Tradeoff Between Sequential and Time Warp Based Parallel Simulation*

Francesco QUAGLIA
Dipartimento di Informatica e Sistemistica
Università "La Sapienza"
Via Salaria 113, 00198 Roma, Italy
E.mail: quaglia@dis.uniroma1.it

Vittorio CORTELLESSA
Dipartimento di Informatica Sistemi e Produzione
Università di Roma "Torvergata"
Via di Tor Vergata, 00133 Roma, Italy
E.mail: cortelle@info.uniroma2.it

Bruno CICIANI
Dipartimento di Informatica e Sistemistica
Università "La Sapienza"
Via Salaria 113, 00198 Roma, Italy
E.mail: ciciani@dis.uniroma1.it

Abstract

Discrete event simulation is a methodology to study the behavior of complex systems. Its drawback is that, in order to get reliable results, simulations have to be usually run over a long stretch of time. This time requirement could decrease through the usage of parallel or distributed computing systems. In this paper we analyze the Time Warp synchronization protocol for parallel discrete event simulation and present an analytical model evaluating the upper bound on the completion time of a Time Warp simulation. In our analysis we consider the case of a simulation model with homogeneous logical processes, where "homogeneous" means they have the same average event routine time and the same state saving cost. Then we propose a methodology to determine when it is time-convenient to use a Time Warp synchronized simulation, instead of a sequential one, for a simulation model with features matching those considered in our analysis. We give an answer to this question without the need to preliminary generate the simulation code. Examples of methodology usage are reported for the case of both a synthetic benchmark and a real world model.

*This paper appeared in IEEE Transactions on Parallel and Distributed Systems (vol.10, no.8, August 1999).

Index terms: Virtual Time, Discrete Event Simulation, Time Warp, Rollback-Recovery, Parallel and Distributed Simulation, Performance Evaluation.

1 Introduction

A parallel discrete event simulator consists of a set of *Logical Processes* (LPs) modeling distinct parts of the simulated system [12]. Each LP executes a sequence of events. The execution of an event possibly triggers a change in the state of the LP and schedules events to be executed at a later simulation time. The scheduling of an event for an LP is realized by sending to it a message with the content and the occurrence time, namely *timestamp*, of the event.

LPs synchronize in order to guarantee *causality* (i.e., timestamp ordered execution at each LP). In conservative synchronization [3] the execution of an event with timestamp t at a given LP is allowed whenever no event with timestamp smaller than t is guaranteed to be ever scheduled for that LP in the progress of the simulation. Instead optimistic synchronization, namely *Time Warp* [19, 20], allows LPs to execute events greedily (i.e., whenever they are available) and uses checkpoint-based rollback for recovering from timestamp order violations.

As synchronization imposes overhead, the question "is it profitable to implement a given simulation model in a parallel way instead of in a sequential one?" comes up quite straightforwardly. In this paper we give an answer to this question for the case of Time Warp synchronization applied to simulation models with homogeneous LPs (i.e., LPs having the same average event routine time and the same state saving cost). This is done by introducing an analytical model for evaluating the upper bound on the completion time of a Time Warp simulation.

In order to compare sequential and Time Warp simulations our analysis requires the knowledge of basic features of the simulation model (e.g., number of LPs and how they interact) and of the hardware/software platform. No simulation code is required for the comparison. In this particular aspect our approach differs from the one in [37] where, for an already existing sequential simulation program based on the process interaction model, a technique for evaluating performance of conservative synchronization is presented.

In our analysis of Time Warp synchronization we do not assume that the overhead due to state saving and rollback is negligible (as supposed by several previous analytical models [18, 21]). Furthermore, the effects of aggregation of LPs onto processors is taken into account. Our major assumptions are: (i) load is balanced among processors (i.e. the mean time spent executing events and the mean time spent for the rollback/recovery mechanism is the same on all processors) and (ii) the rollback behavior of the simulation reaches a steady state. These assumptions are commonly accepted in the literature [18, 21, 26] in order to build analytically tractable models.

The remainder of the paper is organized as follows. A short description of Time Warp basic concepts is given in Section 2. An overview of papers dealing with Time Warp modeling is proposed in Section 3. Section 4 describes our analytical model. The methodology for the choice between sequential and parallel implementation is reported in Section 5. In Section 6 some experimental results highlighting the effectiveness of the methodology are shown for both a synthetic benchmark and a real world model. Conclusions are reported in Section 7.

2 Time Warp Description

In Time Warp synchronization each LP has its own notion of simulation time, namely *Local Virtual Time* (LVT). LPs interact solely by exchanging messages. Any message exchange represents the scheduling of an event for the recipient LP. Each message carries the content of the scheduled event and is "stamped" with a *virtual send time* and a *virtual receive time*. Virtual send time is the LVT of the sender LP at the time the message is sent. Virtual receive time, also known as timestamp, represents the simulation time for the occurrence of that event at the recipient LP. Whenever an event is executed by an LP, its LVT moves to the timestamp of the event. In the remainder of the paper we use "event" and "message" as synonymous, "event execution" and "message processing" as synonymous as well.

LPs process messages whenever they are available, under the optimistic assumption of a timestamp ordered execution. Each time a timestamp order violation is detected by an LP (i.e., a message timestamped in the past of its LVT arrives), it rolls back to its state immediately prior the violation and the execution resumes [19]. While rolling back, the LP cancels some sent messages by sending antimessages. Each antimessage signals to the recipient LP that the corresponding message has to be discarded. Upon the arrival of an antimessage corresponding to an already processed message, the recipient LP rolls back as well. Hence, we can distinguish between rollback generated by the arrival of a message with timestamp in the past of the LVT of the recipient LP (*straggler* message) and rollback due to antimessages. We refer to the former as *primary* rollback, to the latter as *secondary* rollback.

Incoming messages are stored by the LP into an *input-queue*, ordered by timestamps. An *output-queue* is used to store antimessages (i.e., whenever a message is sent, the corresponding antimessage is generated and stored into this queue). Antimessages are ordered by virtual send times. When the LP rolls back to virtual time t, all antimessages stamped with virtual send time greater than t are sent. Both *aggressive* and *lazy* sending schemes have been proposed [15]. In the aggressive scheme antimessages are sent as soon as the LP rolls back. Instead, in the lazy scheme, they are placed into a queue of pending antimessages; after resuming the execution, if a message is produced identical to one that would have been unsent during rollback, then the new message and the corresponding antimessage are discarded, otherwise the antimessage is sent.

State recovery at a past simulation time is guaranteed by an underlying checkpointing scheme. Commonly used schemes are *periodic* and *incremental* state saving. Under periodic state saving the state of the LP is saved each χ event executions (χ being the checkpoint interval) [24]. State recovery to an unsaved state is realized by restoring the latest checkpoint preceding that state and re-updating state variables through the content of intermediate events (*coasting forward*). Under incremental state saving [2, 36] a history of before images of state variables modified during message processing is maintained. State recovery is realized by backward crossing the logged history and copying old values into their original state locations. Under both checkpointing schemes, values of state variables can be saved either via software or through the usage of special purpose hardware [14].

The Global Virtual Time (GVT) is defined as the lowest value among the timestamps of messages either not yet processed or currently being processed or in transit. The GVT value represents the commitment horizon of the simulation because no rollback to a simulation time preceding GVT can ever occur. The GVT notion is used to reclaim memory allocated for obsolete messages/antimessages and state information, and to allow operations that cannot be undone (e.g., I/Os, displaying of intermediate simulation results, etc.). The memory reclaiming procedure is known as *fossil collection.*

3 Related Work

The complexity of the analysis of the rollback parameters pushed the earliest performance models of Time Warp synchronization to be focused on the context of 2-processor scenarios [8, 21, 26, 29]. Performance analyses for the case of multiple processors have been carried out in more recent years. In [23] it is shown that Time Warp performs better than conservative synchronization whenever the checkpointing-rollback cost is negligible. Time Warp synchronization for the case of self-initiating models has been analyzed in [27], where an upper bound on the performance is derived neglecting the effects of rollback propagation (i.e., cascading rollbacks due to the spreading of antimessages are not modeled in the analysis). This model is mainly focused on the effects of communication on synchronization. The case of self-initiating models has been studied also in [9] where, under the assumption of negligible checkpointing-rollback cost, upper and lower bounds on performance of Time Warp are derived by modeling the progress of the GVT of the simulation. The model in [18] describes performance of Time Warp synchronization through a Markov-chain analysis, under the assumptions of exponential distribution for the timestamp increments, negligible checkpointing-rollback overhead and negligible communication delays.

Modifications to the classical Time Warp synchronization have been proposed to embed barrier synchronizations in order to allow optimism of event execution at a limited extent [1, 4, 7, 25, 31, 32, 35]. Among the outcoming protocols, a complete performance model has been presented in [6] for the case of Bounded Time Warp [35]. In this protocol a global window is iteratively defined and, in any iteration, LPs are allowed to process only the messages with timestamp in that window. The analysis in [6] compares performance of Bounded Time Warp to that of a conservative protocol for the case of a queuing network under heavy load for each server. Such analysis considers the effects of the aggregation of LPs onto processors which are not outlined in most previous models.

