AEÜ, Vol. 47
(1993), No. 5/6

Two Processor Time Warp Analysis: Capturing the Effects of Message Queueing and Rollback/State Saving Costs

Robert E. Felderman and Leonard Kleinrock

Two Processor Time Warp Analysis: Capturing the Effects of Message Queueing and Rollback/State Saving Costs

We present two new models and their exact analysis for the problem of two processors running the Time Warp distributed simulation protocol. Our first model addresses the queueing of messages at each processor while the second model adds costs for rollback and state saving. Both models provide insight into the operation of freerunning systems synchronized by rollback.

Zweiprozessorsystem mit Time Warp Protokoll: Pufferung von Nachrichten und Kosten für Rollback und Einsparung von Zuständen

Das Problem eines Zweiprozessorsystems für die verteilte Simulation nach dem Time Warp Protokoll (Time Warp: Zeitverschiebung) wird aufgrund von zwei neuen Modellen exakt analysiert. Das erste Modell betrifft Wartevorgänge von Nachrichten bei jedem Prozessor, das zweite Modell erfasst die Kosten (Zeitaufwand) für die Rollback-Methode und bei der Einsparung von Zuständen. Beide Modelle bieten Einblicke in die Betriebsweise von freilaufenden Systemen, die mittels Rollback synchronisiert werden.

Keywords: Simulation, time warp, parallel and distributed processing, optimistic simulation, rollback, speedup, queueing, performance analysis, Markov chain.

1. Introduction

The systems that we are able to create become larger and more complex every day. It has become necessary to simulate the operation of proposed systems in order to better understand their behavior before huge investments are made in their implementation. As the size of these simulations increase they demand more computing time. Naturally then, one would like to utilize the recent advances in parallel computing technology to speed up the execution of simulations. Unfortunately, it is a non-trivial task to efficiently implement a parallel simulation system, though several techniques have been developed to do so. This paper presents analytical models of the performance of one distributed simulation algorithm, Time Warp (TW) [1].

1.1 Previous Work

Our research focuses on the analysis of the average case behavior of Time Warp when executing on exactly two processors. In our own previous work [2], [3], [4] we introduced a new model for the analysis of two-processor Time Warp. That model did not address message queueing nor did it associate a cost with rollback. Messages were only used for synchronization. This paper examines message queueing in our first model (something that has not been addressed in any model) and rollback and state saving costs in another model. These costs have not been adequately addressed in the previous work on two-processor models. Lavenberg et al. [5] and Mitra and Mitrani [6] have examined models similar to ours, although messages were only used for synchronization in both those models. Lavenberg et al. derived an approximation for speedup of two processors over one processor. Mitra and Mitrani, using a discrete time, continuous state model, solved (as we do) for the distribution of the separation in virtual time between the two processes. Mitra and Mitrani do introduce the concept of a cost for rollback and optimize the system based on it. Their technique was to calculate the average forward progress of the system per unit of real time (D), the average distance rolled back per unit of real time (R), create an objective function $J = D - cR$, then optimize the system with respect to J. Unfortunately this is somewhat artificial. The rollback cost should be an integral part of the model itself. When a process rolls back, it should be forced to pay a time penalty for rollback. A second criticism is that the objective function utilizes a rollback cost that is proportional to the distance rolled back. We believe that the cost is, at most, proportional to the log of the distance rolled back and is probably best approximated by a constant time delay regardless of the distance rolled back. Additionally, Mitra and Mitrani do go on to show how to allow for a different distribution for the size of the advance in virtual time depending on whether there has been a rollback or not. We discussed in more detail the relationship of the work of Lavenberg et al. and Mitra and Mitrani to our work in [2] and [3].

Lin and Lazowska [7] have examined Time Warp and conservative methods by appealing to critical path analysis. Also in [8] they create a model to reduce the state saving overhead in Time Warp. Though their work provides important insights, it generates different types

Received May 29, 1993.

Prof. Dr. R. E. Felderman, USC, Information Sciences Institute, 4676 Admiralty Way #1000, Marina del Rey CA 90292, USA.
Prof. Dr. L. Kleinrock, University of California, Los Angeles, Computer Science Department, 3732L Boelter Hall, Los Angeles CA 900024-1596, USA.

of results than ours. Madisetti [9], [10] provides bounds on the performance of a two processor system where the processors have different speeds of processing and move at constant rates, though again, messages are only used for synchronization. Madisetti extends his model to multiple processors, something we do not address in this work. Recently Nicol [11], [12] has attacked the problem of understanding the behavior of massively parallel simulations, both conservative and optimistic. Felderman and Kleinrock [13] have also looked at the multiple processor case.

1.2 Parallel Discrete Event Simulation

Parallel Discrete Event Simulation (PDES) is generally accomplished by partitioning the simulation into logical processes (LP) that simulate some physical process in the system. Each LP maintains an independent local clock indicating how far forward in simulation time it has progressed. Processes interact by sending and receiving timestamped messages. Each process operates autonomously by receiving messages, performing internal computation and sending messages. A process will terminate once its local clock, the time of receipt of the message currently being processed, has reached some user specified value. Certain simulations only allow the LP to perform operations in response to messages (the messages carry the work), while other simulations allow each LP to perform internal computations regardless of whether any messages have arrived. For example, an LP that is simulating a single server queue only performs an operation in response to the arrival of a message (customer). On the other hand, an LP that simulates a customer arrival process operates without receiving any messages at all. Nicol [11] discusses these two types of logical processes in more detail.

Each LP 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 [14]. 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 [15], [16]. The algorithm of interest for this paper is Time Warp [1] an asynchronous approach that uses a rollback mechanism invoked only when needed for synchronization. The essential problem to address when designing an algorithm for distributed simulation is to maintain causality between events. In the physical system, event A might have a direct causal effect on event B. When these two events are executed on two separate processors, it is non-trivial to *efficiently* make sure that event A actually occurs before B in real time. Time Warp maintains this causality by restoring a previous state and re-executing any operations it finds to have violated causality. The next section describes the algorithm in more detail.

1.3 Time Warp

The basic idea behind Time Warp is to allow each LP to advance forward as fast as it can without regard to the operation of the other LPs in the system. A TW process will choose the message with the minimum timestamp in its input queue; set its local clock to the time on that message; process the message; then find the next smallest message in the queue, etc. It is possible that a "straggler" message could arrive with a timestamp less than the local clock time of the LP. When this happens, the process is forced to "roll back" to a time before the timestamp of the arriving message This is able to 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 [1]. Any possible gain from the aggressive behavior of the Time Warp mechanism does not come without a cost. One of these costs is the overhead associated with the aforementioned state saving. There are two performance tradeoffs to keep in mind when choosing the frequency of state saving. If we save state very often, we pay a large time penalty in real time for all the data saving operations. If we choose to save state less often, we run the risk of having to roll back much further into the simulation time past than the time of the message causing the rollback, thus paying the time cost of re-executing correct events. Lin and Lazowska [8] address exactly this issue and find an optimum state saving interval based on certain assumptions about the arrival of messages and state saving costs etc. We don't examine this tradeoff in our work. Rather, we force each processor to save state after the execution of *every* event so as to keep the model tractable. The other overhead of state saving is the space required to save the history of the LP. Fortunately, we do not need to keep all state information back to the beginning of the run. A concept called Global Virtual Time (GVT) [1] allows the system to periodically throw away obsolete information. GVT is defined as the minimum of all the local LP clocks and the timestamps of all messages in transit. Since nothing in the system has a timestamp less than GVT, no process could ever be forced to roll back to a time prior to GVT. Obviously GVT is a very difficult measure to obtain, since we cannot take a "global" snapshot of this distributed system [17]. Algorithms have been developed to calculate a lower bound on GVT [18] that can be used as an estimate to free up memory space.

2. Message Queueing Model

We now introduce our model for two processor TW that allows messages that arrive in the virtual time future of a process to be queued. Additionally, the messages carry work for the receiving processor.

2.1 A Model for Two Time Warp Processes

Assume we have a job that is partitioned into two processes, each of which is executed on a separate proces-

1 Set local clock (v) to 0.
2 Execute local events for $v = 0$.
3 With probability $q(i)$, send message stamped with 1.
repeat
4 Advance local clock to $v = v + 1$.
5 Process message queue with timestamp $= v$ (if it exists).
6 Execute local events for time v.
7 With probability $q(i)$, send message stamped with $v + 1$.
until ($v \geq$ MAXTIME)
* If a message arrives at *any* time with a timestamp $t_m \leq v$):
– set local clock to t_m,
– goto line 5 and continue from there.

Fig. 1. Code executed by each processor.

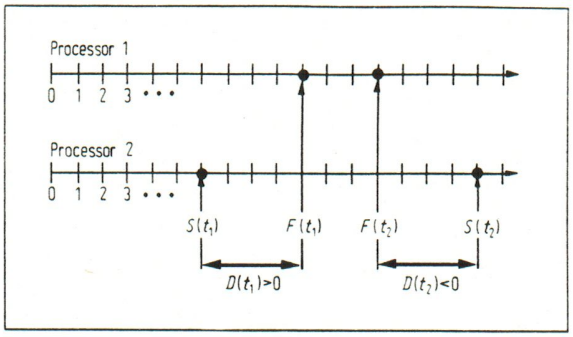

Fig. 2. States of two processors at times t_1 and t_2.

sor. A process at virtual time v operates by first executing any message in its input queue with timestamp v and then executing any locally scheduled work. Once completing its local work at virtual time v, a process advances its clock one unit and will then send a message to the other process with probability q_i. A process places its current virtual time on any message it sends. We will restrict the virtual times in our system to have integer values (i. e. $0, 1, 2, \ldots$). A process will schedule an event for itself at *every* point in virtual time. This means that processes will have their own work to do at every point in virtual time, and occasionally will have work sent to them from the other process. If a message arrives with a timestamp v equal to or smaller than the local clock time of the receiving processor, that processor is forced to rollback (discarding any work performed at a virtual time greater than or equal to v), execute the arriving message, then proceed forward again from virtual time v. We show the execution sequence for each LP in Fig. 1. Let v be the local clock time kept by the LP and let t_m be the timestamp on any arriving message.

More formally, we define two processes each executing on a separate processor. As these processes are executed, we consider that they visit the integers on the x-axis each beginning at $x = 0$ at time $t = 0$. To process a queued message, each processor takes an exponentially distributed amount of time with mean $1/\beta_i$, $i = 1, 2$. To process its locally generated work takes an exponentially distributed amount of time with mean $1/\lambda_i$, $i = 1, 2$. We assume that $\beta_i = f\lambda_i$ where $0 < f \leq \infty$ (f is referred to as the *work ratio*). After process i makes an advance along the axis, it will send a message to the other process with probability q_i, $i = 1, 2$. This message carries a timestamp that is the time of the sender after making the advance. Upon receiving a message from the other (sending) process, this (receiving) process will do the following:

1) If its position along the x-axis is behind the sending process, it queues the message.
2) If its position is equal to or 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 and begin to process that message. All work completed at virtual times greater than or equal to its current position is discarded and must be re-executed.

