Simulation Data Grid: Joint Experimentation Data Management and Analysis

Ke-Thia Yao and Gene Wagenbreth

Information Sciences Institute
University of Southern California
Marina del Rey, California
kyao@isi.edu, genew@isi.edu

ABSTRACT

The need to present quantifiable results from simulations to support transformational findings is driving the creation of very large and geographically dispersed data collections. The Joint Experimentation Directorate (J9) of United States Joint Forces Command (USJFCOM) and the Joint Advanced Warfighting Project (JAWP) is conducting a series of Urban Resolve experiments to investigate concepts for applying future technologies to joint urban warfare. The recently concluded phase I of the experiment utilized and integrated multiple scalable parallel processors (SPP) sites distributed across the United States from supercomputing centers at Maui and at Wright-Patterson to J9 at Norfolk, Virginia. This computational power is required to model futuristic sensor technology and the complexity of urban environments. For phase I the simulation generated more than two terabytes of raw data at rate of over ten gigabytes per hour. The size and distributed nature of this type of data collection pose significant challenges in developing the corresponding data-intensive applications that manage and analyze them.

Building on lessons learned in developing data management tools for Urban Resolve, we present our next generation data management and analysis tool, called Simulation Data Grid (SDG). The design principles driving the design of SDG are 1) minimize network communication overhead (especially across SPPs) by storing data near the point of generation and only selectively propagating the data as needed, and 2) maximize the use of SPP computational resources and storage by distributing analyses across SPP sites to reduce, filter and aggregate. Our key implementation principle is to leverage existing open standards and infrastructure from Grid Computing. We show how our services interface and build on top of Open Grid Services Architecture standard and existing toolkits (Globus). SDG services include distributed data query/analysis, data cataloging, and data gathering/slicing/distribution. We envision SDG to be a general-purpose tool useful for a range of simulation domains.

ABOUT THE AUTHORS

Ke-Thia Yao is a research scientist in the Distributed Scalable Systems Division of the University of Southern California Information Sciences Institute. Currently, he is working on the Joint Experimentation on Scalable Parallel Processors (JESPP) project, which has the goal of supporting very large-scale distributed military simulation involving millions of entities. Within the JESPP project he is developing a suite of monitoring/logging/analysis tools to help users better understand the computational and behavioral properties of large-scale simulations. He received his B.S. degree in EECS from UC Berkeley, and his M.S. and Ph.D. degrees in Computer Science from Rutgers University. For his Ph.D. thesis he implemented a spatial and physical reasoning system that automatically generated grids for novel geometries for computational fluid dynamics simulators.

Gene Wagenbreth is a Systems Analyst for Parallel Processing at the Information Sciences Institute at the University of Southern California, doing research in the Computational Sciences Division. Prior positions have included Vice President and Chief Architect of Applied Parallel Research and Lead Programmer of Pacific Sierra Research, where he specialized in tools for distributed and shared memory parallelization of Fortran programs. He has also been active in benchmarking, optimization and porting of software for private industry and government labs. He has programmed on CRAY, SGI, Hitachi, Fujitsu, NEC, networked PCs, networked workstations, IBM SP2, as well as conventional machines. He received a BS in Math/Computer Science from the University of Illinois in 1971.

Simulation Data Grid: Joint Experimentation Data Management and Analysis

Ke-Thia Yao and Gene Wagenbreth

Information Sciences Institute
University of Southern California
Marina del Rey, California
kyao@isi.edu, genew@isi.edu

INTRODUCTION

The specific motivation to develop the data logging and retrieval system described in this paper is to support simulations by the JUSJFCOM using Joint Semi Autonomous Forces (JSAF) software. JSAF provides entity-level simulation of ground, air and naval forces. Simulation of civilian entities is performed by a separate simulator. Simulation of multiple sensor platforms is performed by a program called Simulation of the Locations and Attack of Mobile Enemy Missiles (SLAMEM). JSAF scales from a single Central Processing Unit (CPU) to hundreds of CPU's. Individual simulators run on a single CPU. The High-Level Architecture (HLA) publish/subscribe software architecture is used to communicate results between simulators. A software router network enables the system to scale to hundreds of processors.

In the last two years a requirement for a large increase in the number and the fidelity of simulated entities justified the upgrade of JSAF simulations from workstations on a local-area network, simulating a few hundred or thousand entities to a wide-area network including multiple Beowulf clusters and hundreds of processors simulating hundreds of thousands of entities.

An important part of the simulations is to log what happens for near real time and after action analysis. The broad range of analysis requires that nearly all data be logged. The mechanism used is to log data when it is published. The earliest implementation included in the simulation a software logger, which subscribed to, and received all data published anywhere in the simulation. The total size of the logged data was limited to two gigabytes (GB). This worked when the number of processors and the number of simulated entities was small.

