Joint Experimentation on Scalable Parallel Processors

Robert F. Lucas, Dan M. Davis
Information Sciences Institute, University of Southern California
Marina del Rey, California
rflucas@isi.edu, ddavis@isi.edu

ABSTRACT

The JESPP project exemplifies the ready utility of High Performance computing for large-scale simulations. J9, the Joint Experimentation Program at the US Joint Forces Command, is tasked with ensuring that the United States' armed forces benefit from improvements in doctrine, interoperability, and integration. In order to simulate the future battlespace, J9 must expand the capabilities of its JSAF code along several critical axes: continuous experimentation, number of entities, behaviors complexity, terrain databases, dynamic infrastructure representations, environmental models, and analytical capabilities. Increasing the size and complexity of JSAF exercises in turn requires increasing the computing resources available to JFCOM. Our strategy exploits the scalable parallel processors (SPPs) deployed by DoD's High Performance Computing Modernization Program (HPCMP). Synthetic forces have long run in parallel on inter-networked computers. SPPs are a natural extension of this, providing a large number of processors, inter-connected with a high performance switch, and a collective job management framework. To effectively use an SPP, we developed software routers that replace multicast messaging with point-to-point transmission of interest-managed packets. This in turn required development of a new simulation preparation utility to define the communication topology and initialize the exercise. We also developed tools to monitor processor and network loading and loggers capable of absorbing all of the exercise data. We will report on the results of J9's December 2002 Prototype Event which simulated more than one million clutter entities along with a few thousand operational entities using 50,000 interest states on a terrain database encompassing the entire Pacific Rim. The exercise was controlled and "fought" from a J9 test bay in Suffolk, VA and the clutter entities were executed on a remote SPP in Los Angeles, CA. We will also present results from the Prototype Event in March 2003, as well as our long-term plans.

ABOUT THE AUTHORS

Robert F. Lucas is the Director of the Computational Sciences Division of the University of Southern California's Information Sciences Institute (ISI). There he manages research in computer architecture, VLSI, compilers and other software tools. His Division includes a group studying Large-scale battlefield simulations. He has been the principal investigator on the JESPP project since its inception in the Spring of 2002. Prior to joining ISI, he was the Head of the High Performance Computing Research Department for the National Energy Research Scientific Computing Center (NERSC) at Lawrence Berkeley National Laboratory. He has also served as the Deputy Director of DARPA's Information Technology Office, a member of the research staff of the Institute for Defense Analysis's Center for Computing Sciences and a member of the Technical Staff of the Hughes Aircraft Company. Dr. Lucas received his BS, MS, and PhD degrees in Electrical Engineering from Stanford University in 1980, 1983, and 1988 respectively.

Dan M. Davis is the Director, JESPP Project, Information Sciences Institute (ISI), University of Southern California, and has been active in large-scale distributed simulations for the DoD. While he was the Assistant Director of the Center for Advanced Computing Research at the Caltech, he managed Synthetic Forces Express, a multi-year simulation project. Prior to that, he was a Software Engineer on the All Source Analysis System project at the Jet Propulsion Laboratory and worked on a classified project at Martin Marietta, Denver. An active duty Marine Cryptologist, he holds a U.S.N.R. commission as a Commander, Cryptologic Specialty. He has served as the Chairman of the Coalition of Academic Supercomputing Centers and the Coalition for Academic Scientific Computation. He was part of the University of Hawai‘i team that was selected to manage the Maui High Performance Computing Center in May of 2001. He received a B.A. and a J.D., both from the University of Colorado in Boulder.

Joint Experimentation on Scalable Parallel Processors

Robert F. Lucas, Dan M. Davis
Information Sciences Institute, University of Southern California
Marina del Rey, California
rflucas@isi.edu, ddavis@isi.edu

Introduction and Background

The United States has a vested interest in being able to simulate more than one million vehicles, all with sophisticated behaviors, operating on a global-scale, variable resolution terrain database. This is driven by the government's needs to accommodate new computer and communications technology (Cebrowski, 1998) and simulate more complex human functions in technically diverse situations (Sanne, 1999). The U.S. Department of Defense (DoD) has begun a series of forces modeling and simulation (FMS) experiments to study the complexities of operations in future battlespaces. In support of their mission, analysts need to conduct interactive experiments with entity-level simulations, using programs such as the Semi-Automated Forces (SAF) family used by the DoD (Ceranowicz, 2002). This needs to be done at a scale and at a level of resolution adequate for modeling the complexities of military operations in urban situations.

Experimenters, using large numbers of Linux PCs distributed across a LAN, found that communications limited the analysts to tens of thousands of vehicles, about two orders of magnitude fewer vehicles than their needs. This paper addresses the benefits of the successful application of computational science and use of parallel computing on SPPs to address this issue. By extension, it illuminates the way for those with similar simulation needs, but faced with similar computational constraints, to make beneficial use of the SPP assets of the High Performance Modernization Program (HPCMP.)

