CHAPTER 2

DISTRIBUTED COMPUTING

Chapter 2. Distributed Computing

2.1. Introduction

The field of distributed computing deals with all aspects of information access and computing across multiple processing elements connected by any form of communication network, be, it local or wide-area in coverage[14]. One can define Distributed Computing as computing or performing the processing using spatially distributed systems. Thus, Distributed Computing makes use of Distributed system. Distributed system is "A collection of independent computers that appear to users of the system as a single computer."[15] Therefore, distributed computing deals with computing using multiple computers placed remotely from each other, where each computer taking part in the distributed computing, plays some role in a computational problem or information processing.

The major difference between networking and distributed computing is that, in networking, two or more computers interact with each other, but typically are not sharing the processing of a single program. Besides, in distributed computing the interaction between computers is transparent to the user. Distributed computing is natural result of the use of networks for allowing efficient communication between computers but this communication for data access or processing happens without the knowledge of the end user, transparently.

---

[14] Ajay D. Kshemkalyani et al., Distributed Computing: Principles, Algorithms and Systems, UK, CUP, 2008.

[15] Andrew S. Tanenbaum et al., Distributed Systems: Principles and Paradigms, NJ, Prentice-Hall, 2003.

A *distributed system* is one in which both data and transaction processing is divided between one or more computers connected by a network, each computer playing a specific role in the system[16]. Distributed computing includes the use of remote machines for storage, retrieval, and processing of data across the network. Consequently, a database application processing system, that software developers more commonly refer to as a client-server database application system is also a distributed application[17], which makes use of distributed computing concepts.

2.1.1. Characteristics of Distributed Computing

- Resource Sharing
- Transparency
- Concurrency Control
- Scalability
- No Global Clock

2.1.2. Features of Distributed System

The features to be dealt with in Distributed Computing include:

- Transparency - Distribution of data/computation without end user being aware of it
- Loose coupling – Alteration of resources of Distributed System is possible without affecting the remaining part of the system.

[16] http://docs.oracle.com/cd/A57673_01/DOC/server/doc/SD173/ch1.htm
[17] http://www.oracle.com

- Heterogeneous computing – Components could include variety of platforms
- Scalability i.e. increasing or reducing the resources dynamically
- Fault-tolerance or handling of failure
- Concurrency control – Simultaneous access of resources by multiple users

2.1.3. Motivation for Implementing Distributed System

The motivation for implementing distributed computing is:

- Possibility or need for inherent distributed computations, when designing applications
- Requirement of resource sharing
- Security of data
- Accessibility to remote data or remote resources
- Availability of resources
- Need of scalability
- Purpose to improve performance to cost ratio

2.1.4. Models for Implementing Distributed System

To implement the distributed system various models of distribution and architecture available are:

- Client Server model
- Processor Pool model
- N-tier architecture

- M-V-C architecture

2.1.5. Issues of Distributed System

The issues that need to be tackled during development and implementation of distributed system are:

- Transparency to the user about resource location and processing environment
- Resource sharing
- Applying concurrency controls and synchronization techniques on shared resources
- Applying appropriate controls for security and accessibility of resources
- Heterogeneous processing environments in terms of network protocols, hardware architectures, operating systems, database servers, file servers, application standards, middleware etc.
- Scalability of data and resources
- Availability of resource on user requests
- Coupling of components between clients and business logic
- Granularity of data and code hiding
- Protocols to be used in terms of proprietary, open or standard protocols
- Form of storing data and information in flat, un-structured, structured, semi-structured form.

- Data formats which are customized to applications
- Development methodologies in terms of sequential, structured, object oriented, component based or service oriented development
- Technologies and Standards used for Middleware like Multi-threading, Remote Procedure Calls (RPC), Java RMI, COM-DCOM, MPI interface, .Net Remoting etc.
- Openness of distributed system in terms of hardware implementations, operating system communications, application portability and compatibility
- Communication and scheduling platforms available like MPI, Java RMI, .Net Remoting, Matlab, Globus, Hadoop, Condor etc.
- The research work includes the above aspects of Distributed Computing in various ways, while developing the Web-based application using distributed approach for storing and retrieving DNA sequencing data.

2.2. Motivation of Applying Distributed Computing in BioInformatics

The ultra-high throughput data generated at an accelerated rate through next generation DNA sequencing[18] techniques need secured storage, speedy retrieval, quick and efficient processing, to gain genetic knowledge at appropriate and requisite time. Cost effectiveness and

[18] Ronaghi Mostafa, Pyrosequencing Sheds Light on DNA Sequencing Genome Res. 11 (2001) 3-11

technical viability of high-performance, high-throughput computer in an organization, where both the data storage and processing is happening on a single machine, are often the matters of concern. To such queries, distributed computing is the best alternative. Hence, a distributed web applications are developed, where the data storage and business logic required for querying the stored data, are, located on different machines. Primary purpose of improving efficiency of storage, processing and analysis of DNA sequences is, to get timely genetic information of an individual or organism, because, their major application is for identification and treatment of diseases like cancer and HIV, drug designing, controlling spread of epidemics by identifying micro-organisms involved in disease causing and prohibiting their growth, enhancing crop breeds, to meet the drought situation, improving animal breeds that provide food, nourishment, medicines or income; for phylogeny identification and in forensics.