For what concerns Time Warp with periodic checkpointing, several performance analyses versus the checkpoint interval have been presented [24, 28, 30].

A performance optimization methodology versus both the number of processors and the checkpoint interval appears in [5]. Such a methodology relies on an analytical model describing the rollback parameters, which takes into account rollback propagation due only to the first wave of antimessages. The model is not solved in closed form, so its application is based on an iterative procedure using observed values of the rollback parameters of the simulation.

Rather different approaches to run time optimization of performance through the reduction of the rollback probability are: the usage of throttling (i.e., delaying event execution) [10, 11] and the reduction of distances between local simulation clocks at distinct LPs through dynamic load balancing based on active migration of LPs [17].

Compared to performance models in literature, our model has features similar to that in [6] (although we analyze a different synchronization protocol - i.e., Time Warp instead of Bounded Time Warp). In our analysis the costs of checkpointing and of rollback are not neglected; furthermore the aggregation of LPs onto processors is taken into account, as the upper bound we state on the execution time of a Time Warp simulation depends also on the probability of communication between LPs running on remote processors. Two main differences exist between the model in [6] and our one: we consider communication as instantaneous (like in [18]); we do not restrict the analysis to a particular simulation model (e.g., queuing networks), homogeneity of LPs is, instead, our only assumption on the simulation model.

4 The Model

We consider a Time Warp system structured as follows. Processors are homogeneous and there is an instance of the Time Warp kernel on each processor. Communication between processors is supposed to be instantaneous (i.e., the delivery delay of a message due to the communication network is supposed to be zero; a time overhead due to the message packing/unpacking is, however, taken into account). The Time Warp kernel manages the local event list consisting of the logical collection of all input queues of local LPs. LPs are scheduled for event execution according to the Smallest-Timestamp-First policy [20]. Rollback is non-preemptive: if a message/antimessage with timestamp in the past of the LVT of the recipient LP arrives while an event is being executed, then the rollback does not take effect until the end of the event execution [18]. Antimessages are sent aggressively. Checkpoints are taken periodically and the checkpoint period has the same value χ for all LPs; furthermore, coasting forward is executed in atomic fashion [12]. The amount of memory is supposed enough large to allow the completion of the simulation without executing fossil collection; therefore time spent for fossil collection is here neglected.

The following basic notations are adopted: p denotes the number of processors, n_{LP} denotes the number of LPs, P_r denotes the rollback probability (i.e., the ratio between the number of rollbacks and the number of executed events), and L_r denotes the average rollback length (i.e., the average number of undone events at each rollback occurrence). Additional notations are introduced where they will be needed.

In our analysis we assume:

a) LPs are homogeneous, that is: (i) the average execution time for the event routine has the same value for all LPs, (ii) the average state saving time has the same value for all LPs[1];

[1]Note that homogeneity of LPs does not imply that their distribution functions for the timestamp increment have the same mean values.

b) the total amount of forecast simulation events is equally distributed among the processors;

c) the rollback behavior of the simulation reaches a steady state; furthermore, the rollback probability P_r and the average rollback length L_r assume the same value for all LPs;

d) at least one non-executed event is stored in the local event list at any time;

e) each event execution schedules a single new event.

From the features of the considered Time Warp system and from assumptions d) and e), the flow-graph for the execution of a simulation event on whichever processor is shown in Figure 1. In correspondence to the decision block *rollback*, the path labeled YES is executed with rate P_r. The YES path outgoing from the decision block *checkpoint* has rate $1/\chi$.

In the following subsections we pass through the steps listed below:

- the relation between the number of forecast events and the total number of events of the parallel execution is stated as a function of the rollback parameters P_r and L_r;
- an analysis of the rollback parameters is presented and upper bounds on P_r and on the product $P_r L_r$ are then derived;
- the upper bound on the simulation execution time is obtained as a function of the bounds on P_r and $P_r L_r$.

4.1 Number of Events of the Parallel Execution

The total number of events N_{par} executed in a Time Warp simulation consists of two components: N_{seq}, the forecast events (corresponding to those events that would be executed by a sequential simulator), and N_{roll}, the events undone by rollback occurrences. Therefore, the following equality holds:

$$N_{par} = N_{seq} + N_{roll} \tag{1}$$

The number of rolled-back events N_{roll} is given by the number of rollback occurrences $P_r N_{par}$ multiplied by the average rollback length L_r, i.e.:

$$N_{roll} = (P_r N_{par}) L_r \tag{2}$$

Substituting (2) in (1) we get:

$$N_{par} = N_{seq} + (P_r N_{par}) L_r \tag{3}$$

and by (3), through simple algebra, the relation between N_{par} and N_{seq} is stated as a function of the rollback parameters as follows:

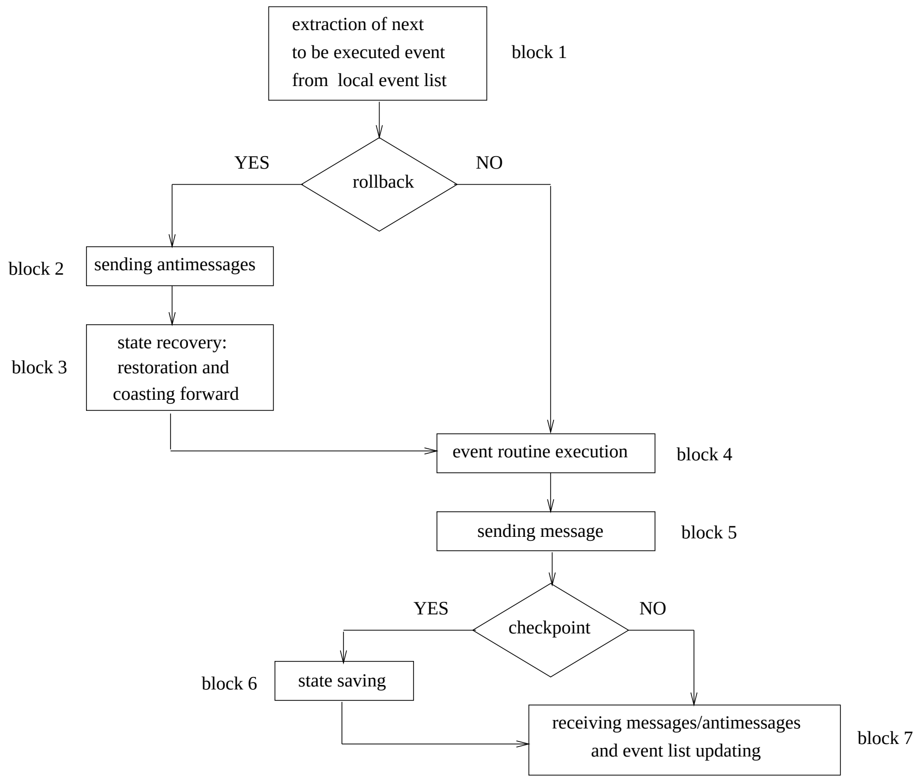
extraction of next to be executed event from local event list
block 1
rollback
YES
NO
block 2
sending antimessages
block 3
state recovery: restoration and coasting forward
event routine execution
block 4
sending message
block 5
checkpoint
YES
NO
block 6
state saving
receiving messages/antimessages and event list updating
block 7

Figure 1: Flow-Graph for the Execution of a Simulation Event.

$$N_{par} = \frac{N_{seq}}{1 - P_r L_r} \tag{4}$$

Expression (4) can be rewritten as:

$$N_{par} = \frac{1}{P_r} \frac{N_{seq}}{(\frac{1}{P_r} - L_r)} \tag{5}$$

where $1/P_r$ is the average distance, in terms of number of events, between two successive rollback occurrences (this quantity is commonly referred to as *rollback interval* [24]). Therefore $(1/P_r - L_r)$ is the average number of events which are not undone in a rollback interval. The quantity $N_{seq}/(1/P_r - L_r)$ is the number of rollback intervals of a simulation with N_{seq} forecast events. Multiplying latter quantity by the rollback interval we obtain, as expected, the total number of executed events.

The consistency of expression (4) relies on the constraint:

$$L_r < \frac{1}{P_r} \tag{6}$$

implying the termination of the simulation (i.e., the total number of events of the parallel execution is a bounded positive quantity). In other words, if the rollback length is less than the rollback interval, then, at any rollback occurrence, the simulation does not move as far back as the last rollback point, thus implying simulation time progress. The investigation on requirements for progress and termination of the simulation is out of the aim of this paper. A formal approach to progress and termination has been presented in [22].

4.2 Rollback Parameters

In this section we first determine analytical expressions for the rollback parameters P_r and L_r; upper bounds on P_r and on the product $P_r L_r$ are then derived.

Let us define *scheduling cycle* the set of activities performed at each processor for the execution of a simulation event (i.e., the activities of the flow-graph in Figure 1). On each processor the simulation consists of a sequence of scheduling cycles. Homogeneity of processors together with assumptions a) and c) imply that the average time spent for a scheduling cycle is the same on all processors. Furthermore, assumption d) implies that no processor is idle till the end of the simulation. Hence, the sequence of scheduling cycles can be thought, on the average, as a step by step computation (see Figure 2), where a set of p concurrently executed scheduling cycles is referred to as *advancement step* of the simulation.

