DISCRETE-EVENT SIMULATION AND THE EVENT HORIZON

Jeff S. Steinman

Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive, Mail Stop 525-3960
Pasadena, CA 91109
(818) 306-6139, jss@pebbles.jpl.nasa.gov

KEY WORDS

Event horizon, SPEEDES, parallel discrete event simulation, Breathing Time Buckets, Time Warp.

ABSTRACT

The event horizon is a very important concept that is useful for both parallel and sequential discrete-event simulations. By exploiting the event horizon, parallel simulations can process events in a manner that is risk-free (i.e., no antimessages) in adaptable "breathing" time cycles with variable time widths. Additionally, exploiting the event horizon can greatly reduce the event list management overhead that is common to virtually all discrete-event simulations.

This paper develops an analytic model describing the event horizon from first principles using equilibrium considerations and the hold model (where each event, when consumed, generates a single new event with future-time statistics described by a known probability function). Exponential and Beta-density functions are used to verify the mathematics presented in this paper.

INTRODUCTION

The event horizon is a term taken from physics and astronomy relating to boundaries of black holes. It is defined by physicist Stephen Hawking as "The boundary of the region of space-time from which it is not possible for anything to escape" (Hawking 88). It is this space-time boundary where a black hole, due to gravity, collapses on itself. In a similar manner, the space-time characteristics (Chandy 89) of a parallel discrete-event simulation can be described through a boundary called the event horizon (Lubachevsky 88).

Assuming that processed events in a discrete-event simulation do not immediately release their event-generating messages (like a black hole not allowing anything to escape), a space-time boundary can be defined. This boundary, known as the global event horizon, is the point where the next event to be processed is an unsent message (Steinman 92).

The event horizon discussion begins by first describing event processing on a single processor. Then, it is generalized for parallel systems. Included is a brief discussion of the SPEEDES Queue (a simple event list data structure that exploits the event horizon). An analytic treatment of the event horizon is given, followed by three appendices that prove some very important properties of the analytic event horizon model.

DISCRETE EVENT SIMULATION

Discrete-event simulation involves two fundamental concepts. First, the concept of *simulation objects*. Simulation objects maintain a set of local variables that describe their state. Additionally (in terms of object-oriented programming), simulation objects may provide some basic methods that characterize the object. Second, there is the concept of *events*. Events "act" on simulation objects to possibly change their state and/or to schedule future events for potentially any other object in the simulation. Causality forbids events to post other events in their past. By processing events in chronological order, causality is always preserved.

A set of time-tagged pending events must be maintained by the simulation machinery. A simple way of maintaining this set is to keep events sorted by time using a linked list. The earliest time-tagged event is popped out of the list and processed. If that event generates new events, then these are inserted back into the list preserving the sorted nature of the list. Inserting new events can be very inefficient for linked lists if the size of the list is large.

Various alternative data structures have been proposed to lower the overhead for event list management (Sleator 85, Jones 86, Ronngren 91). However, there is a simple trick that one can use to make the simple linked-list approach more efficient. This trick will help introduce the concept of the event horizon, that the SPEEDES Queue and other more sophisticated data structures (e.g., trees and heaps) in the *Synchronous Parallel Environment for Emulation and Discrete-Event Simulation* (SPEEDES) exploit. Furthermore, it will provide a mechanism for processing events independently in parallel (Steinman 92).

THE SPEEDES QUEUE

Figure 1 shows a sequence of three event-processing cycles that exploit the concept of the event horizon. In each cycle, the new events that were generated have been collected in an unsorted secondary list (this is the trick). The minimum time-tagged event in this list is separately identified. When it is

time to process that event, the secondary list is sorted and then merged into the primary list. The time value of this boundary is called the event horizon. Event processing then continues for the next cycle. This simple two-list data structure is called the SPEEDES Queue (Steinman 92).

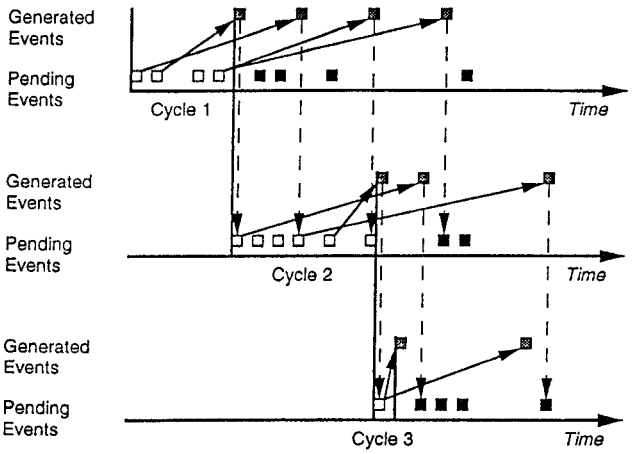

Figure 1: An example of processing events in cycles defined by the event horizon.

A simple timing comparison can be made at this point for the linked list and SPEEDES Queue data structures. Assume that each event generates a single new event and that there are n events always pending in the list (i.e., the hold model). Then for a singly linked list, popping the earliest event out of the list is accomplished in constant time while insertion of the newly generated event requires going through a portion of the list. This fractional portion of the list is called the bias B. An expression that describes the overhead (per event) for linked-list event management is

$$T_{LL} = a + b(Bn) \tag{1}$$

where a and b are coefficients that describe the overhead for popping and insertion using linked lists. Later, this paper will show how to calculate the bias B.

Now consider the SPEEDES Queue data structure. Inserting events into the list is performed in constant time (i.e., the processing time is independent of the length of the list). This is because the events are simply added to the bottom of the secondary unsorted list. However, removing the next event with the earliest time-tag from the two-list data structure is much more involved. This is because the next event may not be in the primary list, but instead, in the secondary unsorted list. If it is in the secondary list, then the secondary list needs to be sorted and then merged into the main list. However, if the next event is in the primary list, it can be removed in constant time by simply popping it out of the list.

Define m as the average number of events accumulated in the secondary list before it needs to be sorted and merged into the primary list (later, this paper will show how to calculate m). Merge sorting the list of m events can be performed in $m\log_2(m)$ time, and the time to merge two lists is proportional to the sum of the sizes of the lists. Thus, the time per event can be written as,

$$T_{SQ} = a + b\log_2(m) + c\frac{n}{m} \tag{2}$$

where a, b, and c are now the coefficients that describe the overhead for popping, sorting, and merging events in the SPEEDES Queue data structure.

One can easily determine the optimal value for m (i.e., the number of events collected in the temporary list) using the SPEEDES Queue data structure to achieve its highest efficiency by taking the derivative of the expression for T_{SQ} with respect to m and setting it to zero. The optimal value for m is,

$$m = \frac{c}{b} n \log_e(2) \tag{3}$$

Sorting a list of events generally requires more overhead (per event) than merging two lists of sorted events (per event) so one would expect the ratio of c/b to be less than one. Thus, the optimal number of events processed per cycle m will be somewhat smaller than n. The exact amount will depend on the efficiency of the implementation of the sort and merge algorithms.