While there are many FMS approaches that are currently in use, simulation and modeling at the entity level (modeling each individual person and vehicle) manifest some very attractive features, both for training and for analysis. Many who would argue that entity level simulations should be employed, maintain that these would generate the most timely, most valid, and most cost-effective analyses. Making these simulations so that the DoD can involve humans, i.e. Human-in-the-Loop (HITL), additionally augments the DoD's ability to assess true impacts on personnel and procedures. (Ben-Ari, 1998) There are several new methods to modeling human behavior (Hill, 2000). While these require significant independent research (vanLent, 1998), they also require significant additional computing power. The PC/LAN environments has not allow the analyst to conduct these experiments at the scale and level of resolution necessary. These constraints have also been commonly found in other varieties of simulation.

In the present experimentation environment, newfound emphases on civilian, "White," and clutter entities have expanded the horizons of entity-count by an order of magnitude. In a typical urban setting, the number of civilian vehicles will easily outnumber t combat vehicles by a factor of ten or, more likely, by a factor of 100. Trying to assess the utility of sensors in discriminating the former from the latter will be ill served by simulations that are limited to a few thousand vehicles total.

In order to make good use of the SPP assets currently available to DoD experimenters, this project applied approaches that others should find easily and reliably implementable on other, similar, efforts. The discussion of the implementation of a more scalable code into the JSAF code base will not only represent a record of where we have been, but show the path for where we may go in the future.

The current work on Joint Experimentation on Scalable Parallel Processor (JESPP) Linux clusters enabled successful simulation of 1,000,000 entities. Software implementations stressing efficient inter-node communications were necessary to achieve the desired scalability. One major advance was the design of two different software routers to efficiently direct information to differing hierarchies of simulation nodes. Both the "Tree" and the "Mesh" routers were implemented and tested. Additionally, implementations of both MPI and Socket-Programmed variants were intended to make the application more universally portable and more organizationally palatable. The development of a visual and digital performance tool to monitor the distributed computing assets was also a goal that has been accomplished, leading to insights gained by using this new tool. The design and selection of competing program initiation tools for so large a simulation platform was problematical and the use of existing tools was considered less than optimal. The analytical process for resolving initiation issues, as well as the design and implementation of the resulting initiation tool developed by the group, is both a demonstrable result and an example of a computation science paradigm for approaching such problems. The design constraints faced are analyzed along with a critical look at the relative success at meeting those constraints.

The requirements of the user are for a truly interactive simulation that is scalable along the dimensions of complexity of entity behavior, quantity of total simulated entities, sophistication of environmental effects, resolution of terrain, and dynamism of features. This is a challenge that the authors assert may only be amenable to meta-computing across widely dispersed and heterogeneous parallel computer assets (Foster, 1997). Just achieving scalability and integration across all of these dimensions would be difficult. Even more so, fielding a stable, dynamically reconfigurable compute platform that may include large parallel computers, Linux clusters, PCs on LANs, legacy simulators, and other heterogeneous configurations produces new obstacles to implementation. Several Computational Science approaches are discussed

The current work is based on the early work headed by Paul Messina at Caltech (Messina, 1997). The Synthetic Forces Express project (SF Express) began in 1996 to explore the utility of Scalable Parallel Processors (SPPs) as a solution to the communications bottlenecks then being experienced by one of the conventional SAFs, ModSAF. The SF Express charter was to demonstrate a scalable communications architecture simulating 50K vehicles on multiple SPPs: an order-of-magnitude increase over the size of an earlier major simulation.

SPPs provided a much better alternative to networked workstations for large-scale ModSAF runs. Most of the processors on an SPP can be devoted to independent executions of SAFSims, the basic ModSAF simulator code. The reliable high-speed communications fabric between processors on an SPP typically gives better performance than standard switching technology among networked workstations. A scalable communications scheme was conceived, designed and implemented in three main steps:

1. Individual data messages were associated with specific interest class indices, and procedures were developed for evaluating the total interest state of an individual simulation processor.

2. Inter-node Communications: Within an individual SPP, certain processors were designated as message routers; the number of processors used as routers could be selected for each run. These processors received and stored interest declarations from the simulator nodes and moved simulation data packets according to the interest declarations.

3. WAN Communications: Additional interest-restricted data exchange procedures were developed to support SF Express execution across multiple SPPs. The primary technical challenge in porting ModSAF to run efficiently on SPPs lay in constructing a suitable network of message-passing router nodes/processors. SF Express used point-to-point SPP MPI communications to replace the UDP socket calls of standard ModSAF. The network of routers managed SPP message traffic, effecting reliable interest-restricted communications among simulator nodes. This strategy allowed considerable freedom in constructing the router node network.

As the simulation problem size increased beyond the capabilities of any single SPP, additional interest-restricted communications procedures were needed to enable Metacomputed ModSAF runs on multiple SPPs. After a number of options were considered, an implementation using dedicated Gateway processors to manage inter-SPP communications was selected.

In March of 1998, the SF Express project performed a simulation run, with more than 100,000 individually simulated vehicles. The runs used several different types of Scalable Parallel Processors (SPPs) at nine separate sites spanning seven time zones. These sites were linked by a variety of wide-area networks. (Brunett, 1997)

This work depended on the existing DIS standard utilized by the SAFs at that time. That standard was replaced by the HLA/RTI standard that was purportedly more scalable, but several years of use has shown the clear limits of this simulation approach. This has not prevented some experimenters from getting very good results while simulating ~ 30,000 entities (Ceranowicz, 2002). These new standards and additional requirements have driven the development of two new router designs, Mesh Routers and Tree Routers.

