Data Fusion of Geographically Dispersed Information: Experience with the Scalable Data Grid

Ke-Thia Yao, Craig E. Ward & Dan M. Davis
Information Sciences Institute, USC
Marina del Rey, California

ABSTRACT

Test and Evaluation (T&E) professionals today often face the distribution of data they need to use over significant distances. Some of these data sources are too large for easy transmission due to costs, delays, losses, security and administrative burdens. ISI has been working with the Joint Forces Command (JFCOM) on its trans-continentally distributed battlespace simulations and they have conceived, architected, implemented and tested a system that uses the data in place. This has been characterized as a Scalable Data Grid (SDG), which uses data cubes for rapid and focused retrieval. It is necessary that these data are available in a timely manner, organized for ease of access, securely stored, and easily manipulated for data mining. Suggestions are offered listing T&E situations to which the SDG approach would seem applicable. Having read the paper, the T&E researchers should be able to make a valid assessment of the utility of this approach.

AUTHORS

Ke-Thia Yao is a project leader and research scientist at the University of Southern California Information Sciences Institute. His research has been centered on the JESPP project, which has the goal of supporting very large-scale distributed military simulations involving millions of autonomous agent entities. He has developed a suite of monitoring/logging/analysis tools to help users better understand the computational and behavioral properties of large-scale simulations and has designed the Scalable Data Grid for distributed data management of large archives. He received his B.S. degree in EECS from the University of California, Berkeley, and his M.S. and Ph.D. degrees in Computer Science from Rutgers University, New Brunswick, New Jersey. kyao@isi.edu

Craig E. Ward is a Parallel Computer Systems Analyst at the Information Sciences Institute. Much of his recent research has focused on large-scale data management in the defense and the health sectors. His concentration has been on open source tools. Previously, he performed computer analysis for law enforcement in California. He has a B.A. in History from the University of California, Irvine, and an M.S. in Computer Science from Loyola Marymount University in Los Angeles California. cward@isi.edu

Dan M. Davis is the Director, JESPP Project, Information Sciences Institute (ISI), University of Southern California (USC), focusing on large-scale distributed DoD simulations. As the Assistant Director of the Center for Advanced Computing Research at Caltech, he managed Synthetic Forces Express, bringing HPC to DoD simulations. Prior experience includes work as a Software Engineer at the Jet Propulsion Laboratory and at a classified project at Martin Marietta. He saw duty in Vietnam as a Cryptologist in the USMC and he retired as a Commander, Cryptologic Specialty, U.S.N.R. He received B.A. and J.D. degrees, University of Colorado in Boulder. ddavis@isi.edu

Data Fusion of Geographically Dispersed Information: Experience with the Scalable Data Grid

Introduction and Background

Test and Evaluation (T&E), while once done via handwritten analyses, has increasingly come to rely on computer collection and manipulation of data from a myriad of sensors and sources. In common with most of the rest of the defense community, T&E can now collect more data than it can easily analyze. More computing power allows for increases in the breadth and depth of the information collected. Now, that same computing power must assist in identifying, ordering, storing and providing easy access to that data. Fast networking allows large clusters of high performance computing resources, often distributed trans-continentally, to be brought to bear on the test and evaluation. This increase in fidelity has correspondingly increased the volumes of data that tests are capable of generating.

Coordinating distant computing resources and making sense of this mass of data is a problem that must be addressed. Unless data are analyzed and converted into information, testing will provide only a fraction of the knowledge that is possible. For the US Joint Forces Command (USJFCOM) *Urban Resolve* exercises, which are used to evaluate new systems and sensors, ISI developed a distributed logging system to capture publish-and-subscribe messages from the High-Level Architecture (HLA) simulation federation. For a two-week exercise, omitting nonessential data, this system logged over a terabyte of data (Yao & Wagenbreth, 2005).

In addition to the SDG approach, the ISI team found that Hadoop provided a scalable, but conceptually simple, distributed computation paradigm which is based on map/reduce operations implemented over a highly parallel, distributed filesystem. Map/reduce implementations of K-Means and Expectation-Maximization data mining algorithms were developed to take advantage of the Hadoop framework. This filesystem dramatically reduced the disk scan time needed by the iterative data mining algorithms. It was found that these algorithms could be effectively implemented across multiple Linux clusters, connected over reserved high-speed networks. The data transmission reductions observed should be applicable in most T&E situations, even those that use lower bandwidth communications.

