Replica Management and High Availability in Hadoop Distributed File System (HDFS)

M. P. Vinod Kumar,
Associate Project
CTS,
Banglore.

P. S. Hema Latha
Assistant Professor,
Dept of MCA
KMMIPS,Tirupathi,

***Abstract*—Hadoop is a software framework that supports data intensive distributed application. Hadoop creates clusters of machine and coordinates the work among them. It include two major component, HDFS (Hadoop Distributed File System) and MapReduce. HDFS is designed to store large amount of data reliably and provide high availability of data to user application running at client. It creates multiple data blocks and store each of the block redundantly across the pool of servers to enable reliable, extreme rapid computation. MapReduce is software framework for the analyzing and transforming a very large data set in to desired output. This paper focus on how the replicas are managed in HDFS for providing high availability of data under extreme computational requirement. Later this paper focus on possible failure that will affect the Hadoop cluster and which are failover mechanism can be deployed for protecting the cluster.**

Keywords — Cluster, HDFS, MapReduce, Node Replica, High Availability, Distributed Computation.

I. INTRODUCTION

Dealing with "Big Data" requires – an in expensive, reliable storage and a new tool for analyzing structured and unstructured data. Apache Hadoop addresses both of these problems. Hadoop is the subproject of Lucene under the umbrella of Apache Software Foundation. Hadoop distribute and parallelize data processing across many nodes in a compute cluster, speeding up large computations and hiding I/O latency through increased concurrency. It is well suited for large data processing like searching and indexing in huge data set. Hadoop includes Hadoop Distributed File System (HDFS) and MapReduce. It is not possible for storing large amount of data on a single node, therefore Hadoop use a new file system called HDFS which split data into many smaller parts and distribute each part redundantly across multiple nodes. MapReduce is a software framework for the analysis and transformation of very large data sets. Hadoop uses MapReduce function for distributed computation. MapReduce programs are inherently parallel. Hadoop take advantage of data distribution by pushing the work involved in analysis to many different servers. Each server runs the analysis on its own block from the file. Results are combined in to single result after analyzing each piece. MapReduce framework takes care of scheduling tasks, monitoring them and re-executes the failed tasks .

This paper does a detailed study on Hadoop architecture and component working. Later focus on how data replicas are managed in Hadoop distributed file system for better performance and high data availability for highly parallel distributed Hadoop Applications. This paper also takes in account the different failure which will affect the Hadoop system and various failover mechanisms for handling those failures. HADOOP DISTRIBUTED FILE SYSTEM (HDFS) HDFS is a distributed file system designed to run on commodity hardware. HDFS is highly fault tolerant and is designed to be deployed on low cost hardware. HDFS is suitable for applications that have large dataset. HDFS maintain the metadata in a dedicated server called NameNode and the application data are kept in separated nodes called DataNode. These server nodes are fully connected and they communicate using TCP based.

Widespread success of machine Learning is in addressing an even broader range of tasks for actionable business insights and optimizations. But amount of data being produced today is so much that, 90% of the data in the world today has been created in the last two years alone. The Internet provides a resource for compiling enormous amounts of data, often beyond the capacity of individual disks, and too large for processing with a single CPU. Hadoop an open source framework developed by Dough Cutting , is built on top of the Hadoop Distributed File System (HDFS) and provides a parallelization framework which has garnered considerable acclaim for its ease-of-use, scalability, and fault-tolerance

1.Hadoop Apache Hadoop is a powerful open source software platform that addresses above mentioned problems of Bigdata. This is the platform used for cloud computing by some of the pioneers like Yahoo!, Amazon, Facebook, eBay, etc. Two major components of Hadoop are: 1. Hadoop Distributed File System (HDFS) for distributed storage

1. Mapreduce for parallel processing Thus, Hadoop offers a reliable and scalable environment for distributed computing, which involves many clusters that houses huge data upon which necessary computing need to be carried out.

