Apache Hadoop 3

Andrew Wang
andrew.wang@cloudera.com

Daniel Templeton
daniel@cloudera.com
Who We Are

Andrew Wang
- HDFS @ Cloudera
- Hadoop PMC Member
- Release Manager for Hadoop 3.0

Daniel Templeton
- YARN @ Cloudera
- Hadoop PMC Member
## An Abbreviated History of Hadoop Releases

<table>
<thead>
<tr>
<th>Date</th>
<th>Release</th>
<th>Major Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2007-11-04</td>
<td>0.14.1</td>
<td>First release at the ASF</td>
</tr>
<tr>
<td>2011-12-27</td>
<td>1.0.0</td>
<td>Security, HBase support</td>
</tr>
<tr>
<td>2012-05-23</td>
<td>2.0.0</td>
<td>YARN, NameNode HA, wire compat</td>
</tr>
<tr>
<td>2014-11-18</td>
<td>2.6.0</td>
<td>HDFS encryption, rolling upgrade, node labels</td>
</tr>
<tr>
<td>2015-04-21</td>
<td>2.7.0</td>
<td>Truncate, Variable-length blocks, YARN Global Caching,</td>
</tr>
<tr>
<td>2017-03-22</td>
<td>2.8.0</td>
<td>Cloud improvement, Azure Data Lake, and etc.</td>
</tr>
<tr>
<td>2017-11-17</td>
<td>2.9.0</td>
<td>Stability Improvement</td>
</tr>
<tr>
<td>2017-12-13</td>
<td>3.0.0</td>
<td>Java 8, Erasure Coding, S3Guard, YARN Timeline Service</td>
</tr>
</tbody>
</table>
Motivation for Hadoop 3

- Upgrade minimum Java version to Java 8
  - Java 7 end-of-life in April 2015
  - Many Java libraries now only support Java 8
- HDFS erasure coding
  - Major feature that refactored core pieces of HDFS
  - Too big to backport to 2.x
- Classpath isolation
  - Potentially impacts all clients
- Other miscellaneous incompatible bugfixes and improvements
  - Hadoop 2.x was branched in 2011
  - 6 years of changes waiting for 3.0
Hadoop 3 Status and Release Plan

- After four alphas and one beta, 3.0.0 is out!
- Took close to two years from inception
- 3.0.1 and 3.1.0 are already in progress

<table>
<thead>
<tr>
<th>Release</th>
<th>Date</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>3.0.0-alpha1</td>
<td>2016-09-03</td>
<td>✔</td>
</tr>
<tr>
<td>3.0.0-alpha2</td>
<td>2017-01-25</td>
<td>✔</td>
</tr>
<tr>
<td>3.0.0-alpha3</td>
<td>2017-05-26</td>
<td>✔</td>
</tr>
<tr>
<td>3.0.0-alpha4</td>
<td>2017-07-07</td>
<td>✔</td>
</tr>
<tr>
<td>3.0.0-beta1</td>
<td>2017-10-03</td>
<td>✔</td>
</tr>
<tr>
<td>3.0.0 GA</td>
<td>2017-12-13</td>
<td>✔</td>
</tr>
<tr>
<td>3.0.1</td>
<td>2017 Mar</td>
<td></td>
</tr>
</tbody>
</table>

https://cwiki.apache.org/confluence/display/HADOOP/Hadoop+3.0.0+release
HDFS & Hadoop Features
3x replication vs. Erasure coding

/foo.csv - 3 block file

b1  b2  b3
3x replication vs. Erasure coding

/foo.csv - 3 block file

b1  b2  b3
b1  b2  b3
b1  b2  b3
3x replication vs. Erasure coding

/foo.csv - 3 block file

3 replicas

3 blocks

3 x 3 = 9 total replicas

9 / 3 = 200% overhead!
3x replication vs. Erasure coding

/foo.csv - 3 block file

b1  b2  b3
3x replication vs. Erasure coding

/foo.csv - 3 block file

b1  b2  b3  p1  p2
3x replication vs. Erasure coding

/foo.csv - 3 block file

3 data blocks

b1  b2  b3

2 parity blocks

p1  p2

3 + 2 = 5 replicas
5 / 3 = 67% overhead!
3x replication vs. Erasure coding

/foo.csv - 3 block file

3 data blocks

p1  p2  p3

2 parity blocks

3 + 2 = 5 replicas
5 / 3 = 67% overhead!

