ISSN No. 0976-5697

Volume 5, No. 4, April 2014 (Special Issue)

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Supplanting HDFS with BSFS

[1]D.Santhoshi
Information Technology
ACE engineering college, Ghatkeser,India
santhudevasani@gmail.com

[2]T.Tejaswini Singh
Information Technology
ACE engineering college, Ghatkeser,India
thakur.teja@gmail.com

[3] V.Teja
Information Technology
ACE engineering college, Ghatkeser,India
virupa.teja@gmail.com

[4]K.Shyam Prasad
Information Technology
ACE engineering college, Ghatkeser,india
shyamloves001@gmail.com

Abstract: Hadoop is a software framework based on the Map Reduce programming model. It depends on the Hadoop Distributed File System (HDFS) as its primary storage system. To improve the performance, the HDFS layer of Hadoop can be replaced with a new, concurrency-optimized data storage layer called BlobSeer file system. Hadoop File System(HDFS) is used to evaluate the performance and efficiency of large file systems. The aim of the project is to compare the performance of HDFS and BSFS. In the proposed system by using create, read, update, delete operations in the distributed environment, performance of HDFS and BSFS is being tested. To find out which file system gives best performance for large and small datasets and also throughput time of the file system.

Keywords: Hadoop, nodes, supplanting, Blobeer, read, write

I. INTRODUCTION

A. Context and motivation of the project:

More and more applications today generate and handle very large volumes of data on a regular basis. Hadoop is with one step ahead of others, as an open source solution. Exposes basic file system operations: create, read, write. Introduces support for concurrent append operations. Some only use it, some builds soft on top of it, some implements solutions for frameworks, they haven't kept their original significance. Several attempts were done to integrate Blobseer with Hadoop, but unfortunately, very poorly documented. Next logical step after integrating it, was to test it's performance, by comparing it with HDFS, on some specific Map-Reduce algorithms.

B. Contributions:

We contribute to this project by trying to find out how the performance is influenced by the file system's implementation and also to compare the results for HDFS and BSFS. Withthis in mind, we have deployed Blobseer over Hadoop.The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models its scheduler and some for its file system. Other file systems integrated with Hadoop, except for HDFS are Amazon S3, Cloud Store, FTP, read-only HTTP and HTTPS. The model is inspired by the 'map' and 'reduce' primitives These were commonly used in functional programming, but in the case of Reduce.There are many contributions on this project which is seen in related work(**II**).

II. RELATED WORK

There were many contributions for developing a reliable and efficient distributed data management system, mainly due to the needs put forward by large companies and enterprises to efficiently manage their data. The Google File system developed by the google people was one among the early approaches, which was designed to meet google data processing requirements. The architecture they followed was basically a master-slave architecture in which there is a master node and many slave nodes in a cluster, which is built using inexpensive commodity hardware. Apart from scalability, reliability and availabilityThere is a large team that worked for the Blobseer project [1]. They have run a complex deployment in order to see how their file system scales in a map-reduce framework. Experiments were done over Yahoo release of Hadoop v0.20. A set of benchmarks that write, read and append data to files through Hadoop's file system API were implemented and throughput was measured as more and more concurrent clients accessed the system. In order to compare with the Blobseer performance, the BSFS replaced HDFS and the same Hadoop map-reduce framework was used for testing. This was possible due to the Blobseer java interface.

Benchmarks:

(a). Huge distributed file is written by single process
(b). Different parts can be read by concurrent readers.
(c). Concurrent writers append data to huge file.
(d). File system can be directly accessed.
(e). Same access patterns in Map Reduce applications.

More tests were done with real-world Map-Reduce applications such as **random text writer** (many tasks, each writing large amount of data, with no interaction between tasks - concurrent, massively parallel writes), **distributed grep** (huge data input that needs to be processed in order to obtain output - counting the number of lines that matches an

expression - concurrent reads from a shared file) and **sort**(concurrent reads from the same file and concurrent writes to different files).The measured
metrics were: the throughput when a single client performs and then the throughput per client, as the number is gradually increased. So, for N concurrent clients, first deploy on HDFS.

III. ARCHITECHTURE

A. Hadoop :

Hadoop is one of the distributed file systems, developed in 1990's. Hadoop was built by Doug Cutting, which was motivated by GFS and has also master-slave architecture. In hadoop the data is stored in different racks of a server. The main features are the name node and the data node. The name node contains the information about the locations of the data nodes and the data node contains the actual data what the user requires. HDFS has a master/slave architecture. The master is the Name Node, that manages the file system namespace (maps blocks to Data Nodes) and regulates clients access to files. A file divides into one or more blocks and stored in data node. There will be only one name node in a cluster and architecture is developed.