One concern is that if m is too small, the SPEEDES Queue data structure will degenerate to linear performance (i.e., its computational complexity becomes proportional to the size of the list). This data structure, then, is not always appropriate for event-list management. On the other hand, the SPEEDES Queue data structure performs exceptionally well for simulations where m is very large (Steinman 92). It also performs well for simulations where n is not too large (e.g., $n < 500$). Later in this paper, an analytic model will be developed to predict m for simulations that are in equilibrium where an event, when consumed, generates a single new event.

Alternative data structures that have worst case $\log_2(n)$ performance have been developed in SPEEDES that also use the event horizon concept (i.e., the SPEEDES Tree and the SPEEDES Queue-Heap) but they will not be discussed here (final results and a publication are expected soon). However, it should be mentioned that the SPEEDES Queue-Heap has typically been measured to be about 2.5 times faster than the Splay Tree for event list management.

IMPLICATIONS FOR PARALLEL SIMULATION

As previously stated, the event horizon is the time interval between the first event of the current cycle in a simulation and the last possible event that can be safely processed independently from all new events generated in that cycle. The event horizon can fluctuate from cycle to cycle as events are processed. This fluctuation is caused by the statistical nature of event-generation time tags.

The causality principle insures that all events with time tags bounded by the event horizon can be processed safely in parallel.

Therefore, a good measure of the *inherent parallelism* of a simulation is the average number of events m that can be processed in an event horizon. If this number is much larger than the number of available computing nodes N, then high parallel performance should normally be achieved.

Parallelism in Discrete-Event Simulations

One might ask, "What fundamental precepts allow simulations to achieve parallel performance?" There are essentially four concepts that have been reported in the literature that provide parallelism in both conservative and optimistic discrete-event simulations (Reynolds 88).

1. *Lookahead:* the time difference in a processed event and an event that it generates. Simulations with large amounts of lookahead usually perform better than those with poor lookahead (Felderman 92). Lookahead has often been characterized by a minimum time value (Fujimoto 88) or sometimes by an average time value (Lin 90). The event horizon gives a better handle on understanding the true amount of parallelism in a simulation because it states how many events (on the average) can be processed independently.

   The event horizon often predicts a large amount of parallelism for simulations that have a minimum lookahead value of zero. Furthermore, it predicts different amounts of parallelism for simulations having different event generation statistics even though they might have the same average lookahead value.

2. *Large Numbers:* as simulations get larger (in terms of numbers of objects and events) more parallelism generally exists. For example, rollbacks in optimistic simulations only occur for the objects that are affected, not for all of the objects on a node. Because of this, it is possible for a message with a time tag smaller than the current local time of its destination node to not cause rollbacks. As another example, conservative simulations that exploit locality block less often when there are many simulated objects on each node.

   In terms of scalability, it is not generally true that doubling the size of a simulation, and then running it on twice the number of nodes, results in the same processing time. Time Warp, for example, will usually experience more rollbacks in this case. The calculations in the next section on the event horizon explain why.

3. *Locality:* knowing that objects only interact locally with a small number of neighboring objects. Locality can allow objects in conservative FIFO simulations to safely processes events when their input queues are not all empty (lookahead is often used in this context also) (Chandy 79, Fujimoto 90). Locality information is also used to decompose a simulation so that highly interacting objects are placed on the same node.

4. *Lazy Cancellation:* events for the same object can be processed out of strict time order if they are independent of each other. This fact can allow simulations to sometimes "beat" the critical path (Wieland 91). Lazy cancellation is not fundamental to this discussion of the event horizon; so it will not be mentioned further.

The Event Horizon and Risk-Free Parallel Simulations

The concept of the event horizon can be exploited through synchronization paradigms such as Breathing Time Buckets, Breathing Time Warp, and Local CoOp (Conservative-Optimistic simulation) (Steinman 92, Steinman 93). All of these strategies use the concept of the event horizon to cut down the amount of risk in sending messages that might have to be canceled due to rollbacks.

It has been observed that instabilities in Time Warp (Jefferson 85) (a strategy with maximum risk) are usually caused by excessive numbers of antimessages, not rollbacks (Lubachevsky 89, Turner 92, Steinman 93). By exploiting the event horizon, completely general parallel discrete-event simulations can be supported (without requiring assumptions on lookahead or locality), while at the same time cutting back or removing the amount of risk involved.

Breathing Time Buckets relies heavily on lookahead and large numbers in order to achieve its performance. Like Time Warp, Breathing Time Warp relies less on lookahead (although lookahead certainly helps performance) and more on large numbers of objects. Local CoOp allows objects with full input queues to safely process their events without state-saving overhead, while using the event horizon to globally synchronize the simulation when deadlocks otherwise would have occurred. Instead of blocking when input queues are empty, events in CoOp are processed optimistically, but risk-free (Dickens 90, Mehl 91).

One further interesting property of risk-free strategies is that the number of rollbacks can sometimes be reduced. Simulations can even be constructed where certain objects never experience rollbacks. This phenomenon, for example, has been observed in solving the parallel proximity detection problem for military simulations. Objects can be guaranteed to never have rollbacks if their pending events are always consumed in each event horizon cycle, or if certain lookahead conditions exist.

Although locality and lazy cancellation can be used in simulations exploiting the event horizon, fundamental to the event horizon are the principles of lookahead and scalability. Characterizing lookahead for simulations with different sizes is not always easy.

The concepts of lookahead and scalability are more generally understood through a mathematical treatment of the event horizon.

ANALYTIC MODEL FOR THE EVENT HORIZON

For this analysis, assume that each event generates a single new event as it is processed (i.e., the hold model) (Chou 93). The time-tag of the generated event is described by a random distribution function $f(t)$. With this model, the total number of events n in the simulation is constant. The goal is to determine the average number of events m processed in a cycle.

Similar analysis has been done by David Nicol (Nicol 91) using statistical methods. A more intuitive way to compute m is to recognize the equilibrium time invariants (i.e., quantities whose time derivative is zero) of the simulation. With this approach, differential equations can be formed and solved. If it turns out that m is much larger than the number of available nodes N, the simulation should be able to achieve high parallel performance. Some definitions must be made at this point:

Definitions

n = Total number of events at the start of a cycle.
m = Average number of events processed in a cycle.
t = Simulation time. $t = 0$ is the start of the cycle.
$f(t)$ = Random distribution for event generation.

$F(t)$ = Cumulative probability distribution of $f(t)$.
$G(t)$ = $1 - F(t)$.
$P(t)$ = Probability of the next cycle boundary exceeding t.
$\rho(t)$ = Density of pending events. $\rho_0 = \rho(0)$.
$\delta t(t)$ = Average time interval between events = $1/\rho(t)$.
B = Event-insertion bias ($0 < B < 1$).
T_{EH} = Average event horizon time interval.