For this analysis, Hadoop jobs were created in order to experiment with the data mining performance characteristics in an environment that was based on connections to sites across widely dispersed geographic regions. Specially configured Linux Cluster computers were installed at the Information Sciences Institute (ISI) in California, at the University of Illinois – Chicago (UIC) in Illinois, and at ISI-East in Virginia. All of these machines had large disk storage configurations, were located on network circuits capable of 10 Gigabits per second transmission, and remained dedicated to this research. The machine at ISI in California served as a control. Special network connectivity was established between UIC and ISI-East to test Hadoop across that geographic distance.

AGILE DATA FRAMEWORK

In general, T&E analysts and customers have been interested, not only in system data, but also in how well higher level mission tasks and objects are satisfied. An Measure of Effectiveness (MOE) is a question, or a measure, designed to illuminate how well particular mission tasks are satisfied with respect to a system (Gentner *et. al.*, 1996).

A Measure Of Performance (MOP) is typically a quantitative measure of a system characteristic used to support an MOE. For example, sample MOE questions are "Can the enemy be located?", or "Can the system effectively detect movement within urban environment?". MOPs supporting these MOEs are typically statistical in nature.

Data loggers do extremely well at capturing detailed operational data. ISI's distributed data loggers have captured terabytes of experimental data for the Urban Resolve Phase I exercises at JFCOM(Wagenbreth, 2010). Operational data included individual entity state changes and interactions among the entities. Depending on the type of entity, entity state changes may include location, orientation, and velocity. For vehicles, additional attributes may include external lights-on and engine-on. Interactions may include collision, damage assessment, sensor detection, and contact report.

The logged data collected from the test is often at too low a level to be of direct use to the T&E analysts. Information needs to be abstracted from the logged data by collation, aggregation and summarization. In order to perform this data transformation an analysis data model (ADM) has to be defined that is suitable for analysts and decision makers. ISI used a multi-dimensional data model as way representing the information from their perspective. Next, a Logging Data Model (LDM) of representing the collected data has to be defined.

Then, to bridge the gap and connect two data models, ISI defined an abstraction relationship that mapped the logging model to the analysis model. This was a part of the Scalable Data Grid toolkit (Yao and Wagenbreth, 2005).

ISI developed a Sensor/Target Scoreboard that provided a visual way of quickly comparing the relative effectiveness of individual sensor platforms and sensor modes against different types of targets (Graebener *et al.*, 2003; Graebener *et al.*, 2004). The Sensor/Target Scoreboard was a specific instance of the more general multidimensional analysis (Kimball *et al.*, 1998).

The Sensor/Target scoreboard is an example of two-dimensions of a multidimensional cube. Its two dimensions can be the sensor dimension and target. One can imagine extending the scoreboard to take into account, say, weather conditions and time of day. This would add two more dimensions to form a four-dimensional cube.

The ADM model consists of two key concepts dimensions of interest and measures of performance. Dimensions are used to define the co-ordinates of multidimensional data cubes. The cells within this data cube are the measure values.

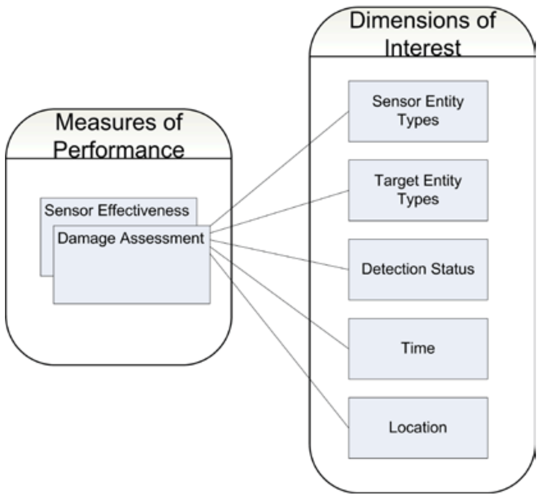

Figure 1. Dimensions of Interest and Measures of Performance

Dimensions of Interest

For large simulations, like the Urban Resolve exercises, the magnitude of data collected ranges into the terabytes. Dimensions categorize and partition the data along lines of interest to the analysts. Defining multiple crosscutting dimensions aids in breaking the data into smaller, orthogonal subsets.

Dimensions have associated measurement units, or coordinates. Choosing the granularity of these units aids in determining the size of the subsets. For example, depending of the dynamic nature of the phenomenon that the analysts are trying to study, they may choose to define the time dimension units in terms of minutes, days, weeks or years.