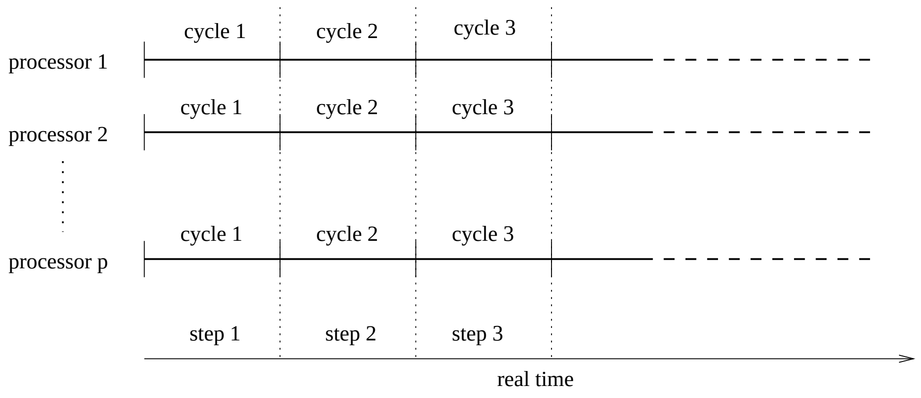

Figure 2: Step by Step Advancement of the Simulation.

As communications are assumed to be instantaneous, any message/antimessage sent in the n-th advancement step arrives at the recipient LP before the end of that advancement step. Therefore, at the beginning of each advancement step no message/antimessage is still in transit (i.e., each processor is completely concerned about all the messages/antimessages already generated in the system and addressed to it). As a consequence, the GVT of the simulation is recorded on one of the p processors. This is exploited in our analysis (see Section 4.2.3) to derive the expressions for the rollback parameters.

4.2.1 Analysis of the Rollback Probability

We model the rollback probability P_r as the sum of several contributions: one due to the arrival of a straggler message (i.e., primary rollback), all the others due to the propagation of antimessages (i.e., secondary rollback). The expression stated for P_r is:

$$P_r = Q_p + Q_pQ_s + Q_pQ_s^2 + \ldots + Q_pQ_s^i + \ldots = \sum_{k=0}^{\infty} Q_pQ_s^k = \frac{Q_p}{1-Q_s} \tag{7}$$

where Q_p denotes the probability to execute a primary rollback and Q_s denotes the probability of rollback propagation through antimessages. Therefore, Q_s^i models the probability of a cascading rollback spreading in i steps, and the quantity $Q_pQ_s^i$ represents the probability that some LP has executed a primary rollback and the rollback propagates in i steps. Each propagation step is due to antimessages timestamped in the past of the LVT of the recipient LP[2].

Consistency of (7) is given by inequality $Q_p + Q_s \leq 1$; at the end of this subsection it will be shown that such inequality holds for the expressions below derived for Q_p and Q_s.

Because of the Smallest-Timestamp-First scheduling policy, there is no primary rollback generation among LPs running on the same processor [19]. Therefore a straggler message always implies communication between LPs running on distinct processors. As a consequence, we state for Q_p the following expression:

$$Q_p = \alpha M_p \tag{8}$$

where:

- α is the probability that an arriving message has been sent by an LP running on a remote processor;
- M_p is the probability that the timestamp of an arriving message is in the past of the LVT of the recipient LP.

The probability Q_s can be expressed as:

$$Q_s = \beta A_p \tag{9}$$

where:

- β is the probability for an LP to receive at least one antimessage (i.e. the probability that some rolling back LP produces at least one antimessage addressed to it);
- A_p is the probability that the timestamp of an arriving antimessage is in the past of the LVT of the recipient LP.

Quantity α represents the probability of remote communication; it takes into account the mapping of LPs onto processors. If frequently communicating LPs are aggregated on the same processor, then the value of α is kept low, thus implying low values for Q_p (i.e., primary rollbacks are infrequently originated due to infrequent remote communication). The quantity α is found out,

[2]As an LP receiving an antimessage timestamped in the past of its LVT at the m-th advancement step may be scheduled for running in the n-th advancement step, with $n \geq m+1$, the rollback propagation steps not necessarily correspond to consecutive advancement steps.

once stated a mapping, by computing the mean communication rate over connections between LPs running on distinct processors.

For what concerns β, in any advancement step of the simulation each of the p running LPs is rolling back with probability P_r. As, on the average, L_r antimessages are sent by a rolling back LP, then pL_rP_r antimessages are produced, on the average, in any advancement step. Considering antimessages equally likely to be addressed to any LP (as in [18]), the probability to be the recipient of an antimessage is $1/n_{LP}$. Therefore, for any LP, the probability to be the recipient of at least one antimessage can be expressed as:

$$\begin{aligned}
\beta &= \text{Prob(to receive at least one antimessage)} = \\
&= 1 - \text{Prob(to receive none of the } pL_rP_r \text{ antimessages)} = \\
&= 1 - [\text{Prob(not to receive an antimessage)}]^{pL_rP_r} = \\
&= 1 - \left(\frac{n_{LP}-1}{n_{LP}}\right)^{pL_rP_r} \qquad (10)
\end{aligned}$$

For the evaluation of M_p and A_p we consider, without loss of generality, that at any real time instant for any pair of LVTs $< LVT_1, LVT_2 >$, it holds:

$$\text{Prob}(LVT_1 < LVT_2) = \text{Prob}(LVT_1 > LVT_2) = \frac{1}{2} \qquad (11)$$

From the point of view of the recipient LP, the probabilities M_p and A_p do not differ. Given a message/antimessage whose timestamp is obtained enhancing the LVT of the sender LP, namely LVT_s, by the quantity ϵ (i.e., the timestamp increment), and denoting with LVT_r the LVT of the recipient LP, we can write:

$$\begin{aligned}
M_p &= A_p = \\
&= 1 - \text{Prob(message/antimessage is timestamped in the future of } LVT_r) = \\
&= 1 - [\text{Prob}(LVT_s > LVT_r) + \text{Prob}((LVT_r > LVT_s) \wedge (\epsilon > LVT_r - LVT_s))] = \\
&= 1 - [\text{Prob}(LVT_s > LVT_r) + \\
&\qquad + \text{Prob}(\epsilon > LVT_r - LVT_s | LVT_r > LVT_s)\text{Prob}(LVT_r > LVT_s)] \qquad (12)
\end{aligned}$$

Expression (12) can be rewritten as follows by applying (11) to the pair $< LVT_s, LVT_r >$:

$$M_p = A_p = 1 - [\frac{1}{2} + \text{Prob}(\epsilon > LVT_r - LVT_s | LVT_r > LVT_s)\frac{1}{2}] \qquad (13)$$

Renaming in (13):

$$\delta = \text{Prob}(\epsilon > LVT_r - LVT_s | LVT_r > LVT_s)\frac{1}{2} \qquad (14)$$

we finally get:

$$M_p = A_p = \frac{1}{2} - \delta \tag{15}$$

Note that, as (11) applies to any pair of LVTs, (15) also represents the probability that a message/antimessage is timestamped in the past of the LVT of any LP distinct from the sender; therefore (15) does not apply only to the recipient LP. This property will be exploited in Section 4.2.3 while defining the upper bound on P_r.

From the definition of δ in (14) we have $0 \leq \delta \leq 1/2$. For what concerns its limit values we get:

- $\delta = 1/2$ implies $\text{Prob}(\epsilon > LVT_r - LVT_s | LVT_r > LVT_s) = 1$; that is, every message/antimessage sent by an LP with LVT_s in the past of LVT_r is timestamped in the future of LVT_r;

- $\delta = 0$ implies $\text{Prob}(\epsilon > LVT_r - LVT_s | LVT_r > LVT_s) = 0$; that is, no message/antimessage sent by an LP with LVT_s in the past of LVT_r is timestamped in the future of LVT_r.

Backward substituting (15) and (10) in (9) and (15) in (8) inequality $Q_p + Q_s \leq 1$ straightforwardly holds, thus implying consistency of expression (7) for P_r. Finally, plugging (8) and (9) in (7), we obtain the following expression for the rollback probability P_r:

$$P_r = \frac{\alpha(\frac{1}{2} - \delta)}{1 - (\frac{1}{2} - \delta)[1 - (\frac{n_{LP}-1}{n_{LP}})^{pL_rP_r}]} \tag{16}$$

4.2.2 Analysis of the Rollback Length

Let N_{safe} denote the average number of events executed in each advancement step of the simulation and not undone by any future rollback occurrence. We refer to these events as *safe* events. The maximum value for N_{safe} is equal to the number of processors p. By using N_{safe} we can again formulate an expression for N_{par} as follows:

$$N_{par} = \frac{N_{seq}}{N_{safe}} p \tag{17}$$

Indeed, N_{seq}/N_{safe} is the number of advancement steps to terminate a simulation with N_{seq} forecast events; multiplying this ratio by the number of events executed in each step (that is the number p of running LPs), we obtain the total number of events N_{par} of the parallel execution. Substituting (4) in (17), through simple algebra, we get:

$$L_r = \frac{1}{P_r}(1 - \frac{N_{safe}}{p}) \tag{18}$$

The quantity N_{safe}/p is the probability for an event, executed on whichever processor in any advancement step, to be not undone in the progress of the simulation. Then $(1 - N_{safe}/p)$ is the rate of undone events. Multiplying latter quantity by the rollback interval $1/P_r$ we get the average rollback length L_r.

