Technologies for Large-Scale Numerical Simulation

Phase I Final Report

PREPARED BY:
WarpIV Technologies, Inc.
5230 Carroll Canyon Road, Suite 306
San Diego, CA 92121
Phone (858) 605-1646
Fax (858) 605-5927
Email: jeffrey.steinman@warpiv.com
Home Page: www.warpiv.com

Contract Number: NSSC20C057080
Final Scientific and Technical Report
February 21, 2021

Project Summary

This project demonstrates the use of the *WarpIV Kernel* Parallel Discrete-Event Simulation (PDES) Engine to support a variety of High-Performance Computing (HPC) simulation applications for NASA. The WarpIV Kernel brings a unique HPC capability to the NASA simulation community because it provides total flexibility concerning how asynchronous computations between distributed models are processed in simulated time. Rather than time-stepping, events occur at their natural times, which supports a (a) far more efficient computational paradigm and (b) higher fidelity mechanism to represent complex models. The WarpIV Kernel is application agnostic and can be used to support any simulation application, including mixed modes involving NASA mission models integrated with computationally intense planetary science models. To demonstrate the general capabilities of the WarpIV Kernel for addressing NASA HPC simulation problems, four distinct scenarios are explored in Phase I: (1) Simulation of planetary rings (Saturn) as an n-body gravitational problem, (2) large-scale space debris with collision prediction for satellites and launched rockets, (3) computation of gravitational fields for ellipsoidal planets, and (4) Radio Frequency (RF) propagation from multiple moving transmitting sources and receivers (e.g., modeling spacecraft signal reception by the Deep Space Network in noisy RF environments). A major goal of this effort is to provide a framework that allows non-HPC software engineers to efficiently develop and test their simulations on multicore laptops or desktops and then move to HPC machines (with no changes to code) as their problem size scales up. In this way, the WarpIV Kernel provides the path for many more simulation users to migrate to HPC platforms using the world's most advanced simulation technology.

Table of Contents

TABLE OF CONTENTS **2**

IDENTIFICATION AND SIGNIFICANCE OF THE PROPOSED INNOVATION **4**

TECHNOLOGY LAYERS OF THE WARPIV KERNEL 5

TECHNICAL OBJECTIVES **11**

WORK PLAN AND ACCOMPLISHMENTS **13**

FIRST PROTOTYPE: MODELING PLANETARY RINGS AS AN N-BODY GRAVITATIONAL SYSTEM 13
Background *13*
Implementation *16*
Performance *21*
SECOND PROTOTYPE: ELLIPSOIDAL GRAVITY CALCULATION 25
Background *25*
Implementation *26*
Performance *26*
THIRD PROTOTYPE: SPACE DEBRIS 27
Background *27*
Implementation *29*
Performance *31*
FOURTH PROTOTYPE: ELECTROMAGNETIC RADIO FREQUENCY PROPAGATION 34
Background *34*
Implementation *35*
Performance *35*

RELATED R/R&D **37**

PATH FOR PHASE II AND PHASE III **38**

COMMERCIALIZATION FOR NASA PURPOSES AND THE DEPARTMENT OF DEFENSE **38**

WarpIV Technologies, Inc. Programs and Sponsors *38*
Potential Users of this Proposed Technology *38*
NASA Users *39*
General Goals and Guiding Principles of WarpIV Technologies, Inc. *40*

SUMMARY AND CONCLUSIONS **41**

KEY PERSONNEL **41**

DR. JEFFREY S. STEINMAN (PRINCIPAL INVESTIGATOR) 41
Education *41*
Biography *41*
MR. CRAIG N. LAMMERS 42
MS. WENDY L. STEINMAN 42

APPENDIX A: HIGH SPEED COMMUNICATIONS **44**

Startup and Shutdown *46*
Distributed Clock Synchronization *46*
Miscellaneous Services *46*
Global Synchronization and Reductions *46*
Global Reductions *46*
Synchronized Data Distribution Mechanisms *47*
Asynchronous Message Passing *47*
Coordinated Message Passing *47*

Object Request Broker (ORB) services *47*
Recent Performance Benchmarks *47*

APPENDIX B: DATA LOGGING **49**

APPENDIX C: FEDERATING LEGACY SIMULATIONS **52**

OVERVIEW OF THE HIGH-LEVEL ARCHITECTURE (HLA) 52
HIGH-PERFORMANCE COMPUTING RUN TIME INFRASTRUCTURE (HPC-RTI) 52
LOCAL RTI COMPONENT PROXY (LRCPROXY) FRAMEWORK 53

APPENDIX D: PRELIMINARY PERFORMANCE RESULTS ON 64-CORE AMD RYZEN THREADRIPPER **56**

Identification and Significance of the Proposed Innovation

In Phase I of this SBIR effort, WarpIV Technologies, Inc. successfully demonstrated the use of the WarpIV Kernel parallel discrete-event simulation engine and its extensive collection of high-performance software tools to address the diverse needs of the NASA high-performance computing community. The WarpIV Kernel contains more than 825,000 lines of cross-platform[1] C++ code, is free and open source for non-commercial use, and runs on all mainstream hardware, software, and networked computing systems ranging from multicore laptops to the world's largest supercomputers. All United States government agencies have an implicit software license with unrestricted usage for the software described in this final report. The WarpIV Kernel can also be exported to friendly, non-sanctioned, and non-embargoed countries without requiring an export license,[2] which means that it can be shared with foreign countries collaborating with NASA on joint science or space mission efforts. There are no comprehensive simulation frameworks (in comparison to the WarpIV Kernel) that provide similar advanced parallel discrete-event simulation capabilities within the (a) commercial sector, (b) research laboratories, (c) government agencies, or (d) academic institutions.[3] The WarpIV Kernel is the state-of-the-art for scalable, composable, and intelligent (i.e., support for Artificial Intelligence) general-purpose (i.e., application neutral) modeling & simulation infrastructure. The commercial sector cannot compete with this technology because (a) of its very specialized subject matter and required expertise, and (b) the WarpIV Kernel is also free and open.

The WarpIV Kernel, and its predecessor SPEEDES,[4] has been widely used by the Department of Defense on several extremely large mainstream programs involving all of the defense agencies over the past 25 years, including the High Performance Computing Modernization Program (HPCMP), Defense Modeling & Simulation Coordination Office (DMSCO), Missile Defense Agency (MDA), Air Force, Army Research Laboratory (ARL), Navy, several very large cross-domain joint programs (JSIMS, JMASS) involving all branches of the armed forces, the Defense Intelligence Agency (DIA), Defense Advanced Research Projects Agency (DARPA), Office of the Secretary of Defense (OSD), Chemical Biological Radiological Nuclear Defense (CBRND), etc.

The WarpIV Kernel is a layered open-system architecture as shown in Figure 1. Each layer in the architecture is permitted to utilize (i.e., depend on) services from lower layers, but not the upper layers. With this very clean hierarchical dependency structure, the architecture layers can be thought of as composable libraries that are ready for standardization by communities of interest such as the international Simulation Interoperability Standards Organization (SISO).[5] With the goal of future standardization, any layer could be swapped out and replaced by a different (perhaps optimized for a particular hardware/network configuration) implementation developed by any contributing organization. The plug-and-play composability design strategy of the WarpIV Kernel promotes

---

[1] The WarpIV Kernel runs on Linux, Windows, and Mac OS X using all mainstream C++ compilers (GCC, LLVM, Visual Studio).

[2] The United States Department of Commerce has categorized the WarpIV Kernel as: Export Administration Regulation EAR-99. The State and Defense Departments have not imposed International Traffic in Arms Regulations (ITAR) restrictions on export. Even though the WarpIV Kernel provides state-of-the-art high-performance computing M&S technology, it does not contain any classified algorithms or data. All technologies of the WarpIV Kernel are openly published in international forums.

[3] We are often asked who our competitors are and if other M&S technology providers are catching up to the WarpIV Kernel's capabilities. In reality, we find that the WarpIV Kernel each year further distances itself from all competitors because no other organization is building a comprehensive simulation framework based on state-of-the-art optimistic time management techniques with adaptive flow control, which is critical for providing execution stability. So, other frameworks claiming to support high-performance M&S are based on inferior technology that cannot compete with WarpIV.

[4] Synchronous Parallel Environment for Emulation and Discrete-Event Simulation (SPEEDES). Dr. Steinman (principal investigator for this effort and the creator of the WarpIV Kernel) developed the Synchronous Parallel Environment for Emulation and Discrete-Event Simulation (SPEEDES) while working as a member of the technical staff at the Jet Propulsion Laboratory in 1990. SPEEDES was awarded five NASA patents for its breakthrough technologies and was eventually used by several very large defense-related M&S programs. The WarpIV Kernel provides a significant upgrade from SPEEDES with much better run-time performance, tools, and extensive software capabilities

[5] The WarpIV Kernel is the open source reference implementation of the Open Unified Technical Framework (OpenUTF), which is defined by three architectures: (1) Open Modeling & Simulation Architecture (OpenMSA), which focuses on the technology of scalable high-performance simulation, (2) Open Simulation Architecture for Modeling & Simulation (OSAMS), which focuses on composable modeling constructs designed to maximize interoperability and reuse while minimizing lines of code to represent complex systems, and (3) Open Cognitive Architecture Framework (OpenCAF), which is designed to support intelligent thought-triggering, goals, and dynamic behaviors for carrying out goals. The OpenUTF is being prepared for future standardization for use by the Department of Defense simulation community.

composability both for (1) technology insertion and (2) plug-and-play models potentially developed by multiple organizations to facilitate Science & Technology collaboration.

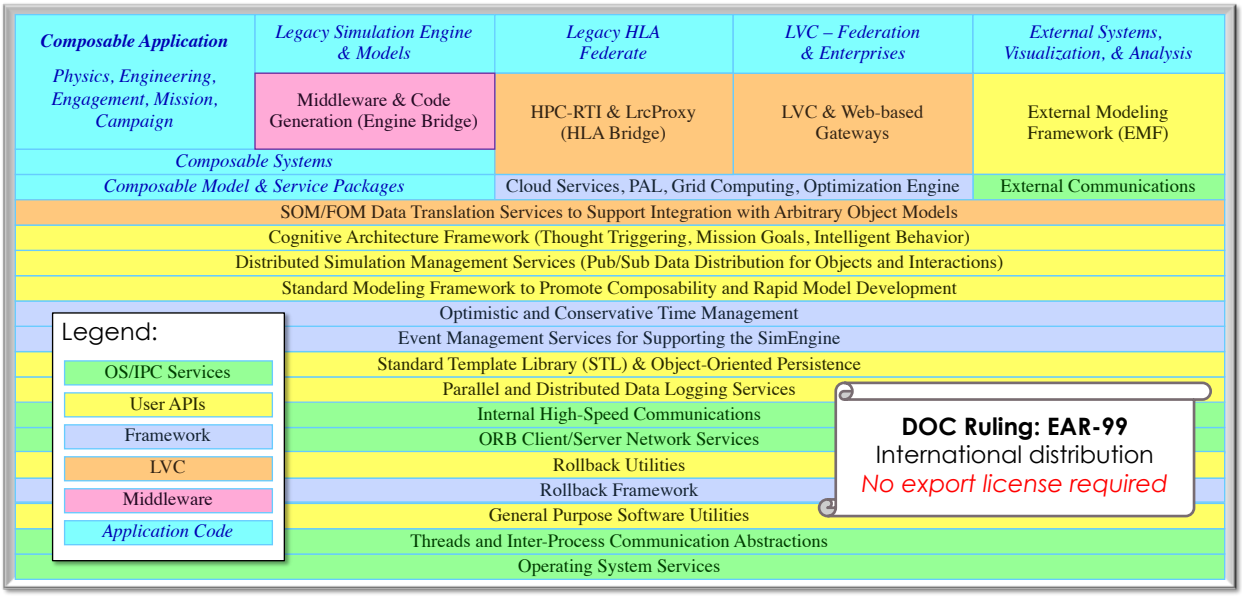


Figure 1: Layered architecture of the WarpIV Kernel.

In practice, this means that the WarpIV Kernel provides an open system for both technologists (e.g., high-performance computing technology experts, tool providers, etc.) and scientists (e.g., astronomers, physicists, engineers, space mission SMEs, etc.)[6] to contribute their expertise to the parallel and distributed community, which includes NASA, DoD, Intelligence Community, Industry, and Academia. From a user's perspective, the WarpIV Kernel provides a powerful multicore simulation framework where models can be developed and tested on laptops or desktops for a wide range of NASA mission and science applications, and then brought to large supercomputers when massive scalability is needed. A brief description of the technology layers is provided below.

Technology layers of the WarpIV Kernel

Operating System Services: provides an abstraction to low-level system services that might be implemented in various ways across different operating systems. The abstraction layer is important for high-performance computing applications to automate cross-platform execution. As a note, the WarpIV Kernel is designed to support mixed 32-bit and 64-bit machines potentially running different operating systems on different platforms.

Threads and Inter-Process Communication Abstractions: provides services for supporting different ways to utilize processing cores within a single-threaded or multithreaded process and to allow different processes executing on the same machine to mutually interact through signals and shared memory.[7]

General Purpose Software Utilities: provides a very large assortment of software utilities including a variety of math algorithms, curve fitting tools, multivariable optimization techniques, motion representation, coordinate system transformations, matrix and vector math utilities with operator overloading to easily implement complex matrix equations in a single line of code, signal processing algorithms such as Fast Fourier Transforms (FFTs),

[6] Science & Technology (S&T) both perform Research & Development (R&D) for Modeling & Simulation (M&S), but in different ways. Technologists develop/optimize high speed communications, advanced time management algorithms, methods for logging large volumes of data, modeling constructs that promote composability and ease of use, visualization and analysis tools, optimization engines and algorithms, etc. Scientists use these tools to develop, validate, use, and analyze the results of models. In the analogy of astronomy, technologists are telescope makers and scientists are astronomers who use those telescopes to do science. Technology always enables better science.

[7] The WarpIV Kernel runs multi-process/single-core. So, IPC is supported by messages through shared memory and network communications.

special-purpose data structures such as Hierarchical Grids (discussed later in this report), optimized container classes, data parsers for reading files of various formats (including XML), error handling, random number generation with support for a large assortment of statistical distributions, dynamic routing within reconfigurable connected graphs that can dynamically change structure and transition costs, statistical algebra with operator overloading to perform mathematical operations on statistical distributions rather than single-value variables, string-based interpreter to evaluate mathematical/logical expressions, various mechanisms to optimize the use of dynamic memory (including memory corruption checks that can be turned on during testing), AI tools such as neural networks and genetic algorithms, etc. The WarpIV Kernel also provides a cross-platform GUI widget set that supports Linux, Windows, and Mac OS X operating systems using each OS's native look and feel widgets.

Rollback Framework: supports the ability for arbitrary operations to rollback, rollforward, commit, and uncommit. These capabilities are essential for supporting optimistic time management within the discrete-event simulation engine and can also be used by any application or graphical interface to generically support undo and/or redo functions. The rollback framework is highly optimized in terms of reducing processing overheads and lowering memory consumption to minimize state-saving overheads on performance. In most scientific applications, the incremental state-saving overheads to support rollbacks are completely negligible. An important goal of the rollback framework is to automate state rollback transparently.

Rollback Utilities: provides a very large assortment of data types, data structures, and operations that are rollbackable. Examples include primitive data types (e.g., integers, doubles, logicals, and strings) with operator overloading that makes state-saving and rollbacks transparent, dynamic memory allocation/deallocation (e.g., new and delete or malloc and free, and memory management with highly optimized object factories to reuse memory whenever possible), container classes (e.g., trees, lists, priority queues, dynamic arrays, hash tables, etc.), I/O for screen outputs or writing to files, releasing network messages to external systems, etc. The very large collection of rollback utilities provided by the WarpIV Kernel are extensible, which means that new capabilities can be readily added to the existing collection of rollback utilities when needed.

ORB Client/Server Network Services: provides an Object Request Broker (ORB) framework (similar to the standardized Common Object Request Broker Architecture, CORBA, but with much lower overheads) for building distributed client-server systems. Clients can invoke remote methods on server objects executing within server processes launched across machines in a network environment. Methods can be one-way (asynchronous), two-way (synchronous with return argument values), or functions (synchronous with a return value and potential return arguments). The Client/Server framework uses advanced double buffering techniques for reads and writes to maximize message-handling throughput when networks are congested or when client input buffers are full. The double buffering mechanism also eliminates potential server deadlocks that could occur in other network-based client/server frameworks.

Internal High-Speed Communications: provides extremely fast, scalable, and memory-efficient communications for parallel computing applications by integrating shared memory with highly optimized network protocols.[8] The abstract interface for the high-speed communications layer in the WarpIV Kernel architecture is designed to support all backplane communication technologies[9] without compromising on performance. Benchmarks show the native WarpIV High-Speed Communications framework operating on multicore systems outperforming MPI[10] by a significant margin for most operations such as synchronizations, fuzzy barriers, global reductions, asynchronous messages, and synchronous messages. One of the key features of the high-speed communications is its reliable

[8] Steinman Jeff, 2007, "WarpIV Kernel: High Speed Communications." In proceedings of the *Fall 2007 Simulation Interoperability Workshop*, 07F-SIW-045.

[9] The current implementation integrates standard network protocols such as TCP/IP with shared memory. Support for higher performance communication protocols and/or high-speed backplanes (Myrinet or InfiniBand) could be easily supported in the future.

[10] The Message Passing Interface (MPI) is the widely used standard for message passing by the high-performance computing community. The WarpIV Kernel High-Speed Communications framework provides a second version of its interface based on MPI (funding to develop this capability came from the HPCMP-PETTT program). Benchmark performance tests using Native and MPI implementations ran the exact same test code on a 64-core shared memory machine at SPAWAR Systems Center, San Diego, to produce a fair comparison. Some of the results of this study are provided in this report.

destination-based multicasting techniques that are integrated within the WarpIV publish/subscribe data distribution mechanisms. No other parallel and distributed optimistic simulation framework provides this essential multicast capability for logical-time event scheduling with support for rollbacks and flow control. Recent upgrades have been made to optimize performance in cloud and/or large HPC cluster environments involving many machines. The High-Speed Communications can be used to support most general-purpose HPC applications with a much-easier-to-use interface than MPI.[11] The WarpIV High-Speed Communications also provides wall clock synchronization, which can be extremely important when measuring message latencies or for distributed applications requiring accurate wall times. Another very important service provided is Fuzzy Barriers, which allow nodes within parallel applications to indicate that they are ready to synchronize (i.e., they enter the fuzzy barrier), but without blocking. Nodes that have entered the Fuzzy Barrier continue performing computations until all nodes have entered the fuzzy barrier, which enables the exit fuzzy barrier condition.

Parallel and Distributed Data Logging Services: provides highly optimized bit-compression algorithms to significantly lower data collection overheads while dramatically reducing log file sizes. This is critical for large parallel simulations that create "Big Data" outputs. A missile defense simulation recently demonstrated log file size reduction (compared to a previous unoptimized text-based capability) by a factor of 23 with virtually no noticeable performance degradation.[12] Tools are provided by the WarpIV Kernel to read and display log files as text, rapidly compare two log files for differences, read back and process data captured in unstructured log files, and capture detailed event traces for all events processed within the WarpIV Kernel during a parallel execution. The WarpIV Kernel provides graphical tools for visualizing scenarios and analyzing results based on generated bit-compressed log files. Log files are organized as a sequence of time-merged records created by parallel and/or distributed processes where time can represent the wall clock or logical time in run-as-fast-as-possible simulation executions. Multiple structured and/or unstructured data values are typically logged in each record. A special highly optimized construct (Named Collections) is provided to log structured data. The data logger learns these named collection structures on the fly, which makes it very easy to use by applications. Named collections track structured data by one or (optionally) two Ids that are used to super-compress data types that are smooth and continuous. In addition to bit-compressed data logging, the WarpIV Kernel also provides a portable parallel and distributed profiling service that instruments computations (in great detail) performed by methods and functions. This capability is extremely important for supporting parallel and distributed code optimization.

Standard Template Library (STL) & Object-Oriented Persistence: provides a rollbackable and persistent version of STL (commonly used by C++ programs) along with an object-oriented persistence framework. Persistence allows (a) fragmented objects that are spread across memory and (b) pointers (within these objects that connects the fragmented objects into complex structures) to be serialized into a buffer that can be used to later reconstruct the objects and pointers. Note that during reconstruction, objects are allocated in different memory addresses. Persistence adjusts all of the embedded pointers within these fragmented objects to correctly reference the new memory locations. Persistence provides a critical (future) capability for migrating complex objects spread across memory to different processors. Persistence can also be used to support checkpointing and restarting simulations from time-tagged checkpoint files.[13]

Event Management for Supporting the SimEngine: provides low-level simulation engine infrastructure for representing abstract time with elaborate tie-breaking fields to guarantee time uniqueness across all events executing in parallel, creating simulation objects with internal models that are distributed across processors, managing time-tagged events in scalable high-speed event queues, handling messages (unicast, multicast, and

[11] Steinman Jeff, Valinski Maria, and Koop Matthew, 2012. "Native and MPI Implementations of the Open Unified Technical Framework High Speed Communications Layer (OpenUTF-HSC)." In proceedings of the Fall 2012 Simulation Interoperability Workshop, 12F-SIW-026.

[12] Technical report delivered to the Missile Defense Agency and the Defense Intelligence Agency can be made available on request: "Data Compression, Intelligent Filtering, and Data Logging in the WarpIV Kernel."

[13] Steinman Jeff, 2003. "The SPEEDES Persistence Framework and the Standard Simulation Architecture." In proceedings of the Parallel and Distributed Simulation (PADS03) conference. (Note that the persistence framework used by SPEEDES was ported from the WarpIV Kernel). The WarpIV Persistence framework (at the time, used by SPEEDES) was used to successfully support checkpoint/restart for the multi-billion-dollar Joint Simulation System (JSIMS) program. The WarpIV Kernel does not currently support checkpoint/restart capabilities, but this capability could be provided and automated in the future.

broadcast) between nodes, and supporting rollbacks with dynamic flow control to ensure stability during parallel execution. The event management layer is designed to support any time management algorithm that might be developed in the future with scalable data structures that can support arbitrarily large simulation scenarios.

Optimistic and Conservative Time Management: provides a collection of time synchronization algorithms to ensure that events are always processed by models in their correct causal time order and that identical simulation runs are always repeatable.[14] The algorithms in SPEEDES were originally developed by Dr. Steinman (the Principal Investigator of this effort) while working at the Jet Propulsion Laboratory in the early 1990's where the work received several best-paper recognitions and was awarded five NASA patents.[15] The WarpIV Kernel goes well beyond these original SPEEDES algorithms and NASA patents (which have now expired) with significant performance improvements. For example, one of the advanced time management algorithms allows simulations to execute in five dimensions, which permits the evaluation of multiple time branches within a single execution. Five-dimensional simulation[16] allows a single simulation execution to run the equivalent of almost an infinite number of stochastic Monte Carlo runs. All computations common to dynamically spawned branches are automatically shared, which provides revolutionary performance.[17] Five-dimensional simulation could be used in the future to support NASA spacecraft testing[18] to exhaustively explore all possible states (including rare ones, e.g., Black Swan Events) of the spacecraft that might arise during operation.

Standard Modeling Framework to Promote Composability and Rapid Model Development: provides extremely powerful modeling constructs that minimize the number of lines of code necessary for complex models to interoperate (e.g., share data and invoke/schedule each other's methods) while providing complete encapsulation between composable models.[19] This layer also provides advanced *process model* constructs that allow event methods to wait for a period of time before waking up, be interrupted, wake up when a specified condition becomes true, and/or when a requested resource becomes available. The process model also allows models to invoke two-way methods in logical time on remote objects residing on other machines without deadlocking.[20] A goal of the Standard Modeling Framework is to provide a natural C++ programming interface to model developers.

Distributed Simulation Management Services (Pub/Sub Data Distribution for Objects and Interactions): provides a publish/subscribe data distribution mechanism for distributing state information (known as Federation Objects, FOs) and events. This capability, which also includes support for basic filtering, is critical for allowing distributed entities to share (publish) state information with other (subscribing) entities. Updated attributes in published

[14] Steinman Jeff, 2005, "The WarpIV Simulation Kernel." In proceedings of the *2005 Principles of Advanced and Distributed Simulation (PADS) conference.*

[15] The fundamental challenge of parallel and distributed simulation is to maintain causality (i.e., that events are processed in their correct ascending time order) while allowing models operating on different processors to freely interact with one another with maximum concurrency, yet without imposing any constraints such as (1) how tightly models on different processors can interact in time, (2) limiting which models can interact with which other models based on a graph interconnection topology, and/or (3) constraining time-ordering of messages sent/received. The WarpIV Kernel imposes none of these constraints, which means that it can support all types of simulation problems at the highest level of fidelity while maintaining scalable performance.

[16] Steinman Jeff, Lammers Craig, Valinski Maria, and Roth Karen, "Simulating Parallel Overlapping Universes in the Fifth Dimension with HyperWarpSpeed Implemented in the WarpIV Kernel." In proceedings of the Spring 2008 Simulation Interoperability Workshop, 08S-SIW-025.

[17] Steinman Jeff, Lammers Craig, Valinski Maria, and Steinman Wendy, 2011. "Scalability of Parallel and Distributed Modeling & Simulation Architectures." In proceedings of the *Spring 2011 Simulation Interoperability Workshop*, 11S-SIW-065.

[18] During a conference presentation by WarpIV Technologies on five-dimensional simulation, several NASA M&S experts commented how this technology could potentially help NASA mission scientists do a better job with exhaustive testing, especially for cases when rare anomalies might occur that could be missed in normal Monte Carlo stochastic testing.

[19] Standard Lines of Code (SLOC) for models developed in the WarpIV Kernel are often reduced by an order of magnitude (or more) when compared to similar implementations in other frameworks. For example, the Missile Defense Engagement Model (MDEM) developed by WarpIV Technologies, Inc. for MDA required only 1,500 lines of code to support the same functionality provided by other frameworks taking hundreds of thousands of lines of code. The code reduction in MDEM was due to (a) use of the modeling framework for supporting complex time-based data-sharing and interactions between models, (b) code generation provided by the WarpIV Model Editor, and (c) use of the large math and physics algorithms provided by the WarpIV software utilities. It should also be noted that MDEM ran orders of magnitude faster than comparable models developed by MDA.

[20] Steinman Jeff, 2013. "The Roadmap." In proceedings of the Spring 2013 Simulation Interoperability Workshop, 13S-SIW-047. This was a 50-page paper that laid out a strategy for transforming defense-related M&S towards a scalable, composable, and intelligent solution.

objects are automatically reflected to subscribers in repeatable logical time.[21] A small portion of this effort focused on providing scalable interest management services. Phase II, if funded will complete this important work. Scalable interest management allows subscribers to impose conditions on the published objects they receive, which if done properly, dramatically improves scalability and performance.[22]

Cognitive Architecture Framework (Thought Triggering, Management of Mission Goals, Dynamic Intelligent Behavior Based on Perception): provides a framework to support artificial intelligence constructs within models to represent a variety of thought types (e.g., rules, emotion, empirically trained, deep learning, multivariable optimization, game theory based on finding Nash Equilibriums, and embedded simulation to make decisions based on predictions during an event in the main simulation) , dynamic management of goals, and the series of tasks/behaviors that must be adaptively performed (based on perception) to complete each goal in the most optimal manner. The cognitive architecture framework could be used to support the testing of NASA command sequences before sending them to remote spacecraft to ensure safe operation.[23]

SOM/FOM Data Translation Services to Support Integration with Arbitrary Object Models: provides the ability to integrate with other systems that have a different data interface or a documented data model. The translation services are able to automatically provide two-way translation services from one set of objects (in a specified object model) to another (with a different object model). Translations are highly optimized based on attribute dependencies between objects from one object model to another. The Federation Object (FO) infrastructure also supports a user-defined C++ inheritance capability, which allows users to customize their own interfaces to FOs where translation services can be applied. What this means from a practical sense is that the WarpIV Kernel is designed to integrate with legacy systems having different data models or interfaces with minimal effort.

External Communications: provides basic services that allow external systems to efficiently interact with a simulation while it is running. Simple external communication services include pause, resume, kill the simulation, get the current time of the simulation, establish time barriers with the simulation to provide flow control and eliminate race conditions, and send/receive messages (events) between external systems and the simulation running in parallel on a supercomputer. A special tool is provided for merging parallel messages into a common window or graphical interface where they are displayed in logical-time or real-time message order. In this way, messages do not step on each other while being printed out.