Another dimension example is in terms of simulation entity groups. For the sensor dimension in the sensor/target scoreboard, the analyst may want to group together sensors by the type of platform: high-flying UAVs (unmanned aerial vehicles), mid-altitude UAVs, OAVs (organic air vehicles) and UGSs (unattended ground sensors). The targets may be group together, for example, by transportation mode: air, ground and sea.

Hierarchical dimensions define how a coordinate relates to other coordinates as its subset. It serves to group together similar units from the analyst's perspective. By defining hierarchical dimensions, analysts in-form the system on how to aggregate and summarize information. For example, the analysts may want to subdivide the sensor platform category into the sensor modes: MTI (moving target indicators), SAR (synthetic aperture radar), images, video, and acoustic.

Hierarchical dimensions can be viewed as a partial ordering relationship. At the top node of the partial ordering is the set containing all the coordinates. The bottom nodes of the partial ordering are singleton sets contain-ing just one coordinate. Edges between nodes indicate superset/subset relationship. A node's parent is its superset. A node's child is its subset.

Nodes in hierarchical dimensions are also given coordinates. Calling the coordinates for non-singleton nodes "abstract coordinates" is the accepted practice. Coordinates for singleton nodes are the concrete coordinate of the single element in set.

Multiple decompositions of the same dimension are also useful. For example, there may be many different simulation federate types playing in the federation. The analysts may want to verify how each federate responds to the sensor contacts, so they can define the category to be the type of federates in which they are interested.

Measures of Performance

After the data have been partitioned along lines of interest, the data subsets may still be large. Measures provide quantitative ways of characterizing the data subsets. A key characteristic that measures should have is that they are can be aggregated. The hierarchical crosscutting dimensions partition the data into a hierarchy of subsets. The measure must be able to provide a meaningful summarization. To be computational efficient the measure aggregation operator must satisfy the associative and commutative properties — the measure of a set must be computable from the measures its sub-sets.

In the case of the sensor/target scoreboard the measure of performance is simply an integer count of the number of times a sensor has detected a target. The aggregation operator is the addition operator.

RTI-s, a highly-scalable implementation of the RTI (Helfinstine *et. al.*, 2003), provides an interceptor plug-in framework that exposes calls to this HLA interface to registered plug-ins, see Figure 2. The SDG exploit this plug-in to intercept and log messages federate attribute updates and interaction sends. With respect to the RTI, the contents of these messages are raw binary strings. RTI provides publish-and-subscribe facilities to exchange these messages, but not to decode their contents. To provide the query and analysis capabilities, these messages must be decoded with respect to the Federation Object Model (FOM), the simulation definitional statement.

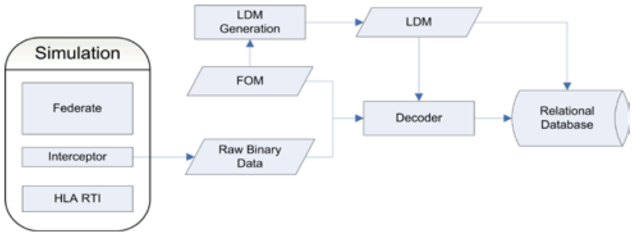

Figure 2. Logging data flow

Cube Computation

The data is made up of the facts or observations from a test. The facts have to be aggregated according to how hierarchical dimensions are defined. The cells that correspond to the abstract coordinates of the cube have to be computed. Here it is assumed that the aggregation operator satisfies the associative and commutative properties. Given these assumptions, measures can be efficiently computed for all the abstract coordinates by doing a bottom-up traversal of the partial ordering hierarchy.

T&E USE OF HADOOP

Easily obtained by anyone in the T&E community, Hadoop is an open source system, hosted by the Apache Software Foundation. It provides a reliable, fault tolerant, distributed file system and application programming interfaces. These enable its map-reduce framework for the parallel evaluation of large volumes of test data.

T&E professionals would likely find that the simplicity of the Hadoop programming model allows for straightforward implementations of many evaluation functions. The popular Java applications have the most direct access, but Hadoop also has streaming capabilities that allow for implementations in any preferred language.

Several other communities that need to handle large amounts of data are using map-reduce implementations to manage that data. Google started using a map-reduce system internally before 2004 (Dean *et al.* 2004). Yahoo runs the largest Hadoop cluster, running over a Linux cluster of over 10,000 cores (Yahoo 2008). Many vendors, *e.g.* Amazon, utilize Hadoop as part of their cloud computing service. A list of organizations who make use of Hadoop can be found at the Hadoop wiki (Powered By, 2009).

