INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI®

WEB SUPPORT FOR AUTOMATED ANALYSIS OF DNA SEQUENCES

Wael Hassan

A Major Report
in
The Department
of
Computer Science

Presented in Partial Fulfillment of the Requirements
For the Degree of Master of Computer Science
Concordia University
Montréal, Québec, Canada

April 2000

Abstract

Web Support for Automated Analysis of DNA Sequences

Wael Hassan

Bioinformatics is a developing science. It contributes greatly to gene research projects like the human genome project and is extremely helpful in drug discovery. Future advances in bioinformatics will be enhanced by how fast gene sequences are produced and selected. This report describes the internet infrastructure for analysing DNA sequences developed for the Centre for Structural and Functional Genomics at Concordia University. It describes the architecture, the technologies, and integration techniques used.

We make use of a web server and three machines to build the system. The system integrates Unix and Perl scripts as well as the Apache web server with its security modules. There are three subsystems: clone, pipeline, and web search. Clone takes care of system storage, backup and recovery. Pipeline extracts results from the sequencers to submit to the search engine and stores the results. Web search accepts search requests using a web interface to a Perl CGI. The searches use BLAST against the Non-Redundant (NR) sequence database from the National Center for Biotechnology Information (NCBI).

Acknowledgements

I would like to express my gratitude to my supervisor Dr Gregory Butler for his guidance and support during the course of my research. Special thanks to Dr Tsang and Marielle Carpenter for allowing me the opportunity to participate in their research group. Best wishes to Dr Lam for reviewing this report. I would like to thank my friends Nour Kadri and Ali Hussieni for their support. To my brother Bilal by whom my life and future extends. I also dedicate it to my uncle Karim, my sisters Lara, Malak, and Vera. Last, but not least, I dedicate this to my father Ali and my mother Fatima for being the greatest parents one could have.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction

Computer science has proved itself to be a necessity in most empirical and theoretical sciences. Biological science is using the computation and storage power to filter out experiments, minimise time, and to produce human understandable results. The last decade has seen a veritable explosion in the amount of raw information generated by molecular biologists worldwide. Bioinformatics is the science of developing computer databases and algorithms for the purpose of speeding up and enhancing biological research. Bioinformatics is being used most noticeably in the Human Genome Project, the effort to identify the 80,000 genes in human DNA. There is another definition of bioinformatics, at the Boston university bioinformatics page[1]: "Bioinformatics, is an intersection of information technologies and applied mathematics with molecular biology/genetics, currently consists of the following research activities:

- The Management and Integration of Biological Information.
- Gene Mining.
- Gene Regulation Network Analysis and Drug Targeting.
- Computer Aided Drug Design."

The field has given rise to numerous important discoveries. Hopes are still very high that it will reveal many more secrets in the near future. Some people downgrade the science of bioinformatics to the process of selecting the right kit from the available

software tools. The view that we take for this project is that we will be selecting the right kit to do our gene matching, yet we will be designing the production pipeline from the sequencer to the Java widgets in the web page interface.

1.2 Motivation

Concordia's research centre, like other centres, uses a semi-manual process to gene search where the biologist has to do experiments, attain results using the sequencer machine. Our main interest is in "Aspergillus Niger" , a fungus. From the sequencer, data files are exported and burnt on a CD to be backed up. The databases on the sequencer are in an MS Access format. The front end of the system will be a web-based client, that will retrieve information from the sequencer to a Unix host. Once there, various tests such as cleanup, file formatting, and searches can be executed. Having a web client, means that we will need a web server. Given that the center wants to serve its clients securely, we had to install a secure server that supports authentication. The authentication is needed so that customers are being identified and appropriate resources are allocated. Whether a new or returning customer, every customer should have a profile.

1.3 System Description

Our system provides gene matching capability against the National Center for Biotechnology Information (NCBI) Non Redundant (NR) database. The system serves two kinds of clients: researchers at Concordia's lab and anybody who wants to access the service after registering to our secure web site. Single or multiple search requests can be made via the web page. Only Concordia researchers can submit searches to the batch processes using our main subsystem called *pipeline*. Pipeline periodically pulls its input data from the gene sequencer machine. The results of all subsystems are stored in SQL retrievable format. Clone is our backup and system recovery option. It creates clones of any machine to a local disk or to a remote network site. Thus our system is subdivided into three subsystems: Clone, Pipeline, and Web Process. Pipeline performs both transport and batching querying of requests. Web search accepts queries via web interface. Clone backs up the system.

1.4 Background

The first book on gene manipulation was written about fifteen years ago. Being a new field with a lot of expectations, many people are interested in participating. Bioinformatics, where computers are applied to biology, is needed because of the unprecented growth in quantity and diversity of the biological information. Some of the applications of gene research are in drug discovery and development.

1.4.1 Bioinformatics and Genes

Gene sequences are the codes which direct the production of proteins that in turn regulate all life processes. Thus, in principle, determination of these sequences can lead to a much fuller understanding of many biological processes. Our understanding of the role and function of most proteins is incomplete, and often non-existent. Thus, there is a great need to understand what protein each gene produces and to determine the role of each protein.

1.4.2 Bioinformatics in Drug Discovery and Development

Since most of the gene sequences are unknown, drug development is restricted to a small fraction of the possible targets. The Human Genome sequencing project will lead to a preliminary description of all human genes. This will enhance the number of potential targets for drug development. Drug developers will have a large number of genes, but they would not be of much help if no information about the genes was available. This information will help in select the best target. Thus, bioinformatics helps drug developers selecting their targets by acquiring and presenting all available information to the researcher

1.5 Related Work

We will overview legacy data collection systems which can be classified according to several criterion[2]:

- Data Acquisition.
- Data Analysis.

- Data Management systems.

1.5.1 Data Acquisition Systems

Data acquisition is required at all research labs that are generating large amounts of data. Laboratories require a local informatics support for acquiring data efficiently. Example of these system include:

Inventory Control Systems

A large research centre may require several hundreds of thousands of reagents, gels, and other materials. Given that manual tracking would be impractical and inefficient, automated inventory control systems are needed.

Reagent Manipulation Software

Robotic systems are now required to carry out high-volume, high precision laboratory manipulations in research, every day computer support for robotics will become increasingly important.

- Sequence Production Software: Presently, computer systems are required in almost all aspects of sequence generation and assembly.
- Image Processing: Most of data begins as images. Interpreting gels, reading filters, and many other steps require computer tools for automation and optimisation.

1.5.2 Data Analysis Systems

Data can not be analysed efficiently without computer systems. Studying sequences, predicting protein structures, and comparing genomes on an extensive scale all require additional informatics tools, such as:

Sequence Analysis Software

Sequence analysis is perhaps the best known, best established area of bioinformatics. Performing alignments, detecting homologies, identifying coding regions, extracting

features, and other computerised analysis of sequences are now so commonly performed as to be routine. At the same time, sequence analysis is a multi-faceted and biologically profound area of research, demanding much continued work.

Protein Folding Software

Genetic information is transformed into proteins whose functional specificities are determined by their three-dimensional shapes. One significant goal is to be able to predict protein structure and function from the amino-acid sequence for the protein.

Map Assembly and Integration Software

With larger maps (composed of different fundamental kinds and combinations of data) being generated, computation plays an increasingly central role in their assembly and integration.

Physical Mapping and Config Assembly Software

Assembling n clones into an ordered physical map usually involves generating a large $(n \times n)$ matrix of comparisons, then deducing a possible order from the matrix. With any reasonable size n, performing this analysis is manually impossible.

Genetic Mapping Software

Software systems play a key role in the analysis of genetic mapping data.

Comparative Genomics Tools

As the Genome project matures and large amounts of genomics information are available for a number of species, comparative genomics will emerge as an active area of study.

Classification Software

Extracting features from DNA sequences, placing proteins into gene families, and tracking protein motifs are all key activities in genome research and all need computer tools.

1.5.3 Data Management Systems

This section comes directly from [3]. The bioinformatics labs are generating information that cannot be accommodated by traditional publishing. Ready access to these findings are essential for interpreting current experiments and planning future work. Now, electronic data management and publishing systems are increasingly crucial components of Genome research. These systems range from highly specialised databases supporting local research projects to general databases that support the entire community.

Local Databases

Local databases are usually developed at one site and designed to handle specific, local needs. They are usually a closed resource, available only to local researchers, and containing both raw data and refined information. Often they are also involved in local inventory control. Tightly integrated into local bench research, they must be flexible and capable of rapidly tracking changes in local experimental protocols. The requirement for flexibility and responsiveness often exceeds a need for robustness and general applicability. Formal user-support systems (in the form of extensive manuals, regularly staffed help desks, etc.) are rare, but informal support is crucial – local users must be able to work with the system. Most importantly, local systems must quickly meet the specific, idiosyncratic requirements imposed by specific protocols used by local researchers.

Collaborative Databases

Collaborative databases are conceptually in the middle ground between local and community systems. Information is collected from a larger set of collaborating researchers, perhaps those working on a particular chromosome. Again, both raw data and refined information are included, with an increasing emphasis on integrating findings from several laboratories. Requirements for robustness are greater than with local systems, since the system may be used by some who do not have access to local computer expertise. More adequate user support is also needed. Collaborative databases are frequently discussed and proposed, but at present it is difficult to identify any extant examples. Many of the problems associated with building such databases seem to be sociological, rather than technical.

Community Databases

Community databases are shared resources, open to the entire research community. Although they have traditionally emphasised refined information over raw data, the trend now is for community databases to play an increasing role in making raw data (e.g., the underlying data on which a published genetic map is based) available as a research resource. Community databases must be robust, since the majority of their users will be located off site. Formal user support, in the form of manuals, help lines, regularly staffed help desks, and training are required. Because these systems must meet the consensus needs of the entire community, care must be taken in their design to ensure that the databases are sufficiently flexible and robust to address the needs of different user communities. Since users need to integrate findings from several different community databases, each community database should be designed as a component of a larger information infrastructure for the biological sciences. Specifically, community databases should recognise the biological interdependence of information in multiple databases and should provide support for integrated queries involving multiple databases. To this end, they should be built from standard components and must be well documented.