External Modeling Framework (EMF): builds on the external communications framework to provide a powerful interface that allows remote applications to publish and subscribe to objects created in the simulation (executing in parallel) and to schedule/receive events with a simulation while it is running. All two-way messages are synchronized in abstract time (i.e., real or logical time). The EMF can also log data for offline playback. Its internal state management infrastructure uses rollback techniques that permits EMF applications to freely move forward and backwards in time. The EMF has been fully integrated with the bit-compressed data logging infrastructure to provide an ideal foundation for supporting live and playback visualization capabilities.[24]

---

[21] Steinman Jeff, Lammers Craig, Valinski Maria, and Steinman Wendy, 2010. “Scalable Publish and Subscribe Data Distribution.” In proceedings of the *Fall 2010 Simulation Interoperability Workshop*, 10F-SIW-027.

[22] The original interest management algorithms were developed (and patented by NASA) in SPEEDES to support parallel proximity detection for missile and air defense applications. Steinman Jeff and Wieland Fred, 1994. "Parallel Proximity Detection and the Distribution List Algorithm." In proceedings of the *1994 Parallel and Distributed Simulation (PADS) Conference*. Pages 3-11 (best paper award). The work was later expanded under an effort funded by the High-Performance Computing Modernization Program. Steinman Jeff, Tran Tuan, Burckhardt Jacob, Brutocao Jim, 1999. “Logically Correct Data Distribution Management in SPEEDES”, In proceedings of the *1999 Fall Simulation Interoperability Workshop*, Paper 99F-SIW-067. The work was then further expanded (but not fully optimized) for SPEEDES to support the billion-dollar JSIMS program. Lammers Craig, Steinman Jeff, 2004, “JSIMS CCSE Performance Analysis.” In proceedings of the *Fall 2004 Simulation Interoperability Workshop*, paper 04F-SIW-34.

[23] Steinman Jeff, Lammers Craig, Valinski Maria, 2009. “A Proposed Open Cognitive Architecture Framework.” In proceedings of the *2009 Winter Simulation Conference.*

[24] Steinman Jeff, Lammers Craig, Valinski Maria, and Steinman Wendy, 2012. “The OpenUTF External Modeling Framework (EMF).” In proceedings of the Spring 2012 Simulation Interoperability Workshop, 12S-SIW-034.

External Systems (Visualization and Analysis): WarpIV provides a 3D and 2D earth-based map tool (the Visualizer) that can be used to visualize simulation runs involving earth (or planetary) sciences. The Visualizer has been extended to support planetary visualization using high-resolution imagery (e.g., Saturn). The WarpIV Kernel also provides a general-purpose analysis tool (the Analyzer) that can be used to form a variety of mathematical plot types (time plots, scatter plots, histograms) of arbitrary logged data. The Analyzer supports general data filtering (i.e., support for human-guided data mining) and curve fitting algorithms that allow analysts to dig deep into logged data to study simulation outputs. The Analyzer has proven to be critical in verifying and validating models.

LVC and Web-based Gateways: provides the hooks for integrating with Live, Virtual, and Constructive distributed applications, including support for the High-Level Architecture (HLA), Distributed Interactive Simulation (DIS), and XML-based web standards.

Cloud Services, PAL, Grid Computing, Optimization Engine: provides services for managing common operations across machines within a cloud, rack-mounted server farm, or HPC cluster environment. The Parallel Application Launcher (PAL) launches parallel and distributed applications across multiple machines to support remote startups. Pre-commands (which execute before the parallel application starts) and Post-commands (which execute after the parallel application completes) are supported to assist parallel execution. The grid computing infrastructure allows taskers to farm expensive computations to workers in networked computing environments. The optimization engine intelligently guides distributed workers (e.g., simulations) towards an optimal solution using a variety of techniques such as genetic algorithms, regional bisection based on response surfaces and statistical rejection methods, particle swarm optimization, Latin Hypercubes, and random search.[25] Ongoing R&D funded by the Missile Defense Agency is exploring scalable high-performance Response Surface Methodologies (RSM) to support large-scale optimization involving up to a hundred tunable parameters.

HPC-RTI and LrcProxy (HLA Bridge): provides support for the High-Level Architecture (HLA) standard (with significant middleware extensions to simplify integration), which is widely used by the international community to integrate disjoint defense simulations into enterprised federations. The High-Performance Computing Run Time Infrastructure (HPC-RTI) in the WarpIV Kernel is designed to support very large numbers of simulations executing on supercomputer clusters. It could be used to parallelize existing science codes used by NASA or to simply integrate multiple simulations into a single parallel/distributed execution.[26] The Local RTI Component Proxy (LrcProxy) framework also implements the exact same HLA standard as the HPC-RTI and is designed to federate large numbers of interacting disjoint simulations on a single multicore machine with the goal of maximizing CPU utilization. These two run-time-swappable infrastructures in the WarpIV Kernel provide the underlying foundation for the Defense Intelligence Agency (DIA) Integrated Threat Analysis Simulation Environment (ITASE).[27] The ITASE program has high visibility within the defense community because they alone have the responsibility for providing a credible representation of adversarial threats. ITASE is currently being used to support real-world studies that are critical for the United States Department of Defense. The HPC-RTI and LrcProxy Framework are also being used by the DARPA SAFE-SiM program, which if successful will become a very large (billion dollar) program for the DoD. In addition, these two frameworks are being integrated with the Advanced Framework for Simulation Integration and Modeling (AFSIM), which is the Air Force's widely used simulation engine. Plans are underway to bring these two frameworks to the Missile Defense Agency (MDA) Objective Simulation Framework (OSF). Both of these frameworks have been enhanced to support native Federation Object interfaces, which dramatically simplifies integration efforts with legacy systems while unifying interoperability with models executing natively within the WarpIV Kernel. These capabilities for integrating/parallelizing legacy simulations will be available to NASA.

[25] Steinman Jeff, Steinman Wendy, Lammers Craig, and Valinski Maria, 2011. "Embedded Grid Computing with the Open Unified Technical Framework." In proceedings of the Fall 2011 Simulation Interoperability Workshop, 11F-SIW-038.

[26] Steinman Jeff, Lammers Craig, Valinski Maria, Steinman Wendy, Steinman Mitchel, Blank Gary, 2017. "The Open Unified Technical Framework High Performance Computing Run Time Infrastructure for the High-Level Architecture." To be published in a future conference.

[27] An early requirement for ITASE was to be able to support scalable and repeatable execution of a federation with as many as 10,000 federates. This would be like running an HPC application on 10,000 cores except that each application is a different software program. Other than the WarpIV Kernel HPC-RTI, no other HLA framework comes close to being capable of meeting this requirement.

Middleware & Code Generation (Engine Bridge): provides a model-editing GUI with code generation to reduce lines of user-written model code. The middleware layer also provides interface translations to port models from a legacy simulation engine to the WarpIV Kernel. Examples of porting models from legacy engines to WarpIV (or its predecessor SPEEDES) include MDA I-Sim (Missile Defense Agency International Simulation),[28] JMASS (Joint Modeling and Simulation System), WARSIM (Army simulation), EADTB (Extended Air Defense Test Bed), GTNETS (Georgia Tech. Network Simulator), etc. Proposed plans with DARPA are being discussed to port models developed for the Advanced Framework for Simulation Integration and Modeling (AFSIM) and the Threat Modeling Analysis Program (TMAP) into the WarpIV Kernel simulation engine. AFSIM is a sequential simulation framework that is widely used by the Air Force. It comes with thousands of models developed by multiple organizations. TMAP models are based on MATLAB/Simulink. WarpIV Technologies, Inc. is investigating the feasibility of porting these two high-profile systems to operate natively within the WarpIV Kernel.

Technical Objectives

WarpIV Technologies, Inc. met all of its primary technical objectives in Phase I, which focused on quickly prototyping, demonstrating, and benchmarking four types of simulation applications, each with heavy computational workloads executing on multicore machines.[29] These four diverse simulation applications were specifically chosen because of their representative nature towards NASA computationally intense problems. These applications were also chosen because they could be quickly developed (or extended from previous work) and demonstrated in-house by our technical staff without requiring additional subject matter expertise or the use of other HPC resources. The primary machines used in Phase I were (1) MacBook Pro with an 8-core I9 processor loaded with 64 GB RAM, and (2) 32-core AMD Ryzen Threadripper machine running Linux with 128 GB RAM. The goal for each of these simulation prototypes was to demonstrate (a) ease of implementation, (b) use of discrete-event techniques to significantly reduce computations to promote rapid execution,[30] (c) cross-platform execution, (d) plug-and-play model composability to promote interoperability and reuse, and (e) unbounded (i.e., in the spirit of grand challenge problems) scalable performance on machines ranging from multicore laptops to the world's largest supercomputers.[31] Phase II will port these four applications to much larger machines provided by the NASA HPC community where further benchmarks will be performed to determine scalability and performance. It is expected that these four WarpIV-based simulation applications will execute very efficiently on large supercomputers. In addition, Phase II will explore the use of parallel discrete-event technologies for computational fluid dynamics problems.

The work performed during this effort supports all of the specific objectives that were listed in the SBIR topic:

- **Decrease the barriers to entry for prospective supercomputing users.**
  ⇒ This is accomplished by providing a powerful and easy-to-use cross-platform simulation framework. Users can develop and test their models on their multicore laptops or desktops before scaling up to large supercomputers. WarpIV Technologies, Inc. has prepared extensive hands-on training materiel to help

[28] I-Sim was successfully ported to the WarpIV Kernel in 2014 as an R&D effort performed by WarpIV Technologies, Inc. but was put on hold due to lack of additional funds for continuation. It is now undergoing a new pathfinder effort to finalize this port and make the transition from SPEEDES to WarpIV permanent.

[29] The original proposal hoped to run benchmarks on a (to-be-purchased) AMD Ryzen Threadripper 64-core machine and the existing 264-core Proliant cluster (twenty-four, 12-core, rack-mountable machines connected with a high-speed, 48-port CISCO switch) located at the WarpIV Technologies, Inc. office in San Diego. However, due to the COVID-19 pandemic, which shifted the entire WarpIV staff to telecommuting (i.e., no work performed in the office), it was decided to wait until early 2021 to purchase the 64-core machine and to support the Proliant cluster. The 64-core machine arrived just as this report was being finalized. Preliminary 64-core results are provided in one of the appendices. WarpIV Technologies, Inc. currently owns a 32-core AMD machine with 128 GB RAM that is used to support the primary set of benchmarks performed in this effort. In addition, benchmarks are also performed on a high-end 8-core MacBook Pro configured with 64 GB RAM and an 8-TB flash hard drive.

[30] The philosophy of discrete-event simulation (in contrast to time stepped simulation) is to only perform computations when necessary. When used effectively, the discrete-event approach can dramatically reduce run times while also providing more accurate results than time stepped approaches. The benefits of this strategy will be shown in all of the demonstrations produced in Phase I.

[31] This effort will demonstrate scalability in the spirit of grand challenge HPC problems where performance scales in an unlimited manner as the size of the problem gets larger. Such problems always benefit from scalable execution on larger machines to produce better fidelity.

model developers get started. The training material includes software lessons with a text-based guide and video instruction.

- **Minimize the supercomputer user's total time-to-solution (e.g., time to discover, understand, predict, or design)**
  ⇒ The comprehensive set of tools and constructs provided by the WarpIV Kernel significantly reduce the time and effort to build complex models. It is not uncommon to see a lines-of-code reduction factor by an order of magnitude (or more) when compared to similar models developed with other frameworks. In almost all cases, the more complex the model, the greater the code reduction factor becomes. So, models can be developed, tested, and validated faster, better, and cheaper. In addition, the model composability mechanisms in the WarpIV Kernel promote interoperability and reuse. This means that as more and more models are developed by NASA scientists, solutions can be quickly assembled from existing model repositories shared by the scientific community.

- **Increase the achievable scale and complexity of computational analysis, data ingest, and data communications**
  ⇒ The WarpIV Kernel inherently provides unlimited scalability with optimized message passing internals, so the only limitation to performance is the size of the problem being simulated and how it is implemented to maximize parallelism. The built-in critical path analysis feature of the WarpIV Kernel provides guidance to model developers for the inherent parallelism of the system being simulated. The bit-compressed parallel and distributed data logging features of the WarpIV Kernel support Big Data output in an intelligent and scalable manner. File sizes are dramatically reduced to simplify management of large sets of data output. The bit-compressed data logger also provides a variety of tools and easy-to-use interfaces for reading back log files and accessing both structured and unstructured data. In addition, the WarpIV Kernel provides 3D/2D planetary visualization and a mathematical analysis tool for generating plots with built-in filtering and curve fitting tools to streamline analysis.

- **Reduce the cost of providing a given level of supercomputing performance for NASA applications**
  ⇒ The capabilities of the WarpIV Kernel dramatically reduce the cost of developing complex simulations executing on HPC platforms. Developers focus on writing code for their models without worrying about messages, rollbacks, data collection, etc. Models can be developed locally on multicore laptops or desktops before scaling up to large supercomputers. Critical path analysis feedback to users accurately predicts the level of performance that can be expected for a given problem size. The critical path analysis lets users know if executing their problem on a supercomputer of any specified size is feasible before any porting work is done. In addition, the WarpIV Kernel provides a detailed, built-in, software profiling service that helps developers understand, analyze, and optimize computation overheads in functions and/or methods. Profiling is critical in optimizing the performance of HPC applications. Because the WarpIV Kernel is self-contained with no third-party software dependencies, and because it supports all mainstream operating systems and compilers, porting efforts to large supercomputers are automatic.

- **Enhance the efficiency and effectiveness of NASA's supercomputing operations and services**
  ⇒ The WarpIV Kernel discrete-event methodology provides unconstrained operation concerning how models evolve, share data, and mutually interact in simulated time. This unconstrained capability means that models can freely interact with any other model at any time scale, which frees models from operating in artificially fixed time steps. The result of the optimistic discrete-event methodology is both (a) faster execution and (b) more accurate simulation results. This was demonstrated during this Phase I effort. In addition, the WarpIV Kernel provides two state-of-the-art capabilities for federating (or self-federating) legacy simulation applications with the goal of obtaining parallel speedup. Federated simulations[32] automatically interoperate with native WarpIV models, which provides the perfect strategy for combining existing capabilities with next-generation high-performance computing capabilities.

[32] The WarpIV Kernel provides two infrastructures for supporting the standards-based High-Level Architecture (HLA) that is widely used by the Department of Defense and the international community for integrating legacy simulations into an enterprised federation.

The four grand-challenge simulation prototypes that were prototyped and demonstrated in Phase I were:

1. Modeling planetary rings (e.g., the rings of Saturn) as an n-body gravitational system
2. Modeling non-central planetary gravitational forces (e.g., ellipsoidal planets such as Saturn)
3. Modeling large-scale space debris collisions (closest approach) with rocket launches and orbiting satellites
4. Modeling RF signals between transmitters and receivers such as spacecraft, satellites, and ground stations

The design, implementation, and performance results of these four prototypes will be described in more detail in the Work Plan and Accomplishments section of this report. In addition, separate benchmarks of the WarpIV Kernel High Speed Communications are provided in an appendix. The goal for each of these simulation prototypes was to achieve massive speedup, both by actual measurements on two primary machines (an 8-core MacBook Pro laptop with an Intel I9 CPU, and a 32-core AMD Ryzen Threadripper 2990WX Linux machine), and by critical path analysis to predict potential performance on much larger supercomputers. In addition to these machines, WarpIV owns a 264-core server cluster[33] and has recently purchased a larger 64-core AMD machine.[34]

The WarpIV Kernel internally performs a critical path analysis of discrete-event simulation runs during execution, which provides a maximum speedup estimation.[35] So, for example, a large simulation involving many entities (object instances) executing on a 264-core server cluster might report a critical path speedup of 100,000, which means that the application could potentially scale up to large supercomputers with as many as 100,000 cores. Critical path analysis is extremely helpful in matching problem size with appropriate numbers of cores.

Phase I results show that scalable speedup is possible on large supercomputers for these four prototypes as (1) the number of particles and sectors in the planetary ring simulation increases, (2) the grid-cell resolution for aggregating mass regions gets smaller relative to the planet size when computing gravitational forces for ellipsoidal planets, (3) the number of space debris particles, satellites, and/or rockets increase, and (4) the number of transmitters (generating signals) and receivers (sampling signals in the time domain) increase.

Work Plan and Accomplishments

This section provides a technical overview of (a) the planned work that was outlined in the original Phase I proposal, (b) the actual work that was performed during Phase I, followed by (c) performance benchmarks.

First Prototype: Modeling Planetary Rings as an N-Body Gravitational System

Background

Dr. Steinman formed a team of subject matter experts in 1995 at the Jet Propulsion Laboratory (JPL) to develop a scalable parallel simulation of planetary rings as a large N-Body gravitational system[36] using the SPEEDES

[33] The 264 (twenty-four machines each with 12-cores) server cluster that resides at the WarpIV office is currently shut down due to COVID-19 restrictions.

[34] The 32-core AMD Ryzen Threadripper currently used by WarpIV has an asymmetric memory management architecture that was designed to work with a 16-core motherboard. It has 4 independent channels to RAM, but with non-uniform memory access. The new 64-core AMD CPU is paired with a motherboard that was designed to support 64 cores using 8 independent channels to RAM, but still with non-uniform memory access.

[35] Critical path analysis keeps track of processing dependencies between events and simulation objects (which can be composed of plug-and-play models) during execution. If each simulation object executed on its own core, and if all communication and framework overheads were zero, then the maximum critical path time across all simulation objects would indicate the shortest wall time the execution could complete (i.e., the simulation runs as fast as the slowest simulation object). Critical path analysis also keeps track of cumulative processing time per simulation object. So, the sum of the processing times for all simulation objects in the simulated system represents how long the simulation would take to efficiently complete on a single processor. The ratio of the cumulative processing time divided by the maximum critical path time is the maximum speedup for the simulation. Critical path speedup provides a true indication of the inherent parallelism of a simulation for a given problem size. It can be accurately calculated during the execution of a simulation executing on an arbitrary number of nodes.

[36] For planetary rings, the particles include the planet, moons, and the actual particles that make up the ring. Particles in the ring vary in size, can rotate with differing angular momentum, can collide, stick together, or potentially break up into multiple new particles during collisions.

framework. This "all-star" team consisted of world renowned experts in planetary ring physics such as Dr. Nicole Rappaport (Cassini Radio Science Team - Deputy Team Leader and recognized expert in theoretical planetary ring physics), Dr. Carolyn Porco (Chief Imaging Scientist for the NASA Cassini spacecraft mission to Saturn), several astronomy and supercomputing subject matter experts from CalTech, the Jet Propulsion Laboratory, and the Aerospace Corporation, and a post doctorial research assistant working for Dr. Porco who at the time had developed the world's first and only stable simulation[37] of planetary rings as an N-Body gravitational system.

With the launch of the Cassini spacecraft to Saturn on October 15, 1997, it made sense in preparation for this important mission that NASA ought to develop a scalable high-fidelity simulation of Saturn's planetary rings that would eventually be observed when the spacecraft arrived at its destination in July 2004. The proposal was enthusiastically supported by several key Assistant Lab Directors at JPL but was eventually not funded due to a lack of discretionary funds. Nevertheless, the ideas behind how to simulate planetary rings using parallel discrete-event simulation technologies that were proposed in 1994-1995 are still valid today and need to be demonstrated to the scientific space community. The first grand-challenge type of simulation problem this effort developed is a prototype of planetary rings modeled as an N-body gravitational system. The purpose of this demonstration is to validate the original approach for accomplishing high-performance scalability of N-body gravitational problems and to lay a strong foundation for supporting high-fidelity planetary ring studies in the future where detailed resonances due to moons and exotic ring features can be recreated.

Before describing the details of how to efficiently simulate planetary rings, it is helpful to first discuss how discrete-event techniques (in contrast to time stepping) could be used in a brute-force strategy to propagate particles with different masses in an N-body gravitational problem. Note that better techniques involving grid sectors to aggregate particles used in long-distance force calculations were actually used in this effort to dramatically improve scalability (i.e., reduced computations, fewer messages, and less memory consumption).

With brute-force time stepping, every particle would compute the gravitational force from every other particle at each time step. Then the combined forces experienced by each particle would allow them to integrate their trajectory to a new position and velocity in space for the next time step. A simple second-order Runge-Kutta technique that integrates velocity first and then position from the gravitational forces could be used to integrate all of the particle trajectories in a fixed time step.

The discrete-event approach flips this paradigm around by allowing each particle to separately determine how often it needs to compute the gravitational force from each of the other particles. Forces from particles that are far away would not have to be computed as often as forces from particles that are nearby. The gravitational force between two moving particles at a given time is expressed as follows:

$$\vec{F}_{12} = \frac{m_1 m_2 G}{\left|\vec{R}_{12}(t)\right|^3} \vec{R}_{12}(t)$$

In this equation, $\vec{R}_{12}(t)$ represents the vector from particle 1 to particle 2 as a function of time. The magnitude of the change in force, δF_{12}, as a function of the change in time, δt, can be written as follows:

$$\delta F_{12} = \frac{m_1 m_2 G}{R_{12}^3} \left|3v_R - v\right| \delta t$$

Particles are aggregated to represent the true mass of the ring. An important optimization is to eliminate tight oscillating near-elastic collisions between close particles attracted by their mutual gravity.

[37] The simulation ran on a Cray vector machine using FORTRAN and time stepping. Challenges in simulating stable planetary rings involve issues with particles in the ring decaying into the planet and/or escaping the gravitational field of the planet altogether while new particles (usually from the atmosphere of the planet, or perhaps meteorites) are brought into the ring.

In this equation, v_R represents the velocity difference between the two particles projected onto the $\vec{R}_{12}$ vector and v represents the overall magnitude of the velocity difference between the two particles. This equation can be inverted to determine δt given a fixed tolerance for δF_{12}.

$$\delta t = \frac{R_{12}^3}{(m_1 m_2 G) \times |3v_R - v|} \delta F_{12}$$

A plot of the time distribution for force updates between particles randomly distributed in a ring (with a typical fixed time step) is shown in Figure 2.

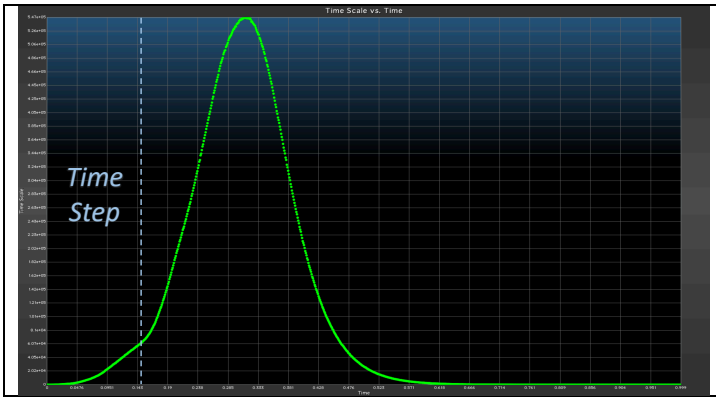


Figure 2: Distribution for force update times between pairs of particles randomly distributed in a circular planetary ring while holding the change in force constant. In this distribution, the horizontal time axis is a log plot, which was necessary to show sufficient detail in the time distribution. Small changes in where to place the time step when using a fixed time step approach actually represents very large update rate differences. The time stepped approach computes most of the forces far more often than necessary, while not accurately supporting the very-tight time scales, which is often when the most interesting physics occurs.

There is an extremely important lesson to learn from the analysis of (a) discrete-event time scales vs. (b) fixed time steps for computing gravitational forces in N-body systems. The discrete-event approach bounds the force calculation error between particles to be within a specified tolerance while allowing time scales for their computations to fluctuate. The time stepped approach holds the update rate constant while allowing the force-calculation errors to vary, which can wreak havoc during verification, validation, and analysis of systematic errors. The only way to semi-validate that the chosen time step is small enough when using fixed time steps is to painfully perform successive test runs with decreasing step sizes to determine when results do not change by a significant amount. Simulations with fixed time steps are generally required to set their time steps to be very small to keep errors manageable, but this causes the simulation to run very slow and inefficient because many calculations are performed far more often than needed. The discrete-event methodology optimizes how often computations are performed, which allows the simulation to run much faster.[38] Yet, the discrete-event approach also supports the tightest time scales, which can be where much of the interesting physical phenomena occurs. So, the discrete-event approach not only runs significantly faster, but it also likely produces more accurate results.

So, an obvious question to ask is, "Why are parallel discrete-event simulation techniques not commonly used by the scientific computing community?" WarpIV Technologies, Inc. believes that the problem comes from the lack of interdisciplinary dialog between the computer science R&D community and the scientific computing community.[39]

[38] Analysis of brute-force N-body gravitational computations for a planetary ring with 1,000 particles show the following over-computation factors for different time step cutoffs relative to the update time distribution. Time Step cutoffs (i.e., the area under the curve to the left of the time step) at 1%, 10%, 25%, 50%, 75%, 90%, 99%, provide over-computation factors of 16404, 961, 229, 53, 13, 3, 0.2 respectively (note that the 99% cutoff actually shows extreme under-computing, which would create very large systematic errors making results invalid). So, for 1,000 particles, traditional time stepping at a 1% cutoff to hopefully produce accurate outputs does more than 4 orders of magnitude computations than needed. These computational results were produced by a software program written specifically for this proposal.

[39] Computer scientists typically simulate queuing systems, computer networks, circuit boards, cellular networks, Petri Nets, and various toy problems designed to test parallel simulation algorithms. Computer scientists do not typically have the math, science, or engineering background to simulate computational fluid dynamics or gravitational force problems. On the other hand, the scientific community generally lacks hands-on knowledge of the advanced optimistic parallel discrete-event simulation technologies researched and developed by the computer science community. This effort bridges the gap between these two diverse communities. Dr. Steinman, Ph.D. in elementary particle physics, principal developer of SPEEDES and the WarpIV Kernel, and scientist at the Jet Propulsion Laboratory from 1988-1996 (including support for the Cassini Mission) is uniquely qualified to bring these two communities together.

It has taken more than 30 years for the SPEEDES/WarpIV simulation technology (with more than 825,000 lines of application-neutral code) to mature into what it is today. Nobody from the scientific computing community starts their research project by first developing a rollback-based discrete-event simulation engine with 825,000 lines of code. Optimistic parallel discrete-event simulation is only deeply understood by a small group of researchers generally within the academic computer science community, and not well known by the scientific computing community where (a) FORTRAN is often the programming language of choice, and (b) a time-stepped, synchronized DO-loop is the effective simulation engine. Time stepping provides an easy way to develop parallel science codes quickly, but it is not always the most efficient way to perform computations in irregular simulation applications. Parallel discrete-event simulation can be much more efficient in supporting models with irregular time scales, but it requires a far more complex simulation engine to host such applications.

Another mistake often made by the supercomputing community is to characterize success of an application running on high-performance computers by measuring speedup rather than overall speed. As previously described for N-body gravitational problems, the discrete-event approach can dramatically decrease computations to run much faster than time-stepped approaches while also providing more accurate results. But obtaining speedup in parallel discrete-event simulations can sometimes be more difficult (especially for irregular problems when workloads are dramatically squashed or shift due to changing time scales). So, it is not uncommon for a discrete-event simulation to run orders of magnitude faster on a small supercomputer than a time stepped code (with near perfect speedup) doing more work than necessary in a regular, perfectly load-balanced, fashion on a larger supercomputer. While *speedup* is of course always desirable, the real goal of any application should be *speed* for a given level of fidelity. In other words, for a given problem size and a given number of cores, the truest measure of performance is always speed, not speedup.

Implementation

The planetary ring simulation achieved both speed and scalable speedup, while also providing a strong foundation for supporting very high-fidelity results in future enhancements. Figure 3 shows the WarpIV Visualizer playing back the particles and ring sectors rotating about the planet. Figure 4 shows the WarpIV Analyzer that is used to form mathematical plots of results. The Visualizer and Analyzer tools were critical in verifying and validating the planetary ring simulations.[40]

[40] An interesting issue arose in modifying the WarpIV Kernel coordinate system conversion routines involving latitude and longitude spatial dimensions, which are used in the Visualizer. The Earth rotates once every 24 hours, while Saturn rotates once every 10.5 hours. This fact initially caused problems in the Visualizer as planets (relative to a non-rotating planet) appeared to be rotating in the wrong direction. The problem went away once the rotational period was corrected.