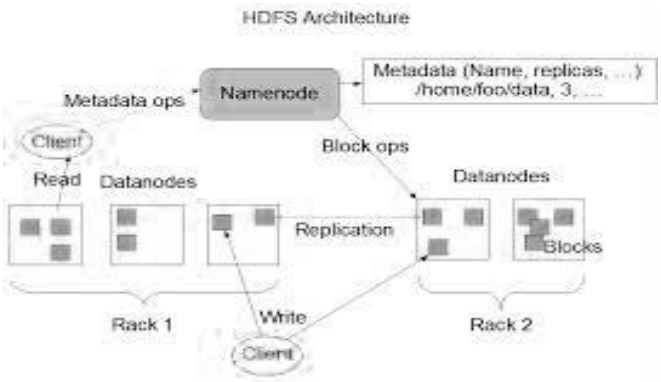


Figure 1. Hadoop Architechture

B. Blobseer:

For managing the highly unstructured data they are organized as Blobs, binary large objects. This architecture is concurrency optimized and versioning based approach, providing high data throughput and lock fee access to data. Blobseer represents a set of distributed communication processes. The distribution and interaction between entities are:

Client can create, read, write and append requests which are accessed by Blob's

Data providers stores and manages the pages generated by write and append requests. In the context of Hadoop Map Reduce , the nodes which act as data providers may also be the computational elements.

Provider manager maintains information about available storage spaces. Using load balancing algorithm the entity is selected and the providers will assign the data.

Meta data providers store the metadata which allows identification of blocks to make up a snapshot version. Distributed meta data management schema is used to raise concurrent access to the meta data.

Version manager assigns Blob version number to every request such that total serialization, ordering and atomicity is achieved.

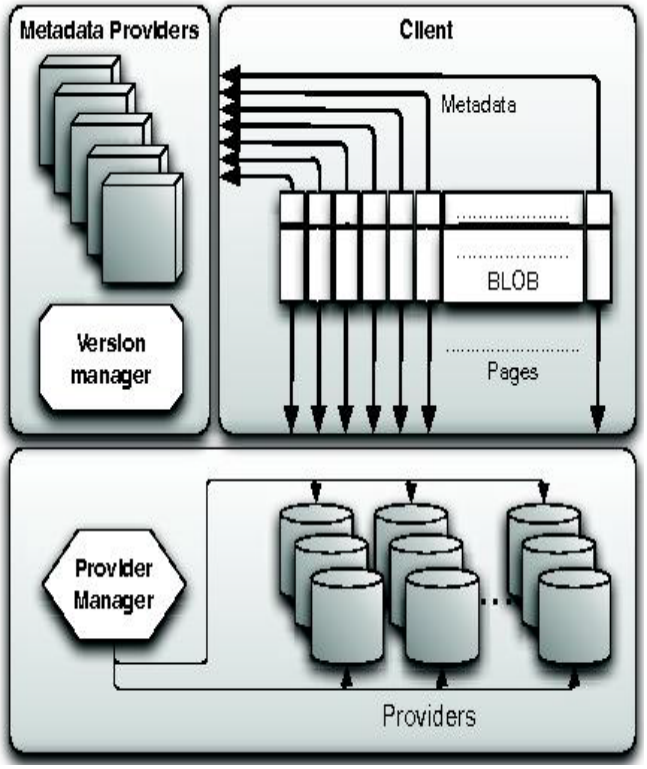


Figure 2. blobseer Architechture

IV. IMPLEMENTATION

When the architechture is being developed we need to implement those by using different technologies.

A. Used Techologies:

a. ***Apache hadoop[2]:*** Apache Hadoop is an open source software framework for storage and large scale processing of data sets on clusters. It is licensed under the Apache license. The design of an apache hadoop is used to scale up from single servers many machines and each machine can offer a large amount of storage and computational work. Hadoop has two key services.

(a). Reliable data storage using HDFS.
(b). High performance parallel data processing using MapReduce

b. ***Blobseer[3]:*** Core principles of Blobseer are: Organizing data as BLOBs (Binary Large OBjects) Treat data as a set of huge unstructured data, so that a BLOB can reach 1TB.This gives scalability since the data can grow as fast as needed without affecting the performance (the program should manage only the offset of the in the blob). This is effective because Blobseer supports an efficient fine-grain access to the BLOBs, for a large number of concurrent processes modify a file in Blobseer.

