This is to certify that the
thesis entitled

A Distributed Approach to Managing
Large Simulation Data Sets

presented by

Brian D. Connelly

has been accepted towards fulfillment
of the requirements for the

M.S. degree in Computer Science and Engineering

Major Professor's Signature

5/11/2005

Date

A DISTRIBUTED APPROACH TO MANAGING LARGE SIMULATION DATA SETS

By

Brian D. Connelly

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Department of Computer Science and Engineering

2005

ABSTRACT

A DISTRIBUTED APPROACH TO
MANAGING LARGE SIMULATION
DATA SETS

By

Brian D. Connelly

SimDB is a project to develop a distributed system that stores and analyzes large simulation data sets from molecular dynamics simulations. Using the resources of all the machines on the network, it is able to store a large amount of simulation data (trajectories), so that interested groups may work with these data. Because it would take a long time to transmit these data over the Internet, users instead specify how they would like to analyze the data using a rich set of analysis functions. The analysis is then done online, and the resulting data are sent to the user. This data requires considerably less storage space than the raw data, so transferring it requires less time. SimDB also stores these analysis results, thereby offering researchers the further benefit of accessibility: Stored analysis results may be used to satisfy future queries without the necessity of re-analyzing the data.

ACKNOWLEDGMENTS

This research was made possible by a grant from the Quantitative Biology and Modeling Initiative at Michigan State University.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Obtaining an accurate representation of a biomolecule's structure and dynamics can be a very challenging problem; nevertheless such information is crucial for understanding that structure's biological function in detail. Traditional experimental techniques such as X-ray crystallography and nuclear magnetic resonance (NMR) spectroscopy are good methods for determining structure, but because these methods use a large number of structures and arrive at results using the average over all of the structures in the experiment, obtaining dynamics information at the atomic- or molecular level is difficult [24]. Molecular dynamics (MD) simulations [23] [28] [2] are one technique which can be used to effectively supplement these experimental structures by providing dynamics information at the atomic level of detail. Starting from experimental structures, the computer can use complex models which create an accurate view of how that biomolecule behave dynamically.

Because the function of a given biomolecule has been linked not only to its structure, but also to its dynamics [32], obtaining dynamics information in addition to structural knowledge is crucial. Conformational changes that occur during biological cycles can be tightly coupled with regulatory processes such as allowing or disallowing the binding of a ligand to a protein or allowing a molecule to pass through a cell wall.

Smaller conformational changes and specific dynamic features also play an important role with respect to catalytic activity [8].

The Simulation Database (SimDB) is a project which aims to use the power of distributed computing to provide many unique resources to those who perform research using MD simulations. Analogous to the Protein Data Bank (PDB) [5], which is further introduced in Section 2.1, SimDB is designed to allow for the storage of biomolecular data so that researchers may share their results with the rest of the scientific community. This is beneficial for a number of reasons. The first is that MD simulations can be quite time- and resource-consuming, with some large simulations taking weeks or months, even on large clusters of computers. In order to review or examine different aspects of another's work, one typically must first set-up and re-run the simulation, which incurs the same time penalty. Also, because running molecular dynamics simulations can be such a resource-intensive job, many groups that may be interested in using simulations simply do not have the resources at their disposal or the know-how. Since SimDB makes these data sets available online, those interested in working with them will have convenient access to do so.

Secondly, because simulation data can be extremely large, on the order of 10s or 100s of gigabytes in size, it is often the case that researchers cannot easily send the data to others using traditional network services. This means that after a simulation has been run, the data are usually either deleted or put on tape, where they are unavailable or not easily accessible to others who may wish to make use of them. Although simulations are often run in order to meet one specific objective, the data generated is potentially useful to many other researchers for many different reasons, so the ability to share them is very beneficial. Instead of requiring interested users to download entire datasets, with SimDB, the data is analyzed online, and the results are transmitted to the user. These results can require several orders of magnitude less disk space, so their transfer is a much faster process.

The second main goal of SimDB is to provide a rich set of advanced trajectory analysis functions that can be used to obtain detailed information about the simulation data stored in the system. This includes such general functions as radius of gyration and coordinate root mean square deviation (RMSD) calculations, as well as structure analyses, such as protein secondary structure and backbone torsion. Additional functions will also be included to support many common- and advanced analyses, offering a unique service that can satisfy a wide variety of research interests. In addition to analysis, users of SimDB may also perform scoring and clustering of conformations. A framework will also be provided that allows users to submit custom analyses as well.

Not only does SimDB store raw trajectory files which result from simulations, but it has the capacity to store results from analyses as well. Because of this caching of analyzed data, results to queries for which similar queries have been previously executed can skip the analysis step, which saves considerable time.

This thesis discusses the Simulation Database in detail. Because of the resource requirements for the system, it is being developed as a distributed system in order to harness the resources of many machines. Chapter 2 introduces the use of databases in the natural sciences and provides some examples of databases currently in use. Chapter 3 offers a brief introduction to molecular dynamics simulations and the data that are produced by such simulations. The architecture of SimDB is outlined in Chapter 4, the current status of the system is detailed in Chapter 5, and Chapter 6 focuses on some directions that the system will be heading in the future. Finally, some general conclusions are drawn in Chapter 7.

Chapter 2

Related Work

The use of databases in biology, physics, chemistry, and other areas of the natural sciences has become very common in recent years. Advances in techniques have resulted in an enormous amount of data being generated, necessitating their use. Often, these databases are made public, which benefits the communities tremendously, as now researchers can start work in areas for which data already exists, thereby bypassing the need to re-run previously-conducted experiments.

Most existing databases in this area serve only two purposes: the storage and retrieval of data. Analysis options are not available in most databases, although some provide links to existing web services, where analysis can be completed.. Online processing of the data is a unique characteristic of SimDB and one of the reasons that its implementation is a more complex undertaking. This is not to say that all such databases should support processing of the data. In most cases, this simply is not necessary, as the data available are not very large, and processing is not nearly as computationally-expensive and time-consuming as with molecular dynamics. Additionally, the data stored in some of these databases is used as-is, so processing simply is not necessary.

This chapter introduces a few databases that are currently in use in the natural

sciences. Although some of them share very little similarity with SimDB, they are interesting to examine, because they offer an insight into how research is done in similar fields, and how the use of databases helps facilitate these research efforts.

2.1 The Protein Data Bank

The Research Collaboratory for Structural Bioinformatics (RCSB) Protein Data Bank (PDB) is a large repository for experimental 3-dimensional biological structural data from large molecules, including proteins and nucleic acids [5]. It was established in 1971 to store crystallographic structures. Since then, it has grown dramatically through the use of modern techniques such as nuclear magnetic resonance (NMR) spectroscopy. Structural data published in journals is almost always made available in the PDB. Currently, the PDB receives more than 50 deposits per week and stores information for over 30,000 structures, and is extremely useful to researchers in many fields such as chemistry, biology, and physics. Aside from serving as a repository for structures, it also maintains a common format for defining those structures.

The main difference between SimDB and the PDB is that the PDB only stores static structural information, while SimDB aims to store dynamic properties of some of these structures. Although much can be done with static structures, they cannot describe the behavior of a given structure unless many structures are examined over time, which is essentially the result of molecular dynamics simulations. It is also important to note that the structures in the PDB are all acquired through traditional experimentation, whereas the simulation data stored in SimDB are secondary data that have been obtained through simulation.

The structural information for the available structures requires roughly 20 gigabytes of disk storage [36]. As this can be accommodated easily by current hard disks, groups can download it and establish local mirrors. Additional data related to the

structures increases the storage requirements to around 1 terabyte. As the PDB is not a distributed repository, this large data size prohibits the number of nodes at which the full archive can be stored, although the rate at which the system grows does not require more sophisticated approaches such as those employed by SimDB.

SimDB aims to become a tool that is as useful or more useful to researchers who run simulations as the PDB, and perhaps as ubiquitous. Due to the fact that a PDB entry may be on the order of 10s or 100s of kilobytes, and data from molecular dynamics simulations can be in the 10s or 100s of gigabytes, SimDB will require considerably more storage space. Also, since the PDB does not do any processing on its data, it does not have to deal with this aspect as well.

2.2 GenBank

GenBank is a database which stores genetic sequences [3]. Like the PDB, it has seen tremendous growth in recent years due to improved techniques, and sequence data related to published work is often made available on GenBank.

As of February 2005, GenBank contained over 46 billion base pairs in over 42 million sequences covering over 140,000 known organisms. Each entry also contains detailed information about the sequence, the scientific name of the organism, bibliographic references, and other metadata. The total size of the flat files that comprise GenBank is approximately 180 gigabytes in release 146.0. Commercial hard disks are large enough to store this, but in order to ensure redundancy, these data need to be stored at multiple sites. Additional nodes storing GenBank data are not organized in an overlay network: They are simply dealt with directly as mirror sites. As the data stored in GenBank is not processed online, storage is of primary concern to the nodes.

2.3 Early Work on SimDB

SimDB was introduced at the University of Houston in 1999 [10] [11]. Since then, considerable work has been done on its design and user interface, from which the current development benefits greatly. Although some of the design decisions have changed since these initial offerings, the motivation and many of the basic ideas remain in the current design.

In the original design, trajectory data would be converted into a standard format prior to insertion into the system to allow for uniformity. These raw data servers and all other components in the system would communicate through a distributed object interface such as the Common Object Request Broker Architecture (CORBA) [17], a standard for creating software components and allowing for them to be executed remotely from different components.

Typical usage patterns were studied extensively in order to design a powerful user interface [35]. Based on this research, a user interface was developed using Java Swing and Java servlets. This interface allowed the user to browse the available trajectories, to search based on a number of important parameters, and to view structures through the use of the Visual Molecular Dynamics (VMD) [19] visualization program. The interface also enabled the user to submit scripts for custom analysis.

Other work dealt with the use of Grid technologies to perform expensive analyses [1]. This work included detailed descriptions of the communication that would need to take place between nodes in the SimDB network in order to handle a user query all the way from the selection of a trajectory and analysis function through processing and finally to formatting and displaying the results.

Considerable effort was also placed into defining and organizing the data in the system [26]. This work led to the creation of detailed database schemas which encapsulated many of the details necessary to fully describe the data contained in the system. These detailed schemas are necessary to determine where each piece of infor-

mation should be stored in order to maintain data integrity. Further, the inclusion of as much metadata as possible allows for complex and powerful searching, allowing users to find similar trajectories which match their criteria exactly.

2.4 BioSimGrid

The BioSimGrid project [40] has much in common with SimDB. Its primary goal is to make the results of simulations of biomolecules accessible. Like SimDB, the BioSimGrid intends to support multiple trajectory formats so that research groups can continue to use the tools that they prefer. When trajectory data are uploaded into the system, they are first converted into a defined format that is independent of the program which was used to generate them. In order to achieve this, conversion utilities must be developed which read a trajectory file in its native format and convert it to the database format. When the data are later requested, they are then translated from the database format into the format chosen by the user or other agent. In contrast, SimDB stores the data in their native format, and any necessary conversions are made when the data are requested. Several database nodes are to be deployed so that a large amount of storage resources are available and so that data sets can be stored at more than one node, thereby increasing redundancy.

The BioSimGrid initially intended to use grid computing to perform the analyses on the data, harnessing the power of any nodes that have been made available to the system. Data retrieval and analysis calls would be made by the user through the use of Python [42] modules and tools included in the Molecular Modelling Toolkit (MMTK) [18]. This would allow the user to define his or her custom analysis routines. Once the analysis functions had been defined, the job would be executed on a grid using available computing resources. There is no mention of BioSimGrid caching the results of processing in a cache so that they may be used to satisfy future queries.

