A Self Manageable Infrastructure for Supporting Web-based Simulations

Yingping Huang Xiaorong Xiang Gregory Madey
Department of Computer Science & Engineering
University of Notre Dame
Notre Dame, IN 46556
{yhuang3, xxiang1, gmadey}@nd.edu

Abstract

Imagine if you would like to deploy your new simulation online and your systems can just take care of itself. Needed web interface could be generated, database schema objects could be created, simulation programs could be installed and configured on your execution servers, data analysis and reports could be generated automatically. In this paper, we describe the design and implementation of a self-manageable multi-tier infrastructure to support scientific simulations. This infrastructure demonstrates not only the successful integration of web servers, execution servers, database servers, reports servers, data warehousing and data mining, but also the ability to achieve self manageability, including self-configuring, self-healing, self-protecting and self-optimizing. A scientific simulation program, NOM, is employed to demonstrate the effectiveness of this infrastructure.

1. Introduction

Simulation is the process of designing a model of a real system and conducting experiments on this model for the purpose of understanding the behavior of the system or of evaluating various strategies for the operation of the system. The power of simulation is the ability to model the dynamics of a real system and to analyze the results. It is important to analyze the simulation data so that the output of the simulation is not misinterpreted. The complexity of simulation data often requires more sophisticated analysis other than statistical analysis, such as data warehousing and data mining.

Most simulation models currently available in the fields of scientific simulations run in stand-alone or traditional client-server architecture. Both of these models require installing software on user's computers. This presents a significant barrier due to incompatibility that complicates or prevents installations. Meanwhile, the stand alone or traditional client-server approach has some significant drawbacks: (1) lack of scalability since it's a single user program, (2)lack of collaboration features, such as information sharing among users, (3) lack of reliability since usually no fault-tolerant feature is built in simulation, and (4) lack of centralized simulation management and data analysis.

To overcome these drawbacks, many Web-based simulations have been developed recently using server-side technologies. But these Web-based simulation environments lack the ability of self management. Professional IT staff seems necessary to maintain the reliability and availability of the simulation environments. In this paper, we present a multi-tier infrastructure such that simulation developers can deploy their simulations online automatically and users can run simulations and view simulation reports just using their web browsers, such as Netscape and Internet Explorer. The users only need to specify inputs for their simulations, data analysis and reports are completed at the back end and delivered to the users through the reports server.

This infrastructure can serve multiple users and each user can run multiple simulations. Scalability and reliability can be achieved through the built-in self-management features. The self-management features include self-configuring, self-healing, self-protecting and self-optimizing. Besides, this infrastructure has other benefits. For example, centralized data repository makes data analysis, information sharing and collaboration easier. Further more, large volume of data often requires more sophisticated analysis besides statistics and thus brings the opportunity for data warehousing and data mining. This infrastructure can serve as a template or guideline for future web-based simulation model design. That's the research goal of this work.

The rest of this paper is organized as follows. In section 2, we list some related Web-based simulations and present our NOM project. We also briefly introduce the components of a self manageable infrastructure. In section 3, we demonstrate our infrastructure and show that our infrastructure can achieve high scalability and availability. In section 4, we present the self management features of our infrastructure. Finally, in section 5, we draw conclusions of this work and point out some directions of future work.

2. Background and Related Work

In this section, we briefly review the Web-based simulation development, then we introduce the NOM simulation project, finally we'll discuss the components of a self manageable infrastructure.

2.1. Web-based Simulation

Web-based simulation represents the combination of WWW technologies and simulation science. The ability of Web technologies enhances the power of simulation in that Web can service large simulation communities and allow developers to distribute models and simulations to end users. Along with the advancement of Web technologies, many Web-based simulations and simulation support systems have been developed.

Holmes et. al. [4] demonstrated an integration of standards based web services technologies, grid-enabling software and a component framework for parallel computing, resulting in a service-oriented architecture which provides end users the ability from their desktops to manage and understand simulation results for very large, complex problems. Fernandez et. al [3] proposed an interval caching strategy to enhance the performance of integrated storage systems. A web-based simulation was developed to compare their strategy and other known algorithms for the purpose of evaluation. Sarhan and Das [9] developed Web-based simulations to evaluate scheduling policies of multimedia servers.

Huang and Miller [6] presented a prototype implementation of a Web-based federated simulation system using Jini and XML. Holmes et. al [5] proposed a multi-tier architecture to allow web access to visualization tools running on MP systems. Belfore and Chitithoti [1] described multiuser extensions of the Virtual Reality Modeling Language (VRML) that integrate a database interface to create web based simulation environments.