Globally accessible, centralized repository of NCBI[19] (National Centre for Biotechnology Information) is available for storage (albeit, physically they are scattered like Swiss-Prot or DDBJ. Centralized storage is only a logical view that is transparently presented to the user) and processing of genomic data. But, at times, it is not possible to upload and disclose the confidential data of some genomes into the centralized repository of NCBI, particularly, when the project is sponsored and controlled under some specified guidelines and contract (Eg. Govt. of any country would not like to publish that some particular disease is prevalent in that country, to avoid affect to tourism). Although, one can upload data and keep it private in NCBI, but that is permissible, only till the manuscript

[19] http://ncbi.nlm.nih.gov/

describing the data gets published. If the sequencing data is uploaded at NCBI, eventually, it becomes an open data in public domain.

To avoid this disclosing, but at the same time use all the web based facilities for storage, retrieval and customized algorithms for processing of DNA sequencing data within an organization, a distributed application has been developed, which is discussed in this thesis.

Moreover, in a spatially located organization or research organization where the data generation experiments like DNA sequencing are conducted at scattered locations or laboratories with the need to exchange data between various locations, it is advisable to maintain the central repository (at least, the common interface, which provides the single point of access to distributed data). This central repository or at least the logical representation of centralized repository, maximizes of utilization of DNA data, at the same time relieves the biological researcher from obtaining the knowledge and responsibility of storage or security of data. The backup and recovery activity, botheration of security or availability of data, then becomes the responsibility of database administrator. Also, the biological researcher need not get involved in knowledge of computer science, in particular, database administration or network administration, and hence can concentrate in his actual research domain, which usually would be biology related.

2.3. Description of the Distributed Application

The first portion consists of a web application for storage and retrieval of DNA sequencing data. The web application uses the Client-Server model

as shown in Figure 3, with M-V-C architecture as a design pattern[20]. The application is initiated by making use of it to upload the FASTA files containing DNA sequencing data to the Application Server. These FASTA files can be uploaded by any registered user of the system. The registered user uploads the data file in *FASTA* format, which contains raw data generated after the DNA sequencing experiments in Sequencing Laboratories. Since, DNA sequencing processes are very complex, time consuming and expensive, it becomes essential to have secured storage of these data files. Hence, this distributed application provides the facility to store these original raw data files on central File Server, which can be referred for future use, if required. The privileged user can then upload the data file to Database Server, which provides the abstraction of central repository of the data with secured storage. The Database Server contains the DNA sequenced data in a structured format, which can be used to fire any types of queries in relational context and with different criteria as per user's need.

As a part of the research work, a distributed application has been designed, for storing, maintaining, and retrieving the DNA sequencing data into a centralized File Server and Database Server. Web-Server facilitates the access to this data stored on File and Database Server. Thus, the distributed application is location independent as well as transparency. The relational data dictionary designed for storing the DNA sequencing data particularly the pyrosequencing data helps in

[20] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, Design Patterns:Elements of Reusable Object-Oriented Software, Addison Wesley, USA, 1994

storing data in a structured form. The advantage of designing the Distributed Application for DNA sequencing data is that, it facilitates resource sharing across local area network/intranet, it provides centralized repository of DNA sequencing data, which is easier to manage and would help in efficient backup and recovery. Distributed approach also allows utilization of low cost resources available across network, instead of implementing the entire set of features i.e. Web-Server, File-Server, and Database-Server on a single computer. This Distributed Application becomes very essential to store the DNA sequencing data in an efficient and secured manner, because the process of DNA sequencing is very expensive and hence its data loss or corruption is not affordable.

In this research work, *Distributed Application* has been designed uses, the *Client-Server model* with *M-V-C architecture* as a *design pattern*. The *Web-Server* manages the entire application. The Web-Server provides the client interface, the control mechanism, as well as accessibility to the remote resources. Thus, the distributed application is *location independent* and uses *Thin Client* concept. Thus, no part of application requires installation or processing on the client's machine. The application helps in uploading the FASTA files containing DNA sequencing data to the File-Server or Database Server, which works like a *central repository* of the data with secured storage. The centralized File Server facilitates storing of data in unstructured form. The Database Server provides facility to store the data in the structured form. Since, the implementation of Database is using a Relational Database Management System; the data is stored in structured form with possibility to query using various Relational aspects. The system

executes the SQL queries on the basis on parameters sent through requests across the Web. In all, *multithreading* is used for *concurrent execution* and *synchronization*, Java Servlets, Hibernate and Struts is used for Web-based/network based invocation, *database independent storage* and *operating system independent* computing is applied. Besides Java RMI and multithreading is used for distributing the same algorithm on multiple machines for execution using different datasets.