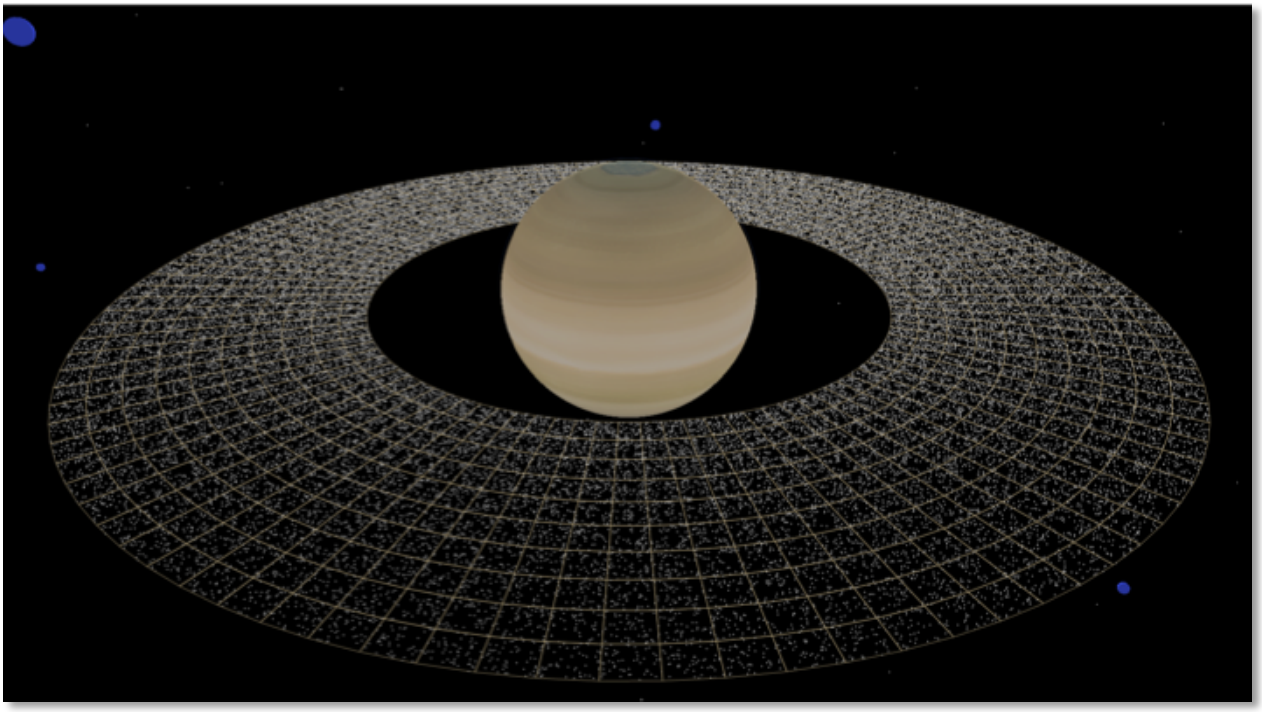


Figure 3: Visualization of the planetary ring simulation implemented in the WarpIV Kernel showing the planet (Saturn), four of its moons (blue spheres), sectors arranged in 10x100 radial/angle cells, and 50,000 particles whose trajectories are integrated over time based on gravitational forces.

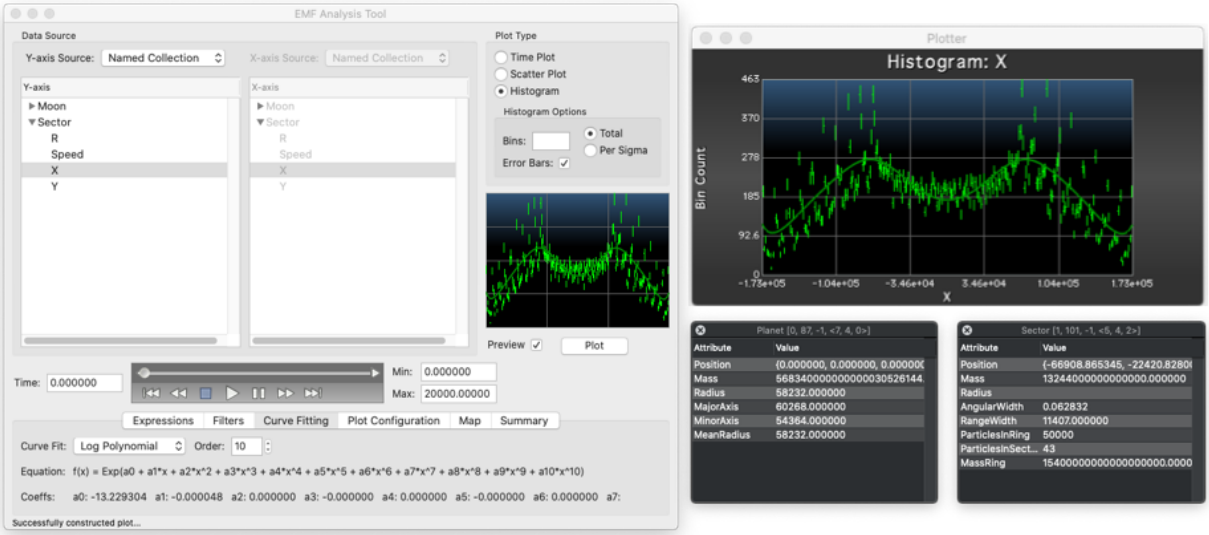


Figure 4: WarpIV Analyzer showing a histogram plot of the X dimension for moving sectors. The histogram is curve fitted using a 10th order log polynomial function. Also, two windows are displayed showing the attributes of published federation objects.

The design of the planetary ring simulation has four key elements.

1. The planet is represented in a generic manner so that any planetary ring system can be simulated. A federation object for the planet is published when the simulation starts up to provide its important physical characteristics to the other critical elements of the simulation that subscribe to the planet. Planet parameters include: (a) name, (b) mass, (c) mean radius, (d) major axis, and (e) minor axis. The planet is assumed to

represent the origin of the inertial coordinate system. A future enhancement might be to define the sun as the origin of the inertial coordinate system and perhaps include the gravitational effects from other planets. The current simulation configuration developed in this effort modeled Saturn as its central planet. All parameters describing its mass, rings, etc., were derived from websites such as https://science.nasa.gov.

2. Like the planet, moons are represented in a generic manner so that any moon system can be represented in the planetary ring simulation. Each moon subscribes to the planet's published federation object and initiates its orbit based on its distance from the planet's surface and the planet's mass. The current implementation assumes circular orbits in the equatorial plane with a random phase angle. However, this can be easily extended in the future to support more carefully specified elliptical orbits.[41] A federation object is published for each moon, which (along with the planet) provides information to sectors about their mass and position in space as a function of time. It is important to note that the position of moons as a function of time is represented as an equation that is abstracted within in its published federation object. So, messages updating moon positions are not required once the moon federation object is constructed and published. Moon parameters include: (a) name, (b) mass, (c) radius, and (d) orbit radius. The current simulation configuration models the seven largest moons of Saturn: Mimas, Enceladus, Tethys, Dione, Rhea, Titan, and Lapetus.

3. Sectors subscribe to the planet and moon federation objects, which are then used to initialize each sector's rotational motion about the planet. Sectors are represented as two-dimensional (i.e., distance from the planet and longitudinal angle) cells that are implemented about the equatorial plane of the planet. Sectors rotate about the planet in circular orbits based on the center location of each sector. Sectors within the same radial band move with the same angular velocity, which simplifies their implementation. In this way, the position of a single sector representing Longitude 0.0 at time 0.0 (i.e., the start time of the simulation) can be determined as a function of time and then rotated by an additional amount to determine the location of any other sector within the same radial band. As defined by gravitational mechanics of circular orbits, sectors closer to the planet orbit quicker in time than sectors further away from the planet. The current simulation configuration modeled sectors with 10 radial bands and 100 angular bands to provide 1,000 sectors, which were evenly distributed across processors to support parallel computations. In the future, larger numbers of sectors could be modeled to achieve further scalability on extremely large supercomputers.

4. Particles are created by each sector during initialization to represent the total mass of the ring. At startup, each sector creates a specified number of particles with random placement within its sector. The velocity of each particle is determined as a circular orbit plus a small three-dimensional random velocity offset. In this way, particles are roughly spread out across the ring in a nearly even manner, yet with somewhat random positions and velocities. Particles are not constrained to orbit in the equatorial plane. Sectors evolve their particle trajectories based on gravitational forces using (a) the mass of the planet, (b) the location and mass of the moons over time, (c) the aggregate mass of the other sectors, which can change over time, and (d) the mass of particles located in the same sector. Particles can dynamically enter and exit sectors over time. However, since the sectors rotate at nearly the same speed as their particles, these events (which potentially involve inter-node communication) are kept to a minimum. Particle trajectories use a novel fifth-order spline integration approach to increase time steps without introducing large errors. This reduces how often trajectory computations occur. The current simulation configuration modeled 10-50 particles per sector, which means that with 1,000 sectors, the trajectories of 10,000-50,000 particles were represented in the ring. In terms of performance, the number of particles that could be modeled in the simulation scales with the number of cores used by the simulation.

In contrast to brute-force, pairwise, gravitational force updates, the planetary ring simulation developed in this effort used a much more efficient technique. As outlined above, the ring was decomposed into a large number of two-dimensional sectors (binned by angle and radial distance from the planet) to aggregate particle masses in each sector for gravitational force calculations that occur between sectors when updating particle trajectories. An

[41] The WarpIV Kernel provides an elliptical orbit motion type that is often used in WarpIV missile defense applications involving the trajectories of warheads in free flight and satellites in non-circular orbits.

important optimization was to monitor the total mass within a sector and only update the other sectors when its change in total mass (due to particles entering or leaving) exceeded a specified threshold. For example, a 10% threshold for a sector with 50 equal-mass particles could change by +/- 5 particles (assuming each particle had the same mass) before updating the rest of the sectors. This way of doing things is consistent with the discrete-event methodology where events are only scheduled and processed when necessary to maintain systematic error tolerances. So, as particles potentially enter and exit a given sector over time, its total mass likely stays relatively constant. This tolerance-based technique for scheduling events significantly reduced the number of messages between sectors passed during the simulation execution. It also reduced time dependencies (and rollbacks) that affect the critical path of the simulation.

Each sector rotated around the planet's equatorial plane with a physically correct circular orbit defined by its distance (specified by the center of the sector) from the planet and the mass of the planet. Because the sectors orbit the planet at roughly the same speed as their internal particles, particles rarely required moving from one sector to another. The technique of sectors roughly moving at the same speed as its particles, in contrast to fixed (non-rotating) sectors, also significantly reduced messages and critical-path time dependencies during the simulation's execution. The simulation was designed to then allow particles within a sector to calculate their pairwise mutual forces (and potential collisions) locally without requiring messages.

Particle trajectory integration used a fifth-order polynomial spline iteration technique that was developed by WarpIV Technologies, Inc. in prior work performed for the Missile Defense Agency to integrate complex missile trajectories where forces were known due to the rocket's thrust, atmosphere-related aerodynamics, and non-spherical gravity.[42] The algorithm begins with a second-order Runge-Kutta integration that uses the known acceleration (due to all combined gravitational forces) to first evolve velocity and then position to a new time as follows. Note that this first step assumes that the acceleration does not change over the time step.

$$\vec{\mathrm{a}}(\mathrm{t}+\delta\mathrm{t}) \approx \vec{\mathrm{a}}(\mathrm{t}) = \frac{\vec{F}(t,\vec{x}(t))}{m}$$

$$\vec{\mathrm{v}}(\mathrm{t}+\delta\mathrm{t}) \approx \vec{\mathrm{v}}(\mathrm{t}) + \vec{\mathrm{a}}(\mathrm{t}) \times \delta t$$

$$\vec{x}(t+\delta t) \approx \vec{x}(t) + \vec{v}(t) \times \delta t + \vec{\mathrm{a}}(\mathrm{t}) \times \frac{\delta \mathrm{t}^2}{2}$$

The new position of the particle can then be used to compute a new acceleration (due to all combined gravitational forces divided by the particle's mass) and then compared to the acceleration that was used in the integration to form a delta acceleration error.

$$\delta\vec{a} = \left(\frac{\vec{\mathrm{F}}(t+\delta t,\vec{x}(t+\delta t))}{\mathrm{m}}\right) - \vec{a}(t+\delta t)$$

Then, the following corrections are applied to adjust the acceleration, velocity, and position at t + δt.

$$\vec{a}(t+\delta t) \leftarrow \vec{\mathrm{a}}(t+\delta t) + \delta\vec{a}$$

$$\vec{v}(t+\delta t) \leftarrow \vec{v}(t+\delta t) + \delta\vec{a} \times \delta t$$
$$\vec{x}(t+\delta t) \leftarrow \vec{x}(t+\delta t) + \delta\vec{a} \times \frac{\delta \mathrm{t}^2}{2}$$

[42] Fifth-order polynomial splines are very powerful because they use position, velocity, and acceleration information at the start time and end time of the segment to form the fifth-order polynomial. The fifth-order polynomial integration strategy essentially computes the difference between the acceleration due to known forces at the end time of the segment (which is computable based on the position) vs. the acceleration in the spline. A correction to the end acceleration is applied that slightly changes the end time position and velocity. This step is repeated until the acceleration error is small.

These corrections are fed into a fifth-order polynomial spline routine, where the process is repeated until the acceleration error is either small enough to ignore or when full machine double-precision is reached.

The reason why this approach works so well is that the calculated from the new position and velocity should match the actual acceleration, which can be directly calculated using the force functions. Mismatches are corrected based on the assumption that the force terms change smoothly during the step, which implies that polynomial correction terms can very accurately represent the changes to the combined forces during the step.

The number of iterations required to reach high precision is typically 3 or 4, which means that the force functions are generally computed about 2.5 times per step. Note that the force terms computed on the final iteration can be used for the start of the next step, which is why the number of function calls per step is 2.5 instead of 3.5. This important optimization trick reduces the number of force term computations by one in each step, which is extremely significant for obtaining optimal performance. In contrast, the often-used fourth-order Runge-Kutta approach requires four combined force computations per step and is far less accurate. The important benefit to the fifth-order polynomial approach is that it can utilize much larger time steps in contrast to even fourth-order Runge-Kutta integration, while producing far better accuracy because the velocity integration error is of order δt^5 while the position integration error is of order δt^6.

Another way to characterize the fifth-order polynomial spline algorithm is that it very efficiently and accurately determines a cubic polynomial representation (rather than a constant value used by second-order Runge Kutta) for the acceleration term.

$$\vec{x}(t+\delta t) = \vec{x}(t) + \vec{v}(t)\delta t + \left[\frac{\vec{a}(t)}{2} + \vec{c}_1\delta t + c_2\delta t^2 + c_3\delta t^3 + \cdots\right]\delta t^2$$

A very interesting phenomena was observed in Phase I that should be explored in Phase II. Particle integration biases, even when using the fifth-order polynomial spline technique, were observed that gradually decayed particle orbits inward towards the planet. This problem has been observed in other planetary ring simulations, so it was expected and not a surprise. The rate of orbital decay naturally increased with the time step size in the fifth-order polynomial (and even worse with second-order Runge-Kutta techniques), which indicates that this is an integration problem and not due to other forces. We believe a better way to integrate particles would be to apply similar fifth-order perturbation/correction techniques based on correcting circular or elliptical orbits rather than simply extrapolating position, velocity, and acceleration. In this way, the perturbation to elliptical orbits would remove the extrapolated polynomial bias, which we believe is what causes the decay. This technique will be developed and tested in Phase II. Its benefit will be to support stable particle integration using much larger time steps, yet without orbital decay. So, the simulation will (a) run much faster and (b) it will also be more accurate.

One more important optimization was provided. The gravitational force within a sector due to all of the other sectors is computed once and then rotated to each particle's position. So, the gravitational force from the other sectors, which can be a very large computation,[43] is computed once and then shared by all of the particles in the sector.

Detailed (and computationally intense) particle interactions, such as potential collisions (which include how particles might stick together or break apart into additional particles)[44] is deferred to Phase II when this work will be hosted on larger machines. Another straightforward capability that might be implemented in Phase II is to support checkpointing, which would allow a simulation to save its state and then resume from a checkpoint. This

[43] The current simulation configuration has 1,000 sectors, where each sector has on average 50 particles. So, computing the gravitational force for 999 sectors for each of the 50 particles takes approximately 50 times longer to compute than when done the computation once and then sharing the result. A future optimization will be to only redo the sector gravitational computation when the mass of any of the sectors change. This optimization could provide another order of magnitude performance improvement.

[44] Particles rotate, which can affect collisions where momentum is conserved.

might be an important feature to implement when very long run times are needed for steady-state resonances in the ring to emerge.[45] Periodic checkpointing reduces the risk of hardware (or software) failures wasting valid computations.

Performance

Performance was measured on (1) an 8-core MacBook Pro laptop configured with 64 GB RAM and 8 TB flash hard drive, and (2) a 32-core AMD Ryzen Threadripper CPU machine running Linux. Benchmarks were performed with data logging enabled and disabled. When data logging is enabled, each sector creates bit-compressed records that store particle trajectories as they are updated (used for visualization playback) and additional data (used for mathematical analysis). Records from all of the sectors distributed across nodes are communicated to node 0 through the WarpIV High-Speed Communications infrastructure where the records are time-merged and written to file. So, enabling data logging creates a small bottleneck as a single file is written. Note that an alternative data logging method that does not have a bottleneck issue would be to have each node write its own file.[46] However, these separate files would likely have to be merged during offline usage for visualization and analysis purposes. In general, it is best to simply write a single time-merged file utilizing the same high-speed communications and computational resources that are available while running the application.[47]

The benchmarking revealed very interesting (cache/RAM) memory bottlenecks that need to be discussed to provide context for the actual results shown in the figures below (see Figure 5, Figure 6, Figure 7, and Figure 8). The lower left plots in each of these figures show average computational timings for the dominant event in the simulation, which in the planetary ring simulation was the event that computes the trajectories of particles located in each sector. The actual computations performed in this event are independent of the number of nodes used in the simulation. So, the fact that the time to process the exact same event increases as the number of nodes increases reveals a memory bottleneck as data blocks from RAM are moved in and out of the CPU's cache.[48] The lower-right speedup plots in each of these figures show what the speedup would have been if there were no memory bottlenecks.

[45] Thresholds and time scales can be tightened up dynamically as steady state resonances emerge to produce even more detail in the physical results.

[46] The WarpIV bit compressed data logger does not assume that all nodes have access to a shared file system. This is one of the reasons why the design of the data logger communicates records to either node 0 or to a dedicated server when merging records into a single file.

[47] The WarpIV Kernel bit-compressed data logger does not assume that all compute nodes have access to a shared (network) hard drive. In fact, the machines used by a parallel or distributed WarpIV simulation that is logging data could even be distributed across wide area networks. Since communications would need to be performed to gather records across such distributed machines, it is best to use the high-speed communications that are already in place when running the parallel simulation.

[48] Although we have not verified this with 100% certainty, we do not believe that the slowdown was due to CPU throttling of its clock speed to prevent overheating.

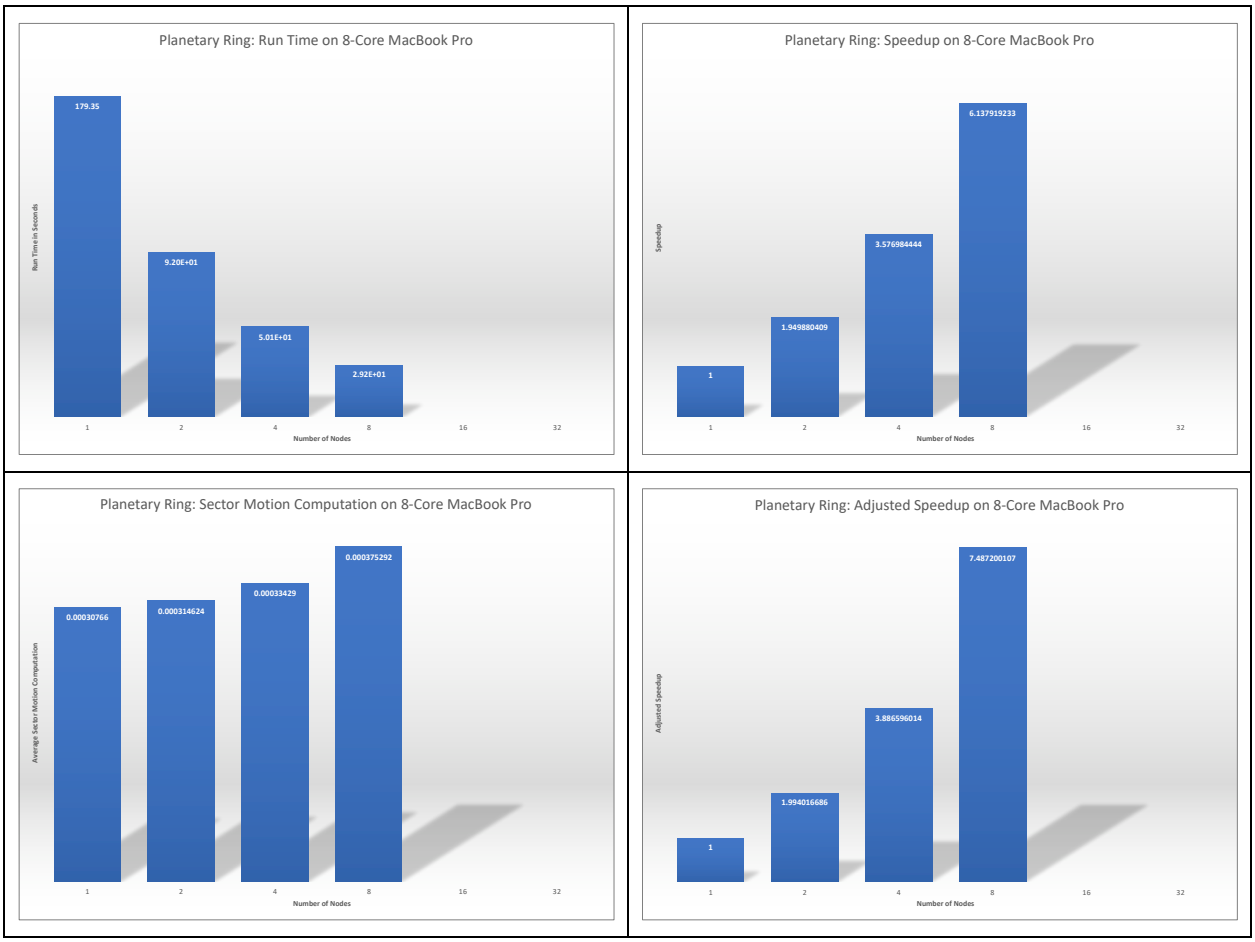


*Figure 5: Benchmarks for the Planetary Ring simulation on the 8-core MacBook Pro with **data logging turned off**. Upper left shows the execution time, Upper Right shows speedup. Lower left shows how computational performance of the most expensive event increased due to Cache/RAM contention issues with the CPU and motherboard. Lower right shows adjusted speedup where the Cache/RAM contention affects are factored out. The adjusted speedup indicates the speedup that would be achieved if there were no computational slowdown effects due to Cache/RAM issues.*

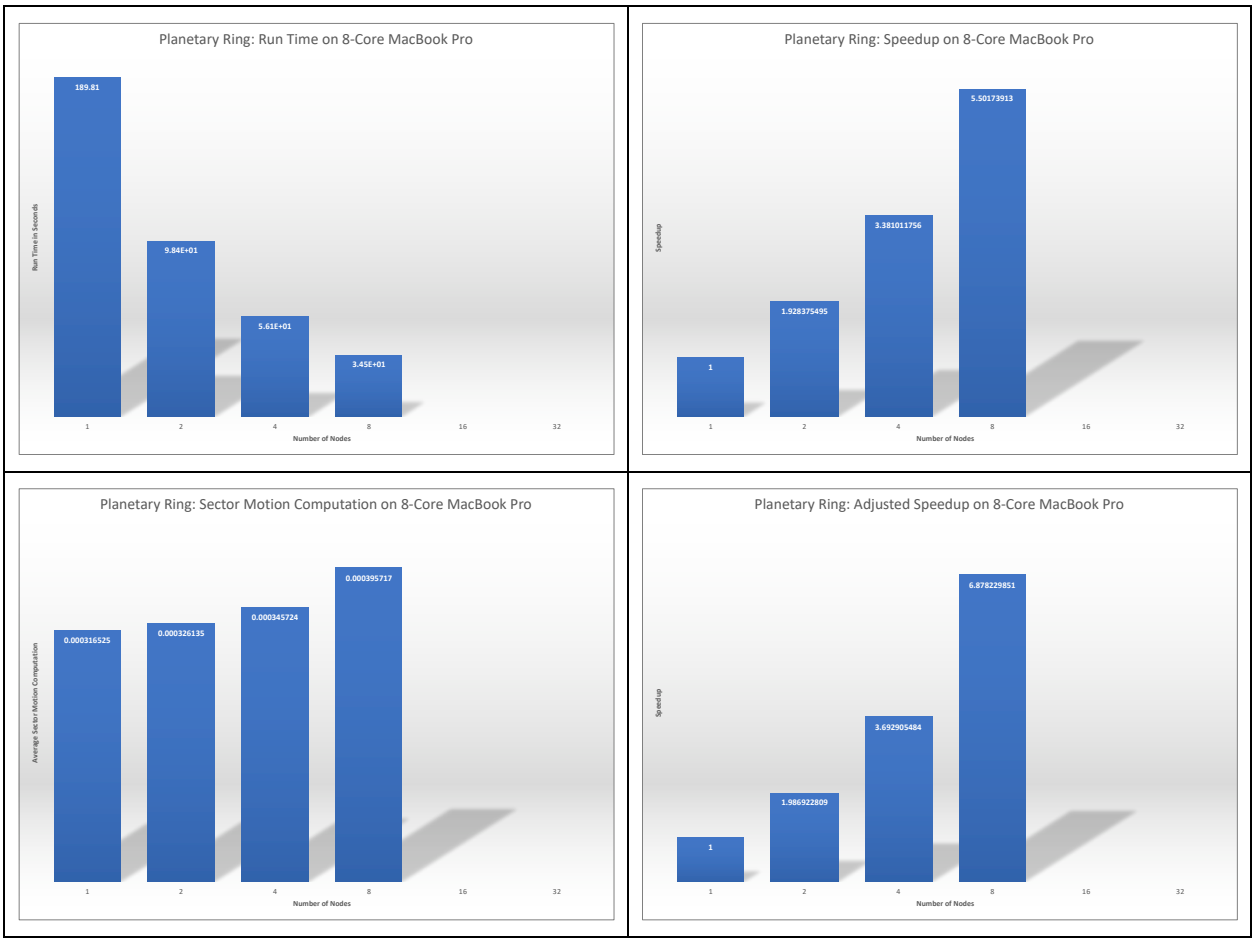


*Figure 6: Benchmarks for the Planetary Ring simulation on the 8-core MacBook Pro with **data logging turned on**. Upper left shows the execution time, Upper Right shows speedup. Lower left shows how computational performance of the most expensive event increased due to Cache/RAM contention issues with the CPU and motherboard. Lower right shows adjusted speedup where the Cache/RAM contention affects are factored out. The adjusted speedup indicates the speedup that would be achieved if there were no computational slowdown effects due to Cache/RAM issues.*

