Creating a Communication Infrastructure for Simulating Urban Operations

Richard Williams
BMH Associates, Inc.
Norfolk, VA
Williams@bmh.com

John J. Tran
Information Sciences Institute, USC
Marina del Rey, CA
jtran@isi.edu

Bill Helfinstine
Lockheed Martin
Boston, MA
bhelf@lads.is.lmco.com

ABSTRACT

Joint Forces Command is currently developing a large-scale, human-in-the-loop (HITL) federation to support a Joint Urban Operations (JUO) experiment. This resulting JUO HITL federation brings together hundreds of simulations running on both Scalable Parallel Processors and standard desktop computers located at sites ranging from Hawaii to Virginia. This endeavor faced the challenge of developing a communication infrastructure that could support a demanding set of simulation requirements while faced with multiple technological hurdles. These diverse issues, which included high latency rates, huge amounts of network traffic, and organizing large numbers of computers, had to be solved to create both a stable and reliable federation.

This paper shall focus on how the communication infrastructure for the JUO HITL Environment was constructed. It shall describe how the capabilities and demands of the network, machines, run-time infrastructure, and multiple simulations affected the communication topology design. The paper shall also describe the resulting infrastructure used for the JUO HITL federation with a discussion of system strengths and weaknesses. The paper shall use quantitative measurements to illustrate how changes to infrastructure affect network traffic levels and performance. This paper shall also introduce the specific tools created to facilitate the rapid generation and distribution of the complex communication topology. Finally, future development work shall be discussed that should result in an even more robust system with improved implementation features.

ABOUT THE AUTHORS

Richard Williams is a Software Engineer with BMH Associates, Inc., supporting the USJFCOM J9 Experiment Engineering Department. He received a B.S. in Computer Science from the University of Central Florida. He has supported federation development for Attack Ops 00 (AO00), United Vision 01 (UV01), Millennium Challenge 02 (MC02) and the Distributed Continuous Experimentation Environment (DCEE). He is currently developing software to support the Joint Urban Operations (JUO) experiment.

John J. Tran is a researcher at the Information Sciences Institute, University of Southern California. He received both his BS and MS Degrees in Computer Science and Engineering from the University of Notre Dame, where he focused on Object-oriented software engineering, large-scale software system design and implementation, and high performance parallel and scientific computing. He has worked at the Stanford Linear Accelerator Center, Safetopia, and Intel. His current research centers on Linux cluster engineering, effective control of parallel programs, and communications fabrics for large-scale computation.

Bill Helfinstine is a federation developer for the USJFCOM J9 Experiment Engineering Department and a developer and integrator of JSAF (Joint Semi-Automated Forces), as well as primary maintainer and developer of the RTI-s experimental RTI. He has worked in M&S for 10 years, with the last several in support of JFCOM-sponsored exercises, culminating in the Joint Urban Operations (JUO) experiment. He is a Staff Software Engineer at Lockheed Martin Simulation Training and Support Advanced Simulation Center (LMSTS-ASC) in Burlington MA. He received his B.S. in Computer Science and Engineering at the Massachusetts Institute of Technology.

Creating a Communication Infrastructure for Simulating Urban Operations

Richard Williams
BMH Associates, Inc.
Norfolk, VA
Williams@bmh.com

John J. Tran
Information Sciences Institute, USC
Marina del Rey, CA
jtran@isi.edu

Bill Helfinstine
Lockheed Martin
Boston, MA
bhelf@lads.is.lmco.com

INTRODUCTION

Joint Forces Command is currently developing a large-scale, human-in-the-loop (HITL) federation to support a Joint Urban Operations (JUO) experiment. This resulting JUO HITL federation brings together hundreds of simulations running on both Scalable Parallel Processors (SPPs) and standard desktop computers located at sites ranging from Hawaii to Virginia. This endeavor faces the challenge of developing a communication infrastructure that can support a demanding set of simulation requirements while faced with multiple technological hurdles. These diverse problems, which include high latency, huge amounts of network traffic, and organizing large numbers of computers, must be solved to create both a stable and reliable federation.

