Hadoop in Secure Mobile Cloud for Bigdata Storage and Processing

[1]Suvarna Vibhute, [2] Dr.Shridevi Soma

[1]Dept of CSE, PDA College of Engineering Kalaburgi, India

[2]Dept of CSE, PDA College of Engineering Kalaburgi, India

Abstract: Nowadays Big data storage and processing in the mobile devices are one of the more challenging processes. This paper addresses these challenges for both big data storage and processing in the mobile cloud by using K-out-of-n computing. By using Shamir secret sharing scheme can retrieve the whole uploaded file if it has at least K chunks out of n chunks of the distributed file. This process is different from the traditional way of downloading files. Hadoop on a mobile cloud enables the devices to run data-intensive computing applications. These applications have energy efficiency and reliability constraints (e.g., caused by unexpected device failures or topology changes in a dynamic network).As mobile devices are more susceptible to unauthorized access when compared to traditional servers. Security is also a consideration in this paper as it is a concern for sensitive data. The Experimental results show that the system is capable for big data analytics of unstructured data like media files, text and sensor data.

IndexTerms - Hadoop mapreduce, cloud computing, mobile cloud, Energy efficient computing, Fault-Tolerant computing.

I. Introduction

With advancement in recent technology, gradually mobile devices are replacing traditional personal computers. These devices are high-end computing hardware and complex software applications that generate huge amounts of data on the order of hundreds of megabytes. This data can vary from application raw data to images, audio, video or text files. With the fast hike in the number of mobile devices, big data processing on mobile devices has become a key emerging necessity for providing capabilities similar to those provid by traditional servers. For processing large datasets, there are numerous cloud services that provides computing infrastructure to end users. Hadoop MapReduce is one of the popular open source programming frameworks deployed in the cloud for cloud computing applications. The framework splits the user job into smaller tasks and runs these tasks in parallel on different nodes, thus decreasing the overall execution time when compared with a sequential execution on a single node. In the case of military or disaster response operations this architecture, however, fails in the absence of external network connectivity. This architecture is also not used in emergency response scenarios where there is limited connectivity to cloud, which leads to expensive data upload and download operations. This proposed system architecture addresses these challenges by using Shamir secret sharing scheme which can retrieve the whole uploaded file if it has at least K chunks out of n chunks of the distributed file. This process is different from the traditional way of downloading files. In this proposed system Cauchy Read-Solomon Coding Algorithm is used for Load balancing and Fault Tolerance. Thus, Hadoop on mobile cloud used in high intensive computing applications. As mobile devices are more susceptible to unauthorized access when compared to traditional servers. Security is also considered in this paper as it is a concern for sensitive data. Previous security mechanisms tailored for static networks are not fully secure for dynamic networks. To address these challenges of energy efficiency, reliability and security of dynamic network topologies, the *k*-out-of-*n* computing framework was introduced.

The traditional mobile distributed file system was primarily targeted for military operations where front-line troops are provided with mobile devices. A collection of mobile devices form a mobile ad-hoc network where each node can enter or move out of the network freely. MDFS is built on a k-out-of-n framework which provides strong guarantees for data security and reliability. k-out-of-n enabled MDFS finds n storage nodes such that total expected transmission cost to k closest storage nodes is minimal. As shown in Fig 1, every file is encrypted using a secret key and partitioned into n1 file fragments using erasure encoding (Reed Solomon algorithm). Unlike conventional schemes, the secret key is not stored locally. The key is split into n2 fragments using Shamir’s secret key sharing algorithm. File creation is complete when all the key and file fragments are distributed across the cluster. For file retrieval, a node has to retrieve at least k1(<n1) file fragments k2(<n2) key fragments to reconstruct the original file MDFS architecture provides high security by ensuring that data cannot be decrypted unless an authorized user obtains k2 distinct key fragments. It also ensures resiliency by allowing the authorized users to reconstruct the data even after losing n1-k1 fragments of data.