More recently, it seems the focus of BioSimGrid has shifted to deal mostly with serving as a repository for simulation data. Researchers could collaborate with one another by storing their data on the system, which others could access and retrieve. Of course, the size of the data stored on the system remains large, so distributed storage must be used in order to accommodate these large requirements. In order to make finding data an easy process, the system must be developed so that the user is unaware of the underlying topology.

As SimDB and BioSimGrid share many common goals and ideas, it is possible that collaboration between the two projects may take place sometime in the future. Such collaboration could result in a common set of analysis functions and a standard for defining them, as well as potential interoperability between the two systems.

2.5 The ABC Database

Recently, a group of 17 investigators from 9 international research laboratories collaborated to obtain molecular dynamics trajectories for all unique tetranucleotide base sequences in DNA [6]. Using Amber, they obtained 15 ns trajectories for 39 different cases. These all-atom, explicit solvent simulations resulted in hundreds of gigabytes of data, which had to be stored in a way which allowed access to each of the collaborating labs. In addition to these coordinate data, additional parameters were calculated by using the Curves [27] algorithm.

To support this task, the Ascona B-DNA Consortium (ABC) Database was created. The data from the Curves analyses were placed into a relational database, and a web-based interface for querying the data is currently in development and will be offered to the public when completed. As the data are stored in a relational database, they can be queried using the structured query language (SQL). In addition to simply querying the available data, the web interface for the ABC database also offers

the user the ability to extract various other properties of the data, including the average helicoidal parameter values and their standard deviations over different periods of time. Additionally, it offers users the ability to make comparisons between simulations.

The ABC Database shares a number of goals with SimDB, most notably the storage of molecular dynamics trajectories and the querying of processed data. The main difference between the two is probably the scope of analysis options available to users. It is not clear whether the ABC Database allows users to perform advanced analysis queries in which the data related would have to be extracted from the trajectories and then processed, or if they can simply deal with the data already generated by Curves. Additionally, there is no mention as to whether the database will support additional trajectories, or if it is simply to deal with the analysis results of trajectories that have been generated in this collaboration.

Chapter 3

Molecular Dynamics Simulations

Before discussing the design and operation of SimDB, it would be prudent to introduce molecular dynamics simulations and discuss the details of the data that is under consideration. This information will help to further motivate the necessity of such a system, as well as provide some insight into the design decisions that are discussed in later sections.

First, the ideas behind molecular dynamics simulations are introduced in Section 3.1. Section 3.2 discusses the data that is generated by molecular dynamics simulations and how much storage space it requires. Finally, Section 3.3 focuses on the time that would be required to transfer data sets generated by molecular dynamics simulations across different computer networks.

3.1 A Brief Introduction to Molecular Dynamics Simulations

In classical molecular dynamics simulations, Newton's Laws of Motion are integrated to obtain the relevant conformations of a given structure that exist over time. By solving complex equations, the velocities and positions of each atom in the system

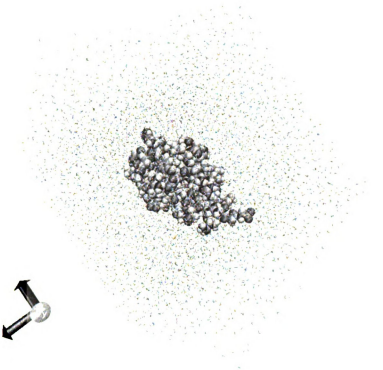

Figure 3.1: Explicit solvent simulation of the protein Ubiquitin

are calculated. More realistic simulations are achieved by including terms for various other phenomena that affect the atoms in the simulation, such as van der Waals interactions. These simulations are run using software packages such as Chemistry at HARvard Molecular Mechanics (CHARMM) [7], Assisted Model Building with Energy Refinement (AMBER) [34], and Groningen Machine for Chemical Simulations (GROMACS) [4].

Simulations can be run with either implicit- or explicit solvent. As the name implies, implicit solvent simulations do not include water (or other solvent) molecules, but additional calculations are made to simulate the effects of solvent molecules. With explicit solvent, the structure to be simulated is placed in solvent molecules with periodic boundary conditions. Explicit solvent simulations, although more accurate, can be more expensive than implicit solvent ones, because the number of atoms in the system is much larger, thereby increasing the number of calculations necessary at

each step. Figure 3.1 shows a single conformation of Ubiquitin, a small protein, from an explicit solvent simulation. Considerable work is being done with implicit solvent models to improve accuracy and the types of systems that can be simulated [13] [41]. One of the primary goals of this work is to allow larger structures to be simulated and also for longer simulations.

Simulations are run as long as resources permit. The time chosen depends on the behavior of the system to be observed. If the simulation length is too small, the behavior might not be seen, and if the simulation length is long, the computational costs rise, which may make the simulation infeasible. For example, translation, the synthesis of proteins from mRNA, may be seen in 10s or 100s of picoseconds, while protein folding and unfolding may require 10 milliseconds or more.

Another property related to simulation length that is important for this work is the sampling frequency, which is the number of time steps that pass between conformation data being written. The more frequently samples are taken, the more data that are produced by the simulation, and the more fine-grained information regarding the dynamics of the structure is.

3.2 Simulation Data Size

Although the price of hard disks and other computer storage media has fallen dramatically in recent years, alongside an increase in their capacities, the amount of data that results from simulations of biomolecules is still a major limiting factor in the ability of researchers to make their data easily accessible to themselves and the scientific community. Further, as computing power has also increased, more complex simulations are being performed over longer periods of time, which results in considerably more data being produced.

Traditionally, research groups perform specific simulations in order to examine

a certain biomolecule or system, then the data resulting from these simulations are analyzed, and the data are finally either moved offline to a tape archive or simply deleted. This has been a necessity, as the disk space used was needed for future simulations. Because of this, however, these data sets are no longer easily accessible to the group or anyone else who may be interested in working with them.

Data files from simulations consist of the set of atoms simulated in 3-dimensional space for each sample taken. In other words, the coordinates of each atom in the simulation at each sampling instance are included. This can be described with Formula 3.1:

$$S = A * 3 * 4 * L * f \tag{3.1}$$

Where S represents the total size of the data in bytes, A is the number of atoms in the system, 3 is the number of dimensions used to describe the position of each atom, 4 is the number of bytes used to represent each coordinate value, L is the length of the simulation in picoseconds, and f is the sampling frequency in samples per picosecond.

The number of atoms used in a simulation depends not only on the molecule or system in question, but also whether the solvent that it is immersed in is defined explicitly or implicitly. For explicit solvents, all of the atoms in each of the solvent molecules are included. The total number of atoms, therefore, can fall within the range of 1 and 1,000,000. One cannot sample less than one atom, and 1,000,000 atoms will result in very complex simulations that may not be practical with current resources. In current simulations, the number of atoms typically falls within the range of 500 and 100,000.

The simulation length argument used in Formula 3.1 represents the total length of time in which the simulation is run. This refers not to the wall time which is required for the simulation to complete, rather to the length of time in which the behavior of the molecule is simulated. Because this number is proportional to the

number of calculations which must be performed in the simulation, the length is limited by the resources available. In current simulations, this number typically is between 0.1 ps and 100 μs. Longer simulations can be performed; however they may take years to complete even on large clusters of computers.

The sampling frequency in Formula 3.1 refers to the frequency at which the coordinates, velocities, and other properties of the system are recorded during the simulation. The more frequently samples are taken, the more data that is written, so this parameter is closely related to the size of data produced by the simulation.

To illustrate how these values contribute to the amount of data generated, let us examine a typical simulation. We will be looking at a molecule in solvent with 10,000 atoms. This structure is simulated for 15 nanoseconds and sampled every 0.1 picoseconds. This will require 120 kB of storage per sample for each of the 150,000 samples taken throughout the simulation. This simulation will produce 18 GB of data plus header information, which is negligible. Current disk drives can store a few such simulation data sets, but will be quickly filled, leaving no room for future simulations.

These data are typically written in binary format in order to keep file sizes as small as possible. Because of this, the files containing simulation data are only easily accessible on machines whose architecture matches those on which the simulation was run. Fortunately, the number of architectures being widely used today is much smaller than in the past, so the number of different binary formats that need to be dealt with is not overwhelming. Additionally, most architectures follow standards in how numbers and data are written, which aids in the conversion between byte formats.

Several applications that deal with binary simulation data support both big- and little-endian data formats, so that they can work with data generated on different architectures. SimDB plans to support these different formats as well. The files will not be converted to match the architecture of the nodes that store them; however

they will be converted if necessary when being transferred to processing servers.

3.3 Transmission of Simulation Data

Once this simulation has been performed, if another research group is interested in the data generated by our example simulation, it must be transferred somehow. Although it would be possible to put the data on a tape which would be mailed, this approach would take many days to complete, and tapes are relatively expensive. Cheaper storage media such as CDs and DVDs to not have the capacity to store the generated simulation data. The easiest option would be to transfer the data over a computer network.

The capacities of the network links between a pair of nodes can vary greatly. On one end of the spectrum would be a transfer in which both the sender and receiver are on the same local network. In this case, the link could allow for very high transmission rates, on the order of 100 Mb/s, 1 Gb/s, or even 10 Gb/s. Even though transfers will probably rarely occur at these rates, it is a plausible upper bound.

The other extreme might be a transfer between one machine in Asia and another in the United States. In such cases, the potential for transmission rates depends on the slowest link that exists between the nodes. Long intercontinental and trans-oceanic links will likely play a major role in this case, due to their congestion and limited capacities. These links could easily lower transmission rates to 10 Kb/s.

The majority of links will likely fall between these extremes. If the nodes communicating are members of the Internet2 consortium [21], high rates of data transfer can be achieved. For example, the Abilene Backbone [31] uses OC-192c links, which support aggregate transfers up to 10 Gb/s. This means that transmission rates on the order of 1 Gb/s are possible between such nodes, however depending on traffic, maximum transfer rates in the 100 Mb/s range are more likely.

Non-Internet2 members will more likely be using the more common Internet backbones. Most of these backbone links are DS-3 circuits and are capable of transmissions of 45 Mb/s. Depending on the research group's connection to these backbones, transfer speeds on the order of 1-10 Mb/s are attainable, although congestion plays a larger role when using these links, as they are most likely shared by a large number of subscribers.

Although it will probably rarely be the case if at all, it is possible that a research group is connected with a hybrid fiber coaxial (cable) or digital subscriber line (DSL) modem. In this case, the current maximum transmission rates deployed for these systems is about 3 Mb/s. However, 300 kb/s is more likely to be an average rate. One potential group of users who may be connected through such links would be high schools or small community colleges, who would use SimDB for educational purposes.

The time to transmit data can be calculated using Formula 3.2:

$$T = \frac{S * 8}{t} \tag{3.2}$$

Where T represents the total time in seconds to transmit the data, S represents the size of the data to be transmitted in bytes, and t is the rate at which the data is transmitted over the network in bits per second. This rate will almost never be constant, so an average rate is used instead.

As an example, let us say that an 18 GB data set is to be transmitted from one site to another, and the link between them allows for an average transmission rate of 5 Mb/s, which may be considerably higher than what may seen in most cases. Using Formula 3.2, the time required to transmit is just under 8 hours. It may be more likely that the average transmission rate seen when using the Internet is 1 Mb/s, in which case the transmission will take over 39 hours.