Recent world events have shown that urban warfare is an intricate and dangerous task. Creating simulations to deal with the complexities of urban warfare is difficult and tedious. We, the authors, believe these simulations can help improve decision making for both the commander and the foot soldier in critical situations, making our efforts worthwhile. This paper shall describe the difficulties we faced, our successes, and our failures in creating an infrastructure to support simulating Joint Urban Operations.

THE CHALLENGE

The goal of the JUO HITL federation is to simulate a high fidelity urban environment for experimentation purposes. This requires simulated pedestrians, civilian vehicles, blue forces, red forces, and sensors to converge to produce a quality C4I picture which can then be used by players. Attempting to tie these simulations together in a reliable manner has brought an array of new challenges which had to be addressed and overcome.

A Balancing of Resources

A number of new features which have been introduced in the JUO HITL environment have been challenging to support. Some features have caused an increase in the amount of data sent on the network and some have required increased computational power. To maximize the size and fidelity of the simulation we are able to produce, these issues had to be addressed and conquered with a variety of techniques.

The JUO HITL terrain brought us new challenges in that it was much more detailed than terrains used previously. Roads in the urban areas have been created at five times Vector Map (VMAP) 1 density. This caused clutter entities to break dead reckoning thresholds and send out updates at rates up to ten times higher than seen in previous exercises. Also, more than 1.8 million buildings are in the two degree by one degree urban area. More than 65,000 of the buildings are Multi-Elevation Surface (MES) structures. Both blue and red sensors were forced to deal with these large numbers of buildings, which made inter-visibility computations an expensive task.

Data Collection has presented developers with new and unique challenges. In prior experiments, such as Attack Ops 00, it had been possible for a single federate to consume all the simulation data for recording. The JUO federation has generated over 200 gigabytes of data per event week. This dilemma required systems to be created which would record the necessary data at the local simulation for a thorough after action review while still being able to provide real time data for simulation controller analysis.

Another difficulty of the JUO HITL Environment was trying to configure, update, organize, and monitor a large number of machines in a quick manner. While systems have been put in place to automate many of these processes sometimes automation is not possible.

Although there were a large number of hurdles to overcome and understand in the creation of the JUO HITL infrastructure our goal was to minimize bandwidth consumption, maximize CPU utilization,

and create the highest fidelity simulation possible with the equipment which we were allocated.

GENERATING THE INFRASTRUCTURE

The JUO HITL environment is comprised of three separate communication structures: A simulation execution system, the simulation, and the data collection system. The simulation execution system, described in "Supporting Distributed Simulation on Scalable Parallel Processors" (Williams, 2003), is used to help generate the structure for the other systems.

SIMULATION INFRASTRUCTURE

The JUO Federation in the early part of 2004 has been comprised of 215 to 360 federates and Interest Management Processors (IMPs). During the May player training event there were 76 Joint Semi-Automated Forces (JSAF) Simulations, 82 Clutter Simulations, 10 Simulation of the Location and Attack of Enemy Missiles (SLAMEM) federates, 52 IMPs, 3 ModStealth 3D viewers, a Dynamic Terrain Simulation (DTSIM), a Track Database, and a C4I Gateway (JSAF Information Paper 2003).

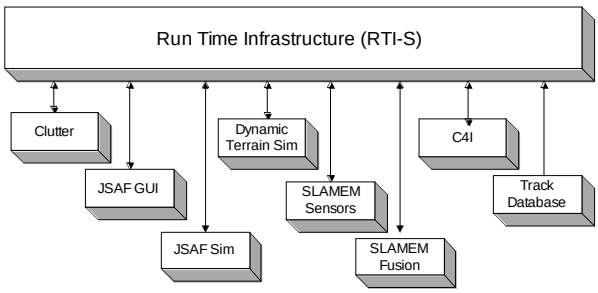

Figure 1. JUO Applications

To bring these applications together we are using RTI-S (Helfinstine 2003) in a point to point mode which allows the implementation of an effective Data Distribution Management scheme. Federations using multicast as their data transport protocol are typically limited to approximately 3,000 subscription regions due to limitations of multicast routers. However, by using point to point protocols and separate routing applications (IMPs) the JUO Federation environment is able to be divided into over 175,000 subscription regions and can still be divided further.

