Volume 5, Issue 11, November 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Literature Survey of HDFS and MapReduce

Rujul Kpadia, Aastha Inani, Surabhi Singh, Priyan Agrawal, Amit Kelotra
Dept CE, SVKM`S NMIMS MPSTME Shirpur,
Maharashtra, India

Abstract— Hadoop Distributed File System (HDFS) is the primary component of the Hadoop framework responsible for storage, which is designed for maintaining and processing huge datasets efficiently among clusters. Files are uploaded from a local file system to the HDFS so as to cooperate with MapReduce which is the computation infrastructure of Hadoop. There have been various system architectures that have been implemented for extensively large data analysis and computing applications which include distributed relational database management systems. However most of the data that is frequently growing is the data in unstructured form and thus new processing methods with more flexible data models were required. Hadoop has provided several solutions including the MapReduce architecture which was pioneered by Google and is now available in an open-source implementation.

Keywords— Hadoop Distributed File System

I. INTRODUCTION

Hadoop is an extensive tool that provides a facility of distributed storage and distributed computing using the Hadoop file system[1] and the map reduce protocol[2]. Hadoop has a property of shredding the files that are to be stored in Hadoop file system. Besides, the storage part the map reduce protocol distributes the task to be performed. Thus, Hadoop supports storage and processing of big data very easily.

Hadoop was developed by google and yahoo and currently is an apache product. Besides, the facility of storage and distribution Hadoop also provides features like HBase, Pig, Hive, Sqoopand Flume, performance.

A. HDFS (Hadoop File System)

The Hadoop file system is one of the core part of hadoop which is responsible for storage of structured as well as unstructured data.HDFS stores metadata and application data separately. HDFS carries out its file storage responsibility through a master-slave system.

The master is typically the namenode which contains the metadata of the files stored in the Hadoop cluster and the slave is the datanode which is basically used to store data.

Using this master-slave system HDFS stores the data by primarily carrying out two steps. First, is to shred the given file that is to be uploaded into small parts. And the second is to allocate the slaves to which each respective part of the file is to be stored.

B. Map-Reduce

MapReduce was developed by Google for distributed processing of large data set on a cluster based system. MapReduce also works on master slave system. Here, the master is jobtracker which perform the task of distributing the given job into the slaves. The slaves are known as the tasktrackers which perform the tasks located to them.

The MapReduce[3] works in which users specify the map function that processes a key/value pair to generate a set of intermediate key/value pairs and a reduce function that merges all intermediate values associated with the same intermediate key.

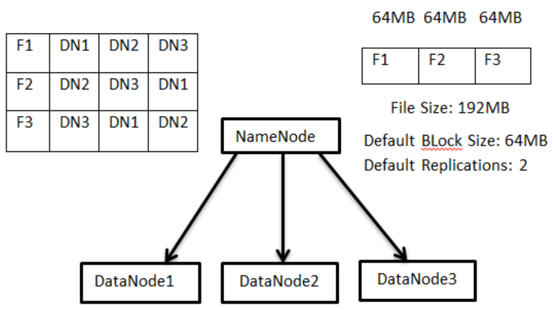
Map: The input is given to the master node which then divide the problem to sub problems and then distribute it to tasktrackers. The tasktracker may again distribute the task into further smaller tasks and allocate them to form a multilevel structure. The tasktracker process their respective allocated task and pass the answer to the master.

Reduce: The jobtracker collects the solution of the tasks performed by thetasktrackers and combines them to form the final solution.

II. ARCHITECTURE

A. NameNode

This is the core component of the Hadoop file system[4]. Infact this is the prime component of the Hadoop cluster since failure of this node result in disruption of the entire cluster. Name node is responsible of shredding the file into parts which are then allocated to respective slaves.