Although the times that have just been calculated are under two days, one must remember that this does not represent the total time involved. This is because the

data that was transferred is raw data, and in order to observe interesting features of the data, processing must first be done, and this can take considerable time as well. It is also the case that many interested parties simply do not have the computational resources or know-how to process and analyze the data.

As the goal of SimDB is to generate and store processed data, these large raw data sets will not need to be transmitted each time someone wishes to deal with them, which will reduce the amount of data which must cross the network. Furthermore, since the data have already been processed, the expensive and time-consuming analyses will not need to be run in many situations. Because of these savings, results to requests submitted through SimDB may be available in minutes or hours instead of the days or even months that may be required with traditional methods.

Chapter 4

System Architecture

This chapter discusses the design of the SimDB architecture. Design consideration is critical to the operation of the system, because it effects the scalability, throughput, and resource utilization of the system. Although SimDB will not meet the large number of users seen in current peer-to-peer and distributed systems, much of the design follows work in these areas.

Section 4.1 discusses the basic design of the system, while Section 4.2 introduces the specific design elements employed by SimDB. Sections 4.3 through 4.6 describe each of the different node types in detail. Finally, Sections 4.7 and 4.8 present two use cases which document how the different components of the system are used and interact with each other to perform common operations.

4.1 Basic Design

Because of the large size of the data that will be stored in the system and the amount of computing resources required to process these data, SimDB has been designed as a distributed architecture to allow the system to grow comfortably. The model used follows many previous works in which the nodes in the system are managed by one centralized node, which makes decisions as to which jobs are completed

where and on which machines. This centralized node also plays other roles such as maintaining information about the nodes available to the system, providing authentication and authorization mechanisms to manage access, and indexing the data on the system so that they can be found easily through search or browsing.

In the peer-to-peer community, many current systems such as Gnutella [16] and Kazaa [25] specifically avoid using a centralized control server as had been done with the original Napster [30], because centralized servers pose two potential problems for the system. The first potential problem is that the centralized server can become a bottleneck if all jobs executed on the system must first pass through them. If there are many concurrent jobs, the centralized node may become heavily loaded, and because of this all jobs will suffer some delay or simply be dropped. The second potential problem is a similar one. Should the centralized server fail for any reason, the operation of the system as a whole would halt, as all queries and jobs require the centralized server for proper execution. It is, however, not believed that a centralized architecture would be a significant weakness in SimDB as in the case of peer-to-peer networks. This is because the number of nodes in the system, as well as the number of concurrent jobs and the frequency with which they will be started, will be significantly less than those of P2P systems, in which these numbers are on the order of millions. It is likely that at most hundreds of computational groups around the world will make use of SimDB, which can be efficiently managed by the central server. While users on file sharing networks may submit new queries as frequently as every few seconds, since jobs in SimDB may take hours or days to complete, the time between queries is likely to be much higher as well.

4.2 Unique Design Elements of SimDB

Unlike most distributed and peer-to-peer systems, there will be a number of different types of nodes participating in the system, each performing a different duty. This can be explained by the SimDB requirement that the different functions require different resources. For example, the storage of simulation data requires a large amount of disk space, but not necessarily a powerful processor. Conversely, a node that performs analysis on a given data set may need significant processing power, yet since only one or a few nodes are dealt with at a given time, less disk space. This heterogeneity of node type leads to some interesting questions, such as how to select a node to interact with based on the job at hand. The different node types and how they interact is shown in Figure 4.1.

Membership in the different node categories is not exclusive. That is, hosts may be able to perform multiple duties at a given time as long as they have the required resources. For example, a host with significant processing capabilities and a large amount of disk space can act both as a processing node and a storage node. These node types are separated to allow for system flexibility, so that each research group may share the resources that they have available, without the need to purchase additional hardware.

4.3 Raw Data Server

Raw data servers are responsible for storing large trajectory data that result from simulations. As such, they are one of the most important parts of the SimDB system. They essentially act as a database in that they support the storage and retrieval of data sets as well as more complex operations such as querying, maintaining metadata related to the data that are stored, and performing operations on those data.

When data sets are added to SimDB, a raw data server is chosen, metadata are

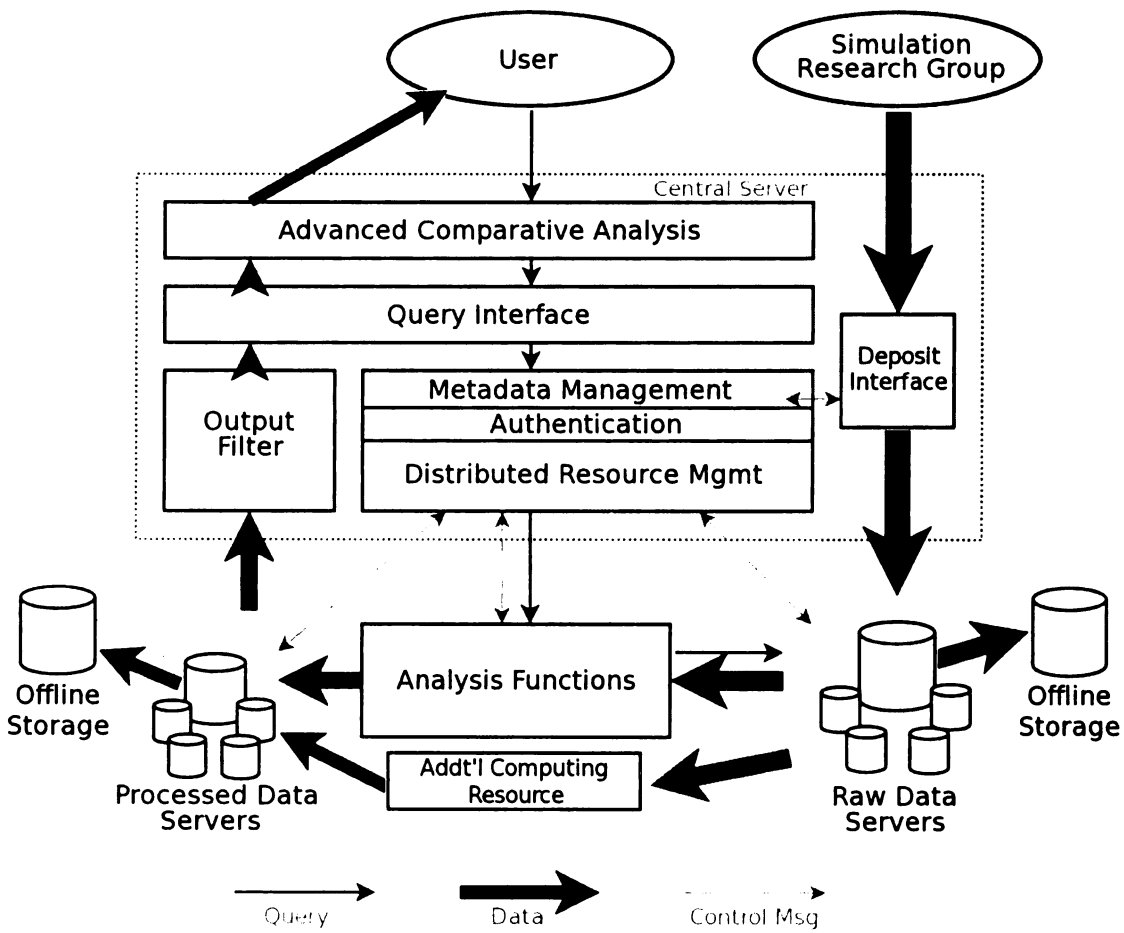

Figure 4.1: The architecture of SimDB consists of several node types which interact with each other.

stored at the central server, and the data set is transferred to the selected raw data server. More than one data server may be chosen in order to increase redundancy. Although storing a given trajectory at multiple sites does require more disk space in total, this redundancy helps ensure system operation should one or more nodes fail. Replication also lessens the burden that a raw data server may experience if it is hosting a "popular" data set.

Data sets are stored as-is. There is no conversion to a program-neutral format or byte swapping done. By implementing functions that can read and write different simulation data formats, however, the data can be converted to different formats as necessary. This process is generally computationally-inexpensive and adds little time to data retrieval.

By supporting many formats, raw data servers can perform basic processing on the data before transmitting them to a processing node. One example of this is the ability to select individual frames and atoms, so that the entire data set does not need to be transmitted prior to processing, just the desired data. Aside from the obvious savings in network transmission time, this benefit also means that the additional data does not need to be filtered out at the processing nodes, so they can perform analyses immediately, with results reaching the user more quickly and the processing node becoming available to new jobs more quickly.

Additionally, raw data servers may be attached to large backup media such as tapes. While it would be ideal if all data were available at all times, the large size of these simulation data sets may make this difficult to achieve. By moving the oldest or least-recently used data sets to tape, additional resources are made available for new trajectories. Should a data set that has been moved to tape be requested, it could be retrieved manually or automatically at the administrator's convenience. Since one of the goals for raw data servers is to provide some redundancy, it is also possible that one node maintains an online version of an older data set while others move their

copies to tape. Finally, it is also possible that parts of a data set that is about to be moved to tape have been previously processed, and are available in a processed data server, which will be discussed in Section 4.5.

Disk space is the most important resource for raw data servers. We have seen in Section 3.2 that raw data sets will typically be around 18 GB. It is hoped that each raw data server will be able to store many data sets. Fortunately, the capacities of modern disk drives are quite large, and their costs are fairly low, so it is easily conceivable that a raw data server will offer at least 500 GB of disk space. Additionally, the raw data server supports a read-only mode in which data sets can be read from, but not added to the server. This allows groups to make their trajectory data available even if they do not have ample storage resources.

4.4 Processing Nodes

The system's processing nodes are responsible for performing operations on the data that exist in the raw data servers. These operations consist of clustering the conformations in a given data set, scoring the trajectories, and running various analyses on the data. They are executed when a user submits a query for which the appropriate result data has not already been calculated and stored in a processed data server, which is discussed in Section 4.5.

The complexity of these functions ranges from simple calculations, such as radius of gyration, which looks at the distribution of the structure around the center, to more computationally intensive ones, such as nucleic acid backbone torsion analysis. As the complexities of the analysis functions can vary tremendously, so can the times required for the execution of these functions to complete. Execution time depends both on the size of the data set and the analysis at hand. Simple calculations such as radius of gyration may be completed within a few seconds on single-CPU machines,

while more complex analyses can require up to several weeks on large clusters of computers. Table 4.1 lists several analysis functions and the minimum and maximum times required to perform them on a few different trajectories on a single CPU on a 2-CPU machine. To account for larger data sets and more complex analysis functions than those used for testing, the last column displays the maximum times seen times 100, which is a fair estimate of potential execution times. Similarly, run times for MMPB/SA ?? scoring are displayed in Table 4.2, and times from clustering runs are displayed in Table 4.3. Some of these analyses can be run in parallel, so these times can be divided by the number of nodes used in the analysis to obtain a rough estimate of their completion times.

Table 4.1: Example Analysis Run Times

Function	**Min**	**Max**	**Max * 100**
Radius of Gyration	5 s	20 s	34 m
Coordinate RMSD (all atoms)	5 s	25 s	42 m
Protein Secondary Structure	15 s	1 h	100 h
Backbone Torsion	4 s	30 m	50 h
Ribose Pseudorotation	30 m	1 h	100 h
Nucleic Acid Helical Analysis (Base Pairs)	25 m	45 m	75 h
Nucleic Acid Helical Analysis (Base Pair)	20 m	50 m	83 h
Nucleic Acid Helical Analysis (Base)	30 m	50 m	83 h

Table 4.2: Example Scoring Run Times

Function	**Min**	**Max**	**Max * 100**
MMBP/SA Scoring	5 s	30 m	50 h

Table 4.3: Example Clustering Run Times

Function	**Min**	**Max**	**Max * 100**
K-Means	5 s	45 s	75 m
Hierarchical	5s	5 m	8.33 h

In addition to creating unique analysis functions for SimDB, existing toolkits such as the Multiscale Modeling Tools for Structural Biology (MMTSB) [12], the Molecular Modelling Toolkit (MMTK) [18], and ptraj [20] can be used to perform

analysis. As these toolkits provide a wide range of analysis options, they can help in the rapid development of functions.