Some of the above Web-based simulations made use of multi-tier architectures. But due to the lack of self manageability, it's often costly to successfully achieve high reliability and high performance of Web-based simulation environments. The major contribution of our work is to design and implement a self manageable infrastructure to support scientific simulations.

2.2. The NOM Project

Natural organic matter (NOM) is a mixture of molecular components with different types of structures, compositions, functional group concentrations, molecular weights, and different degrees of reactivity. NOM comes from animal and plant material in the natural environment. It exists everywhere in the world, from terrestrial ecosystems to aquatic environments. NOM plays a crucial role in ecological and bio-geochemical processes such as the evolution of soils, the transport of pollutants and the global biochemical and geochemical cycling of elements [2]. The evolution of NOM over time from precursor molecules to mineralization is an important research area in a wide range of disciplines, including biology, geochemistry, ecology, soil science, and water resources. NOM, micro-organisms and their environment form a complex system. The global phenomenon of a complex system can often be observed by simulating the dynamic behavior of individual components and their interactions in the system.

The NOM project is a multi-disciplinary project supported by the US National Science Foundation. Currently, two Web-based NOM simulators have been developed and deployed to meet requirements from different groups of users. Each of the NOM simulators used an agent-based stochastic simulation model to simulate the behavior of molecules. The purpose of the NOM simulation is to let scientists to conduct experiments on it to better understand the evolution of natural organic matter [10].

2.3. Self Manageable Infrastructure

A computing system often consists of storage devices, networks, databases and servers. These components have workflow dependencies and interact among themselves. Managing the system involves configuring the individual components so that the overall system goals can be achieved. Jim Gray in his Turing award speech ”What next? - A dozen IT research goals” emphasized the need for self-manageable systems in which the administrator sets system goals and creates high-level policies, while the system by itself decides how they can be achieved. IBM calls self manageable systems ”autonomic computing” [7].

A self manageable infrastructure for support Web-based simulations consists of the following components:

- Self-configuring: New features, new software and additional servers can be added to the system with little human interaction while the system is up and running. Given the input and output metadata of simulations, the system can create user forms and create database schema objects without human involvement.

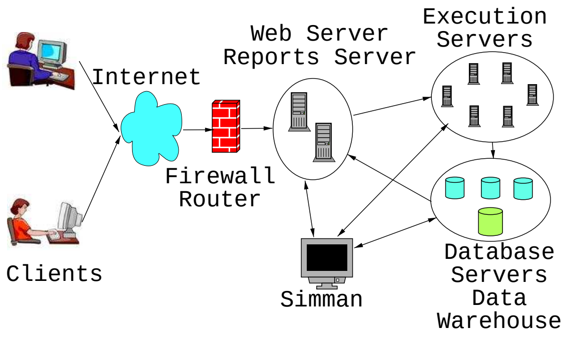


Figure 1. The Multi-tier Infrastructure

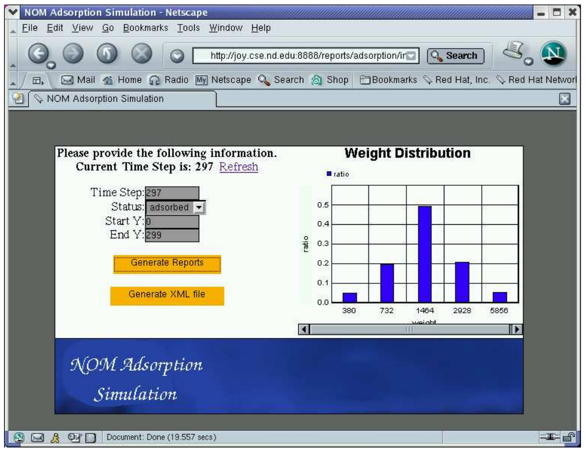
- Self-healing: The system detects malfunctioning components and brings them off-line for recovering and brings them back on-line once recovered. The system needs to be designed with redundancy and backup routines such that self-healing can occur transparently to users.
- Self-protecting: The system resource can only be accessed by authorized users to prevent data lose and misconfiguration.
- Self-optimizing: The system makes best use of resources by providing load-balancing features across Web servers, execution servers and database servers. Further more, self tuning features could be added in the database tier such that the database servers can tune themselves as needed.