The Event Density

Assume that the total number of events n and the random distribution function $f(t)$ for event generation are known. Furthermore, assume (without loss of generality) that the first scheduled event occurs at time $t = 0$. The first step is to obtain an expression for the equilibrium event density $\rho(t)$ for the simulation. Figure 2 shows conceptually (using an exponential event-generation distribution function) how, under equilibrium conditions, the shape of the event density remains unchanged as events are processed.

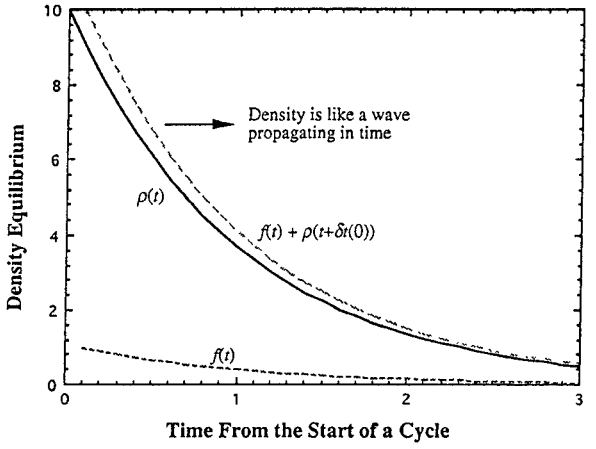

Figure 2: Equilibrium for the density of events. Assume that $f(t) = e^{-t}$ and that $n = 10$. When the first event is processed, a new one is added back in time according to $f(t)$. The event density is restored back to its original shape but it is shifted in time by $\delta t(0)$. The density function can be viewed as a wave propagating in time.

Assume that the first event is processed at time 0. The second event is scheduled to occur at time $\delta t(0)$. Adding the new event (generated by the first event) back into the event-density distribution should restore the density function back to its original shape (except shifted in time by $\delta t(0)$). In other words, the shape of the event-density distribution relative to the starting time of the earliest event in the simulation should be time-invariant. The event density can then be viewed as a wave propagating in time. This leads to the following equilibrium equation:

$$\rho(t) = \rho(t + \delta t(0)) + f(t), \text{ where } \delta t(0) = \frac{1}{\rho_0} \tag{4}$$

Using $\rho(t + \delta t(0)) \approx \rho(t) + \delta t(0)\rho'(t) = \rho(t) + \rho'(t)/\rho_0$, a simple differential equation can be constructed.

$$0 = \frac{\rho'(t)}{\rho_0} + f(t) \tag{5}$$

The solution to this equation is

$$\rho(t) = \rho_0 G(t) \tag{6}$$

Note that the total number of events can be determined by integrating the event density.

$$n = \int_0^\infty \rho(t)dt \tag{7}$$

This gives a way to solve for ρ_0 if n is known.

$$\rho_0 = \frac{n}{\int_0^\infty G(t)dt} \tag{8}$$

One important note about the equilibrium event density is that it is a decreasing function in time. In other words, the list of pending events has fewer events scheduled to occur in the far future (relative to the time-tag of the first event in the list) as compared to the near future. This can be easily understood because the derivative of the density, $\rho'(t) = -f(t)$, must be less than or equal to zero since $f(t)$ is a non-negative function. Because the event density decreases in time, the spacing between events $\delta t(t) = 1/\rho(t)$ always increases as a function of time (relative to the time-tag of the first pending event).

The Event-Insertion Bias

The bias B is defined as a number between 0 and 1 that describes the fractional amount of the list that events on the average must traverse for insertion. For example, if $B = 1/2$, then, on average, inserting an event into a linked list data structure would require traversing halfway through the list. B is easily computed from the density function as

$$B = \frac{1}{n}\int_0^\infty f(t)dt \int_0^t \rho(\tau)d\tau \tag{9}$$

After rearranging the integration order, and substituting for $\rho(t)$, this reduces to the following form:

$$B = \frac{\int_0^\infty G(t)^2 dt}{\int_0^\infty G(t)dt} \tag{10}$$

For example, a flat function $f(t) = 1$ produces a bias of 2/3, an exponential distribution $f(t) = e^{-t}$ produces a bias of 1/2, and a FIFO type of distribution $f(t) = \delta(t)$ produces a bias of 1.

Computing m for the Event Horizon

Now that the event density has been found, the next step is to determine the probability $P(t)$ that an event with time stamp t is processed in the current cycle.

Consider the first event. It is always safe to process the first event. Thus $P(t_0) = 1$. The next event occurs (on the average) at time $t_1 = t_0 + \delta t(t_0)$. The probability of this event being processed in the current cycle is just the probability that the first event scheduled its new event with a time-tag greater than t_1. Since $F(t)$ is the probability of the event being generated with a time-tag less than or equal to t (relative to the generating

event's time tag), $G(t) = 1 - F(t)$ is the probability that the event has a time-tag greater than t. This means that $P(t_1) = G(t_1 - t_0)$. In general, an event at time, t, will be in the current cycle if all the previous events in the cycle generate their events with time tags greater than or equal to t.

The probability for the $n'th$ event being processed in the current cycle can be constructed in a general way. Defining $t_{n+1} = t_n + \delta t(t_n)$ as the time-tag of the $(n+1)'th$ event from the start of the cycle (note that the $0'th$ event is actually the first event of the cycle), the probability of this event being in the current cycle is the product of all the probabilities of the previous events generating their events with time tags greater than t_{n+1}. This is described mathematically as

$$P(t_{n+1}) = G(t_{n+1} - t_0)G(t_{n+1} - t_1)\ldots G(t_{n+1} - t_n) \qquad (11)$$

A very good approximation to this expression is to simply relate the probability of the next event being in the current cycle to the previous event. The error introduced by making this approximation is at most second order in $F(t)$. Because $F(t)$ must be small for the important terms ($0 << P(t) < 1$), this error is negligible. In the worst case, neglecting this error can cause the calculation for m to be off by at most 1 (for further details, see Appendix A). This error is negligible when m is large. Appendix B discusses the associated error for two distributions: flat and exponential.

