John Ziegler
Professor Ranganathan
CSCI 6600
07/12/2022

ARCHITECTURE:

Hadoop is an open-sourced Apache software platform that provides users a reliable means for managing and supporting huge amounts of data in big data environments. Hadoop serves two primary data functions: storage, and processing. Hadoop consists of 4 components: 1) Hadoop Distributed File System (HDFS) - which stores data; 2) YARN (Yet Another Resource Negotiator) - which allocates system resources, and schedules jobs. 3) MapReduce - which enables users to split processing into multiple tasks that run in parallel; and 4) Hadoop Common - a set of Java dependencies required by Hadoop modules. With these components, Hadoop can store and execute computations on enormous amounts of data across thousands of servers in parallel. This paper will focus on the underlying mechanics of the Hadoop Distributed File System (HDFS).

NameNodes are dedicated servers that store metadata. HDFS uses a master-slave design pattern where NameNodes act as the master and central repository of the system. Metadata, for a NameNode includes an Image - a hierarchical mapping of where the files in the cluster are stored , the Checkpoint - a secondary copy of the Image used for both restarts and redundancy, and lastly, a Journal - a log of modifications made to the Image that can be redeployed alongside the original Image to restore the namespace details.

DataNodes are small virtual hosts where application data is stored. DataNodes are characterized by two files, 1) a file including the data itself, and 2) a file including the metadata for the block location where that data resides.These virtual hosts work together in large quantities to balance the computational load for the NameNodes. DataNodes independently preserve the integrity of the system by performing handshakes to verify cluster registration. They also communicate regularly to the NameNodes informing them of replica locations and heartbeats - which are system critical health checks that ensure service by the thousands - similar to how the ICMP protocol is commonly used.

The HDFS Client is how the user interacts with the Nodes of the Cluster. The HDFS Client allows us to perform read, write, and delete operations on the files as needed. When reading, the client asks the NameNode for the list of DataNodes that host the replicated copies of the files blocks. The Client then requests the DataNode to transfer it. When the client writes, the NameNode chooses DataNodes to host the replicas of the first block of the file, and sends the data. When the first block is full, the NameNode requests new DataNodes to host the next replicas and sends the next blocks of the file. HDFS uses an API that shows the location of file blocks, this allows the use of MapReduce to perform the task at the location where data is stored, and increases our read performance.

In HDFS we use *Images* to organize the data into directories and files. We use these images to create a list of changes that have been made to the data, and store them in the journal. If changes are made to the data, the journal keeps track of these changes, and is then synced up to the HDFS client. The NameNode uses the image to replay changes from the journal until the image is updated to the current state. Then it creates a new checkpoint, and replaces the journal with an empty one. It is good practice to save checkpoints and journals in different directories in case one is missing, or becomes corrupt.

CheckpointNode

CheckpointNodes are used to ensure that our journal doesn't get too long, which would slow down processing. They are executed by the NameNode to periodically combine the checkpoint data, and the journal to create a new checkpoint and an empty journal.

BackupNode

BackupNodes has the same capabilities as a CheckpointNode, but they maintain an up-to-date image of the file system in the BackupNode(s) memory that is always synchronized with the NameNodes state. This means we don't have to download checkpoint and journal files, since the active state is already in memory. This allows us to run the NameNode without persistent storage.

Upgrades, File System Snapshots

Snapshots are a way to save the current state of the file system in order to protect ourselves against missing or corrupted files. They are commonly used during system upgrades to give us the ability to rollback to a previous version of our system if something is corrupted/missing.

FILE I/O OPERATIONS AND REPLICA MANAGEMENT:

File Read and Write

HDFS implements a single-writer, multiple reader model. This means that when it needs to add data, it creates a new file and writes to it. After the file is closed, the bytes that were added can't be changed, only added to. When the client is writing to a file it receives a lease to make changes to the file, and this lease is continuously renewed by sending what we call a heartbeat back to the NameNode. Data written to the file is sent in packets, and uses CheckSums to verify that the packets were received and not corrupted. When we create a HDFS file, the blocks data, and the Checksum are sent to the client. The client then verifies that they are the same; if they are not then it lets the NameNode know the replica is corrupt, and then tries a different replica from the next closest DataNode. It will go in order of the closest to the reader to increase efficiency.

Block Placement

HDFS uses racks to allow us to connect our data closer together for optimization. The goal here is to increase bandwidth as much as possible. When we have thousands of DataNodes, the distance between 2 that need to connect with each other grows in size. To fix this we use racks, which are a number of DataNodes and their replicas that live closer together to be used to speed up processing time and increases reliability as rack failure isn't as common as node failure.

Replication management

Another responsibility the NameNode has is to ensure that we have the proper number of replicas made. If a DataNode has too many, or too few copies made available, the NameNode will adjust this accordingly to balance the storage utilization, and to allow the DataNode to still be highly available. If we need to add DataNodes, it will always try to find a location on a different rack then we are now to speed up processing time.

Balancer

Balancing the disk space utilization is done by the balancer. This is used by the cluster to ensure that storage is equally distributed among all disks. It runs with a threshold value, and moves DataNodes to different racks, or creates new ones on different racks to ensure that the utilization of the node doesn't exceed the threshold value. The setback here is how much bandwidth we allow the balancer to use, as it will balance the state faster, but with reduced speed in processing.

Block Scanner

DataNodes run a block scanner to verify that the checksums match between the block data and the stored data. If they are the same, it informs the DataNode and acts as a verification of the replica. If they are not the same, the NameNode marks the file as corrupt, and deletes it only after it replicates a good copy of the block to reach the replication factor. This ensures that we preserve data as long as possible, and allows us to retrieve data even from corrupt replicas.

Decommissioning

The cluster administrator lists the host addresses of which nodes are allowed, and not allowed into the cluster. A node in a cluster can be decommissioned without corrupting any data by replicating the blocks of data to other nodes, and after all blocks are replicated, the node enters the decommissioned state and can be removed safely.

Inter-Cluster Data Copy

In conclusion the HDFS allows us to use multiple different nodes to store our data without compromising fast processing time. It also ensures the reliability, and availability of the data by making multiple replicas of the data across different nodes, on different racks, allowing it to more easily choose a closer path to the data in order for us to achieve faster processing times. They are still attempting new ways to provide more efficient cooperation between different clusters, as they believe that multiple clusters are better than 1 larger cluster due to the increased availability and isolation between applications.