WIKIPEDIA

Google File System

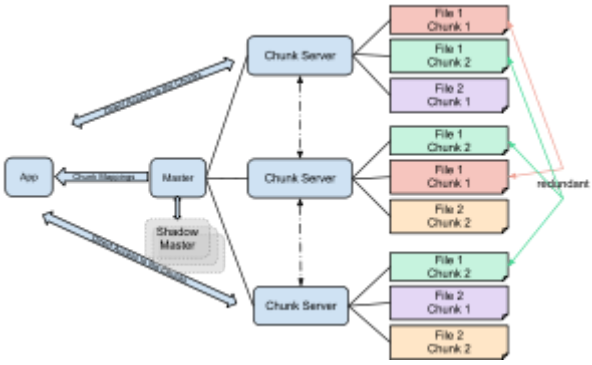
Google File System (**GFS** or **GoogleFS**) is a proprietary distributed file system developed by Google to provide efficient, reliable access to data using large clusters of commodity hardware. The last version of Google File System codenamed Colossus was released in 2010.[1][2]

Google File System	
Operating system	Linux kernel
Type	Distributed file system
License	Proprietary

Contents

Design

GFS is enhanced for Google's core data storage and usage needs (primarily the search engine), which can generate enormous amounts of data that must be retained; Google File System grew out of an earlier Google effort, "BigFiles", developed by Larry Page and Sergey Brin in the early days of Google, while it was still located in Stanford. Files are divided into fixed-size *chunks* of 64 megabytes, similar to clusters or sectors in regular file systems, which are only extremely rarely overwritten, or shrunk; files are usually appended to or read. It is also designed and optimized to run on Google's computing clusters, dense nodes which consist of cheap "commodity" computers, which means precautions must be taken against the high failure rate of individual nodes and the subsequent data loss. Other design decisions select for high data throughputs, even when it comes at the cost of latency.



Google File System is designed for system-to-system interaction, and not for user-to-system interaction. The chunk servers replicate the data automatically.

A GFS cluster consists of multiple nodes. These nodes are divided into two types: one *Master* node and multiple *Chunkservers*. Each file is divided into fixed-size chunks. Chunkservers store these chunks. Each chunk is assigned a globally unique 64-bit label by the master node at the time of creation, and logical mappings of files to constituent chunks are maintained. Each chunk is replicated several times throughout the network. At default, it is replicated three times, but this is configurable [3]. Files which are in high demand may have a higher replication factor, while files for which the application client uses strict storage optimizations may be replicated less than three times - in order to cope with quick garbage cleaning policies [3].

The Master server does not usually store the actual chunks, but rather all the metadata associated with the chunks, such as the tables mapping the 64-bit labels to chunk locations and the files they make up (mapping from files to chunks), the locations of the copies of the chunks, what processes are reading or writing to a particular chunk, or taking a "snapshot" of the chunk pursuant to replicate it (usually at the instigation of the Master server, when, due to node failures, the number of copies of a chunk has fallen beneath the set number). All this metadata is kept current by the Master server periodically receiving updates from each chunk server ("Heart-beat messages").

Permissions for modifications are handled by a system of time-limited, expiring "leases", where the Master server grants permission to a process for a finite period of time during which no other process will be granted permission by the Master server to modify the chunk. The modifying chunkserver, which is always the primary chunk holder, then propagates the changes to the chunkservers with the backup copies. The changes are not saved until all chunkservers acknowledge, thus guaranteeing the completion and atomicity of the operation.

Programs access the chunks by first querying the Master server for the locations of the desired chunks; if the chunks are not being operated on (i.e. no outstanding leases exist), the Master replies with the locations, and the program then contacts and receives the data from the chunkserver directly (similar to Kazaa and its supernodes).

Unlike most other file systems, GFS is not implemented in the kernel of an operating system, but is instead provided as a userspace library.

Interface

The Google File System does not provide a POSIX interface.[4] Files are organized hierarchically in directories and identified by pathnames. The file operations such as create, delete, open, close, read, write are supported. It supports Record Append which allows multiple clients to append data to the same file concurrently and atomicity is guaranteed.

Performance

Deciding from benchmarking results,[3] when used with relatively small number of servers (15), the file system achieves reading performance comparable to that of a single disk (80–100 MB/s), but has a reduced write performance (30 MB/s), and is relatively slow (5 MB/s) in appending data to existing files. The authors present no results on random seek time. As the master node is not directly involved in data reading (the data are passed from the chunk server directly to the reading client), the read rate increases significantly with the number of chunk servers, achieving 583 MB/s for 342 nodes. Aggregating multiple servers also allows big capacity, while it is somewhat reduced by storing data in three independent locations (to provide redundancy).

See also

- Bigtable
- Cloud storage
- CloudStore
- Fossil, the native file system of Plan 9
- GPFS IBM's General Parallel File System
- GFS2 Red Hat's Global Filesystem 2
- Hadoop and its "Hadoop Distributed File System" (HDFS), an open source Java product similar to GFS
- List of Google products

- MapReduce

References

1. Hoff, Todd (2010-09-11). "Google's Colossus Makes Search Real-Time by Dumping MapReduce" (http://highscalability.com/blog/2010/9/11/googles-colossus-makes-search-real-time-by-dumping-mapreduce.html). *High Scalability*. Archived (https://web.archive.org/web/20190404035948/http://highscalability.com/blog/2010/9/11/googles-colossus-makes-search-real-time-by-dumping-mapreduce.html) from the original on 2019-04-04..
2. Ma, Eric (2012-11-29). "Colossus: Successor to the Google File System (GFS)" (https://www.systutorials.com/3202/colossus-successor-to-google-file-system-gfs/). SysTutorials. Archived (https://web.archive.org/web/20190412112114/https://www.systutorials.com/3202/colossus-successor-to-google-file-system-gfs/) from the original on 2019-04-12. Retrieved 2016-05-10.
3. Ghemawat, Gobioff & Leung 2003.
4. Marshall Kirk McKusick; Sean Quinlan (August 2009). "GFS: Evolution on Fast-forward" (https://queue.acm.org/detail.cfm?id=1594206). *ACM Queue*. **7** (7). doi:10.1145/1594204.1594206 (https://doi.org/10.1145%2F1594204.1594206). Retrieved 21 December 2019.

Bibliography

- Ghemawat, S.; Gobioff, H.; Leung, S. T. (2003). "The Google file system". *Proceedings of the nineteenth ACM Symposium on Operating Systems Principles - SOSP '03* (http://static.googleusercontent.com/media/research.google.com/en//archive/gfs-sosp2003.pdf) (PDF). p. 29. CiteSeerX 10.1.1.125.789 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.125.789). doi:10.1145/945445.945450 (https://doi.org/10.1145%2F945445.945450). ISBN 1581137575.

External links

- "GFS: Evolution on Fast-forward", *Queue* (http://queue.acm.org/detail.cfm?id=1594206), ACM.
- "Google File System Eval, Part I", *Storage mojo* (https://storagemojo.com/google-file-system-eval-part-i/).