Because of the large range in possible execution times, SimDB will need to deploy multiple processing nodes in order for all types of calculations to be completed within reasonable time. As we saw in Tables 4.1, 4.2, and 4.3 , many analyses can be done on standard workstations in acceptable time periods, and because of this, a workstation with a single, fairly high-powered, CPU and perhaps 512 MB to 1 GB of memory would be a capable node in the SimDB network. For the most part, analyses can be completed within several hours on these nodes. For the largest jobs, groups with large computing resources, such as clusters of computers, could make these resources available at certain times to help with large jobs.

A final consideration that should be made for processing nodes is disk space. Although these nodes will not be storing many large trajectory datasets, they will need a fair amount of space available to store the data corresponding to the current job being analyzed, the resulting data from the analysis, and any temporary data that may be written. In fact, a processing node may have a few jobs being processed concurrently or sequentially. This number would rarely be more than a few, though, so having 50-100 GB of disk space available to these data sets should suffice. This amount of disk space is available in most current systems.

The amount of data produced by analysis is considerably less than the raw data for the trajectory in question. For example, plain-text, delimited results for a trajectory that contains 18 GB of data may require only 100 MB of storage space. The range of possible processed data set sizes is likely between 1 MB and 1 GB.

Another possible use for storage resources on processing nodes would be to keep a few data sets on the server in anticipation that future queries will be issued for these data. If this would be the case, the cost and time related to transferring the data from a remote raw data server would be saved, and processing could begin immediately.

Because disk space may be limited on these processing nodes, different replacement policies must be considered when we decide which stored data sets to remove when space is required.

A naïve approach would be to simply delete randomly-selected data sets until the amount of free space available is enough to store the incoming data set. While this approach would in fact work, it may not make the best possible decisions, which could result in cache misses in the future, even though a hit would have been possible had another replacement policy been in use.

One simple approach would be to delete the least-recently-used (LRU) data set that is stored on the processing node. Should the deletion of one set not create enough space, we can continue deleting least-recently-used sets until sufficient space has been made available. This approach operates on the assumption that since the processing node has not dealt with the least-recently-used set for the longest amount of time, this set is the least likely to be used in the future.

A third approach would be to delete the most-recently-used (MRU) data set that is stored on the processing node. This approach makes the assumption that since the most-recently-used set was recently analyzed, it is not likely to be needed again. As with the LRU policy above, multiple most-recently-used sets may need to be removed to free up enough space for the incoming data set.

If we consider that it takes less time to transfer smaller data sets than larger ones, one policy might remove the smallest data set until enough space has been freed for incoming data sets. If the deleted sets were needed again in the future, they can be re-transferred in (relatively) shorter time than larger sets. This approach would likely remove the largest number of sets, making a smaller number available than other replacement policies, which could lead to a higher miss rate.

Another approach might be to find and delete one or more data sets whose size is closest, yet greater than the size of the incoming data set. This approach aims to

keep storage resources as fully-utilized as possible, so that it can store as many data sets as possible at a given time.

Finally, if we consider each set's popularity, we can delete the least-popular sets until enough room is made. This is slightly different than the LRU policy, in that older, but very popular sets historically will remain on the system, and more recent but less popular sets will be deleted. The disadvantage of this approach is that newer sets will not have had as much time to become popular, so they may be quickly removed even if these sets will be used frequently in the future. This approach may take this potential problem into consideration by normalizing the popularity, perhaps by considering the number of times a set has been used divided by the number of days or hours that it has been stored on the system.

In reality, a combination of these approaches may be used in order to maximize the effectiveness of this cache and minimize the costs associated with cache misses.

4.5 Processed Data Server

Because of the complexities and costs of running complex analyses on data sets, it would be very beneficial for the system to store the results of previously-run jobs, so that these results can be used to satisfy future queries that deal with the same data. By doing so, the computational resources that would have been allocated to these future queries can be allocated to new queries. An additional benefit to such caching is that users would be able to get results to queries almost instantaneously if the same or a similar query has previously been issued.

The purpose of processed data servers is to store these analysis results so that they may be used to satisfy future queries. As such, the main resource requirement for processed data servers is disk storage. As we saw in the previous section, the disk space required to store one result set is considerably less than that required for raw

data sets. Because of this, each processed data server on the network can store a large number of processed data sets. For example, a single processed data server with 500 GB available disk space could store 5000 100 MB result data sets. The result of this is a system in which many queries can be satisfied without any processing.

Because the output of analyses will have not been filtered to display exactly the data requested by the user in the format that they specify, the processed data is somewhat generic, and has the ability to be used to satisfy a number of queries, as long as the data queried for is contained within the processed data set. For example, if the user who submitted the original query wanted to see a plot of a structure's RMSD over time for a certain period of time, and the analysis calculated the RMSD for all times. These data could then be useful to any future queries for the RMSD of this structure at any time. When the final output is created for the user, the additional time periods are simply removed.

As with raw data servers, it will be possible to move processed data sets onto tape if they are not frequently used. Although it would require additional time to retrieve backed-up data sets from tape, this time may still be less than the time it would require to transfer and re-analyze the data. Also, this re-analysis would require more nodes in the network to be used. It is believed, though, that since the size of processed data sets is so much smaller than that of raw data sets, backing up to tape is something that may very rarely occur, if at all.

4.6 Central Server

The central server serves many roles in the operation of SimDB. These different roles work towards integrating the nodes that exist in the network so that they may inter-operate seamlessly, and so the distributed nature of the network is not visible to the user. Additionally, the central server also controls access to different resources so

that none of the resources are used in a way other than what is intended, and allows groups to define permissions for their data, so that work that is not yet public can be stored and processed safely.

4.6.1 Resource Management

Resource management deals with organizing and controlling access to the nodes available in the network and the data contained on them. This component keeps track of all the nodes in the network as well as additional information about them that may be used in making decisions regarding the use of resources. For each node type in the system, the resource management maintains the name and location of the resource as well as its address. Additionally, the capabilities of the node, such as free hard disk space or number of processors and amount of RAM, may be stored in order to rank the available resources when a job is submitted. This information could be gathered as the node joins the network, or can be requested by the resource manager when the node is being considered for use.

Since the selection of a best set of nodes to use when performing analysis is not a one-dimensional decision, as with most peer-to-peer networks in which network transmission speed is considered, the more accurate information about a node available, the more appropriately the resource management can allocate resources. The important factors that must be considered in choosing nodes depends on the node type to be used.

For example, if the resource management component is selecting a raw data server on which to deposit a new data set, it should consider the amount of disk space available on the nodes as well as their network connections. One node with a large amount of available storage and a slower network connection is not as desirable as another with less, although sufficient, storage resources and a very fast Internet connection, because the node with the faster Internet connection will be able to get

the data to processing nodes more quickly, which results in the execution of the analysis being done earlier and the results being available more quickly. These same criteria can be used for processed data servers as well, since storage is also of primary concern.

A second example would be choosing a processing node to be used for analyzing a data set. In this situation, the available processing resources and perhaps also the amount of system memory should be considered. Of course, the node should also have enough storage resources to store the data set being analyzed, as well as temporary and result data. Because of this, even a large cluster may not be selected for processing, because it simply does not have the storage resources to deal with the job. For processing nodes, it does not matter how much storage resources are available, just whether or not enough resources for the job are. Any node with enough storage space should be considered, and the processing capabilities of these nodes should be used to rank them. Additional consideration can be given to network speed, but since analyses can take many days to run, this is not as important a factor as processing power.

4.6.2 Simulation Metadata Management

Another important set of data that the central server maintains are the metadata related to the data stored on the system. This information provides additional details about simulations to accompany data provided in trajectories, such as who performed the simulation, the environment in which the simulation was run, and which methods were used in the simulation. This information will help researchers to find other sets available on the system that match their interests. By providing powerful search capabilities, users are able to define any aspects of simulations that are of interest to them, and the metadata management can quickly find all sets that match the user's criteria, display the complete metadata set for those records, and offer the ability to

analyze these sets. In a system with many available data sets, this functionality is critical in order to allow users to sort through the available sets and find exactly what they are interested in.

Additionally, the metadata management component stores information about which nodes on the network store the data in question, so once the data sets are found, they can be dealt with. At this point, the metadata management will interact with the resource management component in order to negotiate the transfer of the appropriate data from the raw servers to an available processing node. Finally, the resource management component selects one or more processed data servers on which the resulting data will be stored for future use.

4.6.3 Authentication and Authorization

Authentication and user access allow groups to set limits on the data that they place on the system. Although SimDB intends to make many simulation data sets available to the public, the ability to make a data set private or available to a limited set of users is important. By limiting access to a data set, research groups from around the world can collaborate with each other and use the system to perform research on a biological system before making it available to the public. Additionally, in order to prevent misuse of the system, authorization may be given to known trusted users so that they can take full advantage of the system while others may need to earn access to the system.

4.6.4 Query Management and Output Filters

Once the user has connected with the system, he or she submits jobs to be done on available data. The query management component of the central server receives these queries from the user, parses them into a logical format, and then consults with the resource management component to negotiate resources on which the job can be

run. Once resources have been allocated to the job, the query manager then passes the query on to the processing node for completion.

When the job has been completed by the processing nodes, the results are returned to the query management. At this point, the resulting data will be filtered according to the user's specifications, and the final data will be returned to the user. By using output filters, the same generic processed data can be used in many different ways. For example, the user may request comma-delimited data to be used in another database. The user may also request a plot of the values calculated by the analysis. Other output filters such as tabular data and Ramachandran plots allow the user to receive the processed data in a way that works best for them.

In addition to formatting the resulting data to meet the user's request, filtering the output also has the effect of further decreasing the size of the resulting data. Now, 100 MB of processed data may be converted into a 50 kB plot, which is many orders of magnitude smaller than the original raw data set. By using the SimDB, the user is able to receive the final data in a matter of seconds instead of having to gain access to another group's data, transfer the complete set, run complex analysis on the data, and then finally create the resulting data.

4.6.5 User Interface

Of course, in order for a user to interact with each of these components of the central server, a user interface must be available to them. This interface must allow the user to browse and select the available data in the system, select analysis functions and set parameters for them, and display the results in a flexible manner. The current web services make use of a web site to interact with the user. As a web-based interface allows users of many different web browsers and operating systems to deal with the system in a consistent way, it will be used in future incarnations of SimDB as well.

So as to to not overwhelm the user with available data, functions, and options,

the interface must act intelligently to show the user important and relevant options, while hiding things that either do not deal with the task at hand or are considered advanced options which are rarely used. Of course, some users may wish to have access to advanced features, so they must also be able to make these things visible. Additionally, not all users may understand all parameters to the operations they wish to perform, so all values should be given sensible defaults, so that a user may still perform legitimate analyses after customizing few or no parameters. Of course, some users may wish to understand all parameters being used, so explanations of each parameter should be made available to the user, either in separate documentation or a separate window. Because of this, not only is SimDB a powerful tool for research, it can also be used as a valuable educational tool.

