3. SYSTEM ANALYSIS AND DESIGN

Design is the first phase in the development for any engineers' project. It is a creative process. A good design is a key to effective system. From a project management point of view, software design is conducted in two steps. Preliminary design is concerned with the transformation of requirements in to data and software architectures. Detailed design focuses on the refinements to the architectural representation that lead to detailed algorithm data structure and representation.

3.1 DETAILED DESIGN

Input design:

The input here may be of two kinds. One is the entire files of a particular directory and the other is a set of files that contain numerical values. The majority of files in the directory will be system files or in built application files but the user files may aslso be added. For implementing the file content sorting, we created a set of files that contain unsorted numerical values of larger order in a particular directory.

Output design:

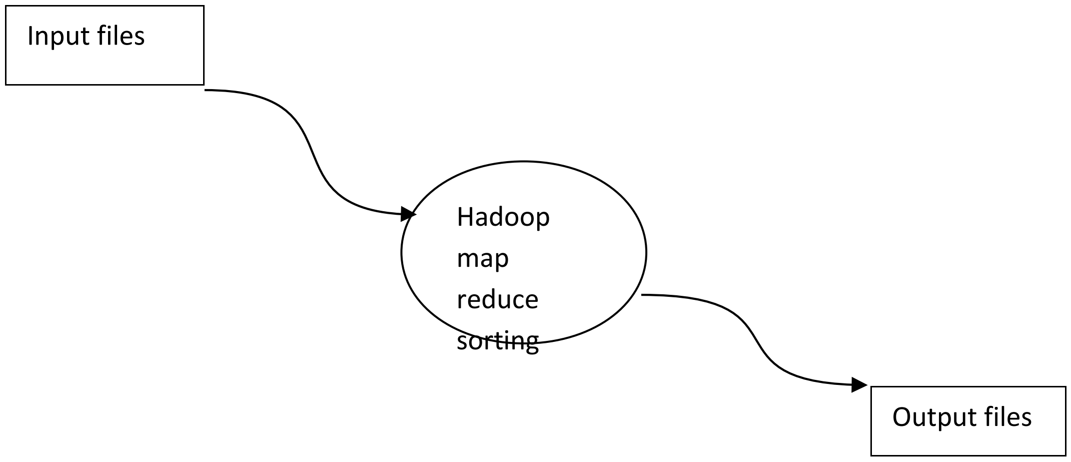
This design part mainly includes the creation of a front end for the system using java swing language that makes it user friendly. The front end of the application mainly includes a login window where the administrator enters his or her username and the respective password. After the verification process, three choices are put before: file sorting, file content sorting and viewing the unsorted files.

Program logic:

As mentioned before the logic runs on Hadoop map reduce framework. There are three modules: mapper, reducer and worker. Mapper is an interface that is implemented in the worker module. There may be any number of mappers based on the number of input files. Each mapper is manipulated as individual threads that run independently. Each mapper sorts the respective file which is done when this interface is implemented in the worker module. The worker module passes the results of every mapper to the reducer. The reducer aggregate the common key value pairs and merges the individual outputs of mappers with efficacy.

