Lecture Notes 5 (4 was guest)

HDFS

HDFS: How are Files stored/managed?

- Data files split into contiguous chunks typically 64MB size distributed at load time
- Each chunk replicated on multiple data nodes
- Name node for a file stores metadata, location o all chunks, etc.
- Optimized for large, streaming reads of files (rather than random reads)
- Files are "write once" - no random writes to fiels allowed - because HDFS batch roots, it was only designed to handle append-only formats

HDFS: Reading Data

HDFS: Writing Data

- Get a Lease -> Write Data -> Close the lease
- Getting Lease
  - The client sends a request to the NameNode to create a new file
  - The NameNode determines how many blocks are needed, and the client is granted a lease for creating these new file blocks in the cluster.
  - Advantage - You can have snapshots, ie revert back your changes

- Write Data
  - Write the data
  - NameNode stores the logs of what happened

- Close Lease
  - DataNode daemons acknowledge the file block replicas have been created,
  - Client application closes the file and notifies the NameNode
  - NameNode closes the open lease. Updates become visible at this point

HDFS: Some Limitations

- **Not appropriate for real-time, low-latency processing** - have to close the file immediately after writing to make data visible, hence a real time task would be forced to create too many files
- **Centralized metadata storage** - multiple single points of failure
- **The Persistence of File System Metadata** - Get the log file from the NameNode and then replay the operations

Hadoop Database (HBASE)

Intro

- A NoSQL db build on HDFS
- A table can have thousands of columns
- Supports very large amounts of data and high throughput
- HBase -> Strong Consistency
- Random access, low latency

RDMS vs NoSQL

- BASE not ACID
- By giving up ACID constraints, you improve availability and performance

HBase Data Model

Column Family

- Data (column families) stored in separate files (Hfiles)
- Tune Performance
  - In-memory
  - Compression

- Needs to be separated by the user

Horizontal Splits - Sharding

HBase Architecture

- Composed of 3 server types in Master-Slave format
  - Region Server
  - Hbase Master

    - ZooKeeper

- Region Server
    - Clients communicate with RegionServers (slaves) directly for accessing data
    - Serves data for reads and writes
    - These region servers are assigned to the HDFS data nodes to preserve data locality

HBase: How do these components work together?

- Region server and the active HBase Master connect with a session to Zookeeper
- A special HBase catalog table "META table" -> Holds the location of the regions in the cluster
- Zookeeper stores the location of the META table

HBase: Reads/Writes

- The client gets the region server that hosts the META table from Zookeeper
- The client will query (get/put) the META server to get the region server corresponding to rowkey it wants to access
- It will get the Row from the corresponding Region Server

HBase: Some Limitations

- Not ideal for large objects, ie videos - problem is **write amplification**, when HDFS reorganizes data to compact large unchanging data, extensive copying occurs
- Not ideal for store data chronologically (time as primary index), ie machine logs organized by time-stamps cause write hot-spots

Midterm Exam Feedback

- Zookeeper - configuration management, communication, etc.
- Challenges in Distrubted Computing
    - Node failures
    - Programming complexity
    - Network failure

Design Question

- What are the resources in the system?
    - Assume the file system and assume the requirements

  - Requirements gathering, if not specified, you should assume

- Algorithms
  - Use a minheap, or use sorting if file fits in memory
  - You can use the same algorithm, but you have to load chunk by chunk. Use a hashtable for parsing and a minheap for calculating

- If file cannot fit into a node
  - You can use several machines to bring in the data
  - Several other machines to do the actual computation
  - Distribute the heaping algorithm
  - Talk about tradeoffs
  - Replication
  - Moving data and chunks of data to each nodes