1.2 HADOOP DISTRIBUTED FILE SYSTEM (HDFS)

HDFS is a distributed file system designed to run on commodity hardware. HDFS is highly fault tolerant and is designed to be deployed on low cost hardware. HDFS is suitable for applications that have large dataset. HDFS maintain the metadata in a dedicated server called NameNode and the application data are kept in separated nodes called

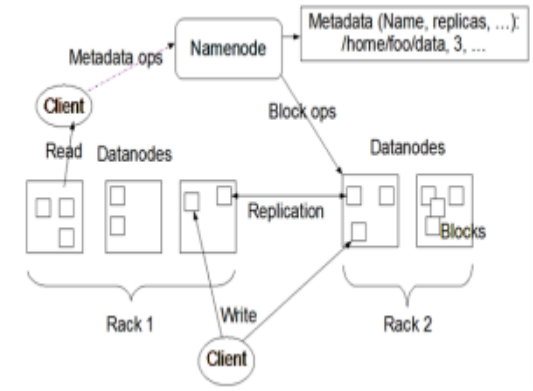
DataNode. These server nodes are fully connected and they communicate using TCP basedprotocols.

In HDFS the file content are replicated on multiple DataNodes for reliability. NameNodes HDFS name space is a hierarchy of files and directories. Files and directories are represented on the NameNode using inodes which record attributes like permissions modification and access time, namespace and disk space quotas. The file content is split into blocks (typically 128MB) each block of file is independently replicated at multiple DataNodes. NameNode maintains the mapping of file blocks to DataNodes. An HDFS client waiting to read a file first contact the NameNode for the locations of data blocks comprising the file and then reads block content from the DataNode closest to the client. When writing the data, the client requests the NameNodes to nominate a set of DataNodes to host the block replicas. The client then writes data to the DataNodes in pipeline fashion

DataNodes Each data block is represented by two files in the host native file system, one file contains the data itself and the other contains block‟s metadata. Handshake is performed between all DataNodes and the NameNode at startup. During handshake, the namespace ID and software version of DataNode is verified with the NameNode. If it does not match with that of NameNode, then that DataNode will automatically shuts down. Namespace ID is assigned to the file system instance when it is formatted. A newly initialized DataNode without any namespace ID can join the cluster and the will receive the cluster‟s namespace ID. Each DataNode persistently store its unique storage ID, which help to recognize it after restarting it with a different IP address or port. Each DataNode send block report to the NameNode to identify the block replicas in its possession. First block report is send during DataNode registration and the subsequent block reports are sent at every hour. This helps the NameNode to keep an up-to-date view of where block replicas are located on the cluster. Each DataNode send heartbeat to NameNode to confirm that it is operating and its block replicas are available. Default heartbeat interval is 3 seconds and if no heartbeat signal is received at NameNode in 10 minutes, the NameNode will mark the DataNode as unavailable. NameNode schedules creation of new replica of those blocks on another DataNode. NameNode use replies to the heartbeat to send instruction to DataNodes. The Instruction include commands to

- Replicate block to other nodes
- Remove local replicas
- Re-register or to shut down the node
- Send an immediate block report

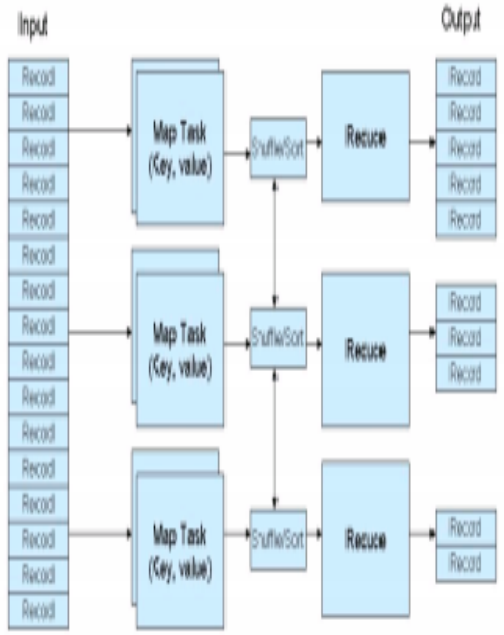
Figure 1. HDFS Architecture



2 .MAPREDUCE

MapReduce is a programming model for processing and generating large datasets. User specify a map function and a reduce function. The map function process a key/value pair to generate an intermediate key/value pair. The reduce function merges all intermediate values associated with same intermediate key. The programs written are inherently parallel and execute on a large cluster of commodity servers. The runtime system take care of all internal details like details of partitioning the input data, scheduling the program‟s execution, handling machine failure. The MapReduce library group together all intermediate values associated with same intermediate key and pass them to reduce function. Reduce function accept an intermediate key and a set of values for that key and merges together these

