{q}_1.
\end{aligned}$$

4 Performance Measures

With the complete solution to the Markov chain in hand, we calculate several interesting quantities. The first is $\overline{K}_i$ which is defined as the average distance that processor i is ahead of the other. This measure is useful in getting a fix on the number of states which will need to be saved on average.

$$\begin{aligned}
\overline{K}_1 &= \sum_{k=0}^{\infty} kp_k + 0 \cdot \sum_{k=1}^{\infty} n_k = \frac{C_p p_0 r_1}{(r_1 - 1)^2} \\
\overline{K}_2 &= \sum_{k=1}^{\infty} kn_k + 0 \cdot \sum_{k=0}^{\infty} p_k = \frac{C_n p_0 s_1}{(s_1 - 1)^2}
\end{aligned}$$

Since the average size of a state jump at processor i is $1/\beta_i$ then average number state buffers needed at processor i is $\overline{K}_i\beta_i$.

Another useful measure, Θ_b^1, is the probability that processor one is ahead of processor two by more than b units. This measure is exactly the probability that a fixed size state buffer of size b at processor one overflows if $\beta_1 = \beta_2 = 1$ (if only single steps forward are allowed).

$$\begin{aligned}
\Theta_b^1 &= \sum_{k=b+1}^{\infty} p_k \\
&= 1 - p_0 - \sum_{k=1}^{b} p_k - \sum_{k=1}^{\infty} n_k \\
&= 1 - p_0 - \frac{p_0 C_p \left(r_1 - \left(\frac{1}{r_1}\right)^b\right)}{r_1 - 1} - \frac{p_0 C_n}{s_1 - 1}
\end{aligned}$$

A similar (symmetric) value, Θ_b^2, can be found for processor two. The quantity of most interest though is speedup, and we calculate its value in the next section.

4.1 Speedup

Using the formulas for p_k and n_k we can calculate the speedup S when using two processors versus using only one. S is simply the rate of virtual time progress per real time step when using two processors (R_2) divided by the rate of progress when using only one processor (R_1). The rate of forward progress for one processor is defined simply as the average rate of progress of the two processes

$$R_1 = \frac{\frac{\alpha_1}{\beta_1} + \frac{\alpha_2}{\beta_2}}{2}.$$

The rate of forward progress for two processors is the expected "unfettered" progress (without rollbacks) per time step minus the expected rollback distance per time step for the two processors.

$$\begin{aligned}
R_2 = \; & \frac{A_1}{\beta_1} + \frac{A_2}{\beta_2} + A_3\left(\frac{1}{\beta_1} + \frac{1}{\beta_2}\right) \\
& - A_3 q_2 p_0 \sum_{i=2}^{\infty} g_i \sum_{j=1}^{i-1} f_j (i - j) \\
& - A_3 q_1 p_0 \sum_{i=2}^{\infty} f_i \sum_{j=1}^{i-1} g_j (i - j) \\
& - A_2 q_2 \sum_{k=1}^{\infty} p_k \sum_{i=1}^{k-1} i f_{k-i} \\
& - A_1 q_1 \sum_{k=1}^{\infty} n_k \sum_{i=1}^{k-1} i g_{k-i} \\
& - A_3 q_2 \sum_{k=1}^{\infty} p_k \sum_{i=1}^{\infty} g_i \sum_{j=1}^{k+i-1} j f_{k+i-j} \\
& - A_3 q_1 \sum_{k=1}^{\infty} n_k \sum_{i=1}^{\infty} f_i \sum_{j=1}^{k+i-1} j g_{k+i-j} \\
& - A_3 q_1 \sum_{k=1}^{\infty} p_k \sum_{i=1}^{\infty} g_i \sum_{j=1}^{\infty} j f_{k+i+j} \\
& - A_3 q_2 \sum_{k=1}^{\infty} n_k \sum_{i=1}^{\infty} f_i \sum_{j=1}^{\infty} j g_{k+i+j}
\end{aligned}$$

As with the p_k calculation we omit the derivation of the following result. Combining all the terms together we find the formula for speedup.