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

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

$D(t) = 0$ 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 Fig. 2 that shows the position of two processors at times t_1 and t_2). We will solve for

$$\lim_{t\to\infty} \mathrm{P}\,[D(t) = k], \quad -\infty < k < \infty$$

namely, the equilibrium probability for the Markov chain $D(t)$. First, let

M_1 = Event : Proc. 1 is processing a msg.,
$\overline{M}_1$ = Event : Proc. 1 is not processing a msg.,
M_2 = Event : Proc. 2 is processing a msg.,
$\overline{M}_2$ = Event : Proc. 2 is not processing a msg.

In order to find the solution, we split the chain into six regions.

$$P_k = \lim_{t\to\infty} \mathrm{P}\,[D(t) = k \text{ and } \overline{M}_2], \qquad k \geq 1,$$

$$Q_k = \lim_{t\to\infty} \mathrm{P}\,[D(t) = -k \text{ and } \overline{M}_1], \qquad k \geq 1,$$

$$S_k = \lim_{t\to\infty} \mathrm{P}\,[D(t) = k \text{ and } M_2], \qquad k \geq 0,$$

$$R_k = \lim_{t\to\infty} \mathrm{P}\,[D(t) = -k \text{ and } M_1], \qquad k \geq 0,$$

$$N_0 = \lim_{t\to\infty} \mathrm{P}\,[D(t) = 0 \text{ and } \overline{M}_1 \text{ and } \overline{M}_2],$$

$$B_0 = \lim_{t\to\infty} \mathrm{P}\,[D(t) = 0 \text{ and } M_1 \text{ and } M_2].$$

Using our solution, we will go on to solve for some interesting performance measures including the average rate of progress of the two-processor system.

AEU, Vol. 47
(1993), No. 5/6

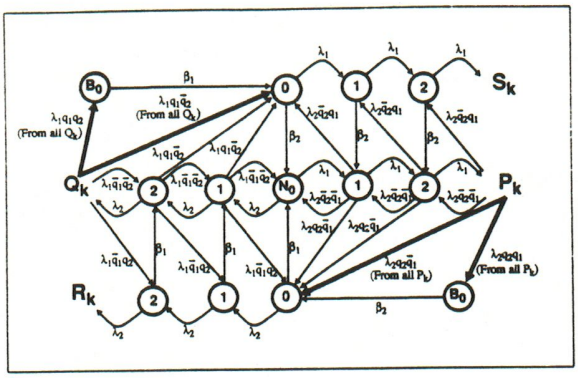

Fig. 3. State diagram for the Message Queueing Model.

There are some implicit assumptions in our description. 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 rollback to a much earlier time, namely, that of the last saved state. When process i causes the other process to rollback, process i immediately discards any messages it has queued in its future. This is as if the rolled back processor is able to transmit anti-messages instantaneously. This is not an unrealistic assumption in a shared-memory environment [19]. Another implicit assumption is that each process always schedules events for itself. We assume that communication between processors incurs no delay from transmission to reception. Finally, the interaction between the processes is probabilistic.

2.2 Analysis of the Message Queueing Model

In this section we provide the exact solution for the continuous time, discrete state model introduced in Section 2.1First, we provide some definitions.

λ_i = Rate at which proc. i executes local events,
$\beta_i = f\lambda_i$,
= Rate at which proc. i processes messages,
$a = \lambda_1/(\lambda_1+\lambda_2)$,
$\bar{a} = \lambda_2/(\lambda_1+\lambda_2) = 1-a$,
$A = a + \bar{a}f$,
$B = \bar{a} + af$,
q_i = P [ith proc. sends a msg. after advancing],
$\bar{q}_i = 1 - q_i$.

A state diagram for this system is shown in Fig. 3. Note that state B_0 is duplicated to reduce clutter in the figure.

The balance equations for our system are:

$$P_k = aP_{k-1} + \bar{a}\bar{q}_2\bar{q}_1P_{k+1} + \bar{a}fS_k,\ k \geq 2, \quad (1)$$

$$P_1 = aN_0 + \bar{a}\bar{q}_2\bar{q}_1P_2 + \bar{a}fS_1, \quad (2)$$

$$Q_k = \bar{a}Q_{k-1} + a\bar{q}_1\bar{q}_2Q_{k+1} + afR_k,\ k \geq 2, \quad (3)$$

$$Q_1 = \bar{a}N_0 + a\bar{q}_1\bar{q}_2Q_2 + afR_1, \quad (4)$$

$$N_0 = a\bar{q}_1\bar{q}_2Q_1 + \bar{a}\bar{q}_2\bar{q}_1P_1 + afR_0 + \bar{a}fS_0, \quad (5)$$

$$fB_0 = \bar{a}q_1q_2\sum_{i=1}^{\infty}P_i + aq_1q_2\sum_{i=1}^{\infty}Q_i, \quad (6)$$

$$AS_k = aS_{k-1} + \bar{a}\bar{q}_2q_1P_{k+1},\quad k > 0, \quad (7)$$

$$AS_0 = \bar{a}\bar{q}_2q_1P_1 + aq_1\bar{q}_2\sum_{i=1}^{\infty}Q_i + afB_0, \quad (8)$$

$$BR_k = \bar{a}R_{k-1} + a\bar{q}_1q_2Q_{k+1},\quad k > 0, \quad (9)$$

$$BR_0 = a\bar{q}_1q_2Q_1 + \bar{a}q_2\bar{q}_1\sum_{i=1}^{\infty}P_i + \bar{a}fB_0, \quad (10)$$

$$1 = \sum_{i=1}^{\infty}P_i + \sum_{i=1}^{\infty}Q_i + \sum_{i=0}^{\infty}S_i + \sum_{i=0}^{\infty}R_i + N_0 + B_0. \quad (11)$$

This system will have a steady-state solution if $\lambda_i > 0$, $q_i > 0$ and $f > 0$. These are fairly straightforward restrictions. The λ_i must be greater than 0 or the system makes no progress at all. The q_i must be greater than zero so that there is some probability that a processor will be rolled back once it gets ahead. Finally, the work ratio (f) must be greater than zero so that when a message is being processed the system will eventually complete the operation.

We define the following z-transforms (note the different ranges on k):

$$P(z) = \sum_{k=1}^{\infty}P_kz^k,\quad Q(z) = \sum_{k=1}^{\infty}Q_kz^k,$$

$$S(z) = \sum_{k=0}^{\infty}S_kz^k,\quad R(z) = \sum_{k=0}^{\infty}R_kz^k.$$

Using the above equations we can solve for $P(z)$, $Q(z)$, $S(z)$ and $R(z)$ by multiplying the appropriate equation by z^k and summing over the applicable range of k. We will only solve explicitly for $P(z)$ and $S(z)$ since $Q(z)$ and $R(z)$ are symmetric in $(a,\bar{a})$, (q_1,q_2) and (β_1,β_2) to $P(z)$ and $S(z)$. To simplify the expressions we define the following:

$$F_S = \bar{a}q_2P(1) + (1-\bar{a}q_2)\,Q(1),$$
$$F_R = aq_1Q(1) + (1-aq_1)\,P(1).$$

Solving for $P(z)$ in terms of $S(z)$ we get

$$P(z) = z \times \frac{-AS(z)\bar{a}f + F_Sa\bar{a}fq_1 + P_1\bar{a}\,(A-aq_1)\,\bar{q}_2 - AN_0az}{A\,(\bar{a}\bar{q}_1\bar{q}_2 - z + az^2)} \quad (12)$$

and for $S(z)$ in terms of $P(z)$

$$S(z) = \frac{q_1\,(P(z)\bar{a}\bar{q}_2 + F_Saz)}{z\,(A-az)}. \quad (13)$$

Solving them simultaneously we arrive at

$$P(z) = N(z)/D(z), \quad (14)$$

$$S(z) = N'(z)/D(z), \quad (15)$$

with

$$\begin{aligned} N(z) &= -z(-F_S a^2 \bar{a} f q_1 z + \\ &\quad + P_1 \bar{a}(A - a q_1)\bar{q}_2(A - az) \\ &\quad - A N_0 a z (A - az)), \qquad (16) \\ N'(z) &= -P_1 \bar{a}^2 q_1 (A - a q_1)\bar{q}_2{}^2 - A N_0 a \bar{a} q_1 \bar{q}_2 z + \\ &\quad + F_S a q_1 (\bar{a}(A - a q_1)\bar{q}_2 - Az + Aaz^2), \\ D(z) &= A(-\bar{a}(A - a q_1)\bar{q}_2 + (A + a\bar{a}q_1\bar{q}_2)z \\ &\quad - (1 + A)az^2 + a^2 z^3). \end{aligned}$$

Moreover, the denominator polynomial, $D(z)$, for $P(z)$ may be factored as follows:

$$D(z) = Aa^2 (z - r_1)(z - r_2)(z - r_3).$$

where r_1, r_2 and r_3 are the roots of the cubic polynomial in $D(z)$.

$$\begin{aligned} r_1 &= \frac{r_0 \cos((2\pi + \theta_r)/3)}{3a}, \\ r_2 &= \frac{r_0 \cos(\theta_r/3)}{3a}, \\ r_3 &= \frac{r_0 \cos((4\pi + \theta_r)/3)}{3a}, \end{aligned}$$

where

$$\begin{aligned} r_0 &= 1 + A - 2\sqrt{1 - A + A^2 - 3a\bar{a}q_1\bar{q}_2}, \\ \theta_r &= \arccos\Big\{[-(A-2)(1+A)(2A-1) + \\ &\quad + 9a\bar{a}\bar{q}_2(-3A + 3aq_1 + (1+A)\bar{q}_1)] \\ &\quad \Big/ \left[2(1 - A + A^2 - 3a\bar{a}q_1\bar{q}_2)^{3/2}\right]\Big\}. \end{aligned}$$

Symmetric roots (s_1, s_2, s_3) for the denominator of $Q(z)$ can be written down directly

$$\begin{aligned} s_1 &= \frac{s_0 \cos((2\pi + \theta_s)/3)}{3\bar{a}}, \\ s_2 &= \frac{s_0 \cos(\theta_s/3)}{3\bar{a}}, \\ s_3 &= \frac{s_0 \cos((4\pi + \theta_s)/3)}{3\bar{a}}, \end{aligned}$$

where

$$\begin{aligned} s_0 &= 1 + B - 2\sqrt{1 - B + B^2 - 3a\bar{a}q_1\bar{q}_2}, \\ \theta_s &= \arccos\Big\{[-(B-2)(1+B)(2B-1) + \\ &\quad + 9a\bar{a}\bar{q}_1(-3B + 3\bar{a}q_2 + (1+B)\bar{q}_2)] \\ &\quad \Big/ \left[2(1 - B + B^2 - 3a\bar{a}q_1\bar{q}_2)^{3/2}\right]\Big\}. \end{aligned}$$