1.6 Overview of Work

This report describes a system implemented at the CSFG lab at Concordia University. The system is divided into three parts: the pipeline, the websearch, and the backup system. The pipeline is the subsystem responsible for transferring of bulk daily or weekly gene sequences generated on one machine called Sequence, then it will perform similarity searches for these gene sequences using a tool called BLAST against NR, the Non-Redundant sequence database. Where the pipeline subsystem runs as a daemon, the web search subsystem has an HTML interface that allows users to upload sequences to match against several databases including NR. The backup system is meant for doing regular backups for databases and user accounts, and to set up a machine that will act as a redundant server. If the main server ever crashes the backup system can take over.

1.7 Report Organisation

This report is organised as follows: the first section talks about motivation, history and related work. Section 2 introduces the building blocks, hardware (sequencer), packages (BLAST), tools (Apache), databases (NCBI), file formats(FASTA), requirements (security), and it provides solutions for the problems faced. Section 3 describes the system topology and design. Section 4 presents the design and implementation specifications. It also includes a set of results attained. Finally, in section 5 we conclude our work and present future enhancements.

Chapter 2

Tools and Building Blocks

2.1 Introduction

Our analysis to the process model shows that we need a distributed system to satisfy the goals for the project. We have several goals:

1. Search the gene database for matches of our extracted sequences.
2. Accept searches from web site and submit them to a blast.
3. Back up the system.

In this chapter, we will go over the machines used, and tools that you need to install for this system to work. The hardware section will talk about, Sequence, our sequencer machine, Discrete the search machine, and Gene the backup server. In addition, we will discuss details on the software components which include, code, the Apache Web Server, BLAST the search engine, and *MySql* the database. Moreover, we will present difficulties that system administrator may face, and the workarounds.

2.2 Hardware

There are three machines upon which the system will be deployed. Sequence, the sequence generator, Discrete the Web Server and search engine; and Gene the backup machine.

2.2.1 The Search Engine Machine

The CSFG Lab database will be installed on Discrete, a machine made by Digital running OSF version 1.4.0B. This machine has an Apache1.3.6[3] Web-Server installed.

2.2.2 The Sequencer

Made by Beckman instruments, the CEQ 2000 sequencer automates the process of

- DNA sequencing.
- Fragment Analysis.

It reduces manual intervention by a large factor. The normal process normally includes[4], reaction preparation, denature, load Samples, separate/detect/analyse, database interaction. This process is reduced to:

1. Reaction Preparation, loading the material into the sequencer.
2. Database Interaction, explore results analysed and figure out if the sequences seem to be erroneous or not.

This machine has multiple analysis capabilities:

1. Provides automatic base calling and is able to produce a report after each row of eight samples.
2. Changes parameter set for base calling algorithm.
3. Audio playback of selected DNA segments
4. Call scores and quality values generated for each sequence for base call assurance.
5. Exports data analysis exporting (.scf, FASTA, PHRED, SEQ, Tab-Delimited)

Most of the graphical screens are used to monitor the results. The biologist has the ability to sanitise the output as many things could go wrong. For our project's purpose we are interested in the FASTA output the machine produces. [Please see Appendix for description of FASTA Sequences]. The FASTA output is stored in clear text files. These files are the ones we are interested in, they will be transferred and stored in the our in-house database of known gene sequences.

2.3 Software

In this section, we will talk about the databases used and the language of implementation. In Figure 1, you can see an image of a sequence chart provided by the CEQ-2000. It is attached to a Windows NT Box, and its results are in MS Access format.

2.3.1 Databases

The project in it's current state produces text search results. This can be easily modified to a *MySql* database format. Moreover, the backup system utilises an *MySql* database. Apache's *dbmmanage* creates a Unix database that keeps track of users and groups that are allowed access to the system.

Clone Database

The database of clone subsystem uses a *MySql* format. There are reasons for choosing this specific database.

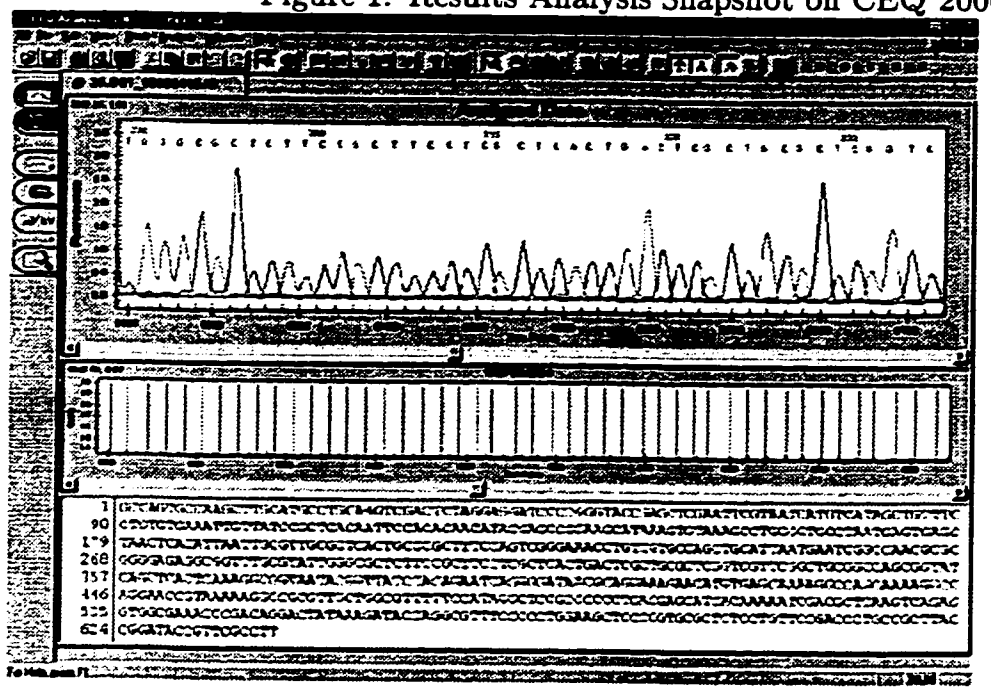
It is a true multi-user, multi-threaded SQL database server. SQL (Structured Query Language) is the most popular and standardised database language in the world. *MySql* is a client/server implementation that consists of a server daemon *MySql* and many different client programs and libraries.

SQL is a standardised language that makes it easy to store, update and access information. For example, you can use SQL to retrieve product information and store customer information for a web site. It is also fast and flexible enough to allow you to store logs and pictures in it.

2.3.2 Perl

Perl is an interpreted high-level programming language developed by Larry Wall in 1987. Since then, it has become the premier scripting language of the Web, as most CGI programs are written in Perl. However, Perl is widely used as a rapid prototyping language and a "glue" language that makes it possible for different systems to work well together. Perl was originally developed for Unix but you will find it on a wide range of platforms. Because Perl is an interpreted language, Perl programs

Figure 1: Results Analysis Snapshot on CEQ 2000



are highly portable across systems. You can down-load Perl for free from several site like (http://www.perl.org, http://www.perl.com).

Perl is recognised as well for its regular expression power. People say that if something can not be parsed in Perl it means that it can never be parsed by any other scripting language . In our system, all the parsing was done using perl regular expression.

2.4 BLAST

BLAST (Basic Local Alignment Search Tool) is a set of similarity search programs designed to explore all of the available sequence databases regardless of whether the query is protein or DNA. The BLAST programs have been designed for speed, with a minimal sacrifice of sensitivity to distant sequence relationships. The scores assigned in a BLAST search have a well-defined statistical interpretation, making real matches easier to distinguish from random background hits. BLAST uses a heuristic algorithm which seeks local as opposed to global alignments and is therefore able to detect relationships among sequences which share only isolated regions of similarity (Altschul et al., 1990). For a better understanding of BLAST you can refer to the BLAST Course which explains the basics of the BLAST algorithm (http://www.ncbi.nlm.nih.gov/BLAST/tutorial/Altschul-1.html).

There is also a simple BLAST tutorial located under the Education link in the side-bar of the NCBI home page (http://www.ncbi.nlm.nih.gov/).

2.4.1 How Many BLAST Types Do We Have?

There are several types of BLAST stand-alone programs. Here is a list and a short description. For our purposes we were only interested in blast-x.

blastp compares an amino acid query sequence against a protein sequence database

blastn compares a nucleotide query sequence against a nucleotide sequence database

blastx compares a nucleotide query sequence translated in all reading frames against a protein sequence database

tblastn compares a protein query sequence against a nucleotide sequence database dynamically translated in all reading frames

tblastx compares the six-frame translations of a nucleotide query sequence against the six-frame translations of a nucleotide sequence database. Please note that tblastx program cannot be used with the NR database on the BLAST Web page.

2.4.2 How To Install?

BLAST 2.0 can be run locally as a full executable and can be used to run BLAST searches against private local databases, or against down loaded copies of the NCBI databases. BLAST binaries are provided for IRIX6.2, Solaris2.5, DEC OSF1 (Ver.4), and Win32 systems. BLAST 2.0 executables may be found on the NCBI anonymous FTP server (ftp://ncbi.nlm.nih.gov) under /blast/executables/. Please read the README file in this directory for more information. You can also find this document on the web page http://www.ncbi.nlm.nih.gov/BLAST/newblast.html#standalone. There is also some information on setting up the programs at the NHGRI site at:
http://genome.nhgri.nih.gov/blastall/blast_install

2.4.3 List of Databases for BLAST

These are the databases that can be searched by BLAST. They can be categorised into :

Peptide Sequence Databases

NR All non-redundant GenBank CDS translations+PDB+SwissProt+PIR+PRF

month All new or revised GenBank CDS translation+PDB+SwissProt+PIR+PRF released in the last 30 days.

swissprot the last major release of the SWISS-PROT protein sequence database (no updates)

yeast Yeast (Saccharomyces cerevisiae) protein sequences.