/bigfoo.csv - 10 block file

10 data blocks

p1  ...  p4

4 parity blocks

10 + 4 = 14 replicas
14 / 10 = 40% overhead!
EC Reconstruction

/foo.csv - 3 block file

Reed-Solomon (3,2)
EC Reconstruction

/foo.csv - 3 block file

Reed-Solomon (3,2)
EC Reconstruction

/foo.csv - 3 block file

Reed-Solomon (3,2)

Read 3 remaining blocks

Run RS to recover b3

New copy of b3 recovered
Erasure coding (HDFS-7285)

- Motivation: improve storage efficiency of HDFS
  - $\sim 2x$ the storage efficiency compared to 3x replication
  - Reduction of overhead from 200% to 40%
- Uses Reed-Solomon(k,m) erasure codes instead of replication
  - Support for multiple erasure coding policies
  - RS(3,2), RS(6,3), RS(10,4)
- Can improves data durability
  - RS(6,3) can tolerate 3 failures
  - RS(10,4) can tolerate 4 failures
- Missing blocks reconstructed from remaining blocks
EC implications

- File data is striped across multiple nodes and racks
- Reads and writes are remote and cross-rack
- Reconstruction is network-intensive, reads $m$ blocks cross-rack
- Important to use Intel’s optimized ISA-L for performance
  - $1+ \text{ GB/s encode/decode speed, much faster than Java implementation}$
- Combine data into larger files to avoid an explosion in # replicas
  - Bad: 1x1GB file -> RS(10,4) -> 14x100MB EC blocks (4.6x # replicas)
  - Good: 10x1GB file -> RS(10,4) -> 14x1GB EC blocks (0.46x # replicas)
- Works best for archival / cold data use cases
EC performance

Write Throughput MB/s

<table>
<thead>
<tr>
<th></th>
<th>10GB+HDD</th>
<th>20GB+HDD</th>
<th>10GB+SSD</th>
<th>20GB+SSD</th>
</tr>
</thead>
<tbody>
<tr>
<td>3x Replication</td>
<td>259</td>
<td>252</td>
<td>259</td>
<td>262</td>
</tr>
<tr>
<td>HDFS-RAID coder</td>
<td>36</td>
<td>40</td>
<td>36</td>
<td>33</td>
</tr>
<tr>
<td>New Java coder</td>
<td>210</td>
<td>207</td>
<td>210</td>
<td>210</td>
</tr>
<tr>
<td>ISA-L coder</td>
<td>460</td>
<td>461</td>
<td>460</td>
<td>476</td>
</tr>
</tbody>
</table>

© Cloudera, Inc. All rights reserved.
EC performance

Spark TPC-H 500GB Query Time (s) — no DN killed

- **count(*) on line item**
  - EC: 4,497
  - Replica: 4,358
- **Q3**
  - EC: 118,381
  - Replica: 108,437
- **Q6**
  - EC: 30,429
  - Replica: 21,465
EC performance

Spark TPC-H 500GB Query Time (s) — 2 DN$\text{s}$ killed

- Q3: EC = 121.448, Replica = 111.381
- Q6: EC = 41.453, Replica = 26.584

$\text{count(*) on line item}$
Erasure coding status

- Massive development effort by the Hadoop community
  - 20+ contributors from many companies
    - Cloudera, Intel, Hortonworks, Huawei, Y! JP, ...
  - 100s of commits over more than three years (started in 2014)
- Erasure coding is ready in 3.0.0 GA!
- Current focus is on testing and integration efforts
  - Want the complete Hadoop stack to work with HDFS erasure coding enabled
  - Ongoing stress / endurance testing to ensure stability at scale
Classpath isolation (HADOOP-11656)

