Nathan Matsuda
CS443
2/1/07

1. Paper title and its author(s).

The Google File System
Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung

2. Brief one-line summary.

The paper describes a file system for large scale distributed, data-intensive applications that has been successful in a real-world scenario,

3. A paragraph of the most important ideas: perhaps a combination of their motivations, observations, interesting parts of the design, or clever parts of their implementation.

The design assumes that some component at the system will always be breaking, so monitoring , error checking, and automatic recovery are carefully designed and constantly deployed to protect the data. The file system is designed to work optimally for large objects in the gigabyte to multi-terabyte range, rather than for small kilobyte range objects since they rarely occur in this type of application. Moreover, in Google's usage, files rarely are overwritten or written in their interiors, they are almost always appended to. These changes are presented in a simple API that was designed for Google's specific applications.
The file system is organized as a tree, with directories, and the usual commands are available to read, write, open, close, etc. Google has added to new commands, "snapshot", which efficiently makes an exact copy of a directory structure or file, and "record append", which allows multiple clients to append data to file while ensuring atomicity. Google files are broken up into chunks, each with a unique handle that is assigned by a master server at creation time. A chunk is represented by a set of Linux files on a chunkserver. These chunks are replicated three times for redundency. The master, after creating a chunk, keeps track of its location, copies, etc, and handles garbage collection. This metadata is stored in memory, not on disk. As a result, it's very fast, but not persistent in a master crash. The chunk information is stored on disk at each chunkserver in an operations log, so on master startup it need only query each of its chunkservers for their chunk information and history. Data itself does not pass through the master, but instead goes directly to the appropriate chunkserver. The system was designed this way so that a client only needs to contact the master at the beginning of a read for the chunk location, and then the rest happens without master interaction. Chunks themselves are 64MB each, which is much bigger than a normal filesystem block size, but is more efficient for Google's applications where files are very large and usually read sequentially. They authors note one problem however: small, frequently accessed files may result in the corresponding chunk server to be flooded with requests for the same chunk. They solve this problem by replicating frequently accessed chunks so the load becomes distributed.
The paper goes into great detail on fault tolerance. The Google filesystem is built with standard off-the-shelf components, but handles critical user data. Safety features go all the way down to the read, write commands of the file system. The master ensures that namespace

changes are consistent across chunkservers, even in the event of a failed command. Chunk replicas are constantly monitored and replaced if corrupted. These replications are given priorities so that, for example, a chunk that has lost all other replicas will be bumped up to the top of the queue waiting for new replicas to be made. Within chunks itself, the master uses a 32bit checksum to check for corruption not detected by hardware.
The system was tested through microbenchmarking and with actual live usage. Read and write rates were considerably below the theoretical limit shown, but there was no comparison to other systems. The authors did note that in a intentionally induced double failure the system was fully prepared for another failure two minutes later.

4. A paragraph of the largest flaws; maybe an experiment was poorly designed or the main idea had a narrow scope or applicability. Being able to assess weaknesses as well as strengths is an important skill for this course and beyond.

The microbenchmarks used show no comparisons between alternate systems, nor do they have any information on the system's fault tolerance, which is a key component. However, in real-world deployment, they do show basic tests that the replication scheme does hold up. More fault tolerance data would be more convincing.

5. A last paragraph where you state the relevance of the ideas today, potential future research suggested by the article, etc.

Google is one of the leading online centralized computing systems with millions of users. This type of filesystem will likely become increasingly widespread as other companies like Mcrosoft follow. This paper stands apart from the other papers we've read because it describes a system that is actually in use - essentially it was proven itself where other paper topics have not.