INFRASTRUCTURE, DATA CLEANSING AND MINING FOR SUPPORT OF SCIENTIFIC SIMULATIONS

A Proposal

Submitted to the Graduate School

of the University of Notre Dame

in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

by

Yingping Huang, M.S.

Kevin W. Bowyer, Director

Department of Computer Science and Engineering

Notre Dame, Indiana

July 2003

INFRASTRUCTURE, DATA CLEANSING AND MINING FOR SUPPORT OF SCIENTIFIC SIMULATIONS

Abstract

by

Yingping Huang

We propose a multi-tier infrastructure which demostrates the successful integration of web servers, application servers, databases, data analysis and reports, data cleansing, data warehousing, data mining, and the Swarm/RePast simulation models.

The goal of the system is to support scientific simulations in the fields of environmental and social science using advanced features available in information technology. We'll design server-side simulation models that employ the advanced J2EE and XML technologies through which users can invoke simulations on the application servers, and obtain simulation reports through reports server.

Technologies such as JMS (Java Messaging Service), JTS (Java Transaction Service), EJB (Enterprise Java Beans), and AQ (Oracle's Advanced Queuing) will be applied with new algorithms to implement features such as load-balancing and simulation-resuming. Meanwhile, some previously developed collaboration utilities such as BBS (Bulliten Board System), chatroom, file-uploading, and simulation-sharing will be integrated to allow users to collaborate with each other.

We'll also explore the full life-cycle of data mining, which includes data cleansing, warehousing and mining. Data generated by the simulations, inputted by users, collected from the Web and through experiments will be transformed, cleansed and

loaded into a well-designed data warehouse. With the help of data mining, some interesting knowledge and patterns can be discovered. The models of data mining can further be deployed to ”tune” the simulation models.

CONTENTS

CHAPTER 1

INTRODUCTION

In this proposal, we explore how the information technologies, such as J2EE, RDBMS, XML, data cleansing, warehousing and mining can be used as a research tool to support scientific applications, such as those in the fields of environmental and social sciences. We propose a multi-tier infrastructure which integrates application servers, database servers, data analysis and reports, data cleansing, warehousing, mining, and the agent-based stochastic simulations models implemented using Swarm or RePast. Swarm is a set of Java APIs implementing the agent-based technology [61]. RePast [53] is similar to Swarm. See also [31] for a short introduction of agent-based technology.

The goal of this integrated multi-tier infrastructure is to employ advanced information technologies and other algorithmic applications to improve the scalability and reliability of the agent-based stochastic simulations and to facilitate analysis of the resulting large datasets.

1.1 Research Context

Most of the proposed work is based on the related projects. Currently, there are two multi-disciplinary projects under way, NOM (natural organic matter) and OSS (open source software development). Both projects are simulating natural phenomenon using agent-based stochastic simulation models. Both of them can take advantage of the infrastructure for web-based simulation invocation and data analysis. Large amounts of data is produced by the simulation programs. The large volume of simulation data, experimental data and simulation configuration data enables us to apply the data cleansing, warehousing and mining technology, eventually to explore the potential patterns implied in the data.

NOM is a simulation project that simulates natural organic matter in the soil and ground water. The scientific objective is to develop simulation programs to predict the properties of natural organic matter over time as it evolves from precursor molecules to eventual mineralization. The methodology used is a stochastic simulation of transformations, including biological and non-biological reactions, adsorption, aggregation and physical transport. NOM is a joint project with environmental scientists and chemists.

OSS is a simulation project that simulates the behavior of open source software developers. Its goal is to understand the topology and evolution of the social network formed by developers. OSS is a joint work with social scientists.

1.2 Proposal

We propose a multi-tier infrastructure which supports web-based server side simulation models. Although web-based simulations have been developed for years, there is a lack of details to integrate middleware technologies and data analysis tools to facilitate scientific simulations. Compared to multi-tier infrastructures built in the e-business realm, our infrastructure has specific utilization of the application servers. To efficiently utilize the resources, we propose new load-balancing algorithms to support scalability. We also propose a simulation-resuming feature of the infrastructure using JTA/JTS (Java Transaction APIs/Service) to support reliability.

As far as we are aware, there is little application of data cleansing, warehousing and mining to the fields of environmental and social science. The massive amounts of data collected from the Web, produced by simulation experiments enable us to invent new procedures to cleanse the data and build data warehouse and data mining models.

To summarize, this proposal has the following goals:

- A multi-tier infrastructure with new load-balancing and simulation-resuming algorithms to support scientific simulations.
- New data cleansing techniques and algorithms to cleanse scientific data.
- A knowledge discovery attempt in the fields of social science and environmental science.
- An integrated collaboration suite for support of scientists.

This proposed work will benefit the scientific researchers by:

1. providing a web-based server side simulation model - As stated before, most simulation models currently available in the field of environmental science run in stand-alone or traditional client-server architecture. Both of these models require installing software on the user's computers. This presents a significant barrier due to incompatibility that complicates or prevents installation. This proposed work will present all the details to build a multi-tier enterprise system accessible from everywhere and can serve as a template for future simulation designs.
2. integrating a web-based collaboration utility suite - BBS system, chatrooms, file-uploading, and simulation sharing can shrink the distance between scientists and enable them to collaborate closely.

3. providing data cleansing, warehousing and mining technologies to further analyze scientific data - Data mining can impact the research of these fields by discovering patterns in the scientific datasets which are otherwise difficult to discover. These patterns can be used to assist experiment design, make reasonable assumptions, and predict behavior of agents.

1.3 Organization of Proposal

The rest of the proposal is organized as the following: in chapter 2, a detailed multi-tier infrastructure with new load-balancing algorithms and simulation-resuming features is presented; in chapter 3, new data cleansing algorithms and comparisons with other known algorithms are proposed; in chapter 4, general introduction of data warehousing and data mining is provided; in chapter 5 and chapter 6, we show how data cleansing, data warehousing and data mining can be applied to social science and environmental science, with the examples of OSS and NOM respectively; and finally, a timeframe to implement the proposed work is given in chapter 7.

CHAPTER 2

INFRASTRUCTURE

We have discussed infrastructures for web-based simulations in [31]. Many improvements will be presented in this chapter, including scalability, reliability, maintainability and efficiency.

We propose to build a small LAN (local area network) on a multi-tier architecture to support high scalability through load-balancing and high availability through simulation-resuming. The users on the HTTP client tier communicates with the HTTP server tier which routes the incoming requests to the application servers. The application servers host scientific simulations and connect to the database servers to store simulation data.

The rest of this chapter is organized as follows. Section 1 gives a short introduction of information technologies used for the infrastructure. Section 2 lists some existent web-based simulation models and their drawbacks in convenience, scalability, reliability and maintainability. Section 3 describe the multi-tier information system and point out its major features which make it better than other approaches. Section 4 demonstrates how to implement load-balancing using JMS, EJB, and Oracle Advanced Queuing, with two different algorithms. Section 5 illustrates implementing simulation-resuming through JTS/JTA. Section 6 shows some collaboration tools which enable scientists around the world to share knowledge. Section 7 shows some snapshots of simulation reports, either graphical or in XML.

2.1 Introduction to Information Technology

The introduction of information technology is having impact on both scientific studies and people's daily lives. Understanding information technology has become a requirement for personal and professional growth and success. Components of information technology include J2EE, XML, data cleansing, warehousing, mining and others. Since our infrastructure is built on these information technologies, a short introduction to them is given below.

2.1.1 J2EE

According to Sun Microsystems, the Java 2 Enterprise Edition (J2EE) utilizes the unified, scalable and platform independent Java language and uses a component-based approach to design, develop, assemble and deploy enterprise applications [36]. Many textbooks exist that demonstrate the J2EE technology and its applications. My personal favorite is *Professional Java Server Programming J2EE, 1.3 Edition* by S. Allamaraju, et. al. [54].

J2EE consists of separate technology areas for building successful systems utilizing these technologies. A quick review of some of the available technologies that are applied in this proposal is given below.

- **JDBC** The Java DataBase Connection (JDBC) technology lets Java programs access almost any tabular or relational data source. JDBC will be used everywhere in our system, including the core simulation programs which need database access to store simulation data, and web interfaces which provide simulation configurations and reports.

- **JNDI** The Java Naming and Directory Interface (JNDI) APIs provide interfaces to directory services such as JDBC data sources, EJB homes, JMS connection factories and destinations.

- **JTA and JTS** The Java Transaction Service and Java Transaction API ensure data integrity by enforcing strict rules for accessing and manipulating data. JTA enables a distributed transactional system or application to access a transaction manager, as defined by JTS. JTA and JTS will be used to implement simulation-resuming, which simply means that when a simulation is crashed or terminated for any reason, it can be resumed. The reason for JTA and JTS is that we need to span updates to multiple databases.

- **Servlet** The Java Servlet API is a container managed, stateful alternative to Common Gateway Interface (CGI) applications. A servlet is used to receive HTTP requests, generate dynamic data, and deliver HTTP response.

- **JSP** The Java Server Page lets developers embed java code into HTML pages. JSPs are compiled into Java Servlets at runtime. We use servlets and JSP to design and implement the interfaces of the web applications.

- **EJB** An enterprise java bean is a server-side component that encapsulates the business logic of an application. Remote clients (including standalone clients and web clients, for example, the core simulation programs) can access the EJBs. Enterprise java beans can be distributed across multiple machines, therefore, applications can be scalable using EJB. There are three types of EJBs: session EJBs, entity EJBs, and message-driven EJBs. Our web application will use all kinds of EJBs to implement different aspects of the application.

- **JMS** The Java Messaging Service supports asynchronous messaging systems that allow decoupled systems to process information. There are two JMS models: publisher/subscriber and point-to-point (PTP). We'll use both models and Oracle's Advanced Queueing (AQ) to implement the load balancing feature of our infrastructure.

- **JavaMail** The JavaMail API enables sending and reading emails in a java program. Emails are sent to users when they submit simulation jobs to the simulation system.

- **JAXP** The Java API for XML Processing (JAXP) supports implementing independent processing of XML documents through Document Object Model (DOM), Simple API for XML (SAX), and eXtensible Style Language for Transformation (XSLT). JAXP will be used in the Oracle environment to process XML datagrams using the XML SQL Utility (XSU).

2.1.2 XML

XML, which stands for the "Extensible Markup Language", defines a universal standard for data exchange. It provides a set of rules that enable the structure inherent in data to be easily encoded and interpreted by human-readable text format [66].

XML provides a lot of convenience for us to simplify web applications. It also simplifies simulation configuration, interpretation of simulation results and data mining results. In this proposal, we will focus on the Oracle XML SQL technology, which is a combination of XML, SQL, XSLT and Java [49]. XML documents are generated whenever there is a need of them so that users can take the XML documents for further processing.

2.1.3 RDBMS (Oracle)

We use Oracle as the database tier of the multi-tier infrastructure. Oracle is one of the popular databases in industry [52]. It will be used to store data from web interface, simulation programs, experiments, etc.

2.1.4 Data Cleansing and Data Warehousing

According to Inmon [34], a data warehouse is a database with the following features: subject oriented (defined by subject matter), integrated (data from disparate sources), nonvolatile (rarely updated), and time variant (grow over time). Dodge and Gorman elaborate in more detail on Oracle data warehousing [17]. To build a good data warehouse, some other techniques such as data cleansing will be needed as a data preparation step. Data cleansing is also called "Merge/Purge" in the business world. Data coming from heterogeneous resources must be combined into a single database. Some errors or redundancy, including approximate duplicates which represent the same entity, must exist in the merged database. Data cleansing refers to removing incorrect or (approximately) duplicate data from a database such that the resulting database is in a consistent format.

2.1.5 Data Mining

Data mining refers to extracting or mining knowledge from large amounts of data [25]. It is an intelligent process in order to find patterns in data stored in databases, data warehouses or other information repositories. Data mining functionalities include association rules mining, prediction and classification, clustering, outlier analysis, evolution analysis and so on [11] [6].

2.2 Related work

The emergence of the Web has produced an environment within which web-based simulation models are being proposed and developed by both researchers and practitioners. To mention a few web-based simulators, Long et. al developed a web-based tool SimTracker to address problems in the field of physics [35]. Campos et. al proposed a web-based simulation of autonomous software agents using Java [10].

There are many more web-based simulation models and tools. Most of them are running on the client side through Java Applets. Some of the disadvantages of applets are

- security: it's hard to access applets behind firewalls.
- inconvenience: to run applets, plug-ins are necessary.
- lack of report tools: report softwares such as Oracle Reports are hard to incorporate with the simulation models..
- incompatibility: some software, such as the Swarm library that we are using to model simulations cannot be embedded into a java applet, due to implementation specifics.
- network traffic: the whole simulation program must be downloaded to the client before executing. Users will be irritated if the download process takes too much time.

To overcome the disadvantages of the applets-based web simulations, we propose to design web-based simulation models using the server side approach, which means simulations will run on the server side instead of on the client side.

Some server side simulation models have been developed by other researchers using CGI. But there is a lack of details to implement a scalable, reliable and easily-maintainable infrastructure. One of our contributions is to build such a infrastructure to support scientific simulations.

We design web interfaces to let users configure simulations and invoke the simulations through their web browser, and get dynamic graphical and XML-based reports of the simulations through the Reports Server, which extracts data from the backend databases.

2.3 The Multi-tier Infrastructure

Figure 2.1 shows the physical layout of a private network designed for scientific simulations. Machine 10.10.0.1 is the firewall and router, meanwhile, it functions as the HTTP server and reports server. It has two network interfaces, one of which has a public network address. Machines 10.10.0.2 through 10.10.0.5 are database servers. One of them is a data warehouse and data mining server which hosts ad hoc queries, data analysis and data mining. Machines 10.10.0.6 through 10.10.0.10 are application servers on which simulations are running. More application servers can be added without much effort.

Figure 2.2 shows the multi-tier infrastructure. The user provides inputs for the simulations through the web interface and then submits his/her simulation. The HTTP server routes the request to an appropriate application server through a load-balancer. As will be described in the next section, JMS, EJB and Oracle AQ are employed to implement load-balancing with a new data mining based approach. The goal of the load balancer is to distribute the simulation to one of the application servers such that it can be completed as soon as possible.

