Rollbacks in Time Warp – Analysis and Modelling

K. A. Iskra, G. D. van Albada, P. M. A. Sloot

Section Computational Science, Faculty of Science, Universiteit van Amsterdam
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
{kamil, dick, sloot}@science.uva.nl

Keywords: parallel discrete event simulation, Time Warp, rollbacks, state saving, analytical modelling

Abstract

This paper presents a study of the interactions between the random number generator used and the run-time behaviour of the parallel Time Warp simulation kernel APSIS. A different rollback length distribution, with a far larger chance of long rollbacks taking place, is observed when the state of the random number generator is not preserved across the rollbacks. An explanation for this phenomenon is provided. An analytical model of the rollback behaviour in Time Warp is developed, for rollback length expressed in either the simulation time or the number of events to be rolled back. The hope is that, once the model is complete, it will be possible to determine under what circumstances it is profitable (not) to preserve the state of the random number generator.

1 Introduction

Parallel Discrete Event Simulation (PDES [1, 2]) is a technique used to speed up discrete event simulations by running them on parallel or distributed hardware platforms. A simulation is divided into *logical processes* which are then mapped onto the parallel hardware. The logical processes schedule events by sending messages to each other. The *causality constraint* requires that the processes to execute events in time-stamp order. *Time Warp* [3] is a prominent PDES algorithm. It lets logical processes run optimistically, ignoring the causality constraint to achieve better parallelism. When a logical process receives an event that should have been executed in the simulation past, a causality error occurs. To resolve the error, the process needs to roll back in the simulation time, undoing any over-optimistically performed operations, which includes cancelling messages sent to other logical processes. To be able to perform a rollback, a logical process needs access to its own past state, which is saved during the forward progress of the simulation. The state consists of a user-level state (simulation variables) and a simulation kernel-level state (message queues and other internal variables, among them the state of the random number generator).

In [4], we have described our experiments with running an Ising spin model under our own Time Warp kernel APSIS [5]. To reiterate, we have observed a very particular rollback length distribution resembling self-organised criticality for some simulation parameters, in particular the Ising spin model temperature. Since then, we have determined that more factors contribute to that sort of rollback length distribution; in particular the properties of the random number generator appear to be important.

This paper attempts to answer the questions of why adjusting the random number generator significantly affects the rollback distribution, and also whether this knowledge can be used to our advantage when performing simulations.

Analytical modelling has already been studied elsewhere. In [6], a model of Time Warp is studied using a Markov chain approach. Their method is however very hard to extend beyond two processes. Another Markov chain model, this one incorporating multiple processes, has been described in [7]. It takes an interesting approach by analysing the interaction of one process with Global Virtual Time instead of the more common analysis of interaction between processes. Important limitations that make it unsuitable in a case like ours are that it assumes a fixed population of messages and that an event can be scheduled at all other processes with equal probability. More recently, [8] contains a description of a probabilistic model of Time Warp which seems to be the most complete to be found in the literature, as it includes multiple processes, various overheads (often ignored in other models) and even rollback cascades. However, the underlying probabilistic model does not match that of the simulation that we want to study. Nevertheless, it does seem to prove that a probabilistic approach is quite promising for the more complex

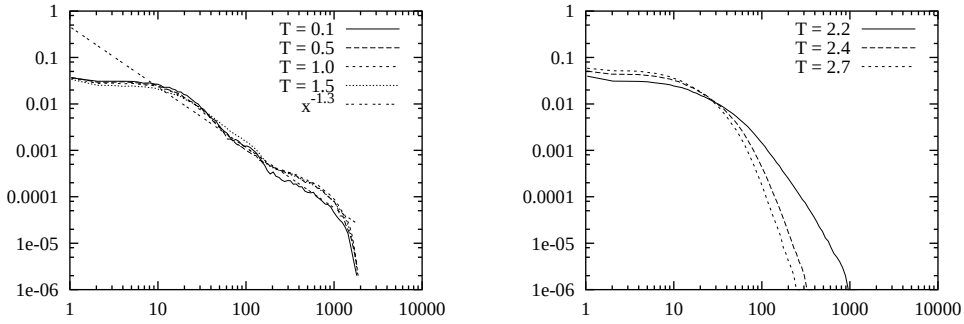

Figure 1: Rollback length distributions from Ising spin experiments, power function-like curve for low temperatures (left) and exponentially decaying for high temperatures (right). *X* axis is the rollback length, *Y* the probability.

models. To our knowledge, none of the existing models pays much attention to the properties of the random number generator used, which is the issue that we intend to focus on.