In the 2008 terabyte sort challenge, Yahoo won by using Hadoop to sort 1 terabyte of data in 209 seconds (O'Malley *et al.* 2008). That cluster consisted of 910 nodes with 2 quad core 2GHz Xeons and 4 SATA disks per node.

Hadoop Distributed File System

The Hadoop Distributed Files System (HDFS) runs on top of a native file system and is only accessible through the Hadoop Application Programming Interfaces (APIs).

HDFS configurations distribute data in equally sized chunks across the available data nodes. This division of data works best for large files that can be stored as multiples of the chunking size configured for the HDFS. If the files are smaller than the chunking size, the HDFS will waste local file system resources with empty, allocated bytes.

Redundancy and fault tolerance are achieved by replicating these chunks on multiple nodes. Hadoop attempts to run the map operations on copies of the data local the mapping task. This reduces the amount of data that needs to be moved around.

ISI experiments used varying HDFS configurations. One configuration kept all nodes within a single rack. Another divided the nodes across half of the continental United States.

Hadoop has three different node types: nodes for processing tasks, nodes for storing data, and a single node, called the name node, to coordinate the others. The tasks that are assigned to processing nodes are monitored for status. If a task appears to fail, it can be reassigned to another processing node. The assignments attempt to keep processing and data near each other, limiting the strain on any underlying communications resources, such as a network.

DISTRIBUTED DATA MINING ALGORITHMS

There are some T&E settings in which discovering un-anticipated or novel data would be useful (Grehrig, 2004). Data mining is a way of finding patterns in what otherwise would be random data. Many data mining algorithms are iterative in nature. They require the data to be scanned several times during the mining process. These algorithms can become prohibitively expensive for very large data sets that do not fit into memory, and have to be stored on disk. Sequential disk access on a single disk can be several orders of magnitude slower than memory access. Hadoop with its potential to access thousands of disks in parallel provides a way of addressing this problem.

In addition, in some situations the test data themselves be stored in a distributed fashion. For example, for JFCOM's Urban Resolve exercises, ISI implemented a distributed logger that stored High-Level Architecture Runtime Infrastructure (HLA RTI) messages locally, *in situ* where the messages were generated (Yao & Wagenbreth 2005; Graebener *et al* 2003). Using Hadoop provides a convenient way to process the data without having to move it to a centralized location.

Two Clustering Algorithms

To test the feasibility of this approach the ISI team implemented two data mining clustering algorithms in Hadoop: K-Means and Expectation-Maximization (EM).

K-Means is a popular data mining clustering algorithm that assigns a set of data instances into clusters (or subsets) based on some similarity metric. The K-Means algorithm requires three inputs: an integer *k* to indicate the number of desired clusters as output, a distance function over the data instances, and the set of *n* data instances to be clustered. The distance of a data instance to itself is zero. The greater the distance between two data instances, the less similar the instances are. Typically, a data instance is represented as a numerical *vector*. The output of the algorithm is a set of *k* points representing the mean (or the center) of the *k* clusters. Each of the *n* data instances is assigned to the nearest cluster mean based on the distance function.

Below is pseudo code for the K-Means algorithm:

```
1. Generate an initial guess for the $k$ cluster (for example, by randomly selecting $k$ points from the data instances as the $k$ means).
2. Assign each of the $n$ data instances to the nearest cluster mean.
3. Based on the data instance assignment, compute the new cluster mean for each of the $k$ clusters.
4. While not done, go to Step 2.
```

Figures 3 illustrates some results of K-Means clustering. Figure 3 shows K-Means correctly finding the means of the 3 distinct clusters. That is, given a set of points generated for this dataset, the algorithm correctly discovered the patterns in the points.

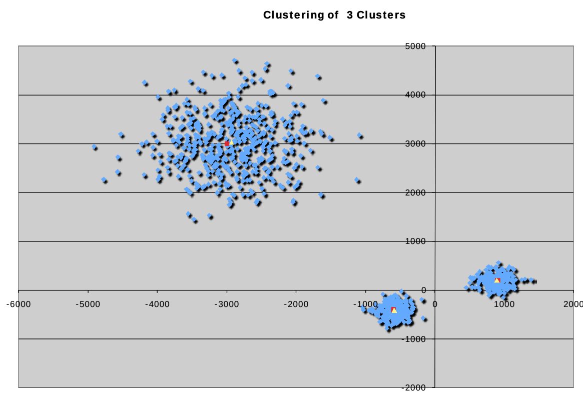

Figure 3. K-Means clustering of three distinct clusters of points.