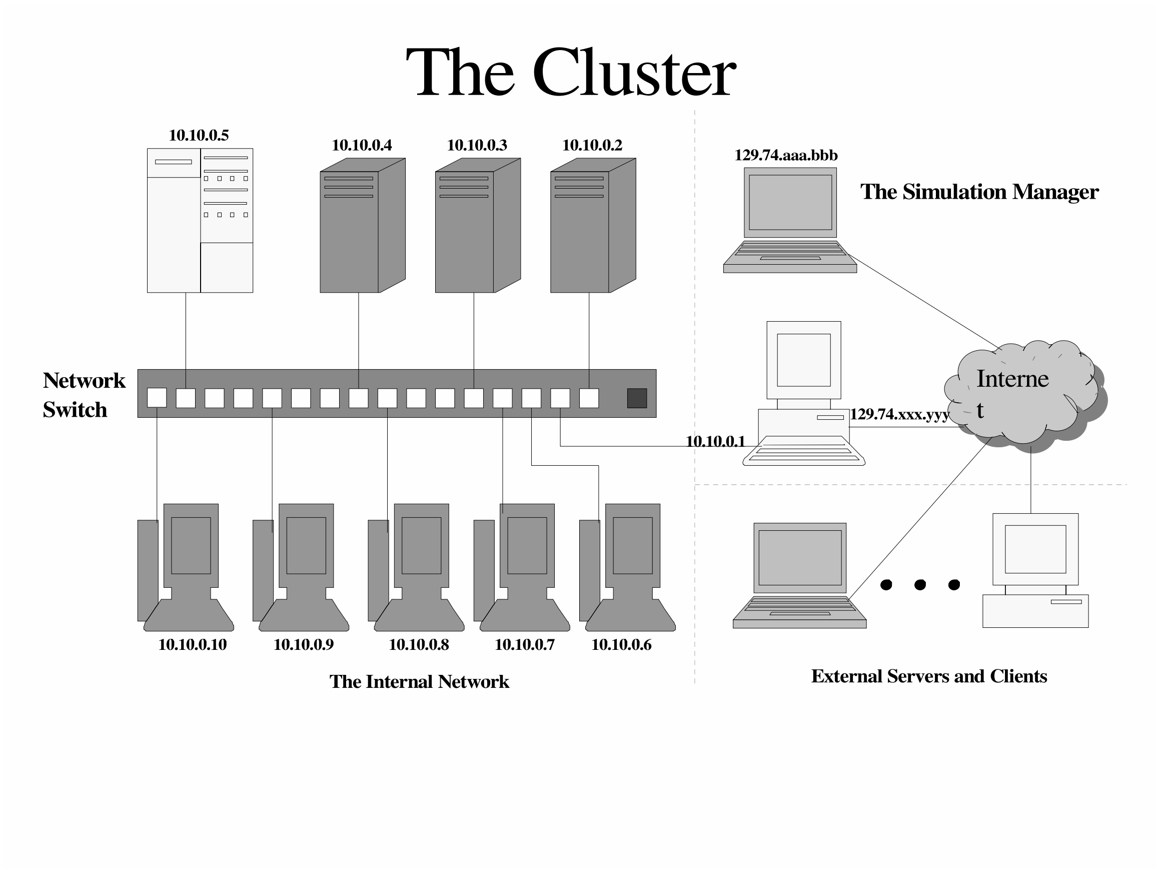


Figure 2.1. The cluster for scientific simulations

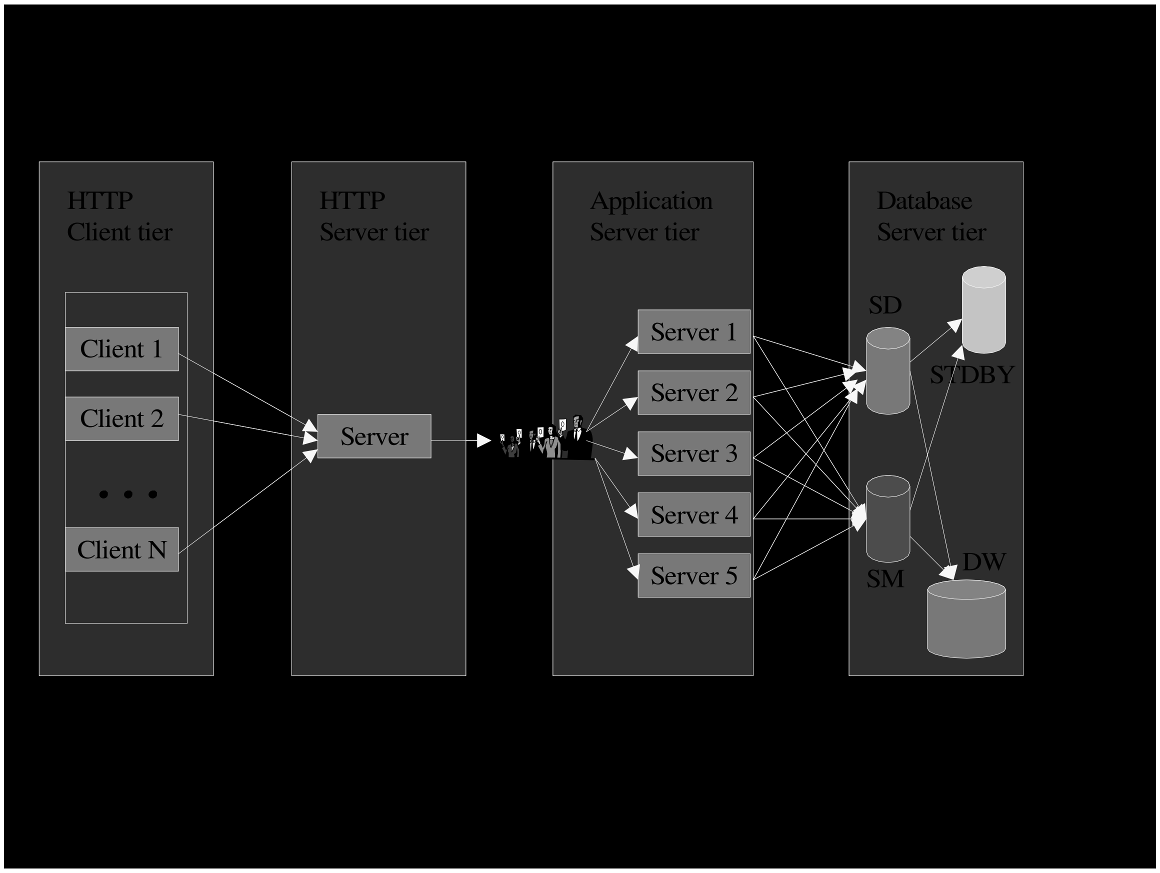


Figure 2.2. The multi-tier architecture

Data generated by the simulation is stored in an Oracle database SD (stands for simulation data). To enable the simulation-resuming feature for high availability and reliability in the case of software or hardware failure, and to resume a terminated simulation, we design a mechanism to checkpoint the simulation periodically. The states of the simulations are stored in another Oracle database SM (stands for simulation management), as shown in Figure 2.2.

Data in SD and SM is archived to corresponding standby database servers (STDBY). The standby database servers will, with little human interaction, take the role of the primary database servers once the primary databases are down. To further analyze scientific data, the data in SD and data collected from the Web or experiments is cleansed, transformed and transported to a data warehouse DW. A data mining server is running on the data warehouse server to mine the scientific data. Dynamic graphical or XML-based simulation statistics can be generated by the reports server and make available to users as JSP files or XSQL files.

The application server tier can be scaled linearly by installing additional application servers running identical simulations. The scalability can be achieved through load-balancing. In the next section, we'll show how load-balancing can be implemented. We provide two algorithms to implement load-balancing. One algorithm chooses an application server with smallest load average. The other predicts simulation completion time using a data mining approach. Based on the prediction, the resulting application server will be chosen such that the simulation can be completed in shortest time potentially.

2.4 Implementation of Load-balancing Using JMS, EJB and AQ

The goal of load-balancing is to distribute work evenly among a set of application servers. There are many load-balancing schemes available in the literature, but none of them can guarantee an even distribution if they are applied directly to the situation in which all simulation jobs have long execution time and are I/O bounded.

Available load-balancing algorithms include the following, to name a few:

- Round Robin: new jobs are assigned to the application servers sequentially. For example, suppose we have 6 jobs to be assigned, then job 1 is assigned to application server 1, job 2 is assigned to application server 2, and so on. Job 6 is assigned to application server 1 again. Round Robin is the simplest load-balancing algorithm. It does not take into account the execution time of jobs.

- Shortest Queue (SQ) [64]: a new job is assigned to an application server with the smallest number of jobs running. It does not count for job complexity.
- Shortest Expected Delay (SED) [5]: A job is assigned to the application server which has the least expected time to complete the new job. In [5], the expected completion time is simply derived from the number of jobs currently waiting in the queue. In our situation, we need a different mechanism to predict the expected completion time of a new simulation.

We propose two load-balancing algorithms. One is simply to assign the simulation to an application server which has the smallest load average. Another approach is to predict the completion time of a simulation job based on simulation configuration, machine load, and history of simulation jobs. We use a data mining approach. This algorithm tries to load-balance the application servers by assigning simulation jobs to an application server which is predicted to complete the job in shortest time. Note that this algorithm is different from SED. In SED, the completion time prediction is simply derived from number of waiting jobs.

2.4.1 Implementing the Simple Load-balancing Algorithm

We use JMS, EJB and AQ to implement load-balancing based on the load averages of the application servers. The machine load average, measuring how may processes are waiting to be executed, can be obtained by running the "uptime" command on the local machine. The load average is updated every 5 seconds. A simulation is assigned to one of the application servers which has the lowest load average.

If two or more simulations are assigned in a short time interval, they might be assigned to the same application server since the load average is not updated yet in the short time interval. To prevent this situation, we want to assign simulations at most once in every 10 seconds to ensure that the load average is updated after one simulation is assigned. Next, we'll show how to implement load-balancing.

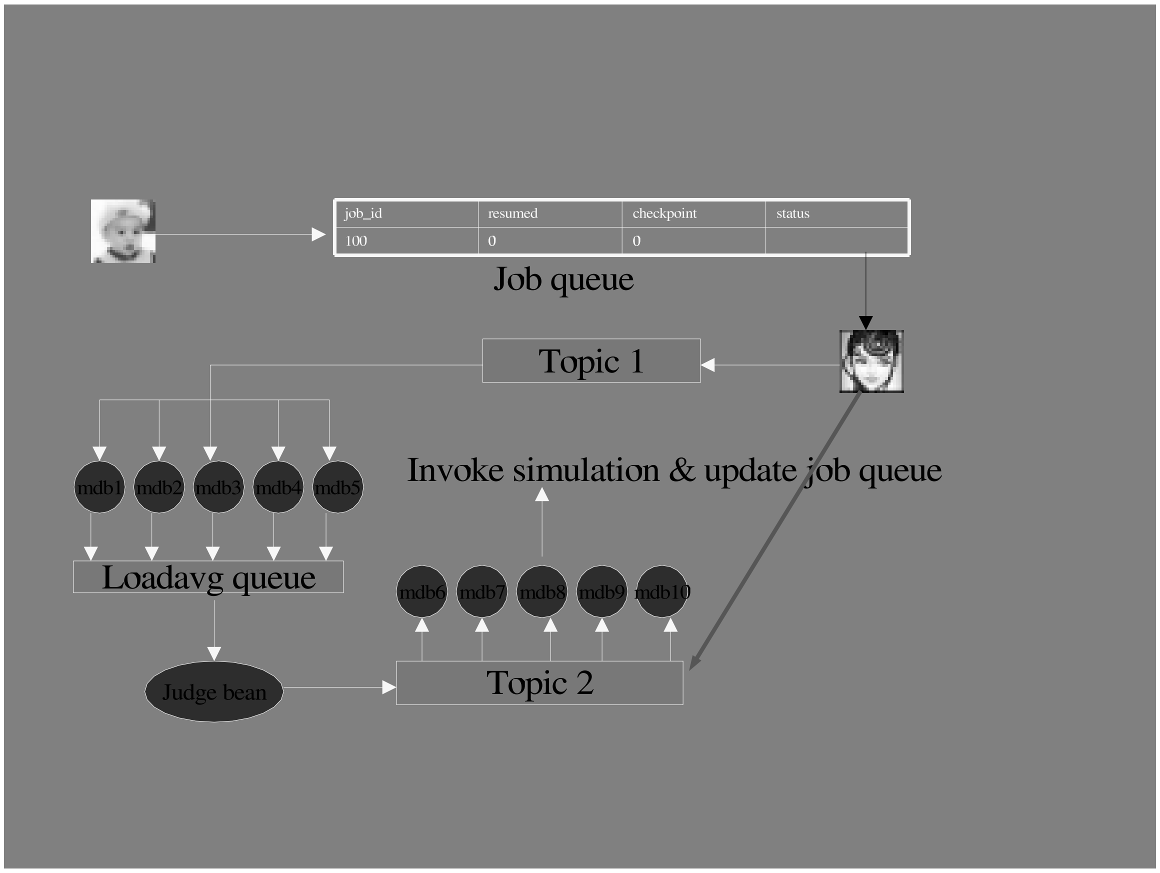


Figure 2.3. The load balancer

Figure 2.3 shows the implementation of the load balancer. After users provide all the inputs for a simulation through the web interface, the inputs will be stored in a database table (Actually, the information about one simulation will span multiple tables for the purpose of normalization and thus performance). Two attributes, the JOB_ID and RESUMED (the times the simulation resumed), are combined as the primary key. A resumed simulation has the same JOB_ID. This table will act as the job queue and it will be queried by a daemon process, we call it the job dispatcher, running on the HTTP server. The job dispatcher queries the table periodically (for example, every 10 seconds) to see whether new jobs are waiting to be executed.

If there are waiting jobs in the job queue, the job with the smallest JOB_ID is picked as the candidate to be executed. A message about the new job, including its JOB_ID, is published to Topic1 by the job dispatcher. Topic1 is stored in an Oracle database.

Message driven beans (MDBs) mdb1 through mdb5 are subscribers of Topic1. Each MDB is deployed on a corresponding application server, for example, mdb1 is deployed on application server 1. After receiving the message in Topic1, they check the load averages of the local hosts, i.e., the application servers, by running the "uptime" command.

Mdb1 through mdb5 put the load averages to a loadavg queue (resides in the same database as Topic1) and a judge bean on the HTTP server will receive all the load averages and the application server with the lowest load average will be the candidate to execute the simulation job. If the lowest load average is above the threshold (for example, 10), which means all application servers are busy, then the job cannot be executed. Therefore, nothing happens further. Once the candidate application server is chosen by the judge bean, the judge bean will publish the message including the jobid and application server to Topic2 (resides in the same database as Topic1). Message driven beans mdb6 through mdb10 are subscribers of Topic2 and they will receive the message in Topic2 and compare their application server ID with the application server ID in the message. The matching application server will run the simulation by reading inputs from database and update the job queue, indicating that the job is being executed.

In the case that some application server does not report their load averages to the load average queue after 1 second (the timeout interval), then this indicate that the application server is down and all simulations running on it will be migrated to other application servers. We'll show how this can be achieved in the section of implementing simulation-resuming.

All the queues and topics are stored in an Oracle database, (here we use the Oracle Advanced Queuing technology, which is similar to IBM's MQseries and thus guarantees delivery of messages), instead of in the memory of application servers. This will ensure submitted simulations will be executed eventually even if there is a system failure.

A simpler method to get the load average of remote machines is to run a command "ruptime", which requires rwhod daemons running on the HTTP server and all application servers. After setting up the rwhod daemons, we can execute "ruptime -l" on the HTTP server to get load averages of all application servers. The "-l" option ensures that load averages are sorted. Since the job dispatcher knows which application server should execute the new job, it can send the message to Topic2, and get around Topic1 through the judge bean, as shown on Figure 2.3.

2.4.2 A new load-balancing algorithm

The goal of the new load-balancing algorithm is to find an application server such that the job can be completed earliest.

The previous simple load-balancing algorithm cannot guarantee that the application servers are balanced (in the sense that the simulation jobs complete earliest). For instance, suppose that we have only two application servers, and 4 simulation jobs are to be executed. 2 of them are shorter and other 2 are longer (in the sense of execution time). The previous simple algorithm may assign two shorter jobs to one application server and the other 2 longer jobs to another application server. After the shorter jobs complete execution, the two longer jobs are still running on one application server, while the other application server is idle.

To make more efficient use of resources, we propose an algorithm which tries to find an application server such that simulation jobs can complete in shortest time. The basic idea is to predict the job completion time for each application server, based on the history of jobs.

Execution time prediction has been studied in the fields of operating systems and parallel computing, [44] [58] [1] [50] [16] [26] [41]. Dinda proposed an algorithm that estimates the load average and execution time of short tasks in the GRID computing environment using an instance learning approach [15].

Next we'll present our new load-balancing algorithm.

There are database tables keeping track of load average, number of simulations, availability of application servers and predicted completion time of each running simulation, as showing in Table 2.1 and Table 2.2.