JSAF

The Joint SemiAutomated Forces (JSAF) is used by the US Joint Forces Command (JFCOM) in its experimentation efforts. It currently is running on a LAN or WAN, using Ethernet. Communication is implemented with High Level Architecture (HLA) and a custom version of Runtime Infrastructure (RTI) software version RTIS. A run is implemented as a federation of simulators or clients. Multiple clients in addition to JSAF are typically included in a simulation.

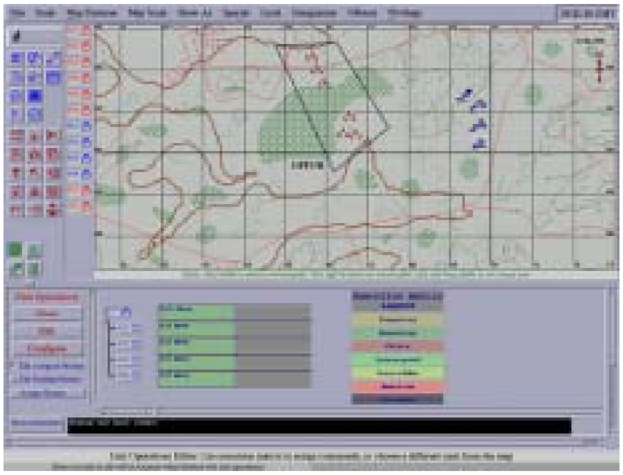

Figure 1
Plan View display from a SAF

HLA and RTI use the publish/subscribe model for communication. Typically, these processors are relatively powerful PCs using the Linux operating system. A data item is associated with an interest set. Each JSAF instance subscribes to ranges of interest. A JSAF may be interested in, for example, a geographic area or a range of radio frequencies. When a data item is published, the RTI must send it to interested clients.

Figure 2
3D Rendered display from a SAF

A typical JSAF run simulates a few thousand entities using a few workstations on a LAN. A simple broadcast of all data to all nodes is sufficient for this size simulation. The RTI on each node discards data that is not of interest to each receiving node. Broadcast is not sufficient when the simulation needs tens of thousands of entities and scores of workstations. UDP multicast was implemented to replace the simple broadcast. Each simulator receives only the data to which it has subscribed, i.e. has a stated interest.

Operational imperatives drive experimental designs that now required further expansions of JSAF capabilities. As noted before, some of the requirements justifying these extensions are the need for:

- More entities
- More complex entities
- Larger geographic area
- Multiple resolution terrain
- More complex environments

The most readily available source of one or more orders of magnitude of increased compute power is the capability presented by Scalable Parallel Processors. In the JESPP project, JSAF was augmented, enabling it to run on multiple Linux clusters, using hundreds of processors on each cluster. Future runs will require thousands of processors on multiple clusters. The primary difficulty in using these resources is the scaling of internode communication.

Another issue is that UDP multicast is limited to approximately three thousand different channels. Based on geography alone, worldwide simulations using JSAF require many more interest states. UDP multicast has been replaced by software routers, which will alleviate that constraint.

Software routers were implemented on individual nodes in a network that included all of the client simulators. Each simulator is connected to only one router. Routers are connected to multiple clients and multiple routers. Each connection is a two-way connection. Two types of information are present in the network. One is data along with interest description. The other is the current interest state of each client. The interest state changes as each node subscribes and unsubscribes to specific interest sets, as is appropriate depending on the simulation progress.

Each router must maintain the interest set of each node to which it is connected, including other routers. A router's interest set is the union of all connected nodes. A router then uses the interest state associated with data it receives to determine how to forward the data. For any topology, communication is minimized in that each client node receives only the data in which it is interested.

The initial router implementation was a tree router. Each router has multiple clients but only one parent. There is one router that is the top of the tree. A second topology has subsequently been implemented. We have referred to it as a mesh router. Instead of a single router at the top of a tree, there is a mesh of routers with all-to-all communication. Each simulator is a client of one of the mesh routers. Like the tree router, the primary task of the mesh router is to maintain the interest state of all clients so as to forward only data that is of interest to each client and router. Further hybrid topologies are possible with little or no code modification, such as a mesh of meshes or a mesh of trees. Conceptually, the mesh should provide better scalability.

Another use of routers is the implementation of gateways providing an interface between different RTI and communication implementations. TCP and UDP are used for communication. Routers can use a different protocol on different connections and perform required data bundling, unbundling, etc. Different RTI implementations, required used by different groups, can communicate via router-based gateways.

The ultimate goal is for the capacity of a simulator network to scale easily up to desired magnitudes, by simply increasing the number of processors. Comprehensive testing and measurement is required to document the performance of various topologies and router implementations. This testing will identify performance bottlenecks and suggest alternative implementations. Multiple simulation scenarios must be tested to construct guidelines for assigning simulators, routers and topologies to multiple SPPs.