E. coli E. coli genomic CDS translations

pdb Sequences derived from the 3-dimensional structure Brookhaven Protein Data Bank

kabat Kabat's database of sequences of immunological interest [kabatpro]

Nucleotide Sequence Databases

NR All Non-redundant GenBank+EMBL+DDBJ+PDB sequences (but no EST, STS, GSS, or phase 0, 1 or 2 HTGS sequences)

month All new or revised GenBank+EMBL+DDBJ+PDB sequences released in the last 30 days.

dbest Non-redundant Database of GenBank+EMBL+DDBJ EST Divisions

dbsts Non-redundant Database of GenBank+EMBL+DDBJ STS Divisions

htgs htgs unfinished High Throughput Genomic Sequences: phases 0, 1 and 2 (finished, phase 3 HTG sequences are in NR)

yeast Yeast (Saccharomyces cerevisiae) genomic nucleotide sequences

E. coli E. coli genomic nucleotide sequences

pdb Sequences derived from the 3-dimensional structure

kabat Kabat's database of sequences of immunological interest [kabatnuc]

vector Vector subset of GenBank(R), NCBI, in ftp://ncbi.nlm.nih.gov/blast/db/

mito Database of mitochondrial sequences

epd Eukaryotic Promotor Database

gss Genome Survey Sequence, includes single-pass genomic data, exon-trapped sequences, and Alu PCR sequences.

2.5 The Web Site

Definitions:

Before we go into details explaining the system, we will present some definitions of what is entailed in setting up a web site.

WebSite:

A web site is composed of three components:

1. Configuration files: Tell the web server how to respond to different events.
2. Documents: Documents, images, data to be served to the general public.
3. logs: Log files that record transactions that take place.

WebServer:

A web server translates a URL either to a file name, or a program name. In the first case, the file is sent to the requester. The later, will run the program and report its output. Known tasks of a web server:

1. Authenticate users: A Web server can allocate or release resources to users based on their passwords and login names. Moreover, during secure communication authentication means that the web server makes sure that the data is coming from a specific user and no other user can mock his identity.
2. Respond to error messages: The web server returns configurable responses when an error in request or if it received a request for a non existing resource.
3. Negotiate a style and language with the inquirer:

   This means that the web server communicates with client programs using the same protocols.
4. Run as a proxy server:

   A proxy server accepts requests from clients, forwards them to the real servers, and then sends the real servers' responses back to the clients. The proxy might be running on the far side of a firewall giving its users access to the Internet. The proxy might cache popular pages to save re accessing them.

5. Be secure:

   Encrypt the data using SSL (Secure Socket Layer) before sending data to the user, and decrypt the data received from the user side.

2.5.1 Apache

One of the most deployed servers on the net, is a robust, commercial-grade, feature full and freely-available source code implementation of an HTTP(Web) server. The project is jointly managed by a group of volunteers located around the world, using the internet and the web to communicate, plan and develop the server and its related documentation.

2.5.2 How to Install?

Get Sources

The compressed version can be down-loaded from http://www.apache.org or any of its mirror site. There is a readme file on how to get it installed. Here are major guidelines on how to install it.

Configure and Install

1. Pick up your target directory, usually it is /usr/localpkgs/apache-1.3.6.
2. Download the source files, make sure you have the sources for the system to work.
3. Uncompress the files using:
   (a) gzip -d apache.tar.gz
   (b) tar -xvf apache.tar
4. Go to the unzipped directory and configure the installation package using the script configure.
5. Run ./configure -prefix=/usr/local/pkgs/Apache –add-module=mod_auth.c.
6. Then type: make; make all; make install &

Configure Apache

Apache uses a set of files that configure several things we need like:

1. Default home page: This is what apache shows when someone accesses your machine name or IP address using a browser like netscape or internet explorer.

2. Authentication: This a set of directives that configure what users are allowed which resources. In addition to these files there is a utility that allows you to create and manage the database if users and groups that are allowed access to the system

The file is called *httpd.conf* and it is present in the directory *conf* where apache is.

2.6 Security, Encryption and Authentication

2.6.1 Web Server

The need for secure transactions calls for having a secure web server. For this reason, we have installed an https (Secure Hyper Text Transfer Protocol) web server. A server certificate will be purchased and 128 bit encryption will be supported.

2.6.2 Configuring Apache Authentication

When it comes to authentication, this will be a political decision. Our system supports a database of fixed users which can be administered using the Apache Authentication scheme. Here is some of the things one needs to do in order to enable the authentication module of Apache:

Figure 2, shows the "Directory" directive that is used to configure Apache to read the database files and figure out user and group permissions.

2.6.3 Configuring Authentication Database

It is the system security administrators responsibility to create and manage the authentication files. There are two authentication files, one for public users, the other is for adminstrators. In the public users file, the adminstrator can grant or deny specific user resources. Moreover, in the group authorisation file properties are attributed for

Figure 2: File http.conf

```
<Directory "/mnt/discrete2/bioinfo/wael/www">
AuthType Basic
$AuthName "Bio Info"
AuthDBMUserFile "/mnt/discrete2/bioinfo/wael/www/Users/users"
AuthDBMGroupFile "/mnt/discrete2/bioinfo/wael/www/Users/users"
require group admin
satisfy all
</Directory>
```

a group at a time. There should be at least two levels of users, admin and public. Figure 2, shows how to direct apache to the user and group databases. These databases, keep track of users and their passwords. These passwords are generated using dbmmanage which is a utility that can be found in /usr/local/apache_1.2.6/support.

2.7 Problems Faced

This is a list of the problems that we faced while deploying the system, the first is with dbmmanage, the authentication database administration utility, and the second is with the digest method related to group authorisation.

2.7.1 dbmmanage

The dbmmanage perl script that comes with the latest version of apache Server version: Apache/1.3.6 (Unix) has something wrong with it. We have submitted a bug report to Apache administrators detailing the failure. The work around this is to download the old version of apache. Compile it and use the old dbmmanage utility with the 1.2.6 apache version. It works.

2.7.2 Digest Method

One of the authentication methods is the Digest method (htdigest). Digest is an Apache directive. You should avoid using it because it does not offer group authentication. It only offers single user authentication. Our system needs to be able to support hundreds of users. If we choose to use this method, we will be introducing a limitation to the system. For this reason we used the basic authentication type.

Chapter 3

System Specification

3.1 Objectives and Scope

The system serves the CSFG lab at Concordia University, it is meant to accelerate the process of gene matching and finding alignments through a tool called BLAST. One the other hand, it has a web interface that is accessible internally and externally to the biology lab. A part of the system will be performing system redundancy in case of any failures. The system performs transport of files from sequencer to the database host. On that host two major functionalities are performed. A periodic querying of sequences, and web transaction processing. Moreover, it includes backup system that will be used for archiving. The web server will be our gateway to the outside world, it accepts requests from clients from within the CSFG lab or from the outside . In the former case no strict authentication or security measures are needed. For the later, however, a secure web server will be installed. Accounting modules should be integrated to the system to allocate user space, information and history.

3.2 Major Software Functions

The software performs the following tasks: Data Transport, Periodic Searches, Web Search, and supports Fault Tolerance.

3.2.1 Data Transport

The sequences produced by CEQ2000, are stored in a MS Access database. Given that the core engine BLAST runs on a Unix clone, namely DigiUnix, data needs to be transferred to the appropriate place, namely Discrete. This system or script may or may not have a visual interface, the goal is to transfer the files over to a specified host. The sequence of operations will be as follows:

1. The biologist performs her/his test trial on sequence.
2. If results are not satisfactory, tests have to be redone.
3. If results were good, the export option which comes with the software running on sequence will export the data to fasta format. FASTA is a desired format because it we can query the NR database using BLAST with FASTA formatted directly.
4. The subsystem will login to the specified site and upload the local pool. This ideally should happen as frequently as desired, there might be a preference to export data one plate at a time.

3.2.2 Periodic Searches

The FASTA formatted files, once on Discrete, should reside in the specified directory were the daemon expects these files to be. The daemon will run periodically on a preset cycle, one should make sure the length of cycles generally will not result in an overlap of daemons running multiple instances. The system, and after figuring out the results per search, will do the following:

1. Move the processed files to a temporary directory. The purpose of this temporary directory is to allow the biologists to rerun several or some of the searches that it had done earlier. Not deleting the files will give a buffer period where one can go back and resubmit the searches. The daemon can be stopped meanwhile so as interaction won't happen.
2. The system will parse the results provided by BLAST and create a summary that will be appended to an HTML file which is readily viewable by users inside

the lab, not outside. The system administrator can use the apache directives to limit external users access.

3. The details of each search, produced initially by BLAST, will go into a specified directory, these can be configured to be in HTML or in text format.

4. In step 3, we created a summary table, each row of that table has a link that will redirect the uses to the actual blast results that we talked about.

5. It is advisable that the system administrators properly track disk space consumed. It should not be a lot, but given that there are multiple mail boxes where the information is stored we might end up with a lot of data that will consume disk space.

6. At any time, the system administrators should be able to kill the server by issuing the command stop in the directory were the server was launched.

3.2.3 Web Searches

This part of the system has the following main functionalities:

Service Web Requests

The system will accept the request by a web user, the request being a text file in FASTA format, and the parameters that he can configure using the web interface that it provides.

