EC/CS 528: Cloud Computing

Distributed Systems for the Cloud

Instructor: Alan Liu
Announcements

• The first sprint demo video will be due on Monday.
  • Questions?

• Project check-ins: Each team sends me a date of next meeting with the mentor(s). I will try to coordinate.

• Lecture (Wed 10/5): Ata Turk (State Street Financial)

• Talk (Wed 10/12): Wei Bai (Microsoft Research)
Announcements

• Quick team update: project description, sprint plan, etc.

• Cloud resource allocations.

• OpenStack quick demo (later).
GFS and MapReduce
Motivation

- Huge amounts of data to store and process
- Example @2004:
  
  20+ billion web pages x 20KB/page = 400+ TB
  
  Reading from one disc 30-35 MB/s
  
  Four months just to read the web
  
  1000 hard drives just to store the web
  
  Even more complicated if we want to process data

- Exp. growth. The solution should be scalable
Motivation

• Buy super fast, ultra reliable hardware?
  • Ultra expensive
  • Controlled by third party
  • Internals can be hidden and proprietary
  • Hard to predict scalability
  • Fails less often, but still fails!
  • No suitable solution on the market
Motivation

• Use commodity hardware? Benefits:
  Commodity machines offer much better perf/$
  Full control and understanding of internals
  Can be highly optimized for their workloads

• Not that easy:
  Fault tolerance: something breaks all the time
  Applications development
  Debugging, Optimization, Locality
  Communication and coordination
  Status reporting, monitoring

• Handle all these issues for every problem you want to solve
How to structure large distributed storage systems?

• An always-interesting research and engineering question. Why?
  • Other requirements: latency, energy consumption, cost.
  • Emerging applications

Figure 1. FAWN-KV architecture.
Typical first year for a new Google cluster (circa 2006)

~1 network rewiring (rolling ~5% of machines down over 2-day span)
~20 rack failures (40-80 machines instantly disappear, 1-6 hours to get back)
~5 racks go wonky (40-80 machines see 50% packetloss)
~8 network maintainances (4 might cause ~30-min random connectivity losses)
~12 router reloads (takes out DNS and external vips for a couple minutes)
~3 router failures (have to immediately pull traffic for an hour)
~dozens of minor 30-second blips for DNS
~1000 individual machine failures
~thousands of hard drive failures
slow disks, bad memory, misconfigured machines, flaky machines, etc.

Long distance links: wild dogs, sharks, dead horses, drunken hunters, etc.

Reliability Must Come From Software
A Series of Steps, All With Common Theme:

Provide Higher-Level View Than “Large Collection of Individual Machines”

Self-manage and self-repair as much as possible
First Step:
Abstract Away Individual Disks
The Google File System
S. Ghemawat, H. Gobioff, S. Leung. SOSP, 2003

Assumptions:
• Inexpensive commodity hardware -> need for fault tolerance / recovery
• Workload (e.g., crawling -> indexing -> PR -> ...)
  • Multiple clients
  • Large streaming reads, Small random writes
  • Concurrent appends to the same file
• High Throughput > Low Latency
Architecture
Architecture

- User-level process running on commodity Linux machines
- Consists of Master Server and Chunk Servers
- Files broken into chunks (typically 64 MB),
- 3x redundancy
- Data transfers happen directly between clients and Chunk Servers
Master Node

- Centralization for simplicity & global knowledge for chunk placement/replication
- Namespace and metadata management
- Managing chunks
  - Where they are (file<-chunks, replicas)
  - Where to put new
  - When to re-replicate (failure, load-balancing)
  - When and what to delete (garbage collection)
- Fault tolerance
  - Shadow masters
  - Monitoring infrastructure outside of GFS
  - Periodic snapshots
  - Mirrored operations log
Master Node

- All Metadata is kept in Master’s memory – it’s fast!
  - A 64 MB chunk needs less than 64B metadata => for 640 TB less than 640MB
- Master learns ChunkServer-to-chunk mapping from Chunk Servers when
  - Master starts
  - A Chunk Server joins the cluster
- Master exchanges periodic heartbeat with Chunk Servers
  - state monitoring & instructions
- Operation log to keep file-to-chunk mapping persistent
  - Is used for serialization of concurrent operations
  - Replicated in master’s disk and on remote machines
  - Respond to client only when log is flushed locally and remotely
Chunk Servers

• 64MB chunks as Linux files
  • Reduce size of the master data structures
  • Reduce client-master interaction
  • Internal fragmentation => allocate space lazily

• Fault tolerance
  • Heart-beat to the master
  • Something wrong => master inits replication
Append Control and Data Flow

1. Master receives write request.
2. Master sends push data to replicas.
3. Client sends ACK/ERR.
4. Client sends write request.
5. Primary Replica receives write request.
7. Secondary Replicas A, B acknowledge or error.