The EM algorithm can be viewed as a probabilistic generalization of the K-Means algorithm. Instead of representing a cluster by just its mean, EM represents a cluster by its mean and its variance (or covariance matrix), *i.e.* each cluster is represented by a Gaussian distribution. In addition, each cluster is associated with a weight, representing the probability of selecting the cluster. The sum of these *k* cluster weights is equal to one. This representation is called a Gaussian mixture model.

The steps of the EM algorithm are similar to the K-Means algorithm. In Step 1 the initial guess now includes the *k* means, *k* variances, and *k* cluster weights. The assignment in Step 2, also known as the Expectation Step, is now slightly more complicated. Instead of assigning each data point to one cluster, each data point is assigned to each cluster with a probability based on a Gaussian distribution. In Step 3, the Maximization Step, the *k* means, *k* variances, and *k* cluster weights are recomputed based on the probabilistic assignment from Step 2.

Hadoop Implementation

Only described the Hadoop implementation of the K-Means algorithm will be described, the EM Hadoop implementation being similar. There exists a variety of ways to generate the initial guess in Step 1. If there is *a priori* knowledge of the range of values of the data, then *k* means can be generated randomly using a uniform distribution. Otherwise, scanning the data instances once will allow computing the range values. Or, scanning the data instances and randomly select *k* instances as the means is possible. To simplify the algorithm description it shall be assumed that there is *a priori* knowledge.

Step 2 corresponds to the map operation. Map functions have the form:

Map: *(in-key, in-value)* → *list (out-key, out-value).*

In this case, the *in-key* is null, and the *in-value* is the data instance vector. The *out-key* is an integer from 1 to *k* representing the cluster identifier, and the *out-value* is a list of pairs, where each pair consists of the data instance vector and the integer one.

K-Means Map: *(null, data-instance)* → *list (cluster-id, (data-instance, 1))*

Step 3 corresponds to the reduce operation. Reduce functions have the form:

Reduce: *(in-key, list (in-value))* → *list (out-key, out-value>*

In this case the input *(in-key, in-value)* is the output of the K-Means Map (*cluster-id, (data-instance, 1))*. For each *cluster-id*, the reduce operation sums all the *(data-instance, 1)* pairs associated with that cluster-id.

K-Means Reduce: *(cluster-id, (data-instance, count))* → *list (cluster-id, (sum-of- data-instances, number-of-instances))*

Here the *sum-of- data-instances* divided by *number-of-instances* is the mean of the cluster.

Here is a simple, but naïve, Hadoop implementation of the K-Means algorithm:

1. Random generate *k* points as initial *k* means.
2. Apply K-Means Map & Reduce.
3. While not done, go to Step 2.

Test Environment Setup

Data mining Hadoop jobs were created for the SIMC-IC project to experiment with the performance characteristics of Hadoop in an environment that provided high-speed network connections to sites across large geographic regions. As mentioned before, clusters in California, Illinois and Virginia were connected via a high-bandwidth link.

Each cluster machine was comprised of:

- 10 nodes
- 5.3 TB local disk
- 2 Clusters running Fedora 8
- 1 Cluster running Debian
- 1 10GigE network card
- 1 1Gig card for management only
- Dual Quad Core (8 cores per node) CPUs

The version of Hadoop used for the experiments was 0.17.2.1. Each cluster used the Java SE Runtime Environment 1.6 (build 1.6.0_11-b03).

Hadoop clusters were configured using the available nodes such that both the control Hadoop cluster and the distributed Hadoop cluster had the same number of nodes, one name node and nine nodes running data and job task services. The only difference being that the control cluster used only local network connections while the other used wide area network connections.

For the wide area network Hadoop cluster, two configurations were used. One configuration used the default network resources and one used dedicated Internet 2 high-bandwidth lines reserved for short time periods.

Data Load

In addition to the data mining jobs developed, the ability of Hadoop to load and store data was tested. A simple data load of six 1.2-gigabyte files was performed using the default settings, each block of data replicated on three nodes. All time data was collected from the *time(1)* command.

Table 1: Data Load Test Results

	User	System	Elapsed
ISI Local	44.85	22.09	2:05.69
ISIE/UIC (standard)	46.98	18.38	14:27.75
ISIE/UIC (fastnet)	49.18	18.94	29:20.78

As would be expected, the quickest data loads were with the local nodes configuration. The actual processing times were not that much different for each configuration. The major difference was in clock time indicating that the distributed systems spent significant time in suspended wait states while the network subsystems performed their functions. The Fastnet version using Internet 2 actual took longer elapsed time than the standard version. However, during the execution of the Fastnet version, Java network exceptions being thrown were observed.