By (18) L_r monotonically decreases vs N_{safe}. Considering the limit values for N_{safe} we get:

- when $N_{safe} = 0$ the average rollback length assumes its limit value $L_r = 1/P_r$; in this case the simulation does not progress since every rollback occurrence moves the simulation back to the last rollback point (the number of events of the parallel execution is unbounded - see expression (4));

- when $N_{safe} = p$ the average rollback length assumes the value $L_r = 0$; in this case no event is ever undone by rollback.

4.2.3 Evaluation of the Upper Bound on the Rollback Probability

By replacing (18) in (16) we obtain for P_r the following expression:

$$P_r = \frac{\alpha(\frac{1}{2} - \delta)}{1 - (\frac{1}{2} - \delta)[1 - (\frac{n_{LP}-1}{n_{LP}})^{p-N_{safe}}]} \tag{19}$$

In this section we state a lower bound on N_{safe} leading to an upper bound on P_r.

Let us consider the p events executed in each advancement step as ordered by increasing values of their timestamps. We denote with $P_{safe}(i)$ the probability that the i-th event in the ordering is a safe event. Because of the Smallest-Timestamp-First scheduling, and due to the absence of messages/antimessages in transit at the beginning of any advancement step, the timestamp of the 1-st event in the ordering represents the GVT of the simulation. This event cannot be ever undone, therefore $P_{safe}(1) = 1$ (i.e., at least one safe event is executed in each advancement step). Then N_{safe} assumes the following expression:

$$N_{safe} = 1 + P_{safe}(2) + \ldots + P_{safe}(p) \tag{20}$$

The safety of the 2-nd event in the ordering depends only on the activities performed on the processor where the 1-st event in the ordering is executed. Two cases have to be considered:

(i) the execution of the 1-st event does not require rollback handling;

(ii) the execution of the 1-st event requires rollback handling.

In case (i) the 2-nd event is safe if the message sent at the end of the execution of the 1-st event

either

e1 : is addressed to a local LP, *and* the timestamp of the next to be processed message stored in the local event list is in the future with respect to the timestamp of the 2-nd event in the ordering

or

e2 : is addressed to a remote LP, *and* its timestamp is in the future with respect to the timestamp of the 2-nd event in the ordering, *and* the timestamp of the next to be processed message stored in the local event list is in the future with respect to the timestamp of the 2-nd event in the ordering.

Denoting with **E** the probabilistic event obtained by the union of the disjoint probabilistic events **e1** and **e2**, we get:

$$\text{Prob}(\mathbf{E}) = \text{Prob}(\mathbf{e1}) + \text{Prob}(\mathbf{e2}) \tag{21}$$

In order to evaluate Prob(**E**), we recall that the timestamp of the 2-nd event in the ordering represents the LVT of the LP executing it, as the LVT of an LP moves to the event timestamp upon the event execution. As expression for M_p in (15) holds for any pair $< X, Y >$ (where X represents the timestamp of a message and Y represents the LVT of an LP distinct from the sender of the message), we obtain the following expressions for Prob(**e1**) and Prob(**e2**) ([3]):

$$\begin{aligned}\text{Prob}(\mathbf{e1}) &= (1-\alpha)(1-M_p) && (22)\\ \text{Prob}(\mathbf{e2}) &= \alpha(1-M_p)(1-M_p) && (23)\end{aligned}$$

Plugging (15) in (22) and in (23) we obtain:

$$\begin{aligned}\text{Prob}(\mathbf{e1}) &= (1-\alpha)(\frac{1}{2}+\delta) && (24)\\ \text{Prob}(\mathbf{e2}) &= \alpha(\frac{1}{2}+\delta)^2 && (25)\end{aligned}$$

and by (21):

$$\text{Prob}(\mathbf{E}) = (1-\alpha)(\frac{1}{2}+\delta) + \alpha(\frac{1}{2}+\delta)^2 \tag{26}$$

Case (ii) cannot be easily modeled to express the safety of the 2-nd event and we do not consider it here. Thus we deduce the following lower bound on $P_{safe}(2)$, which neglects the contribution proportional to the rollback probability P_r:

$$P_{safe}(2) \geq (1-P_r)\text{Prob}(\mathbf{E}) \tag{27}$$

By substituting (26) in (27) the lower bound on $P_{safe}(2)$ becomes:

$$P_{safe}(2) \geq (1-P_r)[(1-\alpha)(\frac{1}{2}+\delta) + \alpha(\frac{1}{2}+\delta)^2] \tag{28}$$

[3]The expressions here derived for Prob(**e1**) and Prob(**e2**) do not consider the case in which the LP executing the second event in the ordering is the sender of the next to be processed message stored in the local event list. Such approximation does not heavily affect results of the analysis, as previous case is usually infrequent.

$$y_0 = \frac{\alpha(\frac{1}{2}-\delta)}{1-(\frac{1}{2}-\delta)[1-(\frac{n_{LP}-1}{n_{LP}})^{(p-\widehat{N}_{safe}(0,\delta))}]}$$

$$y_1 = \frac{\alpha(\frac{1}{2}-\delta)}{1-(\frac{1}{2}-\delta)[1-(\frac{n_{LP}-1}{n_{LP}})^{(p-1)}]}$$

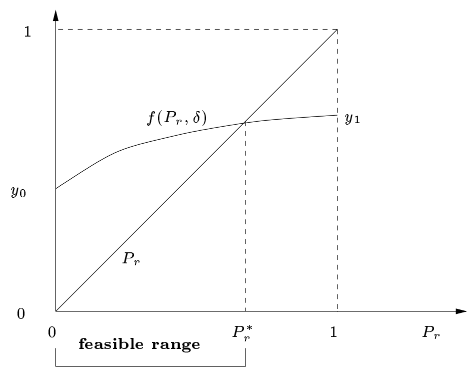

Figure 3: Feasible Range for the Rollback Probability P_r.

The considerations made about the safety of the 2-nd event in the ordering can be iterated. The safety of the i-th event depends on the activities performed on each of the $i-1$ processors where the events preceding the i-th one in the ordering are executed. The i-th event is certainly safe whenever on any processor running the j-th event in the ordering (with $j < i$): no rollback occurs *and* the probabilistic event $\mathbf{E}$, properly redefined substituting the j-th event to the 1-st event and the i-th event to the 2-nd event, occurs. Note that the probability of $\mathbf{E}$ does not change after such redefinition because, as outlined above, probability M_p in (15) does not depends on the timestamp and the LVT which are compared. Therefore the following lower bound on $P_{safe}(i)$ is obtained:

$$P_{safe}(i) \geq \{(1-P_r)\text{Prob}(\mathbf{E})\}^{i-1} = \{(1-P_r)[(1-\alpha)(\frac{1}{2}+\delta)+\alpha(\frac{1}{2}+\delta)^2]\}^{i-1} \tag{29}$$

By plugging (29) in (20) the following lower bound $\widehat{N}_{safe}$ on N_{safe} is obtained, where we use the notation $\widehat{N}_{safe}(P_r,\delta)$ to highlight the dependence of $\widehat{N}_{safe}$ on P_r and on δ:

$$N_{safe} \geq \widehat{N}_{safe}(P_r,\delta) = \sum_{i=1}^{p}\{(1-P_r)[(1-\alpha)(\frac{1}{2}+\delta)+\alpha(\frac{1}{2}+\delta)^2]\}^{i-1} \tag{30}$$

The right side of (19) decreases vs N_{safe}. Replacing in (19) N_{safe} with $\widehat{N}_{safe}(P_r,\delta)$, by (30) we get:

$$P_r \leq \frac{\alpha(\frac{1}{2}-\delta)}{1-(\frac{1}{2}-\delta)[1-(\frac{n_{LP}-1}{n_{LP}})^{p-\widehat{N}_{safe}(P_r,\delta)}]} \tag{31}$$

Inequality (31) determines the feasible range of values for P_r. Let $f(P_r,\delta)$ denote the right end of (31), that is:

$$f(P_r,\delta) = \frac{\alpha(\frac{1}{2}-\delta)}{1-(\frac{1}{2}-\delta)[1-(\frac{n_{LP}-1}{n_{LP}})^{p-\widehat{N}_{safe}(P_r,\delta)}]} \tag{32}$$

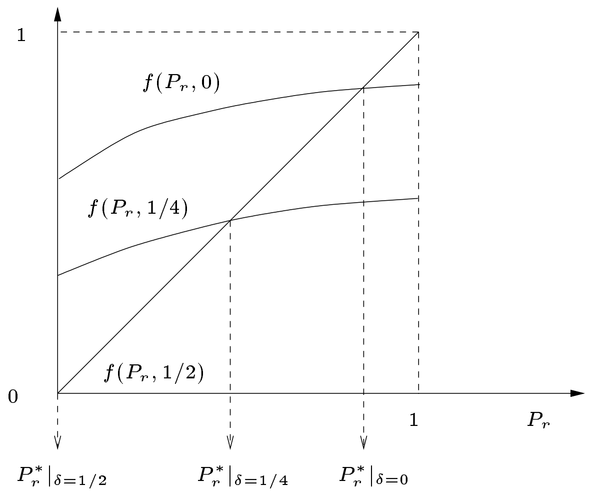

Figure 4: Behavior of P_r^* vs δ.

then, as shown in Figure 3, the feasible range for P_r is the interval where $P_r \leq f(P_r, \delta)$, that is:

$$0 \leq P_r \leq P_r^* \tag{33}$$

where P_r^* denotes the upper bound value for the rollback probability.