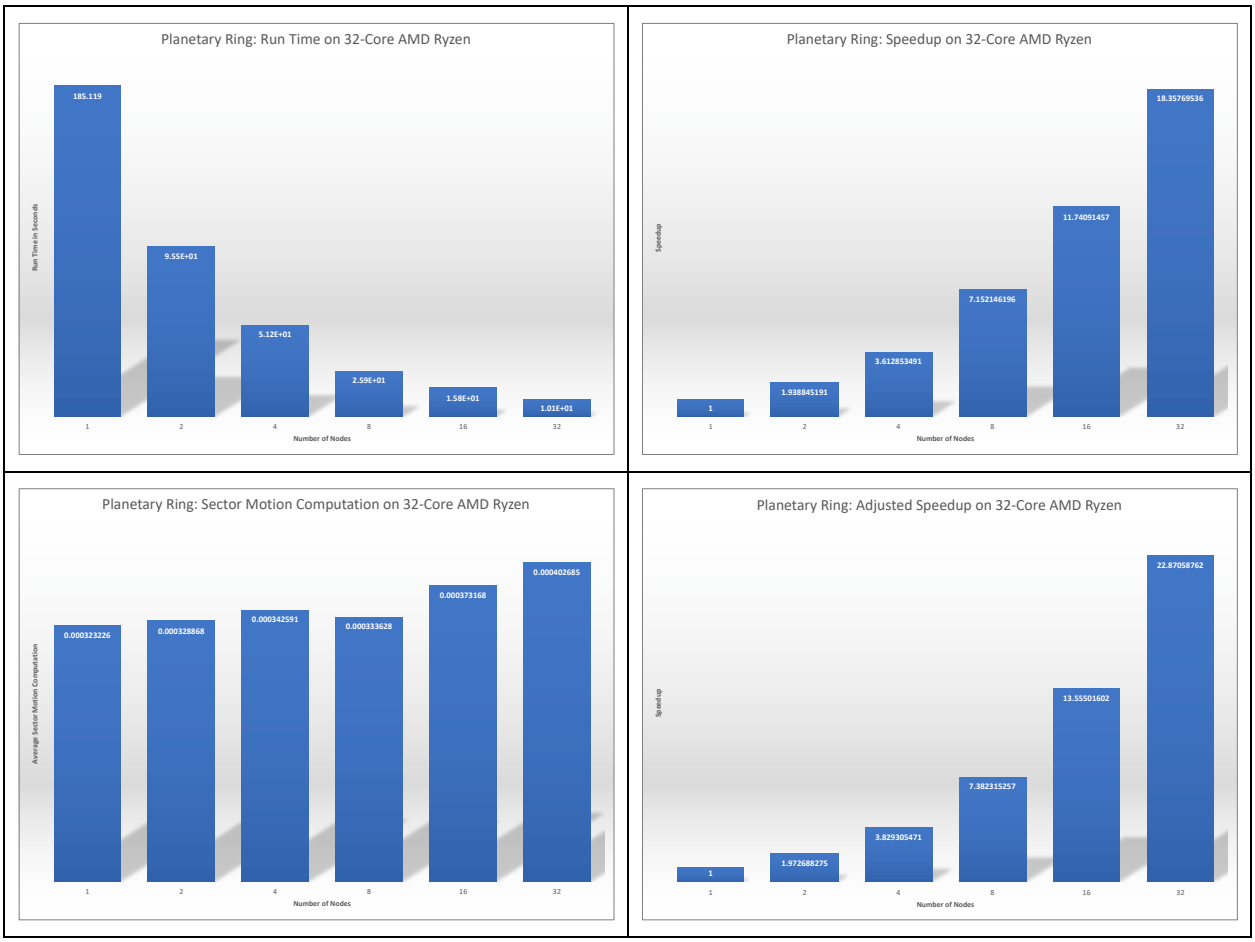


*Figure 7: Benchmarks for the Planetary Ring simulation on the 32-core AMD Ryzen Threadripper with **data logging turned off**. Upper left shows the execution time, Upper Right shows speedup. Lower left shows how computational performance of the most expensive event increased due to Cache/RAM contention issues with the CPU and motherboard. Lower right shows adjusted speedup where the Cache/RAM contention affects are factored out. The adjusted speedup indicates the speedup that would be achieved if there were no computational slowdown effects due to Cache/RAM issues.*

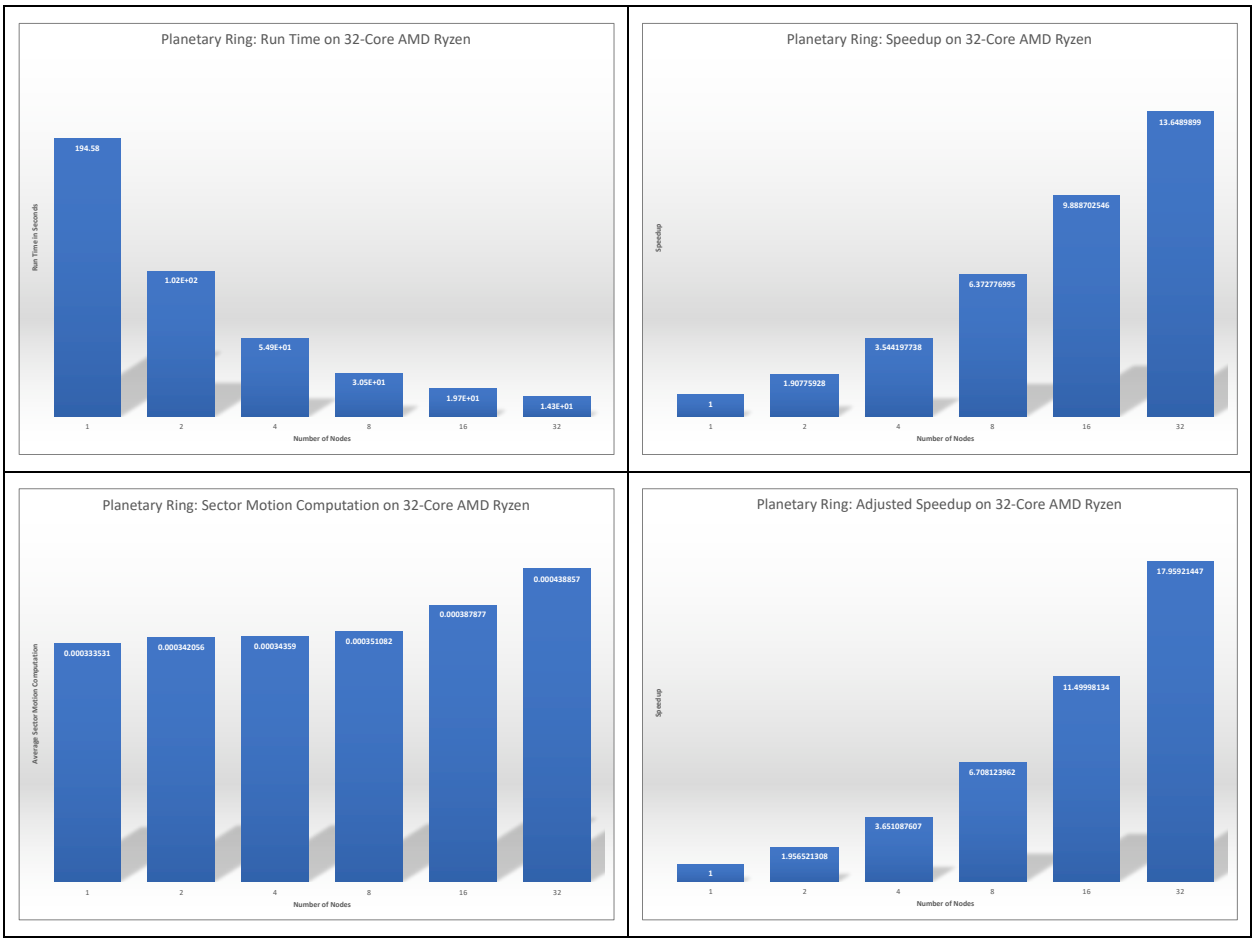


*Figure 8: Benchmarks for the Planetary Ring simulation on the 32-core AMD Ryzen Threadripper with **data logging turned on**. Upper left shows the execution time, Upper Right shows speedup. Lower left shows how computational performance of the most expensive event increased due to Cache/RAM contention issues with the CPU and motherboard. Lower right shows adjusted speedup where the Cache/RAM contention affects are factored out. The adjusted speedup indicates the speedup that would be achieved if there were no computational slowdown effects due to Cache/RAM issues.*

Critical path analysis of the planetary ring simulation showed potential for unlimited speedup. As long as the change in sector mass remains small, which should occur during steady state operation, the amount of parallelism is only limited by the number of sectors and average number of particles per sector.

Second Prototype: Ellipsoidal Gravity Calculation

Background

Integrating planetary orbital trajectories over long periods of time requires an accurate representation of gravitational forces. A recent R&D effort performed by WarpIV Technologies for the Missile Defense Agency (MDA) created a composable ellipsoidal gravitational force model that is used when the World Geological Survey 1984 (WGS84) representation of the earth (rather than round earth) is specified for high-fidelity simulation runs. The WarpIV approach first subtracted a round earth (defined by the WGS84 minor axis) from the ellipsoidal earth and then integrated the remaining gravitational force for a large number of spatial locations with the remaining shell (see Figure 9). To help validate the results, rigorous consistency testing was performed to ensure that the summed mass of all cells in the shell plus the mass of the enclosed round earth combined to the correct mass of the earth. Tests were also performed to verify that the gravitational field at large distances were consistent with the round earth central force representation. Additional tests were performed at low altitudes to ensure that gravitational forces were computed smoothly even at positions located very close to the ground near mass cells. This was

important for missile defense applications involving rocket and interceptor launches as well as impact. Gravitational forces due to the atmosphere were ignored in this previous effort for missile defense.

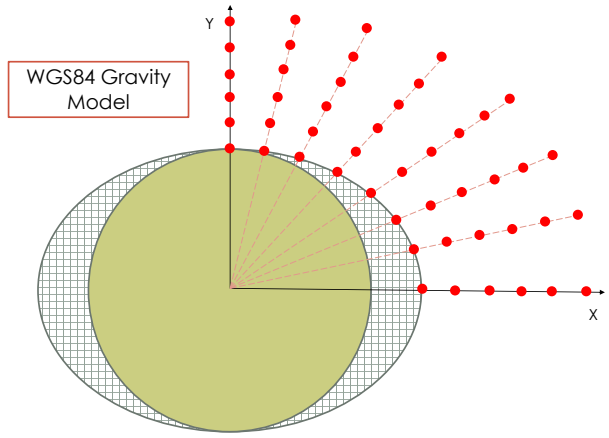


Figure 9: The WarpIV WGS84 gravity model subtracts the round earth defined by the minor axis of the ellipsoidal earth and then integrates the gravity at different (slightly randomized) elevation angles and altitudes (i.e., the red circles that are computed in parallel to achieve scalability) for the remaining shell using different mass cell resolutions. To achieve the highest accuracy, the results are fitted to a two-dimensional 14th order polynomial. Only first quadrant (i.e., X and Y are positive) values are computed due to angular symmetries. The computational time to integrate gravity at all the points is inversely proportional to the specified resolution cubed because (1) the number of mass cells is inversely proportional to the resolution squared and (2) the actual number of positions evaluated was designed to be inversely proportional to the resolution to further improve the curve fit with matching grid-cell resolution.

Various cell sizes were computed with this new gravitational model of the earth, including 10, 5, 2, and 1-kilometer resolutions for the mass cells. The forces from each of the cells in the shell were computed in parallel to achieve scalable performance[49] for a large number of elevation angles and altitudes, and then curve fitted with a high-order polynomial[50] to support very accurate gravitational force computations at any elevation and altitude. The total gravitational force is obtained from the enclosed round earth with a central force plus the additional gravitational force (using the interpolated polynomial curve fit) obtained from integrating the shell. Angular symmetries allowed the gravitational forces to be computed in a two-dimensional plane and then appropriately rotated for three-dimensional use. This model is available for any user of WarpIV.

The NASA Phase I effort was designed to expand on the previous missile defense effort to support higher resolutions on larger machines and additional corrections to represent geological features such as mountains, valleys, oceans, the atmosphere, etc. In addition, the gravitational force calculation was modified to support other planetary gravitational force fields such as Saturn. These minor enhancements were a small effort in Phase I. As one might expect, the rocket-aiming algorithm overheads for missile defense notably increase when using a WGS84 ellipsoidal earth due to (a) the high-order polynomial calculation for gravity performed at each time step, and (b) the slightly more complex impact calculation for ellipsoidal surfaces.

Implementation

The primary software change to the ellipsoidal gravitational force model in Phase I was to extend all of the hard-coded physical earth constants to more general planetary constants. These included physical parameters such as the mass, major axis, and minor axis of the planet, which for this effort was chosen to be Saturn. Other parameters include (a) the maximum distance from the planet for representing its gravitational field, and (b) the number of spatial data points used to calculate the field based on latitude angle and radial distance from the planet's surface.

Performance

The gravity model was executed on an 8-core MacBook Pro laptop and a 32-core AMD Ryzen Threadripper Linux machine. Performance results are shown in Figure 10 and Figure 11. Other benchmarks have shown that computations slow down due to cache/RAM bottlenecks as more cores are employed. So, the speedup of

[49] While the 10-kilometer integration can be performed reasonably quick on a 4-core laptop, the 1-kilometer resolution integration took nearly 3 hours to complete using the 264-core HPC rack-mounted server cluster at the WarpIV office. The Missile Defense Engagement Model (MDEM) developed by WarpIV uses the 1-kilometer resolution gravity model when running with the WGS84 earth representation.

[50] It was determined that a 14th order polynomial of two variables worked best in terms of very accurately portraying the gravitational field at any elevation and altitude.

20.6341521 on the 32-core AMD Ryzen Threadripper was also likely impacted by this effect. We believe that the speedup would be much closer to 32 on a shared memory machine designed for scalable HPC applications.[51] The computations are nearly evenly distributed across nodes. Communications, which primarily occurs at the end of the processing to gather results, are minimal.

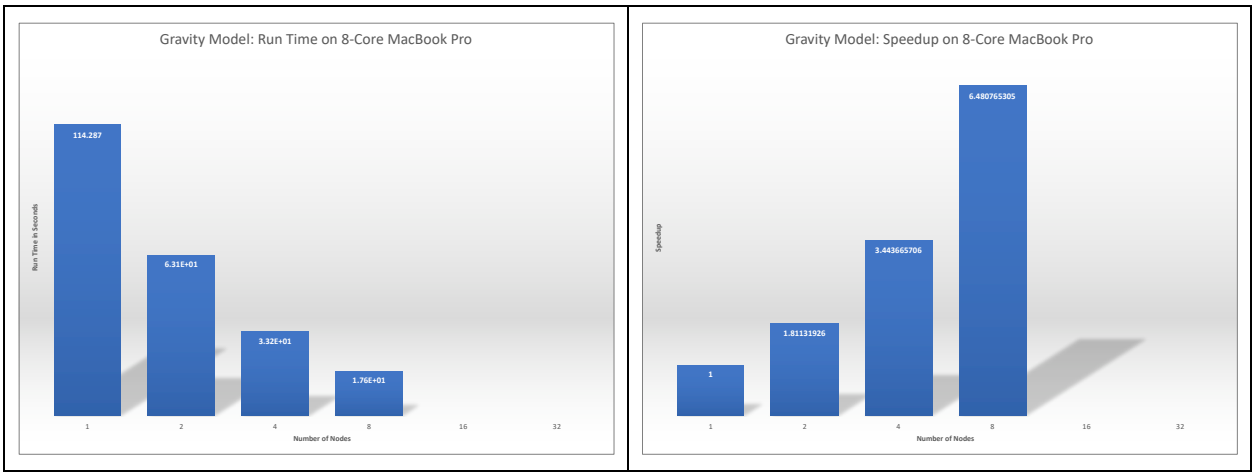


Figure 10: Run time and speedup of the Saturn gravity model using a 350-kilometer mass resolution executing on an 8-core MacBook Pro laptop.

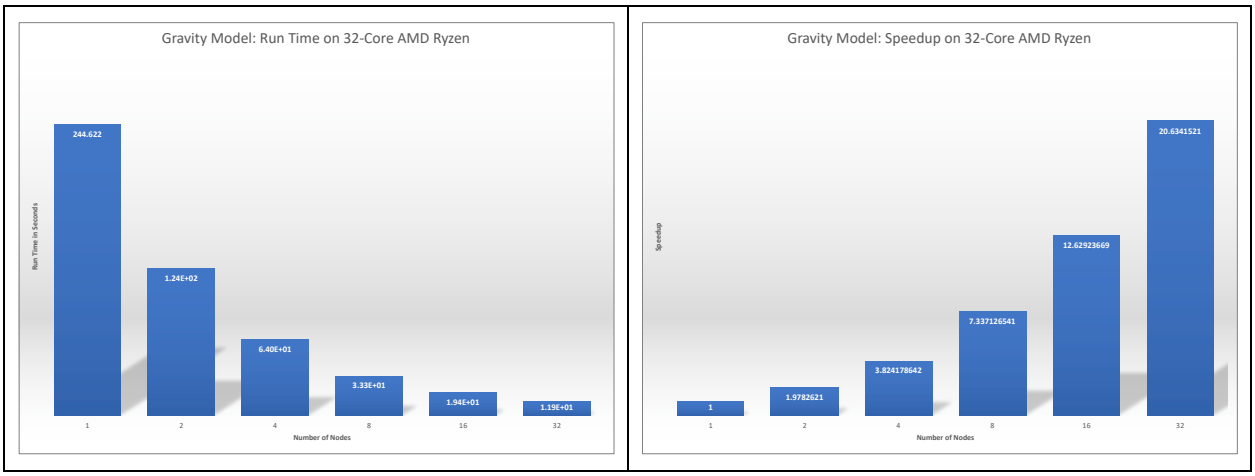


Figure 11: Run time and speedup of the Saturn gravity model using a 350-kilometer mass resolution executing on a 32-core AMD Ryzen Threadripper Linux machine.

Third Prototype: Space Debris

Background

According to NASA, more than 500,000 pieces of debris, or "space junk" are tracked[52] as they orbit the Earth. They travel at speeds up to 17,500 mph, which is fast enough for a relatively small piece of orbital debris to damage a

[51] The Saturn gravity calculation was performed on a 64-core AMD Ryzen Threadripper machine that had twice the number of channels to RAM. Speedup for 32 nodes on that machine was measured to be 25.1987718, which is much closer to a perfect speedup of 32. However, the cache/RAM bottleneck due to the NUMA architecture emerged when using 64 cores, where speedup was measured to be only 36.78729784.

[52] The Department of Defense (primarily the Air Force) maintains a highly accurate satellite catalog on objects in Earth orbit that are larger than a softball. There are more than 20,000 pieces of debris larger than a softball orbiting the Earth. They travel at speeds up to 17,500 mph, fast enough for a relatively small piece of orbital debris to damage a satellite or a spacecraft. There are 500,000 pieces of debris the size of a marble or larger. There are many millions of pieces of debris that are so small they can't be tracked.

satellite or a spacecraft. The space debris problem boils down to (1) detecting and tracking each of the pieces of debris (requires high-power and high-precision ground-based radars with detections taken from multiple aspect angles using advanced discrimination techniques to separate tracks), (2) accurately propagating their trajectories over long periods of time (requires an extremely accurate gravity model), and (3) determining if (or when) any satellite in orbit or launched rocket might be impacted by the debris.

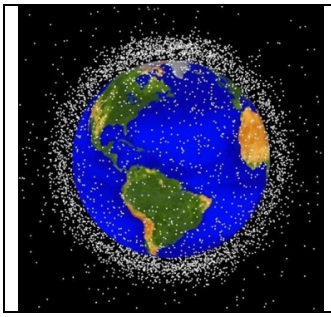


Figure 12: Space debris image taken from the NASA website:

https://www.nasa.gov/sites/default/files/4-8-nrc-evaluates-nasas-orbital-debris-programs.jpg.

This effort focused on combining (a) space-debris trajectory propagation, and possible (b) satellite or rocket impacts (closest approach calculations). The WarpIV Kernel provides a data structure, known as Hierarchical Grids (known as HiGrids), to support these types of calculations in a rapid and scalable manner. HiGrids represent abstract dimensions (including a special 3-dimensional space dimension) as a multidimensional, multiresolution, grid system. Its primary purpose is to rapidly perform publish/subscribe overlap region calculations in a scalable manner. Optimal management of memory within the HiGrid (especially when using extremely high resolutions) is one of its prime features. In one test, the HiGrid data structure was shown to speed up an Air Force simulation, that was bogged down by determining which ground entities were detectable by which aircraft, by several orders of magnitude.[53]

HiGrids were originally designed to be parallelized, but this important capability was never fully implemented in the WarpIV Kernel. This Phase I effort parallelized the HiGrid data structure in preparation for full use in Phase II. The primary purpose of HiGrids is to efficiently manage the publication of hundreds of thousands of space debris for subscribers such as satellites in various orbital constellations and/or launched rockets. HiGrids perform a filtering function so that each space debris particle only performs closest approach computations for satellites and rockets that are within close proximity. This effectively turns the complexity of the problem from $O(n_{debris} \times [n_{satellite} + n_{rocket}])$ to $O(n_{debris} + n_{satellite} + n_{rocket})$. For large numbers of debris, satellites, and rockets, the computational improvement can be significant. A Unified Modeling Language (UML) class diagram of the WarpIV Kernel Hierarchical Grid (HiGrid) data structure is shown in Figure 13.

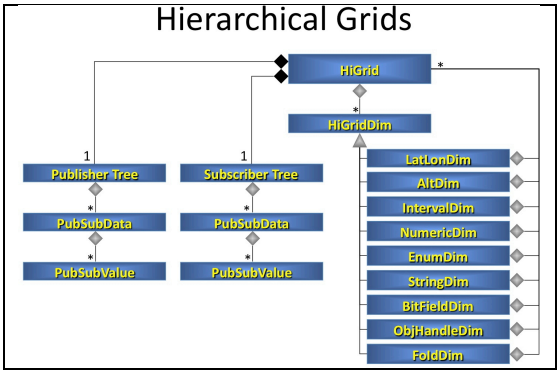


Figure 13: Unified Modeling Language (UML) diagram of the Hierarchical Grid (HiGrid) data structure. HiGrids are defined as a recursive and multidimensional data structure containing various types of HiGrid dimensions, each potentially with multiple resolutions and its own recursively defined HiGrid. Each HiGrid instance in the full data structure manages a publisher tree and a subscriber tree that handles publish/subscribe overlaps. Inserting publication and subscription regions is performed in constant time and is virtually independent of the number of publishers and subscribers represented within the HiGrid. Overlaps between publishers and subscribers are similarly determined in constant time.

[53] Lammers Craig, Steinman Jeff, 2004, "JSIMS CCSE Performance Analysis." In proceedings of the *Fall 2004 Simulation Interoperability Workshop*, paper 04F-SIW-34.

HiGrids are a recursive data structure that has been used for filtering publication and subscription regions defined by abstract multidimensional spaces with multiple resolutions. Each dimension in the HiGrid is characterized during its construction by one or more resolutions that can be tuned for optimal performance and filtering efficiency. The dimensions can be parallelized to support scalable filtering. HiGrids dimensions are specified with a set of fixed resolutions that are chosen to best represent the needs of an application. This approach is in contrast to other binary data structures, such as oct-trees, that rely on bisection techniques. HiGrids function more like a direct hashing data structure that directly finds what it is looking for rather than binary (log base-two) searches. The direct hashing property makes HiGrids much more efficient than other data structures. An example of a simple Interval Dimension with multiple resolutions is shown in Figure 14. A much more elaborate three-dimensional spatial position dimension is provided by the HiGrid data structure that is designed for Earth (or planetary) types of problems. It supports publish and subscribe properties of entities such as submarines located under the sea, ships on sea surfaces, ground units on the land, aircraft moving within the atmosphere, and entities located in space at different altitudes and spatial resolutions.

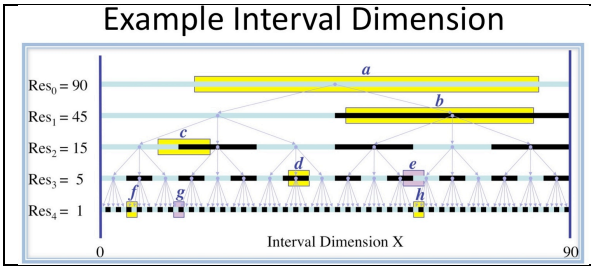


Figure 14: Example of an Interval Dimension in a HiGrid with multiple resolutions. In this example, dark shaded regions e and g represent publication regions, while a, b, c, d, and f represent subscription regions. Vertically (with binning defining resolutions), e overlaps h (below) and b and a (above). Similarly, g overlaps c and a. So, subscribers a, b, and h discover publisher e, while subscribers a and c discover publisher g.

Implementation

The space debris simulation was implemented as two separate efforts. First, critical extensions were made to the HiGrid data structure to distribute user defined multidimensional regions for abstract spatial dimensions across nodes and to provide parallel lookups in a scalable manner. This first step is necessary to implement full publish and subscribe scalable filtering capabilities in preparation for future use in Phase II. It should be noted that several defense-related programs (e.g., DIA ITASE, MDA I-Sim, DARPA SAFE-SiM, Air Force AFSIM) have expressed significant interest in this capability. It is anticipated that this work in Phase II will be augmented using funds from one or more of these defense programs. NASA will reap long-term benefits from this work in the future.

The second effort was to extend the Missile Defense Engagement Model (MDEM) that was developed for the Missile Defense Agency to include space debris calculations. This was done by creating 32 instances of the space debris model, where each instance randomly generated 200 particles (total of 6,400 particles) moving in random elliptical orbits around the earth (see Figure 15). Each space debris model subscribed to all satellites in space, Inter Continental Ballistic Missiles (ICBMs) with nuclear warheads, and Ground Based Interceptors (GBIs). It is noteworthy to mention that satellite and launched missile motion does not change, which means that the closest approach calculation for space debris happens just once. However, interceptors continually change their motion as track updates (used to guide each interceptor towards its warhead target) are received by the interceptors. The bulk of the space debris workload comes from recalculating closest approach for interceptors as they change their motion. Figure 16 provides a histogram of closest approach calculations for interceptors. It should be noted that no space debris particles impacted any of the missiles or satellites in this missile defense simulation. This is partly due to the fact that only 6,400 space debris particles were modeled. But even when scaling up to 300,000 or more particles, space debris impacts are rare in the real world.

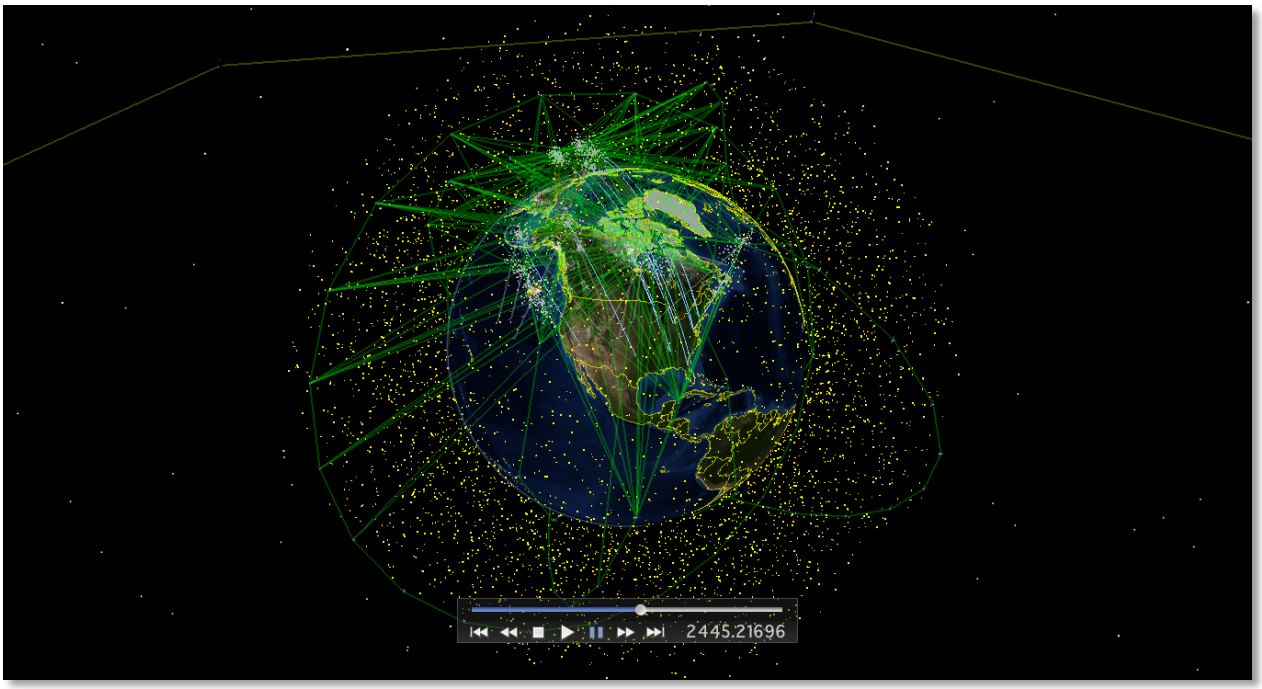


Figure 15: Space debris integrated with a missile defense scenario involving 20 launched ICBMs – each with multiple independently aimed nuclear warheads, 27 low earth-orbiting satellites with IR sensors, 60 medium altitude earth-orbiting satellites with IR sensors, 6 geosynchronous satellites with IR sensors, 4 long-range ground-based radars, and 4 interceptor sites. Interceptors also had IR sensors, which fed into the track fusion algorithm (blue trajectories show track projections to their impact point).