In 2003, to support larger simulations on Beowulf clusters Information Sciences Institute (ISI) implemented a distributed logger. Data is logged locally on each processor running a simulator. Near real time data queries are supported by a simple tree system to broadcast a query and concatenate results. After action queries are supported by transferring compressed binary files to a single host and expanding into a single monolithic database.

In 2005, this implementation is no longer adequate. The current size of a database for a two-week exercise, omitting nonessential data, is over a terabyte. The time required to transfer data to a single site and insert it into a database is inconvenient. Maintenance of hardware and software to support multiple large databases (one per exercise) on a single system is difficult. The current system cannot support anticipated future increases in the size and fidelity of exercises and the amount of data to be logged.

This paper describes SDG, the data logging system designed and implemented to support large JSAF simulations. SDG utilizes the resources of the systems generating the data, distributed processors and storage. Logging resources thus scale as processors are added to support the simulation.

SIMULATION DATA GRID

Overview

Simulation Data Grid is a distributed data management application/middleware that helps people deal with very large, geographically dispersed data sets over heterogeneous environments.

SDG provides data handling capabilities that are essential to current and future simulation analysis needs of the USJFCOM. It is able to collect and store high volume/high rate data from geographically distributed data sources, to browse high-level summaries and overviews of the stored data, to query details of the stored data, and to discover what part of the data has changed.

These capabilities are applicable to multiple domains where large amounts of data are generated, such as distributed event-based simulation (e.g. JSAF), live instrumented exercises, and instrumented physics experiments.

The collected data and the analysis tools provided are of potential use to a variety of people working with the simulation. To the military analysts the logged data can be used to compute effectiveness measures, such as situation awareness. They can compare and contrast simulation ground truth against sensor observations. The simulation developers can use the same logged data for validation and verification. They can query the logs to check simulation events/patterns against expected behavior to find anomalous behavior. The logging can easily be adapted to log resource usage, such as CPU, memory and network usage. Infrastructure managers can use these data to discover faults and resource usage bottlenecks.

The initial performance goal of SDG is to be able to support JSAF simulations running with one million entities. Such high entity counts would generate, to within an order of magnitude, about 100 GB of data per hour. Over a typical two-week event, about eight terabytes of data would need to be collected. Large-scale JSAF simulations are typically distributed across multiple geographically dispersed sites. In the Urban Resolve experiments the simulation was distributed across two supercomputers and multiple workstations at different sites. The sites include Maui High Performance Computing Center (MHPCC), Aeronautical Systems Center (ASC), J9, Space and Naval Warfare Systems Command (SPAWAR), and Topographic Engineering Center (TEC).

Leveraging Grid Computing

SDG is intended to operate in a joint experimentation environment, where the computing software and hardware elements may be quickly assembled on an as needed basis. The constituent elements may change depending on need and on resource availability. Each Urban Resolve exercise in a literal sense is setting up a virtual computing organization to solve a significant problem. This virtual organization spans multiple administrative domains each with its own security policies, and each offering its unique combination of computing, networking and storage capabilities.

The goal of Grid Computing is to provide pervasive dependable access to distributed computing resources. The Grid Computing vision, if realized, promises access to computing as easily as people currently access the power grid through their wall sockets. The main focus of SDG is data collection and analysis. But, in order for SDG to work effectively in a joint environment it must also address many of the same issues that face Grid Computing.

Grid computing research focuses on developing an interoperable common infrastructure that provides dependable consistent access to distributed and decentralized computing resources. It addresses the problem of *coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations* (Foster et al., 2001).

To emphasize the focus on developing interoperable tools and interfaces that work across platforms and organizations, Foster proposes that grid computing coordinate resources that are not subject to centralized control using standard, open, general-purpose protocols and interfaces to deliver nontrivial qualities of service (Foster et al, 2002).

Below we describe the Globus Toolkit, an open source implementation of Grid Computing services (Foster et al., 2002). Through out the paper, we refer back to this description to point ways SDG can potentially leverage these services.

Globus components are classified into five types (Foster, 2005). *Common Runtime* components provide a set of libraries and tools to allow Globus services to be platform-independent. Many Globus services are based on w*eb services* as defined by the World Wide Web Consortium. These services use eXtensible Markup Language (XML) as the data interchange format, Simple Object Access Protocol (SOAP) for messaging and Web Services Description Language (WSDL) for service interface description. Some Globus services, like GridFTP that were developed earlier, do not follow the web services framework.

Security components provide services related to user authentication, authorization, secured communications, and credential management. Security is a very important aspect of distributed simulations, but it is not the focus of this paper.

Data Management components provide services related to distributed data management, which includes data transportation, data replication and data access.

Information Services provide registries to allow services to register themselves, to discover other services and to monitor the status of services.

Execution Management components provide the ability to initiate, monitor, manage, schedule, and/or coordinate remote computations. These components can interface with batch job schedulers typically found at supercomputing sites.