F1	DN1	DN2	DN3
F2	DN2	DN3	DN1
F3	DN3	DN1	DN2

Fig 1. Name Node

Namenode has the responsibility of maintaining the metadata as well as allocating the datanodes for each file .For example let us consider a file of size 192MB. We have default block size as 64MB and the default replication factor equal to 2. When the client sends this file to a namenode, the namenode shreds the file with respect to the block size which in our case is 64MB. Thus, the 192MB file is partitioned into 3 parts. The namenode then allocates a datanode for the first part. After this the namenode allocates the datanode where the replica of this part is to be stored. Once all the replicas of the first are stored in their allocated datanodes, the namenode then allocates the datanode in which the second part of the file is to be stored. This continues until the entire file and its replicas are stored in the Hadoop cluster.

B. Datanode

This is the slave in the namenode- datanode architecture of the Hadoop file system. It is responsible for storing files that are sent by the client.

During start up, datanode connects to the namenode and performs the handshake. The handshake is to verify the namespace ID and the software version of the datanode. If either of the two don't match up with that of the namenode, the datanode automatically shut down. The namespace ID Is a ID that is allocated to the cluster when it is formed. Thus each cluster has a unique namespace ID. Thus nodes with different namespace ID won't be able join a Hadoop cluster when they attempt to join the cluster.

The sole purpose of the datanode[5] is to store the partition of the file that is allocated to it by the namenode. Thus as and when the namenode designates a file partition to be stored the datanode, the datanode stores the files until its storage capacity is full.

Once a file partition has been stored on a particular datanode the namenode designates other datanodes on which the replicas of the respective part is stored. After storing the file the datanode sends an immediate report to the namenode.

C. HDFS Client

Various applications on the user front access the file system using the HDFS. The HDFS provides the following facilities to the client

- Read,Write,Delete file
- Create and delete directories

The client sends the request to the HDFS to perform any of the file operations. When a client wants to read a file from the Hadoop cluster, it requests the namenode[6] for the list of datanodes form the file is to be fetched. Similarly when a file is needed to be stored it requests the namenode to store the file which further designates the datanodes to which the respective file blocks and their replicas are to be stored.

Besides file operations, HDFS provides an API which enables applications like the MapReduce to schedule tasks on the data present in the datanodes. Further it also allows to change the replication factor as well as to change the block size.

D. Backup Namenode

Namenode is an essential part of the hadoop cluster, failure of which may result in the whole cluster being useless. Hence a backup of a namenode was required to prevent the loss of data from any system failures.

The backup namenode gets activated whenever the primary namenode deactivates due to system failure. All the metadata in the primary namenode is copied by the backup node which then acts as the namenode for the entire cluster.

E. Jobtracker

The mapreduce[7] consists of a jobtracker which is the master in master-slave architecture of the framework.

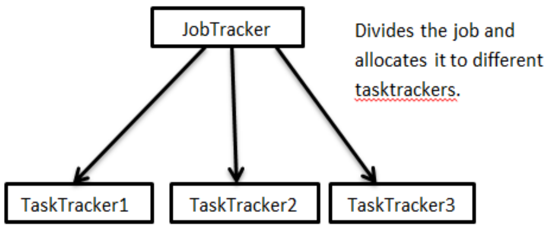


Fig 2. JobTracker

Whenever a client sends a request to perform a job on the hadoop cluster, the jobtracker splits the job into various sub tasks and then allocates the tasktracker on which each of these sub tasks are to be performed. Thus parallel computing environment is formed where the task is simultaneously being carried out on the various slaves.

The jobtracker contacts the namenode to read the file on which a job is to be performed. The namenode provides the list of the datanodes from which the tasktrackers can fetch their respective pieces of file on which they can perform the allocated task.

F. Tasktrackers

The tasktrackers form the slaves in the MapReduce framework of hadoop. They are responsible for computing on a given file with the given task.

When the jobtracker allocates tasktrackers to perform a task, the tasktrackers designate themselves as mapper. The mapper is responsible for performing the given job on a part of the file which it fetches from a datanode. After the mapper part of the job is completed, the mapper designate one or more reducers from the mappers. The no of reducer depends on the type of ask allocated to them. The reducer finally collects the tasks performed by the mappers and provides the combined result to the jobtracker which further provides it to the client.