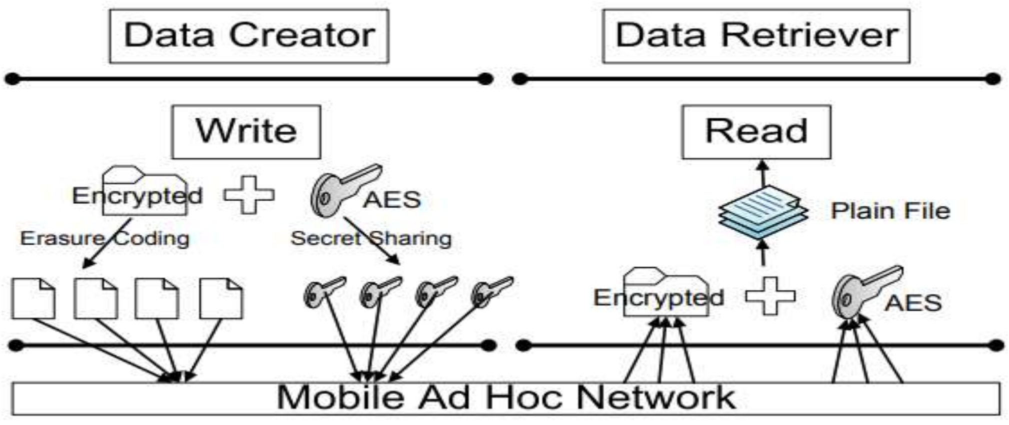


Fig 1: MDFS Architecture

II RELATED WORK & BACKGROUND

Previous work [1] k.shvachko et.al in the year 2010 Discuss The Hadoop Distributed File System (HDFS) is designed to store very large data sets reliably, By distributing storage and computation across many servers. Future plan is to use Zookeeper, Yahoo's distributed consensus technology To build an automated failover solution.[3] E. E. Marinelli et.al.in the year 2009 Discuss Hyrax model a platform derived from Hadoop that supports cloud computing on Android Smartphone's. It is generally difficult or expensive for one Smartphone to share data and computing resources with another. To tackle this problem proposed model Hyrax allows client applications to conveniently utilize data and execute computing jobs on network. Although the performance of Hyrax is poor for CPU-bound tasks. [7] C. Chen et.al. in the year 2013. Discuss Challenges of reliability and energy efficiency which remains largely unaddressed. To solve these challenges proposes the first k-out-of-n framework that jointly addresses the energy-efficiency and fault-tolerance. It assigns data fragments to nodes such that other nodes retrieve data reliably with minimal energy consumption. [4] H. Yang et.al. in the year 2011 Discuss To meet military mission provide securely store and exchange sensitive information on a network of mobile devices with resiliency. Using Concepts of Shamir's threshold based secret sharing algorithm, and symmetric AES cryptography. Thus, MDFS could be deployed as a viable distributed file system. [5]C. A. Chen, et.al, Discuss Resource allocation for energy efficient k-out-of-n system in mobile ad hoc networks. The k-out-of-n resource allocation problem in MANETs. Specifically, propose a resource allocation scheme that finds n service centers in a network with dynamically changing topology, such that the energy consumption for nodes to access k out of the n service centers is minimized. [6]S. Huchton, G. Xie, et.al. in the year 2011 discuss feasibility and real-world performance of MDFS by developing a working prototype. With several simplifying assumptions, MDFS prototype and find performance more than adequate for the majority of practical military missions and tasks. Advantages are increased performance and MDFS is a viable option for secure distribution of file fragments. And some Disadvantages of MDFS is less efficient in processing small files. [11] S. Ghemawat et.al. in the year 2003 Discuss The Google File System demonstrates the qualities essential for supporting large-scale data processing workloads on commodity hardware. Started by re examining traditional file system assumptions in light of current and anticipated application workloads and technological environment. [18]Karthik Kumar et.al in the year 2010 analysis suggests that cloud computing can potentially save energy for mobile users. However, not all applications are energy efficient when migrated to the cloud. Mobile cloud computing services would be significantly different from cloud services for desktops because they must offer energy savings. The services should consider the energy overhead for privacy, security, reliability, and data communication before offloading.

III. SYSTEM DESIGN

The architecture is composed of the main controller, decision maker and load checker where uploaded file splits into the number of chunks in the data splitter and the chunks distributed among the data nodes. Load checker checks the load on each server. Through the main controller, a user can download the file from the least loaded server. As shown in Fig 2, the proposed methodology involves the following stages as such as load balancing, data upload, data download.

