A Brief Review of "The Google File System"

Yi Luo

1 Summary

The Google File System (GFS) is a novel file system to deal with large amount of data and serve a large number of clients and applications under the distributed environment. Unlike traditional file systems, GFS is inspired by the pragmatic requirements of Google's routine business and research. It arises from the reexamination of early file systems and is based on a series of practical assumptions and demands. For example, this system is deployed on inexpensive commodity hardware which means it might be unreliable. This requires the file system has very good fault tolerance. Second, GFS concerns more about files with larger size. The design and implementation expect Multi-GB files as common cases. But it does not provide optimization of the storage and manipulation of small files. Moreover, GFS has some very specific assumptions, such as the dominated writes should be appending data to files instead of overwriting. To me, these assumptions make GFS feasible and effective, but they also constrain the system's generality.
The architecture of GFS is not complicated. In general, it consists of three parts, a single master, multiple chunkservers and clients. The master maintains the system metadata , manages all chunkservers and executes global operations. Chunkservers store data in the form of chunks for the access of clients. Clients start reading or writing operations on behalf of applications and directly communicate to the master. Such simple structure design, incorporating the underlying complicated and accurate implementation realizes good performance of GFS. [1]

2 Merits

I'd like to discuss two interesting advantages in GFS.
First, GFS adopts the strategy to have a single master as the global system controller. This idea simplifies the design and is proved effective. However, I especially appreciate the preventive measures from possible negative effects of this single-master strategy. One is that the single master might cause bottleneck during data transmitting if reading and writing operations involve the master. However, GFS solves the issue gracefully. Clients just query the mater for chunk information and transmit data directly to chunkservers. The master is excluded from the real data transmitting. Another potential issue is when the single master goes down, how to recover the comprehensive system. Fortunately, GFS offers good ways to handle the exception. It maintains shadow masters which contain replica of the primary master from the beginning. If the primary master goes down, one of the backup master is selected as the new primary one, which guarantees the correct work of the system. [2]
Another meaningful point is the separation of the flow of control from the flow of data when clients attempt to make mutations on multiple chunks. Clients do not need to piggyback the controlling information when transmitting data to chunckservers. Instead, they just push data to servers in a pipelined fashion and send initial controlling order to the primary chunkserver. The primary server further forwards the original order as well as some of its own requests to other chunkservers. Decoupling the flow of control, which has smaller size and is more flexible, from the flow of data making the use of the network more efficiently.

3 Drawbacks

The good performance of GFS relies on the summary of Google's practical experience and some specific assumptions. However, this can also be the disadvantage because these assumptions lead to a limited generality. For example, GFS presumes that most writes are appending rather than overwriting. For Google's searching service, they just crawl new data, append them to old files and seldom change them because they can simply provide time stamps for discrimination. But if we transfer to another use case which requires more frequent overwrites, GFS might not be applicable.

4 Personal Thoughts

Above all, I really appreciate the respect of log files as diagnostic tools in GFS. In one of my own experience, I found it was really difficult to understand what had happed without appropriate log files. Now to me, I choose to keep necessary log files as far as space permits.
In addition, I feel a little bit curious about the solution the hot-spot problem which is mentioned in 2.5 Chunk Size. There the paper just gives a potential idea without deeper discussions. I was wondering if there is a good way to tackle it now.

References

[1] Shun-Tak Leung Sanjay Ghemawat, Howard Gobioff. The Google File System. *SOSP*, 2003.

[2] Jonathan Strickland. How the Google File System Works. *How Stuff Works*, 2008.