The Google File System – Critique

- Vidit Jan (862394973)

Summary:

This paper presents the file system developed by Google called the Google file system. Its main goal is to serve distributed data-intensive purposes. It is based on a master-slave architecture. It was built considering various assumptions. Firstly, it assumes that the system is built of many cheap components that are not very robust. Then, it assumes that the typical file is 100MB or larger, and the system does not need to optimize for small files. It also assumes the client would typically perform two types of read operations, random but small and large streaming reads. For the writes, the system is not optimized for frequent large file modifications, but large sequential ones. To serve a distributed architecture, it is optimized for multiple clients simultaneously accessing the same files. Finally, high bandwidth has been given priority over latency while designing the system.

Strengths:

The main strengths of GFS are fault tolerance, high availability and bandwidth. Fault tolerance has been considered at every part of the architecture. It follows a master-slave system. Moreover, it divides files into chunks of a default size of 64 MB, and saves three replicas across different slave machines. By replicating, it is able to manage whenever a slave machine fails or a chunk becomes corrupt. To balance performance and fault tolerance, GFS has relaxed consistency constraint, but still guarantees eventual consistency. It is highly scalable, able to manage petabytes of data across innumerable nodes. To avoid single point of failure at the master node, the master state is also replicated. It can restart instantly in case of a failure. Moreover, if the machine stops working, the system spins up a new master node elsewhere. The replicated "shadow" masters also aid in performance by providing read-only capability to the file system. Finally, the master can rearrange replicas. Based on the distribution it can move chunks for better load balancing.

Weaknesses:

Although GFS is highly optimized and fault tolerant, it might struggle where the use-case is different from those considered during the design process. For example, it might not offer best performance for files of small sizes. While GFS has various optimizations in place at the master node, it might still struggle due to only a single master node. Since all the requests have to go through the master node, this might present a bottleneck. In a similar vein, read/write operations different than those assumed might also cause performance bottlenecks.

Other comments:

Overall, I found the paper highly intuitive and well-explained. The authors showcased comprehensive evaluation demonstrating the superior performance of GFS under the assumed conditions. GFS has become a highly influential article, leading to the creation of industry standard platforms such as Hadoop (HDFS).