1. The data files accepted will be stored in the system directory (/tmp). The file name given will be a date time stamp, we have chosen this method to eliminate deadlocks which would happen by file name reuse.

2. The transaction will be logged in a todo list file. It is a text file that looks exactly like if one was to issue the command from the prompt. Again, the advantage of this scheme is that the web server will not be running the script itself.

3. Now, the fast daemon will have the pleasure of servicing these requests log. The fast daemon should run every 20-30 seconds because we want users not to wait

for ages. The advantages of doing this is that, you will be avoiding webserver timeout. All web servers will timeout if an operation that has to be performed consumes more than 60-90 seconds. Changing the system time out is not a feasible solution because what would happen in case the web server had a real problem were a process gets stuck. Moreover, a ceiling on how long a search can take does not exist.

Manage User Accounts

This task will be done jointly by the system administrators and the web server. The scenario works as follows:

1. The user submits a request through a web page that he needs to open an account on the machine.
2. The system administrator should verify the coordinates and information submitted by the user.
3. Use the *dbmanage* utility to create a user name and password.
4. Configure the appropriate locations that the user is allowed to upload or download information.
5. Specify the user group and level in the apache directives.

Authenticate and Provide Access Passwords to Users

1. With appropriate configuration, Apache will provide the facilities and enforce the limitations on the user.
2. As a future feature, scripts running on the site will create user accounts. This frees the system administrator for system management.

3.2.4 Fault Tolerance

Once the system becomes operational, we have to account for system failures and periodic system backups.

The system will should be able to replicate itself on a remote machine, this can help for two things:

1. The system should provide the ability for machine backup on local or on any remote host.

2. The backup subsystem should be able to run periodically as well, logging on a remote or localhost, upload files and close connection.

This subsystem needs to be configured in a way to transport specific disk trees, to exclude a set files and/or directories and to a subset of file. It should be able to accept one file transfer.

3.3 Major External Databases

Currently the system is connected to the nonredundant database from NCBI. This is the database that search submissions are performed against. This database is in text format and contains about 5Gb of disk. Another database that the transport pipe reads data from is an MS Access data base. The sequencer software is able to export that data in FASTA format. The other databases we are proposing to be integrated into the system hold the query results databases. As two of those will be needed, one for the pipeline system, the other for web request results.

Chapter 4

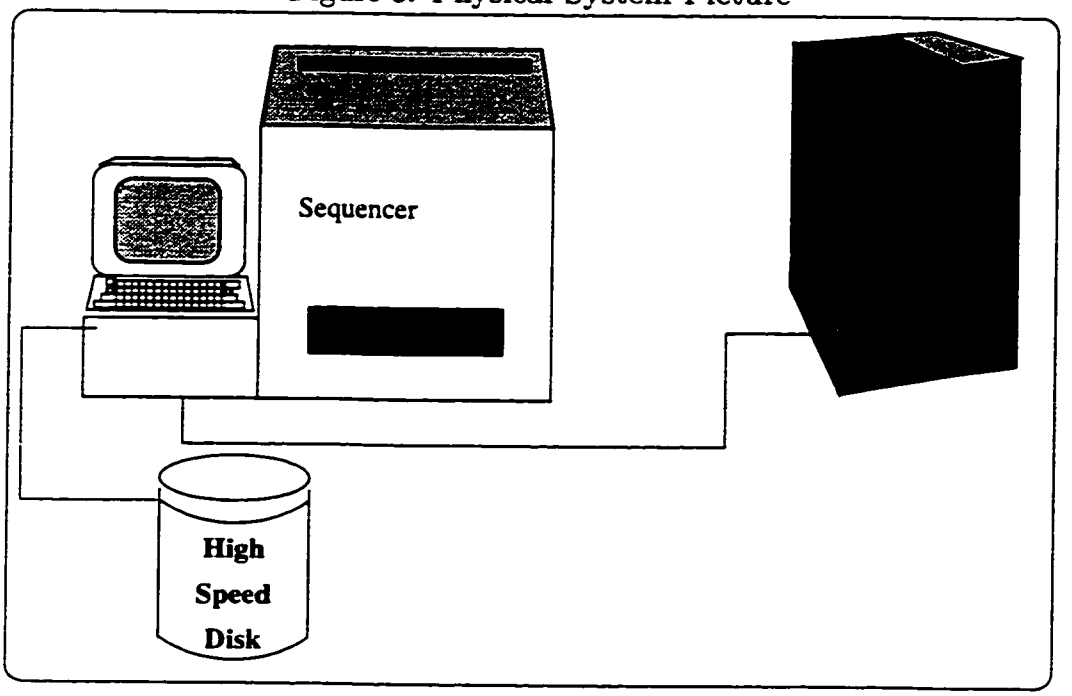
System Architecture

We use the core search engine from NCBI, National Centre for Biotechnology Information . They have developed BLAST (Basic Local Alignment Search Tool) which is a set of similarity search programs designed to explore all of the available sequence databases regardless of whether the query is protein or DNA. The BLAST programs have been designed for speed, with a minimal sacrifice of sensitivity to distant sequence relationships. The scores assigned in a BLAST search have a well-defined statistical interpretation, making real matches easier to distinguish from random background hits. BLAST uses a heuristic algorithm which seeks local as opposed to global alignments and is therefore able to detect relationships among sequences which share only isolated regions of similarity[5]. For a better understanding of BLAST you can refer to the BLAST course which explains the basics of the BLAST algorithm.

4.1 Data-Flow Diagrams

This section goes over the data flow of system, there is a lot of information being transferred as well as a lot of data processing. The first data transfer of the exported database is from the sequence generator to the machine that holds all the database and queries the information (Discrete). Processing happens in Discrete. Output results are stored on Discrete as well. The web server runs on Discrete and does all its processing and reporting there. The backup process runs on discrete and exports files to Gene, the backup machine. Figures 4 to 8 explain the data storage and transfer, called Data Flow Diagrams. In these diagrams we represent a process or

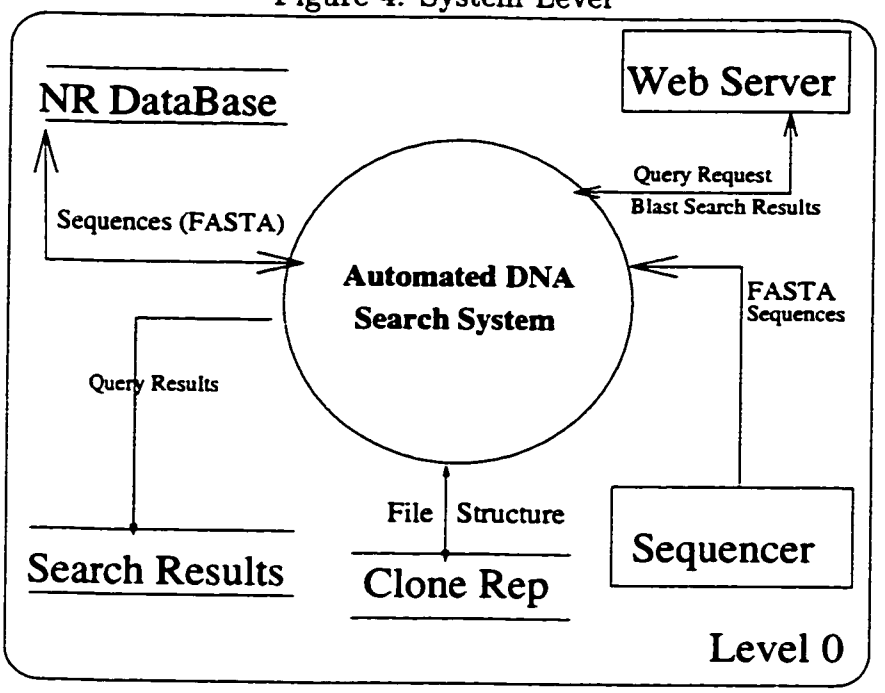
Figure 3: Physical System Picture



computation by circles. Open ended rectangles represent entities or remote systems. Each arrow shows a data path and direction of data. The type of data is explained next to the arrow.

Level-0: In Figure 4, shows level 0 which is the highest abstraction of the system. All of the functionality is inside the circle labelled "Automated DNA Search System." The external objects to our system are the web server and the sequencer. the Automated DNA Search System, accepts data files coming from the Sequencer. It stores query results in a database. There are two types of requests that the system accepts: requests coming from the web-site, and requests filled by the pipeline subsystem. The NR database is needed for query processing. The databases need are the *non redundant* NR database, the *Search Results* database, and Clone repository the *backup file and directory structure database.*

Figure 4: System Level



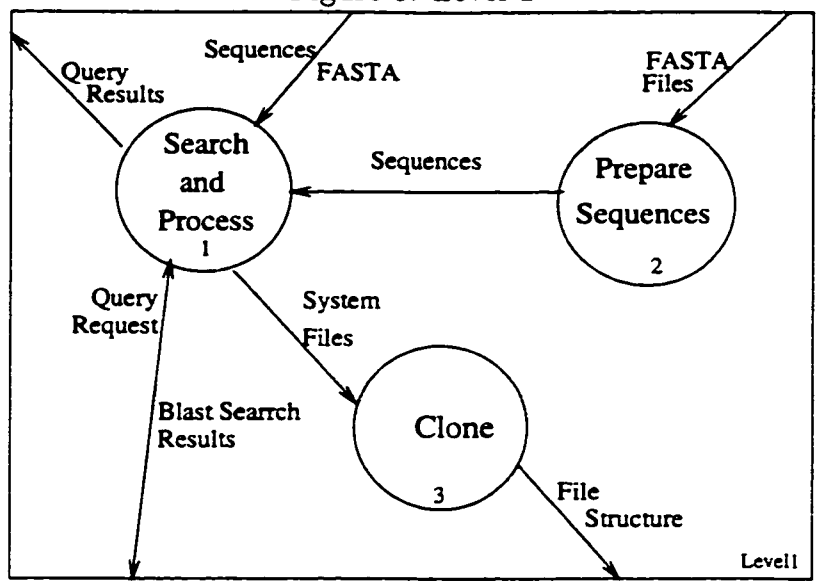
Level-I:

Level I describes goes into more detail. It subdivides the system process into more processes with more detail. You can see the second level over abstraction in Fig. 5. The *Prepare Sequences* process accepts input from the gene sequencing machine, called Sequence. System clients connect to the system from within *Search and Process.* They can connect either from the Web site or from the machine Sequence. The clone engine called Clone maintains a database of the files and directory structure that will copied periodically to Gene.

