Project Reflection: The Google File System

Motivation

The aim of this paper was to design a file system, called the Google File System (GLS), to meet the rapidly growing demands of Google's data processing needs. The motivation for this research was to depart from old file system design assumptions, and design the file system specifically for Google's needs. The system shares the same goals as previous file systems, such as performance, reliability, scalability and availability, but by taking into account Google's application workloads and technological environment, they made some key improvements on traditional design methods.

Related Work

They mention the Andrew File System (AFS), which also provides a location independent namespace, but unlike AFS, GLS spreads a file's data over multiple storage servers, which is similar to the xFS and Swift systems, so it can provide aggregate performance and increased fault tolerance. As disks are cheap and replication is relatively simple compared with the RAID system, GFS only uses replication for redundancy, which consumes more raw storage than the xFS and Swift systems.

Solution

First, component failures are common, so constant monitoring, error detection, fault tolerance and automatic recovery are of vital importance for a good system.

Second, files are huge and managing billions of small objects is difficult, so design parameters, such as I/O operation and block sizes, must be adjusted.

Third, data is added to most files by appending instead of overwriting, so appending becomes the focus of performance optimization and atomicity guarantees, while caching data blocks in the client becomes less useful.

Fourth, co-designing the applications and file system API increases flexibility, such as introducing an atomic append operation to allow multiple clients to concurrently append a file.