Realtime Apache Hadoop at Facebook

Dhruba Borthakur
Oct 5, 2011 at UC Berkeley Graduate Course
<table>
<thead>
<tr>
<th></th>
<th>Agenda</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Why Apache Hadoop and HBase?</td>
</tr>
<tr>
<td>2</td>
<td>Quick Introduction to Apache HBase</td>
</tr>
<tr>
<td>3</td>
<td>Applications of HBase at Facebook</td>
</tr>
</tbody>
</table>
Why HBase? For *Realtime* Data?
OS
Web server
Database
Language
Cache
Data analysis
Problems with existing stack

- MySQL is stable, but...
  - Not inherently distributed
  - Table size limits
  - Inflexible schema

- Hadoop is scalable, but...
  - MapReduce is slow and difficult
  - Does not support random writes
  - Poor support for random reads
Specialized solutions

- High-throughput, persistent key-value
  - Tokyo Cabinet
- Large scale data warehousing
  - Hive/Hadoop
- Photo Store
  - Haystack
- Custom C++ servers for lots of other stuff
What do we need in a data store?

- **Requirements for Facebook Messages**
  - Massive datasets, with large subsets of cold data
  - Elasticity and high availability
  - Strong consistency within a datacenter
  - Fault isolation, quick recovery from failures

- **Some non-requirements**
  - Network partitions within a single datacenter
  - Active-active serving from multiple datacenters
HBase satisfied our requirements

- In early 2010, engineers at FB compared DBs
  - Apache Cassandra, Apache HBase, Sharded MySQL

- Compared performance, scalability, and features
  - HBase gave excellent write performance, good reads
  - HBase already included many nice-to-have features
    - Atomic read-modify-write operations
    - Multiple shards per server
    - MapReduce tools: Bulk Import, Data Scrubber, etc
    - Quick recovery from node failures

HBase satisfied our requirements
HBase uses HDFS

We get the benefits of HDFS as a storage system for free

- Fault tolerance, quick recovery from node failures
- Scalability
- Checksums fix corruptions
- MapReduce

- Fault isolation of disks
- HDFS battle tested at petabyte scale at Facebook

Lots of existing operational experience
Apache HBase

- Originally part of Hadoop
  - HBase adds random read/write access to HDFS

- Required some Hadoop changes for FB usage
  - File appends
  - HA NameNode
  - Read optimizations

- Plus ZooKeeper!
HBase System Overview

Database Layer

- Master
- Backup Master
- Region Server
- Region Server
- Region Server

Storage Layer

- Namenode
- Secondary Namenode
- Datanode
- Datanode
- Datanode

Coordination Service

- ZK Peer
- ZK Peer
- Zookeeper Quorum

HDFS

HBASE
HBase in a nutshell

- Sorted and column-oriented
- High write throughput
- Horizontal scalability
- Automatic failover
- Regions sharded dynamically
Applications of HBase at Facebook
Use Case 1

Titan

(Facebook Messages)
The New Facebook Messages

Messages    IM/Chat    email    SMS
Facebook Messaging

- **High write throughput**: Every message, instant message, SMS, and e-mail. Search indexes for all of the above
- **Denormalized schema**

- **A product at massive scale on day one**
  - 6k messages a second
  - 50k instant messages a second
  - 300TB data growth/month compressed
Typical Cell Layout

- Multiple cells for messaging
- 20 servers/rack; 5 or more racks per cluster
- Controllers (master/Zookeeper) spread across racks
High Write Throughput

Write Key Value

Key val
Key val
Key val
Key val
Key val
Key val

Commit Log (in HDFS)

Sequential write

Sorted in memory

Memstore (in memory)

Sequential write

Key val
Key val
Key val
Key val
Key val
Key val

Sorted in memory
Horizontal Scalability

Region

On click bottom one of first two on the left move over to be added to the third box two clicks one by one.
Automatic Failover

HBase client

Find new server from META

server died

No physical data copy because data is in HDFS
Use Case 2
Puma
(Facebook Insights)
Before Puma
Traditional ETL with Hadoop

Web Tier -> Scribe -> HDFS -> MR -> Hive -> SQL -> MySQL

15 min - 24 hours
Puma
Realtime ETL with HBase

10-30 seconds
Puma

- **Realtime Data Pipeline**
  - Utilize existing log aggregation pipeline (Scribe-HDFS)
  - Extend low-latency capabilities of HDFS (Sync+PTail)
  - High-throughput writes (HBase)

- **Support for Realtime Aggregation**
  - Utilize HBase atomic increments to maintain roll-ups
  - Complex HBase schemas for unique-user calculations
  - Store checkpoint information directly in HBase
Puma as Realtime MapReduce

- Map phase with PTail
  - Divide the input log stream into N shards
  - First version only supported random bucketing
  - Now supports application-level bucketing

- Reduce phase with HBase
  - Every row+column in HBase is an output key
  - Aggregate key counts using atomic counters
  - Can also maintain per-key lists or other structures
Puma for Facebook Insights