Let us make some considerations on the behavior of P_r^* vs δ. Three different curves for $f(P_r, \delta)$, corresponding to three different values for δ (0, 1/4 and 1/2), are plotted in Figure 4. While δ increases in its feasible range, that is [0,1/2], the value of P_r^* decreases. From the definitions of δ and P_r the decrease matches with the expected behavior. For the limit value $\delta = 1/2$, $f(P_r, \delta)$ is identically null, so $P_r^*|_{\delta=1/2} = 0$. This result straightforwardly derives from (15); indeed, when $\delta = 1/2$ the probability to receive a straggler message is equal to zero (i.e., no rollback is ever originated). In this case, our model determines an upper bound on the rollback probability equal to zero. For the limit value $\delta = 0$, the absolute upper bound $P_r^*|_{\delta=0}$ on the rollback probability of the simulation is obtained.

The knowledge of the value of δ allows to determine, combining expressions (30) and (31), the feasible range for the rollback probability (and therefore its upper bound value) in function of: number of LPs n_{LP}, number of processors p and probability of remote communication α.

4.2.4 Evaluation of the Upper Bound on the Product $P_r L_r$

From expression (18), the product $P_r L_r$ decreases vs N_{safe}. Replacing in (18) N_{safe} with $\widehat{N}_{safe}(P_r, \delta)$, by (30) we get for the product $P_r L_r$ the following inequality:

$$P_r L_r \leq 1 - \frac{\widehat{N}_{safe}(P_r, \delta)}{p} \tag{34}$$

Expression (30) also outlines that $\widehat{N}_{safe}(P_r, \delta)$ decreases vs P_r; hence, replacing in (34) P_r with P_r^*, by (33) we get:

$$P_r L_r \leq 1 - \frac{\widehat{N}_{safe}(P_r^*, \delta)}{p} = UB_{P_r L_r}(P_r^*, \delta) \tag{35}$$

where $UB_{P_r L_r}(P_r^*, \delta)$ denotes the upper bound on the product $P_r L_r$.

Let us now make some considerations on $UB_{P_r L_r}(P_r^*, \delta)$ for the limit value $\delta = 1/2$. From (30), $\widehat{N}_{safe}(P_r, \delta)$ is expressed in function of P_r^* as follows:

$$\widehat{N}_{safe}(P_r^*, \delta) = \sum_{i=1}^{p} \{(1 - P_r^*)[(1-\alpha)(\frac{1}{2} + \delta) + \alpha(\frac{1}{2} + \delta)^2]\}^{i-1} \tag{36}$$

where the generic term of the sum in the right end of (36) is the product of two quantities, both increasing vs δ. It is possible to show that:

$$lim_{\delta \to \frac{1}{2}} \widehat{N}_{safe}(P_r^*, \delta) = p \tag{37}$$

therefore from (35):

$$lim_{\delta \to \frac{1}{2}} UB_{P_r L_r}(P_r^*, \delta) = 0 \tag{38}$$

Expression (38) outlines an expected behavior since, for the limit value $\delta = 1/2$, the upper bound P_r^* is null (i.e., there is no rollback at all).

As outlined at the end of Section 4.2.3, the knowledge of the value for δ allows to determine P_r^*; then, by combining (36) and (35), $UB_{P_r L_r}(P_r^*, \delta)$ can be computed.

In conclusion, once stated a value for δ, the upper bounds on P_r and the product $P_r L_r$ can be computed in function of: number of LPs n_{LP}, number of processors p and probability of remote communication α.

4.2.5 Empirical Study on the Parameter δ

The empirical study on δ, here reported, has been carried out on the synthetic benchmark known as PHOLD model [13], which is widely used for testing performance of Time Warp simulators. We have chosen this benchmark also because it shows a rollback behavior similar to many synthetic benchmarks and to several real world models. In this benchmark, a constant number of messages circulate among LPs, messages are equally likely to be forwarded to any LP, and the timestamp increments derive from some stochastic distribution.

We have considered four different values for the number of LPs constituting the model (16, 32, 64, 128). For all these configurations, the exponential distribution has been selected for timestamp increments and a population of 10 messages per LP has been fixed[4]. Each processor runs the same

[4] A population of 10 messages per LP gives rise to simulations enough saturated to avoid idle times, which are not taken into account in our model (see assumption d)).

number of LPs. We simulated each benchmark using different values for the number of processors (between 2 and 8)[5].

As hardware architecture we used a cluster of machines (Pentium 233 MHz - 64 Mbytes RAM) connected via Ethernet. Inter-processor communication relies on message passing supported by PVM [34]. The adopted Time Warp system has all the features described in Section 4, except for limited amount of memory causing fossil collection; the Time Warp kernel reclaims memory on demand.

We study the relation between the values of δ and the average timestamp increments by using the parameter $\mathbf{d}$ (an integer variable) defined as follows. Let ϵ_{max} (resp. ϵ_{min}) denote the maximum (resp. minimum) average timestamp increment overall LPs. The value of $\mathbf{d}$ is determined by the following relation:

$$\frac{\epsilon_{max}}{\epsilon_{min}} = k \times 10^{\mathbf{d}} \tag{39}$$

with $1 \leq k < 10$. Therefore the parameter $\mathbf{d}$ expresses, in terms of order of magnitude, the distance between maximum and minimum average timestamp increments.

In Table 1 observed values for δ (computed as the average over 10 runs) are reported vs the ratio n_{LP}/p and vs $\mathbf{d}$ ([6]). From the obtained results it is evident that, for any pair $< n_{LP}/p, \mathbf{d} >$, the values of δ appear to be quite stable (the maximum distance is within 5%). Therefore we can deduce that, once fixed the average values of the distribution functions for the timestamp increments (whose distance is expressed by $\mathbf{d}$), the value of δ depends only on the degree of parallelism of the simulation (expressed by n_{LP}/p).

As outlined above, the PHOLD model shows a rollback behavior similar to many other models; therefore, the values of δ in Table 1 may be considered as representative for many simulations. This feature will be exploited by the algorithm we present in Section 5 for the choice between sequential and parallel simulation.

In Table 2 and in Table 3 the observed values for P_r and P_rL_r are reported vs n_{LP}/p and vs $\mathbf{d}$.

From Table 2 and Table 3, once fixed values for $\mathbf{d}$ and for n_{LP}, the rollback probability P_r and the product P_rL_r decrease vs the ratio n_{LP}/p. This behavior is due to the lower degree of parallelism that leads to lower probability of timestamp order violation.

As a support to the correctness of our analysis, the solution of the model with values of δ reported in Table 1 leads to values for P_r^* and $UB_{P_rL_r}$ which are actually upper bounds on observed values reported in Table 2 and in Table 3.

[5]Eight is the number of available machines.

[6]The value of δ has been measured as follows. A standard clock synchronization algorithm runs on the processors. For each LP a log of its LVT changes is maintained together with the real time at which each change occurs. Messages/antimessages are also stamped with the real time at which they are sent. Upon the receipt of a message/antimessage, in order to collect data to evaluate δ, the virtual send time of the message/antimessage is compared to the LVT of the recipient LP recorded at the real send time of that message/antimessage.