4.6.6 Administrative Interface

A similar interface for administering the system is also provided by the central server. This interface allows administrators to easily manage all aspects of the system such as the resources and data available on the system. Tasks performed by the administrator through this interface include the insertion, deletion, and modification of resources, data sets, and users as well as the permissions for resources and data sets.

4.6.7 Central Server API

In addition to the web-based interfaces, an application programming interface (API) will allow researchers to make their own interfaces to these central components of SimDB. However, because this API will interact with the different components of SimDB in the same way as the web interface, the distributed nature of the system will not be apparent to the user, nor will they be able to bypass any security mechanisms. Regardless of these restrictions, the API will allow users to interact with SimDB in

a way that benefits them the most.

4.7 Use Case 1: Typical User Query

As an example of how the different components of SimDB interact, consider the steps taken when a user intends to create a plot of the root mean square deviation (RMSD) over time of a structure that is stored on the system. In RMSD calculations, a reference structure is selected, and the and all other conformations are compared against this reference. If the result is high, then the two conformations are quite different, while more similar conformations will have lower RMSD values.

The first thing this user will do is connect to the system and the user interface. He or she then uses the user interface to browse or search through the available metadata, which is maintained by the metadata management component of the central server. This is shown in Figure 4.2. Once the user has found a data set for which he or she intends to perform analysis, the user then builds a query by selecting RMSD from the available, relevant functions, which are determined by the central server, and setting the related parameters to the RMSD function, as is shown in Figure 4.3. The user decides to use the first structure of the trajectory file as the reference structure, and to include all atoms from all residues in the calculation. Once completed, the query is handed off to the query management component of the central server, which first interacts with the metadata management system to see if the request could be satisfied by data from a previous query that is stored in a processed data server. In that case, there are no existing data in the processed data servers, so the query management component interacts with resource management component to determine which available processing node to use for the analysis. Once the query arrives at the processing node, it requests the raw data from the raw data server on which the data are located. After the transmission of the data, the processing node runs the specified

analysis on the data, and sends the resulting data to the processed data server which it was instructed to use by the resource management component. The processing node then notifies the query management component that it has finished analyzing the data and that the results are stored on the processed data server. The query management software then retrieves the data and notifies the user that the analysis has completed. The user now selects plot from the available output filters, and the resulting plot is created for the user. These two steps can be seen in Figures 4.4 and 4.5, respectively.

4.8 Use Case 2: Query with Cached Results

We now examine the behavior of the system when a user wishes to obtain the RMSD over time for a specific time period as a comma-separated list for the data set which was used in the previous case. As in the first case, the user connects to the system and searches for the data set in question. Once finding this, the user selects RMSD from the list of available functions for this structure and defines the parameters for this structure. The parameters that the user selects are the same as those selected by the user in the previous case. Now, when the query is handed off to the query management component, by browsing the available metadata in the system, it is determined that a similar analysis has been performed in the past, and that the results from that query are online on a processed data server, and that they could be used to satisfy the current query. Now, instead of interacting with the resource management component, the query manager retrieves the data from the processed data server. The user then selects his or her desired output filter, which in this case happens to be comma-separated list. The data is then filtered, and the user now is presented with a comma-separated list of the RMSD values for the structure during the time range that he or she specified earlier.