See Appendix A for a derivation of the roots. It can be shown [20] that r_1, r_2 and r_3 are real and that $|r_2| \leq 1$ while $|r_1|$, $|r_3| > 1$. Since $P(z)$ is the z-transform of a probability distribution, it must be analytic in the range $|z| \leq 1$, and we know that $N(z)$ in eq. (16) must go to zero at $z = r_2$. We can use this fact to solve for P_1, yielding

$$P_1 = \frac{a r_2 (F_S a\bar{a} f q_1 + A N_0 (A - a r_2))}{\bar{a}(A - a q_1)\bar{q}_2 (A - a r_2)}.$$

Substituting this value back into $N(z)$ in eq. (16) we may write

$$\begin{aligned} &N(z) = \\ &= \frac{Aaz(z - r_2)(F_S a\bar{a} f q_1 + N_0(A - a r_2)(A - az))}{A - a r_2} \end{aligned}$$

and thus

$$P(z) = z\frac{F_S a\bar{a} f q_1 + N_0(A - a r_2)(A - az)}{a(A - a r_2)(r_1 - z)(r_3 - z)}. \qquad (17)$$

A similar procedure can be carried out on $S(z)$ in eq. (15) resulting in

$$S(z) = \frac{q_1 (N_0 \bar{a}\bar{q}_2 + F_S (1 - a r_2 - az))}{a(r_1 - z)(r_3 - z)}. \qquad (18)$$

Moreover, $Q(z)$ and $R(z)$ are symmetric in $(a, \bar{a})$, (q_1, q_2) and (β_1, β_2) to $P(z)$ and $S(z)$ so we can write them down directly.

$$Q(z) = z\frac{F_R a\bar{a} f q_2 + N_0 (B - \bar{a}s_2)(B - \bar{a}z)}{\bar{a}(B - \bar{a}s_2)(s_1 - z)(s_3 - z)}, \qquad (19)$$

$$R(z) = \frac{q_2 (N_0 a\bar{q}_1 + F_R (1 - \bar{a}s_2 - \bar{a}z))}{\bar{a}(s_1 - z)(s_3 - z)}. \qquad (20)$$

Recalling that F_S and F_R are functions of both $P(1)$ and $Q(1)$, we see that $P(z)$ and $Q(z)$ are functions of $P(1)$, $Q(1)$ and N_0. We solve for $P(1)$ and $Q(1)$ by solving the eqs. (17) and (19) of $P(z)$ and $Q(z)$ simultaneously with $z = 1$.

$$P(1) = C_P N_0 \qquad Q(1) = C_Q N_0$$

where

$$\begin{aligned} C_P &= \frac{C_{pn_0} + C_{pq}C_{qn_0} - C_{pn_0}C_{qq}}{1 - C_{pp} - C_{pq}C_{qp} - C_{qq} + C_{pp}C_{qq}}, \\ C_Q &= \frac{C_{qn_0} - C_{pp}C_{qn_0} + C_{pn_0}C_{qp}}{1 - C_{pp} - C_{pq}C_{qp} - C_{qq} + C_{pp}C_{qq}}, \end{aligned}$$

and

$$\begin{aligned} C_{pn_0} &= \frac{\bar{a}f}{a(r_1 - 1)(r_3 - 1)}, \\ C_{pp} &= \frac{\bar{a}^2 f q_1 q_2}{(r_1 - 1)(A - a r_2)(r_3 - 1)}, \\ C_{pq} &= \frac{\bar{a}f q_1 (1 - \bar{a}q_2)}{(r_1 - 1)(A - a r_2)(r_3 - 1)}, \\ C_{qn_0} &= \frac{af}{\bar{a}(s_1 - 1)(s_3 - 1)}, \\ C_{qp} &= \frac{af(1 - aq_1)q_2}{(s_1 - 1)(B - \bar{a}s_2)(s_3 - 1)}, \\ C_{qq} &= \frac{a^2 f q_1 q_2}{(s_1 - 1)(B - \bar{a}s_2)(s_3 - 1)}. \end{aligned}$$

Noting that $P(1)+Q(1)+S(1)+R(1)+N_0+B_0 = 1$ we solve for N_0.

$$N_0 = \Bigg[1 + \frac{(C_Q a + C_P \bar{a})\, q_1 q_2}{f} + \\ + \frac{C_{F_S} a\bar{a} f q_1 + \bar{a} f (A - a r_2)}{a (r_1 - 1)(A - a r_2)(r_3 - 1)} + \\ + \frac{q_1 (\bar{a} q_2 + C_{F_S} (\bar{a} - a r_2))}{a (r_1 - 1)(r_3 - 1)} + \\ + \frac{C_{F_R} a\bar{a} f q_2 + a f (B - \bar{a} s_2)}{\bar{a} (s_1 - 1)(B - \bar{a} s_2)(s_3 - 1)} + \\ + \frac{q_2 (a \bar{q}_1 + C_{F_R} (a - \bar{a} s_2))}{\bar{a} (s_1 - 1)(s_3 - 1)}\Bigg]^{-1}. \quad (21)$$

Finally, by inverting the transforms we find the probability of being in any state (other than N_0).

$$P_k = K_1 \left(\frac{1}{r_1}\right)^k + K_2 \left(\frac{1}{r_3}\right)^k, \quad k \geq 1, \quad (22)$$

$$Q_k = K_3 \left(\frac{1}{s_1}\right)^k + K_4 \left(\frac{1}{s_3}\right)^k, \quad k \geq 1, \quad (23)$$

$$S_k = K_5 \left(\frac{1}{r_1}\right)^k + K_6 \left(\frac{1}{r_3}\right)^k, \quad k \geq 0, \quad (24)$$

$$R_k = K_7 \left(\frac{1}{s_1}\right)^k + K_8 \left(\frac{1}{s_3}\right)^k, \quad k \geq 0, \quad (25)$$

$$B_0 = \frac{N_0 (C_Q a + C_P \bar{a})\, q_1 q_2}{f}, \quad (26)$$

where

$$K_1 = \frac{N_0 (C_{F_S} a\bar{a} f q_1 + (A - a r_1)(A - a r_2))}{a (A - a r_2)(r_3 - r_1)},$$

$$K_2 = \frac{N_0 (C_{F_S} a\bar{a} f q_1 + (A - a r_2)(A - a r_3))}{a (A - a r_2)(r_1 - r_3)},$$

$$K_3 = \frac{N_0 (C_{F_R} a\bar{a} f q_2 + (B - \bar{a} s_1)(B - \bar{a} s_2))}{\bar{a} (B - \bar{a} s_2)(s_3 - s_1)},$$

$$K_4 = \frac{N_0 (C_{F_R} a\bar{a} f q_2 + (B - \bar{a} s_2)(B - \bar{a} s_3))}{\bar{a} (B - \bar{a} s_2)(s_1 - s_3)},$$

$$K_5 = \frac{N_0 q_1 (\bar{a} q_2 + C_{F_S} (1 - a r_1 - a r_2))}{a r_1 (r_3 - r_1)},$$

$$K_6 = \frac{N_0 q_1 (\bar{a} q_2 + C_{F_S} (1 - a r_2 - a r_3))}{a (r_1 - r_3) r_3},$$

$$K_7 = \frac{N_0 q_2 (a \bar{q}_1 + C_{F_R} (1 - \bar{a} s_1 - \bar{a} s_2))}{\bar{a} s_1 (s_3 - s_1)},$$

$$K_8 = \frac{N_0 q_2 (a \bar{q}_1 + C_{F_R} (1 - \bar{a} s_2 - \bar{a} s_3))}{\bar{a} (s_1 - s_3) s_3},$$

$$C_{F_S} = C_P \bar{a} q_2 + C_Q (1 - \bar{a} q_2),$$

$$C_{F_R} = C_Q a q_1 + C_P (1 - a q_1).$$

This completes our calculation of the explicit expressions for the equilibrium state probabilities of our chain.

2.3 Performance Measures

Using the solution to the Markov chain that was calculated above, we may solve for any performance measure of interest. In the following sections we examine a few important ones.

2.3.1 State Buffer Use

When a processor completes its local processing it advances its clock by one time unit. Therefore, if a processor is ahead by k units of virtual time (k units of distance on the axis), then it will need to have saved k states. The expected number of buffers ($\overline{B}_i$) needed to save state at each processor can be found from

$$\overline{B}_1 = \sum_{i=1}^{\infty} i (P_i + S_i) = \\ = \frac{(K_1 + K_5) r_1}{(r_1 - 1)^2} + \frac{(K_2 + K_6) r_3}{(r_3 - 1)^2}, \quad (27)$$

$$\overline{B}_2 = \sum_{i=1}^{\infty} i (Q_i + R_i) = \\ = \frac{(K_3 + K_7) s_1}{(s_1 - 1)^2} + \frac{(K_4 + K_8) s_3}{(s_3 - 1)^2}. \quad (28)$$

More interestingly, we find that the probability that a fixed size buffer of size $b \geq 1$ overflows at processor i ($\Theta_{i,b}$) is

$$\Theta_{1,b} = \sum_{i=b+1}^{\infty} (P_i + S_i) = \\ = \sum_{i=0}^{\infty} (P_i + S_i) - \sum_{i=0}^{b} (P_i + S_i) = \\ = \frac{(K_1 + K_5)}{r_1{}^b (r_1 - 1)} + \frac{(K_2 + K_6)}{r_3{}^b (r_3 - 1)}, \quad (29)$$

$$\Theta_{2,b} = \sum_{i=b+1}^{\infty} (Q_i + R_i) = \\ = \sum_{i=0}^{\infty} (Q_i + R_i) - \sum_{i=0}^{b} (Q_i + R_i) = \\ = \frac{(K_3 + K_7)}{s_1{}^b (s_1 - 1)} + \frac{(K_4 + K_8)}{s_3{}^b (s_3 - 1)}. \quad (30)$$

2.3.2 Message Queue Distribution

Messages that arrive in the virtual time future are queued until the processor completes all work with a virtual time less than the arriving message. We define the size of the message queue as the number of messages queued in the virtual time future of the processor, plus any message that is currently being processed. The distribution of message queue length at each processor is found by summing over the appropriate ranges of the state probabilities.