The simplified expression becomes

$$P(t_{n+1}) = P(t_n)G(t_{n+1}) \qquad (12)$$

A more general form for this expression is

$$P(t + \delta t(t)) = P(t)G(t + \delta t(t)) \qquad (13)$$

This may now be expanded into a differential equation.

$$P(t) + \frac{P'(t)}{\rho(t)} = P(t)G(t) + \frac{P(t)G'(t)}{\rho(t)} \qquad (14)$$

Using the fact that $G(t) = 1 - F(t)$ and that $G'(t) = -f(t)$, this equation may be simplified to the form

$$P'(t) = -P(t)\left[\rho(t)F(t) + f(t)\right] \qquad (15)$$

The solution to this differential equation is

$$P(t) = e^{-F(t) - \int_0^t \rho(\tau)F(\tau)d\tau} \qquad (16)$$

The average number of events processed in a cycle m can be computed as the sum of the probabilities of each individual event being in the current event cycle. This sum can be recast into an integral by using the fact that $\rho(t)\ \delta t(t) = 1$.

$$m = \sum_{i=0}^{n-1} P(t_i) = \sum_{i=0}^{n-1} P(t_i)\left[\rho(t_i)\delta t(t_i)\right] = \int_0^{\infty} P(t)\rho(t)dt \qquad (17)$$

Finally, the average event horizon time-interval can be calculated as

$$T_{EH} = \sum_{i=0}^{i=n} P(t_i)\delta t(t_i) = \int_0^{\infty} P(t)dt \qquad (18)$$

Minimum Time Delays

One very important class of event-generation distributions should now be noted. If there is a minimum time delay T between an event and its generated event (i.e., a parallel simulation that might lend itself to the standard Fixed Time Bucket approach), then $P(t)$ and $G(t)$ are equal to 1 for $0 < t < T$. This is a very useful piece of information because it allows one to quickly make a conservative estimate of the number of events m processed in a cycle.

$$\begin{aligned} m &= \int_0^T \rho_0 dt + \int_T^{\infty} P(t)\rho(t)dt \\ &\geq \rho_0 T \end{aligned} \qquad (19)$$

Exponential Distributions

Another very important class of simulations use exponential event generation distributions $f(t) = e^{-t}$. For this distribution, m can be calculated analytically (see Appendix C). The results are

$$m = e^{(1/2n)}\sqrt{2\pi n}\left[erf(\sqrt{n} + \frac{1}{\sqrt{n}}) - erf(\frac{1}{\sqrt{n}})\right] \qquad (20)$$

where $erf(x)$ is the area under the Standard Normal Curve (mean = 0, standard deviation = 1) from 0 to x. For large n (i.e., $n > 10$), this expression very accurately reduces to

$$m = \sqrt{\frac{(n+1)\pi}{2}} - 1 - \frac{1}{2n} \qquad (21)$$

This theoretical expression for m was compared with actual measurements. The difference was consistent with the claim that the error in the expression for m is less than one.

THE BETA-DENSITY FUNCTION

Now that an analytic model has been derived to describe the event horizon for general probability distributions, this model should be tested with probability distributions that have different characteristics. The beta-density function is ideal for this purpose because it can be manipulated into a wide variety of shapes. The beta-density function (which is bounded by 0 and 1) is written as (Papoulis 65)

$$\beta^{n_1,n_2}(t) = \frac{n_1 + n_2 + 1}{n_1!n_2!}t^{n_1}(1-t)^{n_2} \qquad (22)$$

Note that more general distributions can be obtained by adding multiple weighted beta-density functions together. For example, a two-hump distribution can be defined (with $n_1 \neq n_2$) as

$$\beta^{n_1,n_2}_{\text{two hump}}(t) = \alpha\beta^{n_1,n_2} + (1-\alpha)\beta^{n_2,n_1} \qquad (23)$$

Table 1 describes the beta-density function for various characteristic values of n_1 and n_2.

Table 1: Beta-density-function characteristics.

n_1	n_2	Description
0	0	Flat
1	0	Triangle-Up
0	1	Triangle-Down
$n >> 1$	0	Far-Future
0	$n >> 1$	Near-Future
n	n	Bell-Shaped $< t > = 1/2$, $\sigma^2 = 1/(8n+12)$
$n_1 \neq 0$	$n_2 \neq 0$	Asymmetric Bell-Shaped ($n_1 \neq n_2$)
∞	0	$\delta(t-1)$
0	∞	$\delta(t)$

For the purpose of analysis in this article, several representative beta functions were chosen to test and measure the equations derived in the previous section. To represent the Far-Future, the Near-Future, and the Bell-Shaped distributions, $n = 20$. For the Two-Hump distribution, $n_1 = 20$, and $n_2 = 0$ with $\alpha = 0.5$. This study was limited to two Asymmetric distributions, one for the Near-Future ($n_1 = 2, n_2 = 18$), and one for the Far-Future ($n_1 = 18, n_2 = 2$). These beta distributions are depicted in Figure 3 below.

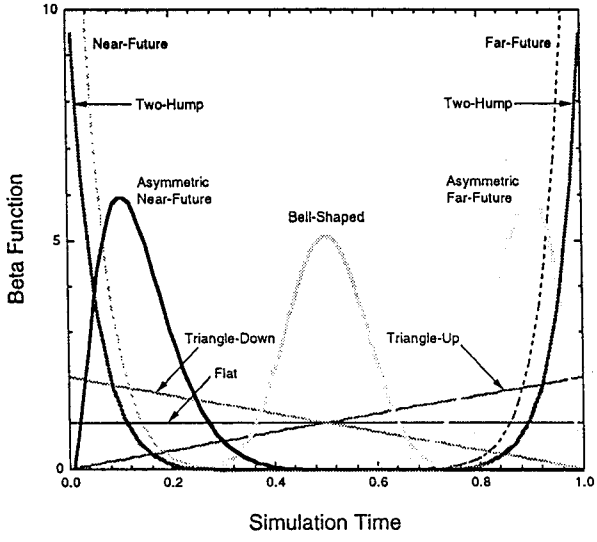

Figure 3: Beta-density functions.

One important note is that each of the different beta functions used in this analysis has a minimum lookahead value of zero; yet vastly different amounts of parallelism are predicted. Furthermore, the Flat, Bell-Shaped, and Two-Hump distributions all have average lookahead values of 1/2, yet they also have large differences in their inherent parallelism.

Event Densities Using the Beta Function

The first step in applying the beta function to the theoretical model is to compute the equilibrium event density $\rho(t)$. Understanding the event density for different types of distributions is instructive and should give developers a feel for what is occurring in their simulations. By expanding the beta function into a polynomial in t, the event density was calculated exactly. The density of events (normalized to the number of events n in the simulation) is depicted in Figure 4.

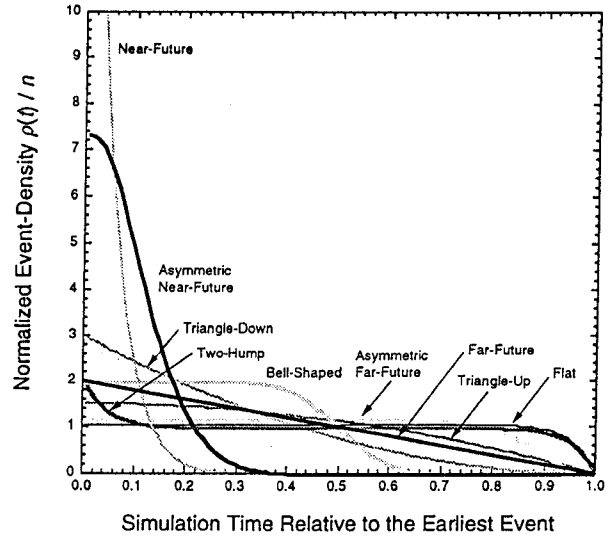

Figure 4: Normalized equilibrium event densities for the beta function.

