The Google File System

By: Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung

Stanley Yang
11/22/2013

Main Idea Behind GFS

- Google has to develop a system that is able to meet its rapidly growing needs.
- It must be able to handle terabytes of data being read and written by multiple clients concurrently.
- The system must be extremely fault tolerant.
- Everything will be control by a single master. The data will be stored on chunkservers.

Implementation - Design

- There will only be one master to simplify the design.
- The master will handle all the metadata (namespace, access control, chunk locations, file to chunk mapping) in its memory.
- This allows it to handle system wide activities such as garbage collection, load balancing, and chunk lease management.
- Files will almost never be overwritten. Any new data will be appended to the end of the file.

Implementation - Master

- The master stores nearly everything in memory. A reference to a 64MB chunk will only use up 64 bytes. If the file system becomes even larger than expected, increasing the master's memory is a small price to pay for the simplicity, reliability, performance, and flexibility that comes from storing it in memory.
- Chunk locations are not store persistently on the master. They are generated when the master is first started by asking its chunkservers and whenever a new chunkserver joins the cluster and the information is stored in memory. They are updated when the master sends out a heartbeat to the cluster.

Implementation – Master Cont’d

- The master logs every action it performs. This log file is critical to the entire file system because files and chunks are uniquely identified by the local time they are created. If the log file were to be lost, entire chunks of data would disappear even though the physical data still exists. Because of this, the log file is stored on disk and cloned onto remote machines.
- The master performs a checkpoint if the log file is becoming too large. A checkpoint is basically a snapshot of the master at that point in time. It is a compact B-Tree that can mapped directly onto memory. Then a new log file is created. This way, in the event of a crash, recovery would only require mapping the checkpoint back then replaying the actions in the most recent log file.

Implementation - Chunkservers

- The hardware will be inexpensive commodity Linux machines. Expected to have failures.
- Chunks will be 64 MB. This is much larger than the standard which is usually 4-8 KB.
- They will use lazy allocation to avoid wasting space. Lazy allocation means a chunk will only be allocated when needed.
- They get the final say on what chunks will be used on their own disks.

Personal Analysis

- I think agree with Google’s decision to have one main master. It simplifies their design a lot. It allows for consistency throughout their giant clusters and enables them to have better control over it.
- I think the log file and checkpoint system is a great idea. It allows for easy and rapid recovery. Worst case scenario is the checkpoint not being complete and the master crashing at that point. Then the master simply has to run through a larger than normal log file.

Advantages

- Older states of files are stored so even if the newest version is corrupted, there is still something useable.
- Metadata being stored in memory, on a single master, allows for fast operations to happen.
- Having a single master also allows the file system to make sophisticated chunk placements and replication decisions with its global knowledge.

Disadvantages

- Single Point of Failure. Everything is reliant on the master. Although there are effective measures in place to prevent total loss.
- Files take up more room because their older versions are not deleted.

Real-World Use Cases

- The design decision to never delete data and only append is probably how Google is able to offer cache's of sites it has crawled onto before.
- The technology behind GFS is what powers Google Docs and how it enables multiple users to work on the same file at the same time.
- Google has used and has been updating this file system. The current version is known as Colossus. It is what has enabled Google to provide real time searches. The big change in Colossus is that it uses triggers which are known to be slow but Google has managed to use them efficiently