Fault tolerance is another area being studied. A JSAF simulator is not affected by the loss of other simulators on the LAN. The use of routers may create a single point whose failure eliminates multiple simulators. The use of dynamic topologies will be studied and implemented to minimize the consequences of single node failures. Several different concepts of providing redundancy or instantaneous recovery are being considered and will be implemented and evaluated.

Tree Routers

The first router implementation is a tree router.

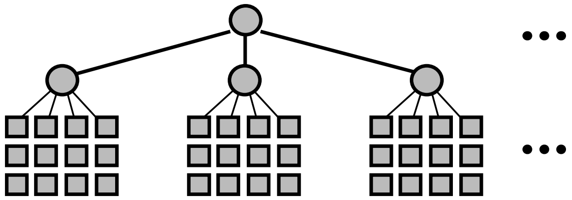

Figure 3
Tree Router Architecture

Each simulator is connected to a router. All communication to and from a simulator goes thru the router. Routers have multiple child clients. All routers, except the single router that is the root of the tree, have one parent router. The root router has no parent. Each simulator has exactly one parent router.

The function of a router is to receive data from clients and parent, and forward (send) the data to any clients or parent that have interest. This requires that simulators and routers communicate interest and data. A simulator or router maintains the interest set of its parent router. A router maintains the interest set of all of its clients. When a simulator changes its subscription, it sends a modified interest set to its router. If this modifies the interest set of the router, the router sends the modification to its other clients, and its parent. Interest modifications propagate across the router network until all nodes possess the interest set of clients and parent.

When a simulator publishes data, the associated interest set is intersected with the interest set of its router. If the intersection is not empty, the published data is sent to the router. When a router receives data from a client, the interest set is intersected with the interest set of the router's other clients. For each other client, if the intersection is not empty, the data is sent to the client. The same is performed for the router's parent. Given the connectivity, or topology, of a tree, this set of operations tends to minimize communication while ensuring that all simulators receive all data of interest to them in a timely manner.

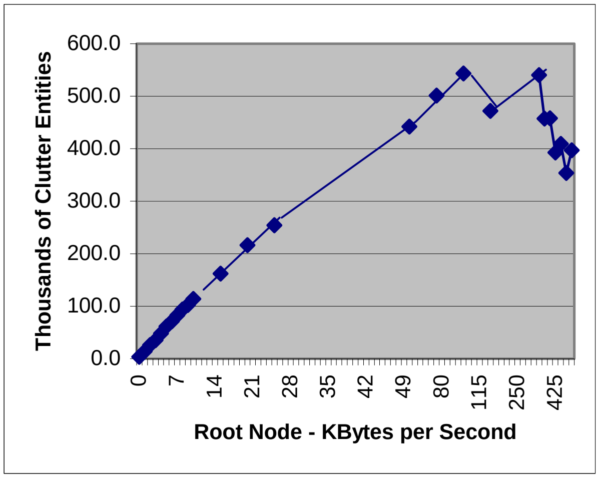

Chart 1
A collection of 32 ClutterSim Nodes, Eight Primary Routers and One Root Node Router

The data in Chart 1 above were collected on the USC IBM Linux Cluster using a test configuration of four ClutterSim Nodes reporting to a single Primary Router and the Eight Primary Routers reporting to a single Root Node Router. While the intent was to demonstrate the scalability characteristics of the Tree Routers, the data suggest that it only demonstrated the capabilities of a free standing ClutterSim which would tend to adjust its reporting rate as it approached the simulator's maximum capacity. The test team is currently developing an improved measurement strategy to better asses Tree Router scalability, which has shown some very definite limits in actual JFCOM experiments run on the MHPCC cluster, Huinalu.

Mesh Routers

Next, we will describe the design of the new mesh router and the impact of that design on scalability.

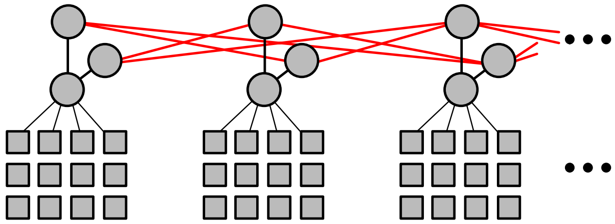

Figure 4
Mesh Router Architecture

The basic communications architecture expands on the original SF Express work, as illustrated in Fig. 2. For purposes of the present discussion, the relevant features of this architecture are as follows:

1. Simulation processors (squares) are grouped with each group associated with a specific router node ("Primary Router").
2. Message flow from a simulator to its Primary Router had three main components:
   a. Interest Subscriptions: Simulators specify data type of (local) interest.
   b. Data messages up: All messages generated within the Simulator are sent up for possible transmission to other simulators.
   c. Data messages down: Messages from elsewhere that match the relevant interest declaration are sent from the router to the simulator.
3. Two additional interconnected layers of router nodes (Pop-Up and Pull-Down) manage all of the interest-screened data communications among the {Primary, Simulators} sets.
4. Strict flow control among the layers prevents communications deadlock, with an additional "token protocol" used to eliminate ineffective data reading attempts.