- Realtime URL/Domain Insights
  - Domain owners can see deep analytics for their site
  - Clicks, Likes, Shares, Comments, Impressions
  - Detailed demographic breakdowns (anonymized)
  - Top URLs calculated per-domain and globally

- Massive Throughput
  - Billions of URLs
  - > 1 Million counter increments per second
Site Engagement

Total Actions

118x

Distribution on Facebook

Total Impressions

0.11%

Referral Traffic to Site

Total Clicks

Site Engagement

- Total Actions
- Likes
- Shares
- Comments
<table>
<thead>
<tr>
<th>Age</th>
<th>Female (31%)</th>
<th>Male (62%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>13-17</td>
<td>7.9%</td>
<td>13%</td>
</tr>
<tr>
<td>18-24</td>
<td>8.5%</td>
<td>22%</td>
</tr>
<tr>
<td>25-34</td>
<td>6.6%</td>
<td>16%</td>
</tr>
<tr>
<td>35-44</td>
<td>3.9%</td>
<td>6.3%</td>
</tr>
<tr>
<td>45-54</td>
<td>2.3%</td>
<td>2.8%</td>
</tr>
<tr>
<td>55+</td>
<td>1.5%</td>
<td>2.3%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Language</th>
<th>(Country)</th>
</tr>
</thead>
<tbody>
<tr>
<td>English (US)</td>
<td>(34%)</td>
</tr>
<tr>
<td>Spanish</td>
<td>(12%)</td>
</tr>
<tr>
<td>Arabic</td>
<td>(6.1%)</td>
</tr>
<tr>
<td>Turkish</td>
<td>(6%)</td>
</tr>
<tr>
<td>English (UK)</td>
<td>(5.3%)</td>
</tr>
<tr>
<td>French (France)</td>
<td>(5.2%)</td>
</tr>
<tr>
<td>Spanish (Spain)</td>
<td>(4.6%)</td>
</tr>
<tr>
<td>Indonesian</td>
<td>(3.9%)</td>
</tr>
<tr>
<td>Italian</td>
<td>(3.7%)</td>
</tr>
<tr>
<td>German</td>
<td>(2.7%)</td>
</tr>
<tr>
<td>Portuguese (Brazil)</td>
<td>(2.6%)</td>
</tr>
<tr>
<td>Thai</td>
<td>(1.2%)</td>
</tr>
<tr>
<td>Russian</td>
<td>(0.96%)</td>
</tr>
<tr>
<td>Polish</td>
<td>(0.91%)</td>
</tr>
<tr>
<td>More</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Country</th>
<th>(Language)</th>
</tr>
</thead>
<tbody>
<tr>
<td>United States</td>
<td>(English)</td>
</tr>
<tr>
<td>Turkey</td>
<td>(Spanish)</td>
</tr>
<tr>
<td>Philippines</td>
<td>(Spanish)</td>
</tr>
<tr>
<td>Indonesia</td>
<td>(Indonesian)</td>
</tr>
<tr>
<td>India</td>
<td>(Italian)</td>
</tr>
<tr>
<td>Mexico</td>
<td>(Spanish)</td>
</tr>
<tr>
<td>Egypt</td>
<td>(Spanish)</td>
</tr>
<tr>
<td>Italy</td>
<td>(Italian)</td>
</tr>
<tr>
<td>Brazil</td>
<td>(Portuguese)</td>
</tr>
<tr>
<td>Argentina</td>
<td>(Spanish)</td>
</tr>
<tr>
<td>Germany</td>
<td>(German)</td>
</tr>
<tr>
<td>Malaysia</td>
<td>(Spanish)</td>
</tr>
<tr>
<td>United Kingdom</td>
<td>(Spanish)</td>
</tr>
<tr>
<td>Spain</td>
<td>(Spanish)</td>
</tr>
<tr>
<td>More</td>
<td></td>
</tr>
</tbody>
</table>
Use Case 3

ODS

(Facebook Internal Metrics)
**ODS**

- **Operational Data Store**
  - System metrics (CPU, Memory, IO, Network)
  - Application metrics (Web, DB, Caches)
  - Facebook metrics (Usage, Revenue)
    - Easily graph this data over time
    - Supports complex aggregation, transformations, etc.

- **Difficult to scale with MySQL**
  - Millions of unique time-series with billions of points
  - Irregular data growth patterns
Dynamic sharding of regions

Region

server overloaded
Future of HBase at Facebook
User and Graph Data in HBase
Looking at HBase to augment MySQL

- Only single row ACID from MySQL is used
- DBs are always fronted by an in-memory cache
- HBase is great at storing dictionaries and lists

Database tier size determined by IOPS

- HBase does only sequential writes
- Lower IOPs translate to lower cost
- Larger tables on denser, cheaper, commodity nodes
Conclusion

- Publish Workload Benchmark (work-in-progress)
  - http://hadoopblog.blogspot.com
  - http://www.facebook.com/hadoopfs
- Much more detail in SIGMOD 2011 paper
  - Technical details about changes to Hadoop and HBase
  - Operational experiences in production
Questions?

dhruba@fb.com