Search and Process (Level-II) :

In Fig. 6, there are two subprocesses in this subsystem "Process Pipeline" and "Handle Web Request." From the names of the processes we can tell what each does. At this level in this subsystem, there are three databases "Requests Repository", "Pipeline Results", and "Web Results." "Process Pipeline" performs matching of FASTA format files brought from Sequence to the NR database. It stores the search results in the "Pipeline Results" database. The we request handler subprocess called "Handle Web

Figure 5: Level 1



Requests" reads in query parameters and options and stores output in the "Web Results" repository.

Prepare-Sequences(Level-II): Figure 7 is mostly an abstraction of what happens in between sequence and Discrete. "Generate Sequences" takes place on Sequence. "Export Data" is the module that is included in Ms-Access software that runs on Sequence. Send files is module that will send the files using ftp to Discrete.

Level-III: Fig 8 shows "Clone" our backup process to the machine Clone, Clone will create a duplicate set of the disk tree and directories on Discrete which can be transferred to Gene if any crash happens on Discrete.

Figure 6: Search & Process

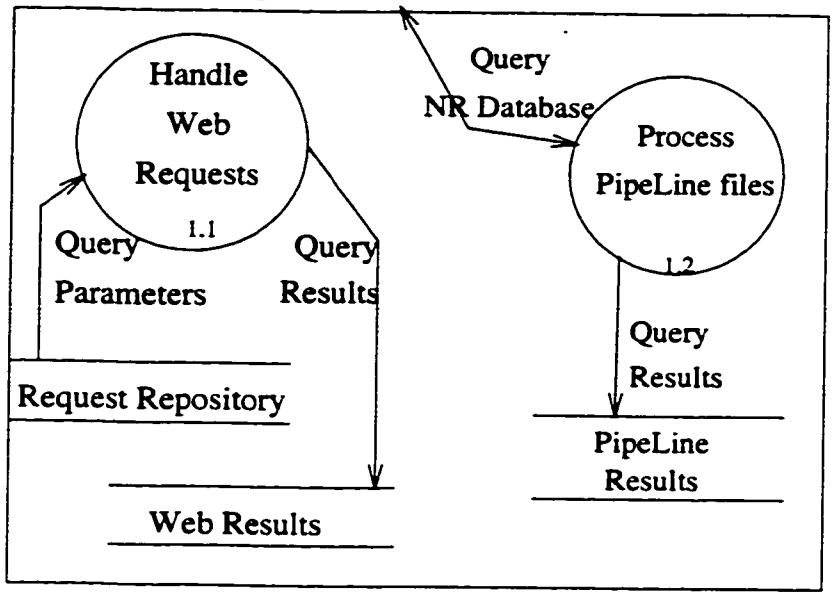


Figure 7: Prepare Sequences

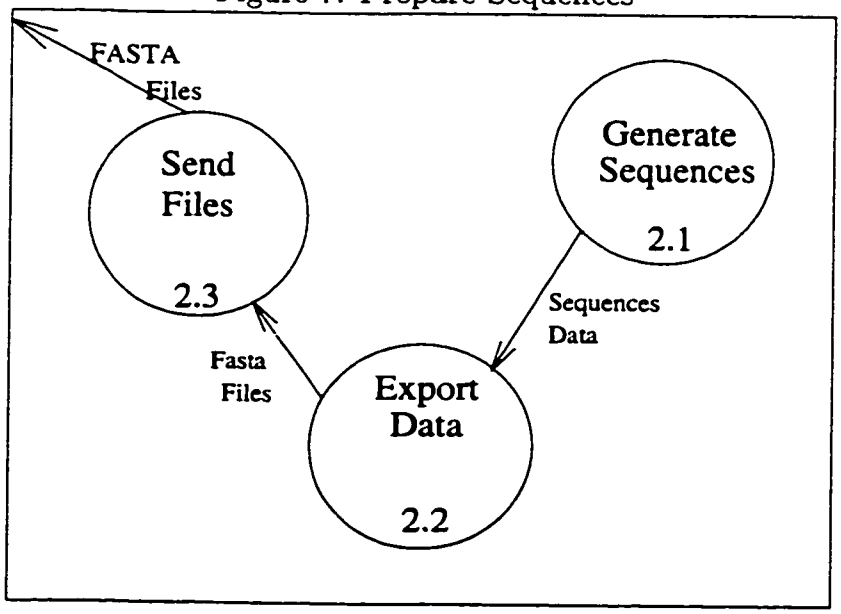
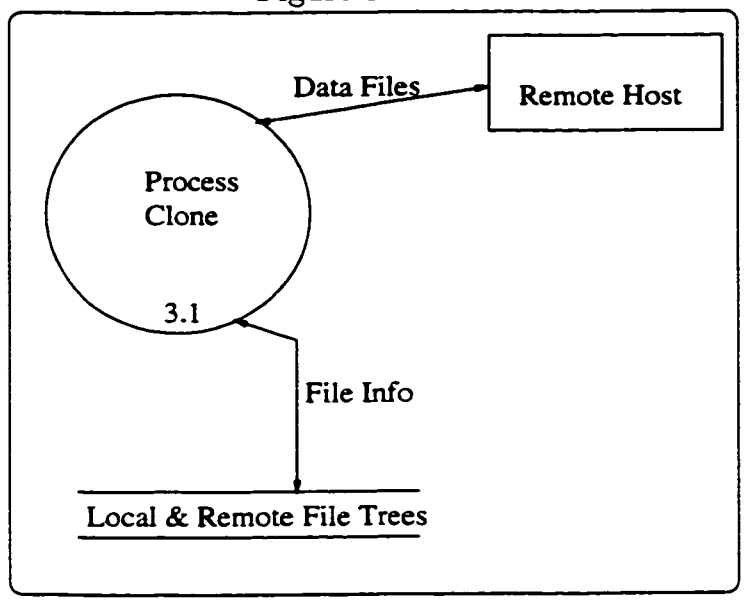


Figure 8: Clone



4.2 Process Communication

Here is a major description of the process communication model. The modules are distributed over the Beckman *CEQ2000* and on Discrete. We used Perl a multi-platform scripting language that is recognised for its:

- Portability.
- Regular Expression power.
- Availability of packaged services.
- Speed of programming.

We will present the major modules with explanations on their requirements, preconditions, and post conditions. Moreover, we will highlight important parsing methods.

4.3 Implemented Modules

The major three subsystem modules are the pipeline, daemon, and Clone. In the following, we will be going over the features and the specifics of both of the subsystems.

Figure 9: Pipeline Configuration

```
11  $debug =1;
12  $BlastPath="/public/httpd/data/data1/Blast/";
13  $NR="/public/httpd/data/data1/DB/nr";
14  $Temp="Users/Temp/";
15  $Output="/public/httpd/data/"."$Temp";
16  $Link="http://gene.concordia.ca/Users/Temp/";
17  $Storage="/public/httpd/data/Users/Done/";
18  $WaitTime= "500000";
19  $Expect= "1000";
20  $WordWidth="3";
```

Figure 10: GetFileList

```
43 opendir (DIR_HANDLE, ".") ||
##die "Could not open dir proc \n\n";
44 @DirList = readdir(DIR_HANDLE);
45 closedir(DIR_HANDLE);
48 for(@DirList) {
49 # print "matching ($_)\n";
50 $_ =~ /(\w*\.fasta)$/;
##;; ///This means Match all files that consist of
##characters followed by a dot then the text "fasta"
```

4.3.1 Pipeline

Figure 6, shows the pipeline module that runs as a daemon on Discrete it will take care of searching the transferred files over from CEQ2000 and querying the BLAST database. It kicks in every 500000 seconds, which approximately equals 13.8 hours. This number can be easily configured using the variable $WaitTime.

GetFileList

This procedure gets a list of files from the current directory where the daemon is residing. It retrieves a list of files ending with "FASTA." These files would have been transferred from the CEQ2000 via ftp to Discrete. See Figure 10 for details.

GetFileName

This procedure generates a unique file name that will be stored in a specified directory. The generated file name is a time stamp. Given that this procedure is sequential, no

Figure 11: GetFileName

```
70 # This is the directory name where Temp files are going to be
71 $dirName = "/public/httpd/data/Users/BioinfoLab";
72 $fileName = "$dirName"."/".$$.time.".txt";
73 printf " Current Temporary file name is (%s) \n\n", "$fileName";
74 return($fileName);
```

collisions should occur. See Figure 11.

QueryUsingBlast

This is the actual procedure that will invoke BLAST with specific parameters to append the output to an HTML table. The results file will be stored as well. (Fig 12)

4.3.2 Fast Daemon

This is another daemon that services requests for searches that come from the web server. That is clients connect to the system via the secure web site they upload their gene sequences, and submit the search. The web server will capture all parameters and input file, stores all details in a file acting like a Mailbox. Then comes the fast daemon which kicks every 10 seconds, the reason we have used the Mailbox method comes from the nature of this electronic exchange. Multiple clients want to connect at the same time performing different transactions. Nevertheless, one client can connect from the same browser more than once to speed up this searches. Given that search operations will take more than 30 seconds to perform the query and that this time may increase without any limit, we discovered that the web request operation will time out on the client side. Moreover, the more concurrent transactions we have, the more we need disk accesses and CPU power. We are limited by resources, the Mailbox will solve the problem because:

- Operations will run sequentially.
- Server will not time out. However, it will redirect the users to the page where the output will be stored. That page can be refreshed to get results at the clients or the servers will.

