cloudera

Apache Hadoop 3

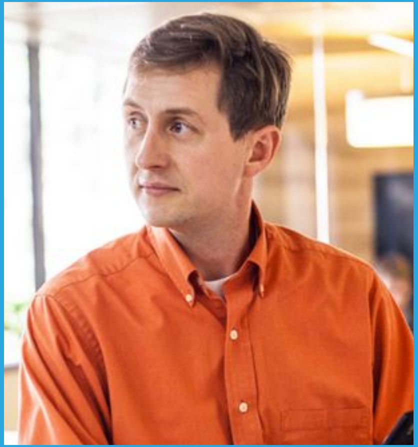
Andrew Wang
andrew.wang@cloudera.com

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

cloudera

An Abbreviated History of Hadoop Releases

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

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

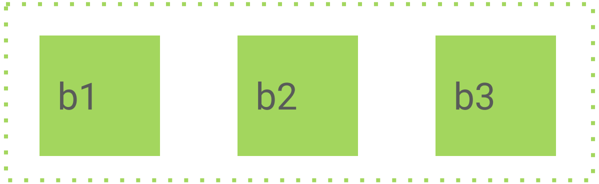
Release	Date	
3.0.0-alpha1	2016-09-03	✔
3.0.0-alpha2	2017-01-25	✔
3.0.0-alpha3	2017-05-26	✔
3.0.0-alpha4	2017-07-07	✔
3.0.0-beta1	2017-10-03	✔
3.0.0 GA	2017-12-13	✔
3.0.1	2017 Mar	

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



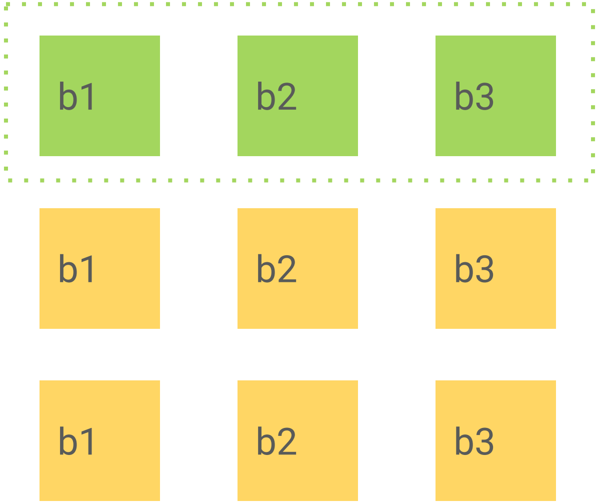
b1

b2

b3

3x replication vs. Erasure coding

/foo.csv - 3 block file



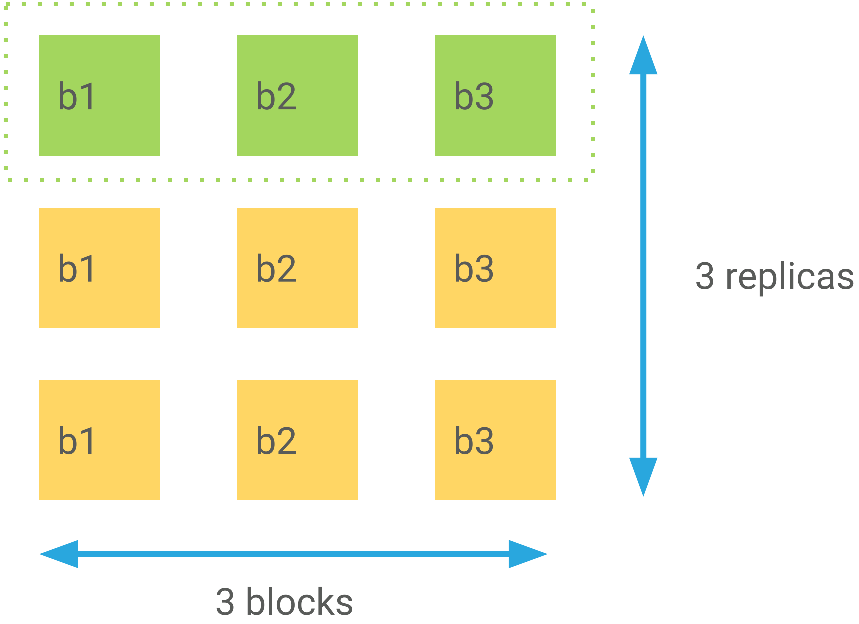
b1 b2 b3

b1 b2 b3

b1 b2 b3

3x replication vs. Erasure coding

/foo.csv - 3 block file



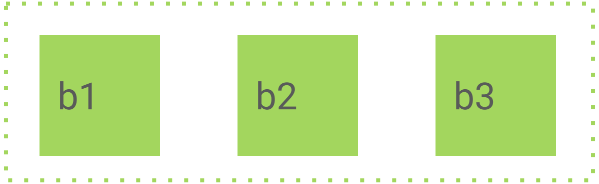
b1	b2	b3
b1	b2	b3
b1	b2	b3

3 x 3 = 9 total replicas

9 / 3 = 200% overhead!

