IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Volume-03, Issue-09 E-ISSN: 2347-2693

# Implementation of Hadoop Distributed File System Services in an Application

Shincy Anto[1] Anila Ramesh K[2], Ladlie Dias[3] and Sabna A.B[4]*

[1,2,3,4*]*Department of Information Technology*
*Jyothi Engineering College, Cheruthuruthy, Thrissur, Kerala, India*

www.ijcseonline.org

Received: Sep /01/2015 Revised: Sep/10/2015 Accepted: Sep/26/2015 Published: Sep/30/ 2015

***Abstract*—** Cloud is the collection of computers on the internet that is being offered as a revolutionary storage method for files and Big Data is an evolving term that describes any voluminous amount of structured, semi-structured and unstructured data that has the potential to be mined for information. In our proposed system, we implement a safe method to save Big Data files using Apache Hadoop technology. And here we provide two web services: uploading and downloading of files. Once we upload the files, they are stored in different nodes across the cluster after partitioning. We make the data highly available, by keeping the different copies of data in several nodes. The uploaded files can be retrieved at any time according to our needs. Even if there is a failure at any node, the copy of data on some other nodes makes our system fault tolerant and reliable. Along with that, we can also make our files public or private and thus providing file sharing facility. We offer a security system for storing and managing Big Data files.

***Keywords*—** Big Data, Fault tolerant, Hadoop Distributed File System

## I. INTRODUCTION

In a world where business units are becoming more self sufficient, it is more important to identify the value of data and its interaction. The backend data comes together to provide necessary ingredients to enable better decision making in the organizations. Data has always been big. Big Data is therefore an all-encompassing term for any collection of large data sets that were once difficult to process.

Data is streaming at an enormous speed and must be dealt with in a timely manner. A lot of factors contribute to the increase in volume and types of data. Many of the recent applications like banking transactions, social media, index web searches, machine learning plays their part in increasing the volume of data. Along with it, logs, blogs, emails, and other structured [6] and unstructured [6] information streams also contribute to it [1]. It can't be effectively managed by using conventional data management tools. Managing, merging and governing different varieties of data are something many organizations still grapple with. The way to store, process and analyze this large volume of data is by using Hadoop

Loss of these data means, a lot of consequences. Studies say that 94% of companies suffering from a catastrophic data loss do not survive where 43% never reopen and 51% close within two years. (University of Texas) 7 out of 10 small firms that experience a major data loss go out of business within a year. 96% of all business workstations are not being backed. And the different reasons for data loss are intentional or accidental deletion of files, administration errors, inability to read unknown file format, data corruption etc. Power failure or hardware failure may also cause the loss of data. Along with it software bugs, worms and virus are major threats for data confidentiality and security now a days.

Hadoop Distributed File System is an open-source Java-based file system that provides scalable and reliable data storage designed to span large clusters of commodity servers. It stores large amount of data in different nodes and they can be streamed according to the user applications. The metadata and application data are kept in separate places [2] i.e. the former one is stored in the name node [4] and the latter one in the data nodes [3]. We have developed a distributed, reliable, fault tolerant file storage service using an Apache Hadoop framework. Users can register to it and authenticated users only are able to perform uploading and downloading of files. Our project also offers an efficient file sharing and security system for storing and managing Big Data.

## II. LITERATURE SURVEY

Sivaraman, E et al. in the paper on High Performance and Fault Tolerant Distributed File System for Big Data Storage and Processing Using Hadoop discusses about Big Data evolution as well as the Hadoop Distributed File System [3]. The future of Big Data is portrayed using Gartner's Hype Cycle. Hadoop's Map Reduce paradigm [5] for distributing a task across multiple nodes in Hadoop is also discussed with simple data sets. The paper tells what Hadoop and HDFS architecture is. Hadoop is fault tolerant and self-healing distributed file system intended to turn a cluster of industry standard servers into a massively scalable pool of storage. It is possible to scale a Hadoop cluster to hundreds and thousands of nodes. It consists of

 

two components: a distributed file system and the computational framework. HDFS uses a write-once-read-many model that breaks data into blocks that it spreads across many nodes for fault tolerance and high performance. Hadoop and HDFS make use of master –slave architecture. The authors conclude in their paper that HDFS produces multiple replicas of data blocks and distributes them on compute nodes throughout a cluster to enable reliable fast computations.