$$
\begin{aligned}
S = \left(\frac{2}{\frac{\alpha_1}{\beta_1}+\frac{\alpha_2}{\beta_2}}\right)\Bigg[&\frac{A_1}{\beta_1}+\frac{A_2}{\beta_2}+A_3\left(\frac{1}{\beta_1}+\frac{1}{\beta_2}\right) \\
&-\frac{A_3\overline{\beta}_1\beta_2 q_2 p_0}{\beta_1(1-\overline{\beta}_1\overline{\beta}_2)}-\frac{A_3\overline{\beta}_2\beta_1 q_1 p_0}{\beta_2(1-\overline{\beta}_1\overline{\beta}_2)} \\
&-\frac{A_2 q_2\beta_1 C_p p_0 r_1}{(r_1-\overline{\beta}_1)(r_1-1)^2}-\frac{A_1 q_1\beta_2 C_n p_0 s_1}{(s_1-\overline{\beta}_2)(s_1-1)^2} \\
&-\frac{A_3 q_2\beta_2 C_p p_0}{(1-\overline{\beta}_1\overline{\beta}_2)(r_1-1)^2}\left(\frac{(r_1-\overline{\beta}_1)}{\beta_1}+\frac{\beta_1\overline{\beta}_2 r_1}{(r_1-\overline{\beta}_2)}\right) \\
&-\frac{A_3 q_1\beta_1 C_n p_0}{(1-\overline{\beta}_1\overline{\beta}_2)(s_1-1)^2}\left(\frac{(s_1-\overline{\beta}_2)}{\beta_2}+\frac{\beta_2\overline{\beta}_1 s_1}{(s_1-\overline{\beta}_1)}\right) \\
&-\frac{A_3 p_0}{1-\overline{\beta}_1\overline{\beta}_2}\left(\frac{q_1\beta_1\overline{\beta}_2^2 C_p}{\beta_2(r_1-\overline{\beta}_2)}+\frac{q_2\beta_2\overline{\beta}_1^2 C_n}{\beta_1(s_1-\overline{\beta}_1)}\right)\Bigg]
\end{aligned}
$$

5 Limiting Behavior

The reason we chose to use a discrete time, discrete state (DD) model was to allow ourselves to take limits on the α and β parameters thus creating models which are continuous in time and state (whereby geometric distributions become exponential distributions). We omit the actual formulae here due to space considerations; see [4] for the full details.

We can transform our model into a continuous time, discrete state (CD) model by taking the limit as α_1 and $\alpha_2 \rightarrow 0$ while keeping the ratio $\frac{\alpha_1}{\alpha_2}$ constant and defining $\frac{A_1}{A_1+A_2} = a$. We can take the limit either on the p_k equations or on our formula for speedup.

Alternatively, we create a discrete time, continuous state (DC) model by taking the limit as β_1 and $\beta_2 \rightarrow 0$ while keeping $\frac{\beta_1}{\beta_2} = b$. We find the value for speedup by taking limits on the speedup formula calculated for the discrete time, discrete state model.

Finally, we can solve a continuous time, continuous state (CC) model by taking limits on α_i and β_i simultaneously. This can be done either by going first to the CD (α_i) or DC (β_i) model from DD, and then finishing by taking limits on the other variable.

5.1 Previous Work on 2-Processor Models

There has been some similar work on two-processor Time Warp models. Lavenberg, Muntz and Samadi [5] used a continuous time, continuous state model to solve for the speedup (S) of two processors over one processor. Their work resulted in an approximation for S which was valid only for $0 \leq q_i \leq 0.05$. Remember that q_i is the interaction parameter; the probability that processor i will send a message to the other processor. Their result is only valid for very weakly interacting processes. Our result for this CC case has no restrictions on any of the parameters and therefore subsumes their work. In fact, we can compare our results directly for a simplified case where $a = 1/2$ (same processing rate for both processors), $b = 1$ (same average jump in virtual time for both) and $q_1 = q_2 = q$ (same probability of sending a message), which is the completely symmetric case. Lavenberg et al. derive the following approximation for speedup:

$$S_L \approx 2 - \sqrt{2q}.$$

Our equation for speedup in this restricted case is:

$$S = \frac{4\left(\sqrt{q}\,(5+q)+(1+q)\sqrt{8+q}\right)}{\sqrt{q}\,(2+q)(7+q)+\sqrt{8+q}\,(2+5q+q^2)}.$$

If we expand this formula using a power series about the point $q = 0$ and list only the first few terms, we see the essential difference between our result and Lavenberg et al.

$$S \approx 2 - \sqrt{2q} + \frac{q}{2} + O(q^{\frac{3}{2}})$$

This clearly shows that our result matches Lavenberg et al. in the first two terms. We see that their result is only accurate for very small values of q as they mention in their paper. Figure 3 shows the Lavenberg et al. result and our result compared to simulation with 99% confidence intervals.