$m_{1,k} = \mathrm{P}[k \text{ messages queued at Processor } 1]$,

$$m_{1,k} = \sum_{i=k}^{\infty} Q_i \binom{i}{k} {q_2}^k {\bar{q}_2}^{i-k} +$$
$$+ \sum_{i=k-1}^{\infty} R_i \binom{i}{k-1} {q_2}^{k-1} {\bar{q}_2}^{i-k+1}, \quad k \geq 2$$
$$= \frac{K_3 {q_2}^k s_1}{(s_1 - \bar{q}_2)^{k+1}} + \frac{K_7 {q_2}^{k-1} s_1}{(s_1 - \bar{q}_2)^k} +$$
$$+ \frac{K_4 {q_2}^k s_3}{(s_3 - \bar{q}_2)^{k+1}} + \frac{K_8 {q_2}^{k-1} s_3}{(s_3 - \bar{q}_2)^k},$$
$$m_{1,1} = \sum_{i=1}^{\infty} Q_i i q_2 {\bar{q}_2}^{i-1} + \sum_{i=0}^{\infty} R_i {\bar{q}_2}^i + B_0 =$$
$$= \frac{K_3 q_2 s_1}{(s_1 - \bar{q}_2)^2} + \frac{K_7 s_1}{s_1 - \bar{q}_2} +$$
$$+ \frac{K_4 q_2 s_3}{(s_3 - \bar{q}_2)^2} + \frac{K_8 s_3}{s_3 - \bar{q}_2} + B_0,$$
$$m_{1,0} = P(1) + S(1) + N_0 + \sum_{i=1}^{\infty} Q_i {\bar{q}_2}^i =$$
$$= P(1) + S(1) + N_0 + \frac{K_3 \bar{q}_2}{s_1 - \bar{q}_2} + \frac{K_4 \bar{q}_2}{s_3 - \bar{q}_2},$$
$$m_{2,k} = \mathrm{P}[k \text{ messages queued at Processor 2}],$$
$$m_{2,k} = \sum_{i=k}^{\infty} P_i \binom{i}{k} {q_1}^k {\bar{q}_1}^{i-k} +$$
$$+ \sum_{i=k-1}^{\infty} S_i \binom{i}{k-1} {q_1}^{k-1} {\bar{q}_1}^{i-k+1}, \quad k \geq 2$$
$$= \frac{K_1 {q_1}^k r_1}{(r_1 - \bar{q}_1)^{k+1}} + \frac{K_5 {q_1}^{k-1} r_1}{(r_1 - \bar{q}_1)^k} +$$
$$+ \frac{K_2 {q_1}^k r_3}{(r_3 - \bar{q}_1)^{k+1}} + \frac{K_6 {q_1}^{k-1} r_3}{(r_3 - \bar{q}_1)^k},$$
$$m_{2,1} = \sum_{i=1}^{\infty} P_i i q_1 {\bar{q}_1}^{i-1} + \sum_{i=0}^{\infty} S_i {\bar{q}_1}^i + B_0 =$$
$$= \frac{K_1 q_1 r_1}{(r_1 - \bar{q}_1)^2} + \frac{K_5 r_1}{r_1 - \bar{q}_1} +$$
$$+ \frac{K_2 q_1 r_3}{(r_3 - \bar{q}_1)^2} + \frac{K_6 r_3}{r_3 - \bar{q}_1} + B_0,$$
$$m_{2,0} = Q(1) + R(1) + N_0 + \sum_{i=1}^{\infty} P_i {\bar{q}_1}^i =$$
$$= Q(1) + R(1) + N_0 + \frac{K_1 \bar{q}_1}{r_1 - \bar{q}_1} + \frac{K_2 \bar{q}_1}{r_3 - \bar{q}_1}.$$

The mean number of message buffers needed at each processor is

$$\overline{m}_1 = \sum_{i=0}^{\infty} i m_{1,i} =$$
$$= \frac{s_1 (K_3 q_2 + K_7 (s_1 - \bar{q}_2))}{(s_1 - 1)^2} +$$
$$+ \frac{s_3 (K_4 q_2 + K_8 (s_3 - \bar{q}_2))}{(s_3 - 1)^2} + B_0,$$
$$\overline{m}_2 = \sum_{i=0}^{\infty} i m_{2,i} =$$
$$= \frac{r_1 (K_1 q_1 + K_5 (r_1 - \bar{q}_1))}{(r_1 - 1)^2} +$$
$$+ \frac{r_3 (K_2 q_1 + K_6 (r_3 - \bar{q}_1))}{(r_3 - 1)^2} + B_0.$$

2.3.3 Normalized Rate of Progress

From the complete solution of the Markov chain we calculate the average rate of progress of the two processor system. We define γ_2 as the average rate of progress in virtual time of the two-processor system. This value is simply the average "unfettered" rate of progress of the two processors minus the average rollback rate.

$$\gamma_2 = (\lambda_1 + \lambda_2) \left(\sum_{k=1}^{\infty} Q_k + N_0 + \sum_{k=1}^{\infty} P_k \right) +$$
$$+ \lambda_1 \sum_{k=0}^{\infty} S_k + \lambda_2 \sum_{k=0}^{\infty} R_k$$
$$- \lambda_2 q_2 \sum_{k=1}^{\infty} P_k(k-1) - \lambda_1 q_1 \sum_{k=1}^{\infty} Q_k(k-1) =$$
$$= (\lambda_1 + \lambda_2) \left(\frac{K_1}{r_1 - 1} + \frac{K_2}{r_3 - 1} + \right.$$
$$\left. + N_0 + \frac{K_3}{s_1 - 1} + \frac{K_4}{s_3 - 1} \right) +$$
$$+ \lambda_1 \left(\frac{K_5 r_1}{r_1 - 1} + \frac{K_6 r_3}{r_3 - 1} \right) +$$
$$+ \lambda_2 \left(\frac{K_7 s_1}{s_1 - 1} + \frac{K_8 s_3}{s_3 - 1} \right)$$
$$- \lambda_1 q_1 \left(\frac{K_3}{(s_1 - 1)^2} + \frac{K_4}{(s_3 - 1)^2} \right)$$
$$- \lambda_2 q_2 \left(\frac{K_1}{(r_1 - 1)^2} + \frac{K_2}{(r_3 - 1)^2} \right). \qquad (31)$$

We can calculate a "normalized" rate of progress (Γ) by dividing the above equation by $(\lambda_1 + \lambda_2)$. We arrive at

$$\Gamma = \left(\frac{K_1}{r_1 - 1} + \frac{K_2}{r_3 - 1} + N_0 + \frac{K_3}{s_1 - 1} + \frac{K_4}{s_3 - 1} \right) +$$
$$+ a \left(\frac{K_5 r_1}{r_1 - 1} + \frac{K_6 r_3}{r_3 - 1} \right) +$$
$$+ \bar{a} \left(\frac{K_7 s_1}{s_1 - 1} + \frac{K_8 s_3}{s_3 - 1} \right) -$$
$$- \bar{a} q_2 \left(\frac{K_1}{(r_1 - 1)^2} + \frac{K_2}{(r_3 - 1)^2} \right) -$$
$$- a q_1 \left(\frac{K_3}{(s_1 - 1)^2} + \frac{K_4}{(s_3 - 1)^2} \right). \qquad (32)$$

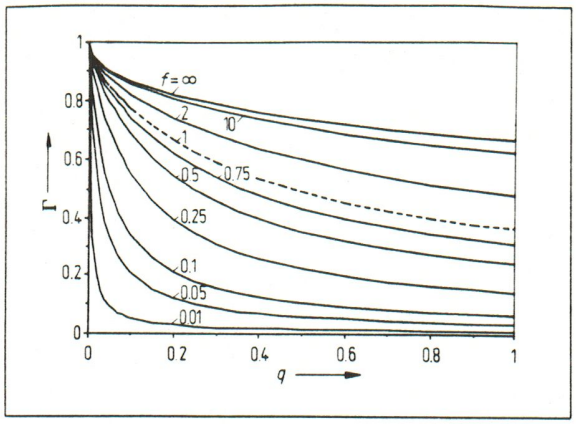

Fig. 4. Γ versus work ratio f and interaction parameter q for the Symmetric, Balanced Case.

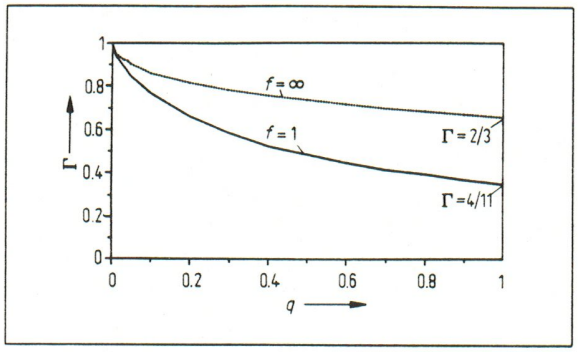

Fig. 5. Γ versus q for the Symmetric, Balanced Case.

It is interesting to note that as the work ratio increases ($f \to \infty$) the message processing time approaches zero, therefore messages are only used for synchronization and our system reduces to our original model [4]. In Fig. 4 we show the value for Γ when $a = 1/2$ and $q_1 = q_2 = q$ which we refer to as the Symmetric, Balanced case. The figure shows Γ versus q for various values of the work ratio (f). We see that for the best performance we want the interaction to be small and the work ratio to be large ($q \to 0$ and $f \to \infty$). This is the case where there is little interaction between the processors and it takes zero time to process a message from the other processor. By setting $f = 1$ we can examine Γ versus q only. This plot is shown in Fig. 5 compared to the average rate of progress for the same system where messages are only used for synchronization ($f = \infty$). We see that the system where messages carry work performs more poorly than where they are only used for synchronization. This is no surprize since there is more work to do. It is interesting to note that this system is not twice as bad as the synchronization-only system even at $q = 1$. In fact, at $q = 1$ we can verify the Γ result for $f = 1$ by realizing that each processor will always have a message to process. Therefore, the rate of progress at each step is governed by the maximum time it takes for the two processors to each finish a message and local work. This is simply the expected

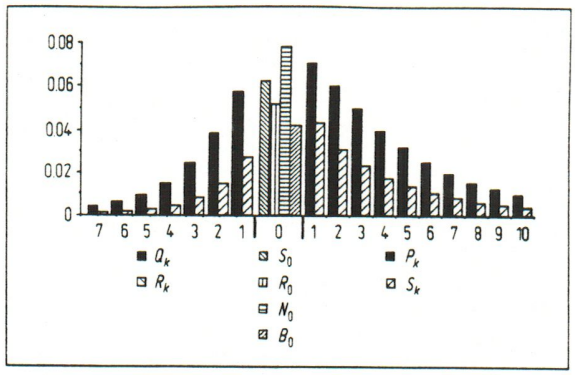

Fig. 6. State probabilities.

value of the maximum of two 2-stage Erlangs at rate λ which is equal to $\frac{11}{4\lambda}$. Taking the reciprocal and dividing by λ to find the rate, we get $\Gamma = 4/11$ which is the value plotted in Fig. 5.

2.4 A Specific Example

To better understand the above results we explicitly calculate values of our performance measures for a specific instance of the parameters of our system. The values chosen are given below.

$$\lambda_1 = 11, \quad \lambda_2 = 9, \quad a = 11/20, \quad \bar{a} = 9/20,$$
$$f = 1, \quad q_1 = 1/2, \quad q_2 = 1/3.$$

Note that processor one will move slightly faster than processor two while the cost of processing a message is the same as processing a locally generated event. Finally, processor one will send a message with probability $1/2$ while processor 2 will send a message with probability $1/3$ after advancing.