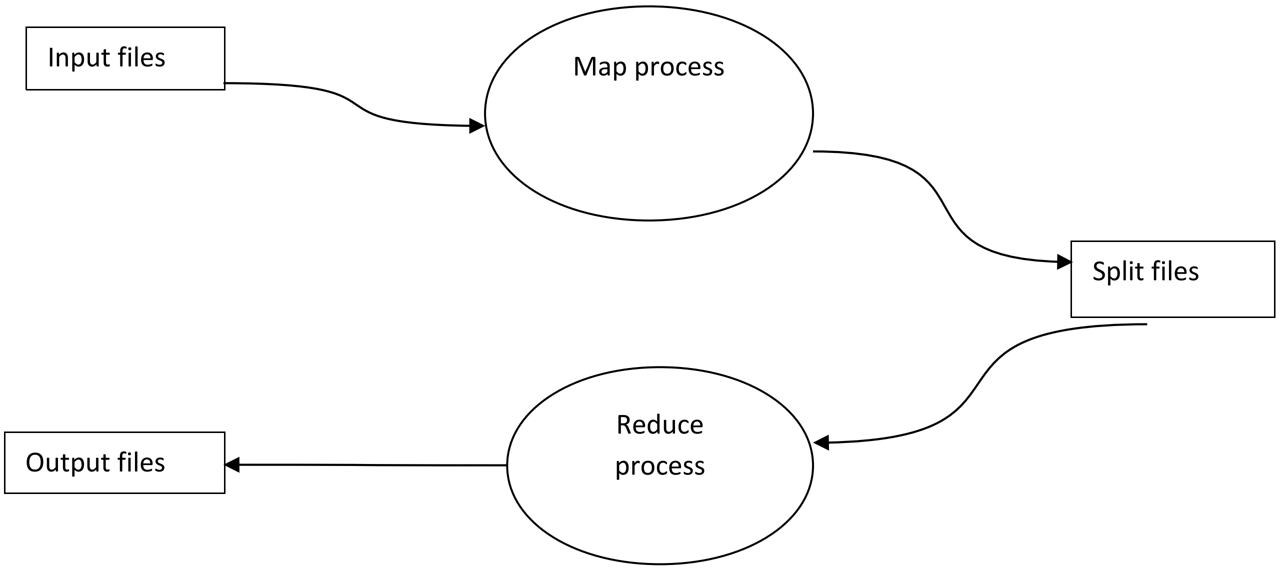
3.2 DATA FLOW DIAGRAM

1. DFD 0



Zeroth level of the data flow diagram defines the overview of what is really the process behind the project. In this level, it is mainly stated that an input set of files is given to the map reduce algorithms that sorts them and results into output set of files.

2. DFD 1



This is the first level of data flow diagram. This level defines the detailed description of the zeroth level of the data flow diagram. Here we input a set of files to be sorted into the mapper function which splits the files into multiple chunks. It then sorts these files in individual parts and then they are combined through the reducer process and give the final outputs.

4. SYSTEM SPECIFICATION

4.1 HARDWARE SPECIFICATION

The selection of hardware is a very important task related to the software development.

System	:	IBM compatible PC
Processor	:	Intel Pentium IV
Memory	:	1GB RAM
Hard Disk Drive	:	40 GB

4.2 SOFTWARE SPECIFICATION

1. Fedora 18

Fedora, formerly **Fedora Core**, is an RPM-based, general purpose collection of software, including an operating system based on the Linux kernel, developed by the community-supported Fedora Project and owned by Red Hat. The Fedora Project's mission is to lead the advancement of free and open source software and content as a collaborative community. The version Fedora 18 is used that provides the operating system.

2. Hadoop framework

Apache Hadoop is an open-source software framework that supports data-intensive distributed applications, licensed under the Apache v2 license. It supports the running of applications on large clusters of commodity hardware. The module of map reduce is made use of here.

3. Eclipse

In computer programming, Eclipse is a multi-language software development environment comprising a base workspace and an extensible plug-in system for customizing the environment. It is written mostly in Java. The map reduce algorithm is coded with java that runs on the platform of eclipse. Java Development kit 1.7 is used.

4. Java Swing

Swing is the primary Java GUI widget toolkit. It is part of Oracle's Java Foundation Classes (JFC) — an API for providing a graphical user interface (GUI) for Java programs. Swing provides a native look and feel that emulates the look and feel of several platforms, and also supports a pluggable look and feel that allows applications to have a look and feel unrelated to the underlying platform. The designed program code is made user friendly with the help of Swing.

5. IMPLEMENTATION

The installation and configuration can be summarized in following steps and later described in brief.

5.1 INSTALLATION

1. Install FEDORA (Any version above Fedora 6), a Linux based operating system.
2. Install the java development kit. It requires jdk 1.6.0 at a minimum.
3. Install hadoop package 0.22.0
4. Install eclipse, the platform for programming in java.

5.2 SETTING THE CLASS PATHS

5. login as root user in FEDORA
6. extract the java and hadoop folders in root directory home
7. set the class path of java and hadoop in .bash_profile as follows:

```
export JAVA_HOME=/root/jdk1.7.0
export PATH=${JAVA_HOME}/bin:${PATH}
export HADOOP_HOME=/root/hadoop
```

export PATH=${HADOOP_HOME}/bin:${PATH}

8. run source ~/.bash_profile
9. check java and hadoop commands
10. check ssh localhost
    10.1 If connection refused port 22

10.2 run command service sshd start

11. Test the sample program in hadoop
    11.1 hadoop jar hadoop-mapreduced-examples0.22.0.jar pi 2 5
    11.2 if it is ok hadoop installation is completed