The simulation uses a tree topology generated by automated methods for generic non-specific resources and by hand where an individual resource is required to be configured in a specific manner. As each simulation is started, its RTI component will find itself in a connectivity map file and then connect to its proper parent machine. The connectivity map tells the federate the name of the parent and what protocol to use to connect. RTI-S supports multiple transports with adjustable parameters. For JUO, we use a TCP transport and two different UDP transports. Using these transports, RTI-S has added support for data rate limiting and overflow handling.

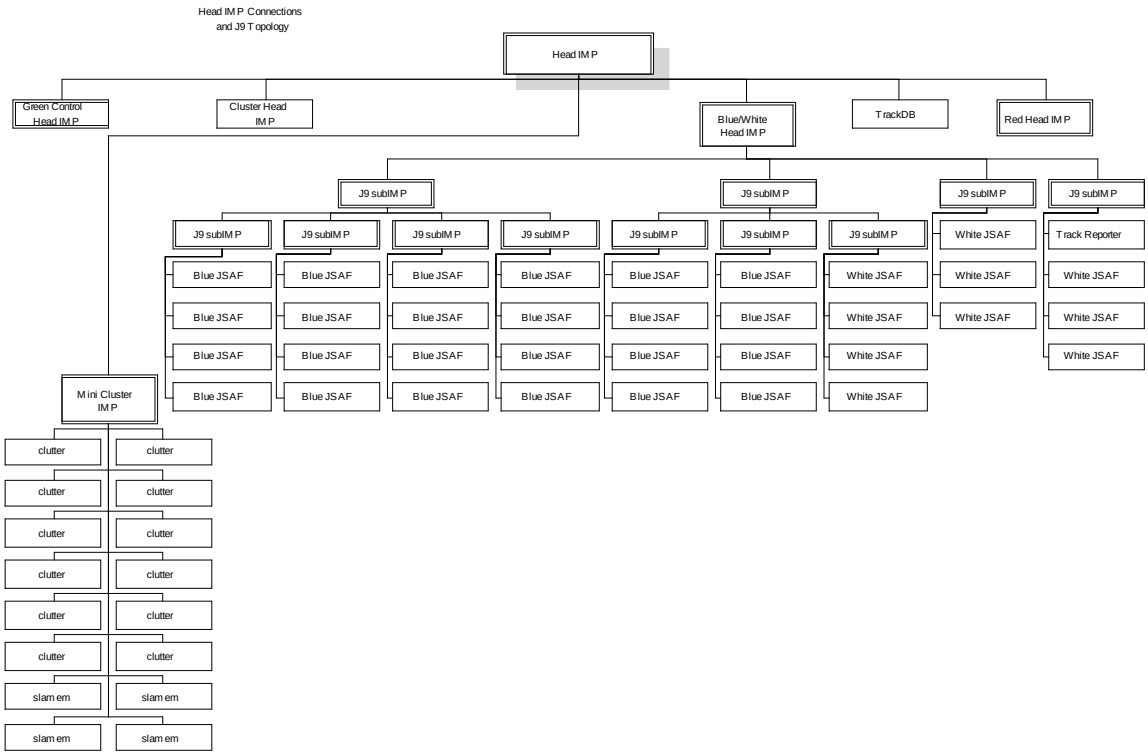

Figure 2. Head IMP and JFCOM Topology

Building the Connectivity Map

In developing the topology we attempted to predict where the weak points might be and limit their impact. For a typical 100Mbit interface we try to limit the child count to five applications or four IMPs. We decided to use a Gigabit interface on the head IMP (see Figure 2) of the federation to prevent spikes in traffic overwhelming the interface. Also, the greater bandwidth allowed us to create a greater spread of branches at the root of the tree which reduced tree depth. We decided to use a Gigabit interface on the J9 mini-cluster head node to save on resources by only using one IMP for 16 child nodes.