T Ragunathan et al. in paper on Performance Evaluation of Read and Write Operations in Hadoop Distributed File System focus on performance evaluation of HDFS [1]. For this a Hadoop cluster of five nodes was used. The results indicate that HDFS performs well for the files with the size greater than the default block size and performs poorly for the files with the size less than the default block size. It is because of the delays in task scheduling and startup or fragmentation and excessive disk seeks caused by disk contention under concurrent workloads. When an application has to read a file, the HDFS client sends request to the Name Node for the list of data nodes that are having replication blocks in the file. Then client requests the data node for the transfer of required block. During the write process, the client first requests the Name Node to identify available data nodes to host the required number of replicas for the first block of the file. Then client organizes a pipeline from one node to other node and transmits data. First the experiment was conducted by dividing data into number of small files with each file size less than the default block size.

## III. PROPOSED SYSTEM

In our Proposed System we are implementing a Service Oriented Architecture which enables us to provide better and secure big data storage in a distributed manner. The Distributed Architecture has been done by dividing the files based on a predefined threshold value. The stored files can be retrieved based on indexing schemes. Our system has main four processes. They are registration process, login process, uploading and downloading.

In registration process the user has got the opportunity to choose a unique username. He can set a password and thus setup his entire profile. This user profile is stored in the database. Next we have the login process in which the user login by using his username and password. Then the system will verify the login information and if they match, the login is granted. Once the user is logged into the application successfully, he can upload big data files. Also, he can view the list of files he has uploaded. Information like file type, file size, id, URL etc is shown. The user can download his required big data file from the application by simply clicking the URL link of the file. It also allows file sharing among the users. Here we use a name node which stores the index of every files and data node where the distributed files have been stored.

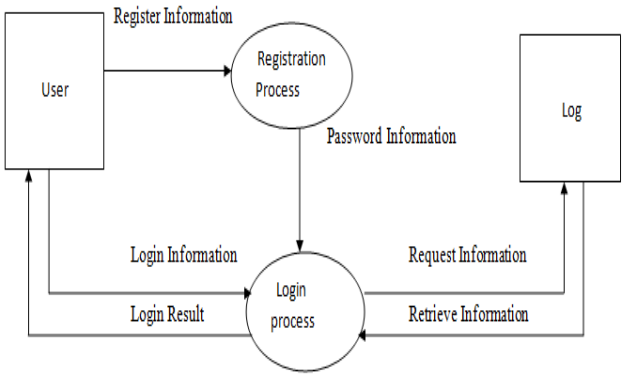


Fig1: Dataflow diagram

The data flow diagram which explains the registration and login process is shown above and working of the proposed application is shown below.

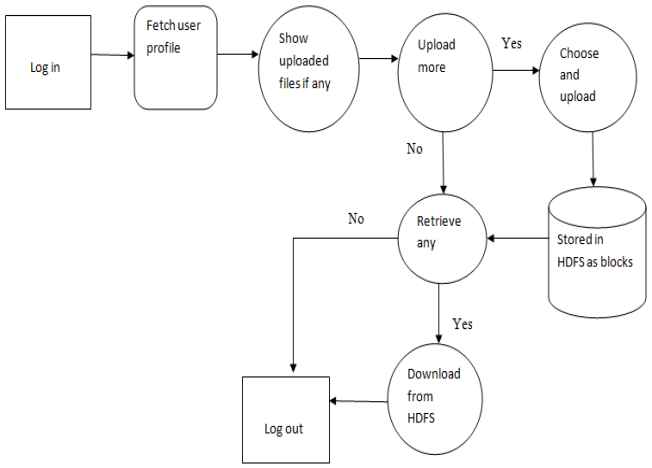


Fig2: Working of the application

## IV. SYSTEM DESIGN AND IMPLEMENTATION

The Project has been implemented in Linux platform using HDFS and Hadoop and MySQL as the database back end. The coding language used is Java and the development tool is Eclipse. Angular JS and Bootstrap frameworks are used for building our proposed system and the main modules of our application are explained below.

