

# The WarpIV Kernel

Jeffrey S. Steinman, Ph.D.  
President & CEO WarpIV Technologies, Inc.

5230 Carroll Canyon Road, Suite 306  
San Diego, CA 92121  
steinman@warpiv.com  
www.warpiv.com  
(858) 531-0643

A Short Position Paper prepared for the Workshop on Manycore Computing  
International Conference on Supercomputing 07

## Abstract

This short position paper discusses some of the techniques that have been used to synchronize asynchronous multicore applications operating in parallel and distributed environments. While these techniques have been developed primarily to support parallel discrete event simulation, a much broader application domain exists. The WarpIV Kernel was developed to support a broad spectrum of parallel and distributed applications requiring asynchronous logical-time scheduling between activities occurring on different processors of a parallel computing system or across machines connected in a distributed network.

## Background

Research and development in High Performance Computing (HPC) technologies began to take off in the mid to late 1980's. Defense programs such as the Strategic Defense Initiative (SDI) faced enormous computationally complex problems involving challenges such as data/tracking fusion of large numbers of missiles and decoys, and battle management to coordinate the defensive response. During this time, SDI funded several HPC hardware and software programs. Other areas of research and development applying HPC resources to grand challenge types of problems also began to emerge in virtually all areas of computational science.

As parallel processing began to mature, one area of research and development emerged known as *Parallel Discrete Event Simulation* (PDES). The basic assertion was that it ought to be possible to execute sequential discrete event simulations in parallel and distributed environments without changing the models. Not only should the parallel and distributed simulation obtain speedup when running on multiple processors, but it should also obtain exactly the same results as the sequential version.

This problem is not easy to solve in general because sequential discrete event simulations impose no constraints on how models interact. Parallel discrete event simulations must first distribute instances of the models, known as *simulation objects*, to different processors so that their combined workloads are roughly equal. Once the simulation objects are distributed and

initialized, they are then permitted to schedule/process time-tagged events for themselves or any other simulation objects that have been instantiated in the simulation.

An event can be thought of as a time-tagged processing activity that occurs on a simulation object. New events may be generated during the processing of an event. To maintain causality, events are never permitted to schedule new events with time tags less than the time of the current event. To maintain correctness, each simulation object must process its own set of events in ascending time order. This is where the synchronization challenges occur. In the worst case, the simulation object with the earliest time tag in the simulation could schedule an event for any other object at its current event time, which means that it may not be safe for any simulation object to process its next event. This condition can serialize the event processing of the entire simulation, resulting in poor CPU utilization and overall performance.

*The fundamental challenge of parallel discrete event simulation is to maintain causality and correctness while also achieving parallel speedup.*

## **Technical Approaches**

Several approaches to synchronize parallel and distributed simulations have been investigated over the years. Primarily, they fall into two categories. *Conservative* approaches safely process events only when it can be guaranteed that they will not receive messages from other simulation objects with earlier time tags. To achieve parallel performance, conservative strategies always place restrictions on how simulation objects may interact. For example, conservative strategies might (1) limit which simulation objects can interact with which other simulation objects, (2) require that all interactions from one simulation object to another simulation object occur in a time-ascending manner, and (3) restrict how tightly in time simulation objects are permitted to interact. Unfortunately, these constraints potentially affect the fidelity of the simulation results. Furthermore, even with such constraints, parallel performance may still be difficult to achieve due to uneven workloads, large synchronization costs, and significant message-passing overheads compared to event processing times.

A more aggressive approach to synchronizing parallel discrete event simulations is to process events *optimistically* and then use rollback techniques to undo events when they are accidentally processed out of order. So, a straggler event, for a particular simulation object, must roll back all of the events with larger time tags that were aggressively processed by that simulation object. It is important to note that events can basically do two things when they are processed. First, they can modify the state of their corresponding simulation object. Second, they can schedule new events.

So, rolling back an event means that (1) all state changes must be undone, and (2) all events that were scheduled must be retracted. Optimistic simulations must provide state-saving mechanisms to support rollbacks. Messages generated and released by the event to schedule new events must be retracted using what are known as *antimessages* to cancel those events. It is possible for antimessages to be received late, meaning that the events they were supposed to cancel were already processed. Further cascading rollbacks are produced when this occurs. These cascading

rollbacks can become unstable and wreak havoc if flow control techniques are not applied to limit event-processing optimism and message-sending risks.

## **SPEEDES and the WarpIV Kernel**

The Synchronous Parallel Environment for Emulation and Discrete-Event Simulation (SPEEDES) framework was created in 1990 to address these fundamental challenges. SPEEDES provided several critical flow control mechanisms that were necessary to stabilize optimistic simulations without sacrificing performance. Over the years, SPEEDES has been used on a number of large mainstream DoD simulation programs. Its technologies have been widely published in the literature. As a government-owned software system, the U.S. government provides all licensing of SPEEDES.

The WarpIV Kernel was developed by industry in 2001 as the next generation replacement for SPEEDES. Lessons learned over the previous 11 years were applied to the WarpIV Kernel. It incorporated all new communications, event management, and time synchronization algorithms with improved flow control techniques to provide stability for optimistic simulations. In addition, its modeling framework was expanded and simplified to make it easier for users to apply PDES technologies to their simulation problems. The WarpIV Kernel is freely available as an open source software system for non-commercial use. In addition to being freely available to industry, academia, and research institutes, its usage and distribution is completely unrestricted for government programs. The WarpIV Kernel was approved by the Department of Commerce for export licensing under EAR-99.

## **Standards: The OpenSSA and OSAMS**

The WarpIV Kernel is the first implementation of two standards that have been proposed as study groups within the Simulation Interoperability Standards Organization (SISO). The first standard is known as the *Open Standard Simulation Architecture* (OpenSSA). The OpenSSA provides a layered architecture for modeling and simulation that modularizes the technologies necessary to support simulations executing on machines ranging from laptops to supercomputers.

A second proposed standard, known as the *Open System Architecture for Modeling and Simulation* (OSAMS) is a subset of the OpenSSA and focuses on those modeling interfaces necessary to provide interoperability between plug-and-play model components. OSAMS focuses on promoting model interoperability, while the OpenSSA focuses on the technology to support parallel and distributed simulation.

## **Summary and Conclusions**

The WarpIV Kernel can support a wide variety of applications beyond parallel discrete event simulation. It solves some of the most challenging synchronization problems faced by parallel and distributed computing applications. Such applications might include (1) data mining, (2) real-time scheduling, (3) estimation and prediction, (4) optimization, (5) parallel applications, and (6) entertainment and gaming. More information on the WarpIV Kernel can be found at the following site: <http://www.warpiv.com>.