We used TCP as our protocol in all situations where latency was low. However, when latency was high, such as the link from Maui, HI to Suffolk, VA, we used the UDP_WAN option. The reason for not using TCP in these situations is that maximum TCP throughput is defined by a function of TCP window size and latency as shown in equation 1(Eshan and Mingyan.)

$$MaxTCPThroughput = \frac{WindowSize}{pingtime} \quad (1)$$

This limited our TCP throughput to Maui using a 64 KB window size with 120 millisecond ping time to around 4 Mb/Sec. We wanted to do some testing with modifying window sizes to improve TCP throughput, however, we have not been able to do so due to schedule constraints.

Federation Data Traffic Patterns

Data Distribution Management (DDM) is a technology that lets federates only receive data to which they subscribe and only publish data which is being requested. However, for this technology to be efficient, it is imperative for Federation Managers to become cognizant of the interactions between the individual simulations.

By analyzing the interaction of the varying federates we were able to strategically place federates at locations that would minimize WAN traffic and increase entity counts. For instance, we decided to place SLAMEM and the Track Database at the same location as the blue cell machines. It was believed that keeping the Track Modification Request and Track Object messages (see Figure 3) at the same location would reduce WAN traffic and allow for a greater number of objects to be simulated.

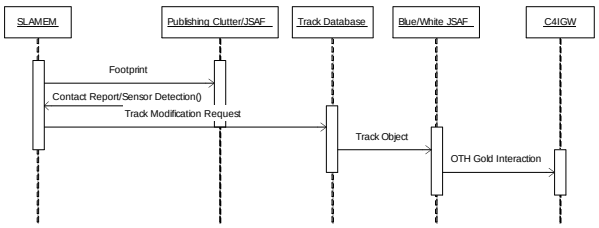

Figure 3. Track Generation Sequence

Early on in testing we tried various configurations with the red cell, such as, running JSAF front-ends (GUIs) on machines at Ft. Belvoir, VA and JSAF back-ends (simulators) on the SPP in Maui, HI. These tests revealed that the communication load for front-ends to control back-ends was actually higher than the simulation load. The current protocol that JSAF uses for communication between GUIs and simulators is very heavy and intolerant of packet loss or high latency. Therefore it works very poorly over WAN links. We then attempted to run both front and back ends on the SPP. To allow the players to interact with the GUI we used Virtual Network Computing (VNC). These tests revealed that the latency between Maui and Virginia was too great to allow for quality user interaction with a GUI updating in real time. Also, as we increased the number of simulations the bandwidth consumption for VNC went up linearly which was not the direction we wished to take the simulation. Eventually we fell back to placing all red simulation on machines at Ft. Belvoir (Figure 4).

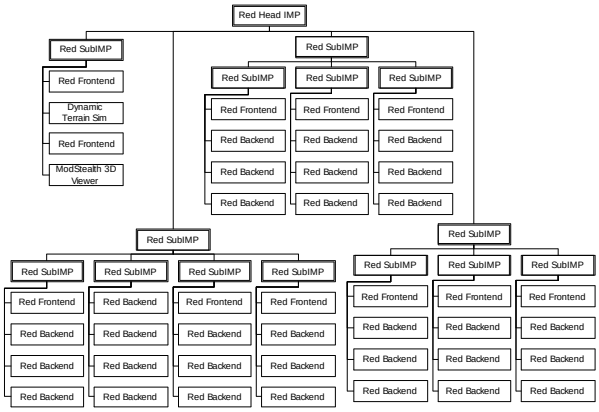

Figure 4. OpFor Topology

A negative aspect of a federation using a point to point protocol versus multicast is initial configuration. In a multicast federation the RTI will subscribe and publish to multicast groups which are automatically handled by a multicast router. In a point to point federation each federate and IMP must be preconfigured with a mapping to its parent federate. For generic resources, such as a Scalable Parallel Processor, where the resource to application mapping is inconsequential we have developed tools to automatically generate and distribute the connection topology. However, for locations where a specific machine is required to run a specific application, the connection topology must be manually configured.