This architecture was central within the SF Express large-scale simulations, with another class of router-like processors ("Gateways") used to manage interest-screened communications among participating SPPs. The present effort involves a number of significant extensions from the original SF Express code:

1. Interest enumeration and interest state declarations are now done using tools within RTI-s.
2. Interest declarations are now "two way", involving both interest declarations and publications.
3. Limitations on message size have been eliminated, thus supporting occasional very large "environmental" messages within typical JSAF applications.
4. The entire code base has been reformulated in a rather rigorous object-oriented (C++) form.
5. Communications (along any link in the figure) are now cleanly factored into a number of objects and supported by extensions now incorporated into the RTI-s libraries.
6. The system fully supports mixed communications protocols. Some of the links in the Figure might represent MPI communications while others could be TCP.
7. The Gateway models from SF Express have been reformulated (now essentially clients rather than "special" routers). Taken together with item 6, this greatly facilitates linking of "meta-systems" incorporating LANs and SPP assets.

Preliminary Performance data collected on the Mesh Routers was again limited by ClutterSim performance and a new testing régime is now being designed, with results anticipated in the month of September. Scalability tests on the previous implementations are presented below in Chart 2 as indicative of the performance anticipated with the current Mesh Router.

Run Size (Nodes)	**81**	**161**	**238**
Number of Router Triads	3	6	9
Number of SAFSim Nodes	60	120	180
Number – Sim. Vehicles	4,327	8,529	12,915
Primary Busy Fraction	0.188±0.04	0.189±0.02	0.207±0.04
Pop-Up Busy Fraction	0.025±0.02	0.025±0.01	0.027±0.01
Pull-Down Busy Fraction	0.030±0.02	0.026±0.02	0.031±0.02
Primary Receive Time	0.560±0.12	0.537±0.06	0.587±0.09
SAFSim Comms. Fraction	0.023±0.02	0.024±0.01	0.030±0.04
SAFSim Receive Time	1.191±2.20	0.978±0.91	1.526±0.65

Chart 2
Latency Data showing Scaling of Mesh Routers

Performance and resource usage monitoring

Abstraction mechanisms found in many distributed programming systems enhance software reusability and interoperability by hiding the physical location of remote software processes. These abstraction mechanisms, which include HLA's concept of federates (Lightner, 1998) and CORBA's concept of components (Keahey, 1997), greatly reduce the complexity of accessing remote components. But, they come at the cost of reduced visibility, which hinders discovery of faults and impedes understanding of performance characteristics of the distributed system. This section describes a performance and resource usage monitoring tool Monitoring Remote Imaging (MRI) that aids developers in understanding the behavior of HLA simulations by displaying the monitoring data within the context of the execution of the distributed system. Similar specialized tools could easily be envisioned, designed and encoded for other simulations.

In Figure 5 each CPU pie chart depicts the CPU usage breakdown for one compute node:

- Red for user-level CPU usage
- Blue for system-level CPU usage,
- Green for idle.

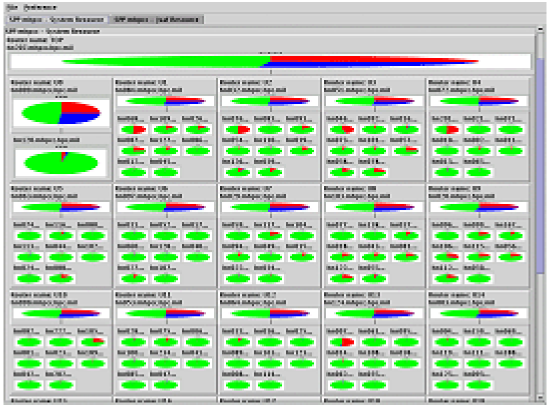

Figure 5
Resource usage data of a JSAF federation displayed within the context of a tree connection topology.

MRI displays monitoring data in the context of the federation connection topology. Figure 5 shows ta captured screen of a MRI client's resource usage gauges displayed in the context of a three-level tree topology. The large oval pie chart at the top represents the root tree router. The set of rectangles underneath the root tree router represents sub-trees or router subgroups. Each subgroup has a tree router (medium-sized pie chart) connected to a set of federates (smaller pie charts). The first subgroup on the left as only one federate, and the other subgroups have eight federates.

Each compute node has two CPUs, but the node is currently only running one process, so typically at most 50% of the CPU is used for non-thread applications like JSAF. At the snapshot when Figure 5 was taken the router nodes within the tree show substantial system-level CPU usage, which indicates the routers are busy accessing kernel-level instructions to send/receive data. The federates in Figure 5 are only lightly loaded. Figure 6 shows alternative XY-plots for displaying time series data. We are currently evaluating the efficacy of the various displays.