Figure 12: QueryUsingBlast

```
80 open(TABLE,">>/public/httpd/data/Users/BioinfoLab/table.html")
|| die("Could not o pen table file");
82 print TABLE "<table border=1>\n";
83 foreach $file(keys %FileList) {
84 print "\nMatching of file ($file)\n" if ($debug);
86 open (TMPFILE,"$file");
87 $FilN=<TMPFILE>; ## read the whole file into the string $FilN
88 $FilN =~ /^>(\w+)/; #Capture the protien name which comes after ">" in inp stream
89 close (TMPFILE);
90 print "Outputing Sequence ($1)";
92 $FileName = "$Output"."$1.html"; ## name the output file name
93 $TempFile = "$Link"."$1.html"; ## Generate the name of the link in the Table.
94 $Link="http://gene.concordia.ca/Users/Temp/";
98 @pars = ( "$BlastPath"."blastall", "-pblastx" , "-i$file","-e$Expect",## store the program
99 $Word Width", "-o$FileName","-T", "-d$NR"); # name etc in string pars
101 $rc = system (@pars); ##Invoke the system call to preform the query operation.
110 print "Moving File to Temp Storage\n";
111 $mv = `mv $file $Storage`;
115 $SearchResult =~ /.*(\*{5}\sNo\shits\sfound\s\*{5}).*/;
##Find out if the output file has the string
##***** No hits found ***** in it.
##If it fails it will log that in the Table
##If it succeeds it will preform the following
121 $SearchResult =~ /Query=<\/b>\s(\w+)/;
143 $SearchResult =~ /<b>Query=<\/b>\s(.*)\n/;
148 $SearchResult =~ /\>\<\/a\>\s(.*)\<a\s.*>\s(\d+)\<\/a\>\s+(\d+)/;
149 print ".....Big Match\n($1)($2)($3)($4)($5)\n"; ##Get Search result details
```

Figure 13: FastDeamon

```
17 fork&&exit; ## Run in background
18 $| =1; ##Refresh output
21 open(PID,">./fastdaemon.pid");
##store the fast daemon ID in the text file fastdaemon.pid
22 print PID $$;
23 close PID;
26 while (1) { ##Forever do
27 @Commands = undef;
28 open(LOG,"/tmp/ProcessLog.dat"); ##Open Log file
29 @Commands= <LOG>; ##Read List of stored commands to a hash from the MailBox
30 close(LOG);
32 for (@Commands) {
34 print "Running $_\n";
35 `$_`; ##excute the default variable which
##h is the ith variable in the loop on line 32
37                                                                    }
38 sleep(10);
39 }
```

Figure 14: Stop Fast Deamon

```
kill -9 `cat fastdaemon.pid`
```

- We will not fall into thrashing that can be caused by two processes trying to access different parts of the disk at the same time.

Figure 13 previews the code of the fast daemon and the log file produced by the web server.

Figure 14, shows the command that will kill the fast daemon. It will read the process ID from the file where it was stored.

4.4 Sample Results

These are sample results of the searches:

Here is the result of the search against the NR data base which will be stored in /Users/WebBlast/

Table 4.1 shows the an example of the table showing the results.

Figure 15: Sample Input

```
>GoodSample|532319|pir|TVFV2E|TVFV2E envelope protein
ATCGTGATGATGCTAGCTAGCTAGCTGCATGCTAGTCGACTGACTGATGCTAGC
TCGATGAACTGCTAGCTGCATGCATGCTAGTCGATGCTAGCTGTAGTGATGCTA
GTCGATGCTGAGCTTAGCTGATGCTGCTGCTGTGTGCTAGTCAGTACGTCGATG
CTAGTCGCTAGCTGACTAGC
```

Table 1: Results Table

GoodSample|532319|pir|TVFV2E|TVFV2E	gi|84221|pir||A25942	2.7	2.5

4.5 Machine Clone

This backup system was created so that any machine holding any database can be cloned to a redundant machine. The program is written in perl it is made in such a way that it will synchronise the specified directory structure. That is it will update one machine with the latest version of the files and directories. It will erase non existing directories as well. It will need a network connection between the two machine as well as FTP access.

4.5.1 Load Configuration

This module loads parameters of local and remote machine in case this cloning will be scheduled to run as a daemon.

Figure 16: `LoadCloneConfiguration`

```
sub LoadConfig {
#print "Inside Load Config\n";
open (CONF_FILE, "Config") || die("Could not open configuration File");
{ local $/ = undef;
#undefining \n to be able to load the whole file.
$ConfFileData = <CONF_FILE>;
close(CONF_FILE);
eval($ConfFileData);
}
for(@ExcludeList) { $Restricted{$_} = 1; };
```

Figure 17: Mirror Configuration File

```
# # # Config file for Mirror Server # #
$FTPHost = "";
#Destination for Data
$RemoteDir= "/home/wael/Work/Mirror/A";
# dont add "/" at the end of string
@IncludeDir = ("/home/wael/Work/Mirror/A/");
# Directories to include
$CurrentDataBase = "./LogDB";
@IncludeFile = ("/home/wael/Work/mirror.tar");
# Specific Files to include
@ExcludeList = ("/home/wael/Work/Mirror/A/");
# List of files and directories to be excluded
```

Table 2: ProcessLog.dat

blastall	-d/public/httpd/data/data1/DB/nr
-d/public/httpd/data/data1/DB/nr	-i/tmp/11080947968758.inp
-pblastp	-e10
-d/public/httpd/data/data1/DB/nr	-o/data/11080947968758.ou t
-m0	

4.5.2 Configuration File

Figure 17 shows how the configuration file of mirror.pl looks like.

4.6 File Structure and Global Data

This is a list of the different files used along with a detailed description of the file formats and data types. It might be worthwhile to mention that these files would be replaced by data bases in any future system upgrade.

- ProcessLog.dat: This file is stored in the system temporary directory */tmp*. This file is created and updated by the web server. The "fastdaemon" process will frequently visit this file to read query commands. This is how what the ProcessLog.dat looks like : This is an initiation of the blastall script with parameters like the input file, output file and other blast result configuration controls. (Table 2)

- table.html: Table 3 shows the output of the webblast version, that is where the results of fastdaemon searches get stored.

Table 3: WebSearch Result

Sequence Name	Search Result	Score	Evalue	Details
An1612	gi|3044064 (AF052205) PHD finger protein[Homosapiens]	32	2	Click Here

4.7 Test Provisions

The testing methodology should be straight forward. Assuming BLAST's accuracy we have restricted the tests to concurrency and performance. Each module required a different testing strategy. For the *pipeline* module we flooded the pipe with a large number of files(300) that would take quite a while to process. Given that this is more than what the biologist would submit in a day the performance was satisfactory and within the time restrictions. One thing we want to avoid in this case is the overlap of the more than one instance of the process. This might overload the system and slows down the queries and might eventually kill the server. As for the *FastDeamon* the test is very similar. Although the process is inherently interactive, yet the reality is that web submissions are logged and queried sequentially with a fast heart beat by the query daemon. The number defaults to 50 Sec. Of course this number can be configured to best suit the needs of the system. The backup system or Clone, is one that requires a little more of boundary and special case testing. The complete subsystem functionality has been tested. We have tested the update, delete, and replace actions that the program does. Any file property change on the host should result in an update at the backup host. Boundry cases and error detection have been allocated for by testing with files that have dates prior to the latest backup.

Integration Strategy

Any modules that need to be added to the system and use the same repository should be implemented Perl, version 5.0 and above.

4.8 Transfer Considerations

The system is designed in a way that makes it fully portable. The only things that need be changed are the configuration parameters of each of the scripts. These parameters are a always located in the very first few lines of all the scripts. A minimal disk

space of nine Gb of disk is needed. Perl 5.0(Perl Web Site http://www.perl.org) or better, the latest version of blast from NCBI home page, (http://www.ncbi.nlm.nih.gov). The latest version of *Apache* from (Apache Web Site, http:/www.apache.org). An Apache related database , which is the system login database has to be preserved and transferred as well.

4.9 Operational Benchmarks

The system is meant service to different categories of clients, the internal lab clients and external web clients. The internal data processing engine runs as a daemon. That is, updates will be done periodically. The frequency can be configured by system administrators. The daemon model will help decrease the interaction between system users and system configuration. Moreover, the big batch processes do not need much configuration any way. One thing this system is required to have, is the average time and volume piped to the daemon. Ideally, in a specific segment of time, an exact number of queries can be processed. In the event that the number of queries increase or if the options get changed the time needed to fulfill the requests will be longer. One would fear that the daemon will start itself and the processes will overlap, that will slow the server down.

The web interface has different requirements, performance is an issue, yet leaving gateways to outside attacks is not something that we want. The web interface is a mocked interactive interface. That is, users will interactively enter their query along with their configuration parameters. The transaction will be logged into a "to do" list. Then the server will perform the queries using a daemon that runs periodically with very short delays. Overlapping of daemons here is not a big issue, because of two things. One, that there can be a limited amount of requests in a given lapse of time. Two, a threshold can be set on the server side.

4.10 Advantages

The system certainly has a several advantages of which we will include: the choice of language of implementation, authentication service, and the backup subsystem. These advantages show in the following contexts.