Table 2.1. APPLICATION SERVER

Server ID	Load Average	Simulations	Availability
1	1.34	1	yes
2	2.14	2	yes
3	1.35	1	yes
4	2.03	2	yes
5	1.66	1	yes

Table 2.2. SIMULATION

Job ID	Server ID	Running	Start Time	End Time	Expected Time
100	1	yes	05/26/03 12:00:00	NULL	23.45 hours
101	2	yes	05/26/03 12:01:00	NULL	12 hours

The execution time of a simulation depends on the requests of iterations (comes from the simulation configuration), denoted by I, the machine load average, denoted by $L(t)$, a function of time, and the average amount of data generated each iteration, denoted by P (the multiple of 8192 bytes, 8192 bytes is the Oracle data block size for each database). We propose the following formula to compute the expected execution time:

$$ExpTime(t) = I * (\frac{C_1}{t}\int_0^t L(s)ds + C_2P) \tag{2.1}$$

where C_1 and C_2 are constants to be determined by history jobs. Our best hope is that ExpTime(t) is approximately constant.

The expected execution of a simulation is updated at each checkpoint (checkpointing will be discussed in the next section). When a new job is submitted, its execution time will be estimated for each application server based on the current load average of the application server, number of simulations running, and expected completion time of each simulation. The application server with smallest expected completion time will be picked.

Build the History Job Database

Each history job is a record in the database. A record consists of two parts, the input part and the output part. The input part includes simulation configuration, load average of each application server, and average expected completion times of running simulations on each application server. The output part includes actual completion time of the simulation on each application server.

To build the history job database, each simulation is executed on all application servers, until a satisfactory number of simulations are stored in the database, for example, 200.

Compute Constants C_1 and C_2

The constants C_1 and C_2 can be computed by completed simulations. The average load average can be recorded during the execution of a simulation for each application server; the amount of data produced for each iteration can also be recorded. Therefore, the two constants can be computed by solving linear equation systems. Obviously, there are many more equations than parameters. To find reasonable constants C_1 and C_2, we can form the following mathematical problem.

Let n be the number of records in the job history database. Let $x = (x_1, x_2)^t$ be the two constants to be solved. Let $A = (a_i j)n \times 2$ be the coefficients. Let $b = (b_1, ..., b_n)^t$ be the completion times for each job in the history database. Our goal is to minimize $\|b - Ax\|$.

We can apply the singular value decomposition (SVD) for A. Thus there exists a n by n orthogonal matrix U, a 2 by 2 orthogonal matrix V, and a n by 2 matrix Λ with the form $\Lambda = (diag(\lambda_1, \lambda_2), O)^t$ where λ_1, λ_2 are positive real numbers and O is a 2 by $n-2$ zero matrix, such that $A = U\Lambda V$. Let $y = Vx$ and $c = U^t b$, then $\|b - Ax\|^2 = \|b - U\Lambda V x\|^2 = \lambda_1^2 y_1^2 + \lambda_2^2 y_2^2 - 2c_1\lambda_1 y_1 - 2c_2\lambda_2 y_2 + c^t c$. Hence, when $y = (\frac{c_1}{\lambda_1}, \frac{c_2}{\lambda_2})^t$, $\|b - Ax\|$ is minimized. Therefore, we can compute x using $x = V^t y$.

Define Distance of Two Jobs

For each record in the history job database, some attributes are numerical while others are categorical. To define the distance of two records, we can use the so called Heterogeneous Euclidean Metric. The distance of record x and y is defined as follows:

$$d(x, y) = \sqrt{\sum_i w_i * d_i(x, y)^2} \qquad (2.2)$$

The summation in the distance is taken over all attributes of the input part of the records. For categorical attribute i, the distance is 0 if they are same, 1 otherwise.

For numerical attributes i, the distance is

$$d_i(x, y) = \frac{|x_i - y_i|}{\max x_i - \min x_i} \tag{2.3}$$

w_i is the weight put on the attribute i, to identify more important attributes. For example, the expected completion time of other running simulations has a huge impact for choosing which application server, thus we put more weight on this attribute.

Nearest Neighbor Algorithm

For a new simulation, a nearest neighbor will be picked from the history job database. Then from the output part of this nearest neighbor, we can pick the application server which completes the nearest neighbor in shortest time. Then this application server will be the one to execute the new simulation.

2.5 Simulation-resuming Using JTA

As shown in Figure 2.2, simulation data and checkpointing data generated from the simulations running on application servers is stored in different databases, i.e., SD and SM. The two parts of data form one distributed transaction and this transaction spans two different databases. Therefore, we need to use the J2EE transaction service (JTS) to ensure data integrity. Data integrity is critical to the implementation of simulation-resuming.

The simulation jobs are normally long running tasks. For example, a normal simulation may take several hours to accomplish. If some software or hardware failure occurs during the execution, we have to find ways to restart the simulation from a checkpoint prior to the time of the failure. Also, users may want to resume a simulation which was terminated earlier. In this case, we want to start the simulation from where it was terminated.

To handle these cases, we need to checkpoint the simulation periodically during its execution. This raises several issues about the checkpointing procedure. For example, what information about the simulation should be checkpointed? How often the checkpoint should take place? More importantly, how do we synchronize the simulation data and checkpoint data (that's why JTS and JTA are involves)? In the next few paragraphs, we'll provide answers to these questions.

2.5.1 What information should be checkpointed?

Basically, we want to checkpoint all information such that a simulation can be started using this information as input. Take the NOM project as an example, we want to checkpoint the following information:

1. Environmental variables, including requested simulation time and remaining time.
2. Molecule attributes, including counts of atoms, counts of function groups, molecule weights, etc.
3. Positions of molecules.
4. Reaction probability tables for all kinds of reactions for all molecules.

To restart a simulation, all the above data will be read by the simulation program. Each instance of molecule object will contribute one row to the checkpoint database table. When the number of molecules increases, the amount of data to be checkpointed also increases and so does the time required to finish the checkpoint process.

2.5.2 How often should checkpoint take place?

The frequency of checkpointing of a simulation depends on the quality of the simulation program. Based on our experience, some simulations may exhaust available physical memory and throw an OutOfMemoryErrorException during their execution. In this case, we want to checkpoint the simulations before they crash.

The frequency of checkpointing also depends on the requested execution time and the amount of data generated by the simulation. The simulation configuration determines statistically how many hours the simulation will execute and how much data will be generated. For example, if a simulation requires 24 hours to run, we may checkpoint it every 1 hour. If another simulation requires to run just 1 hour, we way choose not to checkpoint it, since in the case of failure, we can start the simulation from the scratch.

Two basic approaches to determine checkpoint frequency are

- checkpoint interval: the number of megabytes of data generated by a simulation, for example, 10MB.
- checkpoint timeout: the number of minutes the simulation has been executing, for example, 30 minutes.

The web interface provided for users to configure simulations allows the user to terminate a simulation while it's running. To integrate simulation checkpointing with the terminate option, we allow a user to terminate a simulation with one of the following three options:

- abort: if a user aborts the simulation, it's not possible to resume, since the data between the simulation database and simulation manager database is not consistent.
- immediate: the simulation will rollback to the checkpoint prior to the time of termination.
- transactional: the simulation will continue running until finishing the next checkpointing.

To further improve the reliability of simulations, at each checkpoint, an "I'm alive" message is inserted into a database with current timestamp. The job dispatcher described before will check the message to ensure that the simulation is still running.

2.5.3 How to synchronized simulation data and checkpoint data?

JTS and JTA will play an important role to synchronize the simulation data and checkpoint data. The two kinds of data will go to two different databases in one distributed transaction. This transaction either commits or rollback and ensure the data in the two databases are integrated, using the X/open two phase commit protocol.

The following figure shows the code to implement checkpointing.

2.5.4 Summary

To summarize, we employ JTS/JTA to accomplish checkpointing of simulations. If a simulation need to be resumed, the simulation job is inserted back to the job queue queue1, with a mark that indicates its resuming status (i.e., the number of times of resuming) and the location to read checkpointing data. The job will be executed on one of the application servers as a new simulation. The corresponding fields in queue1 are NULLs for new simulations. Resuming simulations have higher priority than new simulations, since they have smaller JOB_IDs.

In the case of an application server failure, all simulation jobs running on it will be inserted into the job queue queue1. They will be migrated to other application servers by restarting them from the checkpoint prior to the time of failure.

```
import java.sql.*;
import javax.sql.*;
import oracle.jdbc.*;
import oracle.jdbc.pool.*;
import oracle.jdbc.xa.OracleXid;
import oracle.jdbc.xa.OracleXAException;
import oracle.jdbc.xa.client.*;
import javax.transaction.xa.*;

class Checkpoint{
  public static void main(String[] args)
     throws SQLException{
    try{
      //Create XADataSource instances
      OracleXADataSource oxds1=new OracleXADataSource();
      oxds1.setURL("jdbc:oracle:thin:@db1:1521:sd");
      oxds1.setUser("scott");
      oxds1.setPassword("tiger");
      OracleXADataSource oxds2 = new OracleXADataSource();
      oxds2.setURL("jdbc:oracle:thin:@db2:1521:sm");
      oxds2.setUser("scott");
      oxds2.setPassword("tiger");

      // Get a XA connection to the underlying data source
      XAConnection pc1  = oxds1.getXAConnection();
      XAConnection pc2  = oxds2.getXAConnection();

      // Get the Physical Connections
      Connection conn1 = pc1.getConnection();
      Connection conn2 = pc2.getConnection();

      // Get the XA Resources
      XAResource oxar1 = pc1.getXAResource();
      XAResource oxar2 = pc2.getXAResource();

      // Create the Xids With the Same Global Ids
      Xid xid1 = createXid(1);
      Xid xid2 = createXid(2);

      // Start the Resources
      oxar1.start (xid1, XAResource.TMNOFLAGS);
      oxar2.start (xid2, XAResource.TMNOFLAGS);

      // update SD and SM with conn1 and conn2
      updateSD (conn1);
      updateSM (conn2);

      // END both the branches -- THIS IS MUST
      oxar1.end(xid1, XAResource.TMSUCCESS);
      oxar2.end(xid2, XAResource.TMSUCCESS);

      // Prepare the RMs
      int prp1 =  oxar1.prepare (xid1);
      int prp2 =  oxar2.prepare (xid2);

      boolean do_commit = true;
      if (!((prp1 == XAResource.XA_OK) || (prp1 == XAResource.XA_RDONLY)))
         do_commit = false;
      if (!((prp2 == XAResource.XA_OK) || (prp2 == XAResource.XA_RDONLY)))
         do_commit = false;

      if (prp1 == XAResource.XA_OK)
        if (do_commit)
           oxar1.commit (xid1, false);
        else
           oxar1.rollback (xid1);

      if (prp2 == XAResource.XA_OK)
        if (do_commit)
           oxar2.commit (xid2, false);
        else
           oxar2.rollback (xid2);

      // Close connections
      conn1.close();
      conn1 = null;
      conn2.close();
      conn2 = null;
      pc1.close();
      pc1 = null;
      pc2.close();
      pc2 = null;
    }catch(SQLException e){
    }catch(XAException xe){
    }
  }
}
```

Figure 2.4. Implementing checkpoint using JTS/JTA

2.6 Collaboration: BBS, Chatroom, File-uploading, XML processing and Simulation-sharing

The proposed infrastructure will integrate BBS, chatroom and file uploading utilities for the scientists to share knowledge, for example, research papers and simulation configurations. The users can also upload a simulation configuration file in XML format. The XML file will be processed automatically using the Oracle XML Query Utility and will be transferred through JMS to invoke a simulation. This will save a lot of work for the user since the user will not need to go through the configuration wizards to provide inputs for simulations. Simulation configuration and data analysis can be shared between users. Configuration setup can be recommended to the users based on their simulation history. .

Figure 2.6 is a screen shot of the web interface which includes a BBS system, a chatroom and a NOM simulator.

2.7 Simulation reports

Simulation reports can be delivered in two different ways: (1) through the Oracle Reports server and (2) through XML using XSQL. XSQL generates XML files using SQL statements. The XML files can be transformed using XSLT to other formats, HTML for example, and then be published on the web. The two different approaches

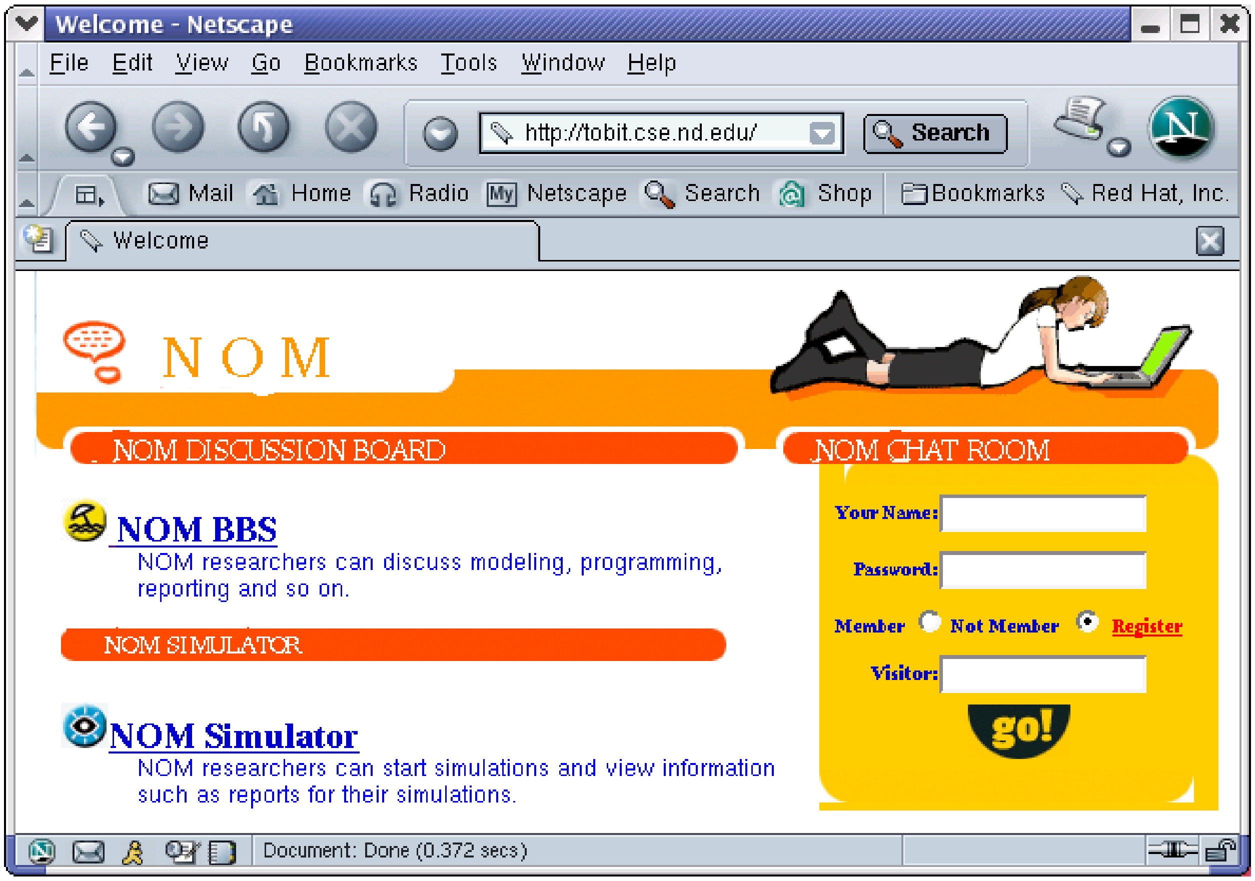

Welcome - Netscape
File Edit View Go Bookmarks Tools Window Help
http://tobit.cse.nd.edu/
Search
Mail Home Radio Netscape Search Shop Bookmarks Red Hat, Inc.
Welcome
NOM
NOM DISCUSSION BOARD
NOM CHAT ROOM
NOM BBS
NOM researchers can discuss modeling, programming, reporting and so on.
NOM SIMULATOR
NOM Simulator
NOM researchers can start simulations and view information such as reports for their simulations.
Your Name:
Password:
Member
Not Member
Register
Visitor:
go!
Document: Done (0.372 secs)