Figure 2. MapReduce Architecture



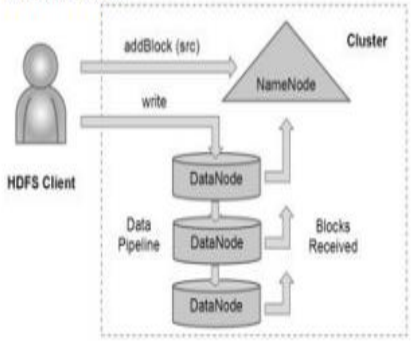
values to form a smaller set of values. Typically just zero or one output value is produced per Reduce invocation. The intermediate values are supplied to the user‟s reduce function via an iterator.

2. DATA DISTRIBUTION IN HDFS HDFS

support operation to read, write and delete file as well as to create and delete directories. For reading a file, the HDFS client request the NameNode for the list of DataNodes that host the replicas of the data blocks of the file. Then it directly contacts the DataNode and request the transfer of desired blocks. During writes, the client request the NameNode to choose a list of DataNode that can host the replicas of

the first block of the file. After choosing the client establishes a pipeline from node to node and sends the data block. After storing the first block, the client request for a new set of DataNode to host the replicas of the next block of the same file. New pipeline will be established between the new set of DataNodes and client sends the further bytes of the file

Figure 3. Interaction among HDFS client, NameNodes and Data Nodes



HDFS provide APIs to retrieve the location of a file block in the cluster. This allow to schedule the task to the node where data are located, thereby improving the read performance. This allows the application to set the replication factor of a file. By default the replication factor is three. For files which are frequently accessed or critical, setting the replication factor improves their tolerance against faults and increases the read bandwidth.

2.1 .MAINTAINING DATA RELIABILITY IN HDFS

3.1.1. Images and Journal

The Namespace image is the file system metadata that describes the organization of application data as directories and files. A persistent record of the image written to disk is called a checkpoint. For each client-initiated transaction, the change is recorded in the journal, and the journal file is flushed and synched before the change is committed to the HDFS client. The NameNode is a multithreaded system and process request simultaneously from multiple clients. To optimize the saving of transaction to disk, the NameNode batches multiple transactions initiated by different clients. When one of the NameNode‟s threads initiates a flush-and sync operation, all transaction batched at that time are committed together. Remaining threads only need to check their transactions have been saved and do not need to initiate a flush-and-sync operation

3.1.2 CheckpointNode

The CheckpointNode periodically combines the existing checkpoint and journal to create a new checkpoint and an empty journal. The CheckpointNode runs on a different host from the NameNode. The system can start from the most recent checkpoint if all other persistent copies of the namespace images or journal are unavailable

3.1.3. ***BackupNode***

It creates periodic checkpoints and in addition it maintains an in-memory, up-to-date image of the file system namespace that is always synchronized with the state of the NameNode. The BackupNode accepts the journal stream of namespace transactions from the active NameNode, saves them to its own storage directories, and applies these transactions to its own namespace image in memory. If the NameNode fails the BackupNodes image in memory and the checkpoint on disk is a record of the latest namespace state. It can perform all operation of the regular NameNode that do not involve modification of the namespace or knowledge of block locations.

3.1.4. FileSystem Snapshots

File system snapshot helps to persistently save the current state of the file system; it helps to rollback in case of failure during upgrade. This helps HDFS to return to the namespace and storage state as they were at the time of snapshot. During handshake the NameNode instructs DataNodes whether to create a snapshot. The local snapshot on DataNode is not created by replicating data files as it double the storage capacity needed for every DataNode on the cluster. Instead each DataNode creates a copy of storage directory and hard links existing block files into it. When a data block is removed, it removes only the hard link and block modification during append use copy-on-write technique. Thus old block replica remains untouched in their old directory