2.4.1 State Probabilities and State Buffer Use

The resulting equations for the probability of being in any state are

$$N_0 \approx 0.0781,$$
$$B_0 \approx 0.0423,$$
$$P_k \approx \frac{0.114}{1.281^k} - \frac{0.0359}{2.086^k}, \quad k \geq 1,$$
$$Q_k \approx \frac{0.1385}{1.702^k} - \frac{0.0605}{2.468^k}, \quad k \geq 1,$$
$$S_k \approx \frac{0.0452}{1.281^k} + \frac{0.0175}{2.086^k}, \quad k \geq 0,$$
$$R_k \approx \frac{0.0319}{1.702^k} + \frac{0.0203}{2.468^k}, \quad k \geq 0.$$

These probabilities are plotted in Fig. 6. As you would expect, $P_k > Q_k$ and $S_k > R_k$ since processor one is moving at a faster rate than processor two. The expected number of buffers needed to save state at each processor

$(\overline{B}_i)$ is given by

$$\overline{B}_1 = \sum_{i=1}^{\infty} i(P_i + S_i) \approx 2.5489,$$

$$\overline{B}_2 = \sum_{i=1}^{\infty} i(Q_i + R_i) \approx 0.5429.$$

From the values for $\Theta_{1,b}$ and $\Theta_{2,b}$

$$\Theta_{1,b} = \frac{0.5663}{1.281^b} - \frac{0.0169}{2.086^b},$$

$$\Theta_{2,b} = \frac{0.2428}{1.702^b} - \frac{0.0273}{2.468^b}$$

we find that with probability greater than 0.99 processor one (p_1) will not need more than seventeen buffers. A similar value can be found for processor two (p_2).

$$\mathrm{P}[p_1 \text{ needs } > 17 \text{ state buffers }] \approx 0.00841 < 0.01,$$
$$\mathrm{P}[p_2 \text{ needs } > 6 \text{ state buffers }] \approx 0.00988 < 0.01.$$

2.4.2 Message Queue Distribution and Buffer Use

The distribution of messages at each processor is given below.

$$m_{1,0} \approx 0.7569,$$
$$m_{1,1} \approx 0.1805,$$
$$m_{1,k} \approx \sum_{i=k-1}^{\infty} \left(\frac{1}{3}\right)^{k-1} \left(\frac{2}{3}\right)^{i-k+1} \times \left(\frac{0.0319}{1.702^i} + \frac{0.0203}{2.468^i}\right) \binom{i}{k-1} + \sum_{i=k}^{\infty} \left(\frac{1}{3}\right)^{k} \left(\frac{2}{3}\right)^{i-k} \times \left(\frac{0.1385}{1.702^i} - \frac{0.0605}{2.468^i}\right) \binom{i}{k}, \quad k \geq 2,$$
$$m_{2,0} \approx 0.4074,$$
$$m_{2,1} \approx 0.2441,$$
$$m_{2,k} \approx \sum_{i=k-1}^{\infty} \left(\frac{1}{2}\right)^{i} \left(\frac{0.04517}{1.281^i} + \frac{0.0175}{2.086^i}\right) \binom{i}{k-1} + \sum_{i=k}^{\infty} \left(\frac{1}{2}\right)^{i} \left(\frac{0.114}{1.281^i} - \frac{0.0359}{2.086^i}\right) \binom{i}{k}, \quad k \geq 2.$$

The values of these functions are plotted in Fig. 7. The mean number of message buffers needed at each processor is

$$\overline{m}_1 \approx 0.3346, \qquad \overline{m}_2 \approx 1.5562.$$

As with the state buffers we can find the number of message buffers needed to store messages such that the buffers will overflow with probability less than 0.01.

$$\mathrm{P}[p_1 \text{ needs } > 3 \text{ msg. buffers}] \approx 0.0063 < 0.01,$$
$$\mathrm{P}[p_2 \text{ needs } > 9 \text{ msg. buffers}] \approx 0.0097 < 0.01.$$

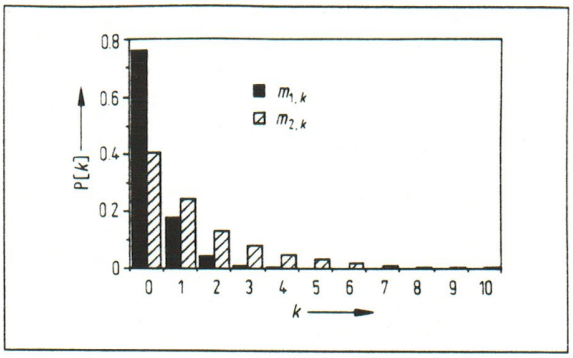

Fig. 7. Distribution of the number of messages queued at each processor.

Finally, the value for the normalized rate of progress is $\Gamma \approx 0.5071$.

2.5 Summary

We introduced and solved exactly a new model for two-processor Time Warp operation. The importance of our new model is that it explicitly accounts for the work that must be performed by each processor in response to the receipt of a message. Messages that arrive in the past cause rollbacks, while messages that arrive in the future are queued until the LP moves forward in simulation time. In all cases the messages create work for the LP.

With the complete Markov chain solution we calculated the normalized rate of progress of the two processors, and the distribution of the number of messages queued at each processor. Further, we found the expected number of buffers needed to save state and/or messages at each processor. Since we have the exact solution to the complete Markov chain we can calculate nearly any parameter that might be of interest.

3. A Model for Rollback and State Saving Costs

If the costs for rollback and/or state saving are high, TW may perform poorly. The following sections examine the two-processor system when we account for rollback and state saving costs.

3.1 The Model

We use a model similar to the one introduced in Section 2.1, a continuous time, discrete state model where each processor makes only single step state advances whenever it advances. Right after a processor is forced to rollback, it pays a cost for restoring state by making the expected rate of forward progress smaller than normal for one event. When processing the "rollback event" each processor moves at a rate $\beta_i = f\lambda_i$ where $0 < f \leq 1$. Once this event is completed, the processor moves again at its normal rate of λ_i. Note that when $f = 1$ there is no additional cost for rollback and this

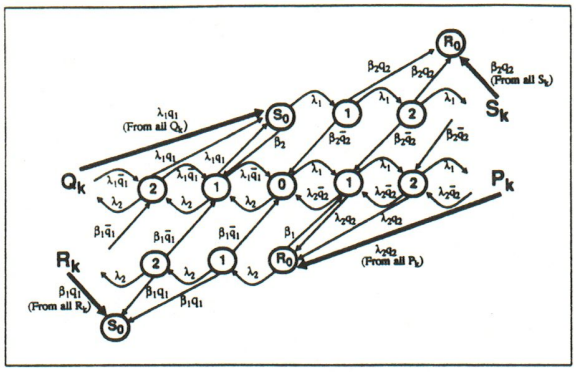

Fig. 8. State diagram for the Rollback Cost Model.

model reduces to the one in [4]. We add a cost for state saving in Section 3.3.2

First, let

$$
\begin{aligned}
RB_1 &= \text{Event : Proc. 1 is in a rollback state,}\\
\overline{RB}_1 &= \text{Event : Proc. 1 is not in a rollback state,}\\
RB_2 &= \text{Event : Proc. 2 is in rollback state,}\\
\overline{RB}_2 &= \text{Event : Proc. 2 is not in a rollback state.}
\end{aligned}
$$

To solve the system we separate the Markov chain into five different regions.

$$
\begin{aligned}
P_k &= \lim_{t\to\infty} \mathrm{P}[D(t) = k \text{ and } \overline{RB}_2], && k \geq 1,\\
Q_k &= \lim_{t\to\infty} \mathrm{P}[D(t) = -k \text{ and } \overline{RB}_1], && k \geq 1,\\
S_k &= \lim_{t\to\infty} \mathrm{P}[D(t) = k \text{ and } RB_2], && k \geq 0,\\
R_k &= \lim_{t\to\infty} \mathrm{P}[D(t) = -k \text{ and } RB_1], && k \geq 0,\\
P_0 &= \lim_{t\to\infty} \mathrm{P}[D(t) = 0 \text{ and } \overline{RB}_1 \text{ and } \overline{RB}_2].
\end{aligned}
$$

3.2 Analysis of the Cost Model

In this section we find the exact solution for the model that addresses rollback and state saving costs. The parameters of this system are

$$
\begin{aligned}
\lambda_i &= \text{Rate at which Proc. } i \text{ processes events,}\\
\beta_i &= f\lambda_i = \text{Rate at which Proc. } i\\
&\qquad \text{processes after a rollback,}\\
a &= \lambda_1/(\lambda_1+\lambda_2),\\
\bar{a} &= \lambda_2/(\lambda_1+\lambda_2) = 1-a,\\
A &= a + \bar{a}f,\\
B &= \bar{a} + af,\\
q_i &= \mathrm{P}\,[i\text{th proc. sends a msg. after advancing}],\\
\bar{q}_i &= 1 - q_i.
\end{aligned}
$$

A state diagram is shown in Fig. 8. Note that the S_0 and R_0 states were duplicated to keep the figure from being too cluttered with transition arcs. As with the previous model, this system will have an equilibrium solution when $\lambda_i > 0$, $q_i > 0$ and $f > 0$.

The balance equations for this new system are

$$
\begin{aligned}
(\lambda_1+\beta_2)S_k &= \lambda_1 S_{k-1}, \quad k \geq 1, && (33)\\
(\lambda_1+\beta_2)S_0 &= \lambda_1 q_1 \sum_{i=1}^{\infty} Q_i + \beta_1 q_1 \sum_{i=1}^{\infty} R_i, && (34)\\
(\lambda_2+\beta_1)R_k &= \lambda_2 R_{k-1}, \quad k \geq 1, && (35)\\
(\lambda_2+\beta_1)R_0 &= \lambda_2 q_2 \sum_{i=1}^{\infty} P_i + \beta_2 q_2 \sum_{i=1}^{\infty} S_i, && (36)\\
(\lambda_1+\lambda_2)P_k &= \lambda_1 P_{k-1} + \lambda_2\bar{q}_2 P_{k+1} +\\
&\quad + \beta_2\bar{q}_2 S_{k+1}, \quad k \geq 2, && (37)\\
(\lambda_1+\lambda_2)P_1 &= \lambda_1 P_0 + \lambda_2\bar{q}_2 P_2 +\\
&\quad + \beta_2\bar{q}_2 S_2 + \beta_1 R_0, && (38)\\
(\lambda_1+\lambda_2)P_0 &= \lambda_1\bar{q}_1 Q_1 + \lambda_2\bar{q}_2 P_1 +\\
&\quad + \beta_1\bar{q}_1 R_1 + \beta_2\bar{q}_2 S_1, && (39)\\
(\lambda_2+\lambda_1)Q_k &= \lambda_2 Q_{k-1} + \lambda_1\bar{q}_1 Q_{k+1} +\\
&\quad + \beta_1\bar{q}_1 R_{k+1}, \quad k \geq 2, && (40)\\
(\lambda_2+\lambda_1)Q_1 &= \lambda_2 P_0 + \beta_2 S_0 +\\
&\quad + \lambda_1\bar{q}_1 Q_2 + \beta_1\bar{q}_1 R_2, && (41)
\end{aligned}
$$

$$
1 = P_0 + \sum_{i=1}^{\infty} P_i + \sum_{i=1}^{\infty} Q_i + \sum_{i=1}^{\infty} S_i + \sum_{i=1}^{\infty} R_i.
$$