- Hadoop leaks lots of dependencies onto the application’s classpath
  - Known offenders: Guava, Protobuf, Jackson, Jetty, ...
- No separate HDFS client jar means server jars are leaked
- YARN / MR clients not shaded

- **HDFS-6200**: Split HDFS client into separate JAR
- **HADOOP-11804**: Shaded hadoop-client dependency
- **YARN-6466**: Shade the task umbilical for a clean YARN container environment (ongoing)
Miscellaneous

- Supportability improvements
  - Shell script rewrite
  - Intra-DataNode balancer
  - Move default ports out of the ephemeral range
- Support for multiple Standby NameNodes
- Cloud enhancements
  - Support for Microsoft Azure Data Lake and Aliyun OSS
  - S3 consistency and performance improvements
- Tightened Hadoop compatibility policy
YARN Features
Job History Server
Job History Server
Job History Server

Resource Manager

Job History Server

jobs
Job History Server

- Resource Manager
- Node Manager
- Job History Server
- HDFS

jobs
Job History Server

Resource Manager

Spark History Server

Job History Server
Job History Server
Application Timeline Service v2

- Store for application and system events and data
  - Distributed
  - Scalable
  - Structured Data Model
- Updated in real time
  - Application status
  - Application metrics
  - System metrics
- Fed by resource manager, node manager, and application masters
- REST API
Application Timeline Service v2
Application Timeline Service v2

Timeline Reader → HBase → Timeline Collector

Resource Manager → Timeline Collector
Node Manager
Application Master
Application Timeline Service v2 Flows
### Recent Flow Activities

<table>
<thead>
<tr>
<th>Flow Name</th>
<th>User</th>
<th>Flow ID</th>
<th>Last Execution Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>QuicksMonteCarlo</td>
<td>halbochen</td>
<td>halbochen-pseudo-tms-cluster/halbochen/QuicksMonteCarlo</td>
<td>2017/06/05</td>
</tr>
<tr>
<td>Sleep job</td>
<td>halbochen</td>
<td>halbochen-pseudo-tms-cluster/halbochen/Sleep job</td>
<td>2017/06/06</td>
</tr>
<tr>
<td>20170608 demo_1</td>
<td>halbochen</td>
<td>halbochen-pseudo-tms-cluster/halbochen/20170608 demo_1</td>
<td>2017/06/05</td>
</tr>
<tr>
<td>long sleep</td>
<td>halbochen</td>
<td>halbochen-pseudo-tms-cluster/halbochen/long sleep</td>
<td>2017/06/05</td>
</tr>
<tr>
<td>pi calculation</td>
<td>halbochen</td>
<td>halbochen-pseudo-tms-cluster/halbochen/pi_calculat...</td>
<td>2017/06/05</td>
</tr>
<tr>
<td>word count</td>
<td>halbochen</td>
<td>halbochen-pseudo-tms-cluster/halbochen/word count</td>
<td>2017/06/05</td>
</tr>
<tr>
<td>TextsGen</td>
<td>halbochen</td>
<td>halbochen-pseudo-tms-cluster/halbochen/TextsGen</td>
<td>2017/06/25</td>
</tr>
<tr>
<td>mapreduce.BorweinPlouffe_10_20</td>
<td>halbochen</td>
<td>halbochen-pseudo-tms-cluster/halbochen/mapreduce.BorweinPlouffe_10_20</td>
<td>2017/05/24</td>
</tr>
<tr>
<td>tsv2 demo_1</td>
<td>halbochen</td>
<td>halbochen-pseudo-tms-cluster/halbochen/tsv2 demo_1</td>
<td>2017/05/24</td>
</tr>
<tr>
<td>tsv2 demo_2</td>
<td>halbochen</td>
<td>halbochen-pseudo-tms-cluster/halbochen/tsv2 demo_2</td>
<td>2017/05/24</td>
</tr>
</tbody>
</table>
Application Timeline Service v2 Flows
## Old YARN UI

