Apache HBase Features for the Enterprise

Jonathan Hsieh | @jmhsieh
Software Engineer at Cloudera / HBase PMC Member
October 2012
Who Am I?

- Cloudera:
  - Software Engineer
  - Apache HBase committer / PMC
  - Apache Flume founder / PMC
  - Apache Sqoop committer / PMC

- U of Washington:
  - Research in Distributed Systems
What is Apache HBase?

Apache HBase is an open source, distributed, scalable, consistent, low latency, random access non-relational database built on Apache Hadoop.
HBase provides Low-latency Random Access

- **Writes:**
  - 1-3ms, 1k-10k writes/sec per node

- **Reads:**
  - 0-3ms cached, 10-30ms disk
  - 10-40k reads / second / node from cache

- **Cell size:**
  - 0-3MB preferred

- Read, write and insert data anywhere in the table
  - No sequential write limitations
HBase On a Cluster

- HDFS NameNodes
- HBase Masters
- ZooKeeper
- Quorum
- Slave Boxes (DN + RS)

Rack 1
- Name node

Rack 2
- Name node
Production Apache HBase Applications

- Inbox
- Storage
- Web
- Search
- Analytics
- Monitoring

Production Systems Need to Avoid Risk

- Unfortunately, all things can fail.
- Enterprises need to minimize risk.
  - Understand potential data loss scenarios
  - Understand potential unavailability scenarios
  - Must have a disaster recovery story
- Downtime, data loss == risk
- Let’s talk about how HBase deals with:
  - Risks from within the cluster
  - Risks from outside the cluster
  - Risks posed by Users
- Goal: Remove or reduce negative impact of potential risks
Expect the best, plan for the worst, and prepare to be surprised.
Risks from within the cluster

Hosts and Services
Causes of HBase Downtime within the cluster

- Unplanned Maintenance
  - Hardware failures
  - Software errors
  - Human error

- Planned Maintenance
  - Upgrades
  - Migrations

Goal: Reduce downtime from hours to minutes to seconds.
Unplanned Downtime

- Two sources of unavailability
  - Detection time
  - Recovery time

Detection Time
- Service realizes there is a problem starts fixing.
- Service still thinks we are ok

Recovery Time
- Service is restored.

Failure Event
Reduce downtime by speeding up recovery time

- Distributed log splitting (0.92)
- Automated metadata repairs with hbck (0.92)
- Enable of writes while recovering from failure (0.96)
Reduce downtime by speeding up detection

- Proactively notify to recover from process failures quickly

<table>
<thead>
<tr>
<th>Time</th>
<th>Event Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.92/0.94</td>
<td>All Master Failure Detection 180s</td>
</tr>
<tr>
<td>0.94</td>
<td>Some Region Server Failure detection 180s</td>
</tr>
<tr>
<td>0.96</td>
<td>Master process failure detection 0-1s</td>
</tr>
<tr>
<td>0-1s</td>
<td>Region Server Failure detection 0-1s</td>
</tr>
</tbody>
</table>

Service still thinks we are ok
Service realizes there is a problem starts fixing.
Service is restored.
Manual Problem detection: Metrics

- **Goal:** Pinpoint root causes of problems faster

- Take a baseline of your system in steady-state

- Anomalies like spikes or dips from baseline can indicate problems
  - Ex: Slow Query Logging

- Integrates with existing infrastructure via JMX or use with Ganglia, Cloudera Manager
Metrics from all levels of the system

- **HBase Region Servers**
  - Operations / sec
  - Get / put latencies (0.92)
  - Per CF metrics (0.94)
  - Per Region metrics (0.94)

- **HBase Master**
  - RIT metrics
  - Replication Metrics

- **System/JVM**
  - GC, RPC metrics
### bfd-a Cluster Report for Sat, 22 Sep 2012 10:08:47 -0700

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPUS Total</td>
<td>80</td>
</tr>
<tr>
<td>Hosts up</td>
<td>5</td>
</tr>
<tr>
<td>Hosts down</td>
<td>0</td>
</tr>
</tbody>
</table>