As before we define the following z-transforms (note, $S(z)$ and $R(z)$ are defined from $k = 1$ *not* $k = 0$ as in the previous model):

$$
P(z) = \sum_{k=1}^{\infty} P_k z^k, \quad Q(z) = \sum_{k=1}^{\infty} Q_k z^k,
$$

$$
S(z) = \sum_{k=1}^{\infty} S_k z^k, \quad R(z) = \sum_{k=1}^{\infty} R_k z^k.
$$

We proceed to find $P(z)$, $Q(z)$, $S(z)$ and $R(z)$ by multiplying the appropriate equation above by z^k and summing over the valid range of k. This leads to

$$
\begin{aligned}
P(z) &= [-Aa(P_0 + R_0 f)z^2\\
&\quad - \bar{a}\bar{q}_2(AS(z)f - AP_1 z - S_0 afz)]\\
&\quad /[A(\bar{a}\bar{q}_2 - z + az^2)], && (42)\\
Q(z) &= [-B\bar{a}(P_0 + S_0 f)z^2\\
&\quad - a\bar{q}_1(BR(z)f - BQ_1 z - R_0\bar{a}fz)]\\
&\quad /[B(a\bar{q}_1 - z + \bar{a}z^2)], && (43)
\end{aligned}
$$

$$
S(z) = \frac{S_0 az}{A - az}, \qquad R(z) = \frac{R_0\bar{a}z}{B - \bar{a}z}.
$$

Substituting the value for $S(z)$ in eq. (42) for $P(z)$ we arrive at the following equation that defines $P(z)$.

$$
\begin{aligned}
P(z) &= \left[z\left(-S_0 a^2\bar{a}f\bar{q}_2 z + AP_1\bar{a}\bar{q}_2(A - az)\right.\right.\\
&\quad \left.\left. -Aa(P_0 + R_0 f)z(A - az)\right)\right]\\
&\quad /\left[A(A - az)\left(\bar{a}\bar{q}_2 - z + az^2\right)\right]. && (44)
\end{aligned}
$$

The denominator of $P(z)$ in eq. (44) can be factored into $A(A - az)(z - r_1)(z - r_2)$ and the denominator

of $Q(z)$ into $B(B-\bar{a}z)(z-s_1)(z-s_2)$ where

$$r_{1,2} = \frac{1 \pm \sqrt{1-4a\bar{a}\bar{q}_2}}{2a},$$
$$s_{1,2} = \frac{1 \pm \sqrt{1-4a\bar{a}\bar{q}_1}}{2\bar{a}},$$

It is simple to show that r_1 and r_2 are real and that $r_1 \geq 1$ while $0 \leq r_2 \leq 1$ [3]. Since $P(z)$ must be analytic in the region $|z| \leq 1$ the numerator of $P(z)$ must go to zero when $z = r_2$. Using this information we solve for P_1.

$$P_1 = \frac{ar_2\left(S_0 a\bar{a}f\bar{q}_2 + A\left(P_0 + R_0 f\right)\left(A - ar_2\right)\right)}{A\bar{a}\bar{q}_2\left(A - ar_2\right)}.$$

We substitute this value back into the equation for $P(z)$ and arrive at

$$P(z) = \tag{45}$$
$$= \frac{z\left(S_0 a\bar{a}f\bar{q}_2 + \left(P_0 + R_0 f\right)\left(A - ar_2\right)\left(A - az\right)\right)}{\left(A - ar_2\right)\left(r_1 - z\right)\left(A - az\right)}.$$

Similarly for $Q(z)$ we find

$$Q(z) = \tag{46}$$
$$= \frac{z\left(R_0 a\bar{a}f\bar{q}_1 + \left(P_0 + S_0 f\right)\left(B - \bar{a}s_2\right)\left(B - \bar{a}z\right)\right)}{\left(B - \bar{a}s_2\right)\left(s_1 - z\right)\left(B - \bar{a}z\right)}.$$

Our task now is to find the values for the unknown constants P_0, S_0 and R_0. We can solve eqs. (34) and (36) for S_0 and R_0 simultaneously to find

$$S_0 = \frac{q_1\left(\bar{a}^2 q_2 P(1) + BaQ(1)\right)}{AB - a\bar{a}q_1 q_2},$$
$$R_0 = \frac{q_2\left(A\bar{a}P(1) + a^2 q_1 Q(1)\right)}{AB - a\bar{a}q_1 q_2}.$$

The above values are substituted into the equations for $P(z)$ and $Q(z)$ and we find $P(1)$ and $Q(1)$ by solving eqs. (45) and (46) simultaneously with $z = 1$ to arrive at

$$P(1) = C_P P_0, \qquad Q(1) = C_Q P_0$$

where

$$C_P = \frac{C_{pp_0} + C_{pq}C_{qp_0} - C_{pp_0}C_{qq}}{1 - C_{pp} - C_{pq}C_{qp} - C_{qq} + C_{pp}C_{qq}},$$
$$C_Q = \frac{C_{pp_0}C_{qp} + C_{qp_0} - C_{pp}C_{qp_0}}{1 - C_{pp} - C_{pq}C_{qp} - C_{qq} + C_{pp}C_{qq}},$$

and

$$C_{pp_0} = \frac{1}{r_1 - 1},$$
$$C_{pp} = \frac{\bar{a}q_2\left(AfA + a\bar{a}q_1\bar{q}_2 - afAr_2\right)}{\left(AB - a\bar{a}q_1 q_2\right)\left(r_1 - 1\right)\left(A - ar_2\right)},$$
$$C_{pq} = \frac{a^2 q_1\left(Afq_2 + B\bar{q}_2 - afq_2 r_2\right)}{\left(AB - a\bar{a}q_1 q_2\right)\left(r_1 - 1\right)\left(A - ar_2\right)},$$
$$C_{qp_0} = \frac{1}{s_1 - 1},$$
$$C_{qp} = \frac{\bar{a}^2 q_2\left(Bfq_1 + A\bar{q}_1 - \bar{a}fq_1 s_2\right)}{\left(AB - a\bar{a}q_1 q_2\right)\left(s_1 - 1\right)\left(B - \bar{a}s_2\right)},$$
$$C_{qq} = \frac{aq_1\left(BfB + a\bar{a}\bar{q}_1 q_2 - \bar{a}fBs_2\right)}{\left(AB - a\bar{a}q_1 q_2\right)\left(s_1 - 1\right)\left(B - \bar{a}s_2\right)}.$$

P_0 is derived from the fact that the probabilities must sum to 1.

Finally, the equations for $P(z)$, $Q(z)$, $S(z)$ and $R(z)$ can be inverted to find the complete solution to the Markov chain.

$$P_k = \left(P_0 + R_0 f\right)\left(\frac{1}{r_1}\right)^k + \\ + \frac{S_0 a\bar{a}f\bar{q}_2\left(\left(\frac{1}{r_1}\right)^k - \left(\frac{a}{A}\right)^k\right)}{\left(A - ar_1\right)\left(A - ar_2\right)}, \quad k \geq 1, \tag{47}$$

$$S_k = S_0\left(\frac{a}{A}\right)^k, \qquad k \geq 0, \tag{48}$$

$$Q_k = \left(P_0 + S_0 f\right)\left(\frac{1}{s_1}\right)^k + \\ + \frac{R_0 a\bar{a}f\bar{q}_1\left(\left(\frac{1}{s_1}\right)^k - \left(\frac{\bar{a}}{B}\right)^k\right)}{\left(B - \bar{a}s_1\right)\left(B - \bar{a}s_2\right)}, \quad k \geq 1, \tag{49}$$

$$R_k = R_0\left(\frac{\bar{a}}{B}\right)^k, \qquad k \geq 0, \tag{50}$$

$$S_0 = C_{S_0} P_0,$$
$$R_0 = C_{R_0} P_0,$$
$$C_{S_0} = \frac{q_1\left(C_Q aB + C_P \bar{a}^2 q_2\right)}{AB - a\bar{a}q_1 q_2},$$
$$C_{R_0} = \frac{\left(C_P \bar{a}A + C_Q a^2 q_1\right) q_2}{AB - a\bar{a}q_1 q_2},$$
$$P_0 = \left(1 + C_{R_0} + C_{S_0} + \frac{C_{S_0}a}{\bar{a}f} + \frac{C_{R_0}\bar{a}}{af} + \right. \\ + \frac{1 + C_{S_0}f}{\left(s_1 - 1\right)} + \frac{1 + C_{R_0}f}{\left(r_1 - 1\right)} + \\ + \frac{C_{S_0}a\bar{q}_2}{\left(r_1 - 1\right)\left(A - ar_2\right)} + \\ \left. + \frac{C_{R_0}\bar{a}\bar{q}_1}{\left(s_1 - 1\right)\left(B - \bar{a}s_2\right)}\right)^{-1}. \tag{51}$$

3.3 Performance Measures

3.3.1 State Buffer Use

Using the state probabilities we find the average state buffer occupancy at processors one and two.

$$\bar{B}_1 = \sum_{i=1}^{\infty} i\left(P_i + S_i\right) = \\ = \frac{AS_0 a}{\bar{a}^2 f^2} + \frac{\left(P_0 + R_0 f\right) r_1}{\left(r_1 - 1\right)^2} + \\ + \frac{S_0 a\bar{a}f\bar{q}_2\left(\frac{r_1}{\left(r_1 - 1\right)^2} - \frac{Aa}{\bar{a}^2 f^2}\right)}{\left(A - ar_1\right)\left(A - ar_2\right)}, \tag{52}$$

$$\overline{B}_2 = \sum_{i=1}^{\infty} i(Q_i + R_i) =$$
$$= \frac{BR_0\bar{a}}{a^2 f^2} + \frac{(P_0 + S_0 f)\, s_1}{(s_1 - 1)^2} +$$
$$+ \frac{R_0 a\bar{a} f\bar{q}_1 \left(\frac{s_1}{(s_1-1)^2} - \frac{B\bar{a}}{a^2 f^2}\right)}{(B - \bar{a}s_1)(B - \bar{a}s_2)}. \quad (53)$$

As with the previous model we also find $\Theta_{i,b}$, the probability that a fixed sized buffer of size $b \geq 1$ overflows.