### Cluster Metrics

<table>
<thead>
<tr>
<th>Apps Submitted</th>
<th>Apps Pending</th>
<th>Apps Running</th>
<th>Apps Completed</th>
<th>Containers Running</th>
<th>Memory Used</th>
<th>Memory Total</th>
<th>Memory Reserved</th>
<th>Active Nodes</th>
<th>Decommissioned Nodes</th>
<th>Lost Nodes</th>
<th>Unhealthy Nodes</th>
<th>Ratios</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>0 B</td>
<td>16 GB</td>
<td>0 B</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Ratios</td>
</tr>
</tbody>
</table>

### All Applications

<table>
<thead>
<tr>
<th>ID</th>
<th>User</th>
<th>Name</th>
<th>Application Type</th>
<th>Queue</th>
<th>Start Time</th>
<th>Finish Time</th>
<th>State</th>
<th>Final Status</th>
<th>Progress</th>
</tr>
</thead>
<tbody>
<tr>
<td>application_1396885203337_0001</td>
<td>siva</td>
<td>word count</td>
<td>MAPREDUCE</td>
<td>default</td>
<td>Mon, 07 Apr 2014 15:42:02 GMT</td>
<td>Mon, 07 Apr 2014 15:42:22 GMT</td>
<td>FINISHED</td>
<td>SUCCEEDED</td>
<td></td>
</tr>
<tr>
<td>application_1396885203337_0002</td>
<td>siva</td>
<td>word count</td>
<td>MAPREDUCE</td>
<td>default</td>
<td>Mon, 07 Apr 2014 16:16:00 GMT</td>
<td>Mon, 07 Apr 2014 16:16:17 GMT</td>
<td>FINISHED</td>
<td>SUCCEEDED</td>
<td></td>
</tr>
<tr>
<td>application_1396885203337_0003</td>
<td>siva</td>
<td>word count</td>
<td>MAPREDUCE</td>
<td>default</td>
<td>Mon, 07 Apr 2014 16:17:51 GMT</td>
<td>Mon, 07 Apr 2014 16:18:08 GMT</td>
<td>FINISHED</td>
<td>SUCCEEDED</td>
<td></td>
</tr>
<tr>
<td>application_1396885203337_0004</td>
<td>siva</td>
<td>word mean</td>
<td>MAPREDUCE</td>
<td>default</td>
<td>Mon, 07 Apr 2014 16:41:08 GMT</td>
<td>Mon, 07 Apr 2014 16:41:24 GMT</td>
<td>FINISHED</td>
<td>SUCCEEDED</td>
<td></td>
</tr>
<tr>
<td>application_1396885203337_0005</td>
<td>siva</td>
<td>word mean</td>
<td>MAPREDUCE</td>
<td>default</td>
<td>Mon, 07 Apr 2014 16:44:08 GMT</td>
<td>Mon, 07 Apr 2014 16:50:08 GMT</td>
<td>FINISHED</td>
<td>SUCCEEDED</td>
<td></td>
</tr>
<tr>
<td>application_1396885203337_0006</td>
<td>siva</td>
<td>word mean</td>
<td>MAPREDUCE</td>
<td>default</td>
<td>Mon, 07 Apr 2014 16:50:08 GMT</td>
<td>Mon, 07 Apr 2014 16:50:24 GMT</td>
<td>FINISHED</td>
<td>SUCCEEDED</td>
<td></td>
</tr>
</tbody>
</table>
New YARN UI

- Rich client application
  - Built on Node.js and Ember
- Improved visibility into cluster usage
  - Memory, CPU
  - By queues and applications
  - Sunburst graphs for hierarchical queues
  - NodeManager heatmap
- ATSv2 integration
  - Plot container start/stop events
  - Easy to capture delays in app execution
New YARN UI: Cluster Overview
New YARN UI: Queues
Resource Types