In [9], an idea of implementing temporal uncertainty in the Time Warp kernel is being studied, and promising results are obtained. External events are not scheduled for a fixed simulation time, but for a small time range, in a hope that some of the rollbacks can be prevented this way while the accuracy of the results stays unaffected. Their idea is analogous to ours, in that they are relaxing the Time Warp protocol hoping to improve on the speedup, only they relax the event execution time, and we the preserving and restoring the state of the random number generator.

The remainder of this paper is organised as follows. Section 2 briefly describes our experiments and provides an explanation for the results obtained. Section 3 provides a (so far incomplete) analytical model for the rollback behaviour observed. Section 4 presents a summary and outlines future directions.

2 Experiments with Ising Spin

As already mentioned, in [4] we have described our experiments with the Ising spin model. We have observed that for low temperatures of the model, the rollback length distribution adheres to the power law, as can be seen in Fig. 1.

Further experiments demonstrated that the power-law behaviour for low temperatures disappears if we save and restore the state of the random number generator across rollbacks. Before, the Ising spin simulation has been using a generator from the C library directly, so when re-running after a rollback, the generator would produce new random numbers instead of repeating the rolled back sequence, simply because the state of this generator wasn't being preserved. This wasn't considered to be a problem due to an inherently stochastic nature of the Ising spin model. However, when state-saving random number generator was introduced into the kernel, it turned out that using it significantly affects the run-time behaviour of the kernel: also for the low temperatures we get the exponentially decaying rollback length distribution as in Fig. 1 (right). We managed to re-create the distribution resembling the power function from Fig. 1 (left) only after increasing the *Virtual Time Window*. The Virtual Time Window is a parameter of the Time Warp kernel that limits excessive optimism by imposing a maximum distance that a logical process can move ahead from the *Global Virtual Time* (which itself is periodically calculated across all the logical processes).

2.1 Explanation of the Results

We are now going to present our interpretation of the phenomenon described above. Figure 2 shows

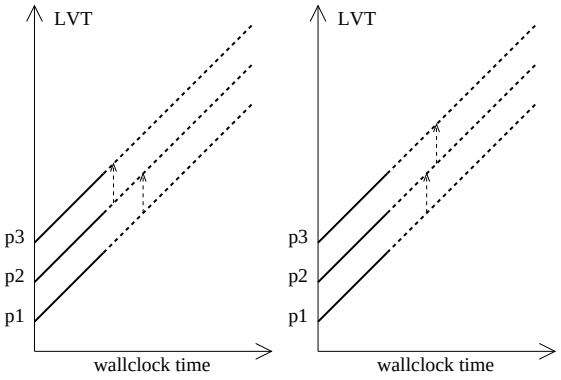

Figure 2: Two possible rollback scenarios: middle first (left) and bottom first (right).

two alternative initial scenarios to be considered. We have the wall-clock time on the *X* axis, and the *Local Virtual Time* (LVT), i.e. the progress of the simulation, on the *Y* axis. We are analysing 3 processes *p1–p3* of a running simulation. The LVT of each of them is different, but they generally progress at the same pace. The solid lines denote already executed operations, whereas the dashed parts present specula-

tion of how the simulation could progress. Sooner or later, the processes will communicate with each other. We are analysing cases in which processes with lower LVT schedule external events at the processes with higher LVT. Because in the Ising spin model simulation used, the schedule time of external events always equals the sender's LVT, scheduling in these cases is guaranteed to result in a rollback. We are going to analyse these rollbacks in greater detail. Figure 3 expands on the left case of Fig. 2 by presenting the behaviour of the APSIS kernel with a plain and a state-saving random number generator. Let's focus

