— HDFS Architecture — # 2.3 Data Replication and Persistence of FileSystem Metadata

Data Replication

HDFS is designed to reliably store very large files across machines in a large cluster. It stores each file as a sequence of blocks. The blocks of a file are replicated for fault tolerance. The block size and replication factor are configurable per file. All blocks in a file except the last block are the same size although users can start a new block without filling out the last block to the configured block size after the support for variable length block was added to append and hsync.

An application can specify the number of replicas of a file. The replication factor can be specified at file creation time and can be changed later. File in HDFS are write-once (except for appends and truncates) and have strictly one writer at any time. The NameNode makes all decisions regarding replication of blocks. It periodically receives a Heartbeat and a Blockreport from each of the DataNodes in the cluster, Receipt of a Heartbeat implies that the DataNode is functioning properly. A Blockreport contains a list of all blocks on a DataNode.

Replica Placement: The first baby steps

The placement of replicas is critical to HDFS reliability and performance. Optimizing replica placement distinguishes HDFS from most other distributed file systems. This is a feature that needs lots of tuning and experience. It is still a work in progress.

Currently, one replica is placed in the same rack, for increased data throughput access in case of failure and the other replica in another rack. The risk of an entire rack failing is quite low, which is why storing it on a node in the same racks is a good trade-off between optimizing performance and risk.

Replica selection

The replica closest for satisfying a read request is chosen.

Safemode

On start-up, the NameNode enters a special state called Safemode. Replication of data blocks does not occur when the NameNode is in the Safemode state. The NameNode receives a Heartbeat and Blockreport messages from the DataNodes. A Blockreport contains the list of data blocks that a DataNode is hosting. Each block has a specified minimum number of replicas. A block is considered safely replicated when the minimum number of replicas of that data block has checked in with the NameNode. After a configurable percentage of safely replicated data blocks checks in with NameNode (plus an additional 30 seconds), the NameNode exits the Safemode state. It then determines the list of data blocks (if any)

that still have fewer than the specified number of replicas. The NameNode then replicates these blocks to other DataNodes.

The persistence of File System Metadata

The HDFS namespace is stored by the NameNode. The NameNode uses a transaction log called the EditLog to persistently record every change that occurs to filesystem metadata. For example, creating a new file in HDFS causes the NameNode to insert a record into the EditLog indicating this. Similarly, changing the replication factor of a file causes a new record to be inserted into the EditLog. The NameNode uses a file in its local host OS file system to store the EditLog. The entire file system namespace, including the mapping of blocks to files and file system properties, is stored in a file called the FsImage. The FsImage is stored as a file in the NameNode's local file system too. The NameNode keeps an image of the entire file system namespace and file Block map in memory When the NameNode starts up, or a checkpoint is triggered by a configurable threshold, it reads the FsImage and EditLog from disk, applies all the transaction from the EditLog to the in-memory representation of the FsImage, and flushes out this new version into a new FsImage on disk. It can then truncate the old EditLog because its transaction has been applied to the persistent FsImage. This process is called a checkpoint. The purpose of a checkpoint is to make sure that HDFS has a consistent view of the file system metadata by taking a snapshot of the file system metadata and saving it to FsImage. Even though it is efficient to read an FsImage, it is not efficient to make incremental edits directly to an FsImage. Instead of modifying FsImage for each edit we persist the edits in the EditLog. During the checkpoint the changes from EditLog are applied to the FsImage. A checkpoint can be triggered at a given time interval (dfs.namenode.checkpoint.period) expressed in seconds or after a given number of filesystem transactions have accumulated (dfs.namenode.checpoint.txns). If both of these properties are set, the first threshold to be reached triggers a checkpoint. The DataNode stores HDFS data in files in its local file system. The DataNode has no knowledge about HDFS files. It stores each block of HDFS data in a separate file in its local file system, The DataNode does not create all file s in the same directory. Instead, it uses a heuristic to determine the optimal number of files per directory and creates subdirectories appropriately. It is not optimal to create all local files in the same directory because the local file system might not be able to efficiently support a huge number of files in a single directory. When a DataNode starts up, it scans through its local file system, generates a list of all HDFS data blocks that correspond to each of these local files, and sends this report to the NameNode. The report is called the Blockreport.

Sources

1. Main