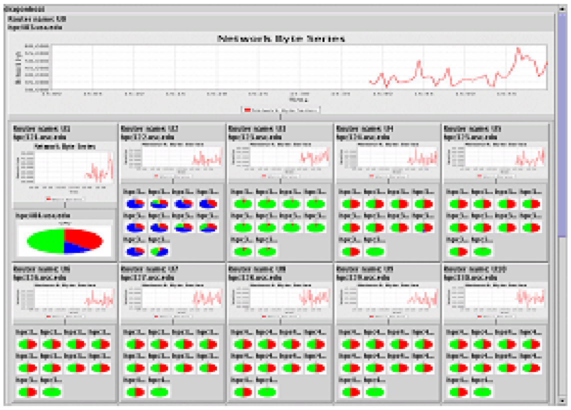

Figure 6
Plotting router network I/O as a function of time

MRI provides a framework for monitoring the performance and resource usage of federations at both at the OS level and at the application federate level. Performance metrics from both levels allow developers to correlate resource usage with JSAF simulation behavior. MRI display clients subscribe to monitoring relay gateways, which periodically push out the monitoring data. This monitoring data is represented in XML for extensibility and flexibility. At the OS level it monitors the CPU load (user, system, idle), memory usage (user, share, cached, free) and network traffic (packets in/out, bytes in/out).

Currently, for such OS level information, MRI uses Ganglia, a cluster monitoring tool from UC Berkeley's Millennium Cluster Project. At the application level it currently monitors JSAF's internal load, heartbeats, and various types of entity counts (remote, local, ground vehicle). See Figure 7.

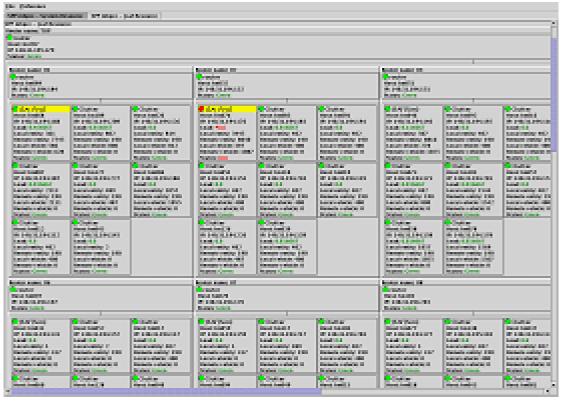

Figure 7.
Custom gauge display for JSAF federates and routers. Green/yellow/red status lights indicate internal JSAF health status. Yellow background

MRI maintains a representation of the federation connection topology in order to generate the gauge displays. This does not violate HLA's information hiding principles of reusability and interoperability, since this topology information is still hidden from JSAF federates, and the federates still have to communicate with each other using HLA RTI's communication infrastructure. The difference is that the connection topology, which is always a vital part of the HLA RTI, is now explicitly represented. Software researchers have argued that explicit representation of software architectures and topologies facilitates better reasoning and understanding [Garlan and Shaw, 1993].

For example, in our case within the context of a topology we can determine the relative importance of node failures/bottlenecks. In Figure 7, node hn068 is highlighted with a yellow background indicating that it failed to emit monitoring data in a timely manner. The failure of node hn068 would bring down the 361 local entities that it is simulating. However, if instead the router node hn084 had failed, then it would have disconnected an entire subtree affecting 6040 entities. If the head router hn207 had failed, then it would result in a forest of disconnect subtrees. Current development is directed at preventing such losses.

Initiation Issues and the "SimPrep Tool"

A major issue when using multiple and geographically distributed SPPs is the effective coordination of intitiation, operation, and termination. There is a large body of research and development literature on various approaches to this issue. (Foster, 1997) While using these existing utilities and tool-kits may perhaps be the smoothest path to an effective implementation, we believe that this is one of the cases where a new tool may be desirable. To illustrate the definition of a need and the implementation of a new tool to serve that need, we will discuss the JESPP "Simulation Preparation" (SimPrep) tool. We do not suggest that it has the broad functionality of a tool-kit like Globus, nor is it suggested that other groups will need or want to develop individual tool-kits in every circumstance.

The preliminary objective of the JESPP exercise is to enable scalable multi-user simulations of synthetic semi-automated battles across multiple SPPs. Accompanying this mission are challenging problems that must be addressed:

1. Overcoming geographical separation that is inherently problematic in terms of latency, and this experiment is particularly interesting due to the requirements of transporting a large amount of data between the clusters.
2. Accommodating the variation of SPP operational policy, e.g. security policy, software and configuration, and network constraints.
3. Implementing interactive computation in a meta-computing environment. This is a new challenge, and requires a new way of doing business. We need to operate the SPPs in interactive mode, as oppose to the more traditional batch-mode model.

Solving the challenges above was accomplished against a backdrop of constraints, which included but were not limited to:

1. Trying to juxtapose between ease of use and flexibility. The GUI application had to be flexible as scripting language scripts. While these challenges are not new to software implementers, they were nonetheless challenges.
2. Having to deal with continuous and large dataset – this along with the need to conduct precise metric. Traditional batch operation on a single or multiple SPPs, while collecting data concurrent to simulations, postpone processing to the post simulation stage.
3. Data collection had to behave as observers and intrude into the collection process, thus be observed.

