Overview

Google File System (GFS) is a kind of distributed file system, developed by Google. It possesses the following two main advantages. One, it provides reliable and efficient access to data with fault-tolerance, which adopts large cluster of commodity machines. The other, GFS allows multiple clients to share and store data on multiple machine concurrently. To attain the above two advantage, GFS has the following three attributes. First, files in GFS are large compared with traditional standard, commonly Multi-GB size of files. Second, given the first property, most files are mutated by appending new data instead of overwriting existing data. Last, because GFS is composed by inexpensive commodity machines which often fails, it must constantly monitor itself and detect, tolerate, and recover promptly from component failures on a routine basis.

Architecture

The architecture of GFS is composed of multiple chunksevers and a single master. The files are divided into fixed-sized, 64MB, of chunks. Besides, such large unit chunk size reduces clients’ need to intect with the master, which has the following three benefits. First, the client can read and write large files efficiently. Secondly, it can reduce network overhead by keeping a persistent TCP connection to the chunksever over the extended period of time. Third, it reduces the size of the metadata stored on the master. The chunksevers store chunks on local disks and read or write chunk data specified by a chunk handle and byte range. In order to maintain reliability, each chunk is replicated on multiple chunksevers, three replicas by default. As a result, when GFS identifies failed chucksevers by regular handshakes between master and all chucnksevers and detects data corruption by checksumming, the data is restored from valid replicas as soon as possible.

The master machine maintains all file system metadata, including namespace, access control information, the mapping from files to chunks, and current locations of chunks. The master periodically communicates with each chunkserver in HeartBeat messages to give it instructions and collect its state. Furthermore, as for metadata, all metadata is kept in the master’s memory. The first two types are also kept persistent by logging mutations to an operation log stored on the master’s local disk and replicated on remote machines. Using a log allows us to update the master state simply, reliably, and without risking inconsistencies in the event of a master crash. Furthermore, recovery needs only the latest complete checkpoint and subsequent log files. Older checkpoints and log files can be freely deleted, though we keep a few around to guard against catastrophes. A failure during check-pointing does not affect correctness because the recovery code detects and skips incomplete check points. The master asks each chunkserver about its chunks at master startup and whenever a chunkserver joins the cluster.

The operation of GFS is following. To begin with, he client sends the master a request containing the file name and chunk index. Secondly, the master response corresponding chunksevers with chunk handle and locations of the replicas. Third, the clients receive the information of the second step as a key to send a request to one of the replicas, which specifies specifies the chunk handle and a byte range within that chunk. Also, further reads of the same chunk require no more client-master interaction until the cached information expires or the file is reopened. That is to say, in this process of operation, neither the client nor the chunksever needs to cache file data for the following respective reasons. For clients, they own little benefit to cache huge files and large working sets. For chunksevers, they need not to cache files data because chunks are stored as local files.