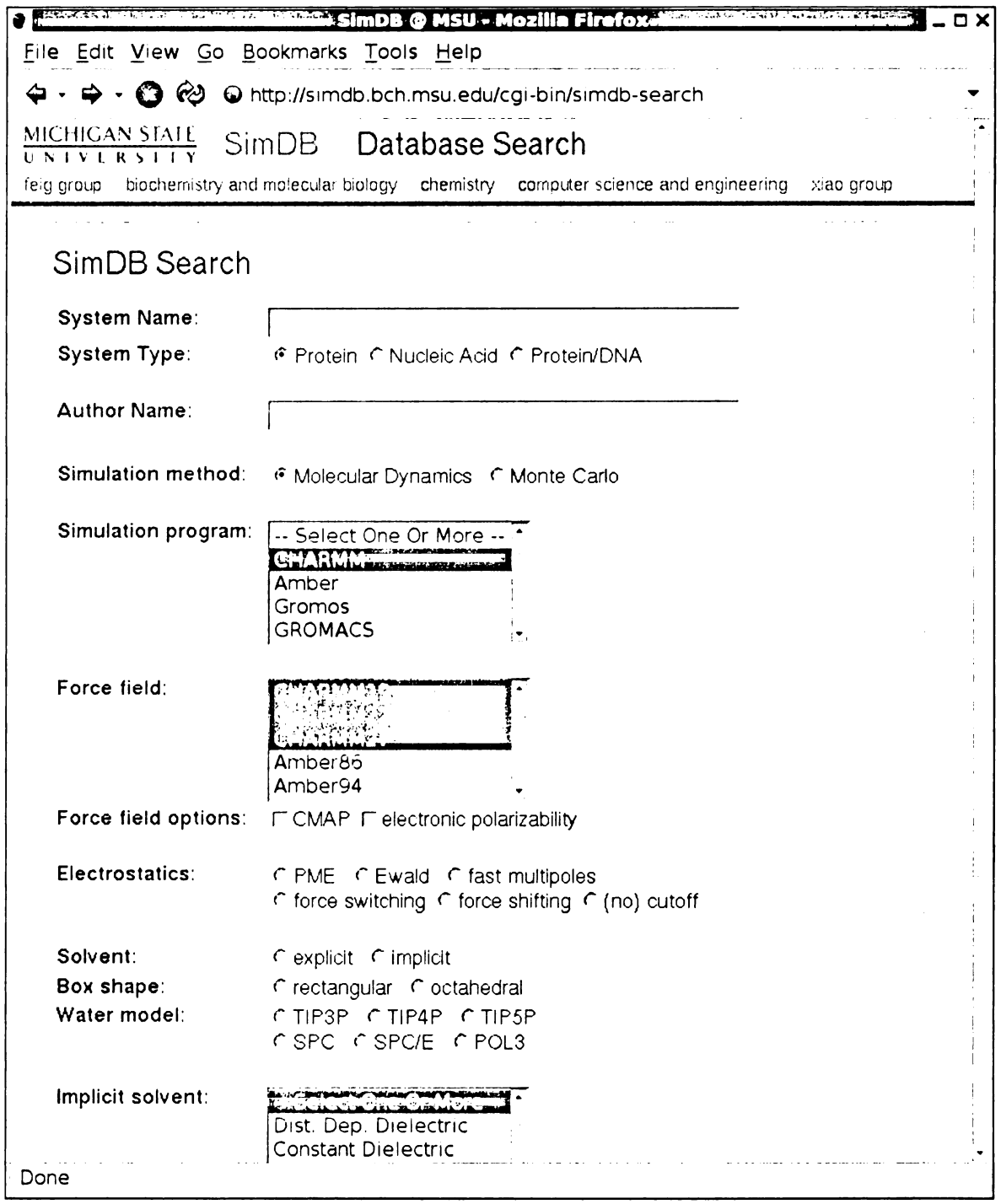

Figure 4.2: Trajectory Search Using Metadata

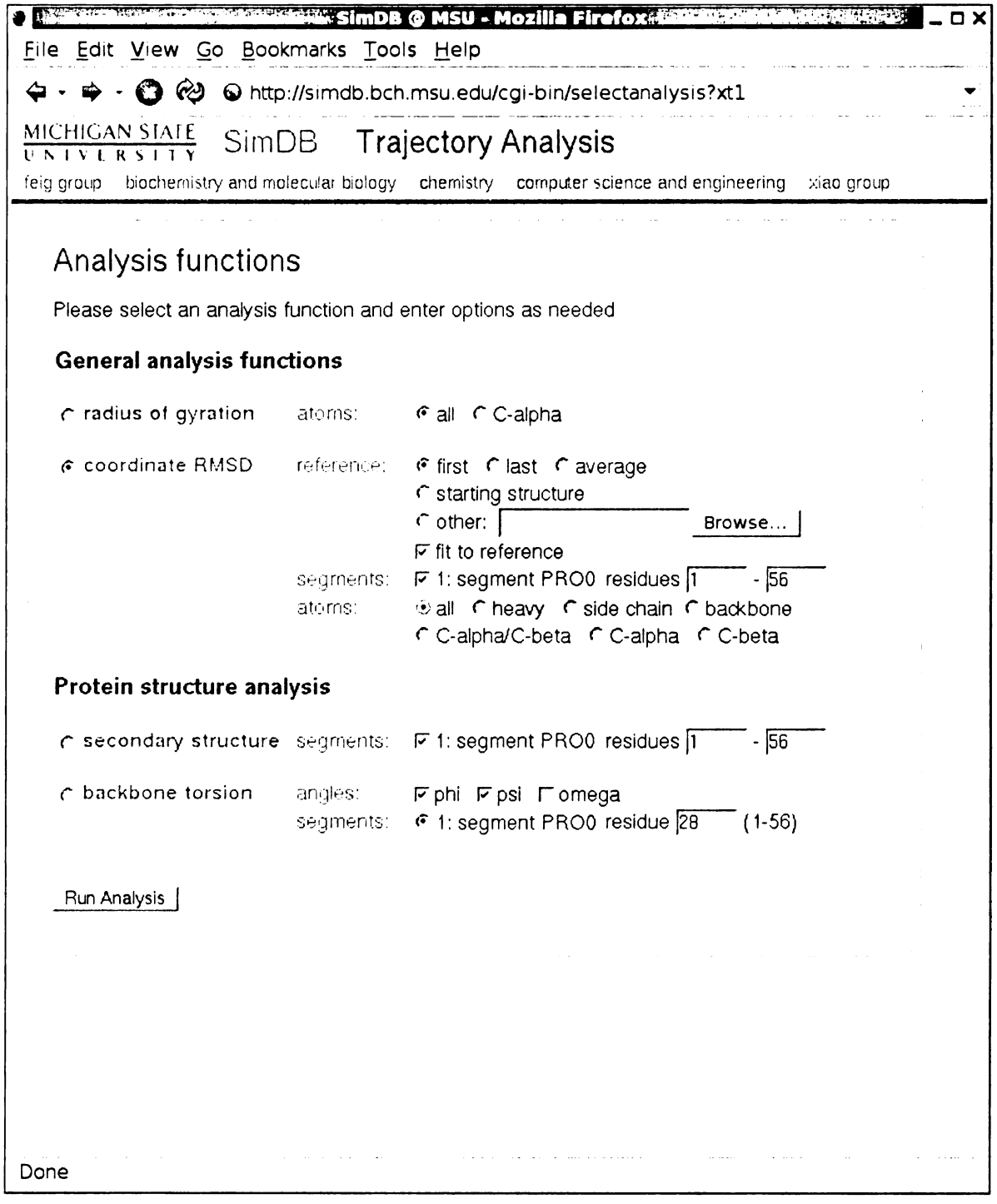

Figure 4.3: Selecting an Analysis Function

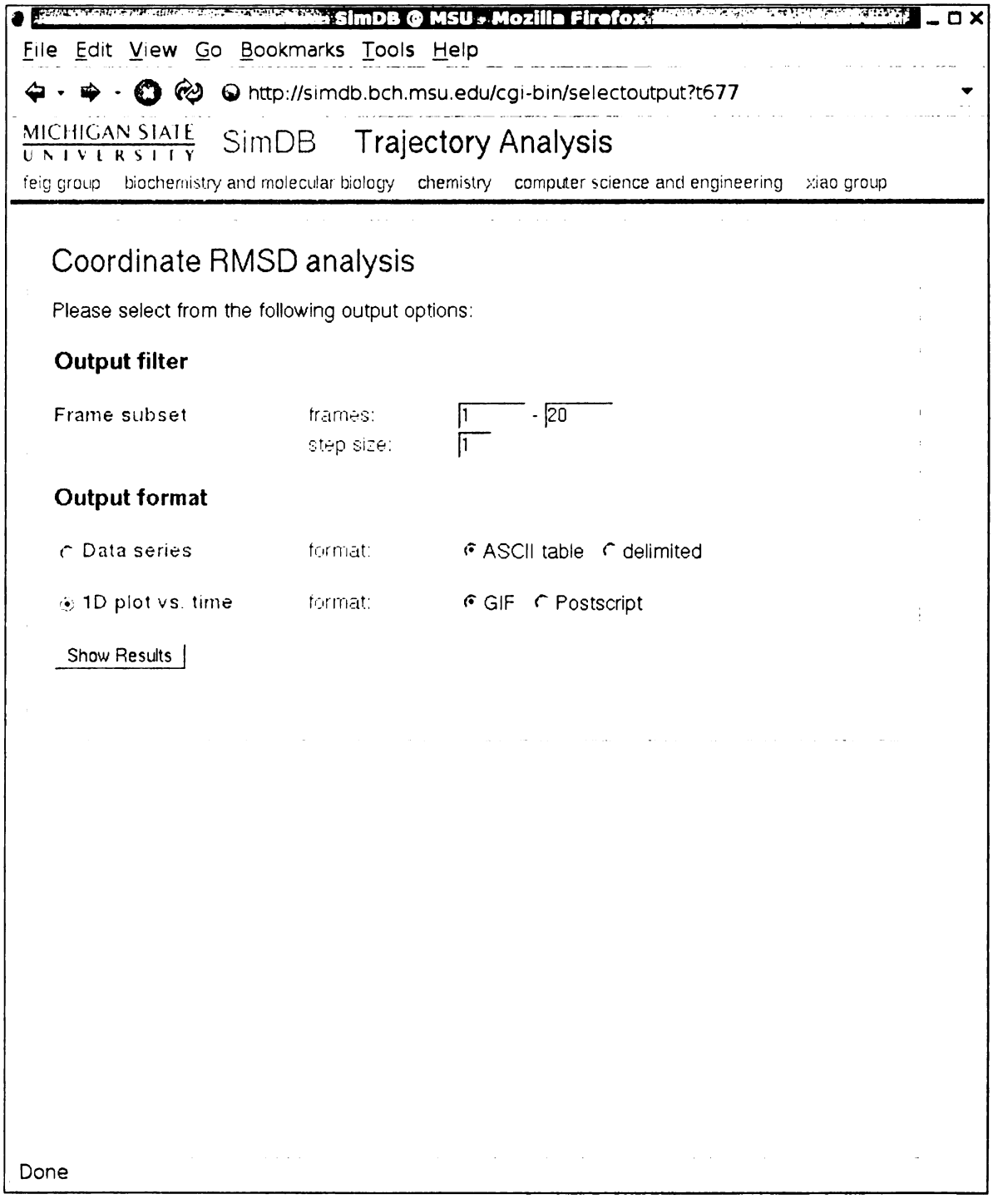

Figure 4.4: Selecting an Output Format

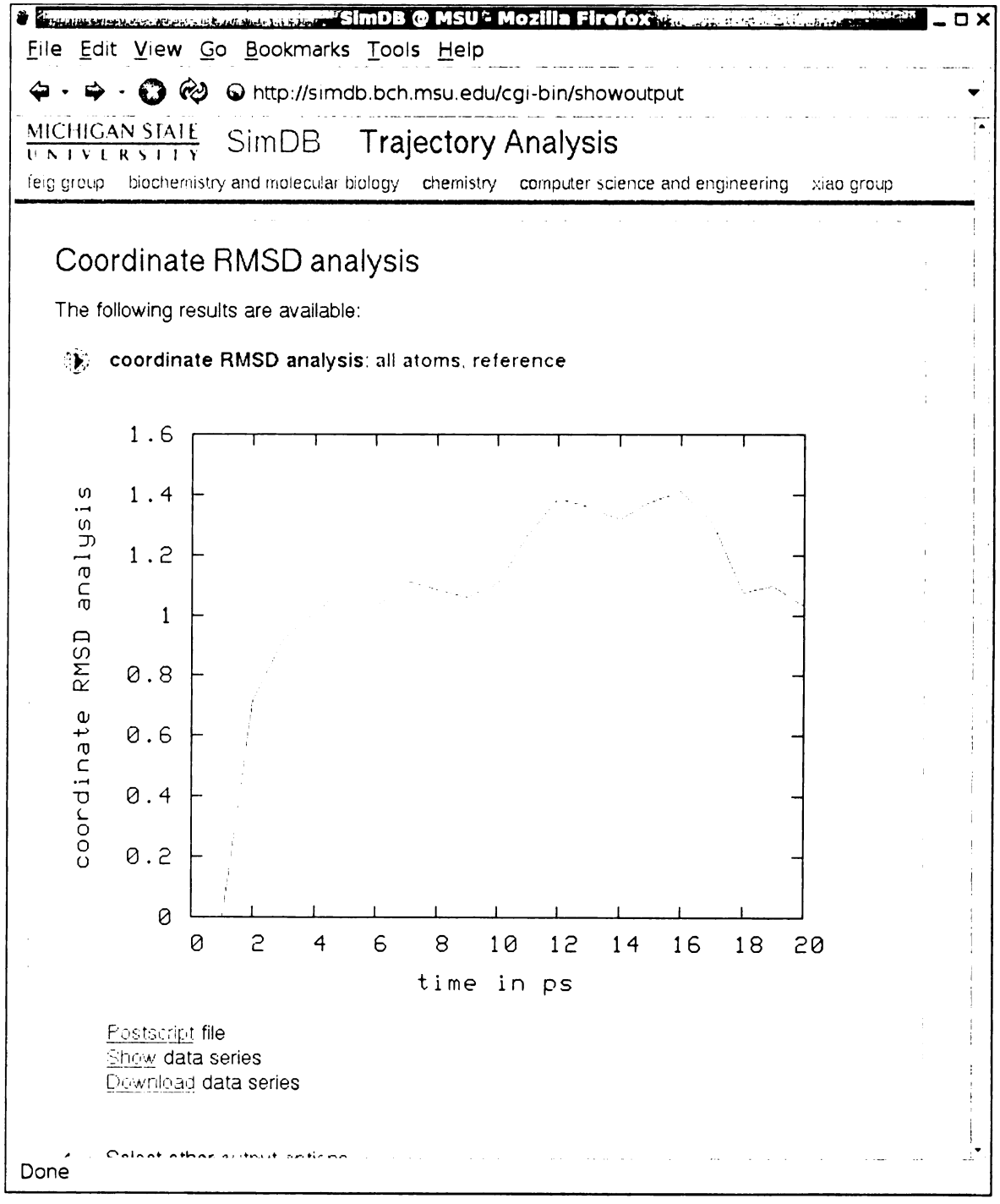

Figure 4.5: Displaying the Final Results

Chapter 5

Preliminary Results

Considerable work has been done on SimDB over the past two years. This work was approached in a bottom-up fashion, where critical components were dealt with first, and additional features have been saved for future work. Of the system components discussed in Chapter 4, many have been implemented at least partially.

Using these components, a website has been established to demonstrate the capabilities of SimDB [39]. These preliminary SimDB web services allow users to perform a number of functions such as analyzing, scoring, and clustering trajectories in a number of different ways.

This chapter outlines the progress that has been achieved regarding the individual components of the system. Section 5.1 discusses work that has been completed on the raw data server. The work that has been done on different pieces used by the central server is discussed in Section 5.2. Finally, the status of the analysis functions available to the system are the focus of Section 5.3.

5.1 Raw Data Server

The first piece of the system to be dealt with was the raw data server. Because all things done on the system revolve around the simulation data that are stored in

raw data servers, the raw data server had to be designed and built before other work could be done.

First, a basic protocol was defined for communication with raw data servers. The protocol developed is a plain-text, command-based protocol which supports several commands and many options to those commands. Once the protocol had been defined, the server software itself was implemented over a period of about 6 months. The implementation was done in C for speed and follows the Portable Operating System Interface (POSIX) [33] standards so that it can easily be ported to other operating systems as necessary. As features were added and the server increased in functionality, small modifications were made to the protocol to allow maximum flexibility and consistency across commands. Flex [15] and Bison [14], free implementations of Lex and Yacc, are used to parse command messages sent by clients, and allow for changes to be made to the protocol with very little work.

One of the most important features of the raw data server are native CHARMM DCD support, which allows it to read and understand trajectories in this format. Because of this, the raw data server is able to receive queries for lists of frames and atoms and sends back only the data requested instead of the entire data set. There are several benefits to this. The first is that submitted data do not need to be converted to a neutral format prior to insertion into the system. Researchers can be confident that the data shared are the data that were produced by them. Second, by allowing only the data with which a job is concerned to be sent, these data will reach the processed data servers more quickly, where they will be processed more quickly since the additional data does not need to be dealt with. This savings in transfer time and resource usage benefits both the user in that their jobs will be completed more quickly, and the system in that resources will be used efficiently and made available as soon as possible. The addition of support for other trajectory formats will expand these savings to more of the system.

Another feature that has been given to the raw data server allows the administrator to set limits. The first of these limits is the ability to put the system in a read-only mode in which the data sets on the machine are shared with the system, but no new data can be added, and the existing data cannot be modified. This allows groups that cannot yet allocate much storage for SimDB to still share their data so that others can benefit from it. Administrators can also limit the number of concurrent connections to a server, which is useful if network- or resource utilization limits are enforced. In actuality, a connection to a raw data server requires very few resources, except for perhaps bandwidth; these limits allow the administrator to have complete control of how his or her resources are used. User limits can also be used to prevent denial-of-service attacks. In testing, a barrage of valid and invalid commands as well as random data were sent to the raw data server, and it showed to be resilient against such attacks.

Additionally, a software development library was created to allow programs to be quickly developed that can communicate with raw data servers. The first such program is a command line client that is used by the current SimDB web services for interacting with the raw data server and has also been used extensively to test the raw data server.

The raw data server is nearly complete at this time; however a few additions are planned for future work. These additions are detailed in Section 6.1.

5.2 Central Server

Several of the different elements of the central server have been implemented for use in the SimDB web services. Although not all of these elements are feature-complete at the moment, they do allow different pieces of the system to work together to allow a user to analyze data in a number of different ways.

5.2.1 Web Services

The current SimDB web services allow users to use the system with a few data sets that have been placed on the system. Using these data sets, it is possible for users to examine the capabilities of the analysis functions and output options. Although these demos are useful, using sample data sets does not give the same feel as when a user uses his or her own data. This can be explained by the user's familiarity with their data, so the results of analyses will be more informative to them than those of the demo sets.

In order to facilitate this, users may upload their data to the system and then analyze it. Using a web-based form, a user may submit a trajectory file and an initial structure in PDB format. After being uploaded, the central server is able to determine the type of system that is being modeled, for example a protein or nucleic acid, and the number of residues contained. After uploading, the user has the option of donating the data to the system so that it can be searched for and used by others. Regardless of whether or not the data are donated, the user may then analyze his or her data using analysis functions that work with the determined system type. This functionality has been used with great success by many people.