### *A. Application Server*

This module helps the user to create an account. It includes the activities of the user such as

*i)* Register in the application

In this phase, the new user can register to our application by providing his/her personal information like name, email address, profile picture etc. He can also set a password to his account which is highly confidential.

*ii)* Log in to the application

Once the user has registered his account, he can login to it whenever he wants by using the email address and password. By logging in to it, the home page of the application will be made available to the user.

*iii)* Select and upload files

After logging in to it, the user can manage his profile by changing the profile settings and also he can upload and download big data files to it. All the files will be stored safely in the hadoop clusters. We can also make the files URL public and private. Once we have made the URL public, then the URL can be copied to any place and can be downloaded from there. If the URL is kept private, the files will remain not accessed by other users. Once all these operations are done he can safely log out of his account.

### *B. Hadoop Distributed File System*

Once the big data files are uploaded to the account, they are divided into several blocks of default size in order to make them manageable. The divided blocks are stored in the data nodes of the hadoop clusters. In order to ensure the safety of the files, we keep copies of each block in several other data nodes. Even if any data loss occurred, the datas can be retrieved from other data nodes. Name nodes are places where the Meta data like name, size, path etc will be saved. So by using these name nodes, we can keep track of the entire big data files.

### *C. Database*

It is an organized collection of related data. It helps in analyzing data. Here all the login activities of each profile are stored. The time of login, time of log out, number of files uploaded and downloaded, size of the files etc will be stored in the database.

## V. RESULTS

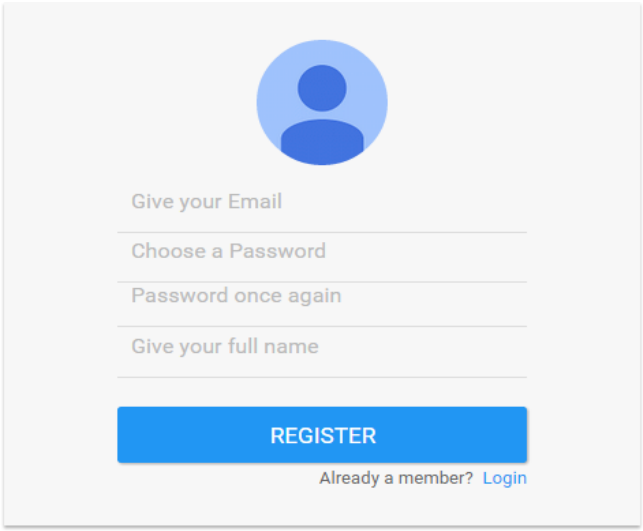


Fig.3 Register page

In our fault-tolerant highly available distributed file system, for the registration process we have a register page where, the user will choose a username and a password. Using this he will create his own profile in the application. As soon as a profile is created, the password and username is stored in the database. The registration page is shown in the Fig.3

Once the user is registerd, he can easily login to his own profile using the username and password. Once the username and password is matched with that stored in the database, login is granted. Fig.4 shows the login page of the application.