Computing m for the Beta Function

The final and most important step in this analysis is to compute m. This predicts how many events can be processed in a cycle on the average for different types of event-generation statistics. Because these events could have been processed in parallel, this calculation also determines how much parallelism is inherent to the simulation. It also provides insight into how to extract parallelism in a simulation by carefully designing a strategy for event generation that maximizes m. Each of these curves (generated by equation 12) was checked both experimentally and analytically (using equation 11). The agreement was excellent (i.e., statistically consistent with the claim that the error in the analytic expression for m is less than one). The curves fall into four groups, all having similar behavior (see Figure 5).

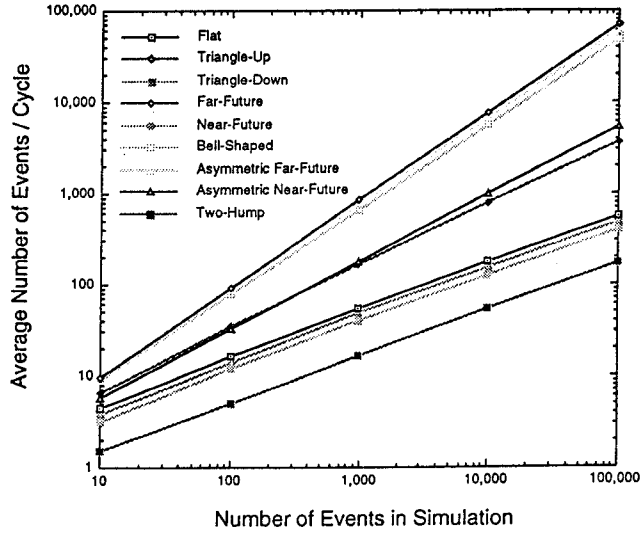

Figure 5: The average number of events processed per cycle m.

The first group consists of the Far-Future, Bell-Shaped, and the Asymmetric Far-Future beta distributions. All three of these distributions have one thing in common: they almost always schedule their events well into the future (see Figure 3). Because of this, many events on the average (tens of thousands for n = 100,000) are processed each cycle.

The second group consists of the Asymmetric Near-Future and Triangle-Up distributions which tend to schedule their events far into the future, but not in the same extreme manner as the first group. Still, many events (thousands for n = 100,000) are processed in each cycle. Simulations with event scheduling of this type can still achieve a high degree of parallelism.

The third group consists of the Flat, Triangle-Down, and Near-Future distributions which tend to schedule their events into the near future. The number of events per cycle (hundreds for n = 100,000) is reduced for simulations that schedule their events in this manner. The lesson here is that one should try to avoid scheduling events in the near future. It is better (in terms of increasing m) to schedule events further out in time.

The last category is the Two-Hump distribution. This type of distribution has the smallest value for m. The density of events in Figure 4 is not too high around t = 0, yet 50 percent of the time (see Figure 3) the two hump distribution schedules an event in the near future. This cuts the event horizon shorter than the other distributions. Nevertheless, there is still a reasonable amount of parallelism in the simulation. However, simulations of this type may not scale very well on large machines (Bellenot 93).

SUMMARY

The event horizon is an important simulation concept. It is defined to be the time tag of the earliest new event generated in an event-processing cycle. Processing events in cycles with boundaries defined by the event horizon can have important consequences for parallel simulations as well as for event-list management.

A mathematical model was derived from first principles that predicts the average number of events, processed in cycles defined by the event horizon. The beta-density function was then used to study how the event horizon is affected by different types of event-generation statistics. The results showed that scheduling events into the far future allows for more parallelism in a simulation than scheduling events into the near future. This confirms the notion that lookahead improves the performance of parallel simulations.

Maximizing the event horizon through lookahead should be the strategy of every parallel simulation developer. Even though parallel simulations that exploit lookahead may become more complex than their sequential counterparts, the performance payoffs can be great.

APPENDIX A

In this appendix, the steps are shown that simplify the expression for the probability of an event being included in the current event horizon cycle. The maximum error, when using equation 12 instead of equation 11 to compute m, is shown to be 1. The starting point is the exact expression.

$$P(t_{n+1}) = G(t_{n+1} - t_0)G(t_{n+1} - t_1)\ldots G(t_{n+1} - t_n) \qquad \text{(a1)}$$

Now, note that $t_{n+1} - t_i = t_n - t_{i-1} + \delta t_n - \delta t_i$. Define $\Delta_i = \delta t_n - \delta t_i$ and rewrite the expression above as

$$P(t_{n+1}) = G(t_{n+1} - t_0)\begin{Bmatrix} G(t_n - t_0 + \Delta_0)G(t_n - t_1 + \Delta_1) \\ \ldots G(t_n - t_{n-1} + \Delta_{n-1}) \end{Bmatrix} \qquad \text{(a2)}$$

Then expand each of the G terms to first order in Δ_i.

$$P(t_{n+1}) = G(t_{n+1} - t_0)\begin{Bmatrix} [G(t_n - t_0) + \Delta_0 G'(t_n - t_0)] \\ [G(t_n - t_1) + \Delta_1 G'(t_n - t_1)]\ldots \\ [G(t_n - t_{n-1}) + \Delta_{n-1} G'(t_n - t_{n-1})] \end{Bmatrix} \qquad \text{(a3)}$$

Multiplying this out and only keeping terms up to first order in Δ_i, the expression becomes

$$P(t_{n+1}) = G(t_{n+1})P(t_n)\left\{1 + \sum_{i=0}^{n} \frac{G'(t_n - t_i)}{G(t_n - t_i)}\Delta_i\right\} \qquad \text{(a4)}$$

This can be simplified as

$$P(t_{n+1}) = G(t_{n+1})P(t_n)\{1 + \varepsilon_n\} \qquad \text{(a5)}$$

where

$$\varepsilon_n = \sum_{i=0}^{n} \frac{G'(t_n - t_i)}{G(t_n - t_i)}\Delta_i \qquad \text{(a6)}$$

Note that $G'(t_n - t_i) = -f(t_n - t_i) \le 0$. Also, $\Delta_i = \delta t_n - \delta t_i \ge 0$. Thus, $\varepsilon_n \le 0$. One can obtain an analytic expression for ε_n as follows:

$$\varepsilon_n = \sum_{i=0}^{n} \frac{G'(t_n - t_i)}{G(t_n - t_i)}\delta t_n - \sum_{i=0}^{n} \frac{G'(t_n - t_i)}{G(t_n - t_i)}\delta t_i \qquad \text{(a7)}$$