$$\Theta_{1,b} = \sum_{i=b+1}^{\infty} (P_i + S_i) =$$
$$= \frac{S_0 a}{\bar{a} f}\left(\frac{a}{A}\right)^b + \frac{(P_0 + R_0 f)}{(r_1 - 1) r_1{}^b} +$$
$$+ \frac{S_0 a\bar{a} f\bar{q}_2 \left(\frac{1}{(r_1-1) r_1{}^b} - \frac{a}{\bar{a} f}\left(\frac{a}{A}\right)^b\right)}{(A - ar_1)(A - ar_2)}, \quad (54)$$

$$\Theta_{2,b} = \sum_{i=b+1}^{\infty} (Q_i + R_i) =$$
$$= \frac{R_0\bar{a}}{af}\left(\frac{\bar{a}}{B}\right)^b + \frac{(P_0 + S_0 f)}{(s_1 - 1) s_1{}^b} +$$
$$+ \frac{R_0 a\bar{a} f\bar{q}_1 \left(\frac{1}{(s_1-1) s_1{}^b} - \frac{\bar{a}}{af}\left(\frac{\bar{a}}{B}\right)^b\right)}{(B - \bar{a}s_1)(B - \bar{a}s_2)}. \quad (55)$$

3.3.2 Speedup

From the complete solution of the Markov chain we calculate the speedup S of the two processor TW system over an equivalent single processor. The speedup is simply the rate of the two processor system δ_2 divided by the rate of progress for a single processor system δ_1. The rate of forward progress for one processor is defined simply as the average rate of progress of the two processes

$$\delta_1 = (\lambda_1 + \lambda_2)/2.$$

At this point we add an additional cost for state saving by allowing a single processor to move at a rate that is C times faster than the TW processors. Thus, state saving increases the average execution time of an event from $1/\lambda_i$ to C/λ_i when running TW. The revised rate of progress for a single processor is

$$\delta_1 = C(\lambda_1 + \lambda_2)/2,$$

while the rate of progress for the two-processor TW system is found from the following equation.

$$\delta_2 = (\lambda_1 + \lambda_2)P_0 + (\lambda_1 + \beta_2)S_0 + (\lambda_2 + \beta_1)R_0 +$$
$$+ (\lambda_1 + \lambda_2)P(1) + (\lambda_2 + \lambda_1)Q(1) +$$
$$+ (\lambda_1 + \beta_2)S(1) + (\lambda_2 + \beta_1)R(1)$$
$$- \lambda_2 q_2 \sum_{k=1}^{\infty} P_k(k-1) - \lambda_1 q_1 \sum_{k=1}^{\infty} Q_k(k-1)$$
$$- \beta_2 q_2 \sum_{k=1}^{\infty} S_k(k-1) - \beta_1 q_1 \sum_{k=1}^{\infty} R_k(k-1).$$

Taking the ratio $S = \delta_2/\delta_1$ (i. e., the speedup) we arrive at

$$S = \frac{2}{C}\left(P_0 + P(1) + Q(1) + \frac{B^2 R_0}{af} + \frac{A^2 S_0}{\bar{a} f}\right.$$
$$- \frac{R_0\bar{a}^2 q_1}{af} - \frac{S_0 a^2 q_2}{\bar{a} f}$$
$$- \bar{a}q_2\left(\frac{P_0 + R_0 f}{(r_1 - 1)^2} +\right.$$
$$\left. + \frac{S_0 a\bar{a} f\bar{q}_2 \left(\frac{1}{(r_1-1)^2} - \frac{a^2}{\bar{a}^2 f^2}\right)}{(A - ar_1)(A - ar_2)}\right)$$
$$- aq_1\left(\frac{P_0 + S_0 f}{(s_1 - 1)^2} +\right.$$
$$\left.\left. + \frac{R_0 a\bar{a} f\bar{q}_1 \left(\frac{1}{(s_1-1)^2} - \frac{\bar{a}^2}{a^2 f^2}\right)}{(B - \bar{a}s_1)(B - \bar{a}s_2)}\right)\right) \quad (56)$$

We note here that this measure S is different from the measure Γ used with the message queueing model in Section 2.3.3In that model we were unable to calculate the average rate of progress for a single processor due to the effect of messages. Since messages carried work, it would be unfair to compare the two processor TW system to a single processor system without messages. The TW system would be doing more work. On the other hand, it is non-trivial to attempt to account for this extra work caused by messages and add it to the single processor system. We finally settled on a measure that was a normalized rate of progress by dividing the rate of progress for two processors by $(\lambda_1 + \lambda_2)$. For the rollback cost system the rate of progress on a single processor is well defined and therefore, we use a speedup measure S.

For the Symmetric, Balanced case where $\lambda_1 = \lambda_2 = \lambda$ and $q_1 = q_2 = q$ we get the following equation for speedup.

$$S = 4f(f + \sqrt{q}) \Big/ \left[C\left(2f^2 + f(2+f)\sqrt{q} + \right.\right.$$
$$\left.\left. + (2-f) fq + 2(1-f) q^{3/2}\right)\right]. \quad (57)$$

We show a plot of this function in Fig. 9 for no additional cost for state saving ($C = 1$).

Using this simple formula for speedup we find the values of f, q, and C that allow two processors running TW to progress faster than a single processor without TW. This is the region where $S \geq 1$. We solve eq. (57) for C when $S \geq 1$ resulting in the inequality

$$C \leq 4f(f + \sqrt{q}) \Big/ \left(2f^2 + f(2+f)\sqrt{q} + \right.$$

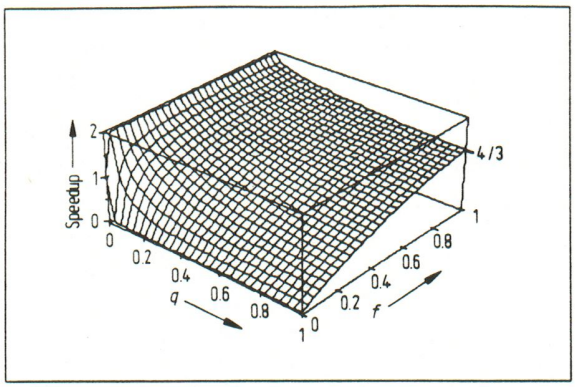

Fig. 9. Speedup versus q and f for the Symmetric, Balanced Case with no additional cost for state saving ($C = 1$).

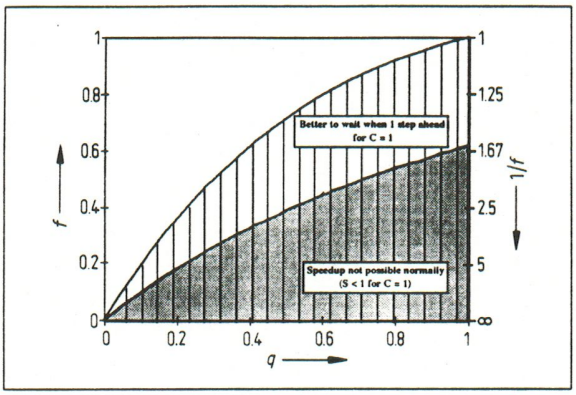

Fig. 10. Region of $q - f$ space where speedup is possible.

$$+ (2 - f) fq + 2(1 - f) q^{3/2}\Big). \qquad (58)$$

Therefore, we find that C must lie below the surface plotted in Fig. 9 for $S \geq 1$. It is clear for $C > 2$ that TW on two processors is always slower than using a single processor without TW. Further, since C must be greater than or equal to one (cost of state saving is ≥ 0), there is a region in the $q - f$ space where speedup is not possible. That is the shaded region shown in Fig. 10.

Since rollbacks can be costly ($C > 1$), there may be an advantage to slowing down or stopping the faster processor when it gets ahead so as to avoid rollbacks. Mitra and Mitrani [6], using their optimization function $J = D - cR$ (see Section 1.), find regions of the parameter space where the maximum of the function is found at the boundary where the processors have zero processing capacity (don't perform the task at all). Essentially, they found that Time Warp could perform poorly if the cost for rollback was high. We, on the other hand, will look to improve TW by slowing down or stopping the processor that gets too far ahead. Looking again at the Symmetric, Balanced case where $\lambda_1 = \lambda_2 = \lambda$ and $q_1 = q_2 = q$ we find that region of $q - f$ space where it is better for a processor to stop processing when it gets *exactly* one step ahead. The state diagram for

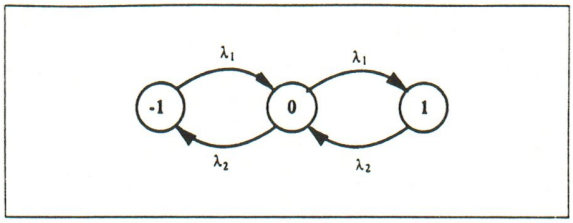

Fig. 11. State diagram when each processor stops when one step ahead.

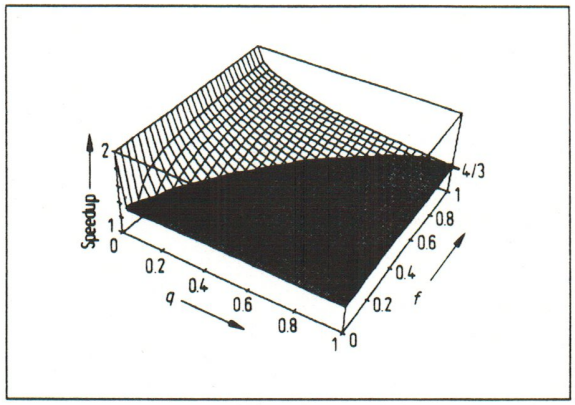

Fig. 12. Achievable Speedup for $C = 1$.

such a system is shown in Fig. 11. Each processor will stop when it gets exactly one step ahead of the other processor. There will be no rollbacks and therefore no need for state saving. When $\lambda_1 = \lambda_2 = \lambda$ we find that $P_1 = Q_1 = P_0 = 1/3$, and that speedup over the equivalent single processor system is $4/3$. Therefore, we can always get a speedup of $4/3$ regardless of the values of f, q and C. For general values of λ_1 and λ_2 the speedup is

$$S = 4a(1 - a)/(1 - a + a^2)$$

that has its maximum of $4/3$ at $a = 1/2$. For the Symmetric, Balanced case we show in Fig. 10 the area of the $q - f$ plane where waiting at one step is better than rushing ahead when $C = 1$. Fortunately this area includes all the area where we would not have been able to get speedup with two processors. Finally, in Fig. 12 we show the achievable speedup when $C = 1$. The shaded region is where a processor waits when it gets one step ahead of the other. In the unshaded region, if C is less than the value plotted in the figure we are able to gain at least some speedup over the equivalent single processor not running Time Warp.

Since it sometimes pays to stop a processor when it gets one step ahead, we might surmise that there are ranges of the parameters where stopping a processor when it gets k ($k > 1$) steps ahead improves performance. For our model, this turns out not to be the case. By examining the Markov chain for $k = 2$ we find that the speedup is never greater than the speedup gained by the standard algorithm. Therefore, it is never practical to stop a processor once it gets more than one step ahead. The Markov chain in Fig. 11 is unique in

the respect that at no point in time will a processor incur a cost for state saving or rollback. Once we allow the processors to get more than one step out of synchronization, we must save state since rollbacks are possible. Intuitively, the fact that we might only stop at one step ahead makes sense since a process at virtual time v can only send a message to the other at time $v+1$. By getting two or more steps ahead, a rollback is already possible and we will incur a cost for rollback if a message is sent regardless of whether we wait further down the line. Waiting now only causes the system to have a smaller speedup. In a more general system where a processor may send a message to an arbitrary point in the future we may find that there are regions of the parameter space where it pays to stop a processor when it gets further than one step ahead. We are currently extending the rollback cost model so that the processors are able to make arbitrary sized jumps when advancing (not restricted to single-steps). This model will give us a better opportunity to examine the improvements we might gain by stopping or slowing down the lead processor when it gets more than one step ahead.