Table 1: Observed Values for δ.

n_{LP}/p	$d=0$				$d=1$				$d=2$			
	$n_{LP}=16$	32	64	128	$n_{LP}=16$	32	64	128	$n_{LP}=16$	32	64	128
2	0.397	-	-	-	0.330	-	-	-	0.248	-	-	-
4	0.436	0.433	-	-	0.387	0.377	-	-	0.273	0.264	-	-
8	0.457	0.454	0.452	-	0.417	0.412	0.411	-	0.304	0.297	0.292	-
16	-	0.467	0.466	0.461	-	0.436	0.429	0.434	-	0.327	0.320	0.319
32	-	-	0.475	0.473	-	-	0.456	0.449	-	-	0.347	0.333
64	-	-	-	0.477	-	-	-	0.457	-	-	-	0.356

Table 2: Observed Values for P_r.

n_{LP}/p	$d=0$				$d=1$				$d=2$			
	$n_{LP}=16$	32	64	128	$n_{LP}=16$	32	64	128	$n_{LP}=16$	32	64	128
2	0.086	-	-	-	0.150	-	-	-	0.230	-	-	-
4	0.040	0.049	-	-	0.071	0.094	-	-	0.170	0.211	-	-
8	0.014	0.023	0.030	-	0.026	0.044	0.052	-	0.076	0.136	0.182	-
16	-	0.009	0.014	0.020	-	0.015	0.28	0.030	-	0.050	0.090	0.129
32	-	-	0.005	0.009	-	-	0.009	0.015	-	-	0.045	0.119
64	-	-	-	0.004	-	-	-	0.007	-	-	-	0.049

Table 3: Observed Values for $P_r L_r$.

n_{LP}/p	$d=0$				$d=1$				$d=2$			
	$n_{LP}=16$	32	64	128	$n_{LP}=16$	32	64	128	$n_{LP}=16$	32	64	128
2	0.195	-	-	-	0.306	-	-	-	0.560	-	-	-
4	0.074	0.091	-	-	0.112	0.155	-	-	0.313	0.404	-	-
8	0.022	0.037	0.051	-	0.038	0.065	0.078	-	0.103	0.209	0.346	-
16	-	0.012	0.023	0.032	-	0.020	0.040	0.045	-	0.074	0.137	0.222
32	-	-	0.008	0.014	-	-	0.011	0.020	-	-	0.077	0.202
64	-	-	-	0.006	-	-	-	0.009	-	-	-	0.086

4.3 Parallel Execution Time

In this section we perform an analysis of the parallel execution time of the simulation and derive an upper bound on it based on the bounds on P_r and $P_r L_r$ previously obtained.

Let T_{ev} denote the average execution time for a scheduling cycle. As outlined at the beginning of Section 4.2, in our model T_{ev} assumes the same value on all processors. Under assumptions b), c) and d), all processors execute the same number of events (i.e., N_{par}/p). Therefore, the completion time T_{par} of the parallel simulation can be expressed as:

$$T_{par} = \frac{N_{par}}{p} T_{ev} \tag{40}$$

and by (4):

$$T_{par} = \frac{N_{seq}}{p} \frac{1}{1 - P_r L_r} T_{ev} \tag{41}$$

The quantity N_{seq}/p is the minimum number of events to be executed on each processor to complete a simulation with N_{seq} forecast events. Because of the rollback mechanism, the real number of events executed on each processor becomes $N_{seq}/[p(1 - P_r L_r)]$.

Denoting with T_i the mean time spent in the i-th block of the flow-graph in Figure 1, T_{ev} can be expressed as:

$$T_{ev} = T_1 + P_r(T_2 + T_3) + T_4 + T_5 + \frac{1}{\chi} T_6 + T_7 \tag{42}$$

Furthermore, denoting with:

- t_{in}, the mean time for receiving a message/antimessage (i.e., the time to unpack it);
- t_{out}, the mean time for sending a message/antimessage (i.e., the time to pack it);
- $t_{extract}$, the mean time to schedule the next to be run LP;
- t_r, the mean time required to reload a checkpoint into the current state location;
- t_{ev}, the mean event routine time;
- t_s, the mean state saving time;

the following expressions for the T_i terms of (42) can be derived:

$$T_1 = t_{extract} \tag{43}$$

$$T_2 = L_r t_{out} \tag{44}$$

$$T_3 = t_r + \frac{(\chi - 1)}{2} t_{ev} \tag{45}$$

$$T_4 = t_{ev} \tag{46}$$

$$
\begin{aligned}
T_5 &= t_{out} && (47)\\
T_6 &= t_s && (48)\\
T_7 &= (1 + P_r L_r) t_{in} && (49)
\end{aligned}
$$

The term T_3 models the mean time for state recovery in case of rollback (its expression has been introduced in [30]). T_3 consists of the time to reload a checkpoint into the current state location plus the coasting forward time; this latter quantity has been shown in [30] to be proportional to the average distance between checkpoints, that is $(\chi - 1)/2$. For what concerns the term T_7 we recall that, in any advancement step, each processor receives, on the average, one message and $P_r L_r$ antimessages. The derivation of all the other terms is quite straightforward. Substituting expressions (43)-(49) in (42) we get:

$$T_{ev} = t_{extract} + P_r(L_r t_{out} + t_r + \frac{\chi - 1}{2} t_{ev}) + t_{ev} + t_{out} + \frac{t_s}{\chi} + (1 + P_r L_r) t_{in} \quad (50)$$

4.3.1 An Upper Bound on the Parallel Execution Time

Plugging expression (50) for T_{ev} in (41), we get:

$$T_{par} = \frac{N_{seq}}{p} \frac{1}{1 - P_r L_r} [t_{extract} + P_r(L_r t_{out} + t_r + \frac{\chi - 1}{2} t_{ev}) + t_{ev} + t_{out} + \frac{t_s}{\chi} + (1 + P_r L_r) t_{in}] \quad (51)$$

Expression (51) shows that T_{par} increases vs P_r and $P_r L_r$. Substituting the upper bounds P_r^* and $UB_{P_r L_r}(P_r^*, \delta)$ in (51), we get:

$$T_{par} \leq T_{par}^* = \frac{N_{seq}}{p} \frac{1}{(1 - UB_{P_r L_r}(P_r^*, \delta))} T_{ev}(P_r^*, UB_{P_r L_r}(P_r^*, \delta)) \quad (52)$$

where $T_{ev}(P_r^*, UB_{P_r L_r}(P_r^*, \delta))$ represents expression (50) for T_{ev} evaluated in P_r^* and $UB_{P_r L_r}(P_r^*, \delta)$.

T_{par}^* represents the upper bound value for the parallel execution time. Once fixed a value for δ, the upper bounds P_r^* and $UB_{P_r L_r}(P_r^*, \delta)$ can be computed in function of: n_{LP}, p and α; therefore, also T_{par}^* is computable as a function of the same parameters and the checkpoint interval χ.

As P_r^* and $UB_{P_r L_r}(P_r^*, \delta)$ decrease vs δ, T_{par}^* decreases vs δ as well. Hence, the following relation holds:

$$T_{par}^* \leq T_{par}^*|_{\delta=0} \quad (53)$$

meaning that $T_{par}^*|_{\delta=0}$ can be correctly used as upper bound on the parallel execution time. The tightness of this bound depends on the distance between the real δ value and the limit value $\delta = 0$.

Values of δ reported in Table 1 could be used to get bounds on the parallel execution time tighter than $T_{par}^*|_{\delta=0}$. The more the simulation model is similar (in terms of rollback behavior) to the PHOLD model, the more those values of δ are used effectively. Such values may be used by the algorithm we present in the next section for the choice between sequential and parallel simulation.

```
BEGIN
    Step1: $R_{min} \leftarrow +\infty$;
    Step2: forall $p \in [2, n_{LP}]$ do
        Step2.1: select a value for $\delta$;
        Step2.2: select a mapping of the LPs onto the $p$ processors and compute the value for $\alpha$;
        Step2.3: find an integer value for $\chi$ (with $\chi \geq 1$) minimizing the ratio $R$;
        Step2.4: if the value of $R$ obtained in Step2.2 is less than $R_{min}$
                 then $R_{min} \leftarrow R$; $\hat{p} \leftarrow p$; $\hat{\chi} \leftarrow \chi$;
    Step3: if $R_{min} \geq 1$
           then use the sequential simulator
           else use the parallel simulator with $\hat{p}$ processors (and the corresponding selected mapping),
                and $\hat{\chi}$ as checkpoint interval;
END
```

Figure 5: Choice Algorithm.

5 Choice Between Sequential and Parallel Simulation

In this section we make use of the proposed analysis to introduce a methodology to choose between sequential and Time Warp simulation. To this purpose we introduce the following expression for the execution time T_{seq} of a sequential simulation with N_{seq} forecast events:

$$T_{seq} = N_{seq} T'_{ev} \tag{54}$$

where T'_{ev} denotes the average execution time per event of the sequential simulation. T'_{ev} depends on the average event routine time, on software features of the sequential simulator and on the adopted hardware platform.

As our analysis does not lead to an exact value for the parallel execution time T_{par}, but only to an upper bound on it, namely T^*_{par} (see (52)), the algorithm we propose for the choice between sequential and parallel simulation is based on the comparison between T_{seq} and T^*_{par}. Such a comparison could lead to an incorrect choice of the sequential approach in case the sequential execution time T_{seq} falls into the interval $(T_{par}, T^*_{par}]$.

The algorithm, reported in Figure 5, chooses either sequential or parallel simulation depending on the value of the ratio R between the upper bound on the parallel execution time, expressed in (52), and the execution time of the sequential simulation, expressed in (54), that is:

$$R(P^*_r, \delta) = \frac{T^*_{par}}{T_{seq}} = \frac{T_{ev}(P^*_r, UB_{P_r L_r}(P^*_r, \delta))}{p(1 - UB_{P_r L_r}(P^*_r, \delta))T'_{ev}} \tag{55}$$

Let us explain the steps of the algorithm.

The selection of the value for δ in Step2.1 can be supported by Table 1 derived form the study of the behavior of the PHOLD model (see Section 4.2.5). Otherwise, $\delta = 0$ has to be selected.

Finding a mapping which minimizes the value for the probability of remote communication α is an NP-complete problem [16]. Solutions by two different costs can be adopted for the selection in Step2.2: (i) mapping based on heuristics leading to low values for α (high cost), (ii) random assignment of LPs to processors (low cost).