- The choice of the language of implementation helps a lot in terms of expand ability of the system. Perl is a modular high level functional language that has an Object Oriented flavour. Any new script that will easily integrate into the system with hardly any conflicts. This has been proven by the fact that each subsystem was designed independently of other subsystems.

- The system utilises the power and the availability of the authentication service provided by Apache. Apache is a solid, robust, and a reliable system that accounts for almost 60% of the deployed servers.

- Another part of the system which might not be involved in searching but certainly will be beneficial is the clone. Clone will be able to replicate Discrete in a timely and intelligent way. It will carry out updates on a frequent basis from one machine to another. In case of a crash of Discrete and or of its connection, the cloned system can take over in a nick of time. The switch over will happen smoothly provided that proper DNS entries are stored on the router that will give the cloned machine the right to respond to traffic that is served by gene.

4.11 Disadvantages

The system works perfectly fine, yet the language and the parsing techniques used are not intuitive. Some of the blame can be put upon the regular expression power of Perl. If the syntax of the BLAST output changes, the BLAST specific modules have to be rewritten. One way of overcoming this burden, is to start from the existing regular expression and modify it to accommodate for the BLAST output change.

Chapter 5

Conclusion and Further Work

5.1 Conclusion

The Automated DNA Search System (ADSS) is an automated DNA search system that interfaces query searches between the biologist and BLAST. These searches are performed, in our system, against the NR (Non-Redundant) database from NCBI. The NR database can be easily replaces by a text based database of our choice. This project comes as part of the fast growing era of *genomics*, which is certainly an advance in the science of biocomputing. This tool will help biologists provide better solutions for gene analysis, it will reduce analysis time, and will improve accuracy by a large factor.

5.2 Future Work

This project, by design, was meant for change. That is there are several issues created to database selection and connectivity, security enhancements, and report generation.

- When administered properly, this system works perfectly. We are suggesting a lot of improvements to it, namely databases connectivity. The system interoperates in between text databases. One very interesting improvement is to have the statistical data and results stored with an easy to use web interfaceable database. One good example of a database that can support that is *MySql*. The easy script and web tools and its availability promotes this database engine to be our candidate.

- Connectivity: The system is connected as far as file transfer from Sequence to Discrete is manual at this time. That is some body has to manually transfer the files to allow the daemons to have access to the files to be searched against.
- Security: Secure ftp can be installed to carry on files in between systems.
- Report Generation: Currently the system generates HTML formatted pages that are a direct translation of the text output of blast. One very interesting feature that researchers and data miners would like to have is some reporting or statistical mechanism that would allow more information to passed on to researchers.

Bibliography

[1] Boston University Bioinformatics Web Site, http://bioinformatics.bu.edu.

[2] Beckman Instrument home page, http://www.beckmancoulter.com.

[3] Report of the Invitational DOE WorkShop on Genome Informatics, 1993.

[4] Altschul, S.F., Gish, W., Miller, W., Myers, E.W., Lipman, D.J., "Basic Local Alignment Search Tool," National Center for Biotechnology Information, National Library of Medicine, National Institues of Health, Oct. 5, 1990.

[5] http://www.apache.org.

[6] http://www.perl.org.

[7] Wall, L., Christiansen, T., and Shwartz, R.L., Programming Perl, O'Reilly & Associates, Calif. 1996.

[8] Killelea, P., Web Performance Tunning, O'Reilly & Associates, Calif., 1998.

[9] Srinivsan, S., Advanced Perl Programming, O'Reilly & Associates, Calif., 1997.

[10] Garfinkel, S., and Spafford, G., Web Security & Commerce, O'Reilly & Associates, Calif, 1997.

[11] National Center for Biotechnology Information, http://www.ncbi.nlm.nih.gov.

[12] Apache Web Server, Computer Magazine, Vol. 1, 1999.

[13] Laurie, B., Laurie, P., Apache: The Definitive Guide, O'Reilly & Associates, 2nd ed., Calif, Feb., 1999.

[14] http://www.tcx.se/.

Appendix A

Bioinformatics Standards

A.1 FASTA Format Description

A sequence in FASTA[6] format begins with a single-line description, followed by lines of sequence data. The description line is distinguished from the sequence data by a greater-than (">") symbol in the first column. It is recommended that all lines of text be shorter than 80 characters in length. An example sequence in FASTA format is:

Sequences are expected to be represented in the standard IUB/IUPAC amino acid and nucleic acid codes, with these exceptions: lower-case letters are accepted and are mapped into upper-case; a single hyphen or dash can be used to represent a gap of indeterminate length; and in amino acid sequences, U and * are acceptable letters (see below). Before submitting a request, any numerical digits in the query sequence should either be removed or replaced by appropriate letter codes (e.g., N for unknown nucleic acid residue or X for unknown amino acid residue).

Figure 18: FASTA Format

```
>gi|532319|pir|TVFV2E|TVFV2E envelope protein
ELRLRYCAPAGFALLKCNDADYDGFKTNCSNVSVVHCTNLMNTTVTTGLL
LNGSYSENRTQIWQKHRTSNDSALILLNKHYNLTVTCKRPGNKTVLPVTIM
AGLVFHSQKYNLRLRQAWCHFPSNWKGAWKEVKEEIVNLPKERYRGTND
PKRIFFQRQWGDPETANLWFNCHGEFFYCKMDWFLNYLNNLTVDADHNE
CKNTSGTKSGNKRAPGPCVQRTYVACHIRSVIIWLETISKKTYAPPREGHLE
CTSTVTGMTVELNYIPKNRTNVTLSPQIESIWAAELDRYKLVEITPIGFAPTE
VRRYTGGHERQKRVPFVXXXXXXXXXXXXXXXXXXXXXXXVQSQHLLAG
ILQQQKNL LAAVEAQQQMLKLTIWGVK
```

Table 4: Amino Acid Codes

Code	Amino Acid	Code	Amino Acid
A	Adenosine	M	A C (Amino)
C	Cytidine	S	G C (Strong)
G	Thymidine	W	A T (Week)
T	Uridine	B	G T C
U	G A (Purine)	D	G A T
R	T C (Pyrimidine)	H	A C T
Y	G T (Keto)	V	G C A
-	Gap Of intermediate Length		

Table 5: Accepted Amino Acid Codes

Code	Amino Acid	Code	Amino Acid	Code	AminoAcid
A	Alanine	I	Isoleucine	S	Serine
B	Aspartate	K	Lysine	T	Threonine
C	Cytine	L	Leucine	U	Selenocysteine
D	Aspartate	M	Methionine	V	Valine
E	Glutamate	N	Asparagine	W	Tryptophan
F	Phenylalanine	P	Proline	Y	Tyrosine
G	Glycine	Q	Glutamine	Z	Glutamate
H	Histidine	R	Argin ine	X	any
*	Translation Stop	-	Gap Of Intermediate Length		

For those programs that use amino acid query sequences (BLASTP and TBLASTN) the accepted amino acid codes are):

Appendix B

System Details

B.1 Installation and Configuration

The sources can be unzipped from the tar gzipped file (system.tar.gz). By default the files will be extracted to the following directories:

```
$WebServerHome =/public/httpd/data #This is were all the configuration files are
$WebServerCgiHome = /public/httpd/data/cgi-bin #Common Gateway interface files
$ResultsTemporaryArchive= /public/httpd/data/Users/Done
$SystemTemp = /tmp # This can not be any thing else but temp
$Clone = ~wael/clone.tar.gz #The compressed files of machine clone, used in system back up.
```

B.2 File Tree

This is a brief description of each of the files used in the system and presented in Figure 5.2.

1. Main pipeline system, when run periodically to service files transferred over from the sequencer.
2. A Wrapper for BLAST.
3. Performs formatting of results output and does file clean up.

4. Main Clone subsystem call, is a deamon can be configured to run every number of hours or days.

5. Please use this file to configure the back up system, you can specify directories/files to include/exclude.

6. This is the ftp client that will receive and updates files.

7. The web deamon that accepts requests, stores them in a file and fires them with a fixed heart beat.

8. stop will kill the fast deamon.

B.3 Directory Tree

The directory tree figure shows where files ought to distributed for the system to work properly. Please use it to configure the system and to keep track of the files.