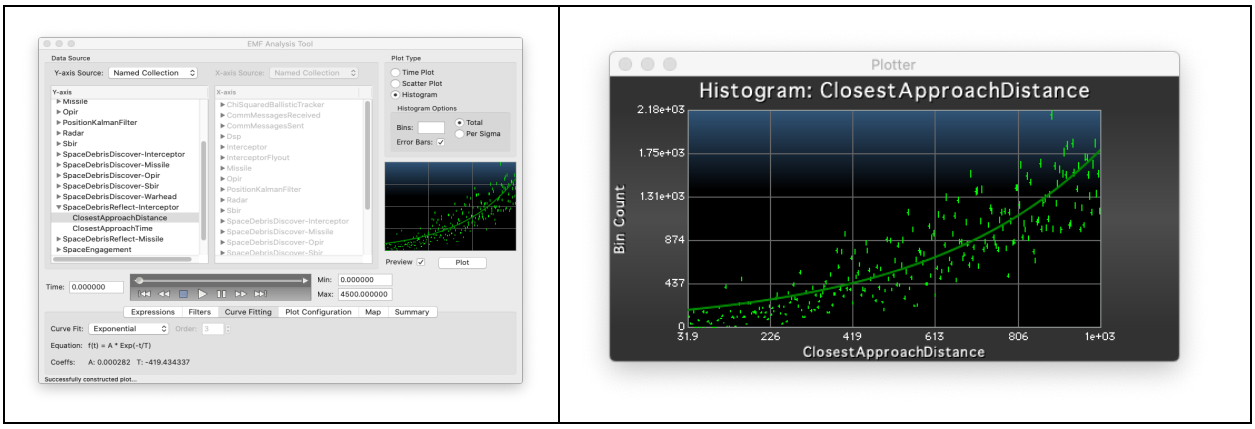


Figure 16: Histogram of closest approach calculations for space debris and interceptors created from the WarpIV Analyzer from generated bit-compressed data log files collected during the simulation execution. The data is curve fitted by the WarpIV Analyzer to an exponential distribution.

The closest approach algorithm in the WarpIV Kernel was re-fitted to support the space debris calculations. It is important to point out that dynamic motion in published federation objects (e.g., for satellites, missiles, and interceptors) is represented as a list of consecutive motion sequences, each with a start and end time. So, the closest approach algorithm that previously operated between two motion segments needed some minor modifications to work with lists of motion segments in federation objects.

All of this work will be reused by the Missile Defense Agency in efficiently computing closest approach of space debris dynamically created when interceptors engage their target warheads. The space debris problem is

extremely important for missile defense simulations when interceptor salvos engage closely spaced targets and fratricide potentially occurs.[54] So, the work performed in this NASA effort will transition to missile defense (and potentially other programs as well).

Performance

The space debris model was integrated with the Missile Defense Engagement Model (MDEM) using a non-classified test scenario that executed on an 8-core MacBook Pro laptop and a 32-core AMD Ryzen Threadripper Linux machine. Performance results are shown with and without data logging in the figures below.

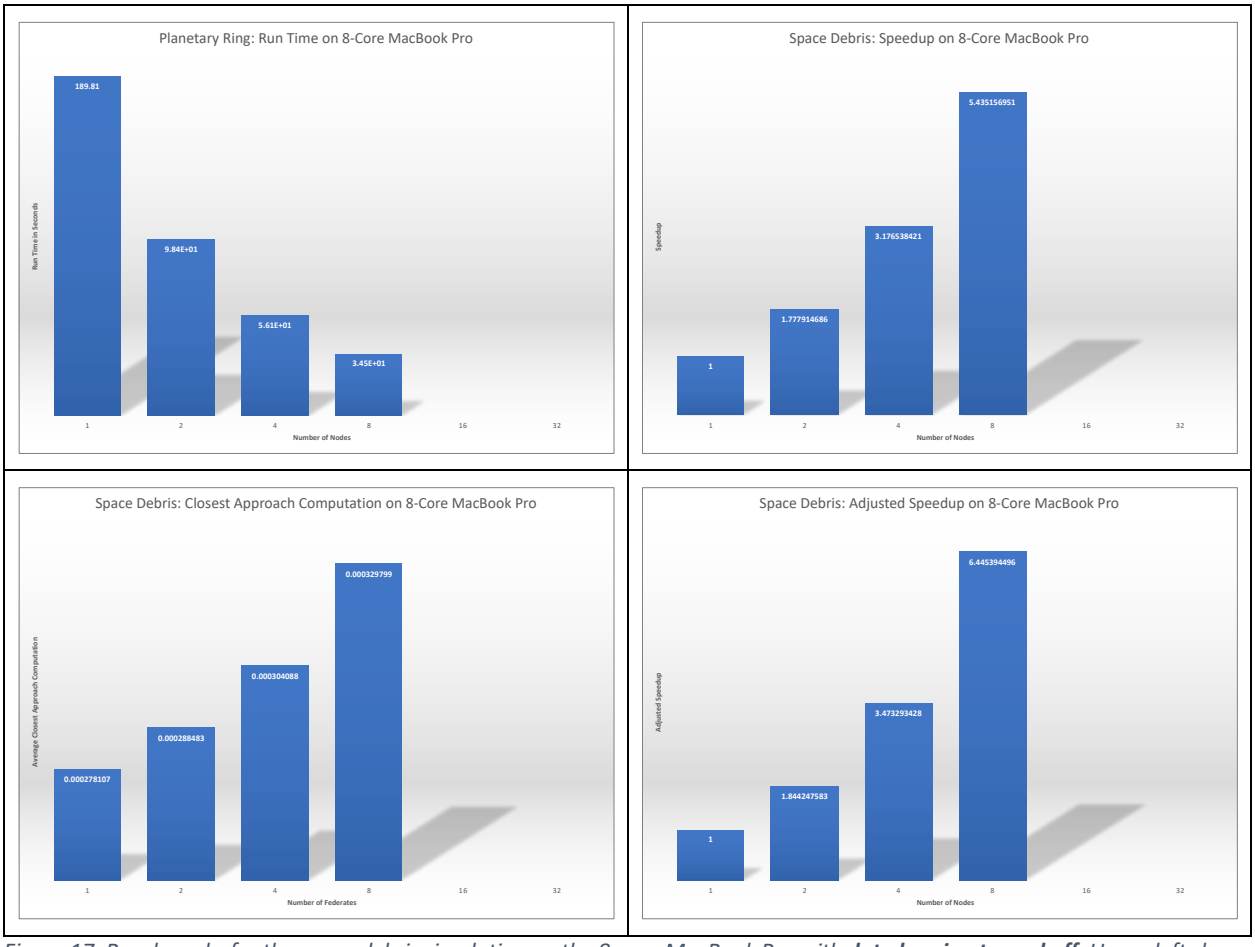


*Figure 17: Benchmarks for the space debris simulation on the 8-core MacBook Pro with **data logging turned off**. Upper left shows the execution time, Upper Right shows speedup. Lower left shows how computational performance of the most expensive event increased due to Cache/RAM contention issues with the CPU and motherboard. Lower right shows adjusted speedup where the Cache/RAM contention affects are factored out. The adjusted speedup indicates the speedup that would be achieved if there were no computational slowdown effects due to Cache/RAM issues.*

[54] An impact between an interceptor and a warhead creates very high-energy space debris particles that can impact other interceptors as they engage nearby targets. This expensive calculation in parallel and distributed missile defense simulations can overwhelm performance when done without filtering based on proximity.

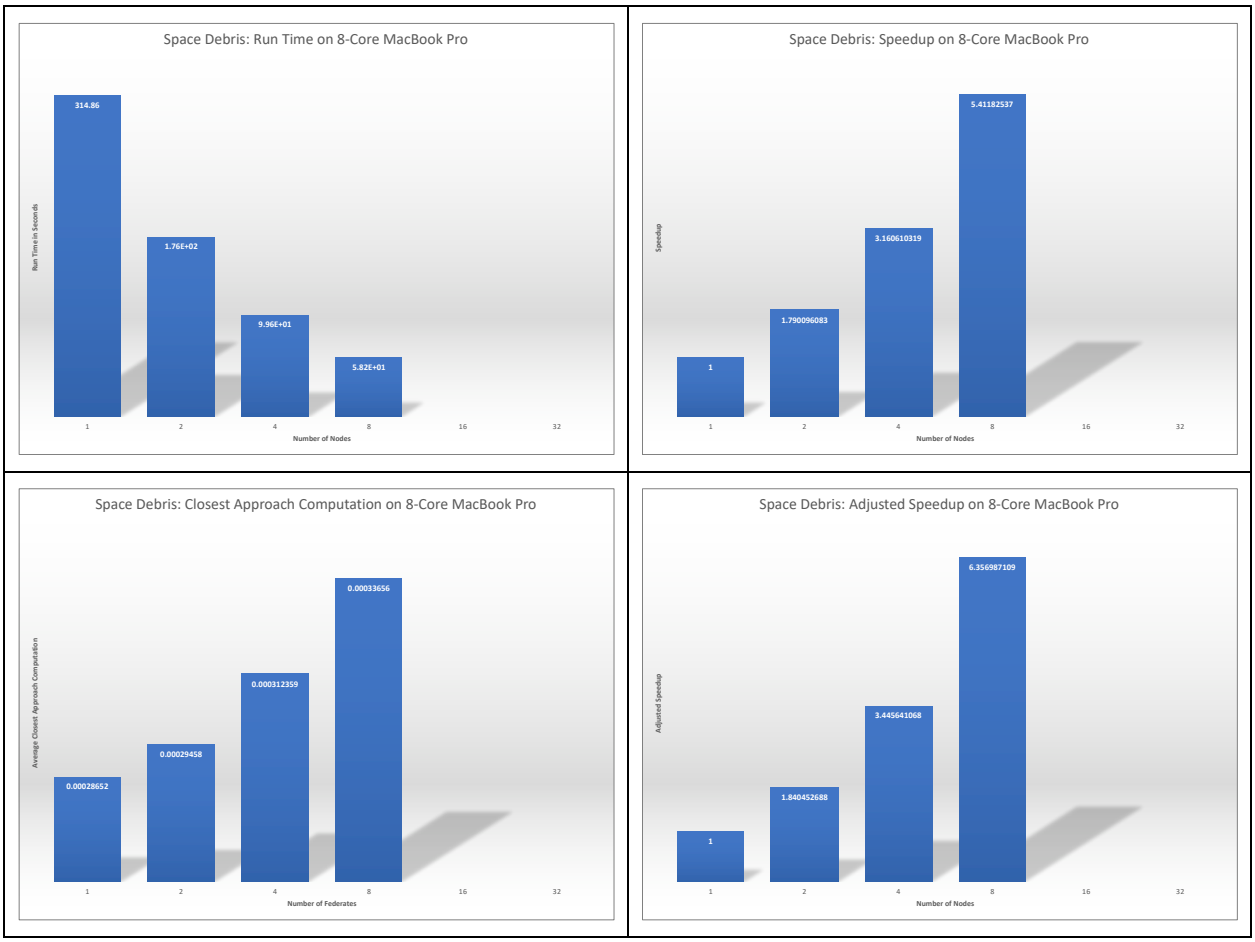


*Figure 18: Benchmarks for the space debris simulation on the 8-core MacBook Pro with **data logging turned on**. Upper left shows the execution time, Upper Right shows speedup. Lower left shows how computational performance of the most expensive event increased due to Cache/RAM contention issues with the CPU and motherboard. Lower right shows adjusted speedup where the Cache/RAM contention affects are factored out. The adjusted speedup indicates the speedup that would be achieved if there were no computational slowdown effects due to Cache/RAM issues.*

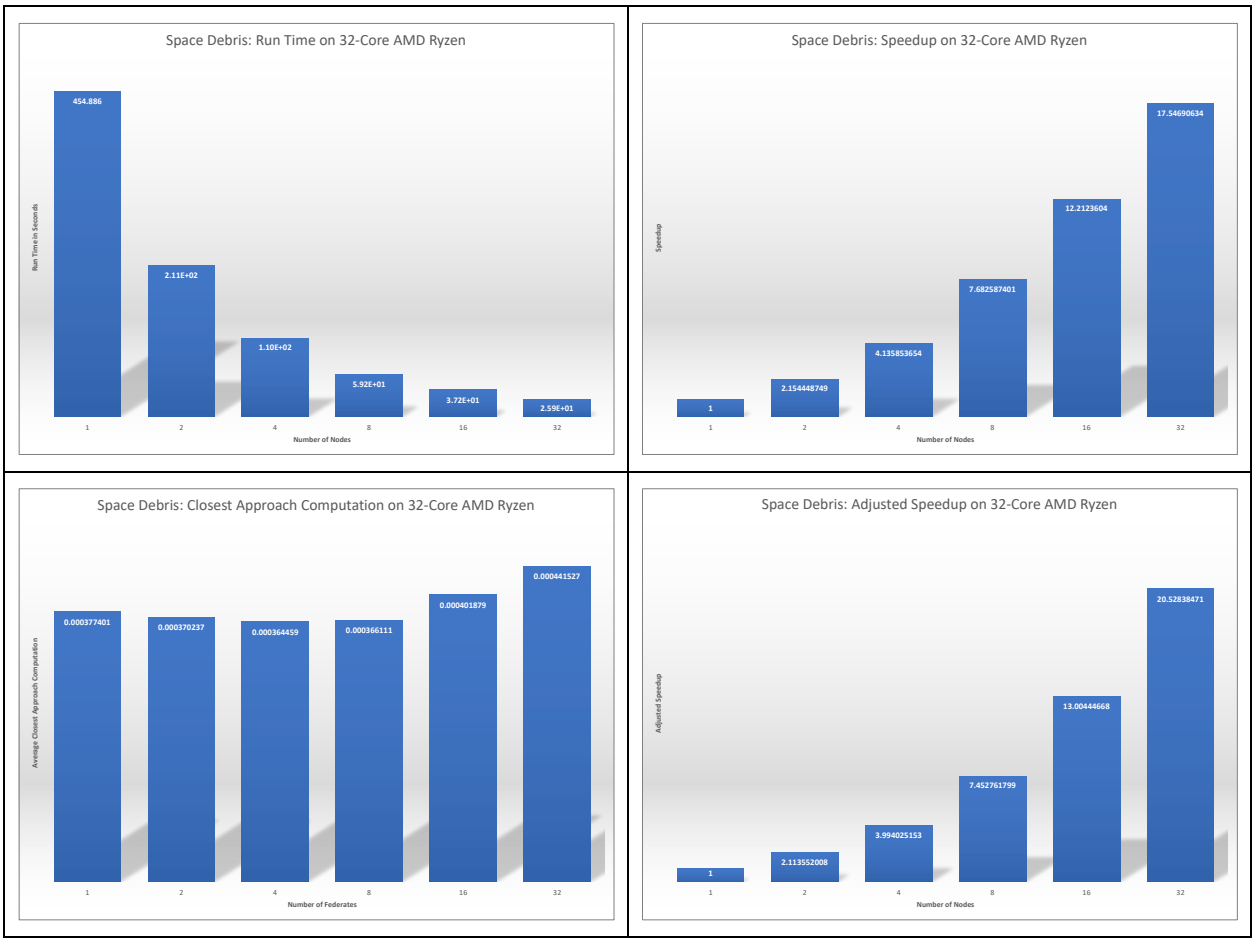


*Figure 19: Benchmarks for the space debris simulation on the 32-core AMD Ryzen Threadripper with **data logging turned off**. Upper left shows the execution time, Upper Right shows speedup. Lower left shows how computational performance of the most expensive event increased due to Cache/RAM contention issues with the CPU and motherboard. Lower right shows adjusted speedup where the Cache/RAM contention affects are factored out. The adjusted speedup indicates the speedup that would be achieved if there were no computational slowdown effects due to Cache/RAM issues.*

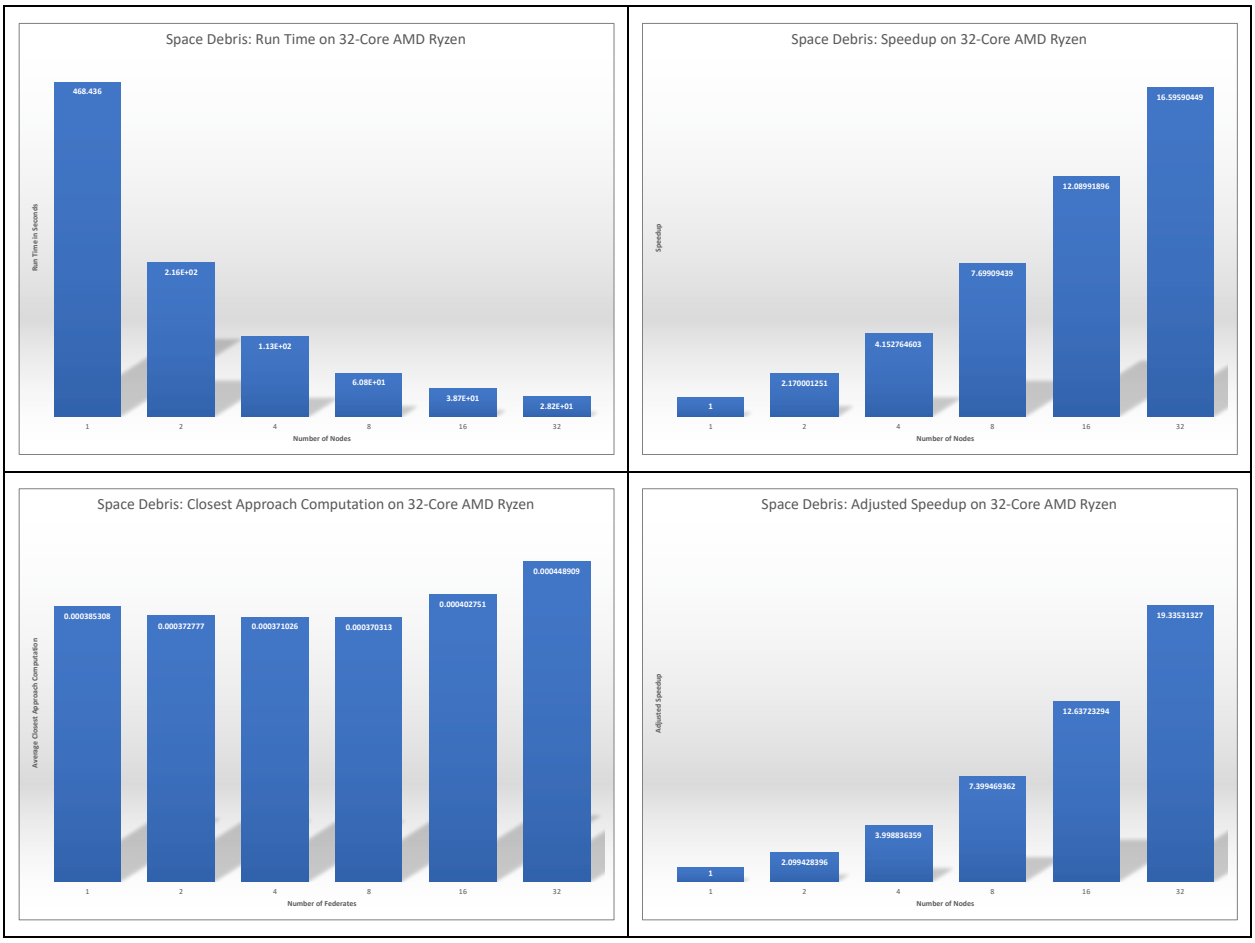


*Figure 20: Benchmarks for the Planetary Ring simulation on the 32-core AMD Ryzen Threadripper with **data logging turned on**. Upper left shows the execution time, Upper Right shows speedup. Lower left shows how computational performance of the most expensive event increased due to Cache/RAM contention issues with the CPU and motherboard. Lower right shows adjusted speedup where the Cache/RAM contention affects are factored out. The adjusted speedup indicates the speedup that would be achieved if there were no computational slowdown effects due to Cache/RAM issues.*

Critical path analysis of the space debris simulation showed the potential for unlimited speedup as the number of space debris particles increases.

Fourth Prototype: Electromagnetic Radio Frequency Propagation

Background

In an R&D effort performed under a contract with the Office of the Secretary of Defense (OSD) Software Producibility Program in 2013,[55] WarpIV Technologies, Inc. developed a scalable and composable collection of models for representing Radio Frequency (RF) environments in support of a variety of electronic warfare scenarios. The prototyping effort was designed for analog radio communications, tactical data transmissions, radars, and jammers. The basic idea was to identify each element in the scenario capable of either sending, receiving, or reflecting RF transmissions. Propagated waveforms from transmitters were modeled as coded packets and/or continuous signals. All receiving or reflecting entities were automatically provided the summed superposition of all transmitted signals in the time domain. Doppler shifts were accurately accounted for with regard to moving entities that could dynamically change their motion at any time. Boundaries between successive packets were verified to be smooth and continuous. Virtually perfect run-time scalability was achieved in all high-performance

[55] This three-year R&D effort funded all of the internal WarpIV model composability mechanisms that will be used in this SBIR effort.

computing benchmarking efforts. Results from 2013 are shown in Figure 44, indicating high performance for even a modest simulation with 120 transmitters and 120 receivers.

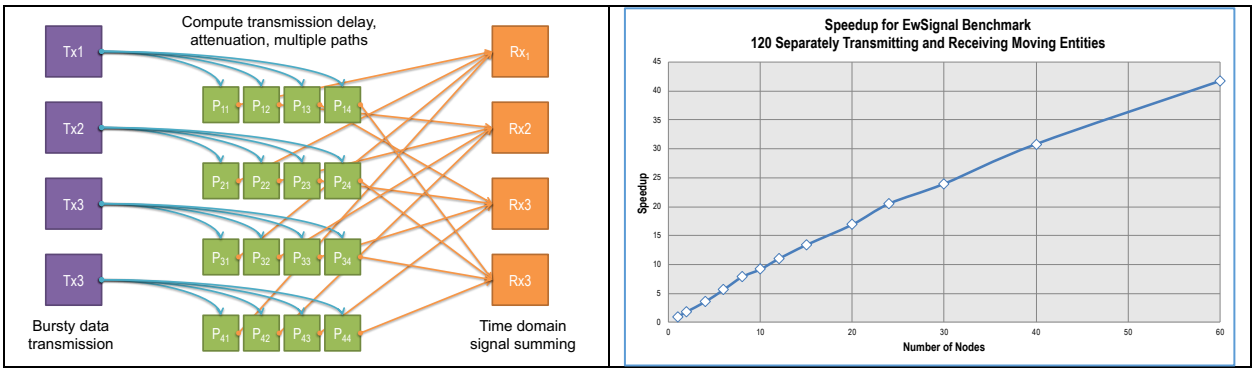


Figure 21: RF Propagation Model prototype in the WarpIV Kernel. The left figure shows how transmitters send their signals through a propagation model to all receivers where the superimposed signals are summed in the time domain as they are sampled. The figure on the right shows speedup in the low 40's on a 60-core machine involving 120 separately transmitting and receiving moving entities.

The WarpIV RF propagation approach is very different from the more traditional RF propagation methods that are commonly used where signals are propagated throughout all regions of space, which are represented as three-dimensional grid cells and are reflected by objects as encountered. The WarpIV approach only propagates signals along paths that actually affect receivers, which is why the propagation model achieves much better performance.

Modeling radio frequencies generated by arbitrary moving transmitters and receivers could be extremely important for NASA in understanding signal-to-noise ratios for (a) space-to-earth communication systems (e.g., the Deep Space Network), (b) radar tracking systems on the ground of orbiting satellites, rockets, and space debris, and (c) low-signal radio astronomy experiments. A prototype effort proposed for Phase II will modify the previous scenario involving moving aircraft to NASA-oriented scenario with ground-based receivers with satellite transmitters. Perhaps one or more spacecraft will also be integrated into the scenario to send and receive RF signals. It is expected that this work will be a relatively small effort since the basic infrastructure is already in place. The goal in Phase II is to demonstrate scalability and speedup for scenarios that would be interesting to NASA space scientists.

Implementation

Phase I used the same implementation and scenario (120 transmitting and 120 receiving aircraft moving randomly within a confined spatial region) test configuration of the RF propagation simulation that was developed for OSD. Future extensions could include space missions with spacecraft signals received by the Deep Space Network competing with other RF signals from a variety of transmitting sources.

Performance

The RF propagation model that was developed for OSD to simulate electronic warfare signal processing was benchmarked on an 8-core MacBook Pro laptop and a 32-core AMD Ryzen Threadripper Linux machine. Data logging was not used in these measurements. Performance results are shown in the figures below. It is important to note that the benchmarks performed in 2013 achieved speedups in the low 40's on an HPC machine at SPAWAR San Diego using 60 cores, where there were no significant cache/RAM bottlenecks (see Figure 21). The speedup on the WarpIV AMD Ryzen Threadripper machine of 12.86232481 on 32 cores is not consistent with the high speedup measured in 2013. However, the adjusted speedup that factors out the cache/RAM bottleneck is perfectly consistent with the 2013 results. This lends confidence that all of the benchmarks performed in Phase I would achieve speedups consistent with the adjusted speedup when executing on higher-end HPC machines.

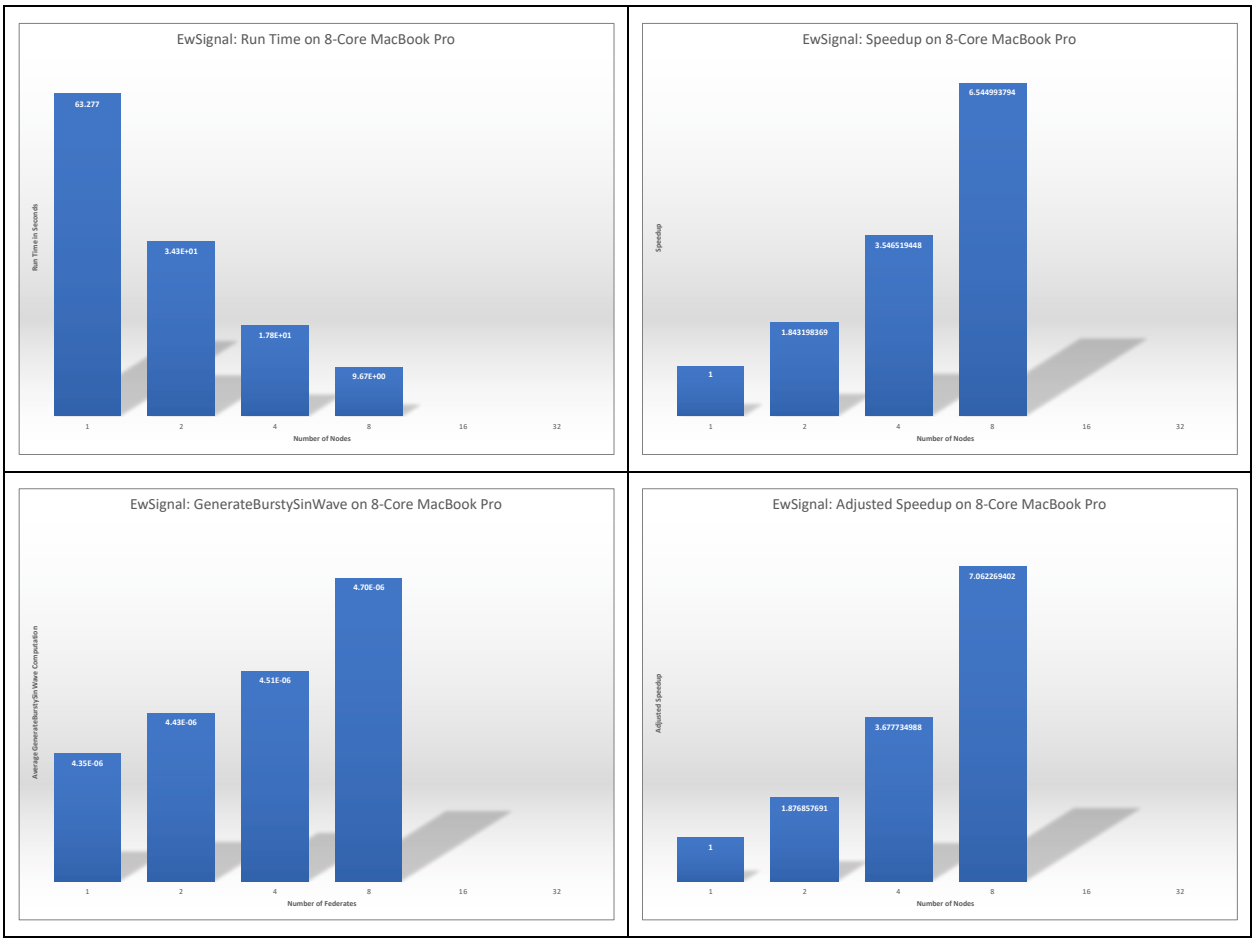


Figure 22: Benchmarks for the RF propagation simulation modeling electronic warfare on the 8-core MacBook Pro (note, there was no data logging in this simulation). Upper left shows the execution time, Upper Right shows speedup. Lower left shows how computational performance of the most expensive event increased due to Cache/RAM contention issues with the CPU and motherboard. Lower right shows adjusted speedup where the Cache/RAM contention affects are factored out. The adjusted speedup indicates the speedup that would be achieved if there were no computational slowdown effects due to Cache/RAM issues.