The experiment process can be decomposed down to four, disjointed processes; along with accompanying software tools we've developed to facilitate each of the stages:

Stage	Applications
Abstraction stage Designing the network and communication topology, and do simulation preparation.	SimPrep and MARCI collector and MARCI GUI
Implementation stage Deploying our software tools and applications to the SPP compute nodes	MARCI application suite deployed and launched applications
Execution stage Conducting the actual experiment by *game players*	JSAF applications, including tree router, JSAF and ClutterSim.
Analysis stage Studying and analyzing the exercise and performance and effectiveness analysis	MRI and post processing and logger tools

Table 1

During the abstraction stage, we planed and designed the network topology. We were primarily interested in how each of the SPPs would be configured and connected (internally) as well as the network connections (externally) between them. To facilitate this process, which was extremely tedious and error-prone, we developed a software program called *SimPrep* that read in as an extensible configuration (network topology specification) file that utilized PERL programming syntax.

During the implementation stage, we used the MARCI applications to query the clusters for resources. Using the resource information and the configuration file defined (designed) in the abstraction stage, *SimPrep* performed resource allocation and map concrete actual compute nodes to abstract network layout.
There were two output files:

(1) the RID, a flattened connectivity file
(2) a mass launch file.

The RID file was in a LISP dialect and was required to be manually stitched into a larger RID file and is understood by the JSAF, clutter, and router applications. The mass launch file was a MARCI specific instruction file on how to launch applications for a specific SPP. Note that the rules for different SPP are specified in the *SimPrep* configuration file.

Once the implementation stage was done, the exercise began. At this point the MARCI application took over. MARCI was responsible for starting and stopping applications – and specifically MARCI along with SimPrep served as the tool with which operators can interface and managed applications on an SPP interactively. This fact contrasted our way of using the SPP with the traditional batch-processing model. The communications between the MARCI GUI and the MARCI collector is a socket-based communication on top of the SSL Layer and it used public/private key for message encryption.

The option to use Globus was limited to resource scheduling and resource discovery. We feel that at this stage, as the experiment policy is still be shaped and defined, Globus would be better used when our way of doing business is solidified. We also feel that Globus does not address the conduct of experiment, instead it serves to facilitate the experiment once the rules of engagement have been defined. For future experiments, we feel the Globus may play an important role – especially in the resource scheduling and discovery stage.

Accomplishments and Future Directions

In December of 2002, the JESPP team ran a successful prototype event using a partition of the USC IBM Linux cluster consisting of some 240 IBM 335 server nodes, with two 2 GHz Xeons, 1 GByte of RAM and both GigE and Myrinet mesh communications. Both the scientists at ISI in California and the operators at JFCOM in Virginia jointly shared control. More than 1,000,000 civilian entities were successfully simulated. They showed appropriate behavior and were stable, even when scanned by the SLAMEM program, emulating two GlobalHawk platforms. To ensure usability and operational validity, about 1,100 warfighting entities were also simulated and controlled in a manner consistent with normal J9 experimentation. Stability and appropriate response to control commands were evident throughout. Several runs were conducted over the course of a week and performance was characterized.

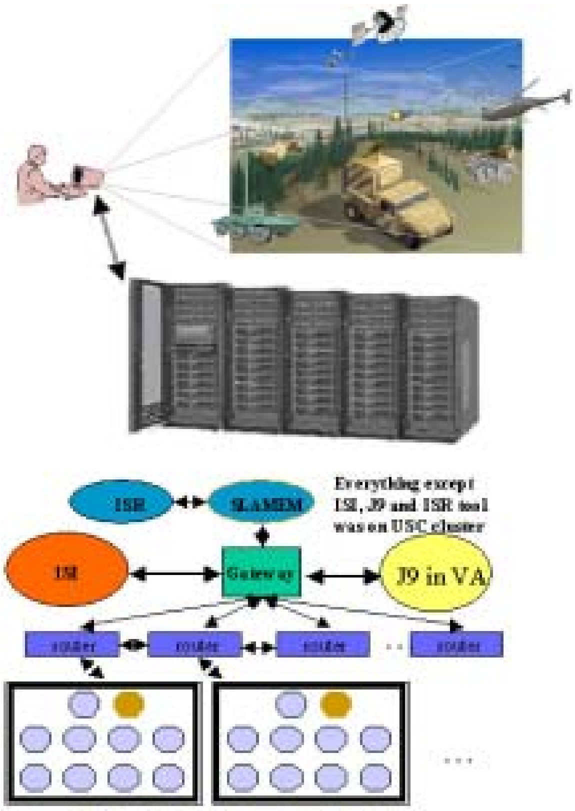

Figure 7
Conceptual diagram of December Prototype Event.

Following the December event, it was decided to show the utility of the DoD's SPP assets by using two Linux clusters, one each at two High Performance Computing Modernization Program sites. Two centers agreed to support this activity, the Aeronautical Systems Center (ASC) in Ohio and the Maui High Performance Computing Center (MHPCC) in Hawai'i. Maui had the larger resource in this case, a several hundred node IBM Linux cluster with Pentium III processors running at 933 MHz and with 1 GByte RAM per node. ASC's cluster was smaller, but exhibited similar processing parameters. With assistance from the HPCMP PET program, the Defense Research and Engineering Network (DREN) was used to interconnect MHPCC, ASC, and the Joint Forces Command in Virginia. Scalability and stability were recorded. Initiation and system configuration issues were studied and addressed.