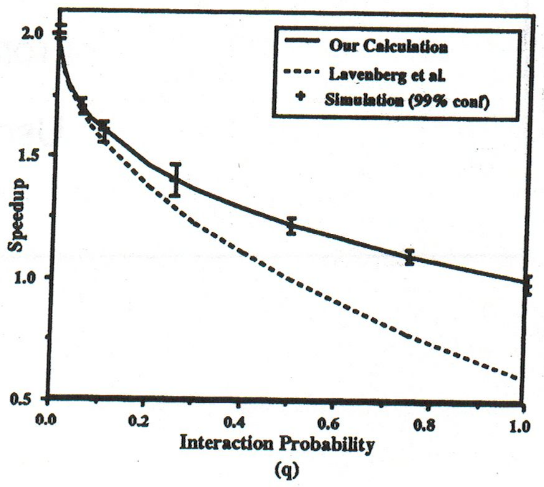

Figure 3: Comparison of speedup results for a simplified case.

Mitra and Mitrani [9] also solve a two-processor model but use a discrete time, continuous state approach to solve for the distribution of the separation between the two processors and the rate of progress of the two processors. In the definition of their model, a processor sends a message (with probability q_i) **before** advancing. Our model has a processor send a message **after** advancing. This difference between the two models disappears in the calculation of the average rate of progress. Their solution allows a general continuous distribution for the state jumps (virtual time), but requires (deterministic) single steps for the discrete time. In our model this is equivalent to setting $\alpha_1 = \alpha_2 = 1$. Since our analysis only supports an exponential distribution for state changes, but their analysis doesn't have a distribution on time, neither model subsumes the other.

Finally, the DD and CD models do not seem to have appeared in the literature, although a simplified version of the CD model where $\beta_1 = \beta_2 = 1$ (a preliminary version of this work which only allowed single-step state jumps) has been published by Kleinrock[3] . Figure 4 shows how all of this work fits together. The work discussed in this paper covers the shaded region.

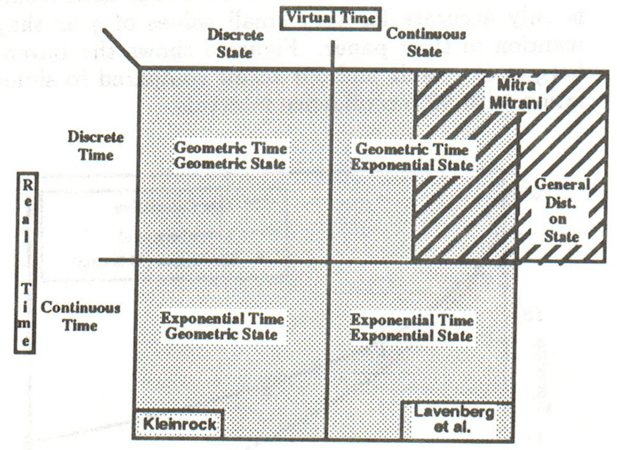

Figure 4: Previous work.

6 Results for a Restricted Model

In order to better understand our results, we examine a restricted version of the CD model (i.e. the model analyzed in [3]). In this less general model we eliminate two variables by forcing the processors to advance **exactly** one step each time they advance ($\beta_1 = \beta_2 = 1$). Again, we define q_i as the interaction parameter; the probability that processor i sends a message to the other processor. We also define a as the ratio $\frac{\lambda_1}{\lambda_1+\lambda_2}$ where λ_i is the rate for the continuous time distribution for processor i (rate at which messages are processed). Figure 5 shows the speedup for the Symmetric case where $q_1 = q_2 = q$. Figure 6 shows the speedup for the Balanced case

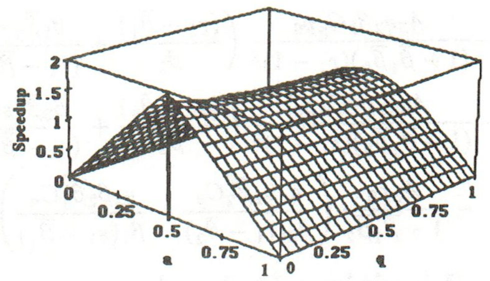

Figure 5: Speedup for the Symmetric case $q_1 = q_2 = q$

where $\lambda_1 = \lambda_2$. Figure 7 shows the speedup for

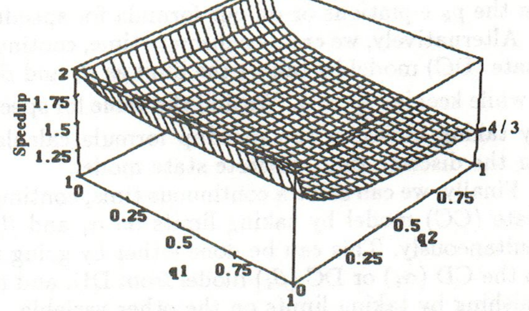

Figure 6: Speedup for the Balanced case $\lambda_1 = \lambda_2 = \lambda$.