Replacing the sums with integrals

$$\varepsilon_n = \frac{1}{G(t_n)}\int_0^{t_n} \frac{G'(t_n - t)G(t)}{G(t_n - t)}dt - \int_0^{t_n} \frac{G'(t_n - t)}{G(t_n - t)}dt \qquad \text{(a8)}$$

The second integral can be performed analytically, while the first integral is dependent on $G(t)$. Thus,

$$\varepsilon_n = \frac{1}{G(t_n)}\int_0^{t_n} \frac{G'(t_n - t)G(t)}{G(t_n - t)}dt - \log G(t_n) \qquad \text{(a9)}$$

Define the quantity A as the first term in this expression.

$$A = \frac{1}{G(t_n)}\int_0^{t_n} \frac{G'(t_n - t)G(t)}{G(t_n - t)}dt \qquad \text{(a10)}$$

Upper and lower bounds can be found for A by computing A_1 using constant $G(t) = G(0) = 1$, and then A_2 using constant $G(t) = G(t_n)$. Note that A is negative because $G'(t_n - t)$ is negative. Thus, $A_1 < A < A_2$.

$$A_1 = \frac{1}{G(t_n)}\int_0^{t_n} \frac{G'(t_n - t)}{G(t_n - t)}dt = \frac{1}{G(t_n)}\log G(t_n) \qquad \text{(a11)}$$

and

$$A_2 = \frac{1}{G(t_n)}\int_0^{t_n} \frac{G'(t_n - t)G(t_n)}{G(t_n - t)}dt = \log G(t_n) \qquad \text{(a12)}$$

The bounds for ε_n are

$$\left\{\frac{1}{G(t_n)}-1\right\}\log G(t_n)\le\varepsilon_n\le 0 \tag{a13}$$

Now, use the fact that $G(t_n) = 1 - F(t_n)$.

$$\left\{\frac{F(t_n)}{1-F(t_n)}\right\}\log(1-F(t_n))\le\varepsilon_n\le 0 \tag{a14}$$

Finally, using the expansion for $\log(1-x) \approx -x$ for small x, the expression becomes

$$-F(t_n)^2\le\varepsilon_n\le 0 \tag{a15}$$

This approximation is valid because as $F(t)$ gets large, $P(t)$ rapidly gets small. Therefore, the important terms in the sum for m are the ones where $F(t)$ is small. Also, note that equation a5 implies that ε_n can never be less than -1. With bounds on ε_n, the expression for $P(t_n)$ in equation a5 can be expanded more generally as

$$\begin{aligned}P(t_n)&=G(t_n)P(t_{n-1})(1+\varepsilon_{n-1})\\&=G(t_n)\{G(t_{n-1})P(t_{n-2})(1+\varepsilon_{n-2})\}(1+\varepsilon_{n-1})\\&=G(t_n)G(t_{n-1})\ldots G(t_0)(1+\varepsilon_{n-1})(1+\varepsilon_{n-2})\ldots(1+\varepsilon_0)\\&\approx G(t_n)G(t_{n-1})\ldots G(t_0)\left\{1+\sum_{i=0}^{n-1}\varepsilon_i\right\}\end{aligned} \tag{a16}$$

Defining m_1 as the case where $\varepsilon_n = 0$ (i.e., the assumption in equation 12 of the text), and m_2 as the case when $\varepsilon_n = -F(t_n)^2$ (i.e., the case with the maximum error), an upper bound for the error in computing m can be determined.

$$m_2=\sum_{n=0}^{\infty}G(t_n)G(t_{n-1})\ldots G(t_0)\left\{1-\sum_{i=0}^{n-1}F(t_i)^2\right\} \tag{a17}$$

Let $\delta m = m_2 - m_1$. Then the maximum error in computing m is given as

$$\begin{aligned}\delta m&=-\sum_{n=0}^{\infty}G(t_n)G(t_{n-1})\ldots G(t_0)\sum_{i=0}^{n-1}F(t_i)^2\\&=-\sum_{n=0}^{\infty}[1-F(t_n)][1-F(t_{n-1})]\ldots[1-F(t_0)]\sum_{i=0}^{n-1}F(t_i)^2\end{aligned} \tag{a18}$$

If the limit in the second sum is raised from $n-1$ to n, an inequality can be formed.

$$\delta m\ge-\sum_{n=0}^{\infty}[1-F(t_n)][1-F(t_{n-1})]\ldots[1-F(t_0)]\sum_{i=0}^{n}F(t_i)^2 \tag{a19}$$

The order of the sums can be reversed by considering each of the terms that multiply $F(t_i)^2$ separately.

$$\begin{aligned}\delta m\ge-\sum_{n=0}^{\infty}&F(t_n)^2[1-F(t_n)][1-F(t_{n-1})]\ldots[1-F(t_0)]\\&\times\{1+[1-F(t_{n+1})]+[1-F(t_{n+2})][1-F(t_{n+1})]+\ldots\}\end{aligned} \tag{a20}$$

Because $F(t)$ is a non-decreasing function, the inequality still holds if all of the time values in the bracket are replaced by t_n. Also, note that by definition, $F(t_0) = 0$.

$$\begin{aligned}\delta m\ge-\sum_{n=1}^{\infty}&F(t_n)^2[1-F(t_n)][1-F(t_{n-1})]\ldots[1-F(t_1)]\\&\times\left\{\sum_{i=0}^{\infty}[1-F(t_n)]^i\right\}\end{aligned} \tag{a21}$$

It is easy to show that the sum in the last bracket is equal to $1/F(t_n)$. Therefore, the expression reduces to

$$\delta m\ge-\sum_{n=1}^{\infty}F(t_n)[1-F(t_n)][1-F(t_{n-1})]\ldots[1-F(t_1)] \tag{a22}$$

Now, consider what happens to the sum if $F(t)$ is a constant value F. Call this sum, S_1.

$$\begin{aligned}S_1&=-\sum_{n=1}^{\infty}F[1-F]^n=-F\sum_{n=0}^{\infty}[1-F]^n+F\\&=-1+F\\&\ge-1\end{aligned} \tag{a23}$$

Suppose that instead of $F(t)$ being constant for all t, $F(t) = F$ for $t \le t_n$, and $F(t) = F'$ for $t > t_n$. Call this sum, S_2.