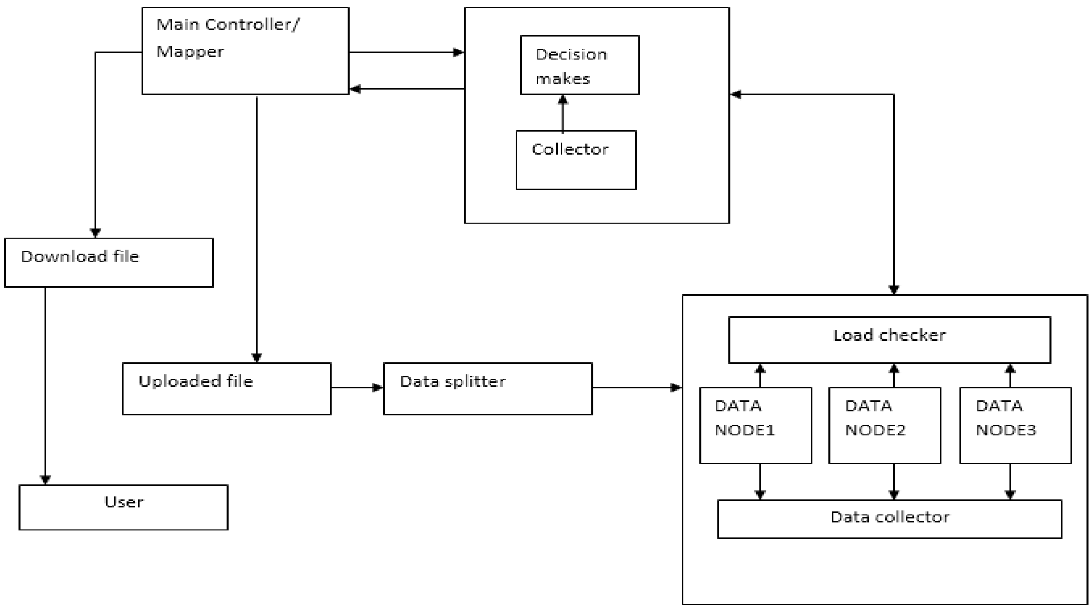


Fig: 2 System Architecture Of The Proposed System

3.1 Load Balancing

Proposed architecture consists of a the main controller server which can monitor the overall activities. There are three shared servers that are connected with the main controller. All the servers are connected with the main controller by HTTP protocol. Each node checks the overall CPU load independently and updates the load to the main controller periodically. The Main controller monitors the CPU load of its nodes and based on that node task has been assigned to the node. By using a proper load balanced algorithm that checks the load on the servers and finds the server with minimal load. Before assigning any task to each server, Load Balancer checks the load and server with the minimal load will be selected. Admin can check and monitors the load of ach server from the remote place in the browsers. In this proposed system Cauchy Read-Solomon Coding Algorithm is used for load balancing and Fault Tolerance.

3.2 Data Upload

Admin can upload resources to the server. While uploading the file to the server the file has been fragmented into multiple chunks and the chunks distributed among the data nodes as shown in the Fig 2.

3.3 Data Download

In proposed system file splitting algorithm is different from the traditional schemes. To split the files into chunks in this work Shamir secret sharing scheme is using. In this scheme, the file is split it into chunks that are different from the traditional way. In a traditional way to download the file or retrieve it all the chunks should be collected and merge together but using Shamir secret sharing scheme can retrieve the whole file if it has at least K chunks out of n chunks.

IV CAUCHY REED-SOLOMON CODING ALGORITHM

Reed Solomon codes are a particular case of non-binary BCH codes. Their capacity to correct burst errors stems from the fact that they are word oriented rather than bit-oriented. A bit-oriented code such as a BCH code would treat this situation as many independent single-bit errors. To a Reed Solomon code, however a single error means any or all incorrect bits within a single word. Therefore the RS (Reed Solomon) codes are designed to combat burst errors in a channel. In fact RS codes are a particular case of non-binary BCH codes. The structure of a Reed Solomon code is specified by the following two parameters: The length of the code-word m in bits, often chosen to be 8.The number of errors to correct T. A code-word for this code then takes the form of a block of m bit words. The number of words in the block is N, which is always equal to $N = 2^m - 1$ words, of which 2T words are parity or check words. The number of data bytes is usually referred to by the symbol K. Thus the RS code is usually described by a compact (N, K, T) notation. When the number of data bytes to be protected is not close to the block length of N defined by $N = 2^m - 1$ words a technique called shortening is used to change the block length. A shortened RS code is one in which both the encoder and decoder agree not to use part of the allowable code space. An error correcting code, such as an RS code, is said to be systematic if the user data to be encoded appears verbatim in the encoded code word. Thus, a systematic (204,188,8) code would have the 188 data bytes provided by the user appearing verbatim in the encoded code word, appended by the 16 parity words of the encoder to form one block of 204 words. The choice of using a systematic code is merely from the point of simplicity as it lets the decoder recover the data bytes and strip off the parity bytes easily, because of the structure of the systematic code.

A programmable implementation of a RS encoder and decoder is an attractive solution as it offers the system designer the unique flexibility to trade-off the data bandwidth and the error correcting capability that is desired based on the condition of the channel. This can be done by providing the user the capability to vary the data bandwidth or the error correcting capability (T) that is required. The C6400 DSP offers a unique and rich instruction set that allows for the development of a high performance Reed Solomon decoder by minimizing the development time required without compromising on the flexibility that is desired. This application note will discuss in the following sections how to develop an efficient implementation of a complete (204, 188, 8) RS decoder solution on the C6400 DSP. Reed-Solomon (RS) codes are based on a finite field, often called Galois field. When encoding data using RS codes, to implement a Galois filed arithmetic operation (addition or multiplication) requires many computations, so the performance is often unsatisfactory. CRS codes modify RS codes and give two improvements. First, CRS codes use a Cauchy matrix instead of a Vander monde matrix. Second, CRS codes convert Galois field multiplications into XOR operations. The key to CRS codes is construction of Cauchy matrices, and can achieve that in the following way.

Step 1: Given a input redundancy configuration (*k, m, w*)
Where $k+m \leq 2^w$, let $X = \{x1\dots xm\}$, $Y = \{y1\dots yk\}$, and $X \cap Y = \phi$ *xi* and *yj* is a distinct element of $GF(2^w)$.