A Look at the Numbers

During the May 2004 player testing event we gathered network metrics to see how much data was being passed between important links. Our goal was to keep traffic at levels that would not exceed the limits of the various interfaces. We were not exactly sure of the numbers we would see since this event was our first opportunity to run with 100,000 urban entities, a full SLAMEM sensor set, and a fully operating player cell.

All data traversing WAN links would either go to or come out of the federation head IMP. By looking at the traffic levels on specific connections, we could see the effects of modifications to the simulation on network traffic. For instance, during the May event SLAMEM changed from generating tracks on all entities to only generating tracks on red. This change improved the ability for players to deal with their perceived picture, and also it dropped the traffic exiting the Track Database by approximately 75 percent.

Most of our predictions on traffic levels were as expected. The greatest producer of data to the federation was the Maui SPP, which was the source for the majority of clutter entities. We were very encouraged to see that the traffic from Maui (see Figure 5) was under 10 Mbps and did not approach our bandwidth limitations.

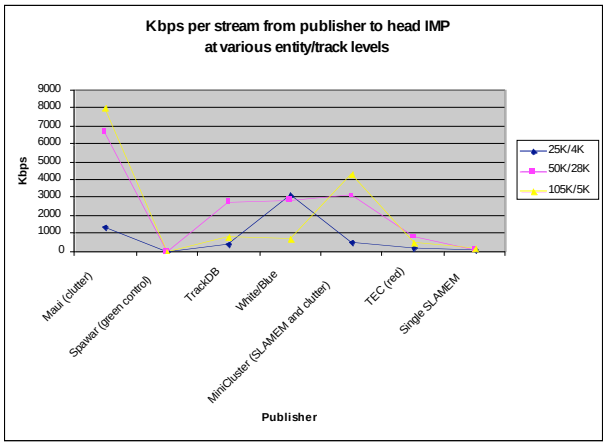

Figure 5. Average traffic levels into head IMP

We were not surprised to see that SPAWAR subscribed (Figure 6) to a substantial amount of traffic and did not publish (Figure 5) very much. SPAWAR's task in the federation was green control, so the only traffic coming out of the site would be for clutter generation. Subscribing applications at the site included JSAF simulations, used to instantiate clutter and observe the federation, and a ModStealth 3D viewer used for trouble shooting and simulation observation.

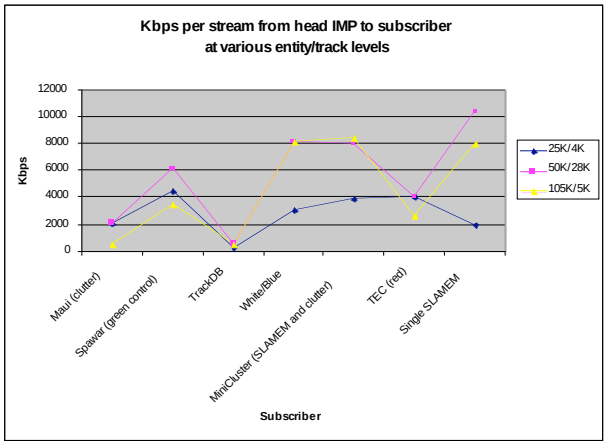

Figure 6. Average traffic levels out of head IMP

There are numerous variables which affect traffic levels within the federation. Clutter federates typically publish entity state information and intersection data to ensure road intersections are simulated realistically. With intersection logic off, federates simulating urban clutter entities will output nearly three times (see Figure 7) as much data as federates simulating rural clutter. If clutter intersection logic is turned on the disparity in network traffic between federates simulating urban entities to those simulating rural grows to a factor of ten.