2.4. Implementation details of the Distributed Application

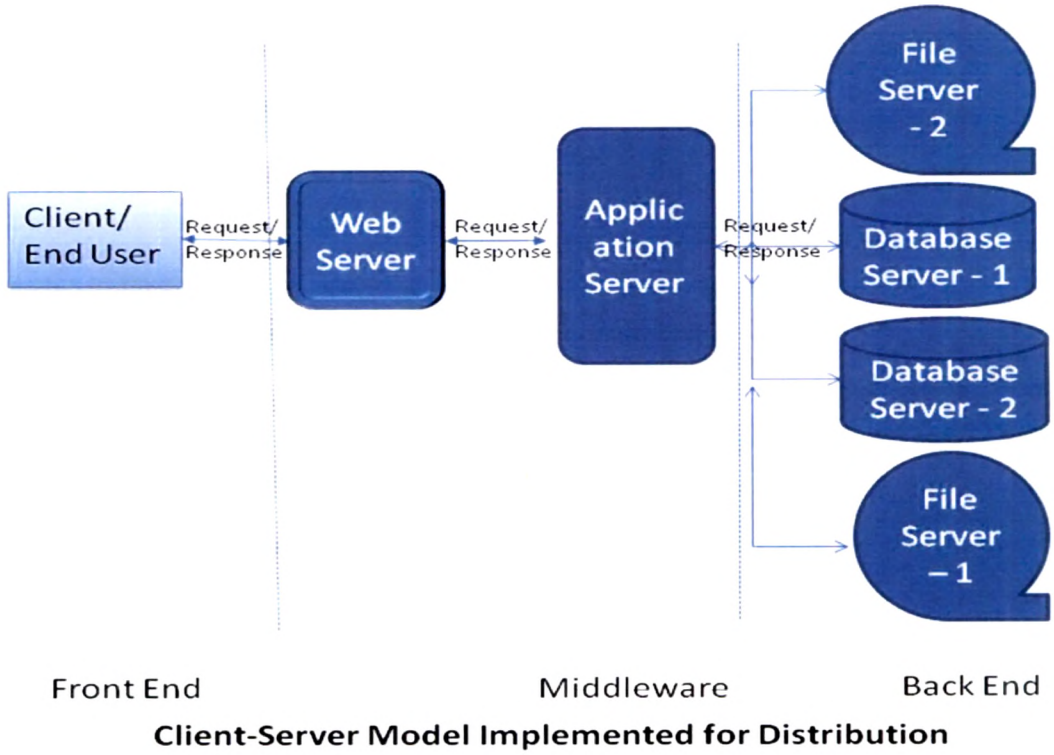

Figure 3. Client Server Model Implemented for Distributed Application

As shown in Figure 3, there are multiple database servers, which store the data in duplicate, which guarantees data availability. The centralized File-Server is also configured as a part of the distributed system, that

allows storage of original files as generated by DNA sequencing equipment. The application allows retrieving the data back from the File Server in unstructured form. The database server provides facility to store the data in the structured form with relational integrity. The relational integrity enables the user to acquire the information in terms of joins and sub-queries, thus, fetching more than just a raw data as was available in a FASTA file. The system executes the SQL queries based on a request received across the Web. The POJO's are written to read the raw FASTA files using org.biojavax[21] and java.util.regex API's, to convert the data, so that they can be appropriately stored as tuples in database. The system also contains POJO's developed using Java RMI, to transfer the data to remote File Server and back, when needed. The end-client can request to download the data stored in the database in form of a webpage or can request to download the FASTA file from file server. The end-users can request for data, using various criteria like a particular species, or sequencing run, or reads of a specific length. The DNA sequencing reads may be stored organism wise. Moreover, many sequencing runs may be executed for each organism or species. The reads table maintains records along with this species and run information. To store analysis data generated after analyzing these raw DNA reads, the database has also been designed. The system has tables for storing contigs, repeats, coding regions, and genes. Data can be stored in these tables and retrieved across the web. Table 1, Table 2 and Table 3, displays the data dictionary for storing this data. This is just the minimal set of data dictionary used in storing various DNA sequencing and analysis data. The list could be extended when further analysis is to

[21] http://www.biojava.org

be done to store appropriate analyzed data. For handling multiple requests from various clients, Multithreading[22] is used.

In addition to the Web-based application, the research work also involves the distributed processing using multi-threading and RMI programming in Java. The distributed processing has been applied for Recognizing Identical Reads from more than one Fasta files simultaneously. The system can be implemented by initially identifying how many nodes are available for processing. The Java program is developed for finding out available number of active nodes. Once the list of available nodes is identified, based on this dynamic status, load balancing is performed. The Java RMI programs using multiple threads, is executed to do processing of more than one FASTA files for recognizing identical reads. Thus, simultaneous execution of the same program using different data sets can be implemented. Thus, distributed processing for more than one data sets is implemented as part of this research work. The details of RMI programs is mentioned in List of Programs developed in Java. The sample of programs are also put in Appendix D.