Step 2: Calculate the Cauchy matrix in element (*i ; j*)
using 1= (*xi* + *yj*) (the addition and division are defined Over Galois field).

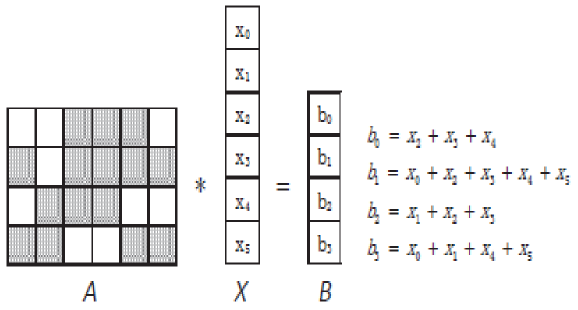
Step 3: The elements of GF (2^w) are the integers from zero to $2^w - 1$,
each element *e* can be represented by a *w*-bit column vector, *V* (*e*),using the primitive polynomial
over Galois Field.

Step 4: Each element *e* of GF(2^w) can be converted to $(w\times w)$ binary matrix, *M*(*e*), whose *i*-th($i = 1; : : : ;w$)
Column is equal to the column vector *V* (*e*2*i*−1).

Step 5: According to the value of *w*, Transform the Cauchy matrix into a $(mw \times kw)$
binary matrix, denoted as *A*.

Step 6: Divide every data block *X* and erasure codes block *B* into *w* trips.

In this way, when there exists "1" in every row of *A*, XOR operations on the corresponding data in *X*, to obtain the elements of *B* as shown in Fig 3. The erasure codes require 11 XOR operations.



$$b_0 = x_2 + x_3 + x_4$$
$$b_1 = x_0 + x_2 + x_3 + x_4 + x_5$$
$$b_2 = x_1 + x_2 + x_3$$
$$b_3 = x_0 + x_1 + x_4 + x_5$$

Fig: 3 Erasure Coding Using Cauchy Reed-Solomon Codes

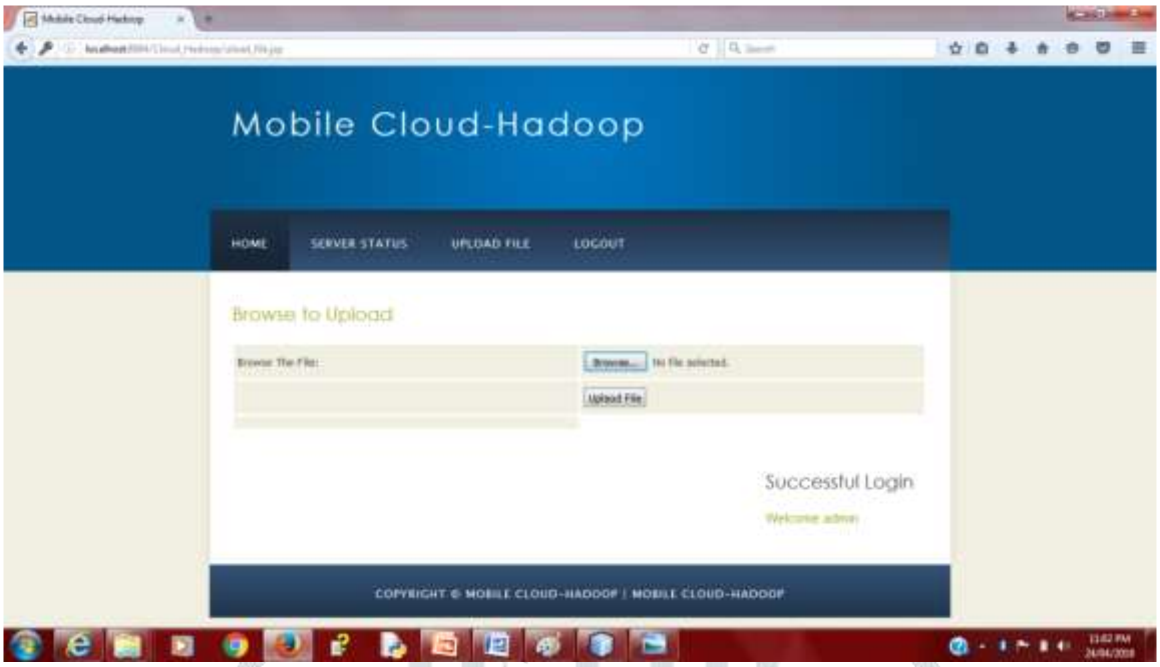
In a storage system using erasure codes, data are encoded to obtain data redundancy when data are written. Therefore, to improve the overall performance of a system, should reduce the cost of erasure coding, i.e., the number of XOR operations. There are two encoding strategies to achieve this goal. Encoding data using Cauchy matrices directly. The above Given Fig 3 shows, the density of a Cauchy matrix decides the number of XOR operations and the encoding performance. Encoding data using schedule. The sequence of XOR operations in the schedule decides the encoding performance.