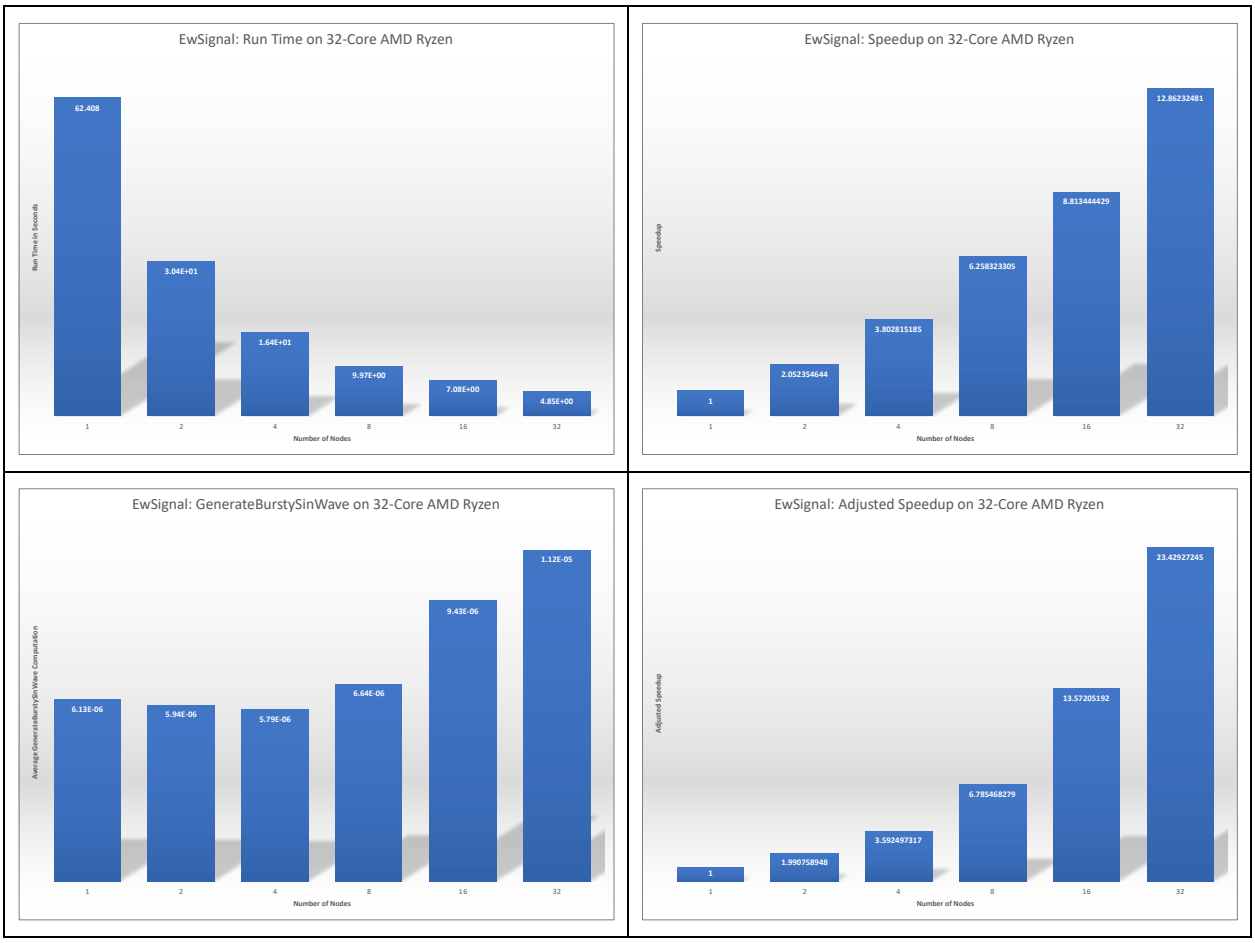


Figure 23: Benchmarks for the RF propagation simulation modeling electronic warfare on the 32-core AMD Ryzen Threadripper (note, there was no data logging in this simulation). Upper left shows the execution time, Upper Right shows speedup. Lower left shows how computational performance of the most expensive event increased due to Cache/RAM contention issues with the CPU and motherboard. Lower right shows adjusted speedup where the Cache/RAM contention affects are factored out. The adjusted speedup indicates the speedup that would be achieved if there were no computational slowdown effects due to Cache/RAM issues.

Related R/R&D

This proposal previously described the background for developing an N-body gravitational simulation of planetary rings. The general strategy for how to develop this simulation was based on technical discussions with the world's top planetary ring experts in 1995. In addition, this proposal leverages HPC work performed on previous R&D efforts for the Department of Defense to calculate gravitational forces for ellipsoidal planets and to simulate RF propagation between moving entities. The space debris simulation leverages publish and subscribe R&D involving HiGrids successfully demonstrated with SPEEDES funded by the High-Performance Computing Modernization Office, Missile Defense Agency, and the Joint Simulation System. One of the original SPEEDES (NASA) patents registered by Dr. Steinman while working for the Jet Propulsion Laboratory (JPL) involved scalable techniques for determining publish and subscribe overlap regions in parallel discrete-event simulations. The original work (parallel proximity detection) received a best paper award[56] from the top conference (Parallel and Distributed Simulation, PADS 1995) in parallel simulation. WarpIV Technologies, Inc. is an acknowledged leader in scalable, composable, and intelligent high-performance discrete-event simulation technologies for the Department of Defense.

[56] Steinman Jeff and Wieland Fred, 1994. "Parallel Proximity Detection and the Distribution List Algorithm." In proceedings of the *1994 Parallel and Distributed Simulation (PADS) Conference*. Pages 3-11.

Path for Phase II and Phase III

All objectives for Phase I were successfully completed as planned. The next steps in Phase II are outlined below:

- Deliver the WarpIV Kernel to Ames Research Center and other NASA organizations or contractors
- Install the WarpIV Kernel on one or more NASA HPC platforms and run benchmarks
- Work with Ames Research Center to provide training and technical support for new users
- Team with NASA researchers to utilize the WarpIV Kernel for CFD problems
- Extend the Planetary Ring simulation to support higher fidelity particle interactions
- Use hierarchical grids to support scalable publish/subscribe services
- Apply the RF propagation model to one or more NASA space missions involving the Deep Space Network
- Communication benchmarks of the WarpIV Kernel and MPI
- Introduce NASA users to other sponsors of the WarpIV Kernel to leverage funding opportunities

In preparation for Phase III, capabilities of the WarpIV Kernel will be assessed to determine priorities (and leveraged funding by other sponsors) for continued R&D of the WarpIV Kernel (e.g., performance optimizations, graphical tools for composing simulations, editing models, visualizing and analyzing results, troubleshooting and debugging, documentation, training materials, etc.). Phase III will focus on remaining high-priority items while also providing technical support for NASA users.

Commercialization for NASA Purposes and the Department of Defense

This section describes the high-level commercialization strategy for the WarpIV Kernel described in this report. While most of our sponsors have been from the DOD, we hope to branch into other areas such as NASA science and spacecraft simulations, Department of Energy supercomputing problems, and large-scale commercial simulations. In addition, the WarpIV Kernel could be used to simulate the spread of diseases in high fidelity, such as the global Coronavirus pandemic. First, a description of ongoing programs and sponsors that have expressed interest in WarpIV Technology's high-performance computing capabilities is identified. Second, potential users of the proposed technology developed in this effort are discussed in more detail. Finally, the general goals and guiding principles of WarpIV Technologies, Inc. R&D is provided to describe the company's long-term open-source business strategy. The scalable performance and model composability technologies described in this report are universally needed by all of the United States armed force and intelligence agencies. These capabilities are also needed in the private sector for modeling very large-scale systems.

WarpIV Technologies, Inc. Programs and Sponsors

Past sponsors, collaborators, and/or current users of the WarpIV Kernel include: Navy (NRL, SPAWAR, NAWS China Lake), Army (ARL, MATREX, STTC, HRED, AMSCO), Air Force (AFRL/RL), Chemical Biological Radiological and Nuclear Defense (CBRND), Defense Threat Reduction Agency (DTRA), Defense Intelligence Agency (DIA), Missile & Space Intelligence Center (MSIC), Missile Defense Agency (MDA), Defense Advanced Research Projects Agency (DARPA), Joint Space Operational Center Mission System (JMS), U.S. Strategic Command (STRATCOM), Marine Corps Expeditionary Biological Test Bed, T&E/S&T Net Centric Systems Test (NCST), Defense Modeling & Simulation Coordination Office (DMSCO), High Performance Computing Modernization Program (HPCMP) – User Productivity Enhancement Technology Transfer and Training (PETTT), Office of the Secretary of Defense (OSD) Software Producibility Program, NASA/JPL, Scripps Research Institute (SRI), Institute for Simulation and Training (IST), Naval Postgraduate School (NPS), Johns Hopkins Applied Physics Laboratory (APL), Oakridge National Laboratory (ORNL), numerous partners from industry, and several academic institutions. WarpIV Technologies, Inc. has worked with most of the large commercial aerospace companies.

Potential Users of this Proposed Technology

The OpenUTF and its WarpIV Kernel reference implementation is being prepared for eventual standardization and widespread use across multiple DoD agencies and programs. The OpenUTF was a featured technology promoted by DMSCO at several "M&S in the Cloud" technical exchange meetings, the Congressional M&S Expo held in

Washington D.C., and at the I/ITSEC kiosk in 2016. The Army Research Laboratory (ARL) recently hosted the French M&S Delegation, where a briefing on the OpenUTF was specifically requested by the French government. ARL, DMSCO, and AMSCO (representing the Army) are currently investigating next generation M&S framework technology based on the OpenUTF Roadmap paper that was published in 2013. The ITASE program has a requirement to support as many as 10,000 federates. To reduce its computational footprint, ITASE is considering adopting the OpenUTF composable modeling framework for next-generation model development. A new M&S group at China Lake has been trained to develop models for the OpenUTF and has already begun developing composable models for the Navy. University of Central Florida is developing OpenUTF models for several R&D efforts. The primary end-user of this particular effort, however, will be the NASA simulation and high-performance computing community. WarpIV Technologies, Inc. will use its one-week hands-on training materials to bring NASA up to speed in the use of the WarpIV composable modeling methodology.

NASA Users

An overview of the WarpIV Kernel and the key work that was performed during this effort was briefed to the NASA supercomputing community (hosted by NASA Ames Research Center) on January 28, 2021. This briefing was well attended with lots of time for questions and answers at the end of the seminar. There have been several follow-up emails from participants who are interested in various aspects of the WarpIV Kernel (e.g., its high-speed communications, parallel discrete-event simulation engine, bit-compressed data logging, planetary ring simulation, etc.). We plan to continue coordinating with the NASA Advanced Supercomputing (NAS) Division at Ames Research Center in Phase II with the goal of installing the WarpIV Kernel on one or more HPC machines, provide hands-on training for NASA HPC R&D technical staff and contractors, perform a variety of benchmarks on large machines, explore new discrete-event techniques for modeling CFD problems, etc.

We believe that the first step in creating a WarpIV user-base within NASA is to bring the technology to the NASA HPC community at Ames, where common HPC applications such as CFD problems can be developed using discrete-event techniques. Ames will then be able to offer this capability to the rest of NASA's M&S community, which would likely include a variety of mission, test, and space-science simulation programs. The outreach by Ames would promote the advanced simulation capabilities of the WarpIV Kernel modeling framework (which simplifies the development of complex simulations) with promise of scalable performance, composability to promote interoperability and reuse, support for big-data using the bit-compressed data logger and future cataloging system, strategy and core infrastructure to integrate and parallelize existing simulations, and many other benefits. As several defense programs are realizing, migrating to a common simulation engine is the key to reducing software life-cycle costs. We believe that the common simulation engine should be open source, cross-platform, scalable with advanced technology to support machines ranging from multicore laptops to the world's largest supercomputers, be able to support very large volumes of data, and provide modeling constructs that support advanced artificial intelligence techniques. All of these factors make the WarpIV Kernel the perfect foundation for this important transformation.

The abstract for the seminar presented by WarpIV Technologies to NASA is provided below:

Advanced Modeling & Simulation (AMS) Seminar Series
Introduction to the WarpIV Kernel

Speaker: **Jeffrey S. Steinman, Ph.D.**, President & CEO, WarpIV Technologies, Inc.
Date: January 28 at 9:00 AM Pacific Time, 2021

Abstract

The WarpIV Kernel is a vast collection of scalable high-performance computing software tools primarily designed to support Parallel Discrete-Event Simulation (PDES) for defense-related applications. It is free and open source (with some restrictions on redistribution), runs on all mainstream platforms (Linux, Mac OS X, other variants of UNIX, and Windows), supports all widely used C++ compilers, is fully self-contained (more than 825,000 lines of application-neutral

code) with no third-party software dependencies, and is exportable to foreign countries that are friendly to the United States. It runs on machines ranging multicore laptops to the world's largest supercomputers. Some of the capabilities of the WarpIV Kernel include: (a) high-performance communications, (b) advanced synchronization techniques for optimistic discrete-event simulation with dynamic flow control to maintain stability, (c) advanced modeling constructs to simplify the development of composable and intelligent models, (d) publish and subscribe services to efficiently share data and schedule events between distributed models in logical time, (e) large assortment of software utilities involving math, motion, random number generation, data structures, algorithms, and data parsers (f) scalable infrastructure for federating simulations (including self-federating to achieve parallel speedup for sequential simulations), (g) bit-compressed data logging for parallel and distributed applications that generate large volumes of data, (h) a variety of tools for debugging and profiling parallel applications including a critical path analysis of simulations to determine maximum inherent speedup, (i) Parallel Application Launching (PAL), (j) distributed grid-computing infrastructure, (k) optimization engine (based on grid computing) for launching and intelligently guiding goal-oriented M&S excursions, (l) various graphical tools for constructing cross-platform user interfaces, developing composable models, visualizing scenarios, and mathematically analyzing results, and (m) five-dimensional simulation that executes multiple timeline branches in a single parallel excursion, where maximal computation sharing is provided between parallel universe timelines. This introductory talk will provide a high-level overview of these WarpIV Kernel capabilities and then demonstrate recent work performed for NASA on a Small Business Innovative Research (SBIR) Phase I effort where four very different NASA-related applications were quickly developed: (1) simulation of planetary rings as an n-body gravitational system, (2) simulation of space debris with closest approach calculations in a missile-defense scenario involving launched rockets, interceptors, and satellites, (3) numerical computation of gravitational fields for ellipsoidal planets (e.g., Saturn), and (4) superposition of radio frequencies involving large numbers of moving transmitters and receivers where packetized signals are digitized.

In follow up discussions with Ames Research Center and the organizers of the AMS Seminar Series, WarpIV Technologies is planning to present at least three more seminars on these topics: (1) optimistic time-management with flow control for parallel discrete-event simulation in the WarpIV Kernel, (2) bit-compressed data logging for parallel and distributed applications generating large volumes of data, and (3) the WarpIV High-Speed Communications library. We believe these three topics will be of great interest to the NASA AMS community. The first topic that will likely be presented is the bit-compressed data logging infrastructure. Its capabilities can be integrated into existing parallel and distributed applications with minimal software impact. These seminars will help bring introduce the AMS community to many of the capabilities of the WarpIV Kernel.

We acknowledge the challenge of bringing new technologies to existing programs without disrupting schedules and/or funding constraints. With this in mind, WarpIV Technologies plans to aggressively create and pursue opportunities for bringing the technologies of the WarpIV Kernel to NASA programs. For technology transition efforts to succeed, it is important to have the technology in place, fully functional, documented, and ready to be used when opportunities arise.

General Goals and Guiding Principles of WarpIV Technologies, Inc.

WarpIV Technologies primary goal is to provide high-performance computing technologies to a large community of DoD agencies, programs, and users. Funding is sought to (a) mature the technologies through cutting-edge R&D efforts that benefit large communities, and (b) to provide technical support on programs using the OpenUTF. WarpIV Technologies, Inc. is committed to the following open-source software principles:

- Commitment to the free and open-source software philosophy for widespread scientific applications
- Secure R&D funds from government contracts and programs rather than seeking private funding
- Develop graphical tools that could potentially be commercialized to support the simulation community
- Form alliances and partnerships with government laboratories, research institutions, industry, and academia
- Utilize existing standards while developing next-generation standards for the international M&S community
- Openly publish technologies at conferences, symposiums, journals, technical meetings, training events, etc.
- Relinquish all proprietary data rights and not seek patents for new technologies to promote an open culture

Summary and Conclusions

The work performed in Phase I was completely successful and showed how scalable performance can be achieved for a wide variety of problems using the WarpIV Kernel. Each of the four applications demonstrated unbounded speedup, while also revealing the hardware limitations of commodity (affordable) CPUs (e.g., 32-core AMD Ryzen Threadripper) due to memory (cache/RAM) bottlenecks that have not been observed in prior benchmarks with other more advanced HPC processors.

It is important to recognize that the Phase I effort intentionally did not focus on Computational Fluid Dynamics (CFD) problems, which generally dominate HPC use. Rather, Phase I explored other types of diverse problems with an outreach goal of bringing non-traditional HPC use cases such as mission planning and spacecraft testing to the world of HPC. We believe that broad use of the WarpIV Kernel will not only expand the HPC user base, but also lower the life-cycle cost of developing/supporting complex simulations because of its (a) advanced modeling framework that reduces lines of model code, and (2) promotion of interoperability and reuse. This effort also laid out a roadmap/strategy for integrating all types of NASA simulation use cases into a single parallel execution operating on HPC platforms using (a) the WarpIV Kernel simulation engine and (b) the HPC-RTI and LrcProxy Framework to federate (and potentially parallelize) legacy simulations.

However, it is important for us to not forget traditional HPC users, which includes a variety of CFD models. We would very much like to address CFD problems in Phase II where discrete-event processing techniques (similar to the Planetary Ring simulation developed in Phase I) could be utilized in a similar manner to reduce how often computations are performed while still producing more accurate results. Success in parallelizing CFD problems using discrete-event techniques could provide a new foundation for providing orders of magnitude performance gains. We believe the likelihood for success in Phase II is high.

Key Personnel

WarpIV Technologies, Inc. has assembled a small, but well-qualified experienced team of engineers and scientists to provide the core work on this effort. Dr. Jeffrey S. Steinman of WarpIV Technologies, Inc. will be the Principal Investigator and will lead all technical aspects of this effort in Phase I. Mr. Craig N. Lammers and Ms. Wendy L. Steinman will provide additional software development, visualization, testing, and benchmarking support. All members of the WarpIV Technologies, Inc. technical staff are United States citizens with secret clearances.

Dr. Jeffrey S. Steinman (Principal Investigator)

Education

- 1981: B.S. Computer Science (Magna Cum Laude), California State University Northridge (CSUN)
- 1981: B.S. Physics (Magna Cum Laude), California State University Northridge (CSUN)
- 1983: M.S. Physics, University of California Los Angeles (UCLA)
- 1988: Ph.D. High Energy Physics, University of California Los Angeles (UCLA)

Biography

Dr. Steinman, President and CEO of WarpIV Technologies, Inc. received his Ph.D. in High Energy Elementary Particle Physics from UCLA in 1988 where he studied the quark structure function of high-energy virtual photons at the Stanford Linear Accelerator Center. While studying physics at UCLA, Jeff Steinman worked at Hughes Aircraft Company from 1981-1984 in the Radar Systems Group, where he led multiple development efforts involving the F/A-18 and F-15 radar airborne tracking systems. Upon completing his Ph.D. dissertation, Dr. Steinman worked at the Jet Propulsion Laboratory/Caltech in the area of parallel and distributed supercomputing technologies for NASA. Dr. Steinman was the key developer of forecasting and prediction tools used by the Cassini Mission to compare actual and predicted signal strengths for the Deep Space Network (DSN). In 1990, Dr. Steinman created the Synchronous Parallel Environment for Emulation and Discrete Event Simulation (SPEEDES) framework as the next-generation replacement for the Time Warp Operating System (TWOS), which was also being developed by

JPL. This work transitioned to industry in 1996 when Dr. Steinman joined Metron to form its High-Performance Computing Division. SPEEDES eventually became the core infrastructure for several large Missile Defense and billion-dollar Joint M&S programs. Dr. Steinman joined RAM Laboratories in 2000 as Vice President, Chief Scientist, and Director of High-Performance Computing Programs. Dr. Steinman launched his own company, WarpIV Technologies, Inc., in 2005 where his focus has been the development and standardization of new high-performance parallel and distributed computing technologies based on the open source WarpIV Kernel and the Open Unified Technical Framework (OpenUTF). Dr. Steinman was the chair of the Parallel and Distributed Modeling & Simulation Standing Study Group (PDMS-SSG), has five patents in the area of high-performance supercomputing, [57] and has published more than 80 papers in the field of M&S (see footnotes throughout this proposal for relevant references). Dr. Steinman is the author of "The Roadmap" for next-generation M&S. He is recognized as a leading expert within the international community in the areas of scalable, composable, and intelligent parallel and distributed M&S technology, AI, numerical methods on supercomputers, modeling of physical systems, verification and validation, Big Data (logging, compression, and playback), design of experiments, statistics, multidimensional optimization, and game theory.

Mr. Craig N. Lammers

Craig Lammers, Vice President of Applied Research and Development at WarpIV Technologies, Inc., graduated with high honors in 2001 from the University of Michigan, Dearborn with a B.S. in Industrial & Systems Engineering. Craig Lammers started his career in 2002 at RAM Laboratories as a Software Engineer where he designed, developed, tested, and documented several software applications for the Joint Simulation System, Air Force Research Laboratory Rome Labs (AFRL/RL), and Missile Defense Agency (MDA). Craig Lammers served as Principal Investigator for an effort with AFRL using neural networks for anomaly detection and was the lead developer of the Dynamic Situation Assessment and Prediction (DSAP) multi-replication framework developed for AFRL. Craig Lammers joined WarpIV Technologies, Inc. in 2006 where he has focused on R&D in the areas of artificial intelligence, signal processing, real-time tracking, 3D visualization, estimation and prediction technologies, and COA planning. Craig Lammers supported the design and development of the next-generation ground control station for Predator UAVs, and played an active role defining infrastructure services for the UAV Control Segment Working Group (UCSWG). Craig Lammers developed the cross-platform 2D/3D visualization and analysis tools provided within the WarpIV Kernel. He was the program manager and lead engineer for a Phase II SBIR effort with MDA focused on resource optimization. Craig Lammers was the program manager for a follow-on Phase I SBIR effort with MDA to provide scalable COA planning based on grid-computing, genetic algorithms, particle swarm optimization, RSM with polynomial bisection, and neural networks. He also developed several new data plotting utilities for tracking optimization convergence progress of potentially large numbers of variables during COA planning. Craig Lammers has authored or co-authored more than 20 papers on M&S and related fields.

Ms. Wendy L. Steinman

Wendy Steinman, Software Engineer at WarpIV Technologies, Inc., graduated from the University of California, San Diego in 2009 with a B.S. in Cognitive Science and a minor in Biology. Upon graduation, Wendy Steinman joined WarpIV Technologies, Inc., as Software Engineer and Project Manager where she contributed to the design, development, testing, and documentation of the Open Cognitive Architecture Framework (OpenCAF) constructs in support of an Army R&D modeling effort with supercomputers. She has continued assisting with the design of the cognitive architecture framework, developing a variety of algorithms supporting thought triggering based on artificial intelligence. She developed key aspects of the data registry infrastructures in the WarpIV Kernel used by composites for data sharing between autonomous plug and play composable models. She also worked extensively on the mathematical interpreter to parse and execute expressions used by the WarpIV Analysis tool and cognitive thought-triggering constructs. Wendy Steinman is one of the key developers of the HPC-RTI and LrcProxy

[57] Dr. Steinman was awarded five patents for his innovative work in creating the SPEEDES framework: No. 5,652,871. "Parallel Proximity Detection for Computer Simulation." July 29, 1997. No. 5,781,762. "Parallel Proximity Detection for Computer Simulations." July 14, 1998. No. 5,794,005. "Synchronous Parallel Emulation and Discrete Event Simulation System with Self-Contained Simulation Objects and Active Event Objects." Aug. 11, 1998. No. 5,850,538. "Priority Queues for Computer Simulations." Dec. 15, 1998. No. 6,324,495. "Synchronous Parallel System for Emulation and Discrete Event Simulation." Nov. 27, 2001.

framework mechanisms that is being used by the Defense Intelligence Community ITASE program and has co-led hands-on WarpIV Kernel training efforts with the Navy at China Lake. She has performed numerous benchmarking studies to measure and model the performance of parallel and distributed simulations.

Appendix A: High Speed Communications

The High-Speed Communications (HSC) framework in the WarpIV Kernel integrates shared memory with reliable network protocols (TCP) to provide communication services between processing nodes. Recent benchmarks on the WarpIV AMD Ryzen Threadripper Linux machine show that more than 32 million short messages can be delivered per second (see Figure 24).

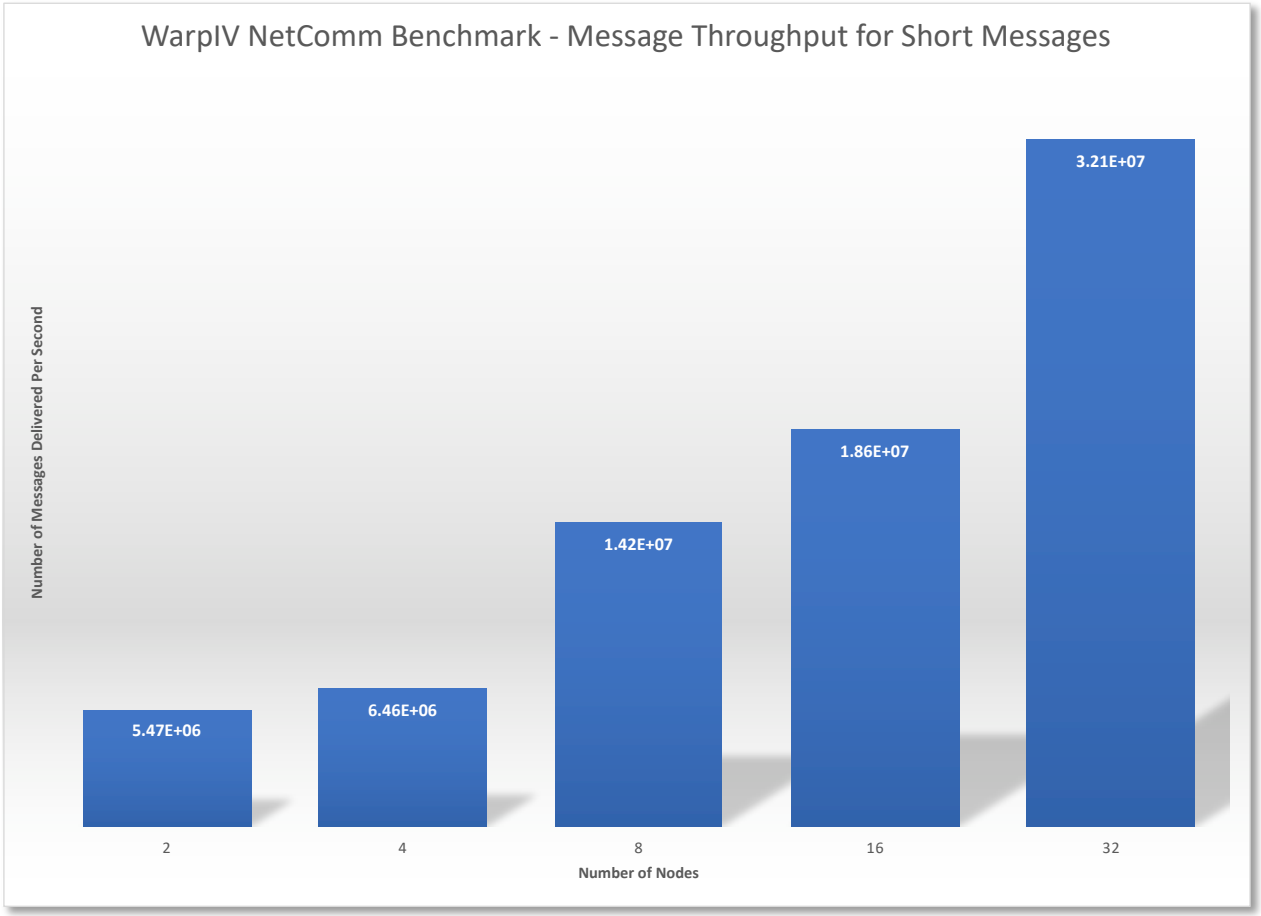


Figure 24: Throughput for small messages on a 32-core AMD Ryzen Threadripper Linux machine.

