The Google File System

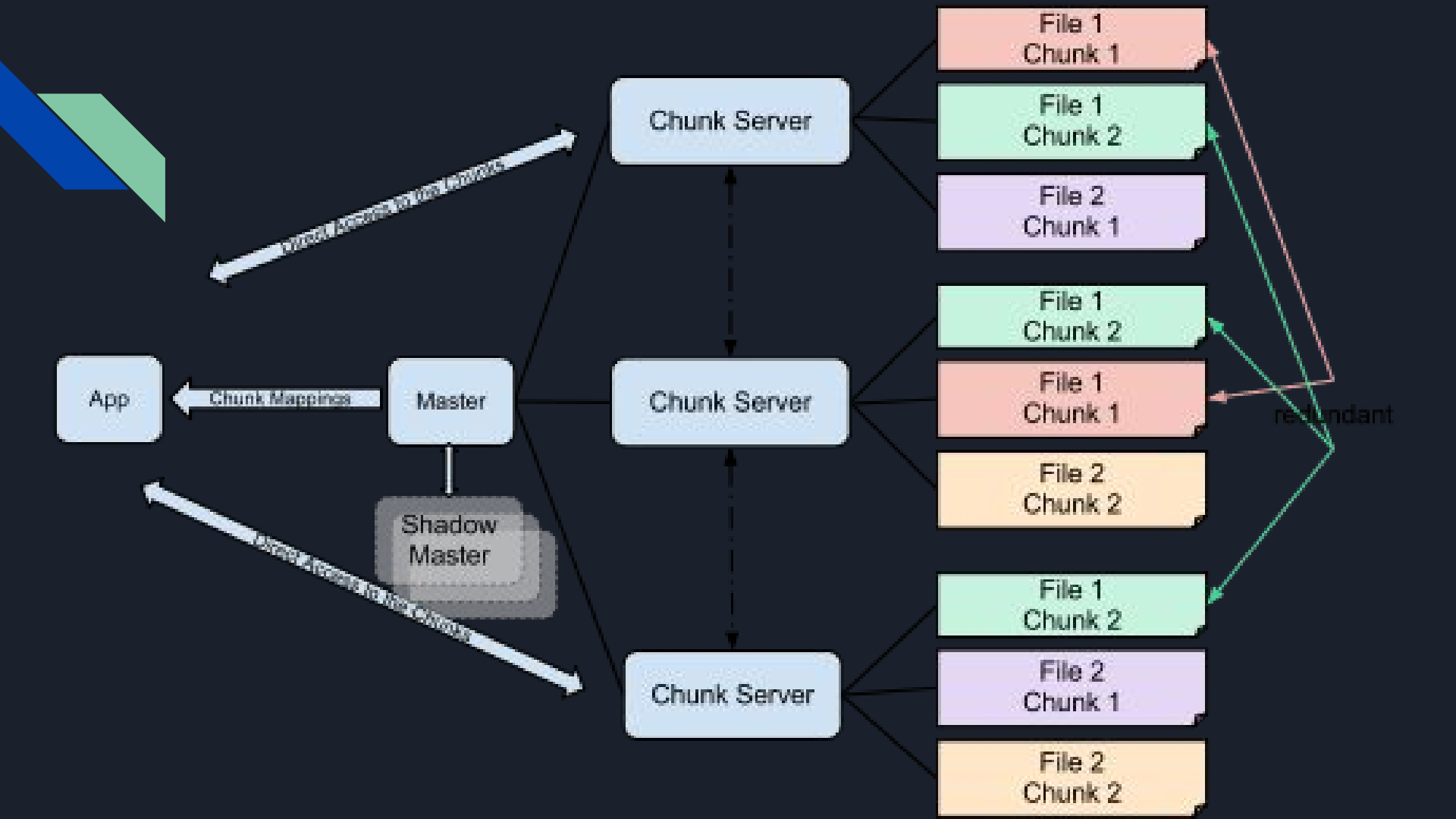
James Holden
10/30/2017

Main Idea

The Google File System(GFS) is a scalable distributed file system. All of Google's data is not stored in one place, so servers are used so all systems can access all data if needed. The hardware used for the GFS is run on inexepensive hardware. The GFS was created for use with data intensive applications. Its design is driven by both current and anticipated observations of the technology environment as well as application workloads

Implementation

- There is one master server, and many chunk servers
- Files are divided into fixed-sized chunks, and are given a 64-bit by the master node so they can be identified
- Chunks are then stored on the chunk servers
- The master server stores metadata associated with the chunks, such as logical mapping of the chunks 64-bit labels
- Chunks are stored as linux files


Chunk Server
File 1
Chunk 1
File 1
Chunk 2
File 2
Chunk 1
App
Chunk Mappings
Master
Chunk Server
File 1
Chunk 2
File 1
Chunk 1
File 2
Chunk 2
redundant
Shadow
Master
Chunk Server
File 1
Chunk 2
File 2
Chunk 1
File 2
Chunk 2

Implementation Analysis

From my understanding, this system efficiency increases alongside the number of chunk nodes in the system. It seems as though this type of system wouldn’t work as well for a smaller system with access to less resources. However, this system works exceptionally for google due to its availability of resources as well as the sheer amount of data the system uses. The inexpensive hardware fails often, but the recovery of the system is quick.

A Comparison of Approaches to Large-Scale Data Analysis

The paper evaluate and compares MapReduce and Parallel Database Management Systems (DBMSs). The paper compares performance and complexity of the systems. Tests were recorded on clusters up to 100 nodes. The findings reveal that the benchmarks were better in almost all tests for the DBMS.

Implementation

- MapReduce only has two functions. Map and Reduce. The Map function reads a set of records from an input file and outputs the records as new key/value pairs. The Reduce function executes as many instances as there are nodes of the Reduce program.
- Hadoop, DBMS-X, and Vertica were used in the tests.
- DBMS systems use indexing, whereas MapReduce does not use any indexing by default. This is the main advantage in efficiency of DBMSs.

Implementation Analysis

DBMS is a better model to use because it is more efficient is most tests.

There is more structure in DBMS, which is very important when used in systems that have extremely large amounts of data, because it can be kept more organized.

The paper states that MapReduce is simpler and easier to use so it may be a better choice for inexperienced users with smaller amounts of data.

Comparison of the Papers

GFS was created specifically for Google, so while it may not be best in every case, it is likely the best system for Google to use until new techniques are developed.

DBMS did outperform MapReduce without question, so in most cases DBMS and its relational diagrams should be used.

According to the Comparison paper, Google has announced plans to make a 1000 processor MapReduce cluster available to teach students distributed programming. MapReduce may be a more useful tool for teaching, rather than in actual use.

StoneBraker Talk

The main idea of this talk is “one size fits none”. This means that there is so much diversity the old style “One size fits all” is obsolete; OSFA was attempted to be achieved but it can not work.

There are many new techniques and ideas that are useful, but are only efficient when used situationally and specific to certain needs.

There is currently a movement in this market towards column stores, which can be more efficient than row stores.

Advantages and Disadvantages

Advantages:

- The chunks and consistency in chunk sizes regulates the system and provides efficiency within the system.
- The GFS system has fast recovery.

Disadvantages:

- The GFS has a high failure rate.
- The GFS is not likely very adaptable, and wouldn't work well in most situations other than the one in which it is used. Classic relational DBMS would work better for most situations.