3.1.5. FILE OPERATION AND DATA PLACEMENT

HDFS implements a single write, multiple-reader model. The client that opens a file for writing is granted a lease for that file and no other client can perform write to the file. The writer client renews the lease by periodically sending heartbeats to the NameNode. The lease is revoked when the file is closed. After writing data to the file, the user application explicitly calls hflush operation. Current packet is pushed to the pipeline and hflush operation waits until all DataNodes in the pipeline acknowledges the successful transmission of the packet. This makes all data written before hflush operation visible to readers . BlockPlacement The placement of replicas is critical to HDFS data reliability and performance. It should improve data reliability, availability and network bandwidth utilization. HDFS use configurable block placement

policy. HDFS place the first replica of the new block on the node where the writer is located and second and third replicas on two different nodes in a different rack and rest of the replicas
are placed on random nodes with the restriction that no more than one replica is placed at one node and no more than two replicas are placed in the same rack. Placement of second and third replica on a different rack distributes the block replicas for a file across the cluster. After selecting the target nodes, the nodes will be organized as a pipeline in the order of their proximity to the first replica. The data is pushed in same order. During read, NameNode check whether client‟s host is located in the cluster or not. If yes, block location are returned to the client in the order of closeness to the reader. This policy reduces the inter-rack and inter-node write traffic and generally improves write performance because the chance of a rack failure is far less than that of node failure. In case of three replicas, it can reduce the aggregate network bandwidth used when reading data since a block is placed in only two unique racks rather than three. HDFS replica placement policy:- 1. No DataNode contains more than one replica of any block. 2. No rack contains more than two replicas of the same block, provided there are sufficient racks on the cluster

3.1.6. PROPOSED STUDY REPLICA MANAGEMENT IN HDFS

The NameNode takes care of the number of replicas of each block to be kept in file system. The NameNode determine the over and under replication of a data block when it receives a block report from the DataNodes. In case of over replication the NameNode choose a replica to be removed. NameNode prefer not to reduce the number of racks that host the replicas, but prefer to remove replica from the DataNode with least amount of available disk space. When a block becomes under replicated, it is put in the replication priority queue. A block with one replica has highest priority, while block with a number of replicas that is greater than two thirds of its replication factor has the lowest priority. If the number of existing replicas is one, HDFS places the next replica on a different rack. In case that the block has two existing replicas, if the two existing replicas are on the same rack, the third replica is placed on a different rack; otherwise, the third replica is placed on a different node in the same rack as an existing replica. If the NameNode detects that a block‟s replicas end up at one rack, the NameNode treat the block as under replicated and replicates the block to a different rack. After the NameNode receives the notification that the replica is created, the block becomes over-replicated. The NameNode then will remove an old replica because the over replication policy prefers not to reduce the number of racks. HDFS use balancer tool (application program run by cluster administrator) that balances disk space usage on a HDFS cluster. It iteratively move replicas based on utilization of DataNodes i.e., from DataNode with higher utilization to DataNode with lower utilization. Balancer decision guarantees that it doesn‟t reduce either number of replicas or the number of racks. It optimizes balancing process by minimizing the inter-rack data copying.\ Each DataNode runs a block scanner to verify the checksum of stored data block. Verification time of each block will be logged. Checksum can be computed by the read client and is reported to the DataNode as checksum verification. When a corrupted block is found, it notifies the NameNode, the NameNode will mark the replica as corrupted and remove it only after replicating sufficient number of good replicas of the block (based on replication factor) to other node
. When a DataNode is marked as decommissioning, it will not be selected for replica placement but continue to serve data request. NameNode schedule the replication of the block in the decommissioned DataNode to other DataNode. When all blocks are replicated to other DataNode, the decommissioned DataNode enter „decommissioned‟ state and then it can be removed safely from cluster without affecting the data availability.

3. ARCHITECTURAL DRAWBACKS

Regardless of the high storage capacity of the HDFS, it has several limitations. A centralized server (NameNode) maintaining the important information is a bottleneck for the architecture.

4.1. Scalability Limitation

The entire HDFS namespace is stored in the live memory of the Centralized NameNode Server. This impose restriction on the scalability of storage capacity i.e., the storage capacity of the cluster cannot grow beyond the available free memory space on the NameNode. It is estimated that approximately 1 GB memory is needed per petabyte of data in the cluster. Thus entire Hadoop cluster depends on the performance of the single NameNode and its capacity to handle the namespace.