All network services between machines go through WarpIV servers (see Figure 25) that have been highly optimized to minimize overheads due to memory allocation, message copying, network congestion, etc. Messages between nodes on the same machine are carefully copied in and out of shared memory segments to achieve extremely high-speed performance. The servers carefully coordinate startup and shutdown processes to ensure robust operation. Shared memory segments are deleted after shutdown and network ports are always closed properly. The HSC starts up and shuts down almost instantaneously, even when operating across very large numbers of machines.

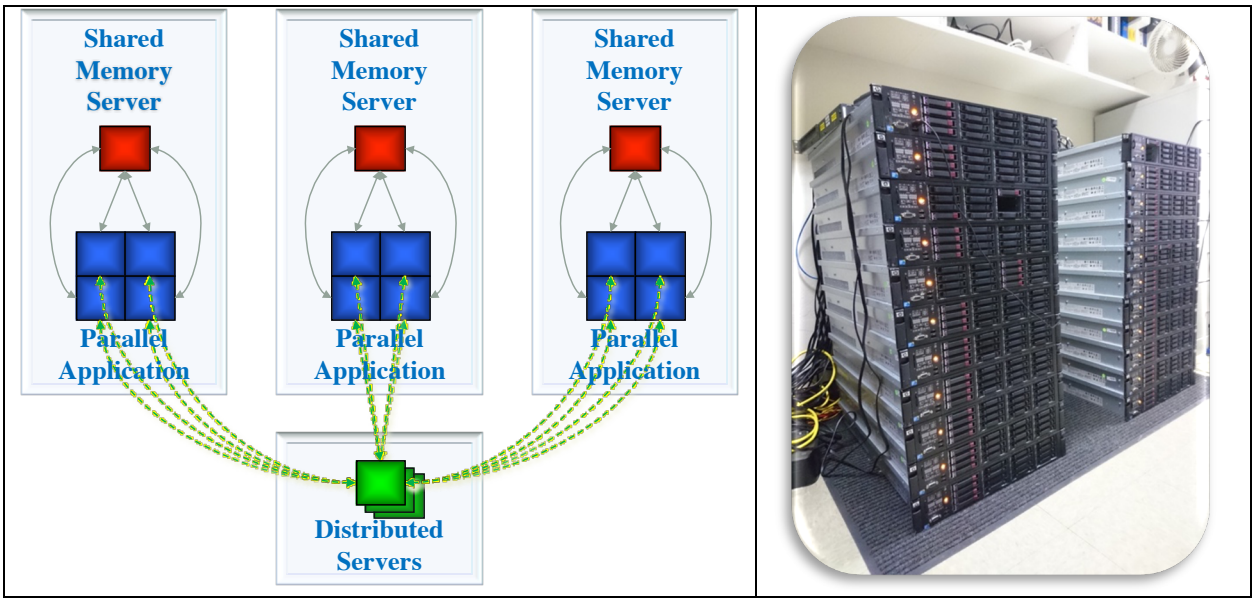


Figure 25: High Speed Communications (HSC) connects to shared memory servers (one per machine to manage the creation and deletion of shared memory segments used locally) and distributed servers (see left figure). The standard configuration is for each machine to have a distributed server that receives messages from local nodes and routes them to receiving nodes. When broadcasting or multicasting, the message is first sent by the sending node to its local distributed server, where it is then sent to all machines that have at least one node requiring the message. The message is then disseminated to all other nodes on the same machine that need to receive the message through shared memory. In this manner, a broadcast message from one node to all other nodes in a 21-machine, 11-nodes per machine setup with a Cisco Router (see right figure), takes only 20 network messages to deliver 230 messages. More than 23,000,000 messages per second were reliably delivered on this 10-year-old parallel computing cluster. This optimized way of broadcasting and multicasting is essential in providing scalability in real-time and logical time execution modes.

To maximize performance, all shared memory operations are performed without the use of semaphores or locks. Rather, the HSC relies on the cache coherency memory-management mechanisms that are provided by all modern shared memory multicore platforms. Reading and writing in shared memory is carefully coordinated to eliminate collisions. Communication between machines is always provided in a reliable manner through the use of distributed servers that can be configured in multiple ways. For example, when running on large HPC clusters or clouds, a distributed server resides on each machine to forward locally sent messages that are then routed across the network to other machines.

Many performance studies have been done throughout the years to compare the HSC with other message-passing libraries such as Parallel Virtual Machine (PVM), Message Passing Interface (MPI), SPEEDES, and various HLA RTIs. The HSC has always outperformed these other implementations by a large margin on multicore machines, especially as the number of nodes (or federates) gets large. Recent measurements performed on the Proliant HP cluster at WarpIV show that a federation with 231 federates operating on 21 machines is able to reliably deliver more than 23,000,000 short messages per second. It is estimated that more than 365,000,000 messages could be delivered per second on a modern cluster consisting of 16 machines with 128 cores per machine. Typical network-based RTIs break down with this many federates[58] and at best are only able to reliably deliver a few hundred thousand messages per second because they rely solely on network-based TCP/IP unicast communications without leveraging high-speed shared memory.

The HSC provides nine categories of services that are described in more detail in the following sections:

[58] Reliable communication with other RTIs broke down around 40 federates with long startup times and large message latencies, while the HPC-RTI is poised to support thousands of federates on large HPC clusters hosting multicore machines.

1. Startup and shutdown services
2. Clock synchronization and wall-time services
3. Miscellaneous services
4. Global synchronizations
5. Global reductions
6. Synchronized data distribution mechanisms
7. Asynchronous message passing
8. Coordinated message passing
9. Object Request Broker (ORB) services

Startup and Shutdown

Applications automatically begin parallel execution when they initiate the HSC through its startup interface. The startup service automatically forks (or spawns when running in Windows) processes to launch the requested number of nodes for parallel processing. Shared memory segments are created within the communications infrastructure to coordinate all of the inter-process communications services occurring on the same machine. Network connections, when executing in a cluster environment, are also established during the startup process. When applications have completed their work, the shared memory segments are deleted by the HSC through its terminate interface.

Distributed Clock Synchronization

After the communications are started up, the HSC goes through a carefully coordinated clock synchronization process that determines time offsets between clocks on different machines. Clocks are synchronized by essentially timing messages between machines and measuring their two-way delay. The offsets are abstracted by the HSC with an interface that allows applications to obtain accurately synchronized clock times during execution. Time zero is defined as the wall time when the clock synchronization was performed. This capability has proven to be especially helpful in accurately measuring latencies (i.e., the receive time minus the send time) in real-time execution modes.

Miscellaneous Services

The HSC provides a number of miscellaneous services that allow applications to determine their node identifier and the total number of nodes executing in the application. Node identifiers range from zero to the number of nodes minus one. Several other services are provided to fine-tune the size of shared memory segments and how they are partitioned into mailboxes for supporting asynchronous message delivery.

Global Synchronization and Reductions

The simplest operations provided by the HSC are the basic synchronization services. Two flavors of these services are supported: (1) barrier synchronization provides blocking synchronization between all processes, and (2) fuzzy barrier synchronization allows applications to enter a fuzzy barrier without blocking. When all processes have entered the fuzzy barrier, the exit criterion is satisfied. Fuzzy barriers are extremely useful for applications indicating a readiness to synchronize but would like to continue processing while waiting for the rest of the nodes to synchronize. It is an essential service used by advanced parallel discrete-event simulation engines that feature optimistic time management.

Global Reductions

Global reductions provide a second category of synchronized operations. These operations allow applications to synchronously obtain the sum, maximum, minimum, etc. of integer and double precision values passed as arguments by each node. A set of commonly used operations for integers and doubles are directly provided. Special minimum and maximum global reduction services are provided for complex simulation time representations. A more generic global reduction capability is also provided that allows applications to define their own abstract reduction operations on larger classes or structures of data. This allows multiple reductions to be combined very efficiently within a single operation. Advanced parallel discrete-event simulation engines use global reduction services on classes of data to gather processing statistics at Global Virtual Time (GVT) boundaries.

Synchronized Data Distribution Mechanisms

The communications library provides five synchronized services that are commonly used to distribute data between nodes performing matrix and vector operations. These services are: (1) the scatter operation allows a node to partition and distribute a block of data to receiving nodes, (2) the gather operation is the inverse of scatter and allows a node to collect partitioned data from the other nodes, (3) the form-vector operation concatenates buffers from each node to construct a replicated vector of elements provided by each node on all of the nodes, (4) the form matrix operation allows each node to provide a vector of data that is used to construct a replicated two-dimensional matrix (i.e., array of vectors) on all of the nodes, and (5) the synchronous broadcast service broadcasts a buffer from one node to all of the other nodes. These synchronized data distribution services require the global participation of all nodes in the application. Each of these services support large-scale vector/matrix operations commonly performed by HPC applications but are not typically used by advanced parallel discrete-event simulation engines.

Asynchronous Message Passing

Asynchronous message passing is supported with up to 500 user-defined queued message types. Three types of services are implemented: (1) unicast that sends a message from one node to another node, (2) destination-based multicast that sends a message to multiple nodes specified by a bitfield where each bit represents a node in the application, and (3) broadcast that sends a message to all other nodes. A special interface is optionally provided that allows applications to allocate their own memory for received messages. This optimization is critical for reducing overheads and maximizing performance in parallel discrete-event simulations. The WarpIV Kernel uses object factories throughout its internals, and especially for supporting communications, to reuse memory whenever possible. If no application-defined memory management routine is provided, the HSC allocates memory for received messages from the system heap, which can be costly in terms of overhead, especially for very short messages.

Coordinated Message Passing

Coordinated message passing allows applications to exchange multiple messages between nodes in a synchronized manner. Each node may have an arbitrary set of messages that must be sent to other arbitrary nodes. After each node finishes sending all of its coordinated messages, each node then begins reading incoming messages that were sent by other nodes. A NULL message pointer is returned to each node when it can be guaranteed that there are no more coordinated messages to be received by the node. All nodes must participate in this synchronized operation. As with asynchronous message passing services, applications can provide their own optimized memory management for handling received messages.

Object Request Broker (ORB) services

An Object Request Broker capability is provided that allows applications to automatically invoke methods on remote server objects residing on other nodes. These object-oriented interfaces support up to twenty strongly type-checked user-defined arguments, along with variable-length data that may be appended to the message. The ORB services are layered on top of the asynchronous message passing services. They take advantage of the highly optimized object factory memory management services, which have minimal overhead in handling received messages. The ORB services provide a much more user-friendly interface for asynchronous or mixed synchronous/asynchronous applications to communicate. The ORB services are heavily utilized by the Embedded Grid Computing mechanisms and the real-time HPC-RTI provided by the WarpIV Kernel. They were designed to support high-speed object-oriented computational services on HPC machines with extremely low overheads.

Recent Performance Benchmarks

Recent raw-communication benchmarks were performed on a single 10-year-old 12-core machine using only shared memory (see Figure 26) and then on 21 of the same machines connected by a 10-year-old Cisco switch (see Figure 27). Much better performance is expected on newer and more modern multicore computers and high-speed switches.

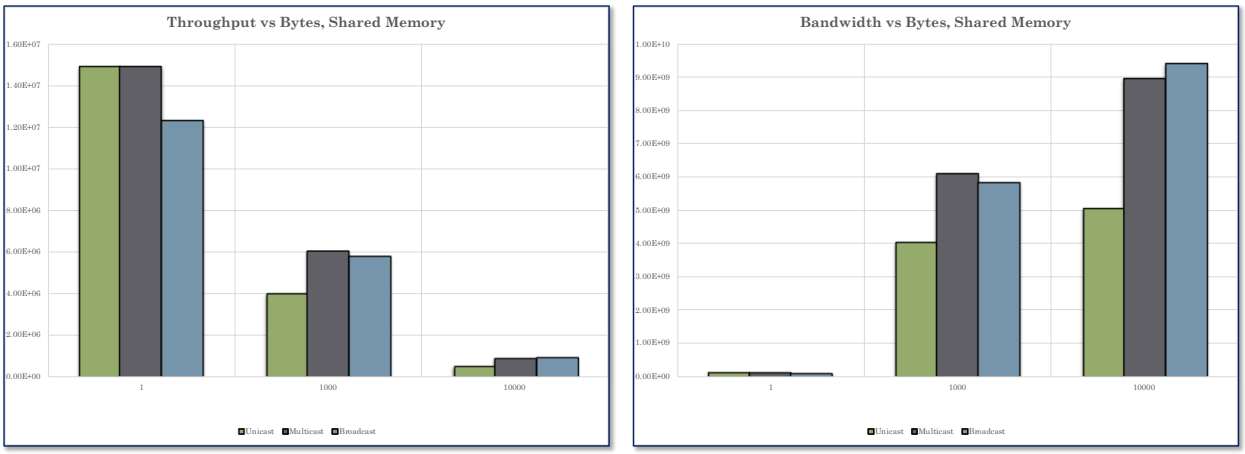


Figure 26: HSC performance on a single 10-year-old 12-core machine measuring throughput (number of messages delivered per second) and bandwidth (number of bytes delivered per second) using only shared memory as a function of message size.

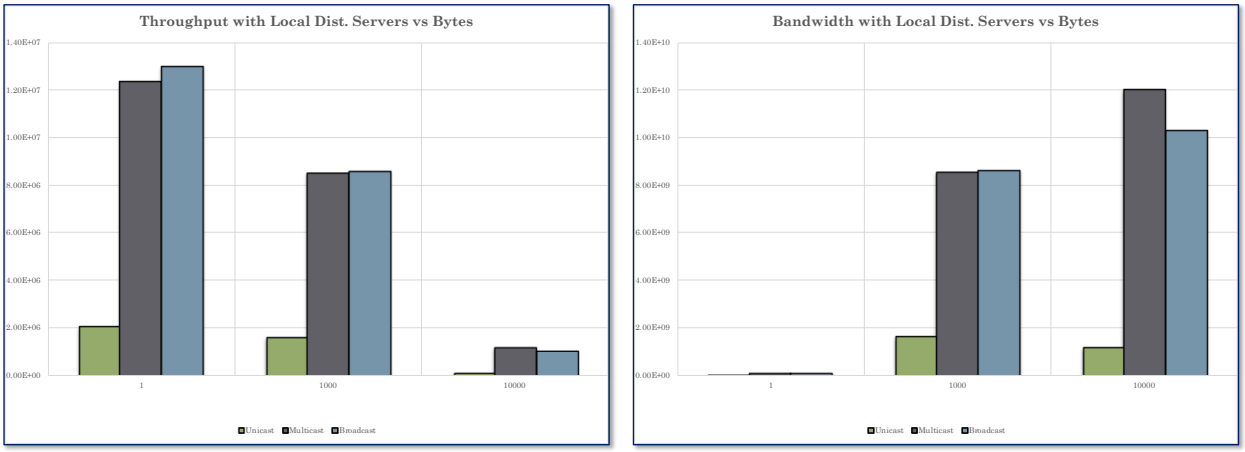


Figure 27: HSC performance on 21 machines, each with 12 cores, measuring throughput (number of messages reliably delivered per second) and bandwidth (number of bytes reliably delivered per second) as a function of message size. Note that message throughput was measured to be even higher (23M messages per second) when benchmarking with the HPC-RTI configured for extremely high stress loads.

Appendix B: Data Logging

The WarpIV Kernel provides a comprehensive bit-compressed data logging service that supports more than 100 commonly used data types and resolutions. Data is logged in time-tagged records that are time-sorted and merged into a single file during execution. Multiple data values can be logged in each record. Unstructured data is logged by name, type, and value. Names are dynamically discovered, bit coded, and stored in symbol tables along with their associated data type to efficiently allow for long variable names. Data is compressed using a variety of techniques. The combined strategy results in extremely high levels of data compression for log files. Supported data types are summarized in Table 1.

Table 1: Data types supported by the WarpIV bit-compressed data logger.

Type	Description	Resolutions
Integer	Uses a Huffman coding scheme to code the number of bits required to store values. Favors small values over large values. For example, the integer 0 takes only two bits of storage rather than 32 bits.	32-bit 16-bit 8-bit
Floating Point	Resizes the number of bits in IEEE standard double-precision values that are used for the mantissa and exponent to compress values. A selectable bit size is also supported for double-precision values given minimum and maximum bounds.	64-bit 36-bit 32-bit 28-bit 24-bit 20-bit 16-bit Selectable
Ratio	Values between 0 and 1 represented by various bit resolutions.	32-bit 24-bit 16-bit 8-bit
Angle	Values between 0 and π or values between 0 and 2π represented by various bit resolutions.	32-bit 24-bit 16-bit 8-bit
Logical	Single bit used to represent true (1) or false (0)	1-bit
String	General strings are replaced by dynamic bit codes. Special support is provided for output messages from print statements containing fixed string characters and integer or floating-point numbers that might vary in each output. Integers are replaced in the string with %d and floating-point numbers are replaced with %f, which is bit coded and reused whenever the message is logged. Then, compressed versions of the integer and/or floating-numbers are packed following the bit-coded string. The data logger provides a stream operator to automate creating logged output messages.	N/A
Time	Time is represented in various integer units (years, weeks, days, hours, minutes, seconds, milliseconds, microseconds, and nanoseconds). Resolutions are selectable.	Selectable
Unit Vector	Unit vectors are stored as two ratios and three sign bits.	32-bit 24-bit 16-bit 8-bit
Vector	Vectors are stored as a unit vector and a magnitude with matching floating-point resolution.	64-bit 36-bit 32-bit 28-bit 24-bit 20-bit 16-bit

Buffer	Buffers are an arbitrary array of bytes. Three algorithms are provided to compress buffers: (1) repeated bytes containing all zeros or ones, (2) repeated bits with 0s or 1s, and (3) optimal 4-bit Huffman coding. Each of these algorithms have optimal properties for different types of buffers (large, small, with lots of zeros or ones, ASCII characters, etc.)	N/A
Continuous Array	Continuous integer or double-precision arrays use a variety of prediction algorithms (native, difference, linear, quadratic, and cubic) to store differences between each value in the array and predictions of the value based on previous values in the array. Double-precision values are converted to integers based on a specified bit resolution and the min and max values stored in the array. Fewer bits are required in representing each data value in the array when the difference between predictions and the actual value is small. For example, if the prediction were perfect, only 2 bits would be required.	Selectable
Text	Text messages are supported by Huffman bit-coding key words and then using Huffman coding for ASCII characters in words that are not known.	N/A
Simulation Time	Simulation time is coded at full resolution. However successive times logged in the same record use differences to compress sizes. So, multiple events with the same time, but slightly different tie-breaking values that are logged in the same record significantly compress.	N/A
Wall Time	Wall times are logged at 36-bit floating point resolutions	36-bit
Named Collection	Supports multiple values that are always logged together as a unit (i.e., structured data). The data logger learns the names of the variables that are logged in a named collection the first time a collection is logged. So, only the actual data values are logged. Prediction and bit-compression techniques are provided to significantly compress values for all data types. Named collections are the primary mechanism for logging analysis data.	Selectable

The following code segment is taken from the composable data logging model for sensors in the Missile Defense Engagement Model (MDEM). The bit-compressed data logging steps for structured data are (1) create a named collection record, (2) log multiple values in the record, and (3) log the named collection record. Note that bit-compressed data logging allows applications to specify resolutions and as an option, prediction algorithm that are used to optimize data compression. Prediction algorithms in the new bit-compressed data logger use previously logged data points to predict values. The difference between predicted and actual values are then logged. If the prediction is accurate, the number of bits required to log values can be significantly reduced (i.e., a 64-bit double can be reduced to just a few bits when the prediction is perfect, or nearly perfect). Continuous variables, such as TrueRange and TrueRangeRate, that are logged work extremely well using these prediction algorithms.

```
////////////////////////////////////////////////////////////////////////////
// Bit compressed data logging with named collections
////////////////////////////////////////////////////////////////////////////

  WpNamedCollectionRecord *record;

  record = APP_DATA_LOGGER.CreateNamedCollectionRecord(
    fo->GetObjType(),
    fo->GetFoId(),
    detection.TargetFoId
  );

  if (detection.SensorType == WP_RADAR_SENSOR_DETECTION_TYPE) {

    record->SetFloat20("TrueRange", detection.TrueRange, WP_BIT_COMP_CUBIC);
    record->SetFloat20("TrueRangeRate", detection.TrueRangeRate, WP_BIT_COMP_CUBIC);

    record->SetFloat20("MeasuredRange", detection.MeasuredRange, WP_BIT_COMP_CUBIC);

    record->SetFloat20("MeasuredRangeRate",
                       detection.MeasuredRangeRate,
                       WP_BIT_COMP_CUBIC);
```

```
   delta = detection.MeasuredRange - detection.TrueRange;
   record->SetFloat20("DeltaRange", delta, WP_BIT_COMP_CUBIC);
   record->SetFloat20("SigmaRange", detection.MeasuredRangeSigma, WP_BIT_COMP_CUBIC);

   record->SetFloat20("DeltaRange/SigmaRange",
                      delta / detection.MeasuredRangeSigma,
                      WP_BIT_COMP_CUBIC);

   delta = detection.MeasuredRangeRate - detection.TrueRangeRate;
   record->SetFloat20("DeltaRangeRate", delta, MDEM_LOG_PREDICTION_TYPE);

   record->SetFloat20("SigmaRangeRate",
                      detection.MeasuredRangeRateSigma,
                      WP_BIT_COMP_CUBIC);

   record->SetFloat20("DeltaRangeRate/SigmaRangeRate",
                      delta / detection.MeasuredRangeRateSigma,
                      WP_BIT_COMP_CUBIC);
}

record->SetInt("NumCorrelatedTargets", detection.NumCorrelatedTargets, WP_BIT_COMP_LOG_DIFF);
record->SetInt("TargetId", detection.TrueTargetId, WP_BIT_COMP_LOG_DIFF);
record->SetInt("SensorId", detection.TrueSensorId, WP_BIT_COMP_LOG_DIFF);

c = DotProduct(detection.TrueDirection, detection.MeasuredDirection);
a = acos(c);

record->SetFloat20("DeltaAngle", WP_RADIANS_TO_DEGREES(a), WP_BIT_COMP_CUBIC);

record->SetFloat20("DeltaAngle/SigmaAngle",
                   a / detection.MeasuredDirectionSigma,
                   WP_BIT_COMP_CUBIC);

APP_DATA_LOGGER.LogNamedCollectionRecord(record);
```

Appendix C: Federating Legacy Simulations

Overview of the High-Level Architecture (HLA)

The High-Level Architecture (HLA) is the de facto standard[59] adopted by the Department of Defense and the international community for integrating multiple simulations (called federates) into an enterprised federation. Simulations integrate with an HLA Run Time Infrastructure (RTI) that follows the HLA standard interface specification. The RTI provides the following basic services for each simulation acting as a federate in an HLA federation.

- Create a federation execution
- Join the federation
- Specify whether the federate operates in real time or logical time[60]
- Publish and subscribe to object and interaction classes[61]
- Process events in real or logical time[62]
- Resign from the federation
- Destroy the federation execution

High-Performance Computing Run Time Infrastructure (HPC-RTI)

The WarpIV Kernel provides the open-source implementation of the High-Performance Computing Run Time Infrastructure (HPC-RTI)[63] [64] [65] [66] [67] [68] that supports interoperability with legacy federates based on the original 1.3 HLA interface specification, which is still used by many M&S programs today (see Figure 28). It is the only RTI in existence that provides both (1) scalable performance[69] (23,000,000 reliably delivered messages per second in a federation with 231 federates operating on twenty-one 12-core machines), and (2) guaranteed repeatable logical-time execution of federations operating on networked multicore machines, HPC clusters, cloud computing architectures, and shared-memory-based supercomputers.

---

[59] HLA was adopted as the primary interoperability standard by the Department of Defense in 1996 and has evolved over the years.

[60] Logical time modes include discrete-event and time-stepped mechanisms for advancing in time.

[61] Objects provide proxies of models in a federate to other (subscribing) federates. Objects contain attributes that persist over time representing the essential state variables of the model (e.g., motion, orientation, cross-section, emitting frequencies, temperature, health, etc.). Interactions are different from objects as they represent events (e.g., detonation events in combat simulations, communication packets sent through tactical data links, detections and tracks, etc.) that are scheduled in time. Interactions do not persist, but rather are one-shot events that trigger a function or method by the subscriber when they are received.

[62] While federates process events, they might update attributes of published objects and/or send interactions. A request is made to the RTI to advance in time. The federate may receive object (discover, remove, reflect attributes) and interaction messages while waiting for the time request to be granted. In this manner, the RTI keeps all of the federates in sync while they process events and exchange messages in time.

[63] Steinman Jeff, et. al., 1999. “Design of the HPC RTI for the High-Level Architecture.” In proceedings of the *1999 Fall Simulation Interoperability Workshop*. Paper 99F-SIW-071.

[64] Steinman Jeff, et. al., 1999. “The SPEEDES-Based Run-Time Infrastructure for the High-level Architecture on High-Performance Computers.” In proceedings of the *High-Performance Computing 1999 Conference, Grand Challenges in Computer Simulation*. Pages 255-266.

[65] Blank Gary, Steinman Jeffrey, Shupe Scott, Wallace Jeff, 2000. “Design and Implementation of the HPC-RTI for the High-Level Architecture in SPEEDES 0.81.” In the proceedings of the *2000 Spring Simulation Interoperability Workshop*. Paper 00S-SIW-153.

[66] Clark Joe, Capella Sebastian, Bailey Chris, Steinman Jeff and Peterson Larry, 2002. "The Development of an HLA Compliant High Performance Computing Run-time Infrastructure" In proceedings of the *2002 Spring Simulation Interoperability Workshop, Paper 02S-SIW-016.*

[67] Steinman Jeff, 2005. “WarpIV Kernel: Real Time HPC-RTI Prototype.” In proceedings of the *2005 Spring Simulation Interoperability Workshop.* Paper 05S-SIW-071.

[68] Steinman Jeff, et. al., 2021. “The Open Unified Technical Framework High Performance Computing Run Time Infrastructure for the High-Level Architecture. To be published at a future date (paper can be made available by contacting WarpIV Technologies, Inc.).

[69] The HPC-RTI was able to reliably deliver more than 23,000,000 messages per second in a real-time federation with 231 federates operating on twenty-one 12-core machines. The HPC-RTI was able to deliver approximately 1/3 the real-time message throughput when executing in logical time.

