Cassandra @ Spotify

Scaling storage to million of users world wide

Jimmy Mårdell <yarin@spotify.com>
About me

• Jimmy Mårdell
• Tech Product Owner in the Cassandra team
• 4 years at Spotify
Contents

• Understanding the problem
• Cassandra
• Future challenges
Understanding the problem
Understanding the problem

Data at Spotify

• Binary data files
  • MP3 files, cover art
• Static structured data
  • Updated once a day
  • Search indexes, personal music suggestions
• Ephemeral data
  • Caches, short-lived data
  • “What’s happening now”
• Read-write persistent structured data
  • User accounts, playlists, notifications, social network, radio stations etc
Relational databases

- Master-slave setups
  - Pros: more read capacity
  - Cons: same write capacity, replication delays, still SPoF
- Sharding
  - Pros: more read and write capacity
  - Cons: more complexity, data integrity issues, many more SPoF’s
- Both
  - All pros and cons of the above
Understanding the problem

Distributed K/V-stores

• More simple data model, but often good enough
• Easier to scale out
• Many options today:
  • Cassandra
  • HBase
  • MongoDB
  • Redis
  • Riak
Why is this a hard problem?

- Hardware failures
- Slow machines
- Network partitions
- Data consistency
- CAP-theorem
The “Dynamo” paper

- DynamoDB - Amazon’s K/V store
- Paper released in 2007
- Simple but powerful distribution mechanism
Cassandra
What is Cassandra?

- Open source distributed database (Apache project)
- More than just a K/V-store
- Built-in sharding
- Asynchronous masterless replication
  - No Single Point of Failure!
- Extremely robust
- Other prominent users: Apple, Netflix, Instagram
Cassandra at Spotify

- Began using it in 2010
- Now 500+ nodes across 50+ clusters
- First database choice when developing a new system
Cassandra architecture

- Combining two technologies:
  - Amazon “Dynamo”: How the data is replicated among nodes
  - Google “BigTable”: How data is stored locally on node
Cassandra

Data distribution

- Nodes are assigned integer tokens
- All nodes know about each other
  - Gossip
- hash(Key) determines location of data
- Replication to succeeding nodes
Write requests

- Client oblivious about tokens
- Coordinator node sends write to replica nodes
- Customizable consistency level
Read requests

- Coordinator sends read to some replica nodes
  - Consistency level
- Latest data wins
- Asynchronous synchronization
Consistency levels

• Determines how many replicas to read/write to
• ONE - fast, fault tolerant, least consistency guarantee
• ALL - slow, no fault tolerance, best consistency guarantee
• QUORUM - “majority”, good compromise
• Specified per request
# Consistency levels

<table>
<thead>
<tr>
<th></th>
<th>Read ONE, Write ALL</th>
<th>Read ALL, Write ONE</th>
<th>Read QUORUM, Write QUORUM</th>
<th>Read ONE, Write ONE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fault tolerant on write</td>
<td><strong>NO</strong></td>
<td><strong>YES</strong></td>
<td><strong>YES</strong></td>
<td><strong>YES</strong></td>
</tr>
<tr>
<td>Fault tolerant on read</td>
<td><strong>YES</strong></td>
<td><strong>NO</strong></td>
<td><strong>YES</strong></td>
<td><strong>YES</strong></td>
</tr>
<tr>
<td>Read always gets latest data</td>
<td><strong>YES</strong></td>
<td><strong>YES</strong></td>
<td><strong>YES</strong></td>
<td><strong>NO</strong></td>
</tr>
</tbody>
</table>
Fault tolerance

- Cluster typically fully operational with one node down
  - Depends on consistency levels
- Eventual consistency
  - Hinted Handoffs
  - Read-repair
  - Anti-Entropy repair
Multiple data centers

- One cluster - multiple rings
- All DC’s contain same data
Multi-DC consistency

- Network partitioning is a problem
- LOCAL_QUORUM and EACH_QUORUM
- Consistency within DC, not to other DC’s
  - Usually good enough
Future challenges
Future challenges

Cassandra automation tools

- Every feature uses its own Cassandra cluster
  - Decoupling is the key to scalability
- Tools to automate common operations
  - Setup new clusters
  - Upgrade clusters
  - Replace broken hardware
  - Schedule repairs and compactions
  - Backup/restore
Recovery after network split

• Important to ensure eventual consistency
  • Old data may resurface!
• Anti-entropy repair slow and potentially risky
  • Merkle-tree calculation
  • Low granularity causes overstreaming
• Causes more problems than one would think
Improving Cassandra

- Performance
  - Code
  - Configuration
  - GC tuning
- New features
  - vnodes
  - CQL
  - Light-weight transactions
  - Incremental repairs
Many data centers

- Cassandra scales linearly within a data center
- Not so well to many datacenters
  - “Waste” of machines
  - Repair an increasing problem
- Do we really need to store all data everywhere?
  - Geolocated data
Thanks for listening!

Questions?

Jimmy Mårdell, yarin@spotify.com