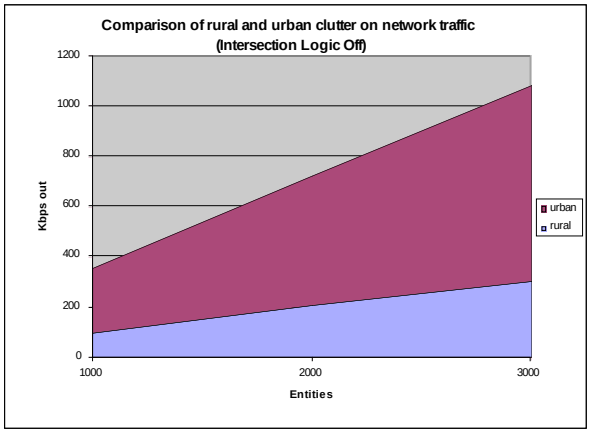

Figure 7. Comparing urban and rural clutter average traffic levels

Limiting subscriptions has become paramount to the JUO federation operations. In previous versions of JSAF, players were able to see and subscribe to all entities if they zoomed out. In the JUO version, players will only subscribe to clutter if they zoom into a fairly close level. Features such as this help maximize the value of DDM.

As we continue our experiment through the year, we will look at traffic levels at both the connection and interface level for all machines at all sites. As changes in the simulation are introduced, we will monitor traffic levels and see how these modifications change data rates.

Ensuring a Solid Federation

To verify the status of each simulation in the federation a new FederateState object was added to the Federation Object Model (FOM) (IEEE Std 1516.2-2000.) Each simulation publishes the FederateState object which contains information regarding memory, processor load, local entity count, remote entity count, software build information, and more. This object allowed applications to be developed which could then display federates joining or leaving the federation, federation entity counts without requiring an application to subscribe to all entities, and an array of troubleshooting information.

We have also created a system to force federation saves at a set periodicity, such as once an hour, so that if a catastrophic failure does occur we can return to a previous save point. This feature has been used a number of times due to network outages, power outages, and federation wide crashes.

DATA COLLECTION INFRASTRUCTURE

The data collection infrastructure is headed up by the researchers at ISI in collaboration with the Topographical Engineering Center to serve the needs that are driven by Future After Action Review Systems (FAARS) team. Its purposes are in three folds:

1. Validation and verification – to provide ground (absolute) truth of the simulation activities,
2. Comprehensive historical record of experiment, and
3. Analysis Tool – to answer complex questions; for example (1) what happened during the event? (2) Who won? (3) How did they win?

Beginning with the JUO experiment, our tools provided a unified method for data collection. In previous experiments, the data collection effort has largely been independent efforts by the various components of the JSAF applications. With improvements, all participating JSAF applications in the JUO federation utilize the same logging mechanism, and as a result, object states, simulation events, and other simulation data are stored in a common database.

There are three stages of data collection:

Table 1. Tools for each of the stage

Stage	Applications	Function
Data Generation	JSAF SLAMEM CLUTTERSIM	the experiment itself
Runtime Query	Aggregator Sqlite Database	capable of answering queries while stage #1 is active
Post Event Query	Rsync dst	support of complex queries

Data Generation Stage

The data generation stage is the experiment itself. During the exercise, at the application level, data is logged to database as followed. Each application publishes message, i.e. its states and interactions, to the federation via the RTI, as the data is being prepared for publication, an intercept library routine is called and these messages are tagged and saved to a local database on the node where that application is running. The logging mechanism is the same for all federate applications. This design is beneficial in two ways: first, in distributed environment, the model is scalable and robust because it does not require a centralized database; and secondly, the model reduces network traffic because data logging is done locally to each node. Note that since the applications only perform query and insert, these databases grow strictly monotonically. As each of the databases on each node grows, the logger tool rotates in a fresh new database to ensure that no single database grows out of range.

Runtime Query

The runtime query tool provides the users the ability to perform real-time (live) queries on the federation's logged data – in short it provides the users the ability to query the data that has been logged as described in the previous section. For example, the user can ask questions such as "how many ground vehicles are out there?" The result from the query is said to be the ground (absolute) truth. Another example query would be how many vehicles does SLAMEM see?" The ability to answer these questions in real time is important and in providing the analysts, and developers an instantaneous snapshot of the system, it enables the capability to make quick critical decisions.

