CSE221 Lecture 18

Aronya Baksy

December 2024

1 Google File System

1.1 Goals

- Very large files (100 MB - GB), millions of files
- Tolerant to failure: application bugs, operating system bugs, human errors, and the failures of disks, memory, connectors, networking, and power supplies
- Appending new data rather than overwriting existing data, throughput more important than latency
- Modifying application design to work with new file system APIs
- Concurrent writes must be supported

1.2 GFS Architecture

- GFS is a **user-space file system**
- File split into 64MB chunks, each chunk has a unique and immutable 64 bit **chunk handle**, each chunk is replicated (default 3, can be configured)
- Chunks are **versioned**, master has all information about versions
- **Chunkservers** store chunks on local disks as Linux files and read or write chunkdata specified by a chunkhandle and byte range
- GFS does not have a per-directory data structure that lists all the files in that directory (only single global lookup table maps absolute paths to metadata which can be compressed using prefixes)
- **Master**: maintains all metadata, answers clients with location and offset of a particular file address (clients then perform reads from chunkservers).
- For **write**: client contacts master, master picks chunkservers to store new chunk and its replicas, client writes to closest chunkserver, chunkserver propagates chunks to replicas on its own, client sends **commit msg** to chunkserver which is propagated
- **Client**: communicate with master and chunk server using GFS client library
- **No caching**: no benefit for client because mainly streaming access patterns, and rely on underlying FS caching for chunkservers

1.3 Append

- POSIX API is tough for concurrent writers (offset calcuation is messed up)
- Primary chunkserver will order all the record appends, and returns offset
- append-at-least-once semantics preserves each writer's output

1.4 Summary

- Distributed file system made using commodity hardware
- Exposes application inconsistencies, which applications are expected to handle
- Challenges: scale, consistency, failures