IV SYSTEM IMPLEMENTATION

The proposed system is implemented in operating system Windows 7, VM player 8 and configured Hadoop in the Linux operating system(centos), Java 1.7, Hadoop 0.8.1, Netbeans IDE and Navicat lite for MYSQL database collection environment are used. MDFS framework consists of Java code, exported as a single jar file. Deployment of Hadoop in Amazon web service cloud by creating a WAR file of Hadoop.

V EXPERIMENTAL RESULTS AND DISCUSSION

This section deals with the results at various stages of experimentation of the proposed system.



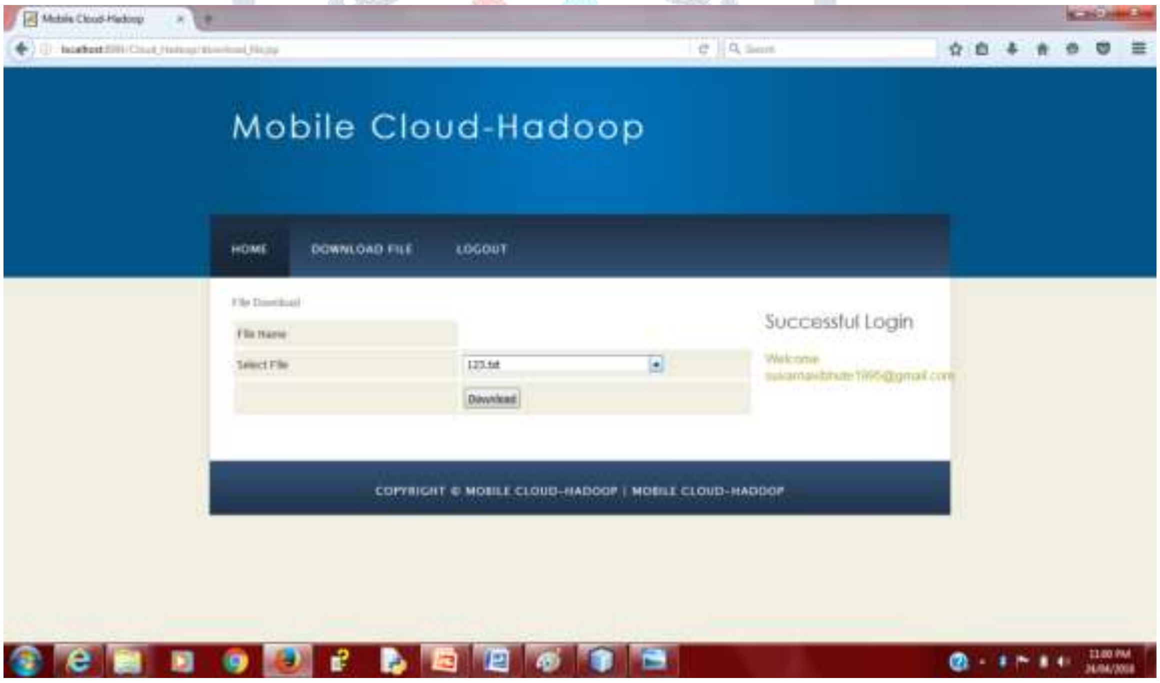
Browse The File: No file selected.

Successful Login

Welcome admin

Fig: 4 Data Upload

Admin can upload resources to the server by browsing file as shown in the Fig 4.While uploading the file to the server each file has been fragmented into multiple chunks and the chunks distributed among the data nodes**.**