Metadata are represented by the range in the inner nodes which leaves identifying pages.Every node in the figure represents as a (key,value) where key includes the blob id,version manager and covered range bounded by an offset and size.value indicates the left and right cjild.Let us give an initial blob which is made of four pages,when a

write generates pages (2,3) and also a new metadata nodes which are represented by grey.New grey nodes are attached with the white nodes corresponding to the unmodified first and fourth page.concurrent writes and appends send data to the storage spaces for each other.

Synchronization will take place at version manager level where the updates being serialized and assigned version numbers in a increasing order.Metadata enable producing metadata in parallel too.

DataStriping is splitting blobs into chunks that are distributed on the nodes which provide storage space.the space from storage provider is configurable at this level and many objectives could be take place such as:

(a). Energy consumption is low
(b). Data localization is high

Chunks size is dynamically adjustable based on the system loading so that the computation is portioned and scheduling is optimized.

The disadvantage od versioning is that it needs lots of extra space but the space becomes more cheaper so here versioning has potential and it brings benefits for a low price such as

(a). Data access of different versions of the file to the clients
(b). Data attainment with data processing is overlapped asynchronization operations are provided so that read-write can be concurred on different versions
(c). Updates are kept as change between versions

Working of a Read

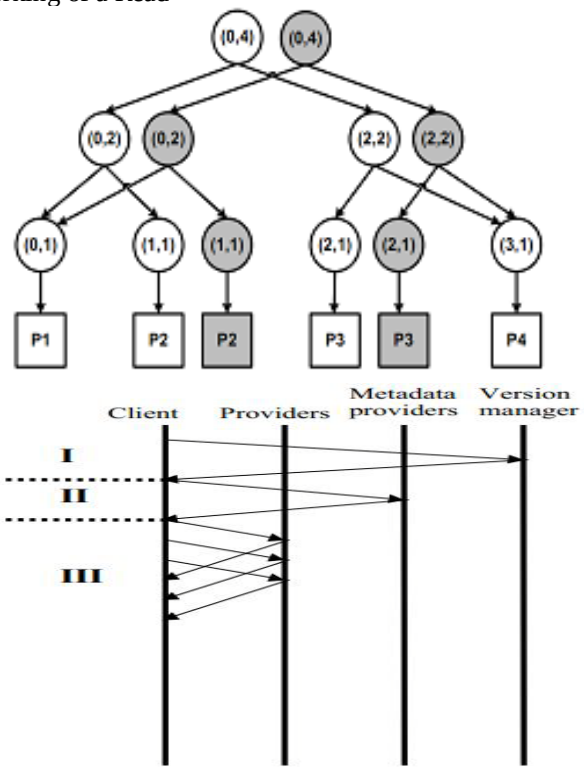


Figure 3. working of a read

a. Optionally ask the version manager for the latest published version.
b. Fetch the corresponding metadata from the meta data providers.
c. Contact providers in parallel and fetch the pages in the local buffer.

Working of a Write

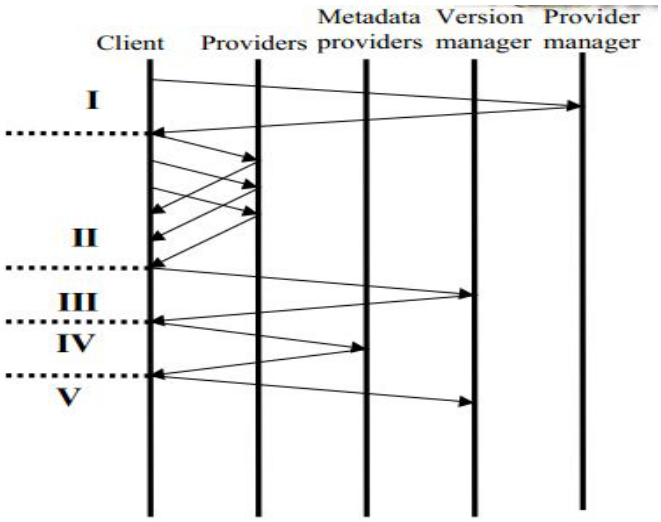
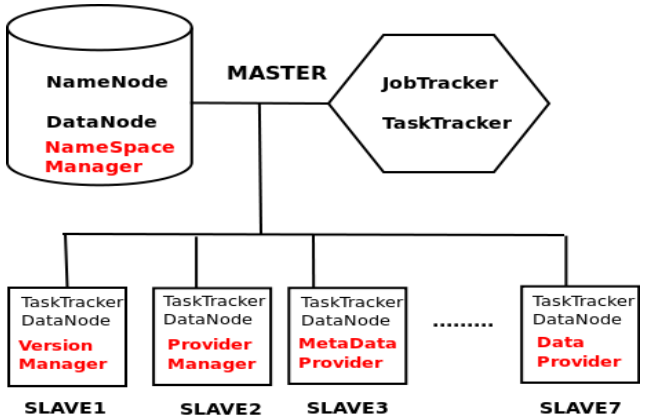