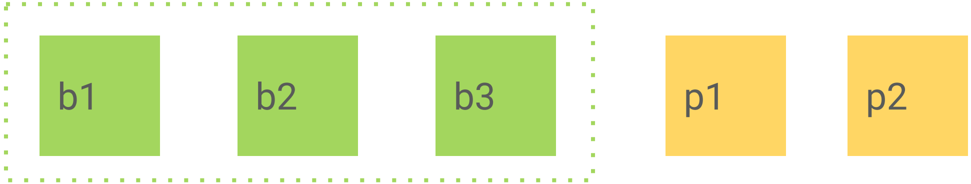
3x replication vs. Erasure coding

/foo.csv - 3 block file

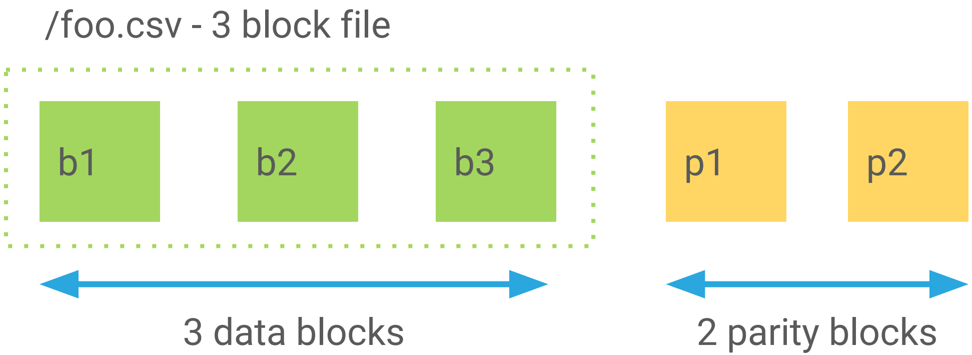


3x replication vs. Erasure coding

/foo.csv - 3 block file



3x replication vs. Erasure coding



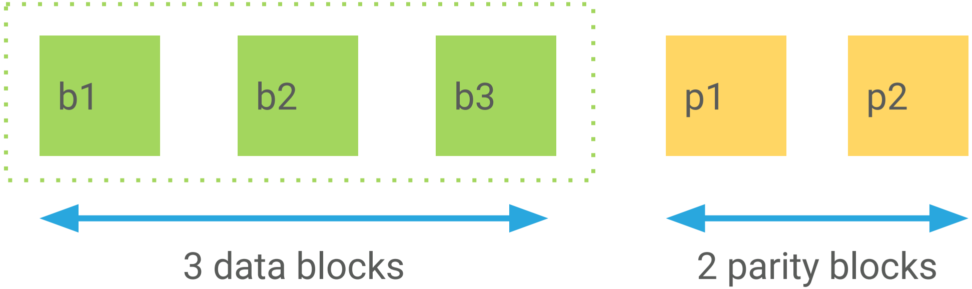
3 + 2 = 5 replicas

5 / 3 = 67% overhead!

cloudera

3x replication vs. Erasure coding

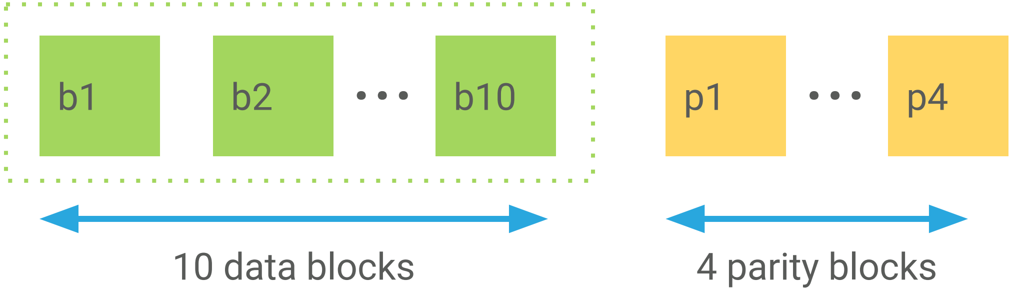
/foo.csv - 3 block file



3 + 2 = 5 replicas

5 / 3 = 67% overhead!

/bigfoo.csv - 10 block file



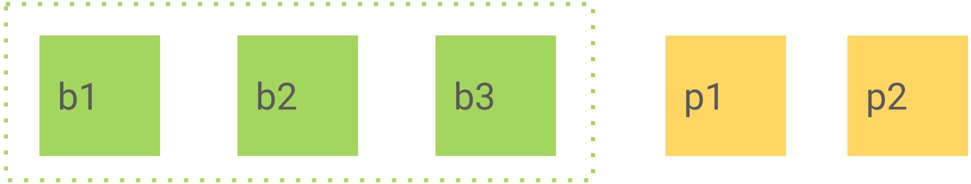
10 + 4 = 14 replicas

14 / 10 = 40% overhead!

cloudera

EC Reconstruction

/foo.csv - 3 block file



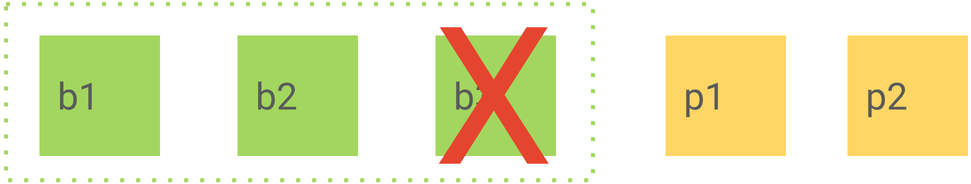
b1 b2 b3

p1 p2

Reed-Solomon (3,2)