G. Mapper

A MapReduce *job* usually splits the input data-set into independent chunks which are processed by the *map tasks* in a completely parallel manner.

When a client sends a request for a performing a job on a file, the jobtracker splits the file into further subparts. These parts are sent to the mapper which perform the jobs on a section of the file. This job is performed simultaneously in the parallel mapper systems. After performing job the mappers forward their output to the reducer.

H. Reducer

Once the mappers perform their respective task, the task of the reducers come into play. The mappers elect one or more reducers from them. The task of the reducer is to collect the individual tasks performed by each mapper designated for a job and then combine the individual tasks.

Thus reducer generates a file that contains the summed up output of the task to be performed. This file is then given to the user.

I. MapReduce Client

The MapReduce client requests for a job to be performed during which he/she provides the name as well as describes what job is to be performed.

This request is processed by the jobtracker which primarily divides the task into various subsections. The jobtracker then goes to the respective namenode of the cluster to get the list of the datanodes from which it can read the file. The namenode provides the datanodes in which the file is stored. The jobtracker allocates the tasktracker which are to perform the job. The tasktrackers go to the respective datanodes in which the file is present and then perform the task on each sub parts of the file. The task done by each tasktracker is combined into a file which is then finally sent to client as output.

II. FILE I/O OPERATION IN HDFS

The primary file operations carried by client in a hadoop file system is to either read a file or write a file.

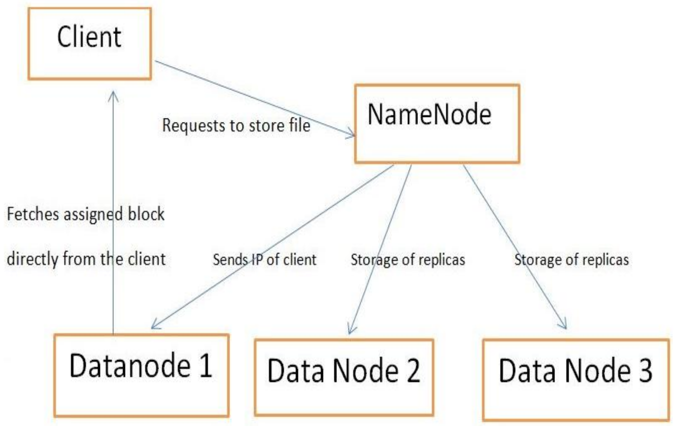


Fig 3. File Operation

When the client requests to write a new file in the hadoop file system, he/she sends a request to the namenode. The namenode considers the client request and shreds the incoming file into various parts. Now for each part the namenode assigns a respective datanode that will store the part of file into it. The default block size is 64MB and the default replication factor is 2.

After allocating each file part to a datanode, the namenode sends the location (IP address) of the client to the datanode. When the first part of the file is getting uploaded on datanode, it stores a part of the file in it. After this first part of file has been stored the namenode allocates the datanodes to which the replicas of the part of the file is to be stored.

After the first part of the file is stored, the datanodes that have been assigned to make replicas are given the client IP from the namenode. These datanodes then eventually fetch the part of file from the client and thus store the replicas of the current part of file.

Once all the replications of a file part are made then and only then do the fetching of next part starts.

Similarly when a client wants to read a file from the hadoop cluster, he/she will again send the request to the namenode. The namenode will consider the request and provide the location of the datanodes (where the file is stored) to the client. The client then directly goes to the datanode and accesses the file. This method is more efficient rather than namenode fetching the whole file and giving it to client which would have been time consuming thus reducing the efficiency of hadoop.

III. WORKING OF MAPREDUCE

The MapReduce framework provided by hadoop is used for the distributed computing. Itperforms various operations on files by entering the hadoop file system using API.

Once the file has been uploaded on the hadoop file system, various operations are performed on it. This can vary from a simple word count operations to different complex analysis operations making hadoop more reliable.