In the next section, we first show the infrastructure for supporting scientific simulations.

3. Overview of the Infrastructure

The NOM project development team has designed an infrastructure built on a multi-tier architecture to support high scalability through load balancing and high availability through redundancy. The HTTP client tier communicates with the HTTP server tier through a firewall. The HTTP server tier routes the requests of simulations to the execution server tier, which connects to the database server tier. The reports servers connect to the database servers and delivers reports to HTTP clients. A simulation manager (Simman) generates necessary Web applications and deploys them to the Web server tier. And it generates SQL scripts to build database schema objects for a simulation. Further more, it talks to intelligent agents on the execution server tier for the purpose of self-management. Each of the tiers can be scaled individually by building clusters of servers for supporting load balancing and redundancy.

Figure 1 shows the infrastructure. Oracle9i Application Server and Reports Server are used in the Web server tier


NOM Adsorption Simulation - Netscape

http://joy.cse.nd.edu:8888/reports/adsorption/in

Please provide the following information.
Current Time Step is: 297 Refresh

Time Step:	297
Status:	adsorbed
Start Y:	0
End Y:	299

Document: Done (19.557 secs)

Figure 2. A sample report

and Oracle9i is used in the database server tier. Users logon to the system, specify inputs for the simulations through some HTML forms, and submit the simulations. The simulation inputs will be stored in the database and the simulation will be executed on one of the execution servers. Data generated by the simulations will be either stored in the databases through JDBC (Java DataBase Connectivity) or transfered to the data warehouse through SQL*Loader. Data analysis and reports will be generated at the backend automatically and delivered to the users through the reports server.

A reports server is configured inside the Web server tier. The major function of the reports server is to deliver simulation results to the users. Both graphical reports and XML reports are supported in our infrastructure. The graphical reports are generated by the simulation manager and delivered by the Oracle9i Reports Server. The XML reports are generated using XSQL (XML SQL) and transformed by XSLT to HTML file. Figure 2 shows a sample report page for the NOM project.

Currently, most scientific data is distributed across a multitude of databases and sites. Scientists have very limited support to access and manage their data. They are forced to manually relate between experimental data and analysis facilities without an infrastructure support. A data warehouse provides the opportunity to store experimental data and data produced by running simulations. The data will eventually be explored to search for patterns by data mining so that scientists can obtain a greater understanding of their research.

Therefore, in our design, a data warehouse is included in the database tier. The purpose of data warehousing includes:

- To store both experimental and simulation produced data.

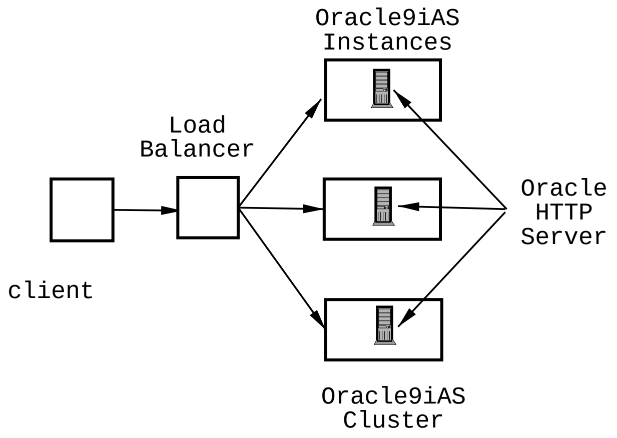


Figure 3. The Oracle9iAS Cluster

- To provide mechanisms for fast reports by creating star schemas.
- To function as a data mining server such that data mining can be applied across simulations.

3.1. Scalability

Our infrastructure is designed to be highly scalable at all tiers. Scalability of the Web server tier and database server tier is be implemented using Oracle technologies: Oracle9i Application Server clustering and Real Application Clusters [8].

- Web Server Tier: A application server cluster is a collection of application servers which can be configured to be a single group. New application servers can be added during the operation of the cluster without downtime. Client interacts with the cluster as if they are interacting with one single application server. Client requests are routed to instances of the cluster through a load balancer, as shown in Figure 3.
- Execution Server Tier: The execution server tier can be scaled linearly by installing new execution servers running identical simulations. The execution servers are included into our new load balancing algorithm which will be discussed in later sections.
- Database Server Tier: Oracle Real Application Cluster (RAC) can be used in this tier. In RAC environment, all active nodes can concurrently execute transactions against a shared database, as shown in Figure 4.