4.2. Performance Limitation

NameNode receives heartbeat and block report from all the DataNodes in the cluster and it is responsible for providing all metadata information of the data blocks in the cluster to the client applications. As the cluster size increases with increase in the number of data blocks, the NameNode performance reduce. At peak usage the NameNode will be in bottleneck.

4.3. Availability Limitations

Even if HDFS provide high level of availability, it had a single point of failure problem which affects

the availability. The entire system relies on a single NameNode to maintain a cluster of Data Nodes. The cluster will be offline until it is fully recovered from the failure. Time taken for a secondary NameNode to boot up depends on the cluster size and the metadata size.

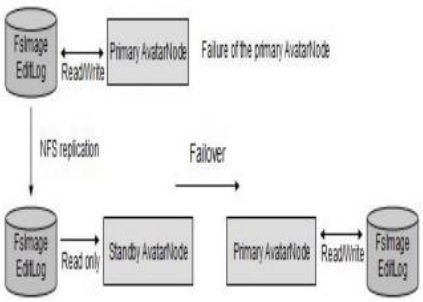
4.4. ARCHITECTURAL SOLUTION FOR THE DRAWBACKS

Several strategies have been proposed for eliminating the single point of failure and increasing the storage capacity of the architecture Some of them are:

4. AVATAR NODE

AvatarNode provide fast failover mechanism for dealing with single point of failure of NameNode. Primary AvatarNode work same as NameNode and writes its transaction log into the shared NFS storage. If another instance of AvatarNode is instantiated, it runs in standby mode. The standby AvatarNode encapsulates an instance of NameNode and secondary NameNode. The standby AvatarNode keep on reading the logs from the shared NFS storage and keeps feeding the transaction logs to the encapsulated NameNode instance. NameNode encapsulated in

Figure 4. Switching from primary to standby AvatarNode during failover

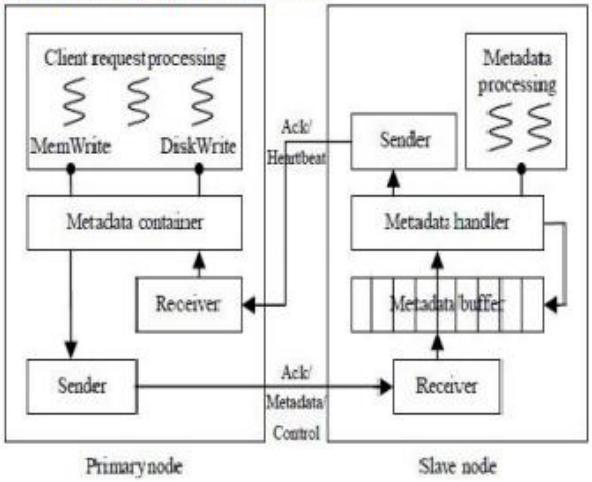


the standby AvatarNode is kept in safe mode to prevent it from participating in cluster activities. HDFS use virtual IP (VIP) address to access AvatarNode.

When a primary fails, the failover is performed by switching the VIP to a standby AvatarNode. File reads are not affected by failover switching, but client receives I/O exception for the file writes occurred during failover event. Failover doesn't affect MapReduce task execution as it is designed to retry failed tasks

5. HIGH AVILABILITY THROUGH METADATA REPLICATION

Figure 5. Metadata replication



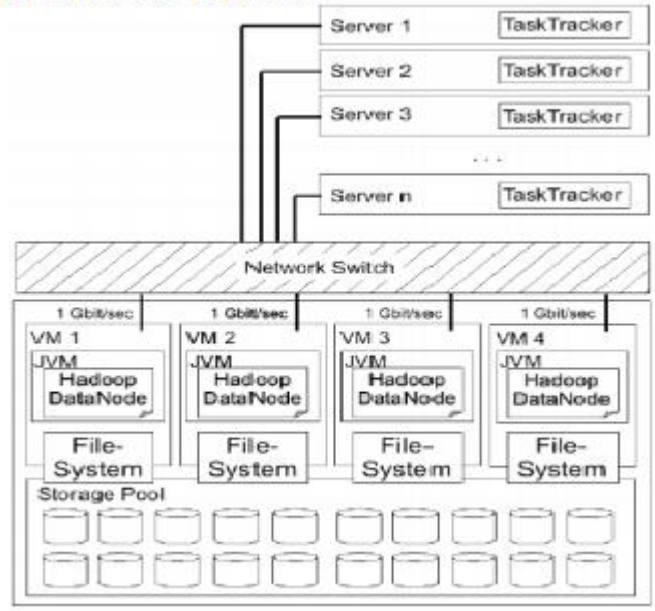
This solution was proposed in Feng Wang‟s paper. There are three major phases, Initialization phase (initialize the execution environment), Replication phase (replicate metadata from critical node to corresponding BackupNode at runtime) and Failover phase (resume the running of Hadoop). During Initialization phase multiple slave node register with the primary node for the up-todate metadata information. In Replication phase, the primary node collect metadata from client request processing threads and send it to slave node. The slave node processes the received metadata and sends a heartbeat signal to primary node to keep track its current status.

