

FINDING LOOKAHEAD IN WARGAMING: A REQUIREMENT FOR SCALEABLE PARALLEL SIMULATION

Emmet Beeker
Senior Software Architect
GRCI, An AT&T Company
Vienna, VA
ebeeker@grci.com

John Chludzinski
Senior Software Engineer
GRCI, An AT&T Company
Orlando, FL
jchludzinski@grci.com

KEYWORDS: discrete event simulation, conservative synchronization, optimistic synchronization, lookahead, parallelism

ABSTRACT: Military simulations generally fall into two main categories: time-stepped and discrete-event simulation (DES). Where it can be used, DES provides performance advantages and can reduce temporal distortions imposed upon time-stepped simulations. In fact, some time-stepped simulations, such as the OneSAF Testbed (OTB), use some discrete-event techniques to mitigate these problems. The main difficulty with DES is scaling beyond a single processor. Parallel processing of discrete events requires that causality be retained, even while events are processed out of strict time ordering. The two approaches to this are optimistic and conservative. Optimistic simulation allows processors to simulate events, assuming that they are temporally correct. When it is discovered that there is a temporal discrepancy, the simulation is “rolled back” to the time of the discrepancy and then proceeds again. Conservative simulation never allows discrepancies – event processing is only allowed when it can be guaranteed that the event will not be altered. Conservative parallel discrete event simulation (PDES) depends upon a temporal separation between cause and effect, i.e. lookahead. PDES has been successful in simulations where lookahead can easily be calculated as a function of physical properties and fixed relationships between

entities – typically in computer processor scheduling and in communications network simulation. In the naïve approach to wargaming it would appear the separation between cause and effect can be arbitrarily small. Using domain knowledge and a more relaxed view of lookahead, a battle can be partitioned into local interest sets. This paper examines a relaxed concept of lookahead in wargaming and what can be done when a non-zero lookahead cannot be guaranteed.

INTRODUCTION

Wargaming models can be roughly divided in two broad categories: training models and analytical models. Traditionally training models have been (fixed) time-stepped, detailed-physics based models where the concern has been for “realistic behavior” for human (trainee) interaction. Typically, these “tick-based” models are updated every clock tick using a round robin scheduler. Analytic models are in some ways more natural candidates for discrete event modeling, in that they need only schedule events when essential to their logical behavior in the simulation. As a consequence, the modeler is left with greater latitude about where to find lookahead.

OVERVIEW OF PARALLEL

SYNCHRONIZATION

The potential to exploit parallelism within a simulation is varied (e.g., replicated trials, distributed functions, etc.); this paper will focus on one: distributing the message list (event queue) among multiple parallel processes, i.e. logical processes (LPs). Each LP has a single message list (is sequential) and sends and receives messages. No other communication, besides time-stamped messages, is used to synchronize the LPs. Models within the simulation are partitioned among LPs where, if two models (M1 and M2) need to communicate and are hosted on separate LPs (LP1 and LP2), a *channel* is said to exist from LP1 to LP2.

Messages sent through a channel are queued in time-stamp order by the receiver. Each LP's *local clock* (or *local virtual time*) is defined to be the time-stamp of earliest message queued on an incoming channel. If an incoming channel has no messages queued, the time-stamp of the last message sent through the channel is used. As messages are processed, the local clock is said to *advance*. To guarantee *causality* within a simulation, messages must be processed in nondecreasing time-stamp order, i.e., an LP may **never** receive a message in its past (with a time-stamp less than the local clock).

Within PDES two fundamental schemes exist for maintaining synchronization among LPs: conservative and optimistic synchronization. The conservative schemes, historically, were the first approaches to parallelizing simulations and guaranteed causal ordering and processing of messages.

Conservative Synchronization

The conservative protocol guarantees that **no** LP will receive a message with a time-stamp in the LP's past. To ensure this, conservative

algorithms use a concept called *lookahead* to enforce two rules:

- i) all messages sent by one LP to another are sent in nondecreasing order, and
- ii) for a given LP, if an incoming channel has an empty queue associated with it, no further messages will be processed, i.e. the LP halts execution.

These two rules are necessary and sufficient to guarantee that a simulation consisting of a set of concurrent LPs, so cooperating, will be causal.

A simulation using the two rules stated above will almost certainly deadlock. To deal with this two approaches to conservative synchronization have been used: *deadlock avoidance* and *deadlock detection and recovery*.

The most commonly used deadlock avoidance algorithm is Chandy-Misra-Bryant (CMB) and its many variants. This approach uses a scheme of *null messages* to keep the clock advancing. A null message contains no data to be processed, only its time-stamp to allow the local clock to safely advance. A primary concern of this scheme is to reduce the number of null messages. Many of its variants have proposed differing approaches to accomplish this: carrier null messages, path lookahead, etc.