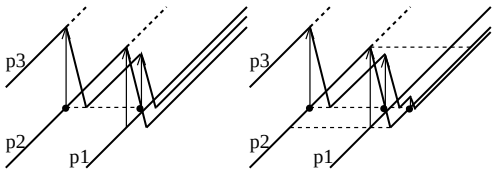

Figure 3: Rollback behaviour for the middle first scenario with a plain (left) and a state-saving (right) random number generator.

on the case with the plain random number generator (left) first. The middle process *p2* schedules an external event at the top process *p3* (the narrow arrow pointing straight up). In order to process the event, which is in its simulation time *past*, process *p3* must roll back to that point in time (steep line going down and to the right – rolling back does take some time, although much less than moving forward). Once it has rolled back, *p3* can take the externally scheduled event into account and restart the forward execution. Soon afterwards the bottom process *p1* schedules an external event at the middle process *p2*, which causes the latter to roll back. While rolling back, process *p2* notices that it has over-optimistically scheduled an external event at *p3*, and it sends an *anti-message* to *p3*. The anti-message causes another rollback at *p3* before it can be taken into account. Once this is over, all 3 processes can move forward in unison again. Using a state-saving random number generator (right) can add an extra twist at the end. Because the random number generator rolls back with the simulation, when process *p2* restarts the forward execution it gets the same stream of random numbers and consequently schedules an external event on *p3* again, which could cause one more small rollback on *p3*. So, in total, we observe two middle-sized rollbacks, one smaller one and, in the case of state-saving random number generator, one very small one. Let's now focus on Fig. 4, which expands on the right case of Fig. 2. In this case, the differences between the left and right side of the figure are more significant. Let's again focus on the case with a plain random number generator (left) first. Process *p1* schedules an external event at process *p2*, which causes the latter to roll back. We know that if

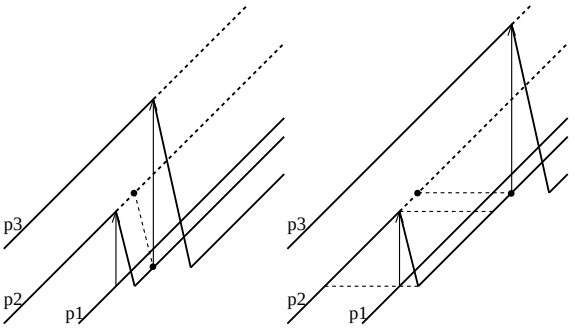

Figure 4: Rollback behaviour for the bottom first scenario with a plain (left) and a state-saving (right) random number generator.

process *p2* hadn't rolled back, it would have scheduled an external event at process *p3* soon afterwards. And, indeed, that's what happens soon after the rollback is over: *p2* schedules an external event at *p3*, which triggers a very large rollback. Moving on to the case with a state-saving random number generator (right), the scheduling of an external event by *p2* on *p3* does not happen so soon. First *p2* must go again through the rolled back sequence of random numbers, and only then is the event at *p3* scheduled. In both cases discussed above, we have a middle-sized rollback followed by a very large rollback. So, statistically, there shouldn't be any difference. However, until now, we failed to take into account one important factor, namely virtual time window. On Fig. 4 (right), process *p3* must be able to significantly advance forward in order for a large rollback to occur. Figure 5 shows the same situation as before, but with a virtual time window. As we can see, for a particular

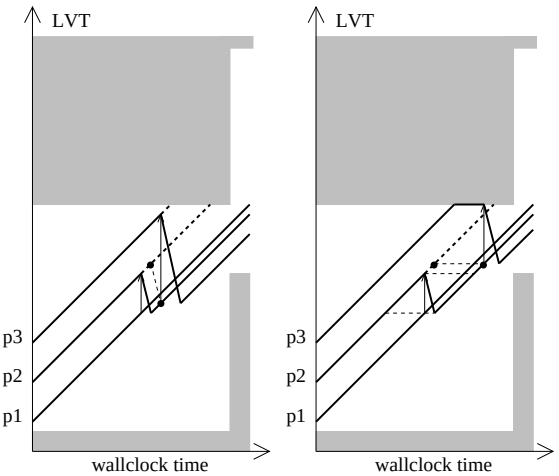

Figure 5: Rollback behaviour for the bottom first scenario with virtual time window and with a plain (left) and a state-saving (right) random number generator.