The value of n_{LP} is known prior the algorithm execution. After Step2.1, the value of δ has been selected; after Step2.2, also the values of p and α have been fixed, therefore the only parameter to be selected to determine the value for R is the checkpoint interval χ. Then, in Step2.3, the value of χ minimizing R is computed. Such value is the distance between checkpoints that optimizes the parallel execution time, supposing the simulation shows the rollback behavior predicted by our model.

Non-commented steps of the algorithm are trivial.

As last remark, if the parallel approach is selected, then any run time optimization scheme of the value of the checkpoint interval can be adopted to further improve performance (for example the one in [30]).

6 Examples of Application of the Choice Algorithm

In this section examples of application of the choice algorithm described in Section 5 are presented. We report results for both a synthetic benchmark (the PHOLD model) and a real world model. The latter is a store-and-forward communication network in a two-dimensional torus configuration (it will be extensively described in Section 6.2).

Simulations have been carried out by using the cluster of machines and the Time Warp kernel described in Section 4.2.5. In order to apply the choice algorithm to simulation platforms having around the same overhead for an operation (e.g., extraction from the event list of the next to be executed event, etc.), a sequential simulator having as much as possible code portions equal to the parallel one has been used.

The choice algorithm has been applied by exploiting values of δ reported in Table 1, considering for any pair $< n_{LP}/p, \mathbf{d} >$ the average value among the reported ones.

6.1 Results with Synthetic Benchmark

The synthetic benchmark considered is a PHOLD model with $n_{LP} = 120$ LPs. LPs have exponential distribution function for the timestamp increments with mean equal to 1 (therefore for this model we have $\mathbf{d} = 0$). Furthermore, the population is equal to 10 messages per LP. The selected simulation length is $N_{seq} = 10^6$ forecast events.

For this benchmark we consider two configurations which differ by the average event routine time; this parameter is modified by introducing a delay loop into the original routine. Instead, the state saving time is fixed at 100 μsec, corresponding to the time for saving a fictitious state of

Table 4: Results for Large State Granularity.

$T_{seq} = 63$	Predicted Time	Observed Time
$< \hat{p}, \hat{\chi} >$	(T^*_{par})	(T_{par})
$< 2, 18 >$	245	242
$< 3, 15 >$	175	162
$< 4, 14 >$	140	123
$< 6, 12 >$	116	83
$< 8, 11 >$	100	67

1.5 Kbytes. The first configuration represents the case of large state granularity of the LPs, as it exhibits a short event routine time (i.e., $t_{ev} = 50$ μsec) compared to the state saving time. The second configuration represents the case of medium/small state granularity, exhibiting a long event routine time (i.e., $t_{ev} = 400$ μsec) compared to the state saving cost.

For both configurations Step2 of the choice algorithm has been executed restricting the domain for p to the set of values $\{2, 3, 4, 6, 8\}$ (this is done in order to let each processor run the same number of LPs). Whenever the value of δ corresponding to a given value of n_{LP}/p is not available in Table 1, it is picked up from the row associated to the closest value of that ratio.

Large State Granularity. We got the following results by applying the choice algorithm:

Step2 $\longrightarrow R_{min} = 1.59$; $\hat{p} = 8$; $\hat{\chi} = 11$;

Step3 $\longrightarrow$ the sequential simulator is chosen.

The data obtained by simulating the benchmark in a sequential and in a parallel way are reported in Table 4 (each observed value is the average on 10 runs). The table reports the execution time of the sequential simulator T_{seq}; furthermore, for any pair $< p, \chi >$ (where the value of χ is the one evaluated in Step2.3 of the algorithm), both the observed value (T_{par}) and the predicted value (T^*_{par}) of the parallel execution time are reported. All execution time values are expressed in seconds. The data show that actually the sequential simulation is time-convenient over the parallel one; therefore, in this case the algorithm leads to an effective choice.

Medium/Small State Granularity. We got the following results by applying the choice algorithm:

Step2 $\longrightarrow R_{min} = 0.40$; $\hat{p} = 8$; $\hat{\chi} = 4$;

Step3 $\longrightarrow$ the parallel simulator is chosen with 8 processors and 4 as checkpoint interval.

The data obtained by simulating the benchmark in a sequential and in a parallel way are reported in Table 5. The data show that actually the parallel simulation is time-convenient over the sequential one. The algorithm makes the right choice; furthermore, it suggests to use a number

Table 5: Results for Medium/Small State Granularity.

$T_{seq} = 411$ $< \hat{p}, \hat{\chi} >$	Predicted Time (T^*_{par})	Observed Time (T_{par})
$< 2, 6 >$	434	429
$< 3, 5 >$	307	288
$< 4, 5 >$	243	217
$< 6, 4 >$	194	148
$< 8, 4 >$	164	113

of processors which actually leads to the minimum value for the parallel execution time among the observed ones.

6.2 Results with a Real World Model

In this section we present simulation results of a communication network in a two-dimensional torus configuration. Messages are transmitted according to the store-and-forward policy and the x-y-routing algorithm [33]. We consider a fixed message population (10 messages for each node); hence, when a message reaches the destination, another message is inserted into the network. Message lengths are selected from an uniform distribution (between 100 bytes and 3 Kbytes) and the message transmission time is proportional to the message length (2×10^{-3} unit times per byte). We assume there are two unidirectional links between neighboring nodes (one link for each direction). As only one message can be transmitted at a time, there may be queuing delays at links (the queuing discipline is FCFS). Hence, a queuing server is associated with each link; the service time is equal to the message transmission time. Service times are precomputed and messages are immediately forwarded to the next node. The network dimension is 4×4. Each node is modeled by an LP and the selected simulation length is $N_{seq} = 10^6$ committed events.

The size of the state of an LP is 1 Kbyte (the corresponding state saving time is 67 μsec), whereas the event routine time is 300 μsec. In this case Step2 of the choice algorithm has been executed restricting the domain for p to the set of values $\{2, 4, 8\}$ and adopting the obvious mapping of LPs onto processors.

We got the following results by applying the choice algorithm:

Step2 $\longrightarrow R_{min} = 0.79$; $\hat{p} = 4$; $\hat{\chi} = 4$;

Step3 $\longrightarrow$ the parallel simulator is chosen with 4 processors and 4 as checkpoint interval.

The obtained results are reported in Table 6 (also in this case, observed values result as the average of 10 runs). The data show that actually the parallel simulation is time-convenient over the sequential one. Also in this case, the algorithm makes the right choice, but it suggests to use a number of processors which actually does not lead to the minimum observed value for the parallel execution time.

Table 6: Results for the Real World Model.

$T_{seq} = 326$	Predicted Time	Observed Time
$< \hat{p}, \hat{\chi} >$	(T^*_{par})	(T_{par})
$< 2, 6 >$	386	381
$< 4, 4 >$	245	226
$< 8, 2 >$	260	138

This "non-optimal" choice of the number of processors arises from the significant overprediction of the model when the number of processors is large. When the number of processors increases, the degree of parallelism becomes higher and the rollback probability increases as well. From a mathematical point of view, our upper bound on the parallel execution time derives from a lower bound $\widehat{N}_{safe}$ on the quantity N_{safe} (see (30)). As $\widehat{N}_{safe}$ is obtained neglecting contributions proportional to the rollback probability (see (29)), the amount of our prediction error is large especially in case of high rollback probability. Therefore, the algorithm should lead to effective results in all the cases of low rollback probability (e.g, simulations of large models mapped onto a small number of processors or simulations with infrequently communicating LPs).

7 Conclusions

In this paper an analytical model describing performance of Time Warp synchronization, applied to simulation models with homogeneous LPs, has been presented. The model takes into account the effects of checkpointing and rollback overhead and the effects of the aggregation of LPs onto processors.

The model predicts an upper bound on the completion time of simulations of models with features matching those considered in our analysis. An algorithm to choose between Time Warp and sequential simulation, based on the model, is presented. The algorithm does not require preventive generation of simulation code.

Experimental results of a synthetic benchmark and a real word model have shown the effectiveness of the algorithm, which suggests the right choice between parallel and sequential implementation.

In case of parallel implementation, the algorithm might not identify the optimal value for the number of processors to be used. Such a drawback derives from a large overprediction of the model in the case of simulations showing high rollback probability (e.g., simulations with high degree of parallelism and/or high communication rate between logical processes). This is because the amount of productive simulation work, identified by the model as the fraction of events that are not undone by rollback, does not include a contribution proportional to the rollback probability. The analysis could be improved by modeling that contribution in order to define tighter upper bounds on the completion time. An orthogonal step ahead consists of a specialization of the analysis to particular classes of simulation models (e.g., queuing networks).

On the other hand, the model itself is highly general, as the major assumption on the considered simulation problem is the homogeneity of the LPs. Therefore it can be applied to a wide class of simulation problems.

Acknowledgment

The authors would like to thank the reviewers for helpful comments which have improved the quality and the presentation of the paper.

References

[1] R.Ayani and H.Rajae,"Parallel Simulation Using Conservative Time Windows", *Proc. 1992 Winter Simulation Conference*, pp.709-717, 1992.