2.4.1. Data Dictionary for DNA Sequencing Data

The Data Dictionary defined for storing DNA sequencing data is as specified below:

(The entire SQL create script for storing various DNA sequencing and analysis data in Oracle Database is given in Appendix A)

[22] Guy Steele, Gilad Bracha, Bill Joy, James Gosling, The Java Language Specifications, SunMicrosystems

Table 1. Table structure for storing the species details, for the species or organisms, whose DNA sequencing is performed

Table Name	*Species*		
Column Name	***Type***	***Constraints***	***Description***
Species_id	Integer(10)	Primary key	Id of a species
Species_name	Varchar(100)	Not Null	Name of species

Table 2. Partial snapshot of the table structure for storing the details of the runs or DNA sequencing experiments, performed for the species

Table Name	*Run*		
Column Name	***Type***	***Constraints***	***Description***
Species_id	Integer(10)	Foreign key, References Species Table	Id of a species
Run_id	Integer(2)		Id of run (Number of runs executed for the given species)
Dt_of_run	Date	Not Null	The Date on which run was executed

Table 3. Table structure for storing the details of the Reads generated from DNA sequencing of the given species and during a given run

Table Name	*Read*		
Column Name	***Type***	***Constraints***	***Description***
Id	Integer(10)	Auto generated	Id of the read
Species_id	Integer(10)	Foreign Key, References Species Table	Id of a species

Run_id	Integer(2)	Foreign Key, References Run Table	Id of Number of runs
Read_id	Integer(10)		Id of a each read
Read_name	Varchar(100)	Not Null	Id generated by the DNA sequencing machine, for each read
Read_rank	Integer(10)		Rank of each read
Read_x	Decimal(10,3)	Not Null	X attribute of each read
Read_y	Decimal(10,3)	Not Null	Y attribute of each read
Read_length	Integer(6)	Not Null	Length of each read
Read_sequence	Long Text / CLOB	Not Null	Sequence of each read
File_name	Varchar(100)		Name of the file

2.5. Features/Technologies Implemented in Distributed Application

The technologies/architecture/features implemented in Distributed Application consist of (As shown Figure 4.)

- Web-based application – The client may use any Web-browser to access the software for Distributed application. The application has been tested using Internet Explorer and Mozilla Firefox
- Open Standard – The web-application uses XML open standard for interfacing between various loosely coupled components, which are communicating with the Web-Server.
- M-V-C Architecture – The software has been developed using Model-View-Controller design pattern. Hence, all the components of the application such as the model, view, and controller are all separate from each other and put in different classes working as independent layers or components of an application, which are easy to manage. View contains the user interface, the model comprises of the set of programs containing business logic as well as data and the controller is the set of programs which controls the flow of application
- Distributed approach - Application has been developed and tested such that all the components i.e. Web-Server, Database-Server, File-Server and client are placed/ executed on different computers of a network.

- Transparency - The end-user of the system i.e. the client, submits the request to the Web-server through web-browser. All the back-end distributed usage and execution is unknown to the client. Hence, transparency feature of Distributed Computing has been implemented. The end-user is unaware of where the application is executing or where the data is stored. Without having the explicit knowledge about the application, the end-user can access the application by simply submitting the URL of the application, over the Intranet/Internet.

- Loose coupling of systems, heterogeneous computing capability and fault-tolerance, or handling of failure is used for the distribution of components. Since, the application is developed using layered approach of M-V-C architecture, each layer or component can be easily altered without disturbing any of the other components. Loose coupling of systems has been implemented by writing server-end programs like Servlets/JSP, for handling user requests across the Web. Implementing the business-logic for storing the FASTA files into File Server, extracting the data from FASTA files and converting it into appropriate form so that it can be inserted into various tables of the database, Handling JDBC connectivity, Forming SQL statements based on users parameters, for executing queries, generating proper format of the output of queries, in form of web-page. All the above programming aspects are written in separate classes and each class communicates with the other using message-passing techniques like RMI.

- Heterogeneity – As shown in Figure 4, Heterogeneity is handled as components like File Servers were implemented both on Windows as well as on Linux file systems. Different Database Servers like Oracle and MySQL were used to store data in databases. Java programming language is primarily used for application development, so that program communication can be handled seamlessly across platforms. Moreover, the Web-Server used for testing is Apache Tomcat server, but the application can be deployed on any Application Server like WebSphere or WebLogic which supports Java based applications.
- Load Balancing and Scalability – Since the application is deployed in Apache Tomcat load balancing can be achieved by configuring it for Load Balancing and Scalability. To do so Apache HTTP server is also required.
- Various Exceptions were handled for smooth communication between components.
- Thin client application – The application developed does not require any programs/software to be installed on the client machine, for executing the application. The application can be accessed by merely submitting the HTTP request to the Web Server where the application is hosted. The execution of business logic takes place on the server side, since Servlets are used in development.
- Multiple Client Accessibility – Many clients can access the Distributed Application concurrently. This facility is provided by

implementing the application in multithreaded Web-Container environment.