the extremely simplified Symmetric, Balanced case where $q_1 = q_2 = q$ and $\lambda_1 = \lambda_2 = \lambda$. For this special case the formula for speedup is

$$S = \frac{4}{2 + \sqrt{q}}.$$

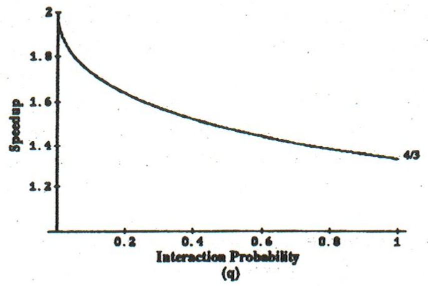

Figure 7: Speedup for the Symmetric, Balanced case $q_1 = q_2 = q$ and $\lambda_1 = \lambda_2 = \lambda$.

6.1 Optimality Proofs

Using the simple model described above, we can prove several results about optimality with respect to the parameters of the system. We first show that the speedup is monotonically decreasing in both q_1 and q_2, the interaction parameters. We do this by showing that $\frac{\partial S}{\partial q_1}$ is negative. First, a definition:

$$h(q_1) = (1 - 4a\overline{a}q_1).$$

If we differentiate S with respect to q_1 we arrive at the following formula

$$\frac{\partial S}{\partial q_1} = \Phi \cdot \left(-(-1+2a)^2 - 2a\overline{a}q_1 + (1-2a)\sqrt{h(q_1)}\right),$$

where Φ is a non-negative function of q_1,q_2, and a.

In order to show that $\frac{\partial S}{\partial q_1}$ is negative, we must show that

$$-(-1+2a)^2 - 2a\overline{a}q_1 + (1-2a)\sqrt{h(q_1)} \leq 0. \quad (4)$$

When $a \geq \frac{1}{2}$ Equation 4 is trivially solved. Our concern is in the range $0 \leq a < \frac{1}{2}$, in which case our condition becomes:

$$-(-1+2a)^2 - 2a\overline{a}q_1 + (1-2a)\sqrt{h(q_1)} \leq 0$$

$$\begin{aligned} -(-1+2a)^2 - 2a\overline{a}q_1 &\leq -(1-2a)\sqrt{h(q_1)} < 0 \\ \left(-(-1+2a)^2 - 2a\overline{a}q_1\right)^2 &\geq \left(-(1-2a)\sqrt{h(q_1)}\right)^2. \end{aligned}$$

Expanding and simplifying, our condition reduces to

$$4a^2\overline{a}^2{q_1}^2 \geq 0.$$

Since a and $\overline{a}$ and q_1 are all non-negative, the inequality holds. A similar (symmetric) proof for q_2 is omitted here.

Optimization with respect to a is a little more difficult. When we differentiate S with respect to a we get such a complicated formula that it is prohibitive to solve for the optimum value of a. Fortunately, by plotting S versus a, q_1 and q_2 we see that S is unimodal and that the optimum value of a is 1/2 ($\lambda_1 = \lambda_2$). When we plug this value ($a = 1/2$) into $\frac{\partial S}{\partial a}$ we see that the result is 0.

$$\frac{\partial S}{\partial a}_{|a=\frac{1}{2}} = \frac{2(-((1-\overline{q}_1)q_2) + q_1(1-\overline{q}_2))}{(1-\overline{q}_1)(1-\overline{q}_2)} = 0$$

To show that this is a maximum we must show that the second derivative is negative at $a = 1/2$. This is not difficult, though we omit the equations here for brevity. For the more general case, where the processors are not restricted to single step advances, the result that $a = 1/2$ ($\lambda_1 = \lambda_2$) for optimal performance generalizes to

$$\frac{\lambda_1}{\beta_1} = \frac{\lambda_2}{\beta_2} \quad \text{or} \quad b = \frac{a}{1-a} \quad (5)$$

meaning that the average "unfettered" rate of progress in virtual time for each processor should be the same. For a fixed value of a the best performance can be found when Equation 5 is true, and overall best performance is found at $a = 1/2$ with Equation 5 holding true. Speedup is not constant for a/b constant.