- Before Hadoop 3 memory and CPU are the only managed resources.
- Resource Types allows adding new managed resources:
  - Countable resources: GPUs, Disks etc.
  - Static resources: Java version, Python version, hardware profile, ...
    - Still in proposal stage
- Resource profiles:
  - Similar conceptually to EC2 instance types
  - Capture complex resource request
- DRF for scheduling
- Current virtual CPU cores and memory resources work as before
YARN Federation

- YARN scalability
  - Twitter runs a 10k node cluster with fair scheduler
  - Yahoo! runs 4k node cluster with capacity scheduler
- Federation
  - Restrict users to sub-clusters based on policy
  - Scalability to 100k nodes and beyond
  - Independent cluster scheduling
YARN Federation

Admin

Policy

Router

Resource Manager

Node Manager
Node Manager
Node Manager
Node Manager

Resource Manager

Node Manager
Node Manager
Node Manager
Node Manager
Opportunistic Containers

- Scheduler’s job is to keep all resources busy
- Scheduling gaps
  - Nothing to run
  - Resource contention
  - Resource reservations
- Opportunistic containers fill those gaps
  - Requested explicitly
  - Dedicated scheduler
  - Queued at the node managers
  - Scheduled locally when resources are available
  - Preempted when guaranteed containers need to run
- Coming in 2.9 and 3.0
Oversubscription

- Resource utilization is typically low in most clusters (20-50%)
  - Provision for peak usage
- Usage < Allocation
  - Mean Usage = ½ Peak Usage
Oversubscription

- Oversubscription
  - Allocate opportunistic containers to use *allocated-but-unused* resources
  - Jobs automatically use these unless they *opt-out*
  - Threshold to control aggressiveness of oversubscription
  - Threshold to trigger preemption

- *Currently in progress*
Other YARN Improvements

- Long Running Services
  - Slider merging into YARN
  - Docker support
- Scheduler improvements
  - Capacity scheduler
    - Performance and preemption improvements
    - Online scheduling ("global scheduler")
    - Queue management
  - Fair scheduler
    - Performance and preemption improvements
- High availability improvements
  - Better handling of transient network issues
  - ZK-store scalability: Limit number of children under a znode
- MapReduce Native Collector (MAPREDUCE-2841)
  - Native implementation of the map output collector
  - Up to 30% faster for shuffle-intensive jobs
Summary: What’s new in Hadoop 3.0?

- Storage Optimization
  - HDFS: Erasure codes
- Improved Visibility into Cluster Operations
  - YARN: ATSvc2
  - YARN: New UI
- Scalability & Multi-tenancy
  - YARN: Federation
- Improved Utilization
  - YARN: Opportunistic Containers
  - YARN: Oversubscription
- Refactor Base
  - Lots of Trunk content
  - JDK8 and newer dependent libraries
Compatibility and Testing
Compatibility

- Strong feedback from large users on the need for compatibility
- Preserves wire-compatibility with Hadoop 2 clients
  - Impossible to coordinate upgrading off-cluster Hadoop clients
- Will support rolling upgrade from Hadoop 2 to Hadoop 3
  - Can’t take downtime to upgrade a business-critical cluster
- Not fully preserving API compatibility!
  - Dependency version bumps
  - Removal of deprecated APIs and tools
  - Shell script rewrite, rework of Hadoop tools scripts
  - Incompatible bug fixes
Testing and Validation

- Cloudera CDH 6 is based on upstream Hadoop 3.0.0
  - Running full test suite
  - Integration of Hadoop 3 with all components in CDH stack
  - Same integration tests used to validate CDH5
- Plans for extensive HDFS EC testing by Cloudera and Intel
- Happy synergy between 2.8.x and 3.0.x lines
  - Shares much of the same code, fixes flow into both
  - Yahoo! doing scale testing of 2.8.0
Conclusion

● Hadoop 3.0.0 GA is out!
● Shiny new features
  ○ HDFS erasure coding
  ○ Client classpath isolation
  ○ YARN ATSv2
  ○ YARN Federation
  ○ Opportunistic containers and oversubscription
● Great time to get involved in testing and validation
Thank you

Andrew Wang  
andrew.wang@cloudera.com

Daniel Templeton  
daniel@cloudera.com