Figure 2.5. The web interface

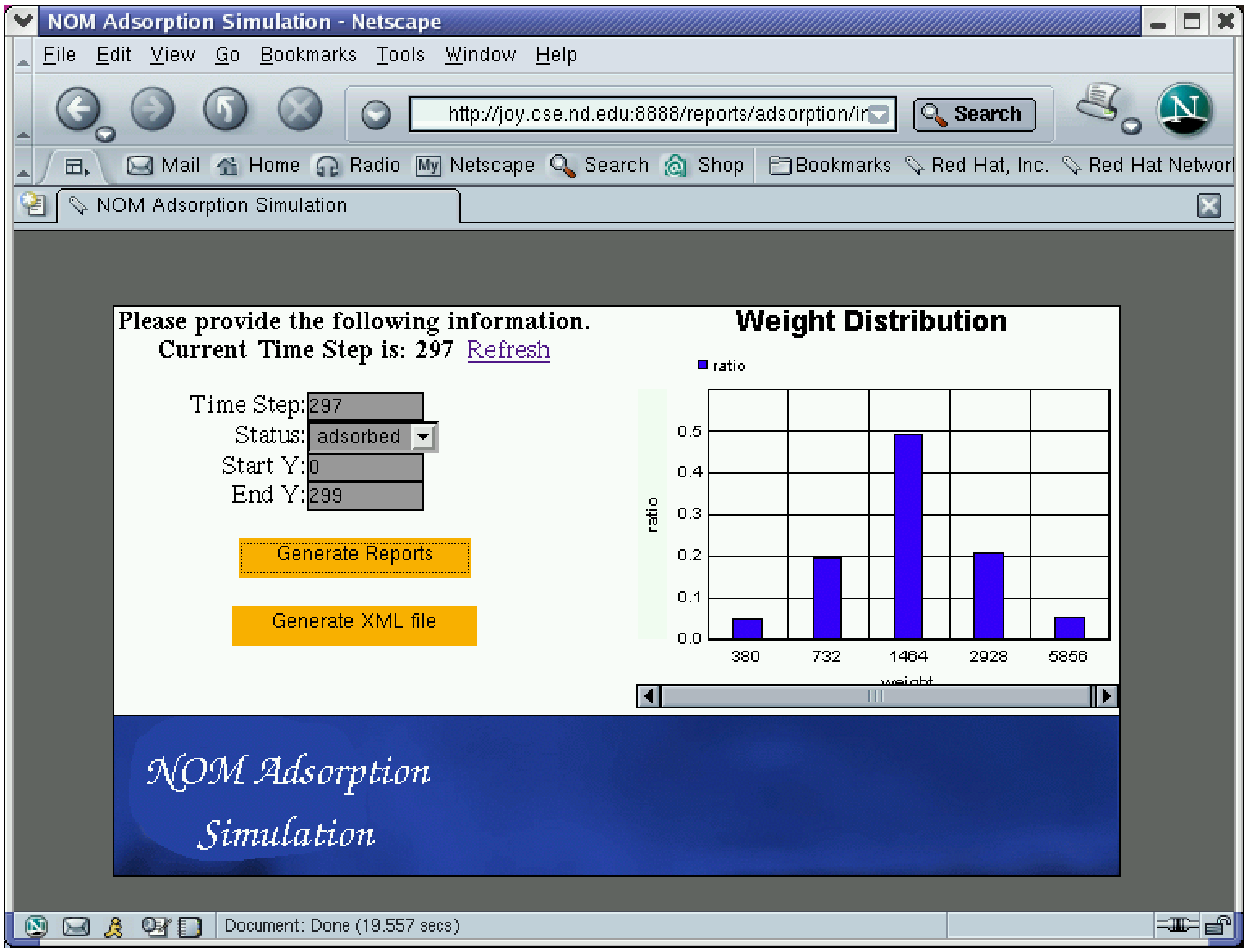
use the new Oracle9i analysis SQL functions. The following statement shows a sample of a report statement using the analysis SQL function ratio_to_report.

```
select weight, count(weight) "count",
sum(count(weight)) over () "total",
to_char(ratio_to_report(count(weight)) over (), '0.9999') ratio
from adsorption ad
where sessionid={@sessionid} and
timestep={@timestep} and
status={@status}
and ad.position.y between {@ystart} and {@yend}
group by weight;
```

Figure 2.6. SQL statement in XSQL

SQL for analysis in data warehouses is new in Oracle9i. Here we take advantage of this to generate statistics using the SQL functions which include ranking functions, windowing aggregate functions, reporting aggregate function, linear regression functions and so on. The statistics are delivered through Reports Server or XML using XSQL.

The sample reports page provides two buttons to generate graphical reports and XML respectively. Figure 2.7 and Figure 2.8 show the two kinds of reports.

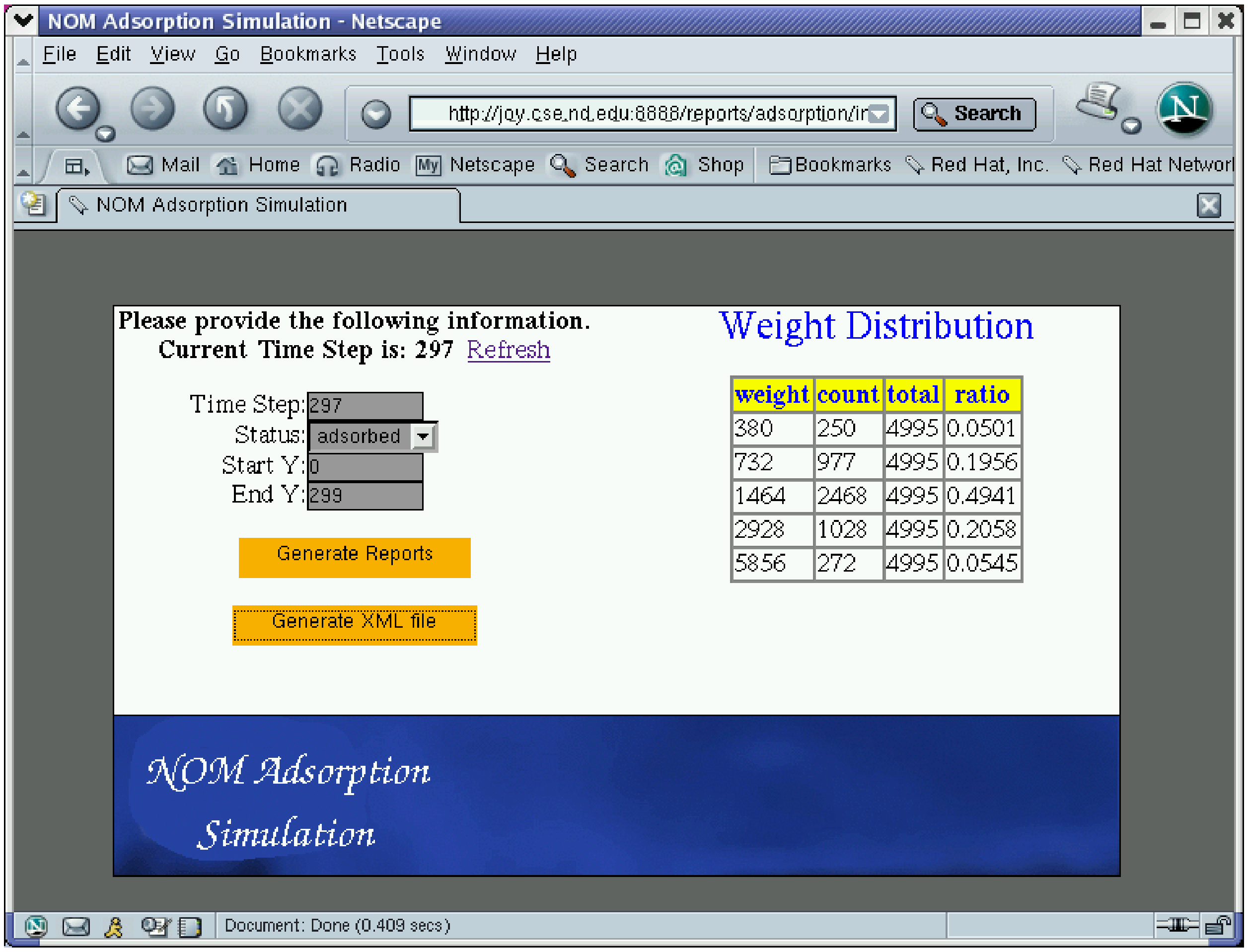

NOM Adsorption Simulation - Netscape
File Edit View Go Bookmarks Tools Window Help
http://joy.cse.nd.edu:8888/reports/adsorption/in
Search
Mail Home Radio Netscape Search Shop Bookmarks Red Hat, Inc. Red Hat Network
NOM Adsorption Simulation
Generate Reports
Generate XML file
Weight Distribution
ratio
0.0
0.1
0.2
0.3
0.4
0.5
380
732
1464
2928
5856
weight
NOM Adsorption Simulation

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 2.7. Graphical reports


NOM Adsorption Simulation - Netscape
File Edit View Go Bookmarks Tools Window Help
http://joy.cse.nd.edu:8888/reports/adsorption/in
Search
Mail Home Radio Netscape Search Shop Bookmarks Red Hat, Inc. Red Hat Networ
NOM Adsorption Simulation
Generate Reports
Generate XML file
NOM Adsorption Simulation
Document: Done (0.409 secs)

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

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

Weight Distribution

weight	count	total	ratio
380	250	4995	0.0501
732	977	4995	0.1956
1464	2468	4995	0.4941
2928	1028	4995	0.2058
5856	272	4995	0.0545

Figure 2.8. Reports through XML

CHAPTER 3

Data Cleansing for Warehousing and Mining

3.1 Introduction

Data Cleansing is a preprocess step for data warehousing and data mining. The process of data cleansing is normally computationally expensive, hence it was not possible to do with old technologies. Nowadays, faster computers allow data cleansing to be performed in an acceptable amount of time on a large amount of data. There are many issues in the data cleansing area that interest researchers. These issues consist of dealing with missing data, determining erroneous data, etc. Different issues require different approaches. In this chapter, we are interested in the so called "dirty data" [27] [28]. Two records whose appearance differs from each other may represent the same entity in the real world. We call such records "essentially duplicate" or "similar". Suppose D is a record in a database. Then the records in the database that are similar to D are called "dirty" and their similarity to D should be identified. Many applications require that such records should be removed from the databases or merged together. We propose two different methods for cleansing data. The proposed methods are both domain independent, in that no domain knowledge of data is necessary. Our methods can handle more general data cleansing problems.

So what is the definition of "data cleansing"? Unfortunately, there is no commonly accepted definition. Different definitions depend on the particular area in which it is applied. The major areas that involve data cleansing are data warehousing, knowledge discovery in databases (KDD), and total data quality management (TDQM). In this chapter, we are interested in data cleansing for data warehousing and mining. The data cleansing process will be applied to the NOM and OSS datasets for the purpose of data warehousing and mining.

A data warehouse is a database that is designed for query and analysis rather than for transaction processing. It usually contains historical data derived from transactions data, but it can include data from other sources. It separates analysis workload from transactions workload and enables an organization to consolidate data from several sources. In addition to a database, a data warehouse environment includes an extraction, transformation and loading (ETL) solution, an online analytical processing (OLAP) engine, client analysis tools, and other applications that manage the process of gathering data and delivering it to business users [34] [39].

A common way of introducing data warehousing is to refer the characteristics of a data warehouse as set forth by Inmon (1996): subject oriented, integrated, nonvolatile, and time variant. We will take advantage of the time variant characteristic when we build our sample database for data cleansing. To build a data warehouse, ETL must be involved and data cleansing is a part of the ETL process.

Several databases are merged to build a data warehouse. Records referring to the same entity are represented in different formats in different databases or are possibly represented erroneously. Thus, duplicate records or essentially duplicate records will inevitably appear in the merged database. Data cleansing is to identify and remove these duplicates. In the business world, this problem is called the merge/purge problem [27] [28]. Some researches have been done in the field of data cleansing [43] [48] [46] [45] [31].

We propose two different data cleansing algorithms in this chapter. The rest of the chapter is organized as follows: in section 2, we present some related works; in section 3, we give an overview of our sample database approach, then provide details; In section 4, we sketch the performance analysis of the sample database approach; In section 5, some conclusions are drawn and future work is proposed for this algorithm; In section 6, a new data cleansing algorithm using SparseMap will be discussed.

3.2 Related Work

In the current marketplace, data cleansing is heavily focused on customer lists. There are many companies providing data cleansing service. Among them are DataFlux [13], Hart-Hanks Data Technologies [21], Innovative Systems Inc [33], and Vality Technologies [62]. Unfortunately, none of them are willing to reveal the algorithms they use to cleanse data.

Recently, companies have started to produce tools and other data cleansing services that do not address specifically the customer address lists but do rely on domain specific information provided by the customer. Among them are Centrus Merge/Purge Module [60], and DataCleanser [12].

More recently, data cleansing is regarded as a preprocessing step in the KDD process [20] [8]. Various KDD and data mining systems perform data cleansing activities in a very domain specific fashion. In [59], data cleansing is regarded as the process of examining databases, detecting missing and incorrect data and correcting them. The Recon Data Mining system is used to assist the human experts to identify a series of errors types in financial data systems.

One of the commonly used data cleansing approaches is based on the following framework [27] [28]: First create keys based on their knowledge of the error patterns of the database, then group similar records close to each other by sorting all records in the databases on the keys, and finally use a sliding window protocol to remove duplicates by scanning the sorted records. In order to improve accuracy, the results of multiple passes can be combined by computing the transitive closure of all discovered pairwise ”is a duplicate of” relationships.

The main difficulty of this approach is at how to create effective keys for the records to capture most errors or duplicates in any database without previous knowledge of the error patterns. The known methods for generating keys are all based on some simple heuristics that target certain presumed error types but may not be effective for most other error types.

3.3 Our Approaches

In view of this drawback of the current data cleansing approaches, we propose a somewhat different approach that aims for general databases rather than specific ones. Our approach works as follows: First use a unified method to create multiple types of keys (each type captures certain types of errors) for any target databases, and then scan the database multiple times to remove duplicates (each scan is based on a different key type).

At first sight, our approach seems quite similar to the known approaches. However this is not the case. A main difference is in how the keys are generated. To let our keys effectively capture the errors in any given databases, we propose to “learn” about the actual errors in the database rather than assuming it always has a few fixed types of errors. This “learning” is done as follows: We first generate a small sample database from the original database, and then do a pairwise approximate record matching in the sample database. Two records are said to be approximately matching if their “distance” is within a certain threshold, defined by the user. Thus, the set of database records form a finite metric space. If two records approximately match each other, then we consider them as duplicates and remember their matching patterns (i.e., error patterns). After we have obtained error patterns from the learning process against the sample database, we create a key for each pattern. We then scan the database multiple times, one for each key. Each scan sorts the database based on the key, and uses some modified sliding window protocol to remove duplicates.