Data movement components include GridFTP and Reliable File Transfer Service (RFT). GridFTP provides secure, robust, fast, efficient, standards based data transfer protocol. Version 4 provides striped transfer mode, where multiple nodes work together to transfer their own portion of the file. RFT, built on top of GridFTP, is a web service that provides the ability to recover from client-side failure by storing the transfer state in databases. Also, it provides a job scheduler to manage multiple transfers.

Data replication components include Replica Location Service, which provides a distributed registry that maps *logical file names* to *physical file names*. Also, the Data Publishing and Replication service provides pull-based services that automatically create local file replicas based on user request.

The Data Access and Integration (DAI) data management component is a federated service that provides registries to discovery data sources, factories to represent data sources, and data services to access data source in different formats (relational databases, XML databases, flat files).

DATA REQUIREMENTS AND OPERATIONAL EXPERIENCES FROM URBAN RESOLVE

J9 and JAWP, staged several training and integration exercises in early 2004, followed by four experiments, each two weeks long, from June through October. Several sites participated in the events. TEC site at Fort Belvoir, Virginia, had 30+ workstations and Saber, a quad-CPU machine with four terabytes of disk space that we used for after event storage. The SPAWAR site at San Diego, California, had 20+ workstations. The J9 Distributed Continuous Experimentation Environment at Suffolk, Virginia, had 50+ workstations and a 16-node mini-cluster. The ASC Wright Patterson Air Force Base at Dayton, Ohio, had the Glenn cluster with 128 dual CPU nodes. The MHPCC site at Maui, Hawaii, had the Koa cluster with 128 dual CPU nodes.

The experiments typically ran five days a week, ten hours a day. Simulators might run all night, but with little activity and usually with logging disabled. Depending on availability and requirements, one or both of Glenn and Koa were used. Up to two hundred thousand clutter entities were simulated on the large clusters. (In this simulation, civilian entities are termed clutter, in that they serve to mask military entities.) Several thousand non-clutter entities were simulated on the other sites. A single node on the large clusters simulated 1000-2000 clutter entities.

Data logging was performed in two modes, near real time and after action. Real time data was inserted in an SQLite database. A node simulating 1000 clutter items would generate an SQLite database of approximately 50 MB in an hour. The databases were deleted and reinitialized when they grew to over a gigabyte. If 100 nodes of the cluster were used for clutter simulators, approximately 5 gigabytes per hour of data was generated. For after action use, compressed binary data was stored in an archive directory. Binary compressed data is approximately $1/7^{th}$ the size of the corresponding database. Each night, the archived data was transferred to Saber, and expanded and decoded into a single MySQL database.

Clutter data from the Glenn and Koa clusters was not entered into the Saber database, due to size limitations. Data from 100 nodes on Glenn for a ten-day event would have been close to a terabyte. Data from TEC, SPAWAR, J9 and J9 mini-cluster for non-clutter entities were entered into the MySQL database. Urban Resolve Phase I exercise generated about a terabyte of data in the MySQL database.

The nightly data transfer was about 15 gigabytes of compressed data. Network transfer rate to Saber was approximately ten megabits per second. Three or four hours was required to do the transfer. Decoding and indexing the data into the MySQL database took 12 hours if everything worked perfectly. Human error and other factors usually prevented a day's data from being entered into the database before the next day's event started. It was usually at least several days after an event before the complete after action database was ready on Saber.

The logging methodology used for the four exercises in 2004 was adequate. It was the first attempt at logging data from hundreds of processors distributed geographically around the country simulating thousands of non-clutter entities. SDG is intended to remove deficiencies in the 2004 methodology and upgrade what was essentially an experimental system into a production system. The design parameters for SDG specifically address the following list of deficiencies in the 2004 system:

1. Near real time and after action data logging are implemented differently. Near real time queries are restricted by the use of simple aggregators.
2. The use of a single database on Saber does not have the capacity to include clutter data from the Glenn and Koa clusters.
3. Data transfers, decoding and indexing are time consuming and error prone, delaying the availability of the database. A goal is to have the complete database kept up to data continuously.

4. Retrieval of data and database generation for multiple exercises is inconvenient.
5. Expansion to more compute nodes, more entities per compute node and more data per entity is impossible. Disk storage, compute power, and network bandwidth all impose serious limitations.
6. The system does not respond gracefully to hardware and network problems. Saber is a single point of failure that makes all data unavailable.
7. Complex queries that may be useful to analysts are slow or impossible.

Database queries used in Urban Resolve are generally summary in nature. They count how many events or entities (database rows) meet specified criteria. Complex join operations were rarely, or never, used. Were it not for this constraint on the queries, an efficient distributed design would be more difficult.

SDG MODELS

User's Conceptual Model

As described in the previous section, in the current system data collected from the simulation are distributed and replicated at multiple locations. There is one access mechanism to query the data during simulation runtime, and another when the simulation is over. From the user's perspective, these data access complexities are unnecessary.