Avg Load (15, 5, 1m): 10%, 11%, 6%

Localtime: 2012-09-22 10:08

**Cluster Load Percentages**

- 25-50 (50% of CPU)
- 5-25 (25% of CPU)

---

**Overview of bfd-a**

#### bfd-a Cluster Load last month

#### bfd-a Cluster CPU last month

#### bfd-a Cluster Memory last month

#### bfd-a Cluster Network last month

#### bfd-a Cluster Write Latency Report last month

#### bfd-a Cluster Read Latency Report last month

---

**bfd-a load_one last month sorted descending**

<table>
<thead>
<tr>
<th>Location</th>
<th>Load</th>
<th>Week 35</th>
<th>Week 36</th>
<th>Week 37</th>
<th>Week 38</th>
</tr>
</thead>
<tbody>
<tr>
<td>c1510.hal.cloudera.com</td>
<td>5.0</td>
<td>6.0</td>
<td>7.0</td>
<td>8.0</td>
<td>9.0</td>
</tr>
<tr>
<td>c1504.hal.cloudera.com</td>
<td>5.0</td>
<td>6.0</td>
<td>7.0</td>
<td>8.0</td>
<td>9.0</td>
</tr>
<tr>
<td>c1512.hal.cloudera.com</td>
<td>5.0</td>
<td>6.0</td>
<td>7.0</td>
<td>8.0</td>
<td>9.0</td>
</tr>
<tr>
<td>c1508.hal.cloudera.com</td>
<td>5.0</td>
<td>6.0</td>
<td>7.0</td>
<td>8.0</td>
<td>9.0</td>
</tr>
</tbody>
</table>
Highly Available stack: HDFS (2.0) / ZK / HBASE
Client Cross-version wire compatibility (0.96)
Rolling Restarts
Online Schema Change (experimental)
High Availability: HBase + HDFS + ZK

<table>
<thead>
<tr>
<th>HDFS NameNodes</th>
<th>ZooKeeper Quorum</th>
<th>Slave Boxes (DN + RS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rack 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name node</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rack 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name node</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Wire Compatibility

- Reduces downtime due to planned maintenance
- Wire compatibility + Extensible Data formats
  - Allow for forwards and backwards compatibility
  - Older clients can talk to newer servers and visa versa
- Rolling upgrade
  - Upgrade a single node at a time while system runs
- Allows API and changes while guaranteeing wire compatibility between different minor versions
- HDFS client-server compatibility between Major Versions
Risks from outside the cluster

Datacenters and Disaster Recovery
Severe storms cause Amazon Web Services outage

For second time in less than a month, Amazon’s Northern Virginia data center outage and is impacting many popular services such as Instagram, Pinterest, and others that previously suffered an outage in its Northern Virginia facilities on June 14.

The Backhoe: A Real Cyberthreat

At half-past noon on Jan. 9, cable TV contractors sinking a half-mile of cable near Interstate 10 in rural Arizona pulled up something unexpected in the bucket of their backhoe: an unmarked fiber-optic cable. "It started pulling the fiber out of the pipe," says Scott Johansson, project manager for JK Communications and Construction. "Obviously, we said, 'Oop, we've hit something.'"
Geographically separated copies of data
Strategy: HBase-Supported Batch Backups

- Export / Dist CP / Import
  - 3 batch MR jobs
  - Several extra copies of data
  - High latency (hours)

- Copy Table
  - 1 MR Job
  - Single copy of data
  - High Latency (hours)
  - Incremental table copies
Strategy: Custom Application-managed Replication

- Application writes to two instances of HBase
  - Low Latency
  - Adds complexity
  - Inefficient
Strategy: HBase replication (0.92+)

- HBase Asynchronously copy edit logs to other clusters.
  - Replication lag measured in seconds
  - Automatically catch up from failures.
  - Eventually consistent
  - Efficient batching
- Master-slave† (0.90)
- Master-master (0.92)
Master-Master Replication

Replicating data reduces chances of data loss.
Risks from Users

“Problem exists between keyboard and chair.”
Oops... User Error

• How do we prevent user error?
• How do we recover from user error?
Prevent user mistakes: User-level Security

- **User Error:** drop ‘table’
  - Operation rejected, insufficient permissions.