File Download

File Name

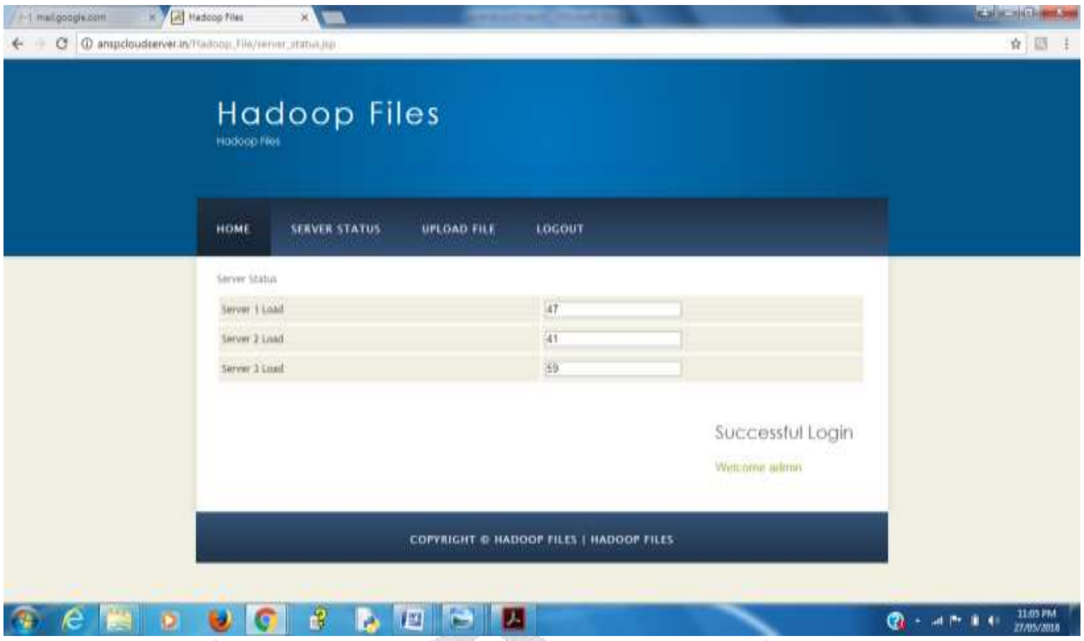
Select File 123.txt

Successful Login

Welcome

Fig: 5 Data Download

The User can download the files by selecting option given as shown in the Fig 5. To split the files into chunks Shamir secret sharing scheme is using. By this scheme, the file or split it into chunks that are different from the traditional way. In a traditional way to download the file or retrieve it all the chunks should be collected and merge together but using Shamir secret sharing scheme can able to retrieve the whole file at least K chunks out of n chunks.



Hadoop Files

Hadoop Files

HOME SERVER STATUS UPLOAD FILE LOGOUT

Server Status

Server 1 Load	47
Server 2 Load	41
Server 3 Load	59

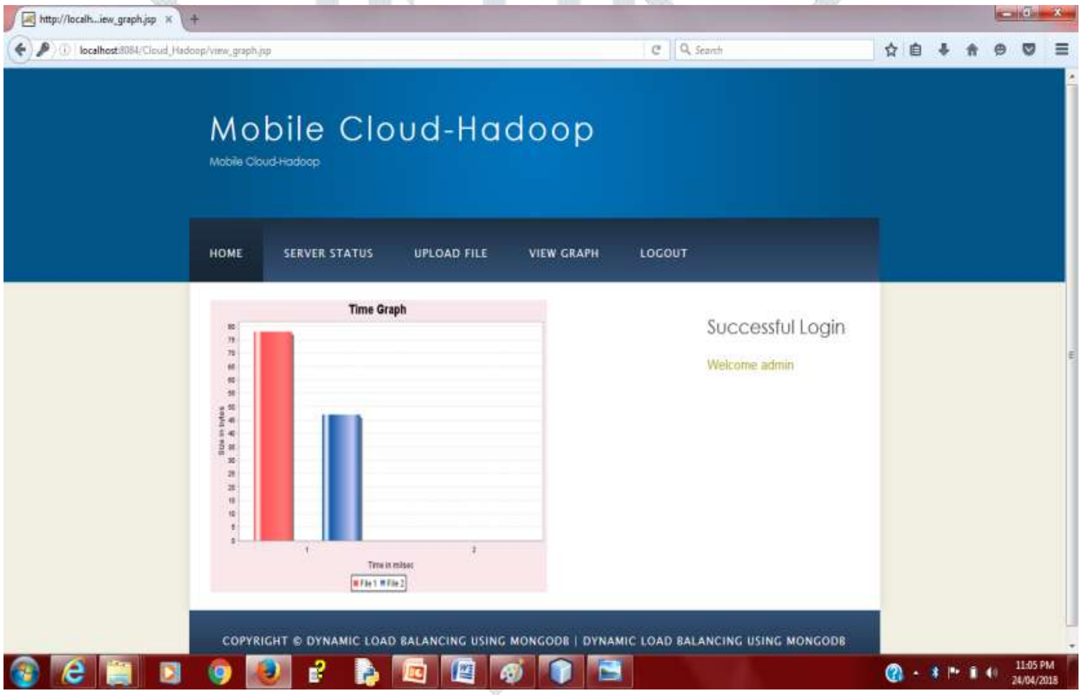
Successful Login

Welcome admin

COPYRIGHT © HADOOP FILES | HADOOP FILES

Fig: 6 Server Load Balancing

Load Balancer checks the load on each server, and server with the minimal load will be selected. Admin can check and monitors a load of each given three servers from a remote place in the browsers. As shown in the Fig 6.



Mobile Cloud-Hadoop

Mobile Cloud-Hadoop

HOME SERVER STATUS UPLOAD FILE VIEW GRAPH LOGOUT

Successful Login

Welcome admin

COPYRIGHT © DYNAMIC LOAD BALANCING USING MONGODB | DYNAMIC LOAD BALANCING USING MONGODB

Fig: 7 Transmissions of Bytes In Time Graph

Admin can view the time graph where File1 and File2 of size measured in bytes and time in milliseconds for file transmission as shown in the Fig 7.

VI PERFORMANCE EVALUATION DISCUSSION

Processing and transmission of input dataset size in MB is measured over network time and I/O time in the sec as shown in the graph of Hadoop job time Vs size of input dataset as shown in the Fig 8(a). The job completion time for different file size ranging from 1MB to 6MB shown in the Fig 8(b). However, MDFS does not have any hard limit on input dataset size(MB). And it can take any input size allowed in the standard Hadoop.