Figure 19: File Tree

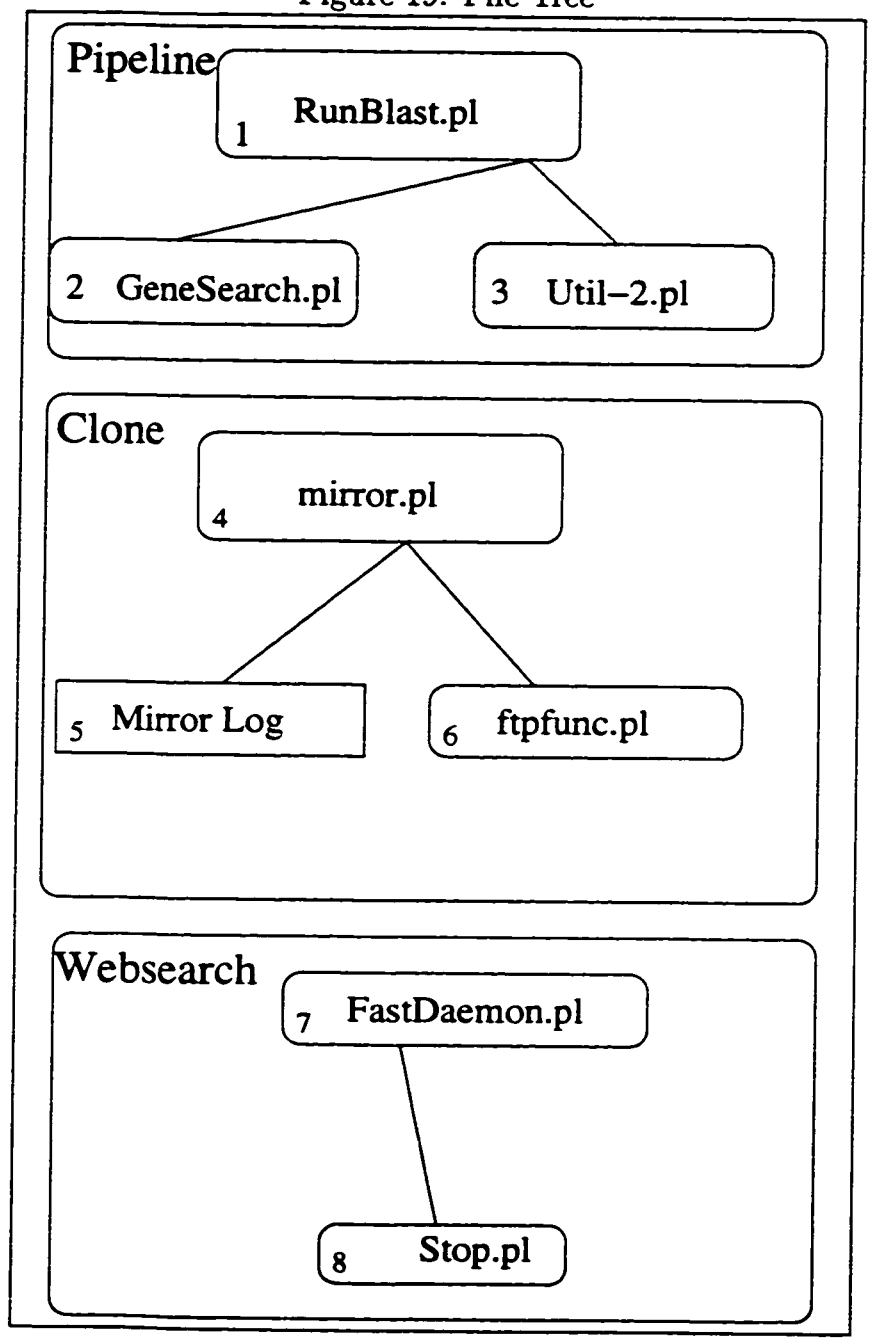
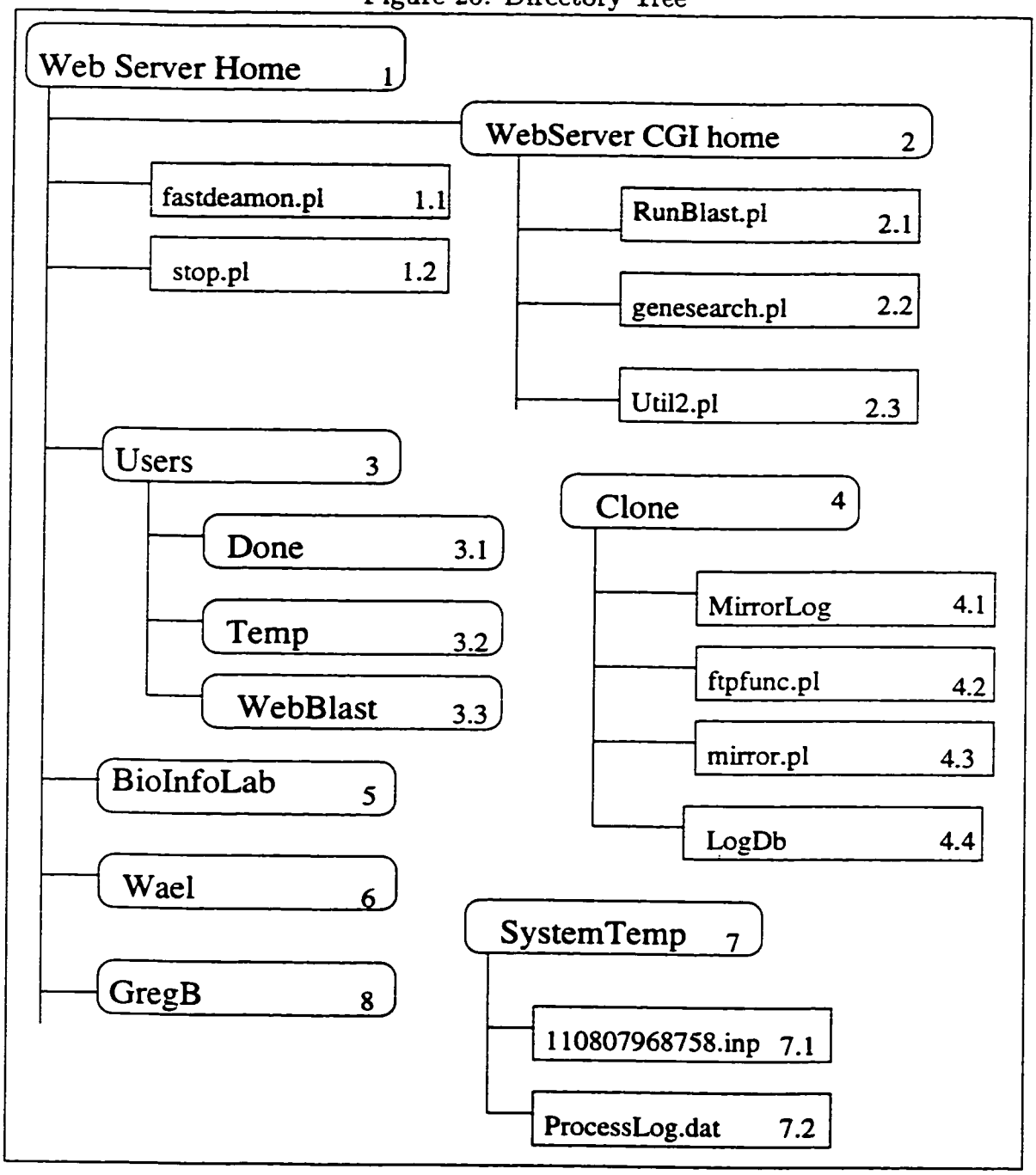


Figure 20: Directory Tree



Appendix C

Glossary of Terms

Apache A Web server system, can be downloaded for free from www.apache.org.

Annotation A functional description of a clone, which may include identifying attributes such as locus name, keywords, and Medline references.

BLAST The Basic Local Alignment Search Tool is a fast technique for detecting ungapped subsequences that match a given query sequence.

BLOCKS database A public database of protein patterns that correspond to the most highly conserved regions in proteins. It is used in Block 2 bioanalysis for comparison with Incyte and WashU-Merck clones that have no GenBank matches.

clone The backup system that we have implemented.

CGI Common Gateway interface, a mechanism that allows a Web server to run scripts and/or programs.

CEQ 2000 The gene sequencer machine made by Beckman instruments.

Clone ID The unique numerical identifier for each LifeSeq clone. A single Clone ID may have more than one associated Sequence ID.

Clones A group of cells derived from a single ancestor.

Cloning vector A DNA molecule originating from a virus, plasmid, cosmid, phage, bacteria, or yeast into which a foreign DNA fragment is integrated and then introduce into host cells, where it can be reproduced in large quantities (cloned).

Cluster A group of clones related to one another by sequence homology. Each cluster has a unique Cluster ID number for a given stringency.

dbmmanage An apache utility that is used to create and manage databases needed for user and group authentication.

discrete The machine that hosts the search engine.

DNA Deoxyribonucleic acid, the double-stranded molecule held together by weak bonds between base pairs of 4 different nucleotides. Encodes genetic information.

EBI European Bioinformatics Institute (EMBL Outstation)

EGCG Extensions to the GCG package (See article by Rice et al.)

EMBL European Molecular Biology Laboratory

EMBnet European Molecular Biology network

EPFL Ecole Politechnique Federale de Lausanne, Switzerland.

EST Expressed Sequence Tag; a sampling of sequence from a cDNA.

ETH Eidgenössische Technische Hochschule, Zürich, Switzerland.

Entrez An online resource provided by the National Center for Biotechnology Information (NCBI). It organizes GenBank sequences and links them to the literature sources in which they originally appeared. From LifeSeq, you can reach an Entrez Document Report by clicking on a GI number in the Hit ID column of a Results screen.

Exons The protein-coding sequences of genes. Exons only comprise about 10introns.

Fast Daemon The subsystem that takes care of web page submission by users.

FASTA A database search tool used to compare a nucleotide or peptide sequence to a sequence database. The program is based on the rapid sequence algorithm described by Lipman and Pearson.

FTP File transfer protocol, a communication protocol used to transfer file between two different systems.

Functional genomics Systematic analysis of gene activity in healthy and diseased tissues.

GCG Assembly A tool using the GCG Fragment Assembly System created by Genetics Computer Group, Inc. It is used to assemble nucleotide sequence fragments contained in a cluster and view how they overlap with each other.

GenBank The public DNA sequence database maintained by the National Center for Biotechnology Information (NCBI), part of the National Library of Medicine.

Gene A specific DNA sequence which carries the information required for constructing proteins. The human genome is estimated to contain 100,000 to 150,000 genes.

Genome The total genetic information possessed by an individual organism. Each cell contains a complete copy of the genome.

Genomics Sequencing and characterization of the genome and analysis of the relationship between gene activity and cell function.

Genotype The unique genetic makeup of an individual organism.

GI GenBank Identifier, a unique number assigned to protein and nucleotide sequences in the GenBank database.

mysql A multi-threaded multi user database.

Pipeline The subsystem that retrieves files from sequencer machine (sequence) to the search engine machine (discrete). It also runs BLAST to search for similarity matches of gene sequences in the non redundant database nr.

Perl A multiplatform scripting language.

Web Server A piece of software that runs on a machine providing services to clients connecting to that machine. The Web server delivers web pages presented in HTML format.