- Database Management System Independent – The data storage can be done on any Database. It has been tested to work on Oracle 10g and MySQL 5.5 Database Server. Apache Tomcat also facilitates JDBC Connection Pool, which is used to store/acquire data from multiple database.
- Operating System Independent – Can be executed on any Operating System. Application has been tested on Windows and Linux operating systems.
- File System Independent - The File-Server has been developed for centralized storage of DNA sequencing data files. The File-Server can be implemented on FAT 32, NTFS and ext3 file system.
- Location Independent – The application being Web-based is accessible from any location on the network/internet from wherever the URL is accessible i.e. security privilege is provided.
- Open Source – All the tools and technology used in developing the Distributed Application like Java Development Kit, Apache Tomcat Web Server, MySQL Database Server, Struts Web Framework, Hibernate ORM tool,
- Linux OS, Mozilla FireFox are Open Source under General Public License. (Albeit, application has also been tested to work on proprietary software).

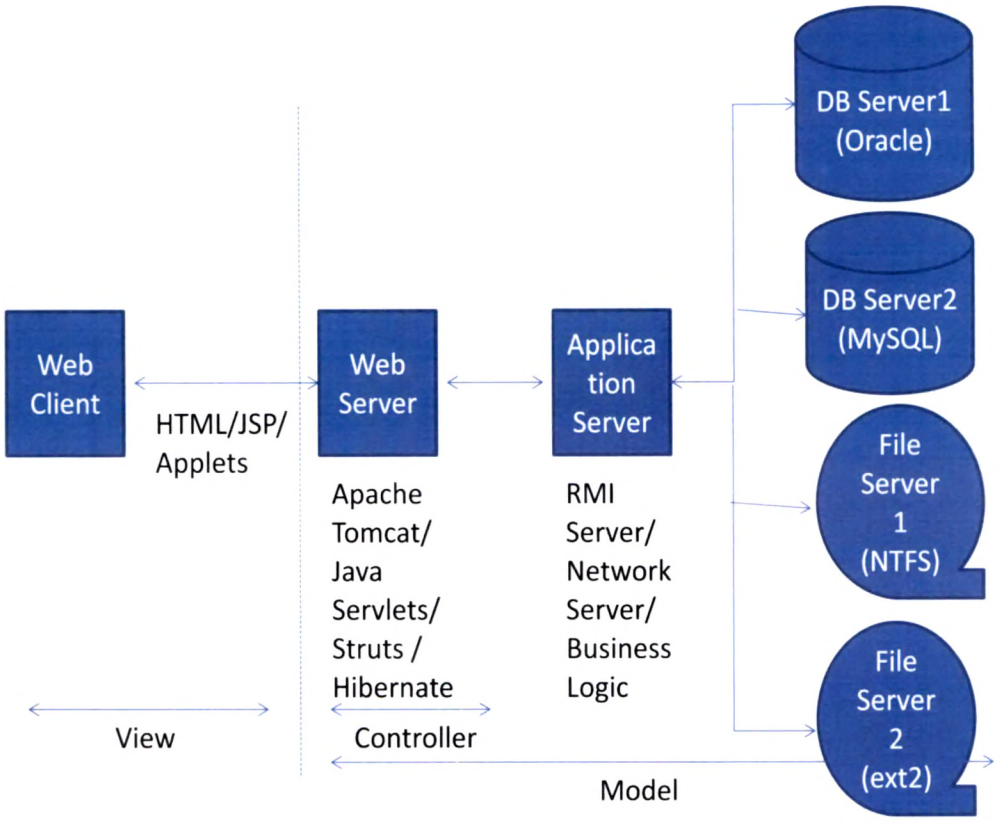

Figure 4. Technologies used to implement the Distributed Application

- The Distributed application uses Java POJO's for business logic.
- Servlets on the Server side as a controller, JSP as a view.
- Struts Web Framework is used to implement the M-V-C architecture.
- Hibernate is used for Object-Relational Mapping.

2.6. Facilities provided through Distributed Application

The facilities provided by the Distributed Application for BioInformatics data are:

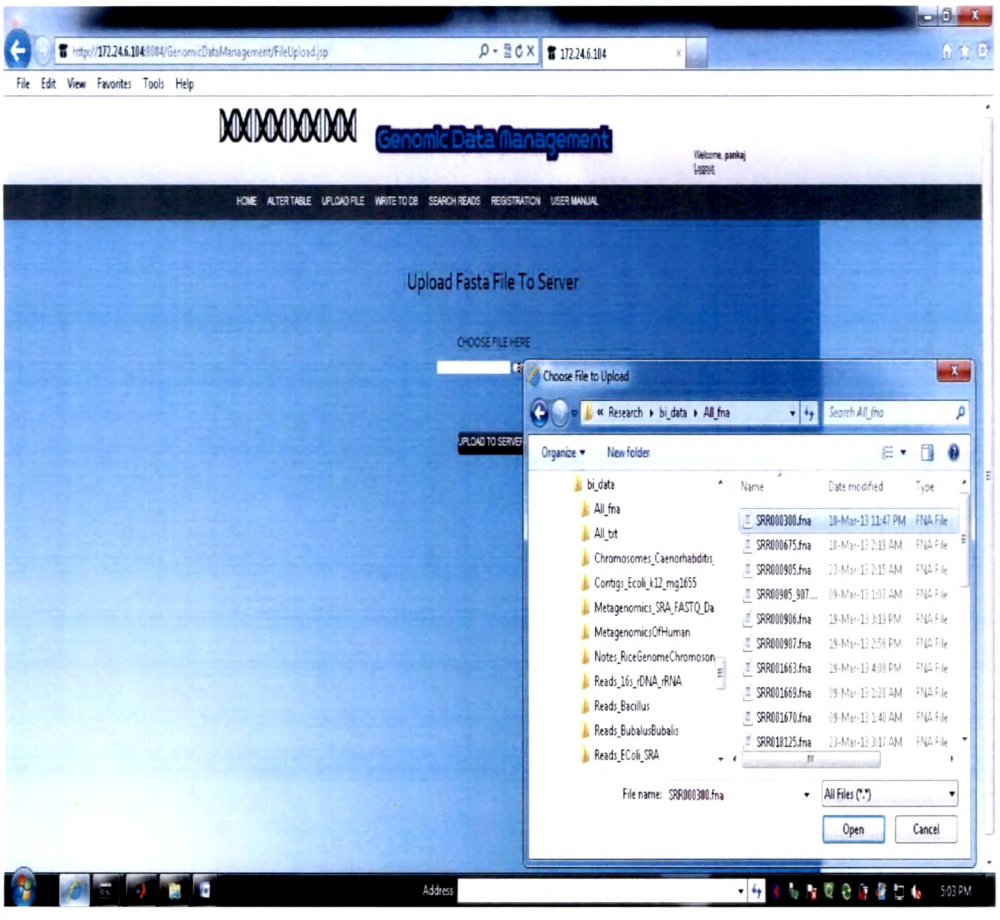

Figure 5. User Interface with the Distributed Web Application, to Upload the FASTA file containing DNA Sequencing Data

- Uploading of Data Files which are in FASTA format (As shown in Figure 5)
- Storage of Data Files on central File Server
- Storage of Data Files on central Database Server

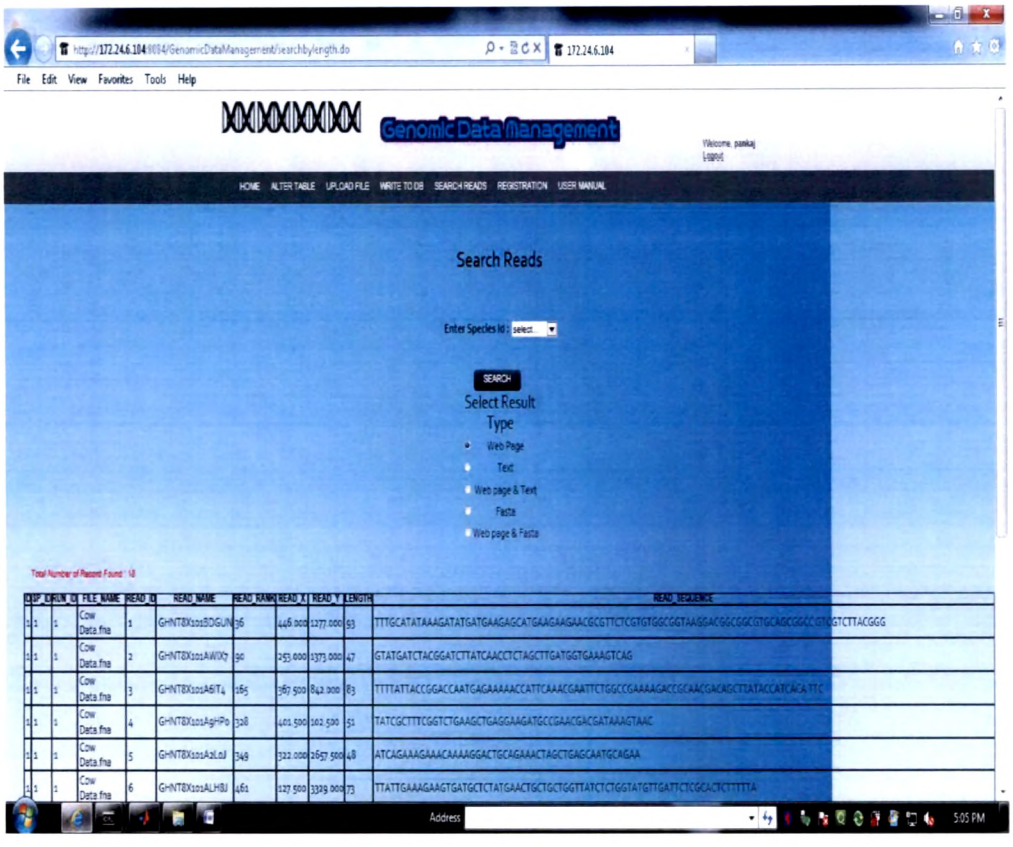

Figure 6. Webpage displaying the Result of the query fired on the database residing on the remote machine

Search for DNA data based on various criteria (As shown in Figure 6.)