5.3 HADOOP CONFIGURATION FOR JAVA PROGRAMS

1. edit some files in hadoop/conf folder
2. first edit the hadoop-env.sh file
3. here edit the java class path to /root/jdk1.7.0
4. run this sh file for it updating command is---- sh hadoop-env.sh or bash hadoop-env.sh
5. the edit core-site.xml

```
6.      <configuration>
7.          <property>
8.              <name>fs.default.name</name>
9.              <value>hdfs://localhost:9000</value>
10.         </property>
11.     </configuration>
```

1. Edit hdfs-site.xml

```
2.      <property>
3.              <name>dfs.replication</name>
4.              <value>1</value>
5.          </property>
```

6. edit mapred-site.xml

```
7.      <property>
8.              <name>mapred.job.tracker</name>
```

9. <value>localhost:9001</value>
10. </property>

11. The run the command bin/start-all.sh
12. run command jps
13. list the datanode
14. secondarynamenode
15. jps
16. command for running dfs section
17. create directory hdfs and create two sub dir name and data under hdfs
18. Edit hdfs-site.xml add following statements
19. <name>dfs.data.dir</name>
20. <value>/root/hdfs/data</value>
21. </property>
22. <property>
23. <name>dfs.name.dir</name>
24. <value>/root/hdfs/name</value>
25. </property>
26. then go to bin
27. stop-all.sh command
28. hadoop namenode -format command
29. start-all.sh command
30. install the hadoop map-reduce plugin

A single-node Hadoop installation is done so that users can get a flavor of the Hadoop Distributed File System and the Map-Reduce framework i.e. perform simple operations on HDFS, run example/simple jobs etc. Hadoop is run on a single-node in a pseudo-distributed mode where each Hadoop daemon runs in a separate Java process. This is enabled by editing conf/hadoop-site.xml as mentioned above

ssh must be installed and sshd must be running to use the Hadoop scripts that manage remote Hadoop daemons. The passphraseless ssh is set up so that we can ssh to local host without a pass phrase.

There are a handful of files for controlling the configuration of a Hadoop installation. The most important ones are hadoop-env.sh, core-site.xml, hdfs-site.xml and mapred-site.xml. All these files are located in the conf subdirectory. Hadoop-env.sh deals with the bash script environment variables that are used in the scripts to run hadoop.core-site.xml deals with the configuration settings for hadoop core, such as I/O settings that are common to HDFS and MapReduce.hdfs-site.xml deals with the configuration settings for HDFS daemons: the namenode, the secondary namenode and the data nodes. Mapred-site.xml deals with the configuration settings for map reduce daemons: the job tracker, and the task trackers. Masters run a secondary namenode and job trackers and slaves run a datanode and a task tracker.

To verify whether installation is proper or not we started the daemons. This will start up a namenode, datanode, jobtracker and a task tracker. Two sample programs provided with the hadoop package are executed. They are the programs for computing the value of pi and word count. The approximate value of pi is obtained as the output for the first program and the number of occurrences of each word in a text is obtained as the output of the second program.

5.2 HDFS STRUCTURE AND MAPREDUCE

A Hadoop cluster running HDFS as a file system always has a central Name-
Node which handles the data distribution and any operations on the data. It communicates with the Data Node which is running on every connected node and performing the disk access. Every DataNode periodically reports a list of its blocks to the NameNode so the NameNode does not need to write this information to disk all the time. The NameNode just persistently stores information about the file system-tree and the metadata of the files. The Data Nodes \are responsible for serving read and write requests from the file system's clients. The Data Nodes also perform block creation, deletion, and replication upon instruction from the NameNode. The read- and write-
requests from clients are managed by the NameNode but executed by the Data-nodes because the NameNode has the file systems' metadata and knows which nodes store the requested blocks. The user data is never going through the NameNode because this would quickly saturate its' network connection.

Hadoop Map-Reduce is a software framework for easily writing applications which process vast amounts of data (multi-terabyte data-sets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner. The computation takes a set of input key/value pairs, and produces a set of output key/value pairs. The user of the Map Reduce library expresses the computation as two functions: Map and Reduce. The division into the two phases, Map and Reduce, permits the framework to distribute the work without respect to the amount of data or the number of nodes operating in parallel. Every Map operation is self-contained and the Reduce tasks get all output data for one key so there is no overlapping in the work of the Reduce tasks. The following sections explain the Map and Reduce phase in more detail.