3.2. Availability

Availability is achieved through some level of redundancy.

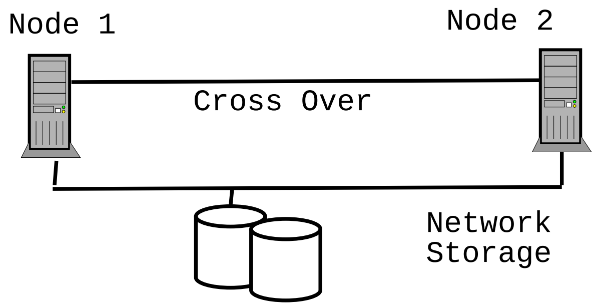


Figure 4. The Oracle9i Real Application Cluster

- Web Server Tier: An Oracle9i RAC eliminates the single point of failure by redundancy and failover in the system. A failure of any single instance does not bring down the whole system. Client session state is replicated throughout the cluster, thereby protecting against the loss of session state in case of process failure.
- Execution Server Tier: Availability of the execution server tier is handled by our new simulation checkpoint and resuming features, which will be discussed in later sections.
- Database Server Tier: Each instance node is isolated from each other so that a failure on one node does not affect the whole database system.

From the above, we see that our infrastructure can achieve high scalability by load balancing and high availability by redundancy. In the next section, we will focus on the self manageability of our infrastructure.

4. Self Managemement of our Infrastructure

A self manageable infrastructure has four components: self-configuring, self-healing, self-protecting and self-optimizing. To achieve these capabilities, a program called Simulation Manager (Simman) plays an important role, as shown in Figure 1.

4.1. Self-configuring

When a new simulation is developed, it can be deployed to the Web automatically. This includes configuring the Web server tier, execution server tier and the database server tier. To automatically deploy Web-based simulations, we need to make use of the simulation metadata. Simulation metadata describes the simulation, including input metadata, output metadata, and other necessary metadata for the purpose of simulation checkpointing, which will be discussed in the

```
<?xml version="1.0"?>
<simulation name="NOM">
<input>
<env>
<name>microbe density</name>
<name>fungal density</name>
<name>ph value</name>
<name>temperature</name>
<name>oxygen density</name>
<name>light density</name>
<name>molecule density</name>
<name>adsorption rate</name>
<name>desorption rate</name>
</env>
</input>
<output>
<name>molecule position</name>
</output>
</simulation>
```

Figure 5. The Simulation Metadata

subsection Self-healing. Simulation metadata is of the format in XML (eXtensible Markup Language). An example of simulation metadata of the NOM project is shown in Figure 5.

4.1.1. Self-configuring Web Server Tier Java Server Pages (JSP) can be generated automatically from the simulation metadata by the simulation manager (Simman). JDBC code can be embedded into the JSP files. After these JSP files are generated, they can be assembled into EAR (Enterprise ARchive) files and deployed to the Web server automatically by the simulation manager using the following commands which are embedded into shell scripts:

```
simman$ java -jar admin.jar
ormi://tobit.cse.nd.edu:23791
admin manager -deploy
-deploymentName nom -file nom.ear
simman$ java -jar admin.jar
ormi://tobit.cse.nd.edu:23791
admin manager -bindWebApp
nom nom http-web-site /nom
```

4.1.2. Self-configuring Execution Server Tier Simulation programs and dependent libraries need to be installed on the execution servers, which can be distributed by the simulation manager to the execution servers. In the NOM project, both the simulation manager and the execution servers are clients of a NFS server. Once the simulation program is installed on the simulation manager, it can be executed on all of the execution servers. Meanwhile, in order to load the data generated by the simulations to the database server tier, a shell script is automatically generated which can be triggered to run after each simulation completes. This shell script calls the SQL*Loader utility to load data into the database (once the database schema objects have been created). The following lines of code shows a snapshot of the shell script:

```
#!/bin/sh

sqlldr username/password@db_name \
```