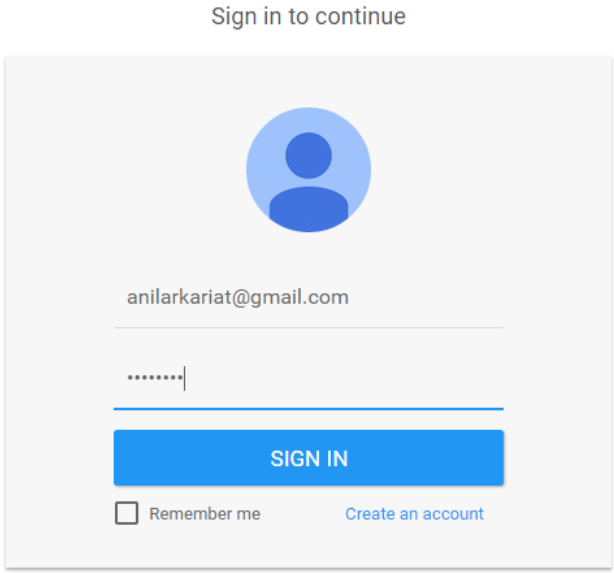


Fig.4 Login page

The home page of the application shows the previously uploaded and downloaded files as shown in the fig 5 we can also make the URL public and private in order to share the files.

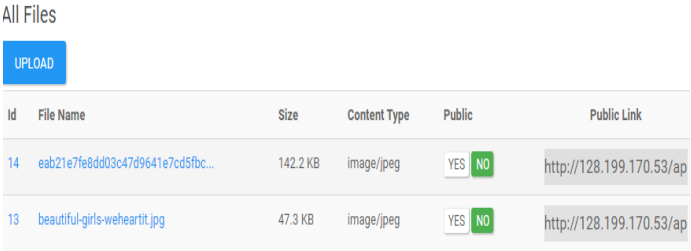


Fig.5 Home page

We can upload and download files to and from the application and the fig.6 and shows the screenshots of these two main processes

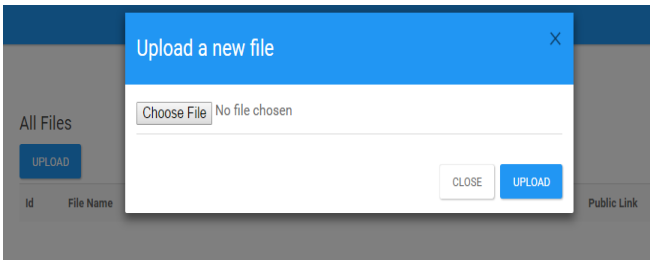


Fig.6 Uplaoding files

## VI. CONCLUSION

The proposed File System scheme promises a usable and safe application mechanism. We are providing a fault tolerant reliable highly available distributed file system which is secure, fast and precise. We assure the safety of data with less maintenance cost in case of any data loss. This helps in fast analysis and decision making at critical times. We could successfully produce services like uploading and downloading files and could also ensure the safe sharing and maintenance of the big data files. It also provides user interface and REST interface for storing the data. Our application is load balancing and fast streaming too. It is highly reliable and scalable for all structured and unstructured large volume of files.

## REFERENCES

[1]Sivaraman, E.; Manickachezian, R., "High Performance and Fault Tolerant Distributed File System for Big Data Storage and Processing Using Hadoop," International Conference on Intelligent Computing Applications (ICICA), 2014, vol., no., pp.32,36, 6-7 March **2014**.

[2]Krishna, T.L.S.R.; Ragunathan, T.; Battula, S.K., "Performance Evaluation of Read and Write Operations in Hadoop Distributed File System," Parallel Architectures, Algorithms and Programming (PAAP), 2014 Sixth International Symposium on , vol., no., pp.110,113, 13-15 July **2014**

[3]Shvachko, K.; Hairong Kuang; Radia, S.; Chansler, R., "The Hadoop Distributed File System," in Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on , vol., no., pp.1-10, 3-7 May **2010**

[4]K.V. Shvachko, "HDFS Scalability: The limits to growth," login:. April **2010**, pp. 6-16.

[5]Manikandan, S.G.; Ravi, S., "Big Data Analysis Using Apache Hadoop," in IT Convergence and Security (ICITCS), 2014 International Conference on , vol., no., pp.1-4, 28-30 Oct. **2014**

[6]Devakunchari, R., "Handling big data with Hadoop toolkit," in Information Communication and Embedded Systems (ICICES), 2014 International Conference on , vol., no., pp.1-5, 27-28 Feb. **2014**