Deadlock detection and recovery algorithms rely on some method to detect when all LPs have halted execution (deadlock) and break the deadlock by determining (calculating) what time all LPs may safely advance to. Typically, to detect deadlock an algorithm is used which constructs a tree of the active LPs in the simulation. Those LPs which are halted (inactive) are added to the tree as they become

active. The root of the tree (an LP) is designated the *controller*. When the root become childless the simulation is deadlocked. It is the responsibility of the controller to calculate the earliest time to which all the LPs may safely advance. Those LPs which may safely process messages become the children of the controller (are activated by the controller). They in turn begin to send messages which activate other LPs; the newly activated LPs become the children of the LP which sent the message activating them, thus beginning a new cycle. One problem with this approach is the number of active LPs can be quite small, especially when near deadlock.

Numerous approaches, centered around the concept of a *barrier*, have been proposed to avoid this problem. A barrier is an artificial time, usually wallclock time, at which all LPs halt execution (*enter* the barrier), instead of allowing the simulation to deadlock.

Again, a controller is used to determine which messages are safe to process and those LPs with safe messages, after resuming execution (*exiting* the barrier), execute them. A simulation repeating this cycle thus avoids waiting for the entire system of LPs to deadlock.

Lookahead

Lookahead in its broadest sense can simply be defined as a lower bound on the temporal separation between cause and effect. It establishes a minimum reaction time and is essential to modeling accurate and credible behaviors. From the point-of-view of the modeler lookahead is a double edged sword: The larger the lookahead the greater the potential to exploit parallelism, but the less tightly models may interact. Conversely, the smaller the lookahead the more tightly models may interact, but the potential to exploit parallelism is less. This leaves it to the modeler to make an engineering judgement: How large

can I make lookahead and still credibly model the behaviors I'm responsible for and how small can I make lookahead and still get the performance I need?

Most commonly, lookahead values are global and fixed. Neither need be the case. Using barrier synchronization points, the value can be changed. In addition, it is possible to group LPs into clusters, each with its own local lookahead. Each cluster may be viewed as a separate virtual-LP where incoming and outgoing channels from LPs external to the cluster define the incoming and outgoing channels to the cluster. In this way, a graph of LPs and channels is partitioned into clusters functioning as virtual-LPs. There is a lookahead for the graph of clusters and a lookahead local to each cluster.

The value of dynamic clustering is to allow LPs hosting models which become tightly interactive to group into a cluster and establish a local, smaller valued lookahead without forcing the entire simulation to use this smaller value. Because events for models on a common LP are executed sequentially, lookahead between them does not matter and can even be zero. This implies that dynamically moving tightly interactive models to the same LP could be beneficial. In practice, it is difficult to determine when the benefit of moving a model exceeds the cost for moving it.

Optimistic Synchronization

Unlike conservative synchronization, the optimistic protocol allows messages to be sent without consideration as to whether they are in the recipient's past. Thus, the optimistic protocol allows causality errors to occur and then corrects the error. To accomplish this, the simulation infrastructure must provide some mechanism to allow the models to maintain a history of the changes to their state data. When a message is received by an LP (a model hosted on the LP) in its past, the LP (all the models hosted on that LP) are required to rollback to a point in time where the state data for those models is consistent (correct) for the time of the message received.

The advantage to optimistic over conservative synchronization is there is no dependence on lookahead, even though lookahead can be exploited in optimistic schemes to improve performance. In addition, there is no need to define message channels between LPs.

The problem with this strategy is one of time and space; the storage dedicated to state data history and the time needed to restore models to some point in their past can be burdensome and counter productive to performance. In addition, implementing the infrastructure to support this is complex and hence, error prone.

Rollback

Rollback is simply a scheme whereby a model may retreat to its past using a history of its state data changes. This is usually accomplished by requiring the modeler to use *rollbackable* variables (containers) to capture all state data. Typically, these are linked lists whose nodes are labeled with the time-stamp of the message which resulted in the variable being updated. If an LP receives a message

labeled in its past (with a time-stamp less than its current local clock), the LP will rollback by chaining through the linked list looking for the first node labeled with a time-stamp less than or equal to that of the message. This must be done for all state data, for all models within the LP.

To establish a time, earlier than which a history of state data need not be maintained, optimistic methods use the concept of *global virtual time* (GVT). GVT is simply the minimum local clock value of all the LPs in the simulation and establishes a point in time beyond which no LP rolls back. In addition, GVT establishes a point in time where state data may be safely committed (to a database) for later analysis of simulation results, i.e. after action reports.

LOOKAHEAD IN WARGAMING MODELS

Parallel simulation has worked well for non-preemptive models. These models have tasks that run to completion and the next event in a causality chain is known. Examples include scheduling computer jobs on a processor, where each job has a minimum run time, and communications networks where each message occupies a link for a minimum time. Discrete event wargaming simulations are generally programmed with preemptive events. Events are conditional with the understanding that an external message can cause a change.