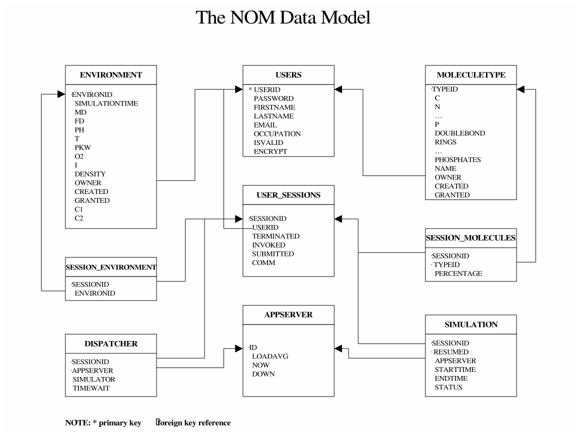


Figure 6. The NOM data model

```
control=nom.ctl
```

A new execution server is detected by the simulation manager through an intelligent agent running on the new execution server. The intelligent agent reports the existence of the new execution server by inserting a new record of the execution server into the database, and the simulation manager checks the database and add the execution server into the cluster of execution servers. We'll discuss more about the intelligent agent in later sections.

4.1.3. Self-configuring Database Server Tier SQL scripts are generated by the simulation manager automatically from the simulation metadata to create database schema objects used by both the simulation and the Web application. Figure 6 shows the data model generated by the simulation manager for the NOM project. Figure 7 shows a star schema for the NOM project generated by the simulation manager for the data warehouse.

4.2. Self-healing

Some level of redundancy is required to enable self-healing. We use Oracle9i Application Server Cluster in the Web server tier and Oracle9i Real Application Clusters in the database tier. Both of them can achieve high availability as discussed before. In the database tier, multiple active nodes concurrently execute transactions against a shared database. It is possible that the database may experience media failure, therefore, routine backup of the database is mandatory. We designed our own backup scripts to backup the database every week, which is invoked by UNIX cron. In the case of database failure, database administrator must be involved to replace the bad media and recover the database, and bring it back online.

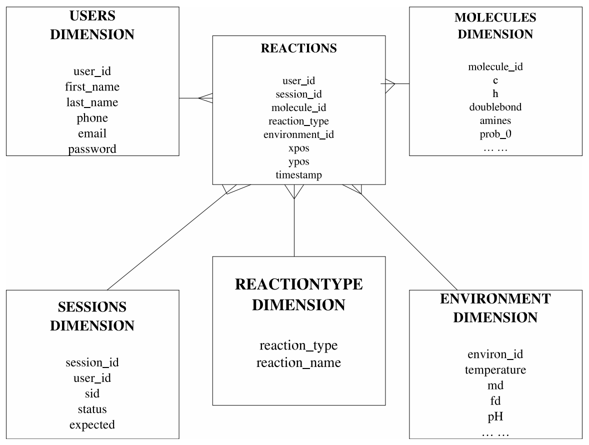


Figure 7. A star schema in the data warehouse

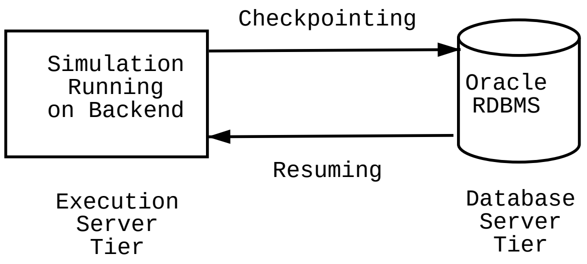


Figure 8. Simulation checkpointing and resuming

Simulations are running on the execution servers which are transparent to the users. What if a simulation dies prematurely? Some mechanism must be provided against simulation failure. In the NOM project, we designed and implemented the simulation checkpointing and simulation resuming features. Simulation checkpointing is the solution for long-running simulations. Figure 8 shows the processes of checkpointing and resuming.

The complete state of a simulation needs to be saved into an RDBMS, such that when a failure occurs, the simulation state can be restored and the simulation can be resumed. It is better to store simulation state into an RDBMS than to store it on the load disk. If the checkpointing data is on a local host, it is costly to transfer it to another node when a simulation is to be migrated. Simulation checkpointing can be useful for the purpose of resuming crashed simulations. It can also be used for the purpose of load balancing.

An intelligent agent is running on each execution server. An intelligent agent monitors the status of all simulations running on its host. It also reports the status of the simulations to the simulation manager. When a running simulation dies prematurely, the intelligent agent detects this event and reports it to the simulation manager. The simulation manager will dispatch the crashed simulation to an appropriate execution server. The assigned execution server loads the checkpointed simulation state from the RDBMS and resumes the simulation.