6.2 Adding a Cost for State Saving

One simple way of examining how state saving overhead affects the performance of the system is to modify the value of R_1, the rate of progress on a single processor. For example, if we examine the CD model with the single step restriction (as above) we arrive at the following value for R_1.

$$R_1 = \frac{c(\lambda_1 + \lambda_2)}{2}$$

The parameter c ($c \geq 1$) indicates how much faster events are executed without state saving. If $c = 2$ state saving doubles the amount of time it takes to process an event. For the CD model we find that the new formula for speedup is simply $1/c$ times the old value. Let us examine a very simple case in detail. If we look at the Symmetric, Balanced case, the updated formula for speedup is

$$S = \frac{4}{c(2+\sqrt{q})}.$$

It is easy to see that as $c \to \infty$ speedup will go to zero. We are most concerned with the boundary where $S = 1$ which is the transition from areas where TW on two processors helps to where it hurts. Setting $S = 1$ and solving for q we find the necessary condition for two processors running TW to be faster than only one.

$$q \leq \frac{4(2-c)^2}{c^2}$$

For $c \geq 2$ TW with two processors is always worse than running on one processor without TW. Conversely, for $c \leq 4/3$ TW wins out. The interesting range is where $4/3 \leq c \leq 2$. In this range, certain values of q will yield speedup, while others won't. Figure 8 shows the regions in the $c-q$ plane where TW on two processors is effective and where it is not. Thus, if we know the values of both c and q for our Symmetric, Balanced system we can immediately tell whether we can speed up the application by running it under Time Warp on two processors.

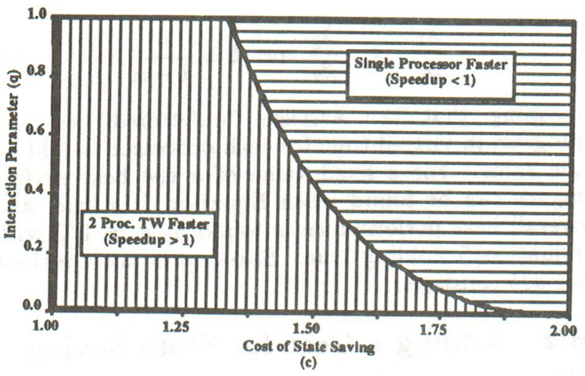

Figure 8: Cost for state saving and it's effect on performance.

7 Future Work and Conclusions

There are several avenues to follow for future work. One is to add message queueing to our model. Currently any message that arrives in the future is ignored. This is unrealistic since the messages in TW actually carry some work. Another addition would be to charge some cost for rollback. In the present model, rollbacks are free and therefore there is no penalty for speculative computing. We have exact solutions for models that address these concerns and they will appear in a future work [1]. We also would like to extend the model to accommodate more than two processors. Certainly, an exact Markov chain analysis will quickly become intractable. We are currently investigating extensions of our present model to many processors without using a complicated Markov chain approach.

In this paper we have presented a model for two processor Time Warp execution and provided the results of its exact solution. The model is general enough to subsume the work of Lavenberg, Muntz and Samadi [5] and to partially subsume the work of Mitra and Mitrani [9]. Further, we examined a simplified version of our model and showed for optimal performance that the processors should send as few messages as possible and that their independent rates of progress in virtual time should be the same. Finally, we addressed the cost of state saving and it's effect on performance. Large state saving costs or frequent message interactions indicate that TW is ineffective in gaining speedup. The detailed analysis of our model and logical generalizations to it will appear in future works [4][1].

References

[1] Robert E. Felderman and Leonard Kleinrock. Extensions to the time warp analysis. Submitted to ACM Transactions on Modelling and Computer Simulation, October 1990.

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

[3] Leonard Kleinrock. On distributed systems performance. In *Proceedings of the 7th ITC Specialist Seminar, Adelaide, Australia.* ITC, September 1989.

[4] Leonard Kleinrock and Robert E. Felderman. Two processor time warp analysis: A unifying approach. Submitted to International Journal of Computer Simulation, August 1990.

[5] Steven Lavenberg, Richard Muntz, and Behrokh Samadi. Performance analysis of a rollback method for distributed simulation. In *Performance '83.* North-Holland, 1983.

[6] Yi-Bing Lin and Edward D. Lazowska. Optimality considerations for "time warp" parallel simulation. Technical Report 89-07-05, Department of Computer Science and Engineering, University of Washington, July 1989.

[7] Vijay Krishna Madisetti. Self synchronizing concurrent computing systems. Technical Report UCB/ERL M89/122, Electronics Research Laboratory, College of Engineering University of California Berkeley, CA 94720, October 1989.

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

[9] Debasis Mitra and I. Mitrani. Analysis and optimum performance of two message-passing parallel processors synchronized by rollback. In *Performance '84.* North-Holland, 1984.