range of values, it is possible that the virtual time win-

dow does not influence the case with the plain random number generator, but might just limit the excessive optimism in the case of the state-saving random number generator. This in turn results in a significantly reduced maximum rollback length. Our assumption is that this phenomenon is responsible for the disappearance of the power-law rollback distribution with state-saving random number generator. Increasing the virtual time window size should result in the increase of maximum rollback length and hence in the reappearance of power function-like rollback length distribution. As already mentioned, this has been confirmed by experiments.

3 Rollback Modelling

Now that the interaction between the rollback behaviour and the random number generator has been explained, an analytical model that would allow to compare the pros and cons of state-saving random number generator is in order. Preserving the state of the generator can be rather expensive for simulations with small state, or those that do incremental state saving, like our Ising spin model simulation. Inherently stochastic simulations such as Ising spin should not need a state-saving random number generator to produce valid results, so using a non-state saving one could be profitable performance-wise.

The simulation time in the Ising spin model is increased by random intervals from a negative exponential distribution with the average of $\lambda = 1$. The probability density function for the exponential distribution is as follows:

$$f(t) = \lambda e^{-\lambda t} \tag{1}$$

In our case, (1) can be interpreted as a probability density that a process has increased its simulation time in a single step by t. From that, we can calculate:

$$f_n(t) = \frac{t^{n-1}}{(n-1)!} e^{-t} \tag{2}$$

Equation (2) is the probability distribution of time advance t after n random intervals from a negative exponential distribution with the average of $\lambda = 1$ (it is the Erlang distribution). We can now attempt to calculate the probability distribution of the distance in simulation time between two processes after n steps:

$$h_n(d) = \int_0^\infty f_n(t)\, f_n(t+d)dt \tag{3}$$

What (3) gives us is a function that specifies the probability that, after n steps, the two processes have diverged from each other by time difference d. Therefore, if the lagging process scheduled an event for its current simulation time on the other process, the rollback would have the length of d (in time units) with the probability given by (3). Equation (3) is only defined for positive values of d, but that's not a problem since the distribution is known to be symmetrical anyway. Further calculations lead to:

$$\begin{aligned} h_n(d) &= \frac{e^{-|d|}}{[(n-1)!]^2} \int_0^\infty [t(t+|d|)]^{n-1} e^{-2t} dt \\ &= \frac{e^{-|d|}}{(n-1)!} \sum_{i=0}^{n-1} \frac{(n-1+i)!\,|d|^{n-1-i}}{(n-1-i)!\,i!\,2^{n+i}} \end{aligned} \tag{4}$$

Therefore, an exact, analytical solution does exist. It is not always practical however, particularly for large (in the range of thousands) values of n, for which the factorials are not representable in typical floating-point computer arithmetic. Another approach would therefore be preferable. We need to begin by defining $h_n(d)$ differently:

$$h_{n+1}(d') = \int_{-\infty}^{+\infty} h_n(d) h_\Delta(d'-d)dd$$

$$h_\Delta \equiv h_1$$

$$h_n \equiv \underbrace{h_1 \otimes \ldots \otimes h_1}_{n}$$

h_n is thus an n-convolution of h_1, so it can easily be expressed in the Fourier domain. From (4) we also know the analytical form of $h_1(d)$:

$$h_1(d) = \frac{1}{2} e^{-|d|} \qquad \Longleftrightarrow \qquad H_1(s) = \frac{1}{1+(2\pi s)^2}$$

$$H_n(s) = H_1(s)^n = \frac{1}{[1+(2\pi s)^2]^n}$$

$$h_n(d) = \int_{-\infty}^{+\infty} \frac{1}{[1+(2\pi s)^2]^n} e^{2\pi i d s} ds \tag{5}$$

Equation (5), can easily be solved numerically using Fast Fourier Transform. Figure 6 presents some results. Next to the numerical results from FFT, the results from a rollback simulator are presented for the number of positive events ranging between 2 and 5000.