The simulation manager keeps sending a KEEPALIVE message to each intelligent agent and each intelligent agent sends back an ACKnowledgement message to the simulation manager. If the ACK message times out, then the simulation manager marks the corresponding execution server as DOWN. In this case, all simulations currently running on this execution server will be marked as CRASHED by the simulation manager and they will be dispatched by the simulation manager to appropriate execution servers.

The following lines of code shows the implementation of an intelligent agent. It is a shell script and running as a daemon on the execution server. This shell script calls other shell scripts which have database access code to check simulation status and reports to simulation manager.

```
#!/bin/sh

while [ true ]
do
  /export/daemon/loadavg.sh $1
  /export/daemon/checkjob.sh $1
done
```

For example, the shell script loadavg.sh is implemented as follows.

```
#!/bin/sh

if [ $# -ne 1 ]; then
  echo "usage: loadavg <appserver>"
  exit
fi

# upload loadavg every 5 seconds
LOADAVG=‘cat /proc/loadavg \
|(read u v w x y;echo $u)‘
# echo Loadavg:$LOADAVG
sqlplus username/password@sm \
>/dev/null <<EOF
update appserver set loadavg=$LOADAVG,
NOW=sysdate, down=’N’ where id=$1;
commit;
exit;
EOF
```

4.3. Self-protecting

The system resource cannot be accessed by unauthorized users. For example, a malicious user may submit many simulations simultaneously and thus consume a lot of resources and prevent normal user from running simulations. This is a sort of DoS (Deny of Service) attack. Therefore, it is critical to protect the infrastructure by taking the following actions.

- As shown in Figure 1, the whole system is behind a firewall and only certain ports are open to the public.
- The Web server is configured to support Secure Socket Layer (SSL).
- Only authorized users can access the simulation web pages and to invoke simulations.
- Network monitoring software is installed for monitoring network traffic around the infrastructure.
- The simulation manager pings all the tiers to ensure that they are working properly.

4.4. Self-optimizing

The Web server tier and database server tier have been configured to be scalable. Oracle9i Application Server Cluster and Oracle9i Real Application Cluster make use of the round-robin load balancing mechanism to ensure all the nodes are load balanced. We can certainly use the round-robin policy on the execution server tier. But it turns out round-robin is not the best load balancing algorithm to be applied to the execution server tier when the execution server tier consists of heterogeneous computers. In the next few paragraphs, we present our load balancing algorithm in the execution server tier and self-tuning features in the database server tier.

4.4.1. Load Balancing the Execution Server Tier The goal of load balancing is to distribute simulations evenly among the execution servers. There have been many load balancing schemes in the literature. Simulations need to be migrated from congested nodes to lightly loaded nodes such that they can be completed faster. These load balancing schemes can be divided into two groups: sender initiated and receiver initiated. In sender initiated schemes, congested nodes attempt to migrate simulations to lightly loaded nodes. In receiver initiated schemes, lightly loaded nodes attempt to find congested nodes from which simulations may be migrated. In this paper, we propose a load balancing algorithm based on simulation checkpointing. We call it checkpointing-initiated. The checkpointing-initiated scheme is similar to the sender-initiated scheme, but it is

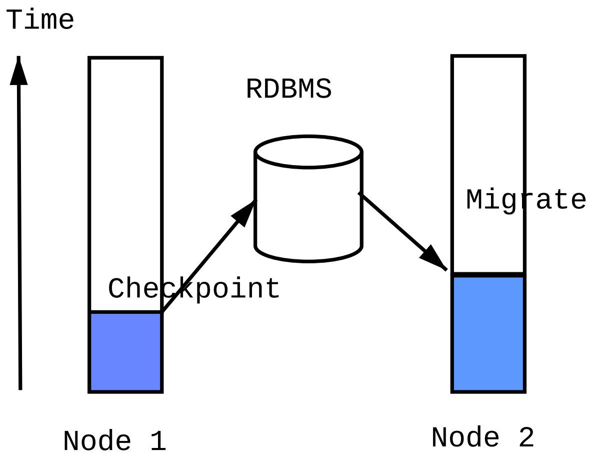


Figure 9. Checkpointing-initiated load balancing

controled by the centralized simulation manager. The simulation manager decides the node to which the simulation should be migrated. Figure 9 shows the checkpointing-initiated load balancing algorithm.