The main advantage of our approach is that it does not rely on any presumed a priori knowledge of the error patterns in a database, since our approach finds the error patterns for any database. The second advantage is that although this approach appears to take a relatively longer running time, it is much more effective. This is because, in practice, the error patterns likely will remain the same for databases of the same kind. That is, once we find the error patterns in one kind of databases, we can keep using this information to guide the data cleansing on such databases, allowing continuous update and growth on them (without having to learn their errors again and again).

3.3.1 Similarity

To measure whether two records in a database are similar to each other, we need some metric. In this paper, we focus on the so called "edit distance", although our approaches are applicable to other metrics too.

Given two strings s and t, the edit distance of s and t denoted by $d(s,t)$ is defined the number of insertions, deletions and replacements on single characters of one string to obtain the other. For example, the edit distance between "abcd" and "abd" is 1 since "abd" can be obtained from "abcd" by deleting "c". Or "abcd" can be obtained from "abd" by inserting "c" into "abd".

In the literature, the search problem is in many cases called "string matching with k differences". The distance is symmetric and it holds $0 \leq d(s,t) \leq max(|s|,|t|)$. Other metrics can be applied with our approach are:

- Hamming distance [55]:allows only replacements.
- Episode distance [22]:allows only insertions.
- Longest Common Subsequence distance [51] [2]:allows only insertions and deletions.

We need an algorithm to determine whether two records in a database are similar. [27] [28] use some production rules or equational theory to determine whether two records are similar. [47] treats the whole record as a long string and then uses edit distance to determine their similarity.

Fortunately, there exists a fast algorithm to determine whether two strings have edit distance less than k. In [63], Ukkomen proposed an algorithm to check in time $O(k^2)$ whether two strings have distance $\leq k$ or not. Interestingly, the time complexity does not depend on the length of the two strings. The threshold value k must be chosen carefully to measure the similarity of two records. In the current data cleansing algorithms, Ukkomen's algorithm has not received any attention and thus is not used. Some other algorithms whose time complexity depends on the lengths of the records are used to determine the similarity of two records. One difference of our approach from other approaches is that we use Ukkomen's algorithms for approximate string matching. The choice of the threshold k is dependent on specific data cleansing problems. Normally, k is a small number between 2 and 5. In practice, a record has hundreds of fields and thus it's time-consuming to compute the distance of two records. Ukkomen's algorithm simplifies this computation.

3.3.2 Build Sample Database

The sample database is a subset of the original database. The goal of the sample database is to enable us to find useful error patterns of approximately matching records. It is not a trivial process to create such a sample database since we don't know what records should be chosen such that we do not miss major error patterns.

One could propose that we can randomly pick a subset of records from the database uniformly. But the drawback of this approach is that there is a high probability that duplicates may not be present in the subset. One also could randomly pick a contiguous subset from the database. But this approach could just pick all the records from one or two different databases. (Recall that several databases are merged together to form a large database.)

Note that as mentioned above, several databases are merged together to build the data warehouse. So duplicates are likely far away from each other, i.e., they are not physically stored close to each other. We don't want to miss such pair of records since they are the contributors for an error pattern. With this in mind, we propose a method to build the sample database as follows.

We treat all fields of the table as strings. Then we concatenate all the fields as a long string. Finally we sort the resulting strings alphabetically. After sorting, similar records are likely to end up near each other in their physical storage. A randomly chosen w contiguous rows of the resulting sorted records are used to be the sample database. w is a small number (for example, 1K) such that duplicates exists in the first w rows of the sorted table. To improve accuracy, the strings are sorted one more time reversely for each string. And contiguous w' (for example, 1K) rows of the resulting sorted records are chosen randomly to be merged into the sample database.

Once the sample database is built, the next step is to find error patterns from the sample database.

3.3.3 Finding Error Patterns and Creating Keys

Since for each created key, we have to pass the database once, it's not practical to create too many keys. For efficiency, we only prefer to create 3 or 4 keys. Therefore, we only want to find out the most popular error patterns. For each error pattern, a key will be created.

How do we detect the error pattern? For each pair of approximate duplicates, we record the fields in which they are different. For example, in column 1 and column 2, they are different. Then we choose the 3 or 4 field combinations for the most found error patterns. To create the key, we concatenate the other remaining fields as a first try. To simplify the construction of keys, we only consider them as a combination of some of the fields of the original database.

3.3.4 Scanning Database

Now we created keys for each error patterns. Next, we sort the database once for each key.

After the database is sorted, we apply the sliding window protocol to scan duplicates as follows. The sliding window protocol is similar to [28], but the window size is not fixed. First choose window size w, for example, 32. Then we scan the first w records in the database, if there are duplicates in this window, mark one of them as the duplicate of the other, then shrink the window with size w/2 and forward the window 3w/4. If there are no duplicates, then we enlarge the window with size 2*w and forward the position w/2. This sliding window protocol is fast since we step forward fairly quickly. Keep this process until all the records are scanned. Then we run the process again against the second key. When we arrive at our goal, for example, we have already removed a certain number of duplicates, we might want to stop it since this process in computationally expensive.

Clusters of approximate duplicates, which is called the transitive closure in [28] can be computed using a Union-Find data structure [47]. If record x and y are approximate duplicates, record y and z are approximately duplicates, then we treat records x, y, z as in the same cluster, no matter whether record x and z are approximate duplicates or not. As we know, the similarity relationship is not transitive, but in practice, it is used in some data cleansing software.

3.4 Performance Analysis

The time complexity of our approach is dominated by the sorting process. In the algorithm, we first made two sorts for the long string formed by concatenating all fields. Then we created the keys based on the error patterns, which are computed from pairwise comparison of the records of the sample database. For each key, a sorting is applied, then the sliding window protocol is applied. Finally, the approximate duplicates clusters are formed.

3.5 Conclusion and Future Work

The data cleansing approach proposed has its advantage over previous approaches in that it utilizes the learning process and creates smart keys from learning, thus can perform the process on any databases without known error patterns. Meanwhile, it utilize a more efficient algorithm to determine whether two strings have small edit distance.

The data cleansing procedure will be implemented using PL/SQL. The reason we choose PL/SQL is that it is a language operating inside the database and thus out performs other languages. Real world data from sourceforge.net will be applied to this algorithm. We also plan to generate sample databases to test the accuracy of this approach. Comparison of our approach against the known approaches in [27] [28] will be made to verify its use.

Another approach of data cleansing is currently under way. The main idea is as follows: in the first step, each record is mapped to a point in the multi-dimensional Hilbert space; in the second step, some spatial access method (SAM) is employed to do a similarity join. The details of this method will be presented in the next section.

3.6 Cleanse Data Using SparseMap

This chapter presents a new data cleansing algorithm. The basic idea is a two step approach. First, we map every record to a point in the l_∞^d space. The map is an isometric under the assumption that every record has at most B approximate duplicates where B is a constant. In the second step, we employ some spatial access method, as in [40]. This algorithm shares the idea of [40], but we use a different mapping method, since the mapping in [40] has potential high distortion and thus cannot guarantee 100 percent recall.

3.6.1 Introduction

In [40], each feature of a record is considered as a string. For each feature of the records, the values of the feature are mapped to a multi-dimensional Euclidean space. A variation of FastMap [18], called StringMap, is used. After mapping, a similarity join algorithm [29] is used to find close point pairs. The authors claim that their algorithm can achieve 99 percent recall in their experiment. The major drawback of the FastMap is that there is a potential high distortion of the mapping and thus cannot ensure the quality of the resulting data cleansing method.

In this chapter, we use another mapping method SparseMap [30]. Actually, our mapping is similar to the original Lipschitz mapping on which SparseMap was built. The original SparseMap can not guarantee 100 percent recall either. But we try to modify the algorithm such that we can control the distortion of the resulting mapping, such that 100 percent recall can be guaranteed.

3.6.2 Mapping to l_∞

A metric space M=(X, D) is called a (1,2)-B metric, if the distance between any two points is 1 or 2, and for any point in X, there are at most B points within distance 1 from it.

We can consider our database of records to be a (1,2)-B metric space in the following way. We say two records are approximate duplicates if their distance (for example, edit distance) is less or equal to k (a predefined threshold). If two records are approximate duplicates, then they have distance 1. Otherwise, they have distance 2. If we also assume that every record has at most B records in the database, then we get a (1,2)-B metric space.

We have the following lemma for a (1,2)-B metric space.

Lemma 3.1. *A (1,2)-B metric space M=(X,D) can be isometrically embedded into $l_\infty^{O(B\log N)}$, where N is the size of the metric space.*

Proof. We use the same approach as in [7], by probabilistic method. Let $d = O(B\log N)$. $\forall i$, where $1 \le i \le d$, choose a subset S_i of X such that each element of X included in S_i independently with probability $\frac{1}{B}$. Define the mapping $F : M \to l_\infty^d$ by

$$F(x) = (D(x, S_1), D(x, S_2), ..., D(x, S_d)) \tag{3.1}$$

Next, we prove that F is indeed isometric. First, we have $Pr[|D(x,S_i) - D(y,S_i)| = D(x,y)] = \Omega(\frac{1}{B})$. In fact, consider the case $D(x,y) = 2$. Then we have: with probability $\frac{1}{B}$, $x \in S_i$, and $Pr[u : D(u,y) < 2 \cap S_i = \phi] = constant$, and the two probabilities are independent. Similar for the case $D(x,y) = 1$. By repeating d times, we are sure that F is isometric with high probability. $\square$

Once we have isometric mapping to l_∞, we can use a spatial access method as in [40]. We do not need to choose thresholds and dimensions as done in [40] and 100 percent of recall is guaranteed.

The assumption that every record has at most B approximate duplicates is very important, since it reduces the dimension of the hosting space from $\log^2 N$ to $B \log N$. And the resulting mapping is not only contractive but also isometric, which is essential to preserve the underlying structure of the original metric space.

However, there is a problem to algorithmically construct the mapping F since it takes quadratic time complexity. And there is no point of mapping the records in the database, because a pairwise comparison takes quadratic time anyway. To remedy this situation, we use similar heuristics as used in SparseMap. The basic idea is loop interchange.

Here is the algorithm.

- For every subset S_i
  - For every point $x \in X$
    - For every point $y \in S_i$, compute the approximate distance $D'(x, y) = \max_{l=1}^{i-1} |x_l - y_l|$, where x_l is the $l - th$ coordinate of x.
    - Find the $\sigma y's$ with smallest D' distance to x.
    - Evaluate the true distance $D(x, y)$ for each such y.
    - Take $x_i = D(x, y')$ where y' is the y with smallest $D(x, y)$.

In the algorithm, for every point, every coordinate, we compute σ distances of the original metric space. Therefore, it takes $O(B\sigma N \log N)$ distance computations to construct F. The time complexity is sub quadratic.

The modified algorithm may make F to be not isometric. But we want to experiment on this resulting F and see how it works for data cleansing.

3.6.3 Finding Point Pairs Within Distance 1

We can use the same method as done in [40]. We first build two R-trees for the resulting points in l_∞. Then we traverse the two trees from roots to leaves to find point pairs with distance within 1.

We can use other methods to find point pairs within distance 1, since each coordinate of the points is 0, 1 or 2. Thus a simpler method should exist.

3.6.4 Future Work For the Algorithm

We plan to find a better algorithm to find close point pairs as described above. We also want to implement the algorithm and apply to real data sets, collected from the Web (for example, SourceForge.com).

CHAPTER 4

INTRODUCING DATA MINING TECHNOLOGY AND SOFTWARE

The purpose of this chapter is to provide a short introduction to the data mining software and tools which are to be applied to our problem. Data mining can be used to analyze the simulation data, configuration data and experimental data. The following are the necessary steps to apply data mining:

1. Identify the problem to address

2. Prepare training data to build models

3. Test and evaluate the models built in step 2 (optional)

4. Apply the model to new data

Before data mining algorithms can be applied, data from various resources must be preprocessed in a variety of ways. These preprocessing methods include data cleansing, discretization, feature selection and so on.

The Data Mining Suite from Oracle will be used in our applications. The data mining suite is a library of APIs that enables users to code data mining programs in Java. Oracle's integrated development environment (IDE) includes a tool called DM4J (data mining for java) which can be applied to generate java code quite efficiently, and therefore save us a lot of programming time.

We will apply the following data mining algorithms to facilitate the OSS project and the NOM project: clustering, classification, association rules. In the following sections, we will introduce each of the algorithms and show when the algorithms can be applied.

The rest of this chapter is organized as follows. In the first three sections, we'll explain and show examples about each available algorithm. In the next section, i.e., section 4, we will discuss various techniques to pre-process scientific data. In section 5, we will discuss the methodology of applying data mining to scientific data and the ethics behind the methodology. Finally, in section 6, we list a few applications of data mining in the fields of science and industry.

4.1 Association Rules

Association rules mining tries to find interesting association or correlation relationship among a large set of data items. A typical example of association rules mining is the market basket analysis. An association rule is something like "80% of people who buy beer also buy fried chicken".

Association rules mining can also be applied to predict web access patterns for personalization. For example, we may discover that 80% of people who access page A and page B also access page C. Page C might not have a direct link from either page A or page B. The information discovered might be used to create a link to page C from page A or page B. One example of this application is amazon.com. We often see something like "customers who buy this book also buy book A".

The association rules mining can be applied to both the NOM project and the OSS project.

The NOM project was designed to explore the behavior of NOM molecules. And we hope to find patterns of behaviors. In the NOM project, we are interested in the chemical reactions and adsorption of natural organic matters. Some possible association rules are "reaction A and reaction B implies adsorption", or "reaction

A implies reaction B", etc.

In the OSS project, we might discover some association rules like "developers who involve project A and project B also involve project C", thus we can obtain some correlationship of the different projects.

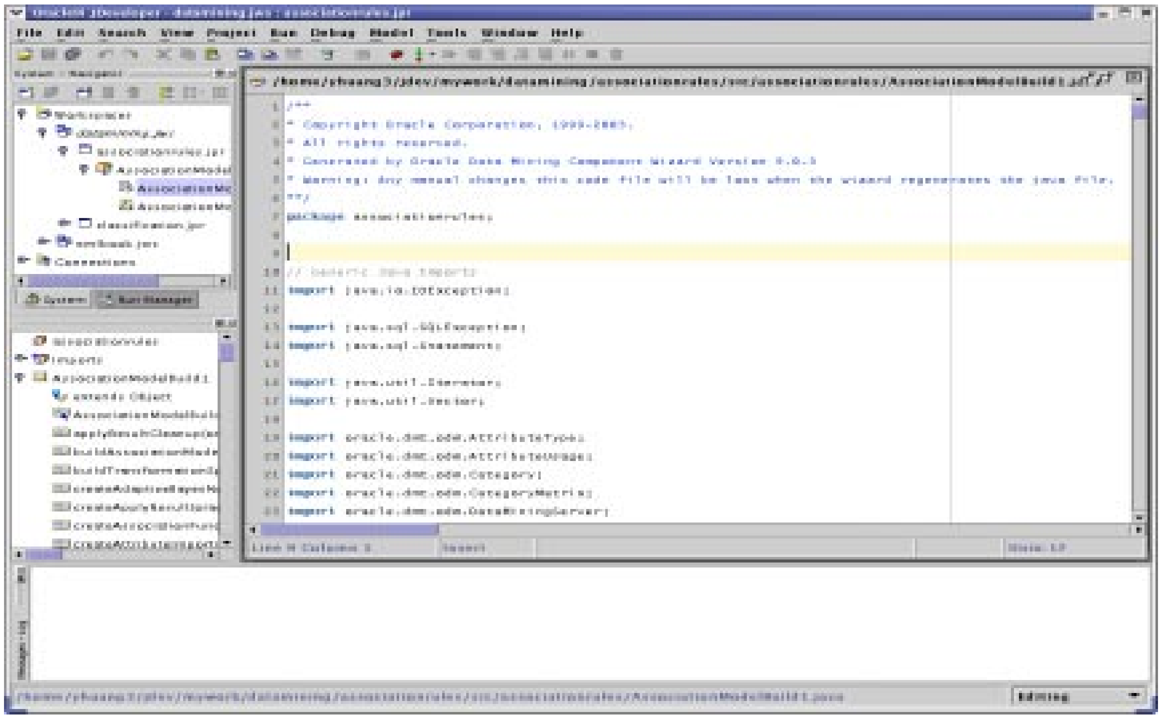
Association rules mining can be formally defined as follows: Let $I = \{i_1, ..., i_n\}$ be a set of **items** and D be a set of **transactions** where each transaction T is a set of items such that $T \subset I$. An association rule is an implication of the form $X \Rightarrow Y$ (X implies Y), where $X \subset I$, $Y \subset I$, and $X \cap Y = \phi$. The rule has **support** s in the data set D if s% of the transactions in D contain both X and Y, and **confidence** c if c% of transactions containing X also contain Y. The problem of association rules mining is to generate all rules that have support and confidence greater than the user-specified support s and confidence c.