If we make an extra assumption that the rollbacks take no time to perform, then after the rollback the two processes become fully synchronised, so (5) can be applied again, taking the time of the rollback as the new start time. This makes it possible to derive the total rollback length distribution across the whole ranges of simulation time and rollback length:

$$l(d) = \sum_{n=1}^{\infty} \left[h_n(d)(1-P_{\text{rb}})^{n-1} P_{\text{rb}} \right] \tag{6}$$

P_{rb} is the probability of a rollback taking place, which is assumed to be constant for every step; the probabilities in different steps are assumed to be independent.

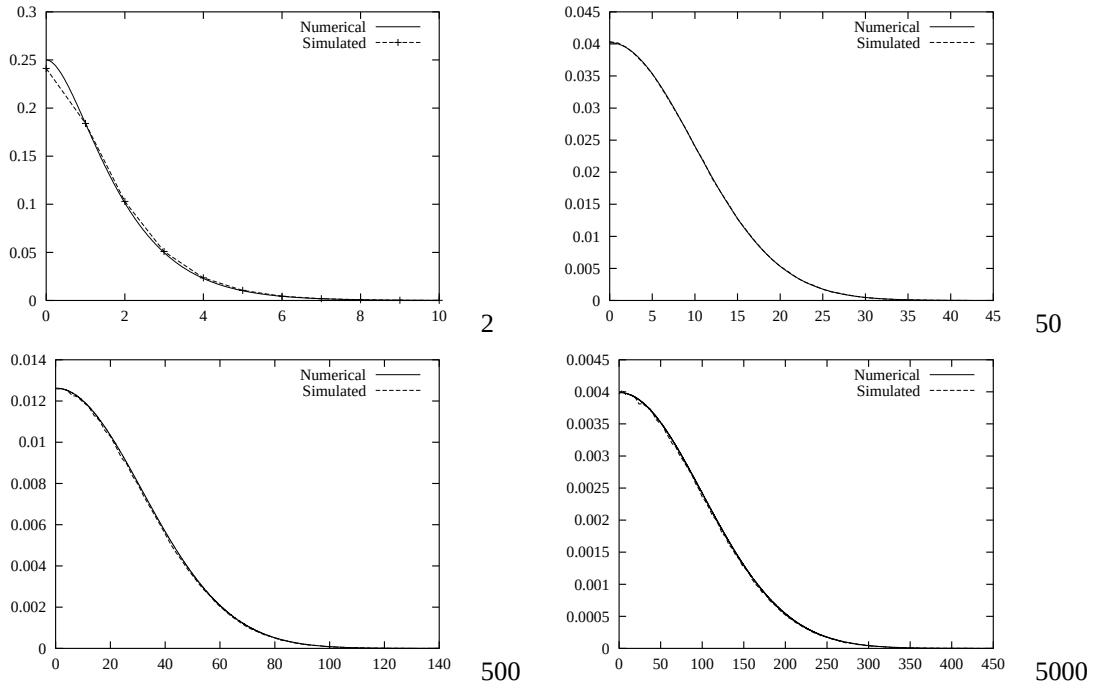

Figure 6: Numerical and simulation results for rollback length expressed in simulation time after different numbers of forward events. *X* axis denotes rollback length, *Y* axis the probability of such a rollback taking place.

3.1 Rollback length expressed in events

In the equations presented up till now, the rollback length was expressed in the units of simulation time. It is however more useful to express it in the number of events that need to be rolled back.

Let's consider two processes, *A* and *B*, each of which has executed *n* positive simulation events. Executing each positive event increases simulation time by a value drawn from a random number generator with a negative exponential distribution. What is the probability that, should a rollback occur at that point, it will have length *n*, i.e. the maximum possible? This can only occur if one of the processes, say *A*, made less progress in all *n* events added together than process *B* in just the first event. In this case, when process *A* schedules an external event on process *B* after *n* events, process *B* will need to undo all *n* events to handle the externally scheduled event. Figure 7 (left) presents this situation.

