The Google File System

Weon Yuan

CMPT 308 – Database Management

November 22, 2013

Ghemawat, Sanjay, Gobioff, Howard and Leung, Shun-Tak. "The Google File System." Google. 2003.

Main Ideas about the Google File System

- Scalable distributed proprietary file system by Google
  - Separate computers are connected together by a network
  - Using large clusters of commodity hardware
- Designed to…
  - Meet Google's rapidly growing demands
    - Data storage and usage needs
  - Be extremely fault-tolerant
  - Handle and process multi-TB data
    - Having files be read and written by multiple clients at the same time
    - Organize and manage all of the files that have to be dealt daily
  - Favor high bandwidth over low latency

Implementations

- GFS's architecture is composed of:
  - A (single) master and multiple chunkservers
  - Accessed by multiple clients
- Master handles all metadata and the operations log in memory
  - Makes chunk placement and replication decisions using global knowledge
  - Stores 3 types of metadata:
    - File and chunk namespaces, mapping from files to chunks, and locations of chunk's replicas
  - Files will never be overwritten (new info are appended to the file)
  - Operation log: a logical timeline that defines the order concurrent operations; only persistent record of metadata
- Divides files into 64 MB chunks
  - Other file system chunk sizes can range from 8KB to 1MB
- Utilizes lazy space allocation
  - Chunks are allocated only when necessary

My Analysis about GFS

- Commodity hardware
  - Who knew that you could use inexpensive components to power the widely-used search engine and one of the most visited website in the world?
  - Cost-effective
    - When scaling GFS, Google can continue to add more inexpensive Linux servers for more storage
  - Linux FTW (enough said)
- Having one master
  - I do agree with Google that having one master in a cluster simplifies the design a lot
  - Allows consistency through its respective chunkservers
  - Although having one master may sound risky, Google does a great job of recovering itself when it goes down
    - Keeping a history of critical metadata changes and replicating each chunk in different chunkservers
    - Both master and its chunkservers are designed to recover itself in seconds “no matter how they [are] terminated”

Advantages

- Optimized for large-scale data processing workloads
  - Larger chunk size
    - Reduces clients' need to interact with the master
    - Reduces network overhead
    - Reduces the size of the metadata stored on the master
  - Metadata is stored in the master's memory
    - Easy and efficient for the master to periodically scan through its entire state in the background
- Redundant
  - By default, chunkservers store three replicas of every chunk
  - Provides higher fault tolerance
- Inexpensive
  - Uses commodity hardware (e.g. off-the-shelf Linux servers)
  - Does not resort to purchasing new hardware or components

Disadvantages

- Not optimized for small-scale data processing workloads
  - Still handles small files and small (random) reads and writes
- Single master may still be a potential bottleneck
  - Practically everything is reliant on the single master
  - If the master faces downtime, so will the entire cluster
- Redundant
  - Wastes disk space
  - All replicas may not be up-to-date
  - Older versions of a file are not deleted = greater file size

Real-world use cases

- Widely used as Google's storage platform
  - For research and development
  - For production data processing
  - Principle: hide data when needed (never delete) and append data (never overwrite)
- Google Docs
  - Enables multiple users to work on a certain file concurrently
  - Keeps a history of revisions made by users
- Succeeded by a new GFS design, called Colossus
  - Updates search index continuously
  - Enables real-time searches