Figure 4. working of a write

(a). Get a list of providers that are able to store the pages, one for each page.
(b). Contact pages in parallel and write the pages to corresponding providers.
(c). Get a version number of the update.
(d). Add new metadata to consolidate the version.

B. Our Deployment Architecture evolution:

After implementationdeployment is necessary,for that the Significant effort was invested in preparing the experimental setup.we had to overcome nontrivial node management and configuration issues to reach this objective. Hadoop and Blobseer are available for both Windows and Linux. We have chosen Ubuntu 13.10 as operating system for the virtual machines. First we have deployed Hadoop, Blobseer then we have extended to 4 and 8 nodes[5]. The architecture evolved as it follows:



Deploy Hadoop, with name node and data node functionalities.Install Blobseer on local host .Components of Blobseer are:2 managers, metadata provider and data node. Add a fifth component to Blobseer, the Namespace Manager in order to replace the Hadoop file system. Integrate Blobseer in Hadoop. Multiply the previously configured virtual machine on the CS grid. From Hadoop's point of view there are:1 master name node and 3 slaves-data nodes (1 master and 7 slaves).

From Blobseer's point of view there are one node which keeps keeps the administrative part with manager and three nodes as data provider metadata provider role.

V. PROPOSED SYSTEM

By using these technologies which are discussed above the proposed system is used to test the performance and throughput when HDFS is replaced by BSFS[4].

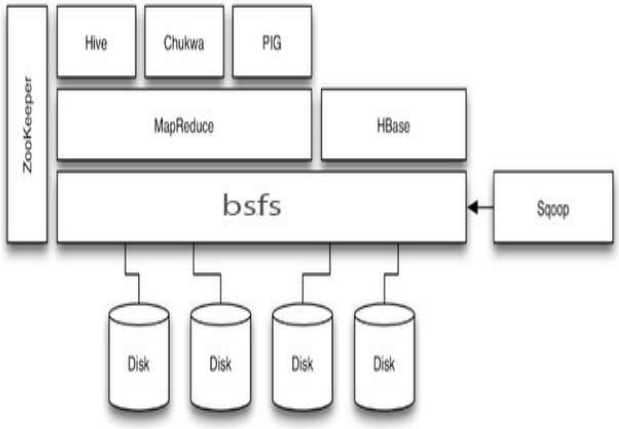
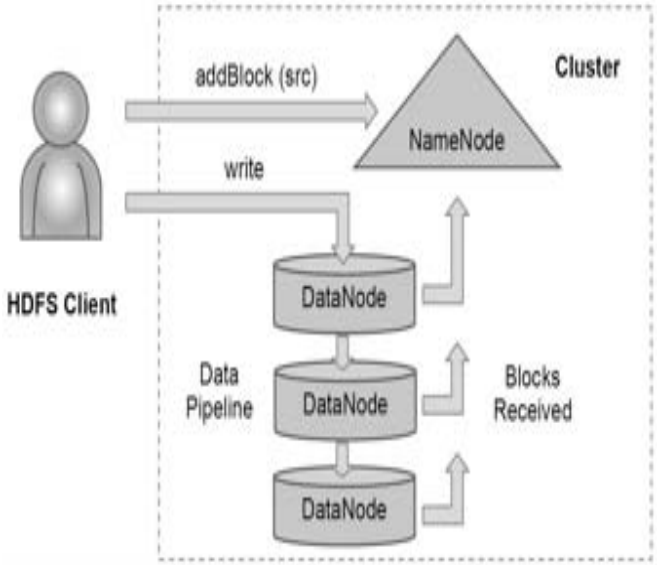


Figure 5. replaced BSFS with HDFS

Hadoop architechture will have a file system called hadoop file system.In the previous systems hadoop gives low performance for small files.In the proposed system the hadoop file system is being replaced by the blobseer file system.To test the performance of filesystem hadoop filesystem should be removed and then replace or substitute with a blobseer file system.Firstly,hadoop and bloobseer configuration must be setup.In proposed system more and more nodes are added so that the distributed environment will be expanded.