- Possibility of firing customized queries or native SQL
- Download the search result in Web-Page or in FASTA file
- Download the original data file, if required for further processing
- Centralized storage of data
- Data sharing within the organization
- User Registration for open accessibility
- User-wise accessibility for security/privacy of data

2.7. Implementation Issues

The implementation issues tackled during the development of the Distributed Application are:

- Making use of Open Standards, so that appropriate collaboration can take place between the distributed components.
- Tackling with transparency, heterogeneity, and failure handling, through appropriate message passing and exception handling concept implementation.
- Identifying and Designing the components of a Distributed Application, so that they are placed/ distributed appropriately at different locations/computers on a network.
- Developing all the collaborating interfaces, so that each component can communicate with each other as required.
- Installing and Configuring the Web-Server, the Database Server, the File Server for various facilities.
- Providing appropriate privileges across all the components and computers to make the application working
- Dealing with Firewalls implemented on different computers of the network
- Writing POJO's using regular expressions, to read and extract the information from the data files. This was important because data files are unique in format for the biological data. These programs were essential for converting the unstructured

data into objects, which, could then be stored in a structured format in the database using Object-Relational Mapping.

- Using specialized API like org.biojavax to apply it for bioinformatics data.

2.7.1. Practical issues dealt with while developing, executing and deploying Distributed Application

Besides dealing with conceptual aspects, several miscellaneous small and big issues were tackled while developing, executing or deploying the distributed application like:

- Just to open the huge Fasta file containing just a single chromosome, it needed a gvim software which is vi editor for Windows because the Windows editors like Notepad or Wordpad simply hang, as they are not designed to handle such big files. Opening in MS Word is dangerous because even slightest formatting happening to the text will add some control characters to the file, and hence the original data can get corrupted. Eg.: A single file containing only one dataset i.e. Chromosome 7 of Human Being is 1259 pages long when opened with MS Word. The actual size of FASTA file is 4780 KB.
- To execute a java program dealing with this large data, one needs to use java –Xms and –Xmx option to deal with Out of Memory errors.
- Simply running java command with –Xms and –Xmx is not enough. One needs to increase the default Virtual Memory size in Windows OS so that additional heap space can be allocated to complete the java command with these options.

- When trying to insert the large dataset into MySQL Database with default parameter settings will not be sufficient. SET GLOBAL max_allowed_packet parameter needs to be modified, to enable insertion of a single large dataset.
- The appropriate JDBC Driver API versions or driver types and correct, precise versions of connector classes or url should be used to establish JDBC connection. Either the incorrect driver version or the mismatched Database version will keep one struggling to resolve errors in ones own program but the fault is lying outside the program.
- While implementing File server, read/write/execute privileges must be pre-assigned in file system, so that application can read/write data on the given File Server.
- While implementing the distributed processing the security controls defined in Anti-Virus software or firewall settings need to be modified, to enable the processing to begin on that machine. Just availability of machine or privileges through operating system, is not sufficient.
- When working with Java APIs, appropriate classpath settings need to be done particularly when using sub-packages or packages available in form of .jar files.
- While working with RMI programs the exceptions of type RemoteExceptions which deal with number of communication related exceptions are very tricky and have to be handled carefully.

- The resource locking for concurrency controls, should be taken care using synchronized methods, when writing multithreaded programs.
- When working with Matlab, the Wavelet Toolbox and BioInformatics Toolbox should be procurred, as they are add-on toolboxes in Matlab. These would be required to run the program for recognizing identical reads, using distributed processing.
- The default port 8080 of Apache Tomcat server should be changed, before use, because, if Oracle 8i or above DBMS software is installed on the same machine, then the request to port 8080 is always redirected to Oracle Homepage and not Apaches' Homepage, hence disallowing access to web-applications.

List of Java Programs that are developed for various aspects of applications are stated in the following table.