When a client requests to perform job on a hadoop cluster, he/she provides with the name of the file on which job is to be performed along with the job which is to be performed. After the job has been allocated the master of the MapReduce i.e. Jobtracker requests the file (on which the job I to be performed) from the namenode. The namenode provides the IP of the datanodes from the file is to be fetched. It is to be noted that different blocks of files are stored in different datanodes. Now, the jobtracker allocates the task to different tasktrackers and also provides the IP of the datanodes from where the file is to be fetched and on which the job is to be performed. Thus, the tasktrackers fetch the file from the datanodes and perform the job. The tasktrackers then provide the reduced (summed up) work to the jobtracker which then finally provides the output of the job performed the client.

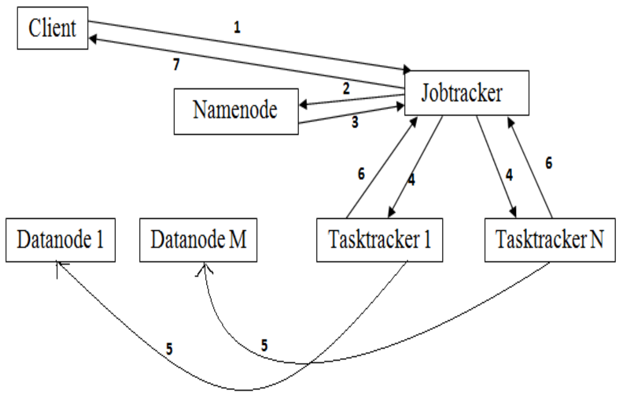


Fig 4. Working of MapReduce

1. Client requests to perform job and sends the file name and the job to be performed.
2. Jobtracker requests file from namenode.
3. Namenode provides IP address to the jobtracker.
4. Jobtracker distributes work to different tasktrackers and provides the IP address of the datanodes from which part of file is to be fetched and job is to be performed.
5. Tasktrackers fetch the file from datanodes and perform the job.
6. Tasktrackers provide the reduced (summed up) work to jobtracker.
7. Jobtracker provides the output to client.

V. FUTURE SCOPE

This section presents the future work that is considering by the hadoop team. If the namenode of the hadoop cluster gets down, it become difficult and takes time for the recovery. For automated failover recovery there is a backupNode that contains all the transaction of the primary node. So this leads to the automated recovery of failover.

Scalability of the namenode has been the major challenge. As the namenode contains all the metadata and its memory size is limited upto few files. If the memory space has completed the namenode become irresponsive for further data due to java garbage collection and requires a restart. So for this quotas are added to manage the usage and achieve tools.

Our solution for the scalability problem is to allow multiple namespaces to share a physical storage of cluster. We are extending our block ID’s to be prefixed by block pool identifier.

In this we are using job centric namespaces rather than cluster centric namespace as the cost of managing the multiple independent namespaces is more especially if the namespaces is large.

REFERENCES

[1] Shamil Humbetov, Data-Intensive Computing withMap-Reduce and Hadoop,

[2] B. Patel, Addressing Big Data Problem Using Hadoop and Map ReduceAditya, NUiCONE,2012

[3] Idris, M. Hussain, S. ; Sungyoung Lee , In-Map/In-Reduce: Concurrent Job Execution in MapReduce, IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications,2014

[4] I. Tomaši , J. Ugovšek, A. Rashkovska, R. TrobecJožef , Multicluster Hadoop Distributed File System.

[5] Youwei Wang J, Zput: a speedy data uploading approach for the Hadoop Distributed File System .

[6] Kaushik, R.T.**,** Bhandarkar, M. ; Nahrstedt, K., Evaluation and Analysis of GreenHDFS: A Self-Adaptive, Energy-Conserving Variant of the Hadoop Distributed File System

[7] Konstantin Shvachko, HairongKuang, Sanjay Radia, Robert Chansler, The Hadoop Distributed File System