Apache Hadoop Goes Realtime at Facebook

~

Borthakur, Sarma, Gray, Muthukkaruppan, Spiegelberg, Kuang, Ranganathan, Molkov, Menon, Rash, Schmidt and Aiyer
Problem and Context

- Ever increasing data at Facebook
- Launch of *Facebook Messages*
- Other Young Turks at Facebook
- Leaving MySQL and its sharding
- Migration challenges
- Problem in words: Unpredictable growth, write throughput and latency requirements
Problem and Context (contd.)

- The Usual Suspects
  - Cassandra
  - Other NoSQL
- Other considerations
- Solution: A near realtime Hadoop/HBase that is modified from the vanilla versions to provide scalability, consistency, availability and a compatible data model.
Key contributions

● Making Hadoop and HBase more real-time
● Adapting Hadoop and HBase to Facebook's unique requirements
  ● Implementation of RealTime HDFS
  ● Implementation of Production HBase
  ● Operational Optimizations
Overview

- Problem and Context
- Facebook stands alone
- (Small) Introduction to Hadoop and HBase
- Enter the Hs
- Realtime HDFS
- Production HBase
- Operational Optimization
- The present future
Facebook's unique requirements

- Facebook and the Hadoop ecosystem
  - Offline and sequential

- Requirement Type 1 – Realtime concurrent read access to large stream of realtime data
  - Example: Scribe

- Requirement Type 2 - Dynamically index a rapidly growing data set for fast random lookups
  - Example: Facebook Messages
Facebook's unique requirements

- Facebook Messaging:
  - Unweildly tables
  - High Write Throughput
  - Data Migration

- Facebook Insights
  - Realtime Analytics
  - Aggregators

- Facebook Metrics System
  - Quick reads
  - Automatic Sharding
Overview

- Problem and Context
- Facebook stands alone
- (Small) Introduction to Hadoop and HBase
- Enter the Hs
- Realtime HDFS
- Production Hbase
- Operational Optimization
- The present future
Introduction to Hadoop
Introduction to HBase

- **Hbase**: A NoSQL database that utilizes an on-disk column storage format.
- **Hbase USP**: Provides fast key-based access to a specific cell or data or a range of cells.
- Based on Google's BigTable but extends it
- Has Row atomicity and read-modify-write consistency
- Simplifies a lot of tasks related to distributed databases.
- **Tagline**: *Random access to web-scale data*
Introduction to Zookeeper

- **Zookeeper**: A software service for a distributed environment that coordinates and configures different machines in a centralized way.

- A change is not considered successful until it has been written to a quorum

- A leader is elected within the ensemble for conflicts

- In HBase, ZooKeeper coordinates and shares state between the Masters and RegionServers.

- **Tagline**: Enables highly reliable distributed coordination
HBase + Zookeeper
Overview

- Problem and Context
- Facebook stands alone
- (Small) Introduction to Hadoop and HBase
- **Enter the Hs**
- Realtime HDFS
- Production HBase
- Operational Optimization
- The present future
The Why Hadoop/HBase question

- Scalability
- Range Scans
- Efficient low-latency strong consistency
- Atomic Read-Modify-Write
- Random reads
- Fault Isolation
The Why Hadoop/HBase question

- High write throughput
- Data model
- High Availability

- **Non-requirements**
  - Tolerance of network partitions
  - Individual data centre failure zero downtime
  - Federation comfort
Overview

• Problem and Context
• Facebook stands alone
• (Small) Introduction to Hadoop and HBase
• Enter the Hs
• **Realtime HDFS**
• Production HBase
• Operational Optimization
• The present future
Realtime HDFS - AvatarNode
Realtime HDFS - AvatarNode
Realtime HDFS – AvatarNode view
Realtime HDFS – Logging

• Enhancements to Transaction logging:
  • Conventional HDFS
  • Change: Let the StandbyNode always know about block ids.
  • Avoidance of partial reads between Active and Standby node
Improved block availability

**Challenge:** Placement of non-local blocks is not optimal; can be on any rack or within any node therein.

**Solution:** A new block placement policy which has reduced the probability of data loss by orders of magnitude.

- Define a 'window' of logical racks and logical machines around the original block.
Hadoop performance improvements

- RPC Timeout
  - *Live free or fail fast*
- File Lease recovery
- Local replica awareness
- New tricks:
  - HDFS sync
  - Concurrent readers
Overview

- Problem and Context
- Facebook stands alone
- (Small) Introduction to Hadoop and HBase
- Enter the Hs
- Realtime HDFS
- Production HBase
- Operational Optimization
- The present future
HBase – ACID compliance

• **Requirement:** Row-level atomicity and consistency of ACID compliance
  • *RegionServer failure during log write for row transactions.*
  • *Consistency of replicas*

• **Solution:**
  • WAL edits ~ Write Ahead Log policy
  • Immediate rollback
HBase – Availability Improvements

- Master Rewrite
  - Store transient state in Zookeeper
- Rolling upgrades
  - Handled by reassigning of regions
- Distributed Logsplitting
  - Outsource to Zookeeper
Hbase – Performance Improvement

- Compaction Improvement
  - *put latency dropped from 25 ms to 3 ms!*

- Read Optimization – Skipping certain unnecessary files for certain queries, reducing I/O
  - Using Bloom filters
  - A new special timestamp file selection algorithm

- Ensuring that Regions are local to their data
Overview

- Problem and Context
- Facebook stands alone
- (Small) Introduction to Hadoop and HBase
- Enter the Hs
- Realtime HDFS
- Production Hbase
- **Operational Optimization**
- The present future
Operational Optimizations

- Facebook's HBase testing program
- HBase Verify
- HBCK
- Added metrics for long running operations too!
- Manual split instead of automatic
Operational Optimizations

- Dark Launch
- Dashboard/ODS integration
  - Cross-cluster dashboards for higher analysis
  - Visualize version differences
- Backups
  - Do it using Scribe as an alternate application log
  - Piggyback on the date sent for Hive analytics
Operational Optimizations

• Importing the data
  • **Challenge**: Importing legacy data in HBase from a Hadoop job saturates the production network
  • **Solution**: Use Bulk Import with compression
    – Enhanced by GZIP of the intermediate map output

• Reducing Network I/O:
  • Decreased the periodicity of major compactions
  • Certain column families excluded from logging
The present future

• Apache Hadoop 2.0 was released in 2012
• One addition was YARN
  • A powerful cluster resource management
  • Added the High Availability feature to NameNode by introducing the **Hot/Standby NameNode**.
  • Greater integration with Zookeeper, especially for the ZKFC (Implementation of failover in DAFS)
Thank you and GG!