Data Mining Jobs

Two implementations of the K-Means algorithm were used to test the processing capabilities of Hadoop. An expectation-maximization job was also developed, but this job was not used for this experiment. The UIC Angle dataset was searched for points within the data where the data clustered. One implementation used a “naïve” approach while the other used a more efficient, “smart” approach. The naïve implementation did not use the combine step allowed by the Hadoop API. This resulted in much more network usage as more data had to be passed around between the task nodes. The smart implementation made use of this step and greatly reduced the amount of data exchanged.

The K-Means jobs iterated over the data set with an initial set of cluster points, each time updating the set of cluster points to better fit the data, each resulting set of cluster points becoming the input for the next iteration. When either the points stopped significantly changing or the maximum number of iterations was reached, the job stopped.

For development and initial testing, the job was tested using points randomly generated using known center coordinates. The results of a run were expected to match the input provided to the random point generator.

Table 2: K-Means Results

	User	System	Elapsed
ISI Local (smart)	1.68	0.18	1:37.76
ISI Local (naïve)	6.55	0.92	40:38.64
ISIE/UIC (smart/stand)	1.67	0.19	1:52.80
ISIE/UIC (smart/fastnet)	2.25	0.27	8:25.08
ISIE/UIC (naive/stand)	5.35	0.96	1:12:03
ISIE/UIC (naive/fastnet)	8.40	1.72	2:14:16 KILLED[1]

As with the data loads, the data mining jobs performed best on the local nodes setup. The differences between local and networked systems are not as pronounced as with the data loads. This is likely due to the ability of Hadoop to process chunks of data in a "rack-aware" manner. The smart implementations tended to not require long haul network services and were able to process data in what to them was a local manner. Again, the Fastnet version took longer elapsed time than the standard version.

[1] The naive run was killed at the elapsed time in the seventh job iteration. The maximum number for a run is 32.

NETWORK UTILIZATION

In the previous section our experiments exercised Hadoop across differing network configurations. One configuration used the "normal" connectivity found in the network while another ran Hadoop over special high-speed links with a theoretical peak throughput of 10 Gbps. But, Hadoop results did not reflect the advantage of the high-speed links.

To rule out the possibility the high-speed links were faulty another software system was used to get independent measurements. The tool used to test this capability was the Meshrouter, which was designed for high throughput HLA RTI communications (Barrett & Gottschalk, 2004; Brunett & Gottschalk, 1998). The tests show the Meshrouter application is capable of achieving 1.5 Gbps with a single TCP stream, and up to 5 Gbps with combined streams.

Based on this throughput experiment it was reasoned that Hadoop was not able to take full advantage of the high-speed network. As mentioned previously it was observed that Java generated network exceptions during the execution. Although Hadoop is designed to be fault tolerant, the exceptions most likely slow downed its execution.

Moreover, in order to achieve 50% capacity of the high-speed network, the Meshrouter application required several TCP streams. It is suspected that even without the network exceptions Hadoop will not be able to take full advantage of the ten-Gig network.

Below, the details of the high-speed network throughput experiment using the Meshrouter are described. The Meshrouter and associated applications implement interest managed communication (RTI) utilized by several entity simulators in general use. Test programs named publish and subscribe were used to exercise the network in a controlled and repeatable manner. The Meshrouter is a complex real-world application.

The bandwidth experiments were done using the standard ISI Meshrouter formalism for interest-managed communications. A schematic of the Meshrouter is shown in Figure 4.

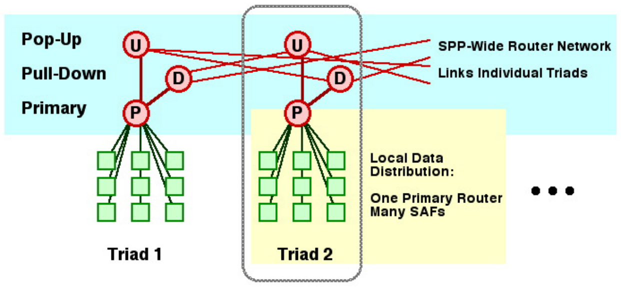

Figure 4. Schematic Meshrouter Topology

The overall communications scheme consists of collections of processors (labeled "SAFs" in this legacy diagram) each communicating with a specified "Primary" router (P). Interest-limited message exchange among the various basic "Triads" is done using a network of additional "Pop-Up" and "Pull-Down" routers. As is described in (Barrett & Gottschalk, 2004), the three routers on a triad are instanced as separate objects within a single Meshrouter process.