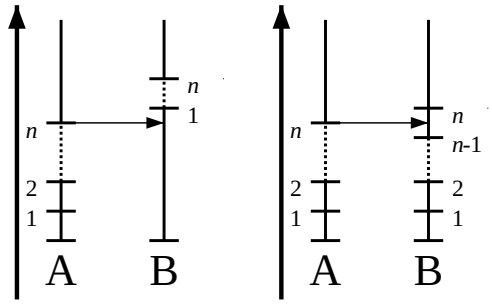

Figure 7: The situations needed to get particular rollback length: maximum rollback length *n* (left) and minimum length 1 (right).

And what is the probability that the rollback will have length 1, i.e. the minimum possible? This can only occur if the sum of all *n* time advances in one processes, say *A* again, is larger than the sum of $n-1$ time advances in *B*, but is smaller than the sum of all *n* advances in *B*. This is presented in Fig. 7 (right).

These probabilities can be described analytically as follows:

$$h_n(k) = 2 \cdot P\left(\sum_{i=1}^{n-k} b_i + c > \sum_{i=1}^{n} a_i > \sum_{i=1}^{n-k} b_i \right) \qquad (7)$$

Equation (7) specifies the probability that if a rollback occurs after two processes made *n* positive steps forward, then the rollback will have a length of *k* events, $k = 1, \ldots, n$. Basically, it is a probability that the sum of all *n* advances of process *A*, denoted as a_i, is larger than $n-k$ advances of process *B*, denoted as b_i, but is smaller than $n-k+1$ advances, where the last advance is denoted as *c*. This probability must be multiplied by 2 in order to take into account the symmetrical situation of process *B* lagging behind process *A*.

For negative exponential distributions of a_i and b_i, the sums in (7) follow the Erlang distribution, and the

equation can be rewritten as:

$$\begin{aligned} h_n(k) &= 2 \cdot \int_{c=0}^{\infty} e^{-c} \int_{\beta=0}^{\infty} \frac{\beta^{n-k-1}}{(n-k-1)!} e^{-\beta} \cdot \\ &\quad \cdot \int_{\alpha=\beta}^{\beta+c} \frac{\alpha^{n-1}}{(n-1)!} e^{-\alpha} d\alpha\, d\beta\, dc \qquad (8) \\ h_n(n) &= 2 \cdot \int_{c=0}^{\infty} e^{-c} \int_{\alpha=0}^{c} \frac{\alpha^{n-1}}{(n-1)!} e^{-\alpha} d\alpha\, dc \end{aligned}$$

The top version of (8) is for $k = 1, \ldots, n-1$, the bottom for $k = n$. Both versions are solvable analytically, the results are presented below:

$$\begin{aligned} h_n(k) &= \frac{\sum_{i=1}^{n} 2^i \left(\frac{(2n-k-1-i)!}{(n-i)!} - \sum_{j=0}^{n-i-1} \frac{(n-k-1+j)!}{j!} \right)}{(n-k-1)!\, 2^{2n-k}} \\ h_n(n) &= \frac{1}{2^{n-1}} \end{aligned}$$

Table 1 presents several sample results for small values of the parameters. Not surprisingly, we get a

	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$
$n=1$	1				
$n=2$	$\frac{1}{2}$	$\frac{1}{2}$			
$n=3$	$\frac{3}{8}$	$\frac{3}{8}$	$\frac{1}{4}$		
$n=4$	$\frac{5}{16}$	$\frac{5}{16}$	$\frac{1}{4}$	$\frac{1}{8}$	
$n=5$	$\frac{35}{128}$	$\frac{35}{128}$	$\frac{15}{64}$	$\frac{5}{32}$	$\frac{1}{16}$

Table 1: $h_n(k)$ for small values of n and k. Please note how $h_n(n)$ changes and also that $h_n(1) = h_n(2)$.

probability of 1 for a rollback length of 1 after 1 step forward. That's because time advances are floating point values drawn from a random number generator, so the probability of both processes drawing the same value (which would result in no rollback) is theoretically 0 (or somewhat more in practice due to finite precision of floating point numbers in computers). The probabilities of maximum rollbacks ($h_n(n)$) decrease geometrically by a factor of 2. The probabilities of shorter rollbacks do not follow such a simple pattern. Interestingly enough however, the probabilities of rollbacks of length 1 and 2 turn out to be the same for all values of $n > 1$, which is hardly intuitive. The above probabilities have been validated against those obtained from the rollback simulator for 1 000 000 simulated rollbacks, and a very good match has been observed.