$$\begin{aligned}S_2&=-\sum_{i=1}^{n}F[1-F]^i-\sum_{i=1}^{\infty}F'[1-F']^i[1-F]^n\\&=-\left\{\sum_{i=1}^{\infty}F[1-F]^i-\sum_{i=n+1}^{\infty}F[1-F]^i\right\}-[1-F]^n[1-F']\\&=-\left\{[1-F]-[1-F]^n\sum_{i=1}^{\infty}F[1-F]^i\right\}-[1-F]^n[1-F']\\&=-\{[1-F]-[1-F]^n[1-F]\}-[1-F]^n[1-F']\\&=-[1-F]+[1-F]^n[F'-F]\\&\ge S_1\end{aligned} \tag{a24}$$

From the fact that $-1 \le S_1 \le S_2$, the following inequality holds.

$$
\begin{aligned}
-1 \le & -\sum_{n=1}^{\infty} F(t_1)[1-F(t_1)]^n \\
& \le -F(t_1)[1-F(t_1)] - [1-F(t_1)]\sum_{n=1}^{\infty} F(t_2)[1-F(t_2)]^n \\
& \le -F(t_1)[1-F(t_1)] - F(t_2)[1-F(t_2)][1-F(t_1)] \\
& \quad -[1-F(t_2)][1-F(t_1)]\sum_{n=1}^{\infty} F(t_3)[1-F(t_3)]^n \\
& \le \sum_{n=1}^{\infty} F(t_n)[1-F(t_n)][1-F(t_{n-1})]\ldots[1-F(t_1)]
\end{aligned}
\tag{a25}
$$

The proof is now complete (compare equation a25 with a22).

$$-1 \le \delta m \le 0 \tag{a26}$$

Therefore, it is safe to conclude that the maximum error (when using equation 12 in the text to compute m) is 1. This error is negligible for large m.

APPENDIX B

In this appendix, the error ε_n described by equation a9 in Appendix A is derived for the flat distribution $f(t) = 1$, and for the exponential distribution $f(t) = e^{-t}$. The maximum error in computing m for flat and exponential distributions is shown to be at most 1/2.

Flat Distribution

For the flat distribution, $f(t) = 1$, $F(t) = t$, and $G(t) = 1 - t$.

$$\varepsilon_n = \frac{1}{1-t_n}\int_0^{t_n} \frac{t-1}{1-t_n+t}\,dt - \log(1-t_n) \tag{b1}$$

This integral can be simplified by letting $u = 1 - t_n + t$.

$$\varepsilon_n = \frac{1}{1-t_n}\int_{1-t_n}^{1} \frac{u+(t_n-2)}{u}\,du - \log(1-t_n) \tag{b2}$$

This can easily be integrated to obtain

$$\varepsilon_n = \frac{1}{1-t_n}\{t_n - (t_n-2)\log(1-t_n)\} - \log(1-t_n) \tag{b3}$$

Expanding this equation to second order in t_n

$$
\begin{aligned}
\varepsilon_n &\approx (1+t_n)\{t_n - (t_n-2)(-t_n - t_n^2/2)\} + t_n + t_n^2 \\
&\approx -t_n^2/2
\end{aligned}
\tag{b4}
$$

Then, using $F(t_n) = t_n$

$$\varepsilon_n \approx F(t_n)^2/2 \tag{b5}$$

Therefore, equations a5 and a18 predict that the maximum error in computing m is 1/2 for flat distributions. This error has been measured for n = 100,000 to be 0.334151.

Exponential Distribution

For the exponential distribution, $f(t) = e^{-t}$, $F(t) = 1 - e^{-t}$, $G(t) = e^{-t}$. Plugging this into the expression for ε_n

$$\varepsilon_n = e^{t_n}\int_0^{t_n} \frac{-e^{-(t_n-t)}}{e^{-(t_n-t)}} e^{-t}\,dt - \log(e^{-t_n}) \tag{b8}$$

This simplifies to

$$\varepsilon_n = e^{t_n}\int_0^{t_n} e^{-t}\,dt + t_n \tag{b9}$$

The result of this integral is

$$
\begin{aligned}
\varepsilon_n &= -e^{t_n}(1-e^{-t_n}) + t_n = 1 - e^{t_n} + t_n \\
&\approx -t_n^2/2
\end{aligned}
\tag{b10}
$$

To first order in t, $F(t) = t$. This results in

$$\varepsilon_n = -F(t_n)^2/2 \tag{b11}$$

Thus, the error in computing m for exponential functions is at most 1/2 (see the above discussion for the flat distribution). The error has been measured for n = 100,000 to be 0.334168.

APPENDIX C

This appendix shows the steps for computing m using the exponential distribution $f(t) = e^{-t}$. First, note that

$$F(t) = \int_0^t e^{-\tau}\,d\tau = 1 - e^{-t} \tag{c1}$$

and

$$G(t) = e^{-t} \tag{c2}$$

Because $\rho(t) = \rho_0\, G(t)$, one can write

$$n = \rho_0 \int_0^{\infty} e^{-t}\,dt = \rho_0 \tag{c3}$$

Thus, $\rho(t) = n\, e^{-t}$. Now, plug this into the expression for $P(t)$ to obtain

$$P(t) = e^{-(1-e^{-t})-n\int_0^t e^{-\tau}(1-e^{-\tau})\,d\tau} \tag{c4}$$

Completing the integral and simplifying the expression results in

$$P(t) = e^{-(1+n/2)} e^{\left\{(1+n)e^{-t} - (n/2)e^{-2t}\right\}} \tag{c5}$$

Then, m can be computed by integrating

$$m = \int_0^\infty n e^{-t} P(t) dt \qquad \text{(c6)}$$

This integral can be written as

$$m = n e^{-(1+n/2)} \int_0^\infty e^{\left\{-t+(1+n)e^{-t}-(n/2)e^{-2t}\right\}} dt \qquad \text{(c7)}$$

This expression looks complicated, but by making the substitution $u = e^{-t}$, the integral can be simplified to

$$m = n e^{-(1+n/2)} \int_0^1 e^{\left\{(1+n)u-(n/2)u^2\right\}} du \qquad \text{(c8)}$$

By completing the square inside the exponential function and factoring the constant term that is outside the integral, the expression becomes

$$m = n e^{-(1+n/2)} e^{(1+n)^2/2n} \int_0^1 e^{-\frac{n}{2}\left\{u-\frac{1+n}{n}\right\}^2} du \qquad \text{(c9)}$$

This expression can be further simplified by combining the two exponential terms outside the integral, and by making another substitution $v = u - (1 + n) / n$.

$$m = n e^{1/2n} \int_{-1-1/n}^{-1/n} e^{-\frac{n}{2}v^2} dv \qquad \text{(c10)}$$

The function inside the integral is even in the variable v. Therefore, the limits in the integral can be changed from negative values to positive values by switching their signs, and then switching their order. Making a final substitution $x = n^{1/2} v$, and putting this in the form of the Standard Normal Curve, the expression reduces to

$$m = \sqrt{2\pi n} e^{1/2n} \left\{ \frac{1}{\sqrt{2\pi}} \int_{n^{-1/2}}^{n^{1/2}+n^{-1/2}} e^{-x^2/2} dx \right\} \qquad \text{(c11)}$$