In first step hadoop is installed and configured.Then the blobseer file system is replaced by deploying on hadoop architechture.Once the replacing is done performance need to be test by using read,write and append files.In hadoop these programs are written in java but in blobseer file system having hadoop architechture in order know the time taken by running or compiling each file.Writing files to HDFS can be done through:



When writing data into hdfs at first

(a). Client consults namenode

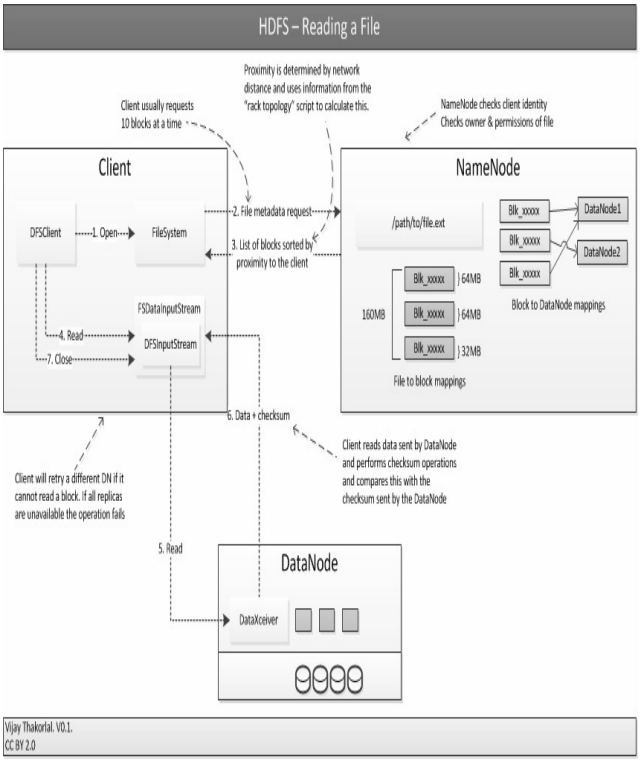
(b). Client writes block directly to one datanode

(c). Datanode replicates block

(d). Cycle repeats for next block

In hdfs writing a file is granted as a lease so that no other clinet can write.The client sends request to the name node for writing a file to hdfs.The lease time is bound by soft and hard limit.when the soft limit expires,the client fails to close the file or renew the lease then another client can also access the file.If the hard limit expires then hdfs assumes the client is quit and will automatically close the file.

Reading files from HDFS can be done through :



When reading data from HDFS at first,

(a). Client consults namenode

(b). Client reads data directly from datanode

(c). Cycle repeats for next block

Hdfs allows a client to read a file that is open for writing.If the read attempt failed then the client tries for the next replica.when the read operation is carried out the client fetches all the blocks and their locations from the namenode.

VI. CONCLUSION

This paper has given an account of the present working strategies and the drawbacks of the various distributed file system. This paper also gives the importance of the efficient distributed system to overcome the existing problems. The reasonable approach to outfit the issue is the "FineGrain access". This research has thrown up many questions in need to implement the "performance of both file systems" for better and effective performance. In present by using create, read, update, delete operations in the distributed environment, performance of HDFS and BSFS is being tested. To find out which file system gives best performance for large and small datasets . So to overcome this issue, "REPLACING WITH BSFS" is proposed. This helps to

overcome the drawback and decrease the chance of losing the data.

VII. REFERENCES

[1]. BlobSeer File System (BSFS) version 1.2.1 released: The patch for Hadoop-1.2.1 is modified so that BSFS is able to work as the interface between Hadoop-1.2.1 and BlobSeer-1.2.1 now

[2]. Hadoopsources http://hadoop.apache.org/docs/r1.2.0/releasenotes.html

[3]. Blobseersources https://github.com/acarpena/blobseer/

[4]. Blobseer integration with hadoop module. https://github.com/acarpena/blobseer/tree/4416a8c141cf03fc8

[5]. Hadoop one node deployment tutorial. https://ncitcluster. grid.pub.ro/trac/PP2010/wiki/Hadoop.

[6]. Russel Berg. The Sun Network File system: Design, Implementation and Experience, 1986.

[7]. D.Borthakur. The Hadoop Distributed File System: Architecture and Design. The Apache Software Foundation, 2007.