When a checkpointing takes place, checkpointing data is inserted into the Oracle RDBMS. The simulation manager tries to find the most suitable node such that it can migrate the simulation on that node and resume the simulation. In our current system, the node with the smallest load-avg is chosen as the best node for migrating a simulation to it. Load average can be used to measure the load of a node, it can be obtained by running the ”uptime” command. The intelligent agent on the execution server tier reports the load average of the local host to the simulation manager periodically. When a new simulation is submitted or an existing simulation is to be migrated, the simulation manager always choose the node with the smallest load average as the target to run the simulation.

In practice, the checkpointing intiated load balancing policy outperforms the round robin policy.

4.4.2. Self Tuning the Database Server Tier The dynamic memory features of Oracle9i makes it possible to create a self-tuning database. We have design and implemented scripts which are invoked by an Oracle package DBMS_JOB periodically. These scripts can adjust the Oracle System Global Area to the most appropriate configuration based on the usage trends of the database. The usage trends of the database can be obtained by collecting historic system data from the database using the DBMS_STAT package.

From the simulation metadata, various reports pages will be generated by the simulation manager. These reports pages connects to the RDBMS through JDBC. The query statements inside JDBC need to be tuned such that they can execute fast. Scripts have been designed and can be invoked by the UNIX cron to identify the most resource consuming SQL statements. Once these SQL statements are retrieved, the scripts can tune them by creating necessary indexes or even rewrite the SQL statements.

In the data warehouse, star schemas are composed of fact tables and dimension tables. The primary key of a fact table is the combination of the foreign keys which reference the primary keys of the dimension table. Bitmap indexes are created automatically for these foreign keys such that a query on the fact table can be accomplished fast.

5. Conclusions and Future Work

Computer systems have been getting more and more complex. Self manageable systems can leviate the cost of professional IT staff. In this paper, we described the design and implementation of a multi-tier infrastructure for supporting Web-based simulations. The infrastructure demonstrated not only a successful integration of web servers, execution servers, database servers, data warehousing and data mining, but also the ability to achieve self-manageability.

In the future, more automonic features and advanced load balancing algorithms will be developed for our infrastructure. To name a few:

- A Web-based simulation deployment tool will be developed.
- A Web-based simulation manager will be developed.
- New load balancing algorithms based on load average, virtual memory statistics and I/O statistics.

6. Acknowledgements

This research was partially supported by a NSF ITR Grant No. 0112820 and by the Center for Environmental Science & Technology at the University of Notre Dame. We acknowledge the contributions of Patricia Maurice and Leilani Arthurs of the Department of Civil Engineering & Geological Science at the University of Notre Dame for their respective discussions on the Web interface, input, and testing of the infrastructure.

References

[1] L. Belfore and S. Chitithoti. Multiuser extensions to the interactive land use vrml application (iluva). In *Proceedings 34th Annual Simulation Symposium*, pages 151–166, 2001.

[2] S. Cabaniss, G. Madey, P. Maurice, L. Leff, Y. Huang, and X. Xiang. Stochastic synthesis model for the evolution of natural organic matter. In *225th American Chemical Society National Meeting*, 2003.

[3] J. Femandez, J. Carretero, F. Garcia, J. Perez, and A. Calderon. Enhancing multimedia caching algorithm performance through new interval defi nition strategies. In *Proceedings 36th Annual Simulation Symposium*, pages 175–182, 2003.

[4] V. Holmes, W. Johnson, and D. Miller. Integrating web service and grid enabling technologies to provide desktop access to high-performance cluster-based components for large-scale data services. In *Proceedings 36th Annual Simulation Symposium*, pages 167–174, 2003.

[5] V. Holmes, J. Linebarger, D. Miller, R. Vandewart, and C. Crowley. Evolving the web-based distributed si/pdo architecture for high-performance visualization. In *Proceedings 34th Annual Simulation Symposium*, pages 151–158, 2001.

[6] X. Huang and J. Miller. Building a web-based federated simulation system with jini and xml. In *Proceedings 34th Annual Simulation Symposium*, pages 143–150, 2001.

[7] IBM. *Autonomic Computing*. http://www.ibm.com/autonomic, 2003.

[8] Oracle. *Oracle9i*. http://www.oracle.com, 2003.

[9] N. Sarhan and C. Das. A simulation-based analysis of scheduling policies for multimedia servers. In *Proceedings 36th Annual Simulation Symposium*, pages 183–190, 2003.

[10] X. Xiang, Y. Huang, and G. Madey. A web-based collaboratory for supporting environmental science research. In *WI/IAT 2003 Workshop on Applications, Products and Services of Web-based Support Systems*, pages 29–36, 2003.