The execution of actual message transfer is implemented by a software stack as shown in Figure 5.

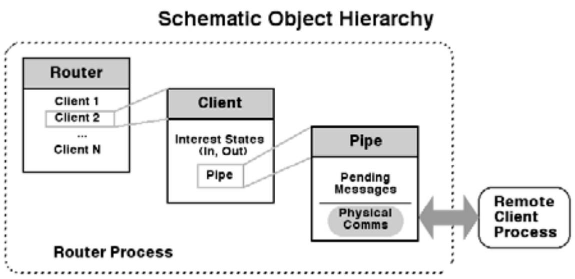

Figure 5 Factored Meshrouter implementation, with application-specific communications primitives.

The results reported here use an RTI-s implementation for both interest enumeration and the lowest-level communications primitives ("dataflow nodes"). While this has enormous advantages, it does have the generic disadvantage of any general purpose "plug and play" system in terms of significant, incompletely understood, overheads.

Standard RTI-s dataflow implementations exist for both TCP and UDP communications. The results presented here use the TCP implementation.

The application processes for the benchmark tests are of two forms:

Publish Processors: Send out messages of specified length and interest state. The nominal total publication rate (Mbyte/sec) is controlled by a data file that is re-read periodically (by all publish processors). This means that the nominal experimental data rate can be controlled dynamically.

Subscribe Processors: Receive messages for a specified interest state, collecting messages from multiple publishers, as appropriate. The subscribe processes are instrumented to measure actual incoming message rates and to detect missed messages.

The routers in Figure 5 direct individual messages from publishers to subscribers according to the interest declarations. The router processes are also instrumented to determine the fraction of (wall clock) time spend in communications management (versus simply waiting for input).

Two modes were tested. In the first mode, a single TCP connection was setup between a pair of Meshrouters at distant locations. The measured bandwidth was approximately 300 megabits per second. The second mode used eight mesh routers at each site, each with multiple clients and multiple TCP connections. Measured aggregate bandwidth was approximately 4.6 gigabits per second.

This test demonstrated that 50% of the capacity of the high speed wide area network can be effectively employed by a real world application.

PROGRAMMING HADOOP

For the T&E programmers, Hadoop should be an easy system to use. Installation was straightforward. The rapid changes in Hadoop releases made keeping up problematic; some releases broke existing code. In the middle of a test evolution, one might be well advised not to install every update, but "freeze" Hadoop for the duration of the test.

Shell scripts might be found useful to reduce the complexity of setup, change, and maintenance of the various Hadoop configurations across sites. ISI experience was that, once these were in place, changing configurations was an easy and straightforward operation.

Developing was convenient for Hadoop jobs. Running and debugging standalone Hadoop jobs in the Eclipse IDE allowed rapid turnaround on application bug fixes.

CONCLUSIONS

This paper supports the proposition that the implementation and use of the Scalable Data Grid and Hadoop show promise for the Test and Evaluation environments. It reported on experiments implementing the SDG and on using distributed data analysis/data mining implemented over the Apache Hadoop framework. ISI experienced that the SDG and Hadoop provided a scalable, but conceptually simple, distributed computation paradigm based on the standard map/reduce operations implemented over a highly parallel, distributed filesystem. ISI found it practical to develop map/reduce implementations of K-Means and Expectation-Maximization data mining algorithms that took advantage the Hadoop framework. The Hadoop filesystem dramatically reduced the disk scan time needed by these iterative data mining algorithms. ISI has successfully executed these algorithms across multiple Linux clusters over dedicated 10 Gigabit per second networks. The ISI team holds that the results of these experiments support the potential for the use of these tools in Test and Evaluation.

ACKNOWLEDGEMENTS

We would like to thank our colleagues at JFCOM, and at UIC, especially Professor Robert Grossman and the Messers Michael Sabala and Yuhong Gu. Further, this work would not have been possible without the assistance of Tom Lehman of ISI East. This material is based in part on research sponsored by the Air Force Research Laboratory under agreement number FA8750-05-2-0204. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory or the U.S. Government.

REFERENCES

Barrett, B. & Gottschalk, T., (2004). Advanced Message Routing for Scalable Distributed Simulations. In *Proceedings of the Interservice / Industry Training, Simulation and Education Conference*, Orlando, FL.