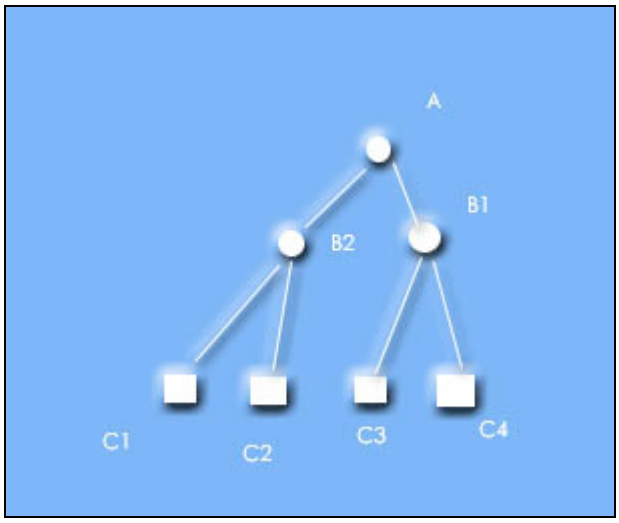

Figure 7. a typical aggregator tree

The runtime query system is comprised of a tree of aggregators and SQL-like database servers. The tree topology is generated automatically using an even-load balance scheme. Figure 7 illustrates a typical aggregator tree. In this case, there are three aggregator nodes and four database (leaf) nodes. In this example, the root aggregator (A1) listens for TCP connection, once an SQL client connects and an SQL command is issued, A1 makes two connections to B1 and B2 and waits for responses from B1 and B2; similarly B1,2 connects A1,2 and A3,4 respectively and pass the query on. Since nodes A1-4 are leaf nodes, they are also database servers, and they return the result (stored in the local node database) back up the tree. Note that the results from these queries can be very large (data size = n x m, where n is the number of leaf nodes, and m is the size of the query result), so we have implemented the aggregator using a stateless model that can support more than one query at a time. In practice, this is not realistic to due to the heavy load that is imposed on the nodes.

Data-staging in support of Analysis

The data staging is a way to move data collected at the distributed nodes into a single (monolithic) database, so that users can use it to perform complex queries to answer difficult questions about the exercise. The

reason that complex queries cannot be executed on distributed database system is due to the fact that some database queries requires operations on the global dataset, e.g. summation. One example question is "give us a track or sequence of a vehicle with the ID xyz" – the result of the query would be a chronological sequence of activities by that vehicle. Two of the challenges of the data staging are (1) geographically dispersed sites (TEC, MHPCC, j9), (2) and the amount of log data (200+ GIG) generated during the experiment.

The data staging process can be broken down into three separate steps, all of which occurs at the end of each simulation day: (1) data pull, (2) data convert, and (3) data import. All three steps are all executed on our terabyte database machine.

Data Pull

We implemented the data pull tools using a data synchronization tool called rsync, which is readily available on most Linux distributions. Rsync is robust, fast, and works over a number of transport layers including SSH and Kerberos-enabled SSH. Rsync only pulls files that have been changed since the last pull, thus reducing the network traffic and wait time for each of the subsequent data pull operations. Typically, an end-of-day data pull on XYZ gigabytes for all of the three sites (TEC, j9 and MHPCC) takes about XYZ time. Note that data pull should not be confused with issuing a query on the root aggregator node during the runtime.

Data Conversion

The data pulled from the distributed sites is a highly compressed file that has multiple tables. Each of the database files requires conversion into a format that can then be inserted into the MySQL database. The amount of data being pulled each night is extremely large and since the conversion routine can be decoupled and is embarrassingly parallel, we devised a simple scheme to farm out the decoding jobs to all 17 nodes of the mini local cluster. In doing so, we are able to achieve close to 17x speed up. We are implementing a number of robustness features, for example the tag-and-proceed technique to ensure that no database table is decoded twice. On average, it takes 5 hours to decode 3.5G amount of data.