3.4 Summary

Our second model incorporated costs for rollback and state saving. In addition to calculating the complete solution to the Markov chain and the speedup over a single processor, we were able to find regions of the parameter space where it was better to stop either processor when it was exactly one step ahead. We could also show that stopping the lead processor when it was two or more steps ahead led to no performance gain. As with our previous model, since we have the exact solution to the Markov chain, we are able to calculate nearly any performance measure of interest.

4. Conclusions and Future Work

In this paper we presented two new models to extend our understanding of the Time Warp distributed simulation protocol when it runs on two processors. Our first model allowed messages to be queued that had not been previously addressed in any of the work on two-processor models. Our second model incorporated costs for both rollback and state saving. In this second model we were able to find regions of the parameter space where it was better to stop a processor when it got ahead of the other one rather than let it rush ahead and potentially incur a cost for state saving and rollback. Both models have given us a clearer and more thorough understanding of the operation of systems synchronized by rollback when run on two processors.

In addition to extending the rollback cost model to accommodate arbitrary sized state advances, our future work will be in the area of extensions to multiple processors. Extending our Markov chain approach has proven to be unwieldy, and we have been pursuing approximations for multiple processors [13].

Acknowledgement

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

Appendix

A Solution to the Cubic Equation

This material is taken directly from the CRC Handbook of Mathematical Sciences [21].

A cubic equation, $y^3 + p_y y^2 + q_y y + r_y = 0$ may be reduced to the form,

$$x^3 + a_x x + b_x = 0$$

by substituting for y the value $x - p_y/3$. Here

$$a_x = (3q_y - {p_y}^2)/3,\ b_x = (2{p_y}^3 - 9p_y q_y + 27r_y)/27.$$

The form $x^3 + ax + b = 0$ with $ab \neq 0$ can always be solved by transforming it to the trigonometric identity

$$4\cos^3(\theta) - 3\cos(\theta) - \cos(3\theta) \equiv 0.$$

Let $x = m\cos(\theta)$, then

$$\begin{aligned} x^3 + ax + b &\equiv 0 \\ &\equiv m^3\cos^3(\theta) + am\cos(\theta) + b \\ &\equiv 4\cos^3(\theta) - 3\cos(\theta) - \cos(3\theta) \\ &\equiv 0. \end{aligned}$$

Hence

$$\frac{4}{m^3} = -\frac{3}{am} = \frac{-\cos(3\theta)}{b},$$

from which it follows that

$$m = 2\sqrt{-a/3}, \quad \cos(3\theta) = 3b/(am).$$

Any solution θ_1 which satisfies $\cos(3\theta) = 3b/(am)$, will also have the solutions

$$\theta_1 + 2\pi/3, \quad \text{and} \quad \theta_1 + 4\pi/3.$$

The roots of the cubic $x^3 + ax + b = 0$ are therefore

$$\begin{aligned} x_1 &= m\cos(\theta_1 + 2\pi/3), \\ x_2 &= m\cos(\theta_1) \\ x_3 &= m\cos(\theta_1 + 4\pi/3). \end{aligned}$$

For the denominator of $P(z)$ in eq. (14) we have

$$\begin{aligned} r_y &= -\frac{\bar{a}(A - aq_1)\bar{q}_2}{a^2}, \\ q_y &= \frac{A + a\bar{a}q_1\bar{q}_2}{a^2}, \\ p_y &= -\frac{1+A}{a}. \end{aligned}$$

These values can then be substituted into the solutions given above to find r_1, r_2, and r_3. The values for s_i are symmetric in $(a, \bar{a})$ and (q_1, q_2) to the r_i values.

References

[1] Jefferson, D. R.: Virtual time. ACM Trans. Programming Languages and Systems **7** (1985), 404–425.

[2] Felderman, R. E.; Kleinrock, L.: Two processor time warp analysis: Some results on a unifying approach. Proc. 5th Workshop on Parallel and Distributed Simulation (PADS'91), Jan. 1991. 3–10.

[3] Kleinrock, L.; Felderman, R. E.: Two processor time warp analysis: A unifying approach. Int. J. Computer Simulation **2** (1992), 345–371.

[4] Kleinrock, L.: On distributed systems performance. Proc. 7th ITC Specialist Seminar, Adelaide, Australia, Sep. 1989. (Also published in "Computer Networks and ISDN Systems" vol. 20, no.1-5, pp. 206-215, Dec. 1990.).

[5] Lavenberg, S.; Muntz, R.; Samadi, B.: Performance analysis of a rollback method for distributed simulation. – In: Performance '83. Amsterdam: North-Holland, 1983, 117–132.

[6] Mitra, D.; Mitrani, I.: Analysis and optimum performance of two message-passing parallel processors synchronized by rollback. – In: Performance '84. Amsterdam: North-Holland, 1984, 35–50.

[7] Lin, Y.-B.; Lazowska, E. D.: Optimality considerations for "Time Warp" parallel simulation. Proc. SCS Multiconf. Distributed Simulation, Jan. 1990. Society for Computer Simulation, 29–34.

[8] Lin, Y.-B.; Lazowska, E. D.: Reducing the state saving overhead for time warp parallel simulation. Tech. Rept. 90-02-03. Dept. Computer Science and Engg., University of Washington, Feb. 1990.

[9] Madisetti, V. K.: Self synchronizing concurrent computing systems. Tech. Rept. UCB/ERL M89/122. Electron. Res. Lab., University of California, Berkeley, Oct. 1989.

[10] Madisetti, V.; Walrand, J.; Messerschmitt, D.: Synchronization in message-passing computers: Models, algorithms and analysis. Proc. SCS Multiconf. Distributed Simulation, Jan. 1990. Society for Computer Simulation, 35–48.

[11] Nicol, D. M.: Parallel self-initiating discrete-event simulations. ACM Trans. Modelling and Computer Simulation **1** (1991), 24–50.

[12] Nicol, D. M.: The cost of conservative synchronization in parallel discrete event simulations. Tech. Rept. 90-20. Inst. Computer Appl. in Science and Engg.(ICASE), May 1990.

[13] Felderman, R. E.; Kleinrock, L.: Bounds and approximations for self-initiating distributed simulation without lookahead. ACM Trans. Modelling and Computer Simulation **1** (1991), 386–406.

[14] Wagner, D. B.: Conservative parallel discrete-event simulation: Principles and practice. Tech. Rept. 89-09-03. Dept. Computer Science and Engg., University of Washington, Sep. 1989.

[15] Misra, J.: Distributed discrete-event simulation. Computing Surveys **18** (1986), 39–65.

[16] Peacock, J. K.; Wong, J. W.; Manning, E. G.: Distributed simulation using a network of processors. Computer Networks **3** (1979), 44–56.

[17] Lamport, L.: Time, clocks, and the ordering of events in a distributed system. Commun. of the ACM **21** (1978), 558–564.

[18] Bellenot, S.: Global virtual time algorithms. Proc. SCS Multiconf. Distributed Simulation, Jan. 1990. Society for Computer Simulation, 122–127.

[19] Fujimoto, R. M.: Time warp on a shared memory multiprocessor. Tech. Rept. UUCS-88-021a, Salt Lake City: Computer Science Dept., University of Utah, Jan. 1989.

[20] Felderman, R. E.: Performance analysis of distributed processing synchronization algorithms. Tech. Rept. 910019. Computer Science Dept., University of California, Los Angeles, June 1991. Ph.D. Dissertation.

[21] Beyer, W. H. (ed.): CRC handbook of mathematical sciences. 6th ed. CRC Press, 1987.

Robert E. Felderman is a Computer Scientist at USC Information Sciences Institute and a Research Assistant Professor in the Computer Science Department at USC. 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 with Hughes Aircraft Company working on guidance systems for torpedos, he returned to the good life of academia, completed his Master's and Ph.D. degree in Computer Science at UCLA in 1986 and 1991 respectively. In 1991 he was named one of three Outstanding Ph.D. Students by the UCLA School of Engineering and Applied Science. His Ph.D. research focused on performance analysis of distributed systems. His current research interests include high-speed local area networking, distributed and parallel systems, distributed simulation and performance analysis.

Dr. Felderman is a member of ACM, IEEE, SCS, Tau Beta Pi and Sigma Xi. He serves as a program committee member for the Workshop on Parallel and Distributed Simulation and as a reviewer for various conferences and journals.

Leonard Kleinrock is Chair and Professor of Computer Science at the University of California, Los Angeles, since 1963. He received his B.S. degree in Electrical Engineering from the City College of New York in 1957 and his M.S.E.E. and Ph.D.E.E. degrees from the Massachusetts Institute of Technology in 1959 and 1963, respectively. His research interests focus on performance evaluation of high speed networks and parallel and distributed systems. He has had over 180 papers published and is the author of five books. He is the principal investigator for the DARPA Advanced Networking and Distributed Systems grant at U.C.L.A. He is also founder and CEO of Technology Transfer Institute, a computer-communications seminar and consulting organization located in Santa Monica.

Dr. Kleinrock is a member of the National Academy of Engineering, is a Guggenheim Fellow, an IEEE Fellow, and a founding member of the Computer Science and Telecommunications Board of the National Research Council. He has received numerous best paper and teaching awards, including the ICC 1978 Prize Winning Paper Award, the 1976 Lanchester Prize for outstanding work in Operations Research, and the Communications Society 1975 Leonard G. Abraham Prize Paper Award. In 1982, as well as having been selected to receive the C.C.N.Y. Townsend Harris Medal, he was co-winner of the L. M. Ericsson Prize, presented by His Majesty King Carl Gustaf of Sweden, for his outstanding contribution in packet switching technology. In July of 1986, Dr. Kleinrock received the 12th Marconi International Fellowship Award, presented by His Royal Highness Prince Albert, brother of King Baudoin of Belgium, for his pioneering work in the field of computer networks. In the same year, he received the UCLA Outstanding Teacher Award. In 1990, he received the ACM SIGCOMM award recognizing his seminal role in developing methods for analyzing packet network technology.

E 20 298 F
ISSN 0001-1096
No. 5/6 · pp. 273–466
September/November 1993
Volume 47

AEÜ Archiv für Elektronik und Übertragungstechnik

International Journal of Electronics and Communications

Special Issue on Teletraffic Theory and Engineering in Memory of Félix Pollaczek

Guest Editors: J. W. Cohen
P. J. Kühn
F. Schreiber
R. Syski

(Contents continued on back cover)