A well known algorithm to mine association rules is the Apriori algorithm. The Apriori algorithm can be implemented using SQL. We use the oracle data mining suite, which implements the Apriori algorithm, the Jdeveloper and DM4J to generate association rules from a table of sample data. Figure 4.1 and Figure 4.2 shows the tools used for association rules mining.

4.2 Classification

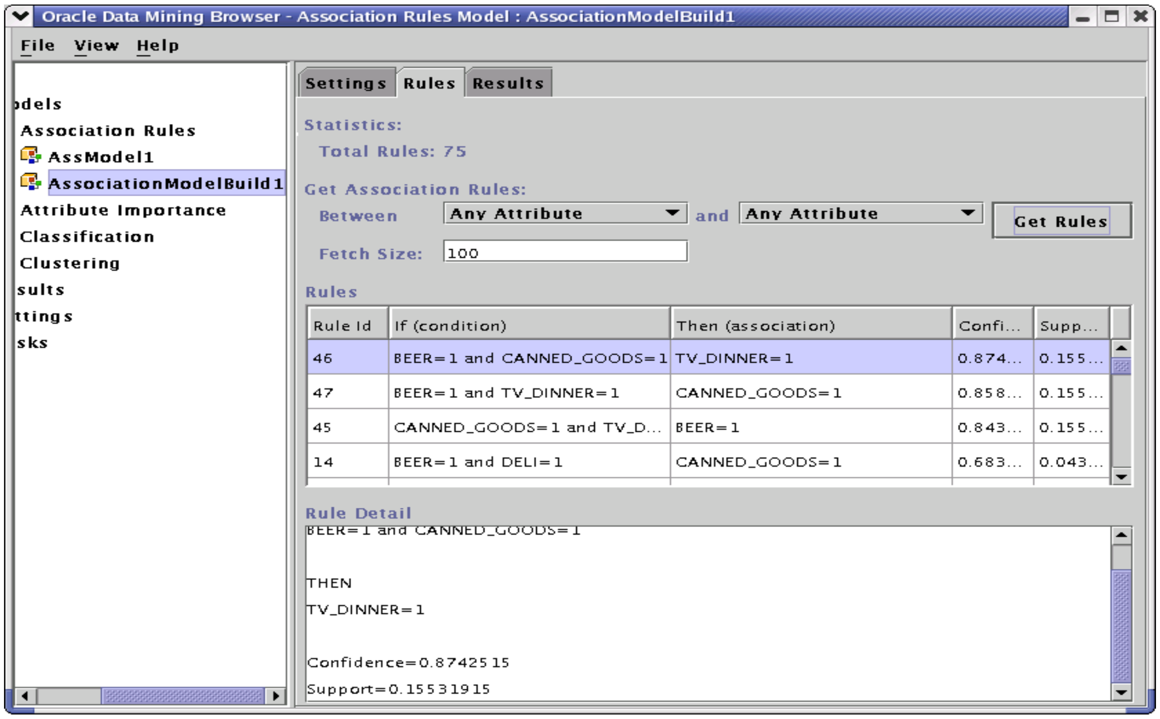
The goal of classification is to predict which of several classes a case (or an observation) belongs to. Each case consists of n attributes, one of which is the target attribute, all others are predictor attributes. Each of the target attribute's value is a class to be predicted based on the $n - 1$ predictor attributes.

Classification is two-step process. First, a classification model is built based on training data set. Second, the model is applied to new data for classification. In the middle of the two steps, some other steps might be taken, such as lift computation.



```
/**
* Copyright Oracle Corporation, 1999-2005.
* All rights reserved.
* Generated by Oracle Data Mining Component Wizard Version 9.0.3
* Warning: Any manual changes this code file will be lost when the wizard regenerates the java file.
**/
package associationrules;


// Generic Java imports
import java.io.IOException;

import java.sql.SQLException;
import java.sql.Statement;

import java.util.Iterator;
import java.util.Vector;

import oracle.dmt.odm.AttributeType;
import oracle.dmt.odm.AttributeUsage;
import oracle.dmt.odm.Category;
import oracle.dmt.odm.CategoryMatrix;
import oracle.dmt.odm.DataMiningServer;
```

Figure 4.1. The Jdeveloper



Statistics:
Total Rules: 75

Rules

Rule Id	If (condition)	Then (association)	Confi...	Supp...
46	BEER=1 and CANNED_GOODS=1	TV_DINNER=1	0.874...	0.155...
47	BEER=1 and TV_DINNER=1	CANNED_GOODS=1	0.858...	0.155...
45	CANNED_GOODS=1 and TV_D...	BEER=1	0.843...	0.155...
14	BEER=1 and DELI=1	CANNED_GOODS=1	0.683...	0.043...

Rule Detail

```
BEER=1 and CANNED_GOODS=1

THEN
TV_DINNER=1

Confidence=0.8742515
Support=0.15531915
```

Figure 4.2. The browser to view association rules

Lift computation is a way of verifying whether a classification model is valuable. A value larger than 1 is normally good.

Classification models can be applied to make business decisions in the industry. Applications include classifying email messages as junk mails, detecting credit card fraud, etc. More recently, data mining has been applied to terrorism detection. The following are some quotes from National Research Council: "Currently one of intelligence agencies' significant problems is managing a flood of data that may be relevant to their efforts to track suspected terrorists and their activities.", "There are well-known examples in which planned terrorist activities went undetected despite the fact that evidence was available to spot it - the relevant evidence was just one needle in a huge haystack."

In the NOM project, we wish to build classification models to predict what kinds of natural organic matter will remain in the system. This is very important in environmental science, as accurate prediction will result in less investment in water treatment, for example. In the OSS project, we wish to build classification models to predict developer churn and acquisition. Both these terms will be explained in chapter 5, as we give a case study of the OSS project.

We will use two different classification models: decision tree and naive bayes. Many models will be built and the accuracy for these models will be compared and the best model will be chosen and deployed to score new data.

4.3 Clustering

Clustering is useful to find natural groupings of data. These natural groupings are clusters. A cluster is a collection of data that are similar to one another. A good clustering algorithm produces clusters such that inter-cluster similarity is low and intra-cluster similarity is high.

Clustering can be used to group customers with similar behavior and to make business decisions in industry. In the NOM project, we want to apply clustering to find groups of similar simulation configurations; we want to apply clustering to find groups of natural organic matters that have similar behavior, for example, they form micelles. In the OSS project, developers may form clusters and behave similarly.

We will apply two different clustering algorithms: k-means and orthogonal-cluster.

4.4 Preprocessing

In most situations, the input data is not ready to feed into data mining algorithms. The data needs to be preprocessed in various ways to make it suitable for data mining.

The most difficult step of data mining is data preprocessing. The data can come from several sources, including experimental data, user input data, and data generated by simulations. Depending on the problem, the data can be two dimensional (spatial), or three dimensional (spatial + time), or even very high dimensional. We need to process these data sources for data mining.

Let's list several approaches to pre-process our data. These approaches include (1) discretize numerical attributes, (2) select important attributes, (3) integrate data across multiple simulations. We'll discuss each of them and show some details of how to accomplish them.

4.4.1 Discretization

The goal of discretization is to reduce the number of values for a given continuous attribute, by dividing the range of the attributes into intervals. Interval labels can be used to replace actual data values. Discretization is specially beneficial when applying the decision tree algorithm. The decision tree algorithm uses a large amount of time to sort data at each iteration. Hence, a smaller number of distinct values will speed up the sorting processes, and thus speed up the whole mining process.

There are several methods to discretize data. The most natural one is to discretize data by creating bin boundaries. The actual attributes data is replaced by bin mean or median. Another technique is to analyze the histograms of attributes. The attribute values are partitioned so that each partition have roughly the same number of records. More advanced discretization techniques includes the entropy-based discretization.

4.4.2 Attribute importance

In the case there are too many attributes for each record, or we want to filter out irrelevant attributes, we need to reduce the number of attributes of data. Then applying the resulting data will smaller number of attributes for the data mining algorithm. Attribute importance is a synonym of feature selection.

4.4.3 Data integration

The data may come from several resources, such as experimental data, simulation data, and configuration data. We need to combine them so that data is integrated. Issues of data integration include identifying same entities, removing redundancy, detecting and removing conflicts and errors.

4.5 Applying data mining to scientific simulations

We apply data mining to find patterns and trends in the data we collect from the real world. Then we apply these patterns and trends to guide simulation design such that the data generated from the simulations contain the patterns and trends. This ensures that the simulations really reflect the real world phenomenon and thus the simulation program is valuable.

This approach differs from the traditional approaches in science, in which researchers would formulate hypotheses predicting certain patterns would exist in the data first. Then the researchers would verify their hypotheses by finding the patterns in the data.

In our approach, we try to find the unexpected patterns in the real world data first and then write simulation programs to verify these patterns. Actually, this approach we first proposed in 1970 by Tukey at Princeton University [38]. He suggested using statistics to explore data, rather than simply to test hypotheses about the data. Here, we use data mining instead of just statistics. Data mining can look through large datasets for patterns that the human might never be able to find. The humans are the ones to explain the patterns.

Many scientists have already been using this methodology (patterns and verify) in the fields of astronomy. Researchers such as U. Fayyad and J. Gray at Microsoft Research used it to mine astronomy data [3], [4]. Here, we use the same approach to the fields of social science and environmental science.

4.6 Applications of Data Mining

Data mining enables us to exploit the full potential of our data collecting abilities. Scientific data can be obtained from experiments, observations and simulations. The diversity of scientific applications provides a rich environment for the practice of data mining.

Data mining has been applied to fields such as astronomy, biology, industry and business.

In 1996, Fayyad used decision trees to classify star/galaxy [19]. In 1998, Burl et al developed a tool called JARTool to detect volcanoes on Venus using data mining

[42], [37]. Other projects such as Diamond eye [14] and Sapphire [56] are developed for identifying astronomy objects.

Recent research in DNA analysis has led to the discovery of genetic causes for many diseases and disabilities, as well as the new discovery of new medicines for treatment. An important focus in this research area is bioinformatics. Bioinformatics analyzes gene sequences using data mining [9].

Data mining has been used for loan payment prediction and customer credit policy analysis, classification and clustering of customers for targeted marketing, detection of money laundering and other financial crimes.

Data mining can even be applied to attack SARS (severe acute respiratory syndrome), one of the dangerous diseases [57].

CHAPTER 5

DATA MINING APPLICATION (1): OSS

The success of open source software (OSS) has recently attracted much attention from researchers in many fields, including economics, sociology, software engineering and computer science. There are many lessons to be discovered from the OSS phenomenon, for example, understanding the motivation behind OSS which drives many developers to dedicate their effort without any monetary benefits.

The open source software (OSS) development phenomenon has been studied by researchers using different simulation models such as social network theory, agent-based modeling, etc [24], [23]. Here we propose a data mining approach to uncover interesting patterns, for example, the evolution patterns of OSS, from the massive amount of data collected from the Web.

In this chapter, we'll give a case study of the OSS problem using the data mining approach. We gathered data from the open source software communities, such as sourceforge.net, freshmeat.net and so on. We combine data collected from these sources in the data preparation step, using some data cleansing techniques developed from the previous chapters. Then we can apply data mining to the data.

5.1 Data preparation

Shell scripts were developed to fetch data from sourceforge.net on a monthly basis from February 2002 to March 2003. The data collected is in flat text format.

To better analyze the data, we need to put it into some relational database. A record of data includes project id to identify a project, developer id to identify a developer, developer's name, developer's role in the project (such as developer, translator, documenter, project manager, etc) and developer's email address. We carefully designed a data warehouse to facilitate the data. First of all, we create a table called SOURCEFORGE in the data warehouse. This was done in the following steps:

5.1.1 Create Table SOURCEFORGE

The data collected is on monthly basis, to accommodate all the data in one large table, we add one more column MONTHID to hold the month name in which the data is collected. Further more, for better performance and easier maintenance, the table is partitioned by MONTHID. Figure 5.1 demonstrates the creation of table SOURCEFORGE.

```
crcreate table sourceforge (
projid number,
username varchar2(100),
userid varchar2(20),
userrole varchar2(50),
email varchar2(100),
monthid varchar2(20),
constraint pk_sourceforge primary key (projid, userid, monthid))
partition by list (monthid) (
 partition feb2002 values ('Feburary_2002'),
 partition mar2002 values ('March_2002'),
 partition apr2002 values ('April_2002'),
 partition may2002 values ('May_2002'),
 partition jun2002 values ('June_2002'),
 partition jul2002 values ('July_2002'),
 partition aug2002 values ('August_2002'),
 partition sep2002 values ('September_2002'),
 partition oct2002 values ('October_2002'),
 partition nov2002 values ('November_2002'),
 partition dec2002 values ('December_2002'),
 partition jan2003 values ('January_2003'),
 partition feb2003 values ('Feburary_2003'),
 partition mar2003 values ('March_2003'),
 partition rest values (default)
)
/
```

Figure 5.1. Create table sourceforge

Note that in the table, the combination of PROJID, USERID and MONTHID form the primary key, as any subset of them is not able to form a primary key. For each month afterwards, new data will be gathered and loaded into the data warehouse. With table partitioning, this process is a 4-step process:

1. Add a new partition to SOURCEFORGE:

```
alter table sourceforge
     add partition apr2003 values ('April_2003')
/
```

Figure 5.2. Add a new partition to SOURCEFORGE

2. Create a new table NEWDATA which has the same columns as SOURCEFORGE:

```
create table newdata as
     select * from sourceforge where 1=2
/
```

Figure 5.3. Create a new table NEWDATA

3. Load the new data to the table NEWDATA: There is no difference between loading data to NEWDATA and loading data to SOURCEFORGE. We'll explain how to load data from data files to the database tables later.

4. Exchange the table NEWDATA with the placeholder partition created in step 1:

There may be some other procedures that must be done in the 4-step process, such as constraints creation and validation, indexes creation, etc. For now, we'll skip these steps here for simplicity.

```
alter table sourceforge
     exchange partition apr2003 with table newdata
/
```

Figure 5.4. Exchange partition

5.1.2 Using the sqlldr Utility to Load Data in Data File to the Table

Data gathered by the shell scripts is in a nice format: the data fields are separated by ”|” (only a small percentage of data is not in this format, and we have to discard them because we don’t know how to delimit them and put them into the data warehouse in a meaningful way), and charater fields are enclosed by quotes. A column MONTHID is added to each record in the datafile. The record is then loaded into the table SOURCEFORGE. Shell scripts for finishing the loading process are provided in the appendix section.