[2] H. Bauer and C. Sporrer,"Reducing Rollback Overhead in Time Warp Based Distributed Simulation with Optimized Incremental State Saving", *Proc. 26-th Annual Simulation Symposium*, pp.12-20, 1993.

[3] K.Chandy and J.Misra,"A Case Study in the Design and Verification of Distributed Programs", *IEEE Trans. on Software Engineering*, vol.SE5, no.5, pp.440-452, 1979.

[4] K.Chandy and R.Sherman,"Conditional Event Approach to Distributed Simulation", *Proc. 1989 SCS Multiconference on Distributed Simulation*, pp.93-99, 1989.

[5] V.Cortellessa,"Performance Optimization of Time Warp Based Concurrent Simulators", *Ph.D. dissertation*, 1995.

[6] P.M.Dickens, D.Nicol, P.F.Reynolds and J.M.Duva,"Analysis of Bounded Time Warp and a Comparison with YAWNS", *ACM Trans. on Modeling and Computer Simulation*, vol.6, no.4, pp. 297-320, 1996.

[7] S.Eick, A.Greenberg, B.Lubachevsky and A.Weiss,"Synchronous Relaxation for Parallel Simulations with Applications to Circuit-Switched Networks", *ACM Trans. on Modeling and Computer Simulation*, vol.3, no.4, pp. 287-314, 1993.

[8] R.E.Felderman and L.Kleinrock,"Two processor Time Warp analysis: some result on an unifying approach", *Proc. 1991 SCS Workshop on Parallel and Distributed Simulation*, pp. 3-10, 1991.

[9] R.E. Felderman and L. Kleinrock,"Bounds and Approximations for Self-Initiating Distributed Simulation without Lookahead," *ACM Trans. on Modeling and Computer Simulation*, vol.1, no.4, pp. 386-406, 1991.

[10] A. Ferscha,"Probabilistic Adaptive Direct Optimism Control in Time Warp", *Proc. 9-th Workshop on Parallel and Distributed Simulation*, pp.120-129, 1995.

[11] A. Ferscha and J. Luthi,"Estimating Rollback Overhead for Optimism Control in Time Warp", *Proc. 28-th Annual Simulation Symposium*, pp.2-12, 1995.

[12] R.M.Fujimoto,"Parallel Discrete Event Simulation", *Communications of ACM*, vol.33, no.10, pp.30-53, 1990.

[13] R.M. Fujimoto,"Performance of Time Warp Under Synthetic Workloads," *Proc. of 1990 SCS Multiconference on Distributed Simulation*, Vol.22, No.1, pp.23-28, 1990.

[14] R.M.Fujimoto, J.Tsai and G.C.Gopalakrishnan,"Design and Evaluation of the Rollback Chip: Special Purpose Hardware for Time Warp", *IEEE Trans. on Computers*, vol.41, no.1, pp.68-82, 1992.

[15] A.Gafni,"Space Management and Cancellation Mechanisms for Time Warp", Tech. Rep. TR-85-341, University of Southern California, Los Angeles (Ca,USA), 1985.

[16] M.R.Garey and D.S.Johnson,"Computers and Intractability: a Guide to the Theory of NP-Completeness", New York: W.H.Freeman & C.,1979.

[17] D.W.Glazer and C.Tropper,"On Process Migration and Load Balancing in Time Warp", *IEEE Trans. on Parallel and Distributed Systems*, vol. 4, no. 3, pp.318-327, 1993.

[18] A.Gupta, I.F.Akyildiz and R.M.Fujimoto,"Performance analysis of Time Warp with Multiple Homogeneous Processors", *IEEE Trans. on Software Engineering*, vol.17, No.10, pp.1013-1027, 1991.

[19] D.Jefferson and H.Sowizral,"Fast concurrent simulation using the Time Warp mechanism; part I: local control", Tech. Rep. N1906AF, RAND Corporation, 1982.

[20] D.Jefferson,"Virtual Time", *ACM Trans. on Programming Languages and Systems*, vol.7 no.3, pp. 404-425, 1985.

[21] S.S.Lavenberg, R.Muntz and B.Samadi,"Performance analysis of a rollback method for distributed simulation", *Proc. Performance 83*, pp.117-132, 1983.

[22] J.I.Leivent and R.J.Watro,"Mathematical Foundations for Time Warp Systems", *ACM Trans. on Programming Languages and Systems*, vol.15, no.5, pp.771-794, 1993.

[23] Y.B.Lin and E.Lazowska,"Optimality Considerations for Time Warp Parallel Simulation," *Proc. 1990 SCS Multiconference on Distributed Simulation*, pp.35-48, 1990.

[24] Y.B.Lin, B.R.Preiss, W.M.Loucks and E.D.Lazowska,"Selecting the Checkpoint Interval in Time Warp Simulation", *Proc. 7-th Workshop on Parallel and Distributed Simulation*, pp.3-10, 1993.

[25] B.Lubachevsky,"Bounded Lag Distributed Discrete Event Simulation," *Proc. 1988 SCS Multiconference on Distributed Simulation*, pp.183-191, 1988.

[26] D.Mitra and I.Mitrani,"Analysis and Optimum Performance of Two Message-Passing Parallel Processors Synchronized by Rollback", *Performance Evaluation Journal*, vol. 7, pp.111-124, 1987.

[27] D.M. Nicol,"Performance Bounds on Parallel Self-Initiating Discrete-Event Simulation", *ACM Trans. on Modeling and Computer Simulation*, vol.1, no.1, pp. 111-124, 1991.

[28] A.C.Palaniswamy and P.A.Wilsey,"An Analytical Comparison of Periodic Checkpointing and Incremental State Saving", *Proc. 7-th Workshop on Parallel and Distributed Systems*, pp.127-134, 1993.

[29] B.D.Plateau and S.K.Tripathi,"Performance Analysis of Synchronization for two Communicating Processes", *Performance Evaluation Journal*, vol.8, pp.305-320, 1988.

[30] R.Ronngren and R.Ayani,"Adaptive Checkpointing in Time Warp", *Proc. 8-th Workshop on Parallel and Distributed Simulation*, pp.110-117, 1994.

[31] L.Sokol, D.Briscoe, A.Wieland and B.Lubachevsky," MTW: a Strategy for Scheduling Discrete Simulation Events for Concurrent Executions", *Proc. 1988 SCS Multiconference on Distributed Simulation*, pp.169-173, 1988.

[32] J.Steinman,"Speeds: Synchronous Parallel Environment for Emulation and Discrete Event Simulation", *Proc. 1991 SCS Western Multiconference on Advances in Parallel and Distributed Simulation*, pp.95-103, 1991.

[33] H.Sullivan and T.R.Bashkow,"A Large Scale, Homogeneous, Fully Distributed Parallel Machine", *Proc. 4-th International Symposium on Computer Architecture*, 1977.

[34] V.S.Sunderam,"A Framework for Parallel Distributed Computing", *Concurrency: Practice and Experience*, vol.2, no.4, 1990.

[35] S.Turner and M.Xu," Performance Evaluation of the Bounded Time Warp Algorithm," *Proc. 6-th Workshop on Parallel and Distributed Simulation*, pp.117-126, 1992.

[36] B.W. Unger, J.G. Cleary, A. Covington and D. West,"External State Management System for Optimistic Parallel Simulation", *Proc. 1993 Winter Simulation Conference*, pp.750-755, 1993.

[37] Y.C.Wong, S.Y.Hwang and Y.B.Lin,"A parallelism Analyzer of Conservative Parallel Simulation", *IEEE Trans. on Parallel and Distributed Systems*, vol.6, no.6, pp.628-638, 1995.

Authors Biographies

Francesco Quaglia received the laurea in electronic engineering in 1995 and the PhD degree in computer engineering in 1999 from the University of Rome "La Sapienza". Currently he is a member of the research staff of the Dipartimento di Informatica e Sistemistica of the same University. His research interests include parallel/distributed simulation, fault-tolerant distributed systems and interconnection networks.

Vittorio Cortellessa received the laurea in computer science from the University of Salerno (Italy) in 1991 and the PhD degree in computer engineering from the University of Rome at Torvergata (Italy) in 1995. His PhD dissertation focused on Time Warp modeling. During 1994, he was visiting research associate in the Concurrent Engineering Research Center of the West Virginia University, WV, working on the "Independent Verification and Validation of Software Systems" NASA project. Currently, he holds a post-doc fellowship in the Department of Computer Science Systems and Production of the University of Rome at Torvergata. His research interests include parallel discrete event simulation, software engineering, performance evaluation of software/hardware systems. He is an active member of the Italian Society for Computer Simulation.

Bruno Ciciani is full professor in computer engineering at the University of Rome "La Sapienza". His research interests include fault tolerance, computer architecture, distributed operating system, manufacturing yield prediction, performance and dependability evaluation of distributed and parallel computing systems. In these fields he has published more than 80 papers. He has published also some books; two of them have been published by Mc Graw Hill and by IEEE Computer Society Press. Prof. Ciciani spent more than one year as visiting researcher at the IBM Thomas J. Watson Research Center (N.Y.). He is a member of the IEEE Computer Society.