Data Import

The data import stage is a simple algorithm that does linear time visit on each of the decoded datum and insert it into the database. We made some interesting observations during one data import experiment:

- Data conversion can be done in parallel,
- Data loading does not enjoy the same level of speed up

We suspect that this is due to the fact that the MySQL database server is I/O bounded so at the moment most of the tuning is done to the database server.

After all the data has been loaded into the database, the other members of the FAARS team can conduct the complex queries such as those mentioned earlier.

FUTURE WORK

While we feel we have been successful at generating a strong foundation for simulating urban operations, there are still many obstacles left to overcome.

Work remains to be done in improving the structure of the simulation. A tree structure will not scale and is not optimal for passing data from publishing federate to subscriber. Work is being done to bring a mesh router (Helfinstine 2003) to the JUO federation which should improve scalability.

A number of times during testing it was discovered that a problem in the federation was a result of a single federate publishing an excessive amount of data. In another instance, it was discovered that if the players used certain panels within JSAF that their processor load would go to near 100 percent. While we have methods to analyze metrics within the individual simulations, we currently do not have an automated method to warn federation managers of these problems as they occur. Some type of metric alarm system would aid greatly in operating a long running, stable federation.

A job which is currently underway is the improvement of the JSAF control approach. This new system will be based on a command/query protocol verses the existing shared database protocol. This change should allow us to split GUIs from simulators more effectively.

More work can be done to simplify the initial setup of the JUO federation. Currently the process of initially setting up a simulation run is quite involved and can take a qualified operator a full day to set up. Although a number of utilities have been created to aid in this process, we think it can still be improved further.

CONCLUSION

We have successfully designed a system which connects hundreds of machines from across the country running a variety of simulations. These simulations have been successfully tested using over 100,000 civilian and OpFor entities being tracked by a variety of simulated sensors in a detailed urban environment.

Our efforts have led to the development of a data collection structure which can provide real time simulation information without overloading the network. This information can then be used to help gauge the successes and failures of players as they occur.

We have designed and developed systems which allow us to verify that simulations are running correctly, ensure network connectivity and monitor federate status. They have also been designed to give real time federation level summaries to support simulation execution. These tools have not simply been nice to have, they have been essential to troubleshooting problems and ensuring a solid federation.

Possibly the most exciting development for the JUO HITL federation has been the impressive effectiveness of data distribution management. DDM has increased the capabilities of this federation by orders of magnitude, while working in flawless fashion from event to event.

While we are encouraged by our accomplishments to date we do not feel as if we are finished. Much more work can still be done to create a structure which can support the largest simulation possible with limited resources.

ACKNOWLEDGEMENTS

The authors would like to thank Andy Ceranowicz, Steve Bixler, Rae Dehncke, Jason Boyer, Glenn Goodman, Mark Torpey, Jackie Tran, Phil Colon, and all the various site support personnel for their help in making this work.

REFERENCES

Eshan, N. & Mingyan, L., *Analysis of TCP Transient Behavior and Its Effect on File Transfer Latency,. Retrieved June 16 from* http://www.eecs.umich.edu/~mingyan/pub/icc03.pdf

IEEE Std 1516.2-2000. IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) --- Object Model Template (OMT) Specification. The Institute of Electrical and Electronic Engineers. March 2001.

Williams, R., & Tran, J. (2003). Supporting Distributed Simulation on Scalable Parallel Processors, *Proceedings of the 2003 Interservice/IndustryTraining, Simulation and Education Conference.*

Helfinstine, B., Torpey, M., & Wagenbreth, G. (2003). Experimental Interest Management Architecture for the DCEE, *Proceedings of the 2003 Interservice/IndustryTraining, Simulation and Education Conference.*

JSAF – Joint Semi-Automated Information Paper, (2003). Retrieved June 17, 2004, from http://www.mstp.quantico.usmc.mil/modssm2/InfoPapers/INFOPAPER%20JSAF_files/INFOPAPER_JSAF.htm