- **Authentication:**
  - Ensure the identity of the services or users that are communicating

- **Access Control:**
  - Ensure user has permission to execute table data operations

**time**

10/25/12 Strata Hadoop World 2012
HBase User-level Security

• Based on Kerberos for HBase, HDFS and Zookeeper
  • Grant privileges to users
  • Revoke privileges from users.
  • Column Family and Table granularity

• Confidentiality:
  • Ensure information is only seen by intended users.

• Audit Trails:
  • Track which users performed particular operations
Recovering from User Mistakes: Table Snapshots

- **Snapshot** the state of a table at a certain moment in time
- **Restore** it or **Clone** it later, creating a new read write table
- **Export** it to another cluster with minimal impact on HBase

User Error: drop ‘table’

Service is down!

Service is restored, minor data loss

Periodic snapshot

restore
Table Snapshots (0.96+)

• Under development, slated for HBase 0.96
• Multiple snapshot flavors planned
  • Offline snapshots
  • Online Snapshots
• Snapshot uses
  • Recover from application or user error.
  • Application experimentation (no need to spin up another cluster for replication)
  • Use MR directly on snapshot files
Conclusions

HBase for the enterprise.
Feature Summary by Category

Avoid Down Time
- Rolling Restart
- Online backups
- Table Replication
- Security Access Controls (0.92)
- Wire Compatibility (0.96)
- Snapshots (0.96)

Reduce Detection Time
- Improved metrics
- Proactive notification of HMaster failure (0.96)
- Proactive notification of RegionServer failure (0.96)

Reduce Recovery Time
- Distributed Log Splitting (0.92)
- Improvements to HBCK (0.94)
- Allow writes during recovery (0.96)
- Snapshots (0.96)
## Feature Summary by Version

<table>
<thead>
<tr>
<th></th>
<th>0.90</th>
<th>0.92</th>
<th>0.94</th>
<th>0.96 (Upcoming Release)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Metrics</td>
<td>• Metrics</td>
<td>• Metrics</td>
<td>• CF+Region Granularity Metrics</td>
<td>• CF +Region Granularity Metrics</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Improved failure detection time</td>
<td>• Improved failure detection time</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Distributed log</td>
<td>• Distributed log</td>
<td>• Distributed log</td>
<td>• Distributed log</td>
<td>• Distributed log</td>
</tr>
<tr>
<td>splitting*</td>
<td>splitting*</td>
<td>splitting*</td>
<td>splitting*</td>
<td>splitting*</td>
</tr>
<tr>
<td>HBCK improvements*</td>
<td>• HBCK improvements</td>
<td>• HBCK improvements</td>
<td>• HBCK improvements</td>
<td>• HBCK improvements</td>
</tr>
<tr>
<td>Copy Table / Import / Export</td>
<td>• Copy Table / Import / Export</td>
<td>• Copy Table / Import / Export</td>
<td>• Copy Table / Import / Export</td>
<td>• Copy Table / Import / Export</td>
</tr>
<tr>
<td>Master-Slave Replication†</td>
<td>• Master-Slave Replication†</td>
<td>• Master-Master Replication</td>
<td>• Master-Master Replication</td>
<td>• Master-Master Replication</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Authentication and</td>
<td>• Authentication and</td>
<td>• Authentication and</td>
<td>• Authentication and</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(Snapshots)</td>
</tr>
<tr>
<td>Recovery in Hours</td>
<td>Recovery in Hours</td>
<td>Recovery in Minutes</td>
<td>Recovery in Minutes</td>
<td>(Recovery in Seconds)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

† experimental (in progress)  
* backported in CDH
Thank You!

Jonathan Hsieh | @jmhsieh
Software Engineer, Cloudera
Apache HBase committer / PMC member