2.7.2. List of Java Programs

Program 1. FinalMainMenu.java
Program 2. FinalMainRead.java
Program 3. Read.java
Program 4. ReadFromFastaFile.java
Program 5. ReadToDB.java
Program 6. ReadFromDB.java
Program 7. CreateInsertScriptFile.java
Program 8. GenerateDNAToken.java
Program 9. GenerateDNALookupTable.java
Program 10. ConvertSequenceToBinary.java
Program 11. ConvertSequenceToLookupArray.java
Program 12. FinalMainContig.java
Program 13. Contig.java
Program 14. ContigFromFastaFile.java
Program 15. ContigToDB.java
Program 16. ContigFromDB.java
Program 17. CreateInsertScriptFile.java
Program 18. FinalMainChromosome.java
Program 19. Chromosome.java
Program 20. ChromosomeFromFastaFile.java
Program 21. ChromosomeToDB.java
Program 22. ChromosomeFromDB.java
Program 23. IPDemo.java
Program 24. Scan.java
Program 25. DistServer.java
Program 26. DistImpl.java
Program 27. DistClient.java

Table 4. Brief Description of Java Programs

Sr. No.	*Name of Java File*	*Java Class Name*	*Purpose*
1.	***FinalMainMenu.java***	FinalMainMenu	The class that provides Menu, the Interface to work with different form of DNA sequencing data like Reads, Contigs, Chromosomes
2.	***FinalMainRead.java***	FinalMainRead	The class that provides Menu, the Interface to work with different operations on the Reads
3.	***Read.java***	Read	The class that defines the attributes and methods for Read objects
4.	***ReadFromFastaFile.java***	ReadFromFastaFile	The class that does file handling to read the data related to DNA sequencing READs from the Fasta files.
5.	***ReadToDB.java***	ReadToDB	The class that writes the records about the DNA sequencing READs to a Table storing the Reads in the Database Server
6.	***ReadFromDB.java***	ReadFromDB	The class that fetches the READs from the Database and displays it to the client

7.	***CreateInsertScriptFile .java***	CreateInsertScriptFile	The class which creates the SQL Insert script dynamically through a Java code, after reading the data from the FASTA file about the READs that exist in the FASTA file. This is needed because the Inserrt script could be used independently to insert the data into the Database directly without involving the Java program if the need be.
8.	***GenerateDNAToken.java***	GenerateDNAToken	The class that defines the objects for defining the mapping between nucleotides bases in DNA sequence to binary values using 2-bit indicators and then combining binary values of 4 nucleotides to define a single byte value. This class is used in algorithm designed for Data Reduction
9.	***GenerateDNALookupTable.java***	GenerateDNALookupTable	The class used to fetch the value for the group of four nucleotides into the int value store in one byte.

10.	***ConvertSequenceToBinary.java***	ConvertSequenceToBinary	The class that actually does the mapping to convert DNA sequence into binary form and then store four binary vlues in a single byte. The mapping can be used for forward as well as reverse conversion
11.	***ConvertSequenceToLookupArray.java***	ConverSequenceToLookupArray	The class that generates the array of integers, representing the single DNA sequence into an array of numbers, which are used as an initial Digital Signal, which is further processes using Wavelet Transforms, for Data Reduction.
12.	***FinalMainContig.java***	FinalMainContig	The class which displays the menu to perform various operations related to the Contig data
13.	***Contig.java***	Contig	The class that defines the objects which deal with the Contigs data which are store in The FASTA files containing the Contigs.

14.	***ContigFromFastaFile.java***	ContigFromFastaFile	The class that does file handling in java to read the data from FASTA files containing the Contigs
15.	***ContigToDB.java***	ContigToDB	The class that contains the JDBC code to interact with the Database to store the Contigs data into the Contig Table created in the Database.
16.	***ContigFromDB.java***	ContigFromDB	The class that contains the Java code to fetch the data from Contigs table and give it to end-user.
17.	***CreateInsertScriptFile.java***	CreateInsertScriptFile	The class that reads the Contigs data from the FASTA file and creates the SQL Insert Script dynamically using Java program so that it can be used later on for offline to do insertion into database tables offline if required.
18.	***FinalMainChromosome.java***	FinalMainChromosome	Class which displays the menu to perform various operations related to data handling for Chromosomes

19.	***Chromosome.java***	Chromosome	The class which defines the object members for the Chromosome data.
20.	***ChromosomeFromFastaFile.java***	ChromosomeFromFastaFile	The class that reads the FASTA file containing the Chromosome data
21.	***ChromosomeToDB.java***	ChromosomeToBD	The class that uses JDBC code to do insert operations on table storing Chromosome data
22.	***ChromosomeFromDB.java***	ChromosomeFromDB	The class that fetches the Chromosome data from the table in database
23.	***CreateInsertScriptFile.java***	CreateInsertScriptFile	The class that creates SQL insert script to perform insertion of Chromosome data into table after reading the FASTA file. This program dynamically creates the Insert Script.
24.	***IPDemo.java***	IPDemo	Takes the two IP Addresses as parameters and calls the Scan program which scans whether the Machines having the IP Addresses falling in the given range are active or not at the given point of time.

25.	*Scan.java*	Scan	Class containing the code which Does the actual scan to check whether a machine with the particular IP Address is actually active or not so that it can be used for processing the application which needs to be Distributed across the network. It uses several network related commands and algorithms.
26.	*DistServer.java*	DistServer	The RMI program containing the server part of the code which maps the object of the class with URL to call the particular method which needs to be invoked remotely.
27.	*DistImpl.java*	DistImpl	The class containing the code of the method which needs to be actually invoked remotely. This is needed because the client which requests for processing does not actually have the code of the process, it needs to execute.