SDG adds a data access middleware layer that hides these complexities and presents a simple coherent view of data to the user. From their desktops SDG users should be able to access and analyze the data without having to know[1]:

1. How to access the data and what is the network interconnection topology (Access transparency).
2. Where is the data located (Location transparency)
3. Whether the data source has moved (Migration transparency)
4. Whether the data is from a replicated source (replication transparency)
5. Whether data sources are shared (Concurrency transparency).

Users interact with SDG through one of SDG's top-level Managers. Users submit queries to a top-level Manager, and they receive query results from a top-level Manager.

SDG is capable of handling static data sets (no new data added), as well as dynamic data sets (new data continuously being added). For dynamic data sets, users can register static queries with top-level managers, and receive asynchronous query results.

SDG provides feedback to the users regarding the queries they submit by letting them know the resources required to execute the query and the resources currently available.

Fallacies of Distributed Data computing

In 1991, Peter Deutsch articulated Eight Fallacies of Distributed Computing (Deutsch, 2005):

1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. The network is secure
5. The topology doesn't change
6. There is one administrator
7. The transport cost is zero
8. The network is homogeneous

Distributed software systems developed under these assumptions tend to be brittle. They later have to be re-engineered to work around these assumptions.

Potential additional fallacies related to Distributed Data Computing:

9. Disk capacity is infinite
10. Disk latency is zero
11. Disk bandwidth is infinite
12. Data processing cost is zero

The initial design of the 2004 system was based on the assumption that there is sufficient local disk space to store the logged data. But, we found nodes on supercomputer clusters tend to have less local disk space than the average desktop computer. We had to first implement a near-realtime system, then an off-site after action post-processing system.

SDG manages these potential pitfalls through a unique division of labor. SDG hides some of the networking details from the user by explicitly managing the five transparencies listed above, but exposes other details (e.g., resource usage

[1]Tanenbaum & van Steen (2002) defines three additional transparency goals for distributed systems. Some of these goals are beyond the current scope of SDG, and some are not applicable.

and storage options) to allow users to examine and if needed override default behavior and manage those details themselves.

To address the Distributed Data Computing Fallacies, we intend to provide multiple data services with varying levels of capabilities to let the user select the appropriate services for the tasks at hand (Table 1). These storage options trade-off storage size, query speed and query preprocessing time. For the Urban Resolve exercises approximately 50% of the messages were not logged because these messages were internal simulation bookkeeping messages. For example, Clutter Intersection (ClutterInt) messages that determine which car should enter an intersection next usually are of no interest to the analysts.

Table 1 Range of storage options that trades-off storage size, query speed and preprocessing time.

Storage Options	*Storage Size*	*Query Speed*	*Query pre-processing*
Do not log	Zero	N/A	None
Compressed (raw)	Small	Very slow	Small
Text (decoded)	Medium	Slow	Small
Database	Medium	Medium	Medium
DB w/ indexing	Large	Fast	Large
Cube (D = # of dimensions)	Large, for high D	Fast	Large, for high D

Typically, a user may want to include a compressed storage option to keep an archive of the simulation data. Then, the user may wish to select another storage option, for all or a partial subset of the messages, for faster querying. If the user chose multiple storage options, he has the option of deleting/truncating data storage to recover disk space.

Designer's Conceptual Model

SDG Managers perform all of the data access/ query/management tasks. Conceptually, there are three types of Managers: top-level, data source and worker.

Top-level managers have published addresses. Users connect through the top-level managers. To minimize network traffic typically there is at least one top-level manager for each local area network. Top-level managers know how to connect to each other. Non-top-level managers know how to connect to at least one top-level manager (Figure 1).

Data Source managers store the actual data. Other applications insert data into Data Source Managers through defined Application Programming Interfaces (APIs).

Worker managers perform most of the work within the system. When given a data processing task, the top-level manager decomposes a task into sub-tasks. Depending on the nature of the task the top-level manager enlists one or more managers worker, data source or other top-level managers. It then assigns sub-tasks to these managers and data source managers. Finally, it defines a data flow topology linking together the sub-task executions.

The mapping of the tasks onto managers must take into account and take advantage of a heterogeneous computing environment. The networking infrastructure within a local cluster typically uses Gigabit Ethernet, or even faster proprietary Myrinet. The inter-cluster networking infrastructure is typically orders of magnitude slower. Computation must be moved closer to the data sources to avoid transportation penalties.

The storage hierarchy varies from cluster to cluster. For example, the original configuration of the Koa cluster at Maui Supercomputing Center does not include local hard drives. A ten terabyte Storage Area network (SAN) mount on Global File System(GFS) functions as the only secondary storage. The Glenn cluster at ASC has a total of ten terabyte storage mounted on local hard drives, and ten terabyte SAN storage mounted on GFS.