5.1.3 Dealing With Wrong Data

Some records are not loaded to the database successfully, because their formats are not compatible with the majority of other records. These errors may be due to mistakes in the shell scripts which gather data. Figure 5.5 shows a snapshot of data which is not loaded successfully.

Dealing with wrong data is part of a large topic called data cleansing. There has been some work done for this issue. Data cleansing includes duplicates detection and removing. Some previously developed data cleansing method can be applied here to provide better data quality for data mining.

Changing wrong records to correct ones is often a time-consuming process. Fortunately, in our case, the wrong records can be changed by modifying ””” to ”|”, after this, these records can be inserted into the database. Only 0.4 percent of data is in wrong format, it’s not too hard to change all the wrong data. Even after

```
308|"Christophe Prud'homme" 'prudhomm|All-Hands Person|prudhomm
310|"Theodore Ts'o" 'tytso|All-Hands Person|tytso
429|"Paul `Rusty' Russell" 'rusty||rusty
614|"Ian O'Neal" 'chico|All-Hands Person|chico
614|"Pablo d'Angelo" 'dangelo||dangelo
965|"O'ksi'D" 'nickasil||nickasil
10280|"Giuseppe Paterno'" 'gpaterno||gpaterno
10536|"Keith O'Hara" 'kohara||kohara
10854|"San'Pee" 'sanpee||sanpee
10857|"Christophe L'HARIDON" 'clharidon|Developer|clharidon
10944|"T'aZ" 'taz-007||taz-007
11050|"A'rpi" 'arpi_esp|Developer|arpi_esp
11073|"Kevin O'Rourke" 'kevin_i_orourke||kevin_i_orourke
11136|"Ian O'Neal" 'chico||chico
11295|"James O'Neill" 'yackass|No specific role|yackass
11392|"John I'anson-Holton" 'jianson||jianson
11778|"Tim O'Brien" 'mild7||mild7
11993|"jake's admin account" 'udforguk|Project Manager|udforguk
12078|"Paul `Rusty' Russell" 'rusty||rusty
12104|"Tadhg O'Meara" 'tomeara||tomeara
12282|"Marc L'Heureux" 'mlheur||mlheur
12377|"San'Pee" 'sanpee||sanpee
```

Figure 5.5. Wrong data

changing the format of error data, there is still some data (abount 0.01 percent) which makes no sense. In this case, this kind of data is deleted from the table.

Once the data is cleansed, it can be analyzed using a statistical approach or data mining.

5.2 Statistics and data mining

We'll dicuss the power law phenomenon, then analyze the acquisition and churn processes of projects and developers. The results of the analysis can be used to guide the simulation design, for example, choose appropriate parameters derived from statistics, or add new features found by data mining.

5.2.1 Power law

We'll mention the work in [24], where the authors pointed out that the size x of projects and the count of projects with size x obey the power law. Based on the table SOURCEFORGE, we create some view

LOGPROJSIZE_LOGCOUNTS and use the following query to obtain the actual linear regression for each month's data.

```
select monthid, regr_slope(logcounts, logprojsize) log_regr_slope,
       regr_intercept(logcounts, logprojsize) log_regr_intercept,
       regr_r2(logcounts, logprojsize) log_regr_r2
  from logprojsize_logcounts
 group by monthid
/
```

Figure 5.6. Query for regression of the log-log data

The following table shows the result of the above query.

```
MONTHID              LOG_REGR_SLOPE LOG_REGR_INTERCEPT LOG_REGR_R2
-------------------- -------------- ------------------ -----------
April_2002               -2.6013312         15.1110083  .946789843
August_2002              -2.6764609         15.7115279  .953628133
December_2002            -2.6600011         15.8051842  .964057934
Feburary_2002            -2.5254713         14.7314841  .928144164
Feburary_2003            -2.6058242          15.828709  .972926868
January_2003             -2.7006866         15.9981189  .966991411
June_2002                -2.5594851         15.1882059  .950524549
March_2002               -2.5170802         14.7575338  .942937747
March_2003                -2.652212          15.982759  .963745005
May_2002                 -2.6090457         15.1906168  .946064727
November_2002            -2.6476532         15.7898782  .962124207
October_2002             -2.6465484         15.6951324  .960079213
September_2002           -2.6652868         15.7173293  .959730213
```

Figure 5.7. The power law

From the data of the above table, we see that the log-log (base 2) plot of project size x and the count of projects with size x fit a regression line very well. The fourth column of the above table is the R-squared of the linear regression, also called the coefficient of determination or goodness of fit. A value greater than 0.9 for R-squared is considered as a very good fit. From the table, we also see that the linear regression for each month data has almost the same regression slope and regression intercept. Let x denote the size of projects, y denote the count of projects with size x, then we have the following formula:

$$y = 46341x^{-2.62}$$

As time passes by, new projects and developers may join, while old projects and developers may churn, either because projects have been finished, projects have failed, or developers lose interest in the projects. Using the history data we gathered from the sourceforge website, we can analyze the aquisition and churn behavior of projects and developers. We'll discuss the acqusition analysis and churn analysis in the following subsections.

5.2.2 Retention and churn analysis

The percentages of projects retained after months passed by are called retentions. With the help of the following two PL/SQL procedures as in Figure5.8, we can compute retention of projects.

```
::::::::::::::
pop_proj_retention.sql
::::::::::::::
create or replace procedure pop_proj_retention as
total_months number;
start_month varchar2(20);
following_month varchar2(20);
begin
  select max(monthid) into total_months from month;
  for i in 1..total_months loop
    for j in i..total_months loop
      select monthname into start_month from month where monthid=i;
      select monthname into following_month from month where monthid=j;
      proc_proj_ret(start_month, following_month);
    end loop;
  end loop;
end;
/
::::::::::::::
proc_proj_ret.sql
::::::::::::::
create or replace procedure proc_proj_ret(start_month varchar2,
following_month varchar2)
as
total number;
churn number;
begin
  select count(distinct projid) into total
    from sourceforge where monthid=start_month;
  select count(distinct projid) into churn
    from (select distinct projid from sourceforge where monthid=start_month
          minus
         select projid from sourceforge where monthid=following_month);
  insert /*+append*/ into proj_retention values (start_month,
    following_month, (total-churn)/total);
  commit;
end;
/
```

Figure 5.8. The PL/SQL procedures to compute projects retention

The above procedures will compute retention of projects for each month from Feburary 2002 through March 2003. For example, let's pick August 2002. After one month, the retention might be 0.99, after two month, the retention might be 0.97, etc. After sliding the start month together, and taking average for the retentions, we can get the retention of projects after 1 month, after 2 month, etc.

The following table lists the retention values after months. We do not list retention values after 7 months since they are changing even more slowly and we may assume that a project survives 6 months will stay forever! (This might not be true, but the data we gathered showed this trend. This can also be explained that if a project has been under development for 6 or more months, it would be a good project and will not fail.)

Table 5.1. Retention of projects

Months Later	Retention
0	1
1	.994
2	.992
3	.991
4	.990
5	.989
6	.987

Our data also shows that the churned projects are all small projects in the sense that there is almost only one developer for each churned project. Next we'll investigate the churn behavior of developers. More precisely, we are going to compute the churn rate of developers and distribution of sizes of projects from where the developers churn.

The following table shows the retention rate of developers after each number of months. This result is obtained using PL/SQL.

Only 9 months of retentions are listed in the above table. The above table can be interpreted as the following: 1.3 percent of developers churn in the first month, 0.9 percent churn in the second month, etc. (Note: the base is the number of developers

Table 5.2. Retention of developers

Months Later	Retention
0	1
1	.987
2	.978
3	.970
4	.963
5	.957
6	.950
7	.943
8	.933
9	.925

in the starting month.)

The retention value does not change after 10 months. So we'll assume that developers will not leave a project after 9 months. This might be a reasonable assumption if we believe that the developer has been one of the core developers in the project, and thus he gains some reputation in the project's community and will not leave the project.

Next, we compute the distribution of sizes of projects from where the developers churn. With the help of the following two procedures in Figure 5.9, the actual churn distribution of project size and count of projects can be computed.

Surprisingly, it turns out that the churn distribution obeys the power law also, as illustrated in the following table. The R-squared is not satisfactory, but it can be improved after removing outliers.

The churn analysis of projects and developers can be helpful to the design of OSS simulation. For example, at each iteration, we can specify the probabilities of projects churn and developers churn. Another important issue that will be addressed here is using data mining to build decision tree or naive bayes classification models to predict churn of projects and developers based on the starting month, size of

```
::::::::::::::
all_churn.sql
::::::::::::::
create or replace procedure pop_proj_retention as
total_months number;
start_month varchar2(20);
following_month varchar2(20);
begin
  select max(monthid) into total_months from month;
  for i in 1..total_months loop
    for j in i..total_months loop
      select monthname into start_month from month where monthid=i;
      select monthname into following_month from month where monthid=j;
      dev_churn_dist(start_month, following_month);
    end loop;
  end loop;
end;
/
::::::::::::::
churn_dist.sql
::::::::::::::
create or replace procedure dev_churn_dist(s varchar2,
 f varchar2) as
begin
insert into churn_dist (projsize, counts)
select projsize, count(projsize)
  from (select a.userid, a.projid, b.projsize
          from upsm b,
               (select userid, projid from upsm
                 where monthid=s
                minus
                select userid, projid from upsm
                 where monthid=f
               ) a
        where a.userid=b.userid and a.projid=b.projid
          and b.monthid=s
       )
 group by projsize;
update churn_dist set start_month=s where start_month is null;
update churn_dist set following_month=f where following_month is null;
commit;
end;
/
```

Figure 5.9. The PL/SQL procedures to compute developers churn distribution

Table 5.3. Power law distribution of churned developers

Months Later	Log Regr Slope	Log Regr Intercept	Log Regr R2
1	-.469	5.487	.632
2	-.466	5.826	.685
3	-.449	5.915	.708
4	-.429	5.943	.702
5	-.421	5.985	.694
6	-.413	5.993	.695
7	-.421	6.108	.704
8	-.423	6.192	.710
9	-.406	6.135	.682

projects, number of projects, number of developers, and role of developers, etc. We propose to examine this part. Next, we'll analyze the acquisition process.

5.2.3 Acquisition analysis

When we do the acquisition analysis, we might have the following questions in mind:

- what's the increase rate of projects?
- what's the project size distribution of growing projects?
- what's the distribution of new developers and existing developers for growing projects?

Or in other words, we want to know how developers choose to join projects. For example, do developers prefer to join larger projects rather than smaller projects? After addressing all these questions, we might start to ask questions such as how to create data mining models to predict what developers will start new projects and what developers will join existing projects based on the attributes such as size of projects, number of projects involved, etc.

For new developers, the distribution of projsize and count of project forms the power law. At this point, there is not enough data to support the power law distribution because the data collection interval is two large, which is one month. We suspect the correlation of the projsize and count of projects should be linear, i.e., it's the preferential attachment. Thus the power law distribution described in the previous section can be deduced from this property. More data will be gathered at a more tight interval to better evaluate this conjecture.

5.2.4 Build classification models to predict churn and acquisition

With the help of Oracle Data Mining and DM4J (Data Mining for Java), we can build a series of models to predict churn and acquisition based on the properties of projects and developers. At this moment, not enough data is available to build these models. We plan to gather more data from sourceforge.net to build these models.

The properties of projects include:

- development status: 1=planning, 2=pre-alpha, 3=alpha, 4=beta, 5=production/stable, 6=mature, 7=inactive
- enrironment: 1=cocoa (MaxOS X), 2=console (text based), 3=handhelds/PDA's, 4=no input/output (daemon), 4=other environment, 5=web environment, 6=win32 (MS windows), 7=X11 applications
- intended audience: 1=customer service, 2=developers, 3=education, 4=end user/desktop, 5=financial and insurance industry, 6=healthcare industry, 7=information technology, 8=legal industry, 9=manufacturing, 10=other audience, 11=religion, 12=science/research, 13=system administrators, 14=telecommunications industry
- license: 1=OSI approved, 2=other/proprietary license, 3=public domain
- natural language: 1=english, 2=others
- operating system: 1=BeOS, 2=MacOS, 3=Microsoft, 4=OS independent, 5=OS/2, 6=other OS, 7=PDA systems, 8=POSIX
- programming language:
- topic: 1=communications, 2=database, 3=desktop environment, 4=education, 5=games/entertainment, 6=internet, 7=multimedia, 8=office/business,

  9=other/nonlisted topic, 10=printing, 11=religion, 12=scientific/engineering, 13=security, 14=sociology, 15=software development, 16=system, 17=terminals, 18=text editors

- project name
- register date
- activity percentile
- bugs
- support requests
- patches
- feature requests
- mailing lists
- CVS repository commits
- messages in public forums
- number of members

Not all these attributes are important to build models. We have to select important attributes from the property list. The feature selection methods will be employed here to select only the most relevant attributes. The acquisition and churn prediction models will be built use the data collected from sourceforge. Once the models are built, they can be deployed and provide useful information to guide the simulation design.

CHAPTER 6

DATA MINING APPLICATION (2): NOM

As far as we know, data mining technologies have never been applied to the NOM research area. In this chapter, I'll investigate how data mining can impact the research of NOM. This chapter is divided into several sections:

- Background information
- Issues to be addressed
- How data mining can help

6.1 Background information

Natural organic matter (NOM) is a heterogeneous mixture of organic molecules found in terrestrial and aquatic environments. It plays a vital role in ecological and biogeochemical processes. No current models of NOM production and evolution describe both quantitative aspects of organic carbon transfer and qualitative aspects of NOM structural and functional heterogeneity.

An agent-based stochastic model for the evolution of NOM was proposed and developed recently. In this approach, NOM is treated not as a single organic matter carbon entity, but as a large number of discrete molecules with varying chemical and physical properties. Prior to this agent-based stochastic model, previous models are either too simplistic (for example, the carbon cycling models) to represent the heterogeneous structure of NOM and its complex behavior in the environment, or too complex (for example, the connectivity maps models) and computational intensive to be useful for large-scale environmental simulation simulation.

The agent-based stochastic approach records all the intermediate steps of the simulation in databases. The tremendous amount of data needs to be analyzed carefully to help address some issues in the area of NOM research. Here data mining should be quite helpful to discover interesting patterns from the huge datasets.

6.2 Issues related to NOM

The scientific goal of the stochastic approach is to produce both a new methodology and a specific program for predicting the properties of NOM over time as it evolves from precursor molecules to eventual mineralization. As scientists experiment and analyze NOM, they raised problems such as

1. Do most NOM molecules have similar carbon 'skeletons', differing principally in specific functional groups and average size?

2. Do the carbon 'skeletons' differ greatly according to the precursor material, with similar collection of functional groups imparting similar reactivity?

3. From what precursors and by what biochemical pathways is NOM formed?

4. How is NOM linked to microbial and chemical processes in the environment?

After understanding both the structural heterogeneity and the evolution of NOM, we can reasonably predict the outcomes of environmental processes in which NOM plays a key role.

6.3 How data mining can help

Data mining might not be able to answer the above questions directly. But the models built using data mining can be quite helpful to the scientists to make reasonable assumptions, and thus to guide experiment design, simulation model design and so on.

Using models built by data mining, we can predict the molecules' behavior; we can make molecule segments in which molecules behave similarly; we can find the correlation of functional groups and molecule behavior.

These models can be very valuable to understand the NOM evolution.

6.3.1 Data source

Data for the NOM data mining can be from both experiments and simulations. Experiment data includes elemental composition by combustion analysis, molecular weights, acidity, rate of CO_2 release from lab experiments, Rate of N release from lab experiments, lability by microbial uptake measurements, etc. This kind of data is generally in text file format and can be loaded into the database for further processing.