Brunett, S. & Gottschalk, T., (1998) A Large-scale Metacomputing Framework for the ModSAF Real-time Simulation . In *Parallel Computing*: V24:1873-1900.

Chu, C., & Kim, S., (2006). *Map-Reduce for Machine Learning on Multicore.* Retrieved June 15, 2009, from: http://www.cs.stanford.edu/people/ang/papers/nips06-mapreducemulticore.pdf

Dean, J., Ghemawat S. (2004) *MapReduce: Simplified Data Processing on Large Clusters. Operating System Design and Implementation.* Retrieved 25 June 2009 from: http://labs.google.com/papers/mapreduce.html

Gehrig, J.F., Holloway, G., & Schroeter, G. (2002), Reflections on Test and Evaluation: T&E Infrastructure, Reengineering Army T&E, and Building a Viable Test Range Complex, in *Program Manager*, July-August, 2002

Graebener, R., Rafuse, G., Miller, R. & Yao, K-T. (2003) The Road to Successful Joint Experimentation Starts at the Data Collection Trail. In *Proceedings of the Interservice/Industry Training, Simulation and Education Conference*, Orlando, Florida.

Graebener, R., Rafuse, G., Miller, R. & Yao, K.-T. (2004). The Road to Successful Joint Experimentation Starts at the Data Collection Trail - Part II. in the *Proceedings of the Interservice/Industry Training, Simulation, and Education Conference.* Orlando Florida

Hadoop Map-Reduce Tutorial (2007) Retrieved 25 June 2009 from: http://hadoop.apache.org/core/docs/r0.17.1/mapred_tutorial.pdf.

Hammond, J., Dey, M., Scrudder, R., & Merkin, F., (1998). Populating the HLA Object Model Data Dic-tionary—A Bottom-Up Approach. *Simulation Inter-operability Workshop*. 98S-SIW-075.

Helfinstine, B., Torpey, M., & Wagenbreth, G. (2003). Experimental Interest Management Architecture for DCEE. *Interservice/Industry Training, Simulation, and Education Conference*, Orlando, Florida

Kimbal, R., Reeves, L., Ross, M. & Thornwaite, W. (1998) *The Data Warehouse Lifecycle Toolkit.* Hoboken, New Jersey: Wiley.

Lucas, R., & Davis, D. (2003). Joint Experimentation in Scalable Parallel Processors. *Interservice / Indus-try Training, Simulation, and Education Conference.*

Macannuco, D., Dufault, B., & Ingraham, L. (1998). An Agile FOM Framework. *Simulation Interoperability Workshop*.

O'Malley, O. (2008) *TeraByte Sort on Apache Hadoop*. Retrieved 29 June 2009 at: http://www.hpl.hp.com/hosted/sortbenchmark/YahooHadoop.pdf.

Powered By, 2009. Retrieved 29 June 2009 at http://wiki.apache.org/hadoop/PoweredBy

Rathnam, T., & Paredis, C.J.J. (2004) Developing Federation Object Models Using Ontologies. *Proceedings of the 2004 Winter Simulation Conference.*

Tan, G., Hu Y., & Moradi, F. (2003). Automatic SOM Compatibility Check & FOM Development, *Distributed Simulation and Real-Time Applications.*

The Hadoop Distributed File System: Architecture and Design (2007) Retrieved 25 June 2009 from http://hadoop.apache.org/core/docs/r0.17.1/hdfs_design.pdf.

Wagenbreth, G., Ward, G. E., Yao, K-T., & Davis, D.M., (2010, Pending), Nondisruptive Data Logging: Tools for JFCOM Large-scale Simulations, in the *Proceedings of the Simulation Interoperability Workshop*, Orland Florida

Wilbert, D., Macannuco, D. & Civinskas, W. (1999). A Tool for Configuration FOM Agility. *Simulation Interoperability Workshop*. 99F-SIW-116.

Yahoo! Launches World's Largest Hadoop Production Application, 2008. Retrieved 29 June 2009 from: http://developer.yahoo.net/blogs/hadoop/2008/02/yahoo-worlds-largest-production-hadoop.html.

Yao, KT., & Wagenbreth, G. (2005)Simulation Data Grid: Joint Experimentation Data Management and Analysis. In *Proceedings of the Interservice/Industry Training, Simulation and Education Conference*, Orlando, FL.

Yunhong G., Grossman R. 2007. UDT: UDP-based Data Transfer for High-Speed Wide Area Networks, *Computer Networks* (Elsevier). Volume 51, Issue 7.