Map

The Map function as used by Map Reduce is provided with a list of key-value pairs as input data and returns a list of key-value pairs as its' output. The keys and values from the input do not need to be of the same type as the output. All values are then grouped and sorted by the key and sent to the Reduce function.It is optionally possible to call a user-defined Combine function before the data is sent to minimize the usage of bandwidth. The Combine function is mostly the same as the Reduce function but is only applied to the output of one Map task at once.

Reduce

The Reduce function in the context of Map Reduce is invoked with the intermediate output of the Map functions, grouped and sorted by the key. These values are typically merged together and a possibly smaller amount of key-value pairs is produced as output.

5.3 APPLICATION

The file sorting is done through the following steps:

1. Write the program code in eclipse platform as different classes of mappers and reducers
2. Run as a hadoop application
3. The output is generated.

6. TESTING

Testing is a dynamic technique of verification and validation, which ensures that the software meets the expectations of the user. It involves executing an implementation of the software with test data. Test data is the set of data that the system will process as a normal input. System testing verifies that whole set of programs hang together. It makes a logical assumption that if all the parts of the system are correctly, the goal will be successfully achieved .The whole system has been tested and found to be working.

Testing is the crucial to any software that is being developed. The various levels of testing are:

1) Unit Testing.
2) Integration Testing.
3) Validation Testing.
4) Output Testing.

The following are the ways that we performed tests on our product.

6.1 Unit Testing

In this section each modules is tested as a separate entity. The following are the various modules that are contained within:

1. <u>The user interface</u>

Here the login window for the administrator name and password was tested. It shows “Invalid admin name or password when the proper words are not given. The menu driven window contains 3 choices: file sorting, file content sorting and viewing unsorted files. It was tested that whether the user is driven to the specified destination based on the choice.

2. <u>File sorting</u>

The sorting of files on the basis of length, name and date are tested. The different testing is done on the files of the same directory so that different sorting can be verified easily.

3. <u>File content sorting</u>

The numerical values of different orders and different numbers were given in the files and the outputs were verified. If an empty file is given, the output screen will be blank.

4. Viewing unsorted files

This is tested just by clicking the button that performs the required event. This shows the unsorted files in the particular directory.

6.2 INTEGRATION TESTING

The modules tested above are combined into groups and the output as a whole is verified.

6.3 OUTPUT TESTING

The whole package with different classes is run on eclipse as a Hadoop application with the different cases specified in unit testing and the output was verified.

7. CONCLUSION

Depending on the data processing needs, hadoop workload can vary widely over time. We may have a few large data processing jobs that occasionally take advantage of hundreds of nodes, but those same nodes will sit idle the rest of the time. We may be new to hadoop cluster and may own a startup firm that needs to conserve cash and wants to avoid the capital expenses of a hadoop cluster. In these and other situations, it makes more sense to rent a cluster of machines than buying it.

The rapid adoption of hadoop at facebook, flicker, yahoo etc has been aided by a couple of key decisions. First, developers are free to write map-reduce programs in the language of their choice. Second, we have embraced SQL as a familiar paradigm to address and operate on large datasets. Most data stored in hadoop file system is published as tables much like they would do with a good old database. When they want to operate on these data sets, they can use a small subset of SQL to specify the required dataset. Operations on datasets can be written as map and reduce scripts or using standard query operators or as a mix of the two. Over time, they have added classic data warehouse features like partitioning, sampling and indexing to this environment. This in-house data warehousing layer over hadoop is called Hive and they are looking forward to releasing an open source version of this project in the near future.

The file sorting using hadoop framework coupled every advantages of using a distributed system. It works efficiently with data on any scale. The required software were installed and configured. Sample programs were executed. Then we had move on to our application of file sorting. A sample of files was sorted and output as the hadoop application is obtained. Beyond some technical reasons and time constraints we achieved almost our goal.

8. FUTURE ENHANCEMENTS

The results we obtained were not as good as we expected. Hence we plan to conduct more work on this project in the future. Some of the future enhancements that can be done in this project is priority to study the configuration of hadoop clusters in more details and implement a better cluster than the one that we use in the project. This can be used to obtain better results.

Also, finally, we aim to study more heuristic algorithms for say, NP complete problem and try and parallelize them also.