While (8) denotes a probability and hence has values in the range of $[0,1]$, calculating these values for n and k in the range of thousands involves factorials which are not representable even in 80-bit `long double` arithmetic. Maximum rollback lengths observed were around 2000 (see Fig. 1), while the arithmetic used currently limits us to around 1500. Not only the range, but also the precision of the arithmetic is a problem, since we need to compute sums of elements differing by many orders of magnitude, which in the end mostly cancel each other out. We signalled this problem earlier (see the discussion of (4) above), and then we could overcome it by switching to Fourier domain. This doesn't appear to be an option here. Possible solutions include switching to an even higher precision software floating-point library or not using the final analytical solution, but trying to solve the simpler intermediate equations with the help of numerical integration.

Even with the $h_n(k)$ currently limited to 1500, we can attempt to calculate the rollback length distribution, for the length expressed in the number of events to roll back:

$$l(k) = \sum_{n=1}^{\infty} \left[h_n(k) (1 - P_{\text{rb}})^{n-1} P_{\text{rb}} \right]$$

This is analogous to (6).

4 Conclusions and Future Directions

We have presented an analytical model of the rollback behaviour in Time Warp, for rollback length expressed either in simulation time or in the number of events to roll back. The methods of calculating the rollback length distribution based on the model, using numerical approximation, have also been presented.

Nevertheless, important elements are still missing. The model as presented is only valid for non-state saving random number generators, where after the rollback the system returns to a fully synchronised state and where we have no knowledge of the events to be scheduled in future (as we would have with state-saving random number generator). The "full synchronisation" requires only two logical processes in the system, since the other logical processes, which did not take part in the rollback, would not synchronise.

These are important restrictions and we are currently attempting to address them.

References

[1] R.M. Fujimoto, Parallel Discrete Event Simulation, *Communications of the ACM*, vol. 33, no. 10, pp. 30–53, October 1990.

[2] R.M. Fujimoto, *Parallel and Distributed Simulation Systems*, Wiley, 2000.

[3] D.R. Jefferson, Virtual Time, *ACM Transactions on Programming Languages and Systems*, vol. 7, no. 3, pp. 404–425, July 1985.

[4] B.J. Overeinder, A. Schoneveld, and P.M.A. Sloot, Spatio-Temporal Correlations and Rollback Distributions in Optimistic Simulations, In

Proc. 7th Annual Conference of the Advanced School for Computing and Imaging, Heijen, The Netherlands, pp. 391–399, May 2001.

[5] B.J. Overeinder, *Distributed Event-driven Simulation – Scheduling Strategies and Resource Management*, Ph.D. thesis, Universiteit van Amsterdam, The Netherlands, November 2000.

[6] R.E. Felderman and L. Kleinrock, Two Processor Time Warp Analysis: Capturing the Effects of Message Queueing and Rollback/State Saving Costs, *Int. Journal of Electronics and Communication*, vol. 47, no. 5-6, pp. 353–367, September-November 1993.

[7] A. Gupta, I.F. Akyildiz, and R.M. Fujimoto, Performance Analysis of Time Warp with Multiple Homogeneous Processors, *IEEE Transactions on Software Engineering*, vol. 17, no. 10, pp. 1013–1027, October 1991.

[8] S.C. Tay, Y.M. Teo, and R. Ayani, Performance Analysis of Time Warp Simulation with Cascading Rollbacks, In *Proc. 12th Workshop on Parallel and Distributed Simulation*, Banff, Alberta, Canada, pp. 30–37, May 1998.

[9] R. Beraldi and L. Nigro, A Time Warp Mechanism Based on Temporal Uncertainty, *Transcactions of The Society for Modeling and Simulation International*, vol. 18, no. 2, pp. 60–72, June 2001.