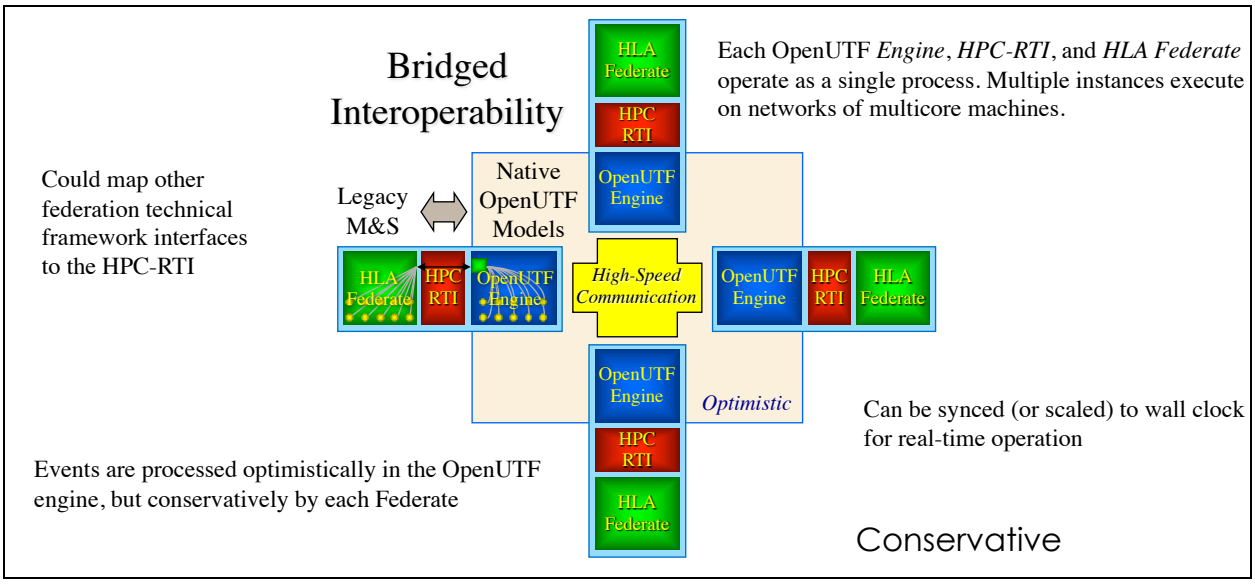


Figure 28: The WarpIV HPC-RTI operates federates as a computational node within a parallel and distributed execution. When running in logical time, all federation events are implemented by the HPC-RTI using the internal publish and subscribe data distribution and event distribution mechanisms with optimistic time management. Special techniques are applied to ensure that only committed events are released to federates, while maintaining scalable performance and guaranteed repeatability. When running in real-time, all HLA messages bypass the optimistic time management mechanisms inside the WarpIV simulation engine and are directly delivered to subscribing federates by the High-Speed Communications.

The HPC-RTI provides a bridge for next-generation scalable, composable, and intelligent modeling technology while continuing to support integration with legacy HLA federates. Models running natively within the OpenUTF can fully interoperate with HPC-RTI federates. The HPC-RTI could be easily extended to integrate with other federation-based Live, Virtual, and Constructive technical frameworks such as OSF, AMIE, DSS, TENA, and DIS.

Local RTI Component Proxy (LrcProxy) Framework

The LrcProxy Framework allows a large federation to execute on a single multicore computer. It is designed to expand to support multiple machines operating in a cloud or in an HPC cluster environment using the HPC-RTI as a gateway to interconnect multiple LrcProxy Framework federations into a larger federation (see Figure 29). A third RTI (still in development) based on the External Modeling Framework (EMF) provides HLA connections to the external world, which could include remote federates not operating locally on the machines executing the core federation. The EMF-RTI is designed to support dynamic joins and resigns, remote displays with two-way human interaction, external hardware or software systems, remote federates that for whatever reason cannot execute locally, etc.

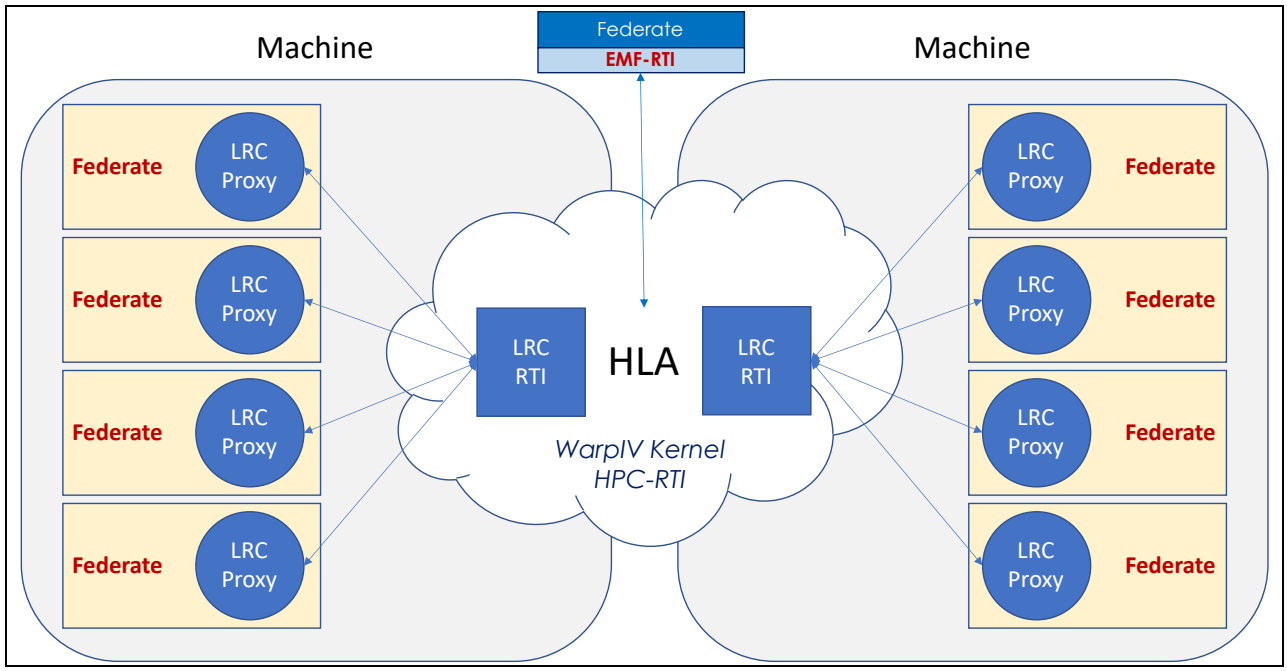


Figure 29: Conceptual integration of the LrcProxy Framework with the HPC-RTI and the EMF-RTI operating on several multicore computers in a cloud environment. Each computer has one LRC-RTI process that manages many more local federates than the number of cores on its machine. Each LRC -RTI coordinates (a) message delivery between the federates, and (b) which federates are granted time, which wakes them up to do their processing. federates always sleep in their HLA call to tick after they request time until they are granted time. Each LRC-RTI also acts as a gateway federate within the HPC-RTI that is distributed across multiple machines. In this manner, the LRC RTI on each machine coordinates time and messages between its local federates and automatically through the HPC-RTI with other federates that are managed by their LRC-RTI processes. The EMF-RTI is unique in that it allows remote federates to dynamically join and resign with the rest of the federation. It uses a different mechanism with lower overheads for higher latency network connections to synchronize their federate with the core high-speed federation operating in the cloud.

The processing loop within each federate as it coordinates with the LRC-RTI is shown in Figure 30. A federate requests time and then calls tick where it goes to sleep. The time request message is sent to the LRC-RTI that is running on the same machine, where time requests and grants are managed in a Time Request Tree and Time Barrier Tree. Both of these trees are sorted by time. The difference between these two trees is that time barriers for each federate are set to their granted times plus their lookahead value. Note that granted time plus lookahead specifies the earliest time tag for a message that might be produced by the federate. By definition, any time request for a time that is less than the earliest time in the barrier tree can be granted. The LRC-RTI only grants time to federates when one or more cores are available. In this way, there are never more activated federates doing their computations than cores.

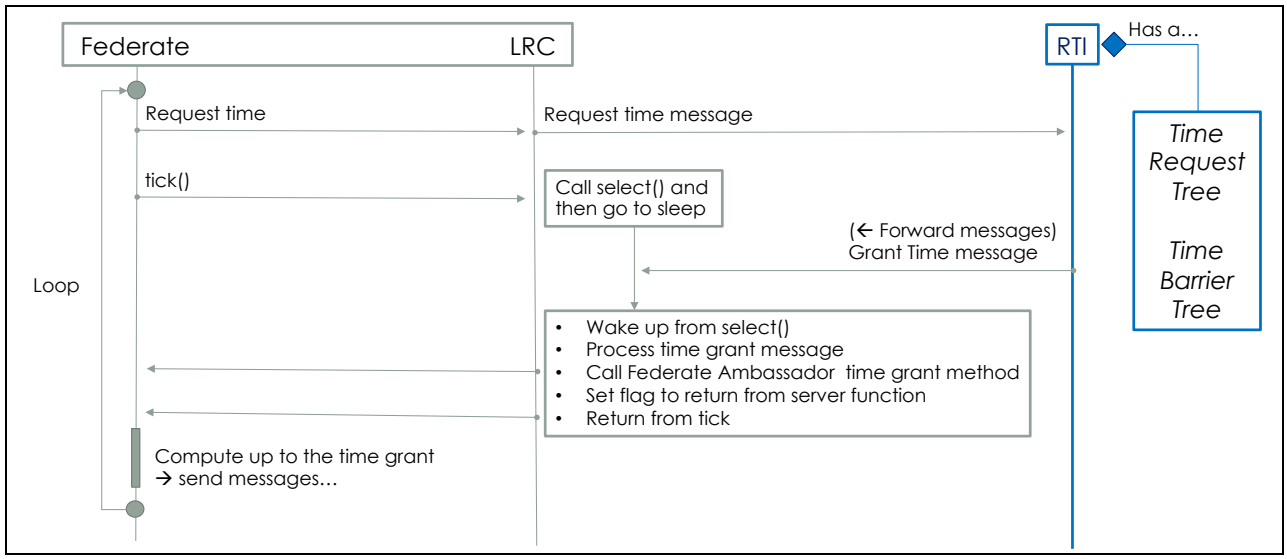


Figure 30: Processing loop within each federate as it coordinates with the LRC-RTI.

The key to putting federates to sleep is the *select* system call that is traditionally used to support client/server applications. Servers typically go to sleep on a select system call as they wait for messages to be received and processed. This mechanism keeps servers from wasting CPU cycles when there is no work to be performed. Servers wake up from select whenever a message arrives, and then after the message is serviced, the server calls select again to go back to sleep. The LRC embedded within each federate actually wakes up from select when HLA messages involving objects or interactions are received, but they are handled as callbacks within the federate ambassador. The call made by the federate to tick only returns when a time grant message, sent by the LRC-RTI, is received.

Appendix D: Preliminary Performance Results on 64-Core AMD Ryzen Threadripper

WarpIV purchased a new 64-core AMD Ryzen Threadripper machine in February 2021. Preliminary results for the four simulation applications developed in Phase I are shown below. As a note, the new 64-core machine ran faster than the older 32-core machine on a single core. It also scaled better up to 32 nodes before exhibiting similar cache/RAM bottlenecks that were seen on the 32-core machine. This was expected since both machines are Non-Uniform Memory Access (NUMA) with half of the cores requiring two hops to access memory in RAM. Effects of the NUMA architecture became more evident once going beyond 32 cores.

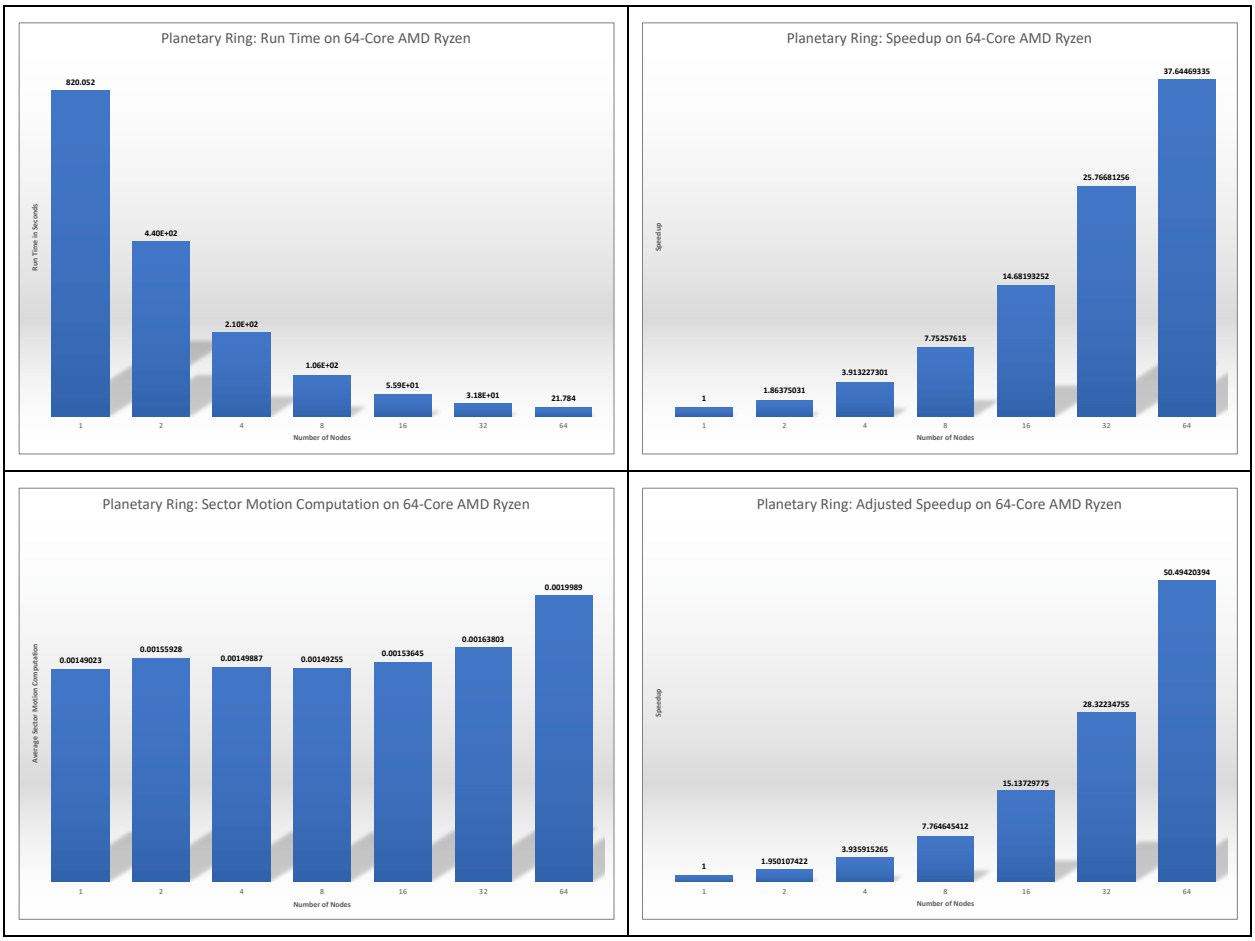


*Figure 31: Benchmarks for the Planetary Ring simulation on the 64-core AMD Ryzen Threadripper with **data logging turned off**. Upper left shows the execution time, Upper Right shows speedup. Lower left shows how computational performance of the most expensive event increased due to cache/RAM contention issues with the CPU and motherboard. Lower right shows adjusted speedup where the cache/RAM contention affects are factored out. The adjusted speedup indicates the speedup that would be achieved if there were no computational slowdown effects due to cache/RAM issues.*

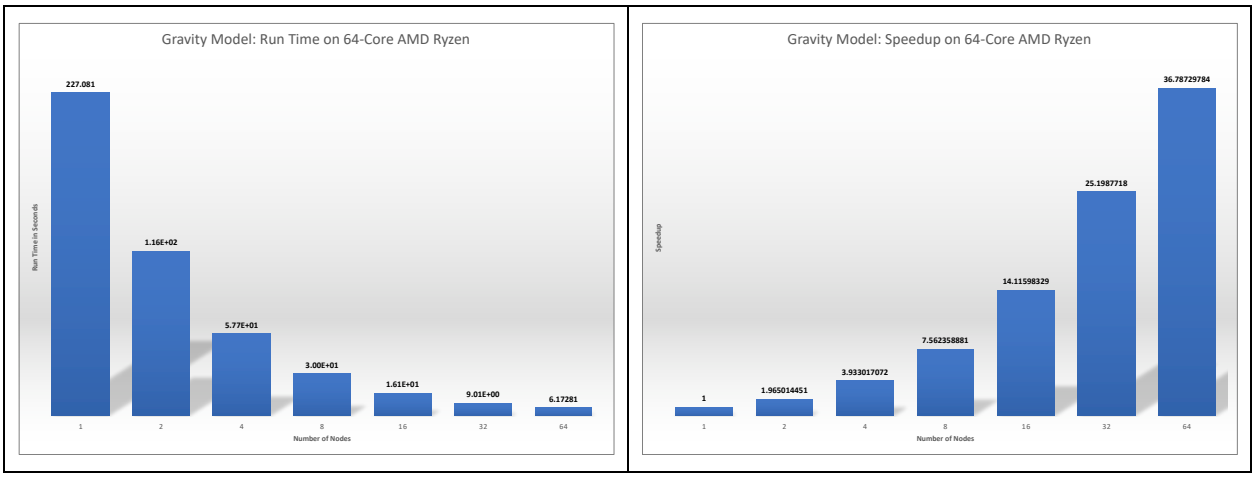


Figure 32: Run time and speedup of the Saturn gravity model using a 350-kilometer mass resolution executing on a 64-core AMD Ryzen Threadripper Linux machine.

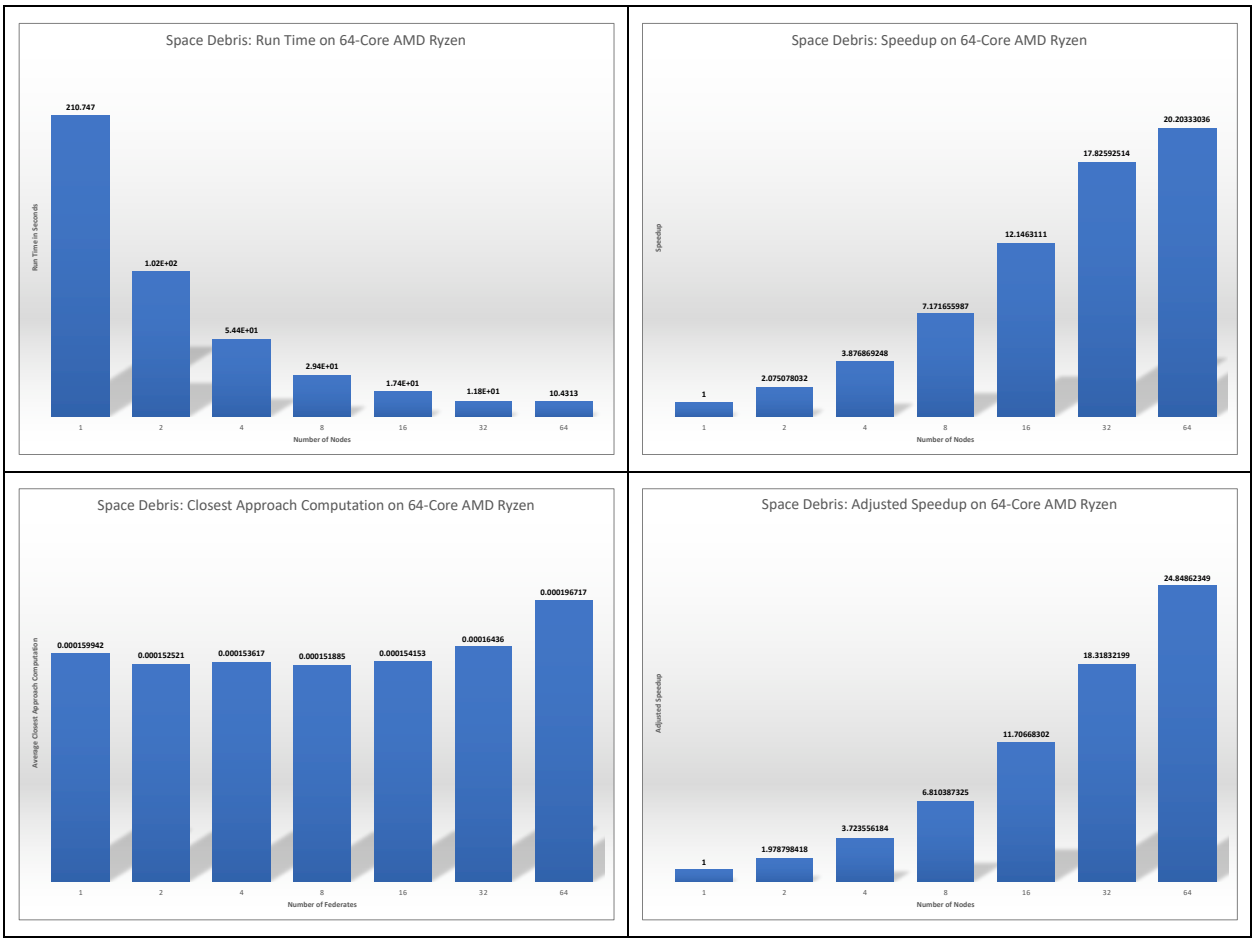


*Figure 33: Benchmarks for the space debris simulation on the 32-core AMD Ryzen Threadripper with **data logging turned off**. Upper left shows the execution time, Upper Right shows speedup. Lower left shows how computational performance of the most expensive event increased due to cache/RAM contention issues with the CPU and motherboard. Lower right shows adjusted speedup where the cache/RAM contention affects are factored out. The adjusted speedup indicates the speedup that would be achieved if there were no computational slowdown effects due to cache/RAM issues. Note that speedup was limited due to critical path dependencies causing rollbacks. The number of rollbacks could be significantly reduced in the future by using the HiGrid filtering techniques described in this report to minimize rollback-causing messages as interceptors change their motion due to track updates for their targeted warheads.*

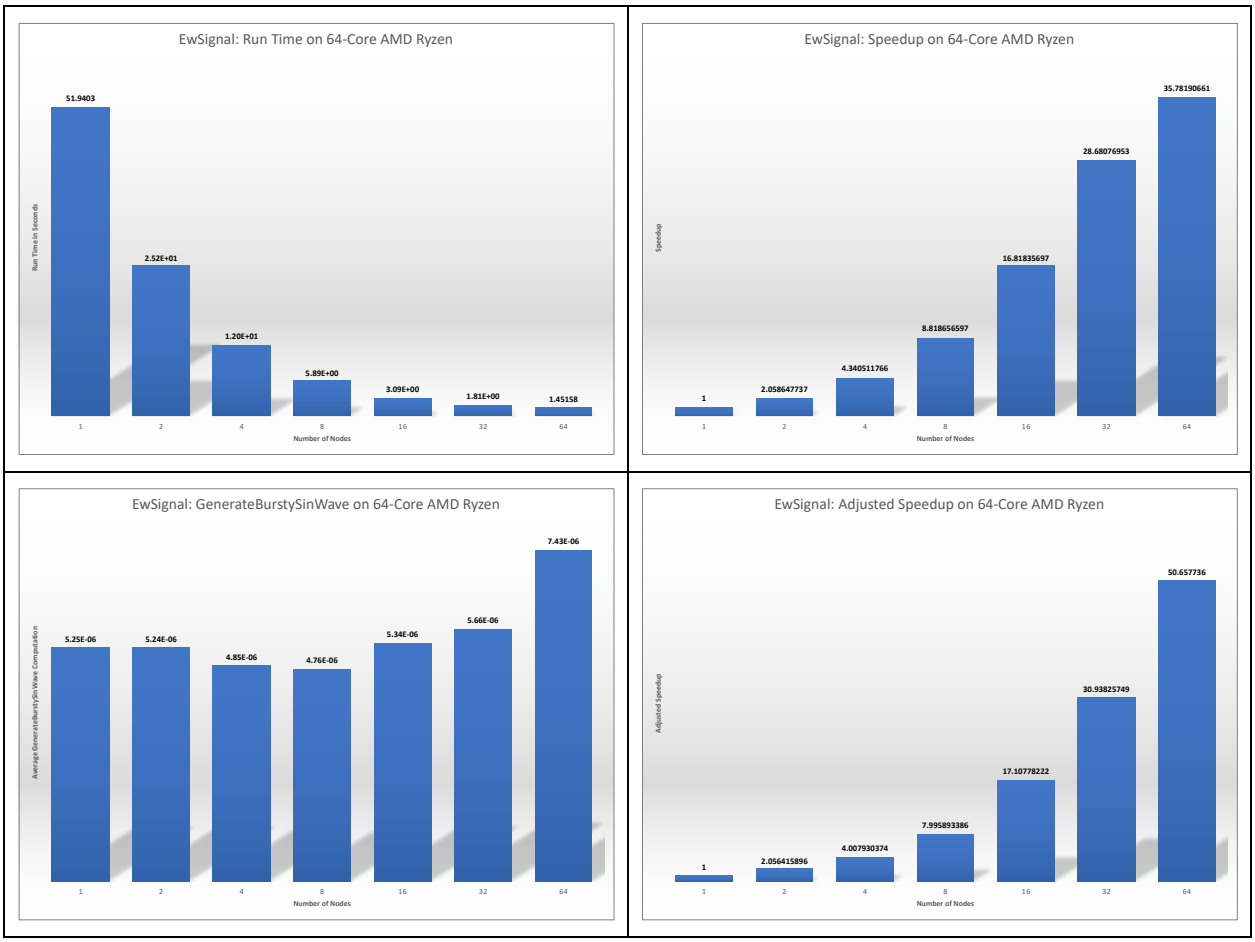


Figure 34: Benchmarks for the RF propagation simulation modeling electronic warfare on the 64-core AMD Ryzen Threadripper (note, there was no data logging in this simulation). Upper left shows the execution time, Upper Right shows speedup. Lower left shows how computational performance of the most expensive event increased due to cache/RAM contention issues with the CPU and motherboard. Lower right shows adjusted speedup where the cache/RAM contention affects are factored out. The adjusted speedup indicates the speedup that would be achieved if there were no computational slowdown effects due to cache/RAM issues.

REPORT DOCUMENTATION PAGE	Form Approved 0MB No. 0704-0188

The public reporting burden for this collect,on of information is estimated to average 1 hour per response , including the time for reviewing instructions, searching existing data sources, gathenng and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to the Department of Defense, Executive Service Directorate (0704-0188). Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid 0MB control number.

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ORGANIZATION .

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
21-0 2-2021	Final Sc ie ntific and Technical Repo rts	August 31, 2020 to March I , 2021

4. TITLE AND SUBTITLE	Sa. CONTRACT NUMBER
Phase I Fi nal Report SBI R Topic N um ber: S5.01 -4746 Project Title: Techn ologies for Larg e-Scale Nume rical Si mu lation	NSSC20C057080
	Sb. GRANT NUMBER
	Sc. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Jeffrey Stei nm an	**5d. PROJECT NUMBER** 4200738504
	Se. TASK NUMBER
	Sf. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Warp!V Technologies, Inc. 5230 Carroll Canyon Road Suite 306 San Diego, CA 92121	IT EM 0001
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) NASA Shared Serv ices Center (NSSC) Building 1111 , Jerry Hlass Road Sten ni s Space Center MS 39529-000 I	**10. SPONSOR/MONITOR'S ACRONYM(S)** NASA
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This project demonstrates the use of the WarplV Kernel Parallel Discrete-Event Simulation (PDES) Engine to support a variety of H igh-P erformance Comput in g (HPC) simulation appli cations for NASA. To demonstrate the general capabi li ties of the Warp IV Kernel for add ressin g NASA HPC simu lation problems, four distinct scenarios are explored in Phase I: (I) Simulation of planetary rings (Saturn) as an n-body grav itationa l problem, (2) large-scale space debris with collision prediction for satellit es and launched rockets, (3) computation of gravitational fields for elli pso idal planets, and (4) Radio Frequency (RF) propagation from multiple moving transmitting sources and receivers (e.g., modeling spacecraft signal reception by the Deep Space Network in noisy RF environments). A major goal of this effort is to provide a framework that allow s non-HP C software engineers to effic ie ntly develop and test their simulations on multi-core laptops or desktops and then move to I-IPC machines (with no changes to code) as their problem size scales up. In this way, the WarplV Kernel provides the path for many more simu lation users to migrate to HP C platforms usin g the world's most advanced si mu lation technology.

15 . SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE	SAR		Jeffrey Steinman
					19b. TELEPHONE NUMBER *(Include area code)* (858) 531-0643