The group contributing to the JESPP project has made several noteworthy advances in high performance computing. We note the two-router designs, both of which merit further testing and use. Also, a fresh look at performance monitoring on heterogeneous and geographically dispersed SPPs has yielded a robust and useful tool that both generates data and presents status information in a visual manner that is useful for both parallel processing experts and simulation professionals. Some unique initiation problems have resulted in a new approach to complex synchronization issues not adequately addressed by either the SAF family software or by more general meta-computing tools.

Open issues for future work:

There is much to be done, of course, in terms of instrumenting and analyzing the existing system, contrasting performance with that from communications options within the current RTI-s baseline. The more interesting studies here will involve comparisons of new qualitative features of the underlying simulations. An example here is the difference between "reduced capability" and "self-aware" clutter (i.e., do clutter objects interact).

Many of the more interesting near-term development paths can be characterized in terms of "special purpose gateways" (now supportable in view of the reformulated Gateway models). Examples include:

- Translation Gateways: Processors to interpret and convert interest declarations among simulations (federates) that do not use a common interest-enumeration protocol.
- Visualization Gateways: Processors (quite possibly multi-processor collections) to request, collect, process and simplify (e.g., iconify) visualization data within very large simulations. (Current model does most of this work within the visualization workstations, giving rise to ample opportunity for death by communications overload.)
- Input Gateways: The "Collect, Preprocess, Summarize" objectives of the Visualization Gateway could be extended to other processes interested in large subsets of the simulation entities. An important example here is SLAMEM.

That is:

> This is not "merely" a translation of existing (i.e., RTI-s) communications procedures.

This is the first of a number of steps to qualitatively new capabilities that follow from:

1. The scalable communications capabilities of the basic architecture.
2. The additional capabilities of the "intelligent gateways" supportable within this architecture.

Acknowledgements

This work was directed and funded by the Joint Experimentation Group at the Joint Forces Command and the authors wish to thank Gen. Dubik, Dan Franken, and Anthony Cerri. We especially would like to acknowledge the direction, support and counsel of Rae Dehncke who has been the program manger and guiding light for this project. The authors would like to acknowledge the excellent technical support provided by the members of the Computational Sciences Division of the Information Sciences Institute, Gene Wagenbreth, John Tran and Brian Barrett, as well as the guidance and assistance from members of the Scalable Systems Division at ISI, the Dr.s Ke-Thia Yao and Robert Neches. Also contributing greatly were the scientists at OTCI, Dr. Tom Gottschalk, Mike Boughton and Marvin Shannon. None of this could be accomplished without the help of the JSAF team Dr. Andy Ceranowicz, Mark Torpey, Bill Hellfinstine and many others.

References

Ben-Ari, E. (1998). Mastering Soldiers: Conflict, Emotions and the Enemy in an Israeli Military Unit. New Directions in Anthropology, V. 10. Oxford: Berghahn Books.

Brunett, S., Davis, D., Gottschalk, T., and Messina, P., (1998), Implementing Distributed Synthetic Forces Simulations in Metacomputing Environments, Seventh Heterogeneous Computing Workshop, Orlando, Florida

Cebrowski, A.K., & Garstka, J.J., (1998), Network Centric Warfare: Its Origin and Future, Naval Institute Proceedings, 124/1, 28-35.

Ceranowicz, A., Torpey, M., Hellfinstine, W., Evans, J. & Hines, J., (2002), Reflections on Building the Joint Experimental Federation, Proceedings of the 2002 I/ITSEC Conference, Orlando, Florida

Foster, I. & Kesselman C., (1997), Globus: A Metacomputing Infrastructure Toolkit, Intl J. Supercomputer Applications, 11(2): 115 – 128

Garlan, D. and Shaw. M., (1993), An Introduction to Software Architecture: Advances in Software Engineering and Knowledge Engineering, volume I. World Scientific Publishing,.

Hill, R. W., Gratch, J., & Rosenbloom, P.S., (June, 2000). Flexible Group Behavior; Virtual Commanders for Synthetic Battlespaces. Proceedings of the Fourth International Conference on Autonomous Agents, Barcelona, Spain.

Keahey, K.& Gannon, D., (1997), and PARDIS: A parallel approach to CORBA, Proceedings. The Sixth IEEE International Symposium on High Performance Distributed Computing, pp: 31-39, Portland, Oregon

Lightner, G.; Zeswitz, S.; & Graffagnini, J., (1998), Practical insights into the process of extending a federation-a review of the High Level Architecture Command and Control Experiment, Proceedings, 2nd International Workshop on Distributed Interactive Simulation and Real-Time Applications, pp: 41-51, Montreal, Quebec

Messina, P. C., Brunett, S., Davis, D. M., Gottschalk, T. D., (1997, April) Distributed Interactive Simulation for Synthetic Forces, In J. Antonio, (Chair), Mapping and Scheduling Systems, International Parallel Processing Symposium, Geneva, Switzerland.

Sanne, J. (1999). Creating Safety in Air Traffic Control. Unpublished doctoral dissertation, Institute of Tema Research, Linköping University, S-581 83 Linköping, Sweden.

van Lent, M. & Laird, K., (1998). Learning by Observation in a Complex Domain. Proceedings of the Knowledge Acquisition Workshop, Banff, Canada.