Simulation data is obtained by running the simulations against various sets of configurations. These data is stored in the database and can be combined with the experimental data after necessary transformation.

6.3.2 Data mining attack of the above questions

Let's have a close look at the questions listed above.

1. Question 1 can be handled using clustering algorithms to build clusters based on function groups and average molecular weights. We can also build classification models with functional groups and molecular weights as predictors and

to predict carbon skeletons.

2. Question 2 and 3 can be attacked using with classification models also.

3. Question 4 should be able to solved using association rules.

Of course, multiple algorithms can be combined to thoroughly attack the listed problems. We expect to see more variations of the problem during our study of the above problems.

CHAPTER 7

PROPOSED WORK AND TIME FRAME

In this chapter, we summarize the proposed work and provide the time frame to complete. This proposal will in part extend and refine some work completed in the theses [32] [65].

7.1 Proposed work

We propose to build an integrated multi-tier infrastructure to support scientific simulations and data analysis. It integrates the Oracle9i application server, Oracle databases federating heterogeneous scientific data for analysis and reports, data cleansing, data warehousing and data mining, and the Swarm/RePast simulation library. Part of the motivation for this infrastructure is the lack of details to implement such systems to support large scale scientific simulations.

To make the infrastructure scalable and reliable, we implement such features as load-balancing and simulation-resuming using J2EE technologies with new algorithms.

Data collected from the Web and experiments is often dirty. In particular, approximate duplicates may exist in the combined data. We designed two algorithms to cleanse the data. We plan to implement these algorithms using JDBC or PL/SQL. Then we'll test these algorithms against real world data and compare their performance with some known data cleansing algorithms.

Part of the proposed work is to support scientific simulations in the fields of social and environmental science. Two simulation programs are developed or under developing, namely, the NOM project and the OSS project. The two projects are similar in that they are developed using the agent-based technology with Swarm. To understand and monitor the behavior of agents, a large amount of data is generated by the simulations and stored in the database for analysis. The data will be combined with real world data collected from Web and experiments for data mining.

7.2 Time Frame

Part of the proposed work has been done in the past few months. We plan to finish all the work in 12 more months. An estimated timeframe for the different stages of the research is provided in Figure 7.1

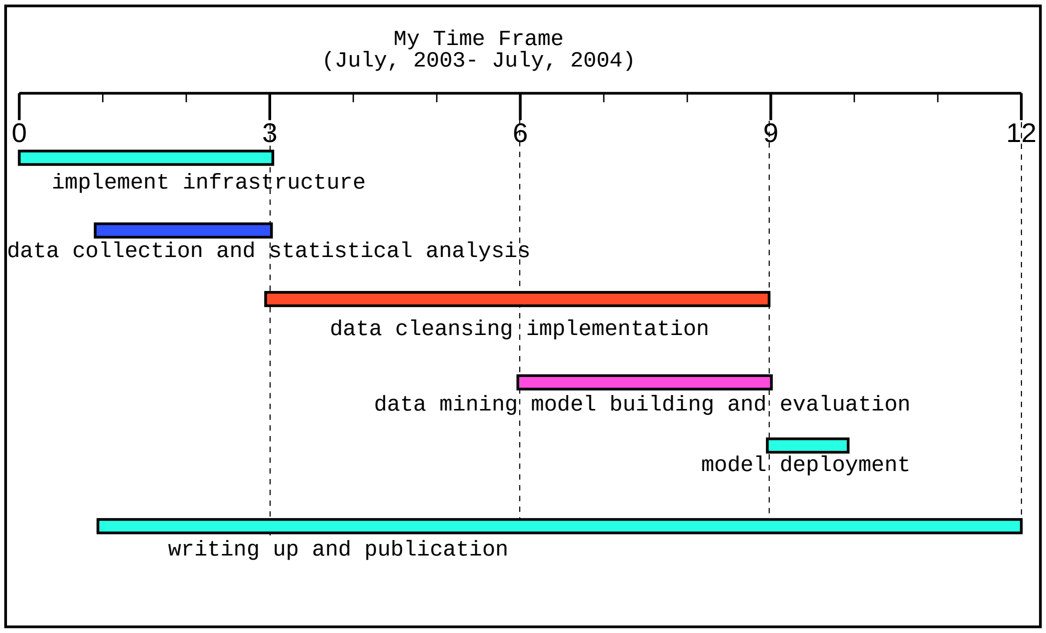


Figure 7.1. Timeframe

7.2.1 Implementing infrastructure

The multi-tier infrastructure will be implemented in the NOM computer cluster. Features such as load-balancing and simulation-resuming will be implemented using J2EE and Oracle with new algorithms. Existing collaboration utilities such as BBS, chatroom, file uploading and simulation sharing will be integrated also.

7.2.2 Implementing data cleansing algorithms

The two data cleansing algorithms proposed will be implemented using JDBC or PL/SQL. To compare our algorithms with other known ones, we gathered data which is available from papers which describes the known algorithms. PL/SQL is preferred because it operates inside the database and reduces network traffic. The disadvantage is that the data must be loaded into an Oracle database before data cleansing can be performed.

7.2.3 Data collection

It's expected data can be retrieved from sourceforge.net for the OSS project. NOM experiment data can be obtained from our collaborating scientists. Simulation data for both simulations can be obtained by running a sufficient number of simulations.

7.2.4 Data analysis

Data analysis will be carried out on the OSS real data from sourceforge.net and on the NOM experimental data from scientists. We seek to prepare data for data mining. This preparation includes data cleansing, data binning (discretization), split transformation, and data statistics. These statistics will be compared with data generated from simulations. Simulations might need to be modified to conform real world data.

7.2.5 Data mining and model-building

Issues to be addressed include

- clustering of projects and developers for OSS
- classification models to predict developers and projects behavior for OSS and molecular behavior for NOM (such as churn and acquisition behavior of developers and projects, adsorption process of NOM)

7.2.6 Model evaluation and deployment

Models will be tested against new real world data. Once tested, lifts will be computed to verify the value of the models, and models will be deployed as stored java procedures in the Oracle database. Simulations can call these procedures when making decisions on agents behavior.

7.2.7 Writing up dissertation

Chapters of the dissertation will be written as the research progresses, with the final three months reserved for the final draft.

7.2.8 Expected publication

- 3 months: Implementation details of the infrastructure by August 2003
- 6 months: Data cleansing for scientific simulation data
- 9 months: Data mining applications for NOM
- 10 months: Data mining applications for OSS

7.2.9 Expected results and impact

- A scalable and reliable infrastructure to support scientific simulations

- Data cleansing algorithms for better data quality
- Attempt of data mining applications in the fields of social and environmental sciences.
- Better understanding of OSS phenomenon and NOM molecules with the help of data analysis and data mining.

BIBLIOGRAPHY

[1] R.H. Arpaci A.C. Dusseau and D.E. Culler. Effective distributed scheduling of parallel workloads. In *Proceedings of ACM SIGMETRICS*, pages 25–36, 1996.

[2] A. Apostolico and C. Guerra. The longest common subsequence problem revisited. In *Algorithmica*, pages 315–336, 1987.

[3] A. Thakar J. Gray D. R. Slutz A.S. Szalay, P.Z. Kunszt. Designing and mining multi-terabyte astronomy archives: The sloan digital sky survey. In *SIGMOD*, pages 451–462, 2000.

[4] A. Thakar J. Gray T. Malik J. Raddick C. Stoughton J. vandenBerh A.S. Szalay, P.Z. Kunszt. The sdss skyserver: public access to the sloan digital sky server data. In *SIGMOD*, pages 571–581, 2002.

[5] S.A. Banawan and J. Zaborjan. Load sharing in heterogeneous queueing systems. In *Proceedings IEEE INFOCOM*, pages 731–739, 1989.

[6] M.J.A. Berry and G.S. Linoff. *Mastering Data Mining.* John Wiley and Sons, Inc, 2000.

[7] J. Bourgain. On lipschitz embedding of finite metric spaces in hilbert space. In *Israel Journal of Mathematics, v52*, pages 46–52, 1985.

[8] R.J. Branchman and T. Anand. The process of knowledge discovery in databases: A human-centered approach. In *Advances in Knowledge Discovery and Data Mining*, pages 97–158, 1996.

[9] Venter C. The sequence of the human genone. In *Science*, 2001.

[10] A. Campos and D. Hill. Web-based simulation of agent behavior. In *Proceedings of the International Conference on Web-based Modeling and Simulations*, pages 9–14, 1998.

[11] H. Mannila D. Hand and P. Smyth. *Principles of Data Mining.* MIT Press, 2001.

[12] DataCleanser. *http://www.npsa.com.* Last accessed, 2002.

[13] DataFlux. *http://www.dataflux.com.*

[14] Diamond. *http://www-aig.jpl.nasa.gov/public/mls/diamond-eye.*

[15] P. Dinda. Online prediction of the running time of tasks. In *Cluster Computing, 5(3)*, 2002.

[16] E.D. Lazowska D.L. Eager and J. Zahorjan. The limited performance benefits of migrating active processes for load sharing. In *Proceedings of ACM SIGMETRICS*, pages 63–92, 1998.

[17] G. Dodge and T. Gorman. *Essential Oracle8i Data Warehousing.* John Wiley and Sons, New York, 2000.

[18] C. Faloutsos and K. Lin. Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. In *Proceedings SIGMOD*, pages 163–174, 1995.

[19] P. Smyth M. Burl Fayyad, U. and P. Perona. A learning approach to object recognition: applications in science image analysis. In *Early Visual Learning, Oxford University Press*, 1996.

[20] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery: An overview. In *Advances in Knowledge Discovery and Data Mining*, pages 1–36, 1996.

[21] Trillium Software System for Data Warehousing and ERP. *http://www.trilliumsoft.com/products.html.* Last accessed, 2002.

[22] L. Gasieniek D. Gunopulos G. Das, R. Fleisher and J. Karkamen. *Episode matching.* Springer-Verlag, 1997.

[23] V. Freeh G. Madey and R. Tynan. Agent-based modeling of open source using swarm. In *AMCIS2002*, 2002.

[24] V. Freeh G. Madey and R. Tynan. The open source software development phenomenon: an analysis on social network theory. In *Eighth Americas Conference on Information Systems*, 2002.

[25] J. Han and M. Kamber. *Data Mining Concepts and Techniques.* Morgan Kaufmann Publishers, 2001.

[26] M. Harchol-Balter and A.B. Downey. Exploiting process lifetime distributions for dynamic load balancing. In *Proceedings of ACM SIGMETRICS*, pages 13–24, 1996.

[27] M. Hernandez and S. Stolfo. The merge/purge problem for large databases. In *Proceedings of the 1995 ACM-SIGMOD*, 1995.

[28] M. Hernandez and S. Stolfo. Real-world data is dirty: Data cleansing and the merge/purge problem. In *Data Mining and Knowledge Discovery 2(1)*, pages 9–37, 1998.

[29] G.R. Hjaltason and H. Samet. Incremental distance join algorithms for spatial databases. In *Proceedings SIGMOD*, pages 137–248, 1998.

[30] G. Hristescu and M. Farach-Colton. Cluster-preserving embeddings of proteins. In *Technical Report 99-50, Rutgers Univ.*, 1999.

[31] Y. Huang. Data cleansing: A sample database approach. In *Technical Report*, 2002.

[32] Y. Huang. Infrastructure, query optimization, data warehousing and data mining in support of scientific simulations. 2002.

[33] Innovative Systems Inc. *http://www.innovativesystems.net*. Last accessed, 2002.

[34] W. Inmon. *Build the data warehouse*. John Wiley and Sons, New York, 1996.

[35] P. Spencer J. Long and R. Springmeyer. Simtracker - using the web to track computer simulation results. In *Proceedings of the International Conference on Web-based Modeling and Simulations*, pages 171–176, 1999.

[36] J2EE. *http://java.sun.com/j2ee*.

[37] JarTool. *http://www-aig.jpl.nasa.gov/public/mls/mgnsar/jartool-home.html*.

[38] V. Kierman. Sophisticated software is reshaping the way scientists use statistics. In *The Chronicle of Higher Education, Information Technology*, 1999.

[39] R. Kimball. *The Data Warehouse Toolkit*. John Wiley and Sons, New York, 1998.

[40] C. Li L. Jin and S. Mehrotra. Efficient record linkage in large data sets. In *Proceedings 8th International Conference on Database Systems for Advanced Applications*, 2003.

[41] W.E. Leland and T.J. Ott. Load balancing heuristics and process behavior. In *Proceedings of ACM SIGMETRICS*, pages 54–69, 1986.

[42] P. Smyth U. Fayyad P. Perona L. Crumpler M. Burl, L. Asker and J. Aubele. Learning to recoginze volcanoes on venus. In *Machine Learning*, pages 165–195, 1998.

[43] et. al. M. Lee, T. Ling. Cleansing data for mining and warehousing. In *Proceedings DEXA*, pages 751–760, 1999.

[44] D. Scales M. Rinard and M. Lam. Jade: A high-level machine-independent language for parallel computing. In *IEEE Computer, 26(6)*, pages 28–38, 1993.

[45] J.I. Maletic and A. Marcus. Data cleansing: Beyond integrity checking. In *Proceedings of Information Quality (IQ)*, 2000.

[46] A. Maydanchik. Challenges of efficient data cleansing. In *DM Review*, 1999.

[47] A. Monge and C. Elkan. An efficient domain-independent algorithm for detecting approximate duplicate database records. In *Proceedings SIGMOD Workshop on Research Issues on DMKD*, pages 23–29, 1997.

[48] L. Moss. Data cleansing: A dichotomy of data warehousing. In *DM Review*, 1998.

[49] S. Muench. *Building Oracle XML Applications*. O'Reilly, 2000.

[50] M.W. Mutka and M. Livny. The available capacity of a privately owned workstation environment. In *Performance Evaluation 12(4)*, pages 269–284, 1991.

[51] S. Needleman and C. Wuncsh. A general method applicable to the search for similarities in the amino acid sequences of two proteins. In *Journal of Molecular Biology*, pages 444–453, 1970.

[52] Oracle. *http://www.oracle.com*.

[53] RePast. *http://repast.sourceforge.net*.

[54] et. al. S. Allamaraju. *Professional Java Server Programming J2EE, 1.3 Edition*. Wrox Press Inc., 2001.

[55] D. Sankoff and J. Krunskal. *Time Warps, String Edits and Macromolecules: The Theory and Practive of Sequence Comparison*. Addison-Wesley, 1983.

[56] Sapphire. *http://www.llnl.gov/casc/sapphire*.

[57] SARS. *http://www.ehealth.org*.

[58] B. Siegell and P. Steenkiste. Automatic generation of parallel programs with dynamic load balancing. In *Proceedings of the Third International Symposium on High-Performance Distributed Computing*, pages 166–175, 1994.

[59] F. Simoudis, B. Levesey, and R. Kerber. Using recon for data cleansing. In *Proceedings KDD*, pages 282–287, 1995.

[60] Qualitative Marketing Software. *http://www.qrmsoft.com*. Last accessed, 2002.

[61] Swarm. *http://www.swarm.org*.

[62] Vality Technology. *http://www.vality.com*. Last accessed, 2002.

[63] E. Ukkonen. Algorithms for approximate string matching. In *Information and Control*, pages 100–118, 1985.

[64] W. Winston. Optimality of the shortest line discipline. In *SIAM J. Appl. Prob., 14:181-189*, 1977.

[65] X. Xiang. Agent-based scientific applications and collaboration using java. 2003.

[66] XML. *http://www.w3.org/XML*.