5.2.2 Prototype Metadata Database

An additional database has been defined and set up to allow for the storage of metadata related to simulation data. Because simulation data files do not contain additional information detailing how the simulation was run, it is very useful to make these data available so that they can be used to further understand the simulation. Currently, this information is added manually via a web form, however more advanced methods will appear in the future that make adding this information easier.

5.2.3 Search Interface

Considerable effort was spent implementing advanced searching of the metadata for the data sets that are stored on the system. With the available search forms, a researcher can search through the data using any combination of attributes such as the type of system being simulated, the environment in which the simulation was run (i.e. explicit solvent using the TIP4P water model [22] and constant temperature), the authors, and many other fields. By taking advantage of this sophisticated search capability, the user can find simulations that deal with systems or tools that they are interested in and examine their dynamics. Once one or more data sets have been found that match the user's criteria, he or she can then directly analyze, score, or cluster the data.

5.2.4 Resource Management

The current system has some rudimentary ability to schedule the execution of jobs that have been submitted. This queueing of jobs allows the system to maintain a reasonable load so that individual jobs are given suitable resources to finish the job quickly. Currently, if a job is found to take more than a few seconds or is waiting behind others, the user is presented with a dialog that allows him or her to enter an email address to which a notification email will be sent when the job has been completed. The user can also continue to check a URL, which will contain options for output format selection when the job has been completed.

5.2.5 Output Filtering

Users have a wide variety in the formats that can be used to represent the results of analyses on their data. The data can be exported in a standard tabular- or delimited representation for importation into a database or used with external

analysis tools. Additionally, the user can plot the resulting data using standard figures or Ramachandran plots, which show the distribution of dihedral angles for the given structure. These plots can be exported in multiple graphics formats as well, which easily allows them to be included in papers and presentations.

5.2.6 Additional Capabilities

As the central server is the most complex component of SimDB, some work remains to be done before the SimDB reaches its full potential. This includes the addition of user authentication and authorization, the creation of tools which allow research groups to easily add their data to the system, and sophisticated resource management capabilities that utilize the available nodes in the network as efficiently as possible. These and other items are discussed more thoroughly in Section 6.2.

5.3 Analysis Functions

Several analysis functions have been implemented and are available on the SimDB web service. These analyses range in complexity from straightforward to fairly complex. For proteins, users may select among radius of gyration, coordinate RMSD, secondary structure analysis, and backbone torsion calculation. For nucleic acids such as DNA, users may select functions such as radius of gyration, coordinate RMSD, backbone torsion calculation, ribose pseudoration analysis, and helical analyses. Each of these functions also has user-definable parameters which allow the user to specify exactly which parts of the structure he or she wishes to analyze. For example, with coordinate RMSD calculation, the user can choose the first conformation, the last conformation, an average conformation, the starting structure given with the trajectory, or another conformation as a reference for the calculation. After this, he or she may choose which segments to analyze and the specific residues within those seg-

ments. Finally, the user may choose whether to include all atoms in the calculation or the heavy ones, those in the side chain, those that make up the backbone, or the alpha- or beta carbons. Similar arguments exist for the other functions as well, and allow the user to define the analysis to be run at very fine detail.

In addition to these types of analysis, SimDB also currently supports MMPB/SA scoring, which calculates free energies of binding or absolute free energies of molecules, and conformational clustering, which clusters the conformations in a trajectory. As with the trajectory analyses, the researcher may customize parameters to these functions in order to get exactly the results required.

To allow for analysis functions to be added to the system in an efficient and standardized way, work has been done designing a function database. In order to design the function database, a function had to be defined in general terms so that the implementation would allow for all possible functions. This design process has led to an initial implementation of the function database which is able to build web-based forms that allow a user to see the various parameters to the function and set them, just as with the current hard-coded analyses available on the SimDB web services. The function database can also check calls to defined functions to ensure that all required parameters have been defined and all arguments are of the correct data type. Finally, based on a call to the function, the function database can assemble code in the proper order for execution and then execute the analysis.

Much remains to be done with the analysis portion of SimDB. Section 6.3 details work that will be done in the future to improve analyses on SimDB.

Chapter 6

Future Work

Although much has been accomplished in the past two years, a number of refinements must be made before SimDB meets its goals of being a powerful, flexible, and scalable tool for researchers who deal with molecular models and simulations. Fortunately, SimDB has momentum, and continuing efforts will assist in achieving these goals in the near future.

The tasks that remain to be done are detailed in the following sections. Most of these tasks revolve around implementing new and innovative features to SimDB that will increase its power and usefulness to researchers. Section 6.1 discusses what remains to be done on the raw data server. Section 6.2 outlines the work that is planned for the central server. As the central server is comprised of many smaller pieces, work on them is discussed individually. Future work for the processed data server is discussed in Section 6.4. Should the centralized design of the central server prove problematic once the system is implemented, possible solutions are detailed in Section 6.5. Finally, an application development interface (API) for interacting with the system as a whole is discussed in Section 6.6.

Since it is an effort that is being attacked from many different angles, SimDB currently lacks a complete uniformity between the components as far as interacting

with the system is concerned; however once they are further developed and begin to inter-operate more, these differences will disappear.

6.1 Raw Data Server

The raw data server has been tested extensively and used for several months as part of the current SimDB web services. During this time, it has been found to be reasonably stable, operating without failure or inconsistency.

Support for trajectory formats other than CHARMM DCDs will be the focus of future work on the raw data server, so that it can successfully read, write, and parse as many formats used in the community as possible. This flexibility may also allow for conversion between the formats, which may become useful so that trajectories can be analyzed using functions that have only been defined for other formats. Further, while supporting many formats is very convenient in that researchers' data are maintained in their original form, developing and supporting a common format may be useful so that analysis functions only need to be developed for the common data format. The least obstructive option may be to continue to support many formats, but also support a common format, as well as provide conversion routines to and from that format.

As user authentication and authorization are added to the central server, capabilities will also need to be added to the raw data servers to support these control mechanisms. This may simply be through a token given by the central server to a client that allows access the data in the raw data servers or more finely-grained access control mechanisms that control access to trajectories on a per-user or per-group basis. These permissions would essentially make protected data sets invisible to those without proper credentials by prohibiting any commands to be issued that deal with the data on the server.

6.2 Central Server

As the "brains" of the system, the central server performs many functions in order to combine the available resources and data into one system. Although some of the functions of the central server such as metadata management, searching, data deposition, and submitting jobs have been implemented, there are numerous more that have yet to be implemented before SimDB realizes its full potential and ease-of-use. Because of this, a considerable amount of the work that is currently being done, as well as in the near future, focuses on different components used by the central server.

6.2.1 Tools for Data Deposition

One of the first things that needs to be done in order to promote the growth and use of SimDB is to establish a set of easy-to-use tools which allow researchers to add their simulation data to the system. Although these capabilities exist in the current web services, the process of adding data to the system is less than straightforward. More advanced deposition tools will support the different trajectory formats and make use of their metadata when the data is added to the system, so that the user does not need to manually enter this information with each trajectory. These tools will also support batch depositions, so research groups can quickly make all of their data available.

6.2.2 Sophisticated Resource Management

In order to allow SimDB to expand easily and take advantage of more nodes, sophisticated resource management capabilities need to be added. More specifically, these capabilities would be concerned with maintaining information about the nodes connected to the system, the resources that are being used at each node at a given

time, and the resources that are available on each node at a given time. Additionally, this resource management software could establish a connection between two nodes to measure capacity of the link between them. This information would be used in conjunction with predefined estimates regarding an analysis function's cost to determine which nodes should be used to handle which jobs in order to minimize the time required to complete jobs as well as maximize the number of jobs that can be run concurrently. The resource management software may also establish parallel downloads of data in order to facilitate faster transfers and the use of multiple processing nodes and moving jobs mid-processing to more capable nodes in order to facilitate shorter processing times. An additional duty for the resource management software would be to determine on which raw data servers new data sets would be placed. Finally, data mining concepts may be employed to estimate popular queries that may occur in the future so that the data can be processed beforehand or at least transferred to processing node in order to avoid transmission delays.

6.2.3 Access Control Mechanisms

Eventually, user access control mechanisms will be added to the central server so that the data and resources available on SimDB can be controlled. Although SimDB is primarily a system that will be open to research groups around the world, the ability to control access to certain trajectories would enable researchers to collaborate and take advantage of the available tools privately before the work is camera-ready. Techniques such as Access Control Lists (ACLs) may be used to give finely-grained permissions to each data set available and the processed data that is a result of analyses on these sets. These control mechanisms will also have to be added to the other components of the system, but the main access control mechanisms will be managed by the central server.

6.2.4 Additional Output Options

Although the current SimDB web services allow the user to choose many formats to display output in such as plain ASCII text, delimited, Ramachandran maps, and .gif and postscript plots, additional output options will be added to meet the demands of users. These additions include more control over plots, SQL statements for insertion into databases, tables, and other types. These features are not necessary for the operation of SimDB; however they contribute toward making it a powerful tool that can be used for all stages of work on simulation data.

6.2.5 Central Server Development Library

A software library will also be developed for use by SimDB and custom clients that allows users to perform all tasks associated with the central server from searching the available data sets to submitting analysis jobs to choosing the output format for the results.

6.2.6 Data Mining Capabilities

Finally, once the system has become more mature, additional use of data mining may be used in the system to support more advanced features. This may include the ability of the system to find other simulations that it determines to be of interest to a user based on a currently-selected simulation or past analyses. As little is known about how to perform such data mining tasks, it may result in considerable research in the future, resulting in very powerful and unique features.

6.3 Processing Nodes and Analysis Functions

Processing nodes are another important component of SimDB, and will be given a considerable amount of attention in the near future. This work will aim to make

the operations performed by processing nodes as powerful and as efficient as possible, making SimDB an attractive platform for researchers.

Although the current SimDB web service is already a useful tool, it has been observed through testing that some of the analysis functions available can occasionally run more slowly than their traditional counterparts. As a reaction to these findings, the current analysis tools are being rewritten to improve their efficiency. This may also be due to the fact that analysis using the entire SimDB architecture can be more complex than one-time analyses performed using traditional tools. Aside from offering unique analysis options to researchers, a further goal of SimDB is to support more common analyses. However, to promote the use of SimDB for research, these common analysis options must be as fast if not faster than current solutions. The more quickly analyses can be run on data sets, the faster researchers can use the resulting data and begin further work. Consequently, if jobs are completed more quickly, this will have the added benefit of freeing up resources in the system, which can then be allocated to other jobs.

In addition to improving upon the current analysis functions, a large number of functions will be added to the system to support a wide variety of analyses. In order to accomplish this, it was decided that functions available in the system needed to be defined in a powerful yet flexible way, so that each does not need to be hard-coded to deal with the different ways in which it is called or with different types of systems being analyzed. To facilitate this, a function database and related software are being developed so that analysis tools can be defined in a uniform way, and using these definitions, all tasks related to the functions can be automated. These tasks include creating interfaces with which the user can enter parameters, validating arguments to the functions, executing commands, and all other functionality. Using this database, a user can select from all functions that can be used on the currently-selected data instead of having to call specific functions explicitly. This flexibility will

enable researchers to use tools that that they may not have previously had at their disposal, offering them beneficial research prospects. Most of the operations and the design of the function database have been completed. What remains to be done is porting the currently hard-coded analysis functions to the database and adding new analysis functions. The ease with which new analysis functions will be added to the database will allow SimDB to offer research a large array of extremely practical and unique tools to researchers.