This can be written in terms of the error function which is defined for the Standard Normal Curve.

$$m = \sqrt{2\pi n} e^{1/2n} \left\{ erf(\sqrt{n} + 1/\sqrt{n}) - erf(1/\sqrt{n}) \right\} \qquad \text{(c12)}$$

Now, for large n, this simplifies to

$$erf(\sqrt{n} + 1/\sqrt{n}) \approx \frac{1}{2} \qquad \text{(c13)}$$

and

$$erf(1/\sqrt{n}) \approx erf(0) + \left(1/\sqrt{n}\right) erf'(0) \approx \frac{1}{\sqrt{2\pi n}} \qquad \text{(c14)}$$

Expanding $e^{1/(2n)} \approx 1 + 1/(2n)$, and then putting this all back together into the expression, the result is

$$m \approx \sqrt{2\pi n}\left(1 + 1/2n\right)\left(1/2 - 1/\sqrt{2\pi n}\right) \qquad \text{(c15)}$$

This expression simply reduces to

$$m = \sqrt{\frac{(n+1)\pi}{2}} - 1 - \frac{1}{2n} \qquad \text{(c16)}$$

The error introduced by approximating the error function in the above expression is much less than 1 percent for $n > 10$.

ACKNOWLEDGMENTS

The research described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, and was sponsored by SDIO through the National Test Facility at Falcon Air Force Base, Colorado Springs, and through Innovative Science & Technology at the Pentagon by agreement with the National Aeronautics and Space Administration.

Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government or the Jet Propulsion Laboratory, California Institute of Technology.

BIOGRAPHY

Jeff Steinman received B.S. degrees in computer science and in mathematical physics from California State University Northridge in 1980. He then worked at Hughes Aircraft Company in the Radar Systems Group for four years while studying physics at UCLA. In 1988, he received his Ph.D. in experimental high-energy particle physics from UCLA, where he measured the quark content of virtual photons generated at the Stanford Linear Accelerator Center. Since then, he has been a member of the technical staff working at the Jet Propulsion Laboratory building simulations for strategic missile and air defense. Jeff Steinman is the principal developer of the SPEEDES operating system.

REFERENCES

Bellenot S. 1993. "Performance of a Riskfree Time Warp Operating System." In Proceedings of the *7'th Workshop on Parallel and Distributed Simulation* (PADS93). Vol. 23, No. 1, July 1993, Pages 155-158.

Chandy K. and Misra J. 1979. "Distributed Simulation: A Case Study in Design and Verification of Distributed Programs." *IEEE Transactions on Software Engineering*. Vol. SE-5, No. 5, Pages 440-452.

Chandy K. and Sherman R. 1989. "Space, Time, and Simulation." In *Proceedings of the SCS Multiconference on Distributed Simulation*. Vol. 21, No. 2, pages 53-57.

Chou C., Bruell S., Jones D. 1993. "A Generalized Hold Model." In *Proceedings of the 1993 SCS Winter Simulation Conference*. Pages 756-761.

Dickens P. and Reynolds P. 1990. "SRADS With Local Rollback." In *Proceedings of the SCS Multiconference on Distributed Simulation*. Vol. 22, No. 1, Pages 161-164.

Felderman R. and Kleinrock L. 1992. "Two Processor Conservative Simulations Analysis." *Proceedings of the SCS Multiconference on Advances in Parallel and Distributed Simulation*. Vol. 24, No. 3, pages 169–177.

Fujimoto R. 1988. "Lookahead in Parallel Discrete Event Simulation." *International Conference on Parallel Processing*. Vol. 3, pages 34–41.

Fujimoto R. 1990. "Parallel Discrete Event Simulation." *Communications of the ACM*. Vol. 33, No. 10, pages 30–53.

Hawking S. 1988. "A Brief History of Time." Bantam Books, New York, New York, 1988.

Jefferson D. 1985. "Virtual Time." ACM Transactions on Programming Languages and Systems. Vol. 7, No. 3, Pages 404-425

Jones D. 1986. "An Empirical Comparison of Priority-Queue and Event-Set Implementations." *Communications of the ACM*. Vol. 29, No. 4, pages 300–311.

Lin Y. and Lazowska E. 1990. "Exploiting Lookahead in Parallel Simulation." *IEEE Transactions on Parallel and Distributed Systems*. Vol. 1, No. 4, pages 457–469.

Lubachevsky B. 1988. "Bounded Lag Distributed Discrete Event Simulation." *Multi-Conference on Distributed Simulation*. (February) Pages 183-191.

Lubachevsky B. 1989. "Rollback Sometimes Works... If Filtered." In *Proceedings of the SCS Winter Simulation Conference*. Pages 630-639.

Mehl H. 1991. "Speed-Up of Conservative Distributed Discrete-Event Simulation Methods by Speculative Computing (Short Version)." In *Proceedings of Advances in Parallel and Distributed Simulation*. Vol. 23, No. 1, Pages 163-166.

Nicol D. 1991. "Performance Bounds on Parallel Self-Initiating Discrete Event Simulations." *ACM Transactions on Modeling and Computer Simulation*. Vol. 1, No. 1, Pages 24-50.

Papoulis A. 1965. "Probability, Random Variables, and Stochastic Processes." McGraw-Hill Series in System Science, New York, pages 104, 147.

Reynolds P. 1988. "A Spectrum of Options for Parallel Simulation." In *Proceedings of the 1988 Winter Simulation Conference*. Pages 325-332.

Ronngren R., Riboe J., and Ayani R. 1991. "Lazy Queue: An Efficient Implementation of the Pending-event Set." Proceedings of the *24'th Annual Simulation Symposium*, pages 194-204.

Sleator D., and Tarjan R. 1985. "Self Adjusting Binary Search Trees." *Journal of the ACM*. Vol. 32, No. 3, pages 652–686.

Steinman J. 1992. "SPEEDES: A Multiple-Synchronization Environment for Parallel Discrete-Event Simulation." *International Journal in Computer Simulation*. Vol. 2, Pages 251-286.

Steinman J. 1993. "Breathing Time Warp." In Proceedings of the *7'th Workshop on Parallel and Distributed Simulation* (PADS93). Vol. 23, No. 1, July 1993, Pages 109-118.

Turner S. and Xu M. 1992. "Performance Evaluation of the Bounded Time Warp Algorithm." *Proceedings of the SCS Multiconference on Advances in Parallel and Distributed Simulation*. Vol. 24, No. 3, pages 117-126.

Wieland F., et al. 1992. "A Critical Path Tool for Parallel Simulation Performance Optimization." *Proceedings of the Hawaii International Conference on System Sciences*. Vol. 2, Pages 196-206.