In case of failure of the primary node, the slave node doesn't receive any acknowledgement for the heartbeat message. Then a leader election algorithm will be initiated to select a slave node to replace the failed primary node. The selected slave node change its IP address to the IP address of failed primary node and start processing the client requests

DECOUPLING STORAGE AND COMPUTATION FROM DATANODE In Hadoop, the storage and computation are tightly coupled which impose so many limitations. First the ratio of computation to storage changes over time which results in having more data than the storage initially provided. Secondly when workload varies there will be a need to power down or repurpose some node for other application. Since in Hadoop data is spread over all nodes, there will be a need of migration of data to some other node before power down or repurposing the node. SuperDataNode is server containing more disks than a traditional Hadoop DataNode. SuperDataNodes decouples the amount of storage from the number of DataNodes. It requires a large amount of aggregate bandwidth to the network. Each SuperDataNode also host a set of virtual machines. Each VM execute unmodified copy of Hadoop

DataNode process. Disks are organized as a single storag

Figure 6. SuperDataNode



Using SuperDataNode provides several advantages like: Decoupling the amount of storage from numberof nodes make it possible to power down or repurpose them to run other application during the period of low utilization. It supports processing archival data which will• be accessed more infrequently. All archival data will be migrated to SuperDataNode which allows the availability when needed. Use of SuperDataNode increases the uniformity• for job scheduling and data block placement, since all the virtual DataNode process running on it shares same underlying storage pool, so any local DataNode is good candidate for scheduling the given task. There are some disadvantages associated with using SuperDataNode Task execution performance depends on• bandwidth of the network link. Result of failure of a single SuperDataNode is• significantly worse compared to the failure of single traditional Hadoop DataNode. Implementation of SuperDataNode is costly

7. CONCLUSION

Hadoop represent an increasingly important approach for data-intensive computing. This paper explore through the components of the Hadoop system, HDFS and MapReduce. It also successfully pointed out the architecture of HDFS, distribution of data across the cluster based on the client applications. The paper took a deep study on the replication management done for better performance and architectural drawback exists in traditional HDFS which affect the High Availability of data. Several architectural solutions have been analyzed which can overcome the drawbacks and possibility of achieving High Availability. Enhanced Hadoop system can effectively serve as better solution for the analysis of "Big Data" compared to any other complex computational system.

8. REFERENCES

1. Shvachko K, Kuang H, Radia S and Chansler R. The Hadoop Distributed File System in Proceedings of the 26th IEEE Symposium on Massive Storage Systems and Technologies, 2010.
2. Shafer J, Rixner S, Cox AL. The Hadoop Distributed Filesystem: Balancing Portability and Performance, in Proceedings of IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS 2010), White Plains, NY, 2010.
3. Feng Wang et al. Hadoop High Availability through Metadata Replication, IBM China Research Laboratory, ACM, 2009.
4. George Porter. Decoupling storage and computation in Hadoop with SuperDataNodes, ACM SIGOPS Operating System Review, 44, 2010.
5. Derek Tankel. Scalability of Hadoop Distributed File System, Yahoo developer work, 2010.
6. Jeffrey dean and Sanjay Ghemawat. Map Reduce: Simplified Data Processing on Large Clusters, Google, 2004.
7. Anonymous. Hadoop Distributed File System: HDFS Architecture: http://hadoop.apache.org/common/docs/r0.20.1/hdfs design.html, 2010.
8. Anonymous. High Availability