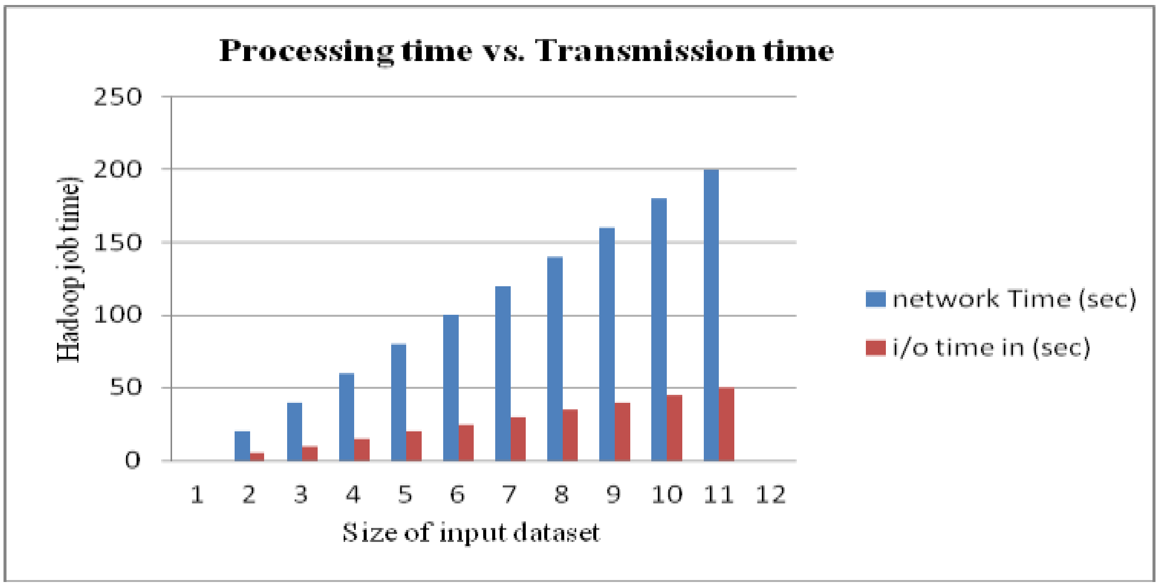


Fig: 8 (a) Processing time vs. Transmission Time

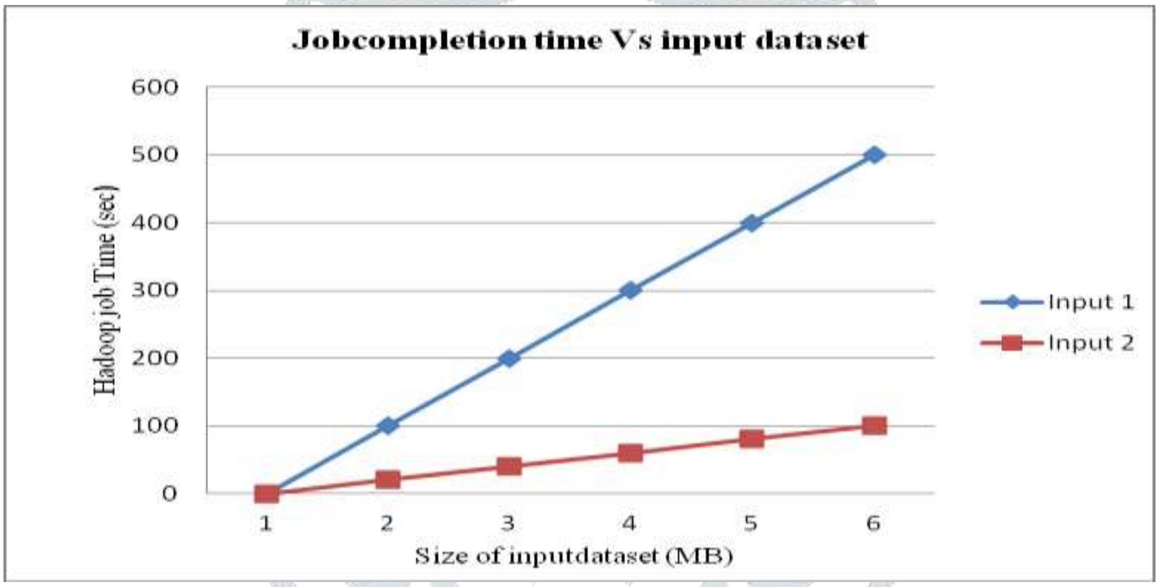


Fig: 8(b) Job Completion Time vs. Input Dataset Size

VII CONCLUSION AND FUTURE SCOPE

In This Work, The Hadoop MapReduce framework over MDFS is used in the system for big data in the mobile environment. In a traditional way to download the file or retrieve it, all the chunks should be collected and merge together. In this system by using Shamir secret sharing scheme whole file can be retrieved if it has at least K chunks out of n chunks of the distributed file by k-out-of-n computing. The System addresses all the constraints of data processing in mobile cloud energy efficiency, data reliability and security. The Experimental results show that system is capable of data analytics of unstructured big data such as media files, text and sensor data. The Future plan is to integrate the both energy-efficiency and fault-tolerant for *k*-out-of-*n* processing framework in Mobile Hadoop to provide better energy-efficiency and fault-tolerance in a dynamic network environment.