One of the main future goals for SimDB is to support comparative analysis, or analysis of more than one trajectory. Once SimDB has been matured, more time can be spent determining ways in which one could compare the dynamics of different systems and obtain new and useful information about them. Because the function of a given biomolecule has been shown to be partially determined by its dynamics [32], discovering similarities in the dynamics of a structure whose function is not known and one for which more is known could yield new insights into how these structures operate within the body. Since the tools to easily perform such research have not existed previously, comparative analysis is an excellent example of a tool that would help magnify SimDB's extreme usefulness for research.

6.4 Processed Data Server

The processed data server is a key component of SimDB; however the operation of SimDB does not depend on any processed data servers being online, so it will be one of the latter features to be implemented.

The behavior and functionality of the processed data server will be similar to that of the raw data server, so it is likely that the code from the raw data server will be re-used when implementing the processed data server. First, a command-response protocol will be defined, similar to that of the raw data server, and then functionality

will be added to support the protocol.

Additionally, a library for dealing with processed data servers will be implemented in Perl [9] or a similar language so that programs that interact with processed data servers can be developed quickly. This will also include a command-line client for processed data servers that functions very similarly to that which was developed for raw data servers.

6.5 Moving to a Decentralized Architecture

It is not currently believed that the centralized architecture used by SimDB will lead to problems with reliably and throughput, but how well the central server performs as the system grows will be seen as soon as more nodes are deployed in the near future. If the current centralized architecture is found to in a sub-optimal fashion as far as system efficiency and throughput are concerned, then new designs may be employed. Although a large number of design possibilities exist, two stand out. The first, detailed in Section 6.5.1 simply replicates the data on the central server on several nodes. The second borrows the concept of structured networks from the peer-to-peer community, and is discussed in detail in Section 6.5.2.

6.5.1 Replicating the Central Server

The most straightforward possibility would be to use multiple central servers that share system data and are accessed in a round-robin fashion in order to balance the load. This design would also allow the system to continue functioning should one of the central nodes fail. In fact, the system could continue if many nodes failed as long as at least one node remains online. This approach could encounter problems with synchronization if each node maintains the complete system information. It would also be possible to have the nodes read data from another server, but this would re-

introduce the problems that occur with a central server, namely the bottleneck and single point of failure problems.

6.5.2 Using a Structured Peer-to-Peer Network

Another approach that could be taken to increase the redundancy and throughput of the central server would be to have each central server maintain a different subset of the system information. These data could then be shared among the central servers using techniques developed in the peer-to-peer community. One particular class of peer-to-peer networks that would lend themselves effectively to this problem would be structured, decentralized networks such as Chord [29], CAN [37], and PASTRY [38].

Structured networks maintain a distributed hash table (DHT), which is similar to standard hash tables, which store key/value pairs. With DHTs, however, the key space is divided among the nodes so that each node is responsible for its own subset of the key space, and the union of each of these subsets is the entire key space. In addition to storing values related to the subset that it is responsible for, each node may additionally store replicas of another node's values. This redundancy allows the network to continue to maintain the entire key space in the event of node failure. The hash function that is used to hash values to the keys in the system is also used to identify the nodes. Nodes then are responsible for storing values whose hash matches their id. Depending on the implementation of the network, queries then follow this hash function to efficiently locate the node on the network that stores the value associated with the query.

Using this structured approach, the central nodes in the system would evenly distribute system information among themselves. By maintaining a separate DHT for each type of data stored, such as metadata or the location of data sets, when a query is issued to one of the nodes, the node could use the querying capabilities of

the chosen network to locate the node on which the matching values reside. Once the data has been found, the server that received the query could then retrieve the appropriate data and then proceed to handle the user's query as is done currently.

In addition to using this technique for just the central server, structured peer-to-peer networks could be used for each of the other components in the system. This means that the raw data servers would arrange themselves in an overlay network, and the processing nodes and processed data servers would do the same, creating a system of several overlays. One benefit of this would be that the central server(s) would no longer need to maintain information about where each data set is located in the network. The query manager could then simply query one node in the overlay, which would then locate the content among the other nodes, and the location of the data could then be returned, allowing the query manager to continue the query knowing the location of the data.

6.6 SimDB Application Development Interface

As the different pieces of the system become completed, software packages for interacting with them need to be developed, not only for use within SimDB, but in order to support interacting with SimDB in different ways. Such tools with consistent interfaces will be bundled together into a development toolkit and allow for many tasks, such as submitting new data easily, submitting jobs to the system, performing custom analyses, and using special output filters. This would allow researchers to use SimDB in ways that maximize its usefulness to them, while maintaining the integrity of the system.

Chapter 7

Conclusion

SimDB provides a unique and useful resource for those interested in studying the dynamics of biomolecules, and it creates possibilities that have not previously existed for research groups without considerable resources at their disposal. Because SimDB is built as a distributed computing platform, it is able to support the large amount of storage requirements and processing capabilities required for such an undertaking, which would not be possible otherwise.

A good amount of progress has been made implementing SimDB. Several of the components have been implemented and already offer many features. For some time already, a website has been available which uses these completed components to offer a limited amount of services to be included in future versions of SimDB.

Although much work and refinement remains to be done on SimDB, the project shows promising momentum, and more powerful and useful features will be added in the near future.

Bibliography

[1] Matin Abdullah. Simdb - a grid software environment for molecular dynamics simulation and analysis: Design and user interface. Master's thesis, University of Houston, December 2002.

[2] M.P. Allen and D.J. Tildesley. *Computer Simulation of Liquids.* Oxford Science Publications, 1987.

[3] Dennis A. Benson, Ilene Karsch-Mizrachi, David J. Lipman, James Ostell, and David L. Wheeler. Genbank: update. *Nucleic Acids Research*, 32(Database-Issue):23–26, 2004.

[4] H.J.C. Berendsen, D. van der Spoel, and R. van Drunen. GROMACS: A message-passing parallel molecular dynamics implementation. *Computer Physics Communications*, 91:43–56, 1995.

[5] H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, and P.E. Bourne. The protein data bank. *Nucleic Acids Research*, 28:235–242, 2000.

[6] David L. Beveridge, Gabriella Barreiro, K. Suzie Byun, David A. Case, Thomas E. Cheatham III, Surit B. Dixit, Emmanuel Giudice, Filip Lankas, Richard Lavery, John H. Maddocks, Roman Osman, Eleanore Seibert, Heinz Sklenar, Gautier Stoll, Kelly M. Thayer, Peter Varnai, and Matthew A. Young. Molecular dynamics simulations of the 136 unique tetranucleotide sequences of dna oligonucleotides. i. research design and results on d(cpg) steps. *Biophysical Journal*, 87:3799–3813, December 2004.

[7] B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. Swaminathan, and M. Karplus. CHARMM: A program for macromolecular energy, minimization, and dynamics calculations. *Journal of Computational Chemistry*, 4:187–217, 1983.

[8] R.M. Daniel, R.V. Dunn, J.L. Finney, and J.C. Smith. The role of dynamics in enzyme activity. *Annual Review of Biophysics and Biomolecular Structure*, 32:69–92, October 2003.

[9] Larry Wall et al. Practical extraction and report language, http://www.perl.org.

[10] Michael Feig, Matin Abdullah, Lennart Johnsson, and B. Montgomery Pettitt. Large scale distributed data repository: Design of a molecular dynamics trajectory database. *Future Generation Computer Systems*, 16:101–110, 1999.

[11] Michael Feig, Matin Abdullah, Lennart Johnsson, and B. Montgomery Pettitt. Molecular dynamics trajectory database design manual, April 1999.

[12] Michael Feig, John Karanicolas, and Charles L Brooks III. MMTSB tool set: Enhanced sampling and multiscale modeling methods for applications in structural biology. *Journal of Molecular Graphics and Modeling*, 22:377–395, 2004.

[13] Michael Feig, John Karanicolas, and Charles L Brooks III. Recent advances in the development and application of implicit solvent models in biomolecule simulations. *Current Opinion in Structural Biology*, 14:217–224, 2004.

[14] GNU Bison Parser Generator. http://www.gnu.org/software/bison/.

[15] GNU Flex Lexical Analyser Generator. http://www.gnu.org/software/flex/.

[16] Gnutella. http://www.gnutella.com.

[17] Object Managment Group. Common object request broker architecture (CORBA), 1991.

[18] Konrad Hinsen. The molecular modeling toolkit: A new approach to molecular simulations. *Journal of Computational Chemistry*, 21:79–85, 2000.

[19] William Humphrey, Andrew Dalke, and Klaus Schulten. *Journal of Molecular Graphics*, 14:33–38, 1996.

[20] Thomas E. Cheatham III. ptraj, http://www.chpc.utah.edu/ cheatham/ptraj.html.

[21] Internet2. http://www.internet2.edu.

[22] W.L. Jorgensen, J.Chandrasekhar, J.D. Madura, R.W. Impey, and M.L. Klein. Comparison of simple potential functions for simulating liquid water. *Journal of Chemical Physics*, 79:926–935, 1983.

[23] Martin Karplus. Molecular dynamics simulations of biomolecules. *Accounts of Chemical Research*, 35:321–323, 2002.

[24] Lewis E Kay. NMR methods for the study of protein structure and dynamics. *Biochemistry and Cell Biology-Biochimie et Biologie Cellulaire*, 75:1–15, 1997.

[25] Kazaa. http://www.kazaa.com.

[26] Seonah Kim. Database of simulation data: Simdb design and implementation. Master's thesis, University of Houston, May 2003.

[27] R. Lavery and H. Sklenar. Curves 5.1: Helical analysis of irregular nucleic acids, 1996.

[28] Andrew R. Leach. *Molecular Modelling Principles and Applications.* Prentice Hall, 2 edition, 2001.

[29] Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *ACM SIGCOMM 2001*, San Diego, CA, September 2001.

[30] Napster. http://www.napster.com.

[31] Abilene Backbone Network. http://abilene.internet2.edu/.

[32] Fritz G Parak. Proteins in action: the physics of structural fluctuations and conformational changes. *Current Opinion in Structural Biology*, 13:552–557, October 2003.

[33] IEEE Computer Society Portable Application Standards Committee (PASC). http://www.pasc.org.

[34] D.A. Pearlman, D.A. Case, J.W. Caldwell, W.R. Ross, T.E. Cheatham III, S. DeBolt, D. Ferguson, G. Seibel, and P. Kollman. AMBER, a computer program for applying molecular mechanics, normal mode analysis, molecular dynamics and free energy calculations to elucidate the structures and energies of molecules. *Computer Physics Communications*, 91:1–41, 1995.

[35] Lavanya Prabu. Implementation of an architecture for a simulation database. Master's thesis, University of Houston, July 2002.

[36] RCSB FTP Mirror Procedure. http://www.rcsb.org/pdb/ftpproc.final.html.

[37] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard M. Karp, and Scott Shenker. A scalable content-addressable network. In *SIGCOMM*, pages 161–172, 2001.

[38] Antony Rowstron and Peter Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, Heidelberg, Germany, November 2001.

[39] SimDB Web Analysis Services. http://simdb.bch.msu.edu.

[40] Kaihsu Tai, Stuart Murdock, Bing Wu, Muan Hong Ng, Steven Johnston, Hans Fangohr, Simon J. Cox, Paul Jeffreys, Jonathan W. Essex, and Mark S.P. Sansom. BioSimGrid: Towards a worldwide repository for biomolecular simulations. *Organic and Biomolecular Chemistry*, 2:3219–3221, 2004.

[41] Seiichiro Tanizaki and Michael Feig. A generalized born formalism for heterogeneous dielectric environments: Application to the implicit modeling of biological membranes. *Journal of Chemical Physics*, 122, 2005.

[42] Guido van Rossum et al. Python programming language, http://www.python.org.