Traditionally, wargaming simulations have been divided into two categories: training and analytic. Training simulations create a virtual environment for human-in-the-loop actions. Human participants play an active roll *within* the simulation. The most important characteristic is that the simulation has the look and feel of realism. Human involvement in an analytic simulation is normally as an external observer. The central requirement for an

analytic simulation is to be accurate in its models; i.e., to maintain causality and precisely reproduce before and after relationships. This leads to different requirements for synchronization and look ahead.

Lookahead in Training Models

At first glance, it would appear that lookahead would not be available nor required in human-in-the-loop simulations. First, many of the human inputs are unpredictable. If the simulation must respond “immediately” to human inputs, then there is zero lookahead for those events. Human actions can preempt and invalidate the next expected event.

Secondly, the training simulation need not be causal ordered. It is important for human-in-the-loop training simulations to give the *appearance* of reality. The cues and actions must give valid training results. Accuracy in models is less important. This allows for some relaxation in causality. As long as a human cannot detect the difference, effect can precede cause. Generally, as long as events are not delayed beyond the visual frame rate, discrepancies can be tolerated.

The frame rate can be used for minimum lookahead. Any events that occur between frames must be accounted for by the next frame. Generally, a delay of $1/15^{\text{th}}$ of a second can be tolerated. However, this limit only needs to be met in the range of humans-in-the-loop. For events occurring elsewhere in the simulation, events that give context but do not interact with humans, lookahead can be found using the same techniques as in analytic simulations.

Lookahead in Analytic Models

In analytic models, state changes of entities in the model and the interactions

between the entities must be traceable to a strict causality chain. Generally we would like the results of a simulation to be exactly reproducible. When an unexpected outcome occurs, the analyst’s mission is to explain it. This would be much more difficult if the strange outcome were not reproducible. Correctness in the model becomes of prime importance.

It would appear that in wargaming models, the unpredictable is occurring throughout the simulation. Because wargaming simulations have sensor models and communication models, both working at the speed of light, it would seem that there would be little lookahead available. However, lookahead may be found by asking whether the physical phenomena or the effects of the phenomena are important. Most often, it is not the physical model itself that we are interested in.

The model of communications exists to “deliver” a message from one point to another. The effect of the communication is to cause some action to take place. For example, a fire request will result in artillery striking somewhere on the battlefield. The fact that the communication message travels at the speed of light is immaterial. The delay from request to artillery detonation may be all that we are interested in. From this viewpoint, the time from request to detonation may provide lookahead.

The modeler, who would like to execute a simulation in parallel, must always look for external interactions, those events that can affect other models hosted on different LPs. Any amount of internal interaction (within an LP) can be permitted, e.g. determining connectivity in a radio network, as long as the external interactions can be predicted with some advance warning.

There is both static and dynamic lookahead available in wargaming models. Static lookahead can be calculated before the models are executed in the simulation. The minimum reaction time from the call for artillery until the artillery is fired is an example of this. Dynamic lookahead depends upon conditions in the simulation. The time of flight for the artillery depends upon the distance the shells must travel. This could be calculated in the simulation and used as lookahead. It is dynamic lookahead that shows promise for wargaming. When it is known that external events will not affect local relationships for a calculable period of time, then this period is lookahead.

The issue now becomes whether the modeler can define a local situation in which external events will not affect local actions. This occurs in battle all of the time. Forces engaging in combat will analyze the situation and concentrate on relatively few other forces at any time. The major threat and/or the major target becomes all consuming. A pilot or tank driver will concentrate on the enemy that poses the greatest threat or offers the best target. If a minimum engagement time can be calculated, then this can provide lookahead needed to operate a wargaming simulation in parallel.

CONCLUSIONS

Based on our work with conservative protocols, it has been our experience that finding sufficient lookahead in wargaming models to allow adequate parallelism to significantly increase performance is very doable. And, that this can be done in such a fashion as to not compromise the credibility of the models. We believe this makes conservative synchronization a good choice for wargaming and have made it the underlying protocol for a new parallel simulation engine being developed by GRCI - *Ingenium*.

REFERENCES

D. Nicol and R. Fujimoto, *Parallel Simulation Today*, in *Annals of Operations Research: Simulation and Modeling*, edited by O. Balci, Vol. 53, pp. 249-286, 1994.

J. Banks, J.S. Carson, II, B.L. Nelson and D. M. Nicol, *Discrete-Event System Simulation*, Prentice Hall, New Jersey, 2000.

Fujimoto, Richard M., *Parallel and Distributed Simulation Systems*, John Wiley, Hardcover, Published February 2000

Wood, K. R., Turner, S. J., *A Generalized Carrier-Null Method for Conservative Parallel Simulation*, Proceedings of the 8th Workshop on Parallel and Distributed Simulation (PADS94), SCS, July 1994, pp. 50-57.

Voon-Yee Vee and Wen-Jing Hsu. *Parallel Discrete Event Simulation: A Survey*. Technical report, Centre for Advanced Information Systems, Nanyang Technological University, Singapore, August 1999.