REFERENCES

[1] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, “The hadoop distributed file system,” in Proc. of MSST, 2010 978-1-4244-7153-9/10/$26.00 ©2010 IEEE.pp.1-9.

[2] Huchton, Scott “Secure mobile distributed file system (MDFS)”, Monterey, California. Naval Postgraduate School,2011 http://hdl.handle.net/10945/5758

[3] E. E. Marinelli, “Hyrax: Cloud computing on mobile devices using mapreduce,” Master’s thesis, School of Computer Science Carnegie Mellon University, 2009.pp.1-107

[4] H. Yang, H. Luo, F. Ye, S. Lu, and L. Zhang, “Security in mobile ad hoc networks: challenges and solutions,” WirelessCommunications, IEEE, 2004.pp.38-47

[5] C. A. Chen, M. Won, R. Stoleru, and G. Xie, “Resource allocation for energy efficient k-out-of-n system in mobile ad hoc networks,” in Proc. ICCCN, 2013.pp.1-9.

[6] S. Huchton, G. Xie, and R. Beverly, “Building and evaluatinga k-resilient mobile distributed file system resistant to devicecompromise,” in Proc. MILCOM, 2011. http://hdl.handle.net/10945/34802.pp.1-6.

[7] C. Chen, M. Won, R. Stoleru, and G. Xie, “Energy-efficient fault-tolerant data storage and processing in dynamic net- work,” in MobiHoc , 2013. IEEE transactions on cloud computing, vol. 3, no. 1, january-march 2015.pp.28-41

[8] Shabin D,G Jaspha w.karthrine “A Comprehensive Overview on secure Offloading in Mobile Cloud Computing” in Proc.Karunya University,2017 978-1-5090-3355-3/17/$31.0©2017 IEEE.pp.121-124.

[9] G. Huerta-Canepa and D. Lee, “A virtual cloud computing provider for mobile devices,” in Proc. of MobiSys , 2010. 978-1-5090-2047-8/16/$31.00©2016 IEEE.pp.95-98

[10] T. Kakantousis, I. Boutsis, V. Kalogeraki, D. Gunopulos, G. Gasparis, and A. Dou, “Misco: A system for data analysis applications on networks of smartphones using mapreduce,” in Proc. Mobile Data Management, 2012.978-0-7695-4713-8/12$26.00©2012 IEEE.pp.356-359.

[11] S. Ghemawat, H. Gobioff, and S.-T. Leung, “The google file system,” SIGOPS Oper. Syst. Rev., 2003. *SOSP’03,* October 19–22, 2003, Bolton Landing, New York, USA.2003 ACM 1-58113-757-5/03/0010.pp.1-15.

[12] S. A. Weil, S. A. Brandt, E. L. Miller, D. D. E. Long,and C. Maltzahn, “Ceph: A scalable, high-performance dis-tributed file system,” inProc. of OSDI, 2006.pp.1-13.

[13] F. Marozzo, D. Talia, and P. Trunfio, “P2p-mapreduce: Parallel data processing in dynamic cloud environments,”J.Comput. Syst. Sci. 2012.pp.1-27.

[14] P. H. Carns, W. B. Ligon, III, R. B. Ross, and R. Thakur,“Pvfs: A parallel file system for linux clusters,” in Proc. Of Annual Linux Showcase & Conference 2000. pp.1-11.

[15] Ciprian Barbieru,Florin pop “Soft Hadoop Scheduler for Big Data Processing in smartCities”in.

[16] Proc.University Politechnica of Bucharest,Romania, 2016 IEEE 30th International Conference on Advanced InformationNetworking and Applications.1550-445X/16$31.00©2016 IEEE.pp.863-870.

[17] “Apache hadoop,” http://hadoop.apache.org/

[18] K. Kumar and Y.-H. Lu, “Cloud computing for mobile users:Can offloading computation save energy?” *Computer*, 2010. 0018-9162/10/$26.00 © 2010 IEEE.pp.1-20.

[19] P. Elespuru, S. Shakya, and S. Mishra, “Mapreduce system over heterogeneous mobile devices,” in *Software Technologies for Embedded and Ubiquitous Systems*, 2009., 2009.cIFIP International Federation for Information Processing 2009. pp. 168–179.

[20] S Koley, S Nandy” Big Data Architecture with Mobile Cloud in CDroid Operating System for Storing Huge Data”in coputer science,2016 978-1-5090-1338-8/16/$31.00 ©2016 IEEE.pp.12-17.

[21] “Lustre file system,” http://www.lustre.org.