Large Scale Storage Systems

CSE 550: Systems for All
Autumn 2022

Lequn Chen

Chord


LEFT
EXIT 12
Peer-to-peer Systems
Large Scale Storage Systems
CSE550 Au22

Storage Systems

“Read this before you start another storage system project.” – BigTech Intranet

There are so many storage systems already.

Why do we need different storage systems?

How about keep everything on a file system?

- e.g. Maildir

Cse550a_au22 list run by *ratul at u.washington.edu*
Cse550a_au22 administrative interface (requires authorization)
Overview of all mailman11.u.washington.edu mailing lists

DELIVERED BY MAILMAN
version 2.1.17

UW-IT
help@u.washington.edu

*mailman2 uses database instead of raw filesystem

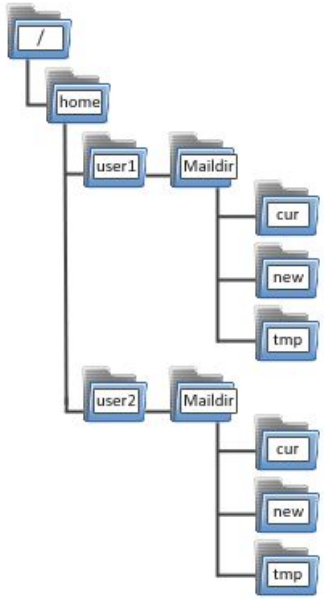

Image: Wiki: Maildir

Examples of Data Store

- File System
- Network File System
- Chunk-based file system
- Object storage
- Key-value Database (“key-value pair”)
- Relational Database (“table”)
- Document Database (“nested dicts”, “json”)
- Time-series Database (“vector”)
- Graph Database
- Blockchain
- Distributed hashing table

- Google File System / HDFS / Ceph
- Amazon S3
- Memcached / Redis
- Spanner / Postgres / MySQL
- MongoDB / Dynamo / BigTable
- InfluxDB
- F1 / Dremel
- Chubby / ZooKeeper / Etcd
- Chord
- DNS (Domain Name System)

Challenges on Large Storage Systems

- Scalability
  - “The file system” itself needs to span across multiple machines
  - 1 machine / 10s, 100s, 1000s of machines
  - Same network switch; Same datacenter; Across datacenter; Across region; Globe
  - 1 user / 10, 1k, 1m users
  - Byte, KB, MB, GB, TB, PB, EB, ZB, YB
- Replication / Availability / Redundancy
- Storage Media
  - Memory, Hard Drive, SSD, Tape
  - Read/Write performance
  - Random access performance
  - Durability (Data Loss)
- Semantics
  - POSIX File System :(
  - Transaction, Atomicity
- Workload
  - Read/Write ratio
  - Append-only vs. Random access
- Consistency
- Permission Control
  - Who can join the system
  - Who can read/write
  - Who can decide event ordering
- Security and Data Safety (Attack Model)
- …

What are some dimensions to categorize a storage system?

Dimension: Structure

- Unstructured data
  - Key-value store (e.g., Memcached, Redis)
  - Object store (e.g., Amazon S3) “object” means big blobs (e.g., images, files)
  - Hierarchical namespace: file systems, DNS, object store
  - Flat namespace: memcached, redis, DHT (Chord)
- Structured data: Table
  - Postgres / MySQL
  - Spanner
  - BigTable
  - F1 / Dremel
- Structured data: Graph
  - Facebook TAO

Dimension: Durability

- Caching
  - Memcached, Redis
  - Focus on in-memory performance
- Persistent data
  - Dynamo
  - Databases…
  - Write to disk
  - You don’t want to lose data
    - Write-ahead log
    - fsync semantics
    - “completion” signal

Dimension: Consistency

- Strong consistency
  - Lock service; Cluster management metadata
  - Chubby / ZooKeeper / Etcd
  - Slower performance; Easier to reason about (easier to use)
- Eventual consistency
  - Web crawler; Shopping cart; Social media profile
  - Dynamo / BigTable
  - Higher performance; Harder to program correctly
- Transactional consistency
  - Relational Database

Dimension: Cluster Size

- 1 machine
  - Filesystem
  - SQLite / Postgres
- A few machines
  - NFS
  - Postgres
  - Fault tolerance; Read optimization;
  - Primary-backup; Replication;
- 100s / 1000s of machines
  - Spanner
  - Throughput scalability
  - Replication + Sharding
- Many machines but they don’t chat with each other
  - Redis, Memcached
  - Clients need to know the list of servers
  - Probably some other systems tell a client which server to talk to. e.g., Google’s Slicer

Dimension: Abstract Level

- Chunk
  - Google File System / HDFS
  - Optimized for write, append, big chunk of data.
- Key-value Store
  - Random access
  - Key locality
- Filesystem
  - NFS
  - POSIX file system semantics; Compatible with existing software
- Database
  - Transaction semantics
  - Analytical performance
- Can be built upon each other

Dimension: Data Size

- Metadata (B~KB)
  - DHT (Chord)
  - Chubby / ZooKeeper / Etcd
  - Correctness
- Big chunks (MB~TB)
  - Object store (Amazon S3)
  - Google file system / HDFS
  - Network bandwidth; Pipelining; Caching (e.g., CDN)
- Somewhere in between
  - Databases
  - Transaction semantics
  - Requests per second

Dimension: Read/Write Workload

- OLTP (online transactional processing)
  - INSERT / UPDATE / DELETE / BEGIN TRANSACTION
  - Write tiny bit of data; Simple transactions
  - Latency; Concurrency; Availability; Atomicity; Consistency; Isolation; Durability
  - Spanner
- OLAP (online analytical processing)
  - SELECT
  - Read a huge amount of data
  - Throughput
  - F1
- Column-based storage
  - Time-series DB
  - Vectorized computation (e.g., add, max)
  - Data compression (e.g., store difference)

Dimension: Permission Control

- Peer-to-peer
  - DHT (Chord) / BitTorrent / IPFS
  - Free to join; Free to leave; Everyone can read/write; Might have bad actors
  - How to distribute information effectively and efficiently
  - How to tolerate stale or bad information
- Centralized
  - Databases
  - Under the same administrative domain.
  - Vulnerable to hackers. Need to harden security.
- Permissionless blockchain
  - Bitcoin / Ethereum
  - Free to join; Everyone can send txs;
  - Agree on ordering; Everyone can take part in ordering
- Permissioned blockchain
  - Binance Smart Chain
  - Everyone can send txs
  - Only a few validators can decide the ordering (hence “permissioned”)