Legend:
- Control
- Data
Control & Dataflow

- Decouple control flow and data flow
- Control flow
  - Master -> Primary -> Secondaries
- Data flow
  - Carefully picked chain of Chunk Servers
    - Forward to the closest first
    - Distance estimated based on IP
Some other important notes

• Smart chunk creation policy
  • Chunk Servers with below-average disk utilization, limited # of recent, distribute chunks across racks

• Smart re-replication policy
  • Under replicated first
  • Chunks that are blocking client
  • Live files first (rather than deleted)

• Rebalance and Garbage Collect periodically
Some other important notes

• Fault tolerance at master
  • master state is replicated
    • operation logs and checkpoints are replicated
    • a mutation is considered committed after it is written to log and log replicas
    • shadow-masters for read availability
• Data integrity handled at chunk servers
  • 32 bit checksums in memory for each 64K block
  • realize corruption during read
  • restore from other chunk servers
Measurements (2003)

**Micro-benchmarks: GFS CLUSTER**
- 1 master, 2 master replicas, 16 chunkservers with 16 clients
- Dual 1.4 GHz PIII processors, 2GB RAM, 2x80GB 5400 rpm disks, FastEthernet NIC connected to one HP 2524 Ethernet switch 24 ports 10/100 + Gigabit uplink
Read measurements

- each client read random 4MB region 256 times (1GB) from 320GB file
Write measurements

- each client writes 1GB data to a new file in a series of 1MB writes

67MB/s (need to write 3 times)

~35MB/s => 50% for 16 clients
- collision is more likely (3 replicas)
Measurements (2003)

- **Real world clusters: STORAGE & METADATA**
  - Cluster A used regularly for R&D by +100 engineers
  - Cluster B used for production data processing

<table>
<thead>
<tr>
<th>Cluster</th>
<th>R&amp;D</th>
<th>Production</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>Chunkservers</td>
<td>342</td>
<td>227</td>
</tr>
<tr>
<td>Available disk space</td>
<td>72 TB</td>
<td>180 TB</td>
</tr>
<tr>
<td>Used disk space</td>
<td>55 TB</td>
<td>155 TB</td>
</tr>
<tr>
<td>Number of Files</td>
<td>735 k</td>
<td>737 k</td>
</tr>
<tr>
<td>Number of Dead files</td>
<td>22 k</td>
<td>232 k</td>
</tr>
<tr>
<td>Number of Chunks</td>
<td>992 k</td>
<td>1550 k</td>
</tr>
<tr>
<td>Metadata at chunkservers</td>
<td>13 GB</td>
<td>21 GB</td>
</tr>
<tr>
<td>Metadata at master</td>
<td>48 MB</td>
<td>60 MB</td>
</tr>
</tbody>
</table>

Cluster A store 55/3 = **18 TB** of data
Cluster B store 155/3 = **52 TB** of data

**Chunkservers metadata** = checksums for each 64 KB data block + chunk version number
Measurements (2003)

- **Real world clusters: READ/WRITE RATES & MASTER LOAD**
  - Cluster A used regularly for R&D by +100 engineers
  - Cluster B used for production data processing

<table>
<thead>
<tr>
<th>Cluster</th>
<th>R&amp;D</th>
<th>Production</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>Read rate (last minute)</td>
<td>583 MB/s</td>
<td>380 MB/s</td>
</tr>
<tr>
<td>Read rate (last hour)</td>
<td>562 MB/s</td>
<td>384 MB/s</td>
</tr>
<tr>
<td>Read rate (since restart)</td>
<td>589 MB/s</td>
<td>49 MB/s</td>
</tr>
<tr>
<td>Write rate (last minute)</td>
<td>1 MB/s</td>
<td>101 MB/s</td>
</tr>
<tr>
<td>Write rate (last hour)</td>
<td>2 MB/s</td>
<td>117 MB/s</td>
</tr>
<tr>
<td>Write rate (since restart)</td>
<td>25 MB/s</td>
<td>13 MB/s</td>
</tr>
<tr>
<td>Master ops (last minute)</td>
<td>325 Ops/s</td>
<td>533 Ops/s</td>
</tr>
<tr>
<td>Master ops (last hour)</td>
<td>381 Ops/s</td>
<td>518 Ops/s</td>
</tr>
<tr>
<td>Master ops (since restart)</td>
<td>202 Ops/s</td>
<td>347 Ops/s</td>
</tr>
</tbody>
</table>

Cluster A network configuration can support read rate of 750 MB/s
Cluster B network configuration can support read rate of 1300 MB/s
Measurements (2003)

- **Real world clusters: RECOVERY TIME**
  - **Kill 1 chunkserver in cluster B (Production)**
    - 15,000 chunks on it (= 600 GB of data)
    - Cluster limited to 91 concurrent chunk cloning (= 40% of 227 chunkservers)
    - Each clone operation consume at most 50 Mbps
    - **15,000 chunks restored in 23.2 minutes effective replication rate of 440 MB/s**
  - **Kill 2 chunkservers in cluster B (Production)**
    - 16,000 chunks on each (= 660 GB of data)
    - This double failure reduced 266 chunks to having a single replica...
    - **These 266 chunks cloned at higher priority and were all restored to a least 2xreplication within 2 minutes**
The Google File System

*in one slide*

Google File System (Ghemawat, Gobioff, & Leung, SOSP‘03)

- Centralized master manages metadata
- 1000s of clients read/write directly to/from 1000s of disk serving processes
- Files chunks of 64 MB, each replicated on 3 different servers
- High fault tolerance + automatic recovery, high availability

![Diagram of GFS file system clients](image)