USE CASES

In this section we describe various use cases to capture the functional requirements for SDG. Also, we describe how to map these requirements to Grid Computing functionality provided by Globus. We divide the use cases into three categories: How SDG manages itself (system administrators), how SDG manages the data sets (data administrators), and how SDG is used to query/analyze data (data analysts).

System Administrators

System administrative functions are used to manage the SDG system itself. The functions needed include the ability to remotely manage and control startup and shutdown of managers, the ability to remotely monitor the health of managers, and the ability to map task decomposition hierarchies and data flow diagrams onto the managers.

These system administration functionalities match well with Globus' Execution Management components and Information Services' Monitoring and Discovery System (MDS). MDS's Index Service is able to register services, as well as maintain resource properties associated with the service. MDS's Trigger Service can be used to send alerts when certain conditions occur, such as when a local disk is nearing capacity. Execution Management provides ways to submit, cancel and manage remote job executions.

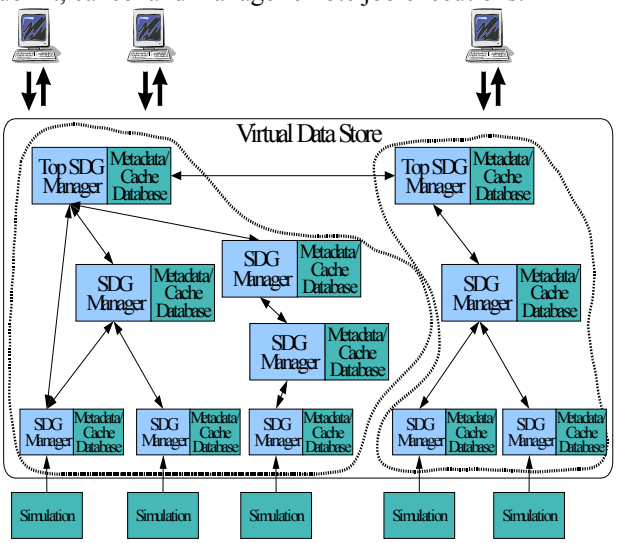

Figure 1 Designer's Conceptual Model. Dashed loops indicate the boundary of local area networks.

Data Administrators

Data administrative functions are used to manage the data collected and stored within the system.

Monitor Data/Resource Usage Statistics. Monitor the rate and size of data flowing into the SDG system, monitor the available disk capacity, monitor network usage, and monitor CPU usage.

Archive data sources. Copy/move data sources into one centralized location. This is useful for archiving data into a centralized SAN or a tape archival system.

Merge/split data sources. Combine multiple data sources into one source. Partition one data source into multiple data sources. These operations are useful to take advantage of parallelism when extra compute resources are available.

Here we again plan to leverage MDS. MDS is able to interface with cluster monitoring tools, such as Ganglia (Ganglia, 2005), to produce up-to-date system load/usage information. In addition, GridFTP, Reliable File Transfer service and Replica Location Service also play important roles.

Data Analysts

Select Queries. User sends a *select* Structured Query Language (SQL) query to a top-level manager. The top-level manager returns the query result in a result set table. The underlying data is stored in multiple locations, but to the user it appears to be one big centralized database.

We further classify select queries into aggregation queries, union queries and simple queries. Aggregation queries involve operators like sum, min, max, and average. Union queries access data from more than one table and/or results from sub-queries. Simple queries do not involve aggregation operators or unions.

Sample simple queries include: return all entity weapon damage reports within the last 30 minutes; and return red tank movements within the last 10 minutes.

Sample union queries include: return all entities that were painted by a sensor; and return marking information of the entity that fired a weapon within the last 10 minutes

Sample aggregate queries include: count the sensor tracks grouped by sensor type, or group by degree of assuredness; return killer/victim scoreboards; and return sensor/target scoreboards

For simple select queries SDG managers only need to concatenate results returned by sub-tasks without further processing. The previous implementation supports simple queries. The next section describes our current effort to extend to aggregation queries and multidimensional analysis.

Resource Usage Explanation Queries. This is similar to MySQL's EXPLAIN command. Given a select query, SDG traces through the execution of the query, and explains which resources and how much were used to answer the query.

Canned Queries. User defines periodic/trigger select queries. Based on the defined period, or the trigger, SDG executes the query and asynchronously returns the result to the user.

Sample uses include: receive alerts when a missile is launched; and automatically update killer-victim scoreboard when a weapon is fired.

The Globus' Data Access and Integration (DAI) service provides a common web services interface for accessing heterogeneous data sources (files, relational, XML). The relational part of the interface allows clients to submit SQL queries to Data services and to receive query result sets from the Data services. Also, DAI supports asynchronous delivery, which may be useful for periodic canned queries. In addition, Distributed Query Processing (DQP) service layers distributed join capabilities on top of DAI.

However, one key reservation we have about using DAI/DQP is the overhead of using SOAP/XML based communication for query result processing. Using standards based communication make sense if the data sources are heterogeneous. But, in our case we are focused just on relational data. Furthermore, scalability to handle very large simulation data sets is one of our overriding concerns.

DISTRIBUTED MULTIDIMENSIONAL ANALYSES

In this section we focus on the implementation of select aggregate queries, such as the Sensor/Target Scoreboards.

Background: Sensor/Target Scoreboards

One of the key focus areas of Urban Resolve Phase I is to study the effectiveness of future Intelligence, Surveillance and Reconnaissance (ISR) sensors in helping soldiers operate in complex urban environments. The Sensor/Target (S/T) Scoreboard provides a visual way of quickly comparing the relative effectiveness of individual sensor platforms and sensor modes against different types of targets. S/T Scoreboard is a specific instance of the more general multidimensional analysis.

In the Urban Resolve federation, a simulated sensor entity lays down sensor footprints to delimit sensor coverage sweep. For each target entity within the footprint, a *contact report* is generated to hold the result of the sensor detection. The contact report includes information about the sensor entity, the platform the sensor entity is mounted on, the sensor mode, the target entity, the detection status, the perceived target type, the perceived target location, the perceived target velocity and so on.

Sensor/Target scoreboards have the capability of providing summary views by aggregating individual sensor platforms into sensor platform types, such as *high altitude*, *medium altitude*, and *low altitude*. And, it aggregates individual target entity objects into target classes, which can range from the generic (like *Civilian Large Trucks*) to the specific (like Russian *MAZ-543 MEL*). As described by Graebener (Graebener, 2003), the current implementation of the scoreboard provides four levels of details. The information provided are:

1. Table of contact report counts broken down by sensor platform types and by target classes.
2. Given a sensor platform type and a target class, table of number of contact report counts broken down by sensor platforms and by sensor mode.
3. Given a sensor platform and a sensor mode, list of target objects.
4. Given a target object, list detailed target object attributes.

Initially, the S/T Scoreboard displays the level one aggregate table of sensor platform types and target classes. By clicking on a table cell (i.e., specifying a particular platform type and target class), the S/T scoreboard brings up the level two display of sensor platforms and sensor modes. Sensor modes are methods detection, such as Moving Target Indicator (MTI) and Synthetic Aperture radar (SAR) Spot and SAR strip.

Analysis of S/T Scoreboard from Multidimensional Perspective

The current implementation of S/T Scoreboards projects the contact reports along three dimensions for analysis. The three dimensions are sensor platform, target object and sensor mode. In addition, sensor platforms are aggregated into sensor platform types, and target objects are aggregated into target classes. Figure 3 depicts these three dimensions as linear partial orderings. These dimensions can be crossed to create lattices, as shown in Figure 2.

With respect to the right lattice in Figure 2, the information contained in level one S/T scoreboard correspond to the node *tc*, sensor platform type by target class. The level two information corresponds to slices of node *pcm*, where *p* is restricted to a particular sensor type, *t,* and entity class, *c*. Levels three and four correspond to target objects specified by cells in node *pom*.

The four levels of the S/T scoreboard present information useful to the analysts. But, other nodes within the lattices may be of potential interests. For example, node *cm* summarizes the effectiveness of sensor modes with respect to target class. Or, node *pc* summarizes the effectiveness of sensor platforms with respect to target class. In addition, other dimensions not used in S/T Scoreboards may be of potential interest, for example detection status, time,

location, terrain classification (high-rises, low-rises, flat), weather conditions, and so on.

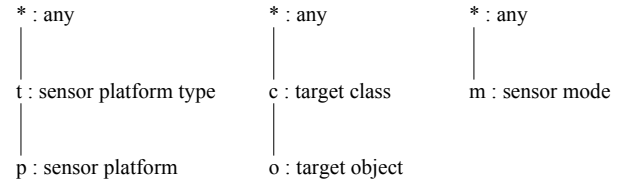

Figure 2 Three possible dimensions to partition the data for analysis.

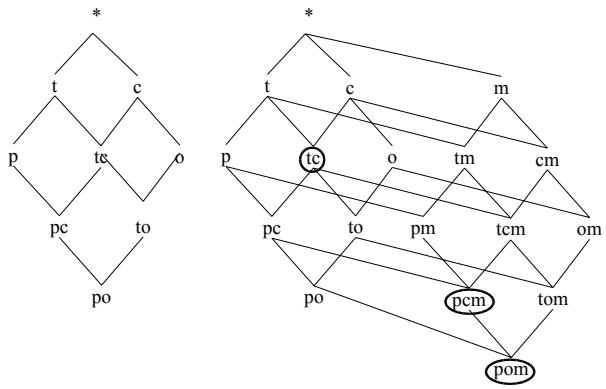

Figure 3 Lattices are generated by crossing the dimensions. Crossing the sensor dimension with the target dimension generates the lattice on the left. Crossing the left lattice with the sensor mode dimension generates the lattice on the right.

Multidimensional Analysis

S/T Scoreboard falls into a analysis class called multidimensional analysis, or sometimes called On-Line Analytical Processing (OLAP) or data warehousing (Kimbal *et. al.*, 1998). Other types of scoreboards, like Killer/Victim and Truth/Perception, are also multidimensional in nature. Conceptually, the data structure used to store multidimensional analysis data is the *cube*. The two-dimensional array data structure used to store a two-dimensional scoreboard is extended to higher dimensions.

Users could query the OLAP system for the entire cube, but typically the users are more interested in projections and partial views of the cube. Operations on the cube include roll-up, drill-down and slice & dice. Roll-up aggregates data along a dimension to hide details. This corresponds to walking up the dimension lattice. Drill-down partitions data along a dimension to reveal more details. This corresponds to walking down the dimension lattice. Slice & dice selects subsets of the cube elements.

Query and Data Characteristics

Query and data characteristics within the simulation differ from traditional OLAP assumptions in two significant ways: 1) a query is concurrent with insertions, and 2) the data is distributed.

Typically, OLAP is performed on historical data. For example, retail chains may keep sales transaction records to determine their best performing stores, or emerging consumer trends. This analysis is usually performed off-line. The analysis need not be updated as individual sales transactions occur.

In addition, data is typically sent to a centralized facility to be analyzed. In our case, it is not feasible to centralize the data because of the amount of data and near-realtime nature of the query. Our data is logged locally at the point of generation. If there are 100 simulation nodes, then we have 100 local logs.

Previous works have studied distributed OLAP implementations (Goil and Choudhary, 2001; Beynon *et. al.*, 2002). Typically, they employ some type of data partitioning scheme to perform load balancing and/or to reduce input/output (I/O) overhead. For example, in the *chunking* data partitioning scheme the data cube is partitioned into smaller sub-cubes. The number of dimensions of the sub-cubes remains the same, but now each dimension holds just a subset of the possible dimension values.

In our case these data partitioning schemes are not applicable. The simulation setup and placement dictate our data partitioning scheme. Moving these messages creates network traffic that may disrupt the actual simulation. Since we are not able to preposition data, we are investigating cube compression techniques to minimize storage and the I/O needed to aggregate the local cubes. These techniques include partial cube materialization (Harinarayan, 1996) that selectively pre-computes a subset of lattice nodes, coalesced cubes (Sismanis and Roussopoulos, 2004; Sismanis et al., 2002), and shell fragments (Li et al., 2004) that offer compact ways of storing the cube.

IMPLEMENTATION STATUS AND EXPERIMENTAL RESULTS

Implementation of a Simple Distributed Sensor Target Scoreboard

OLAP systems on single processors are widely used and described in the literature. Two implementations are frequently used. Multidimensional On-Line Analytical Processing (MOLAP) stores multidimensional data in an explicit multidimensional structure. Relational On-Line Analytical Processing (ROLAP) stores multidimensional data in a relational database. MOLAP provides faster access to data. ROLAP stores sparse data more efficiently. We chose ROLAP for the implementation of SDG for two reasons. First, we want the ability to scale to very large data sets with potentially high number of dimensions. ROLAP implementations tend to provide better scaling with respect to storage. Second, the current logger is implemented on top of relational databases. We want to maintain backward compatibility to allow the analysts to use SQL to directly query underlying logged data.

We develop the system and implement features in an incremental fashion in order to deliver capabilities to J9 in a reasonable fashion. This has the added benefit of providing feedback, which can be applied to future development. We identified the sensor target scoreboard, discussed earlier, as a critical feature representative of many key features that would ultimately be required, and that could be implemented quickly and efficiently.

We chose one week of archived data from one Urban Resolve event as test data. The sensor target scoreboard is prepared from a table in the database named I_ContactReport. We are interested in deriving a unique value for the type of sensor, the type of target and the detection status for each row of the table. This information is used to create a three-dimensional table of counts for each unique combination. Other information is discarded at this time. Future enhancements will incorporate information such as time and location to create a five-dimensional table (or larger).

The I_ContactReport table from our test case has approximately 18 million records. One column, node, identifies the machine on which the row was generated. To simulate the distributed generation of the data we created four new databases based on applying a regular expression to the value in the node column. Only 16 columns were copied to the four new databases. Two additional columns were added to identify unique combinations of the target and of the sensor.

Next, a procedure was applied to each of the four new databases. In a real exercise, this procedure would be applied independently and concurrently on each computer maintaining a database.

The procedure consists of the following steps using MySQL commands:

1. Create a table of unique combinations of sensor values and unique combinations of target values. Assign an enumerated type to each.

2. Create a row in the table for each combination of sensor type, target type and detection status that occurs in the I_ContactReport table. Compute a column count giving the number of times the combination occurs. With appropriate indexing this takes six minutes for six million records in one of the four sub-databases.

3. Add rows to the table for "wildcards" as appropriate for a data cube. A row is created for any sensor, any target, any detection status, three wildcards. This should equal the number of rows in the contact table for a three-dimensional table. Do the same for all permissible use of two wildcards and one wildcard.

This procedure is applied when a new dataset is introduced to the system. It is then applied to any new data that is added to the system. The data cube is always up to date.

There are now four relatively small databases containing complete and nearly instantaneously accessible information on the count of any combination of sensor types, target types and detection types. A user query to a top-level data manager is relayed to low level data managers connected to each of the four subdatabases. The responses are merged by combining responses with the same dimension value and summing the count field. The result is returned to the user.

CONCLUSION

The use of large clusters (hundreds of nodes or more) of processors to meet the demand for high performance computing is becoming a mature technology. The extension to use multiple clusters is likewise common, but less mature. The use of OLAP technology to analyze large data sets also is becoming a mature technology. To support USJFCOM and JAWP we have combined distributed clusters and OLAP. Using this approach, and innovation in key areas, we are able to support our customer's current and expanding needs. A key principle is to store data close to its source to minimize network traffic. A second principle is to utilize the computational and storage resources of distributed clusters for database functions. These two principles reinforce, rather than interfere with, each other in the design and implementation of the data grid. A key area

of innovation is the intelligent manager. The intelligent manager categorizes a query, creates an execution plan, distributes the work for the query, aggregates the results, and delivers the results. The queries required and commonly used by JSAF analysts are efficiently executed by this system. Fault tolerance and realistic data archiving are additional benefits of our implementation. We will maintain and extend the system to include more processors, more clusters, larger datasets and more robust queries as required by our customer.

ACKNOWLEDGEMENTS

The authors wish to acknowledge the members of the ISI JESPP team who have contributed to this paper through their efforts and their intellectual stimulation. Much of the success reported here came from the JESPP project, initiated, directed and funded by the USJFCOM and to a very large degree conducted on the compute assets of the Maui High performance Computing Center, ASC-MSRC at Wright Patterson Air Force Base and other members of the High Performance Computing Modernization Program.

This material is based on research sponsored by the Air Force Research Laboratory under agreement numbers F30602-02-C-0213 and FA8750-05-2-0204. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes, notwithstanding any copyright notation thereon.

REFERENCES

Beynon, M., Chang, C., Catalyurek, U., Kurc, T., Sussman, A., Andrade, H., Ferreira, R., & Saltz, J. (2002). Processing large-scale multi-dimensional data in parallel and distributed environments, *Parallel Computing.*

Deutsch, P. (n.d.). The Eight Fallacies of Distributed Computing. Retrieved Aug 05, 2005, from http://today.java.net/jag/Fallacies.html

Foster, I., Kesselman. C., & Tuecke, S. (2001). The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications, Vol. 15, No. 3*, p200-222.

Foster, I. (2002). What is the Grid? A Three Point Checklist. *Grid Today*, vol. 1, no 6. http://www.gridtoday.com/02/0722/100136.html

Foster, I., Kesselman, C., Nick, J., & Tuecke, S. (2002). The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. *Open Grid Service Infrastructure WG, Global Grid Forum*, June 22, 2002.

Foster, I. (2005). A Globus Toolkit Primer. http://www.globus.org/toolkit/docs/4.0/key/GT4_Primer_0.6.pdf

Ganglia (2005). http://ganglia.sourceforge.net/

Graebener, R.J., Rafuse, G., Miller, R., & Yao, K.T. (2003). Successful Joint Experimentation Starts at the Data Collection Trail. I/ITSEC 2003.

Goil, S., & Choudhary, A. (2001). PARSIMONY: An Infrastructure for Parallel Multidimensional Analysis and Data Mining, *Journal of Parallel and Distributed Computing*, v61.

Harinaryan, V., Rajaraman, A., & Ullman J.D. (1996). Implementing Data Cubes Efficiently. Proc. ACM SIGMOD '96, p 205-216.

Li, X., Han, J., & Gonzalez, H. (2004). High-Dimensional OLAP: A Minimal Cubing Approach. *The Proceedings of the 30th International Conference on Very Large Database.*

Kimball, R., Reeves, L., Ross, M., & Thornthwaite, W. (1998). The Data Warehouse Lifecycle Toolkit. New York: John Wiley & Sons, Inc.

Sismanis, Y., Deligiannakis, A., Roussopoulos, N., & Kotidis, Y. (2002). Dwarf: shrinking the PetaCube. *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data.*

Sismanis, Y., & Roussopoulos, N. (2004). The Complexity of Fully Materialized Coalesced Cubes. *The Proceedings of the 30th International Conference on Very Large Database*.

Tenenbaum, A. S., & van Steen, M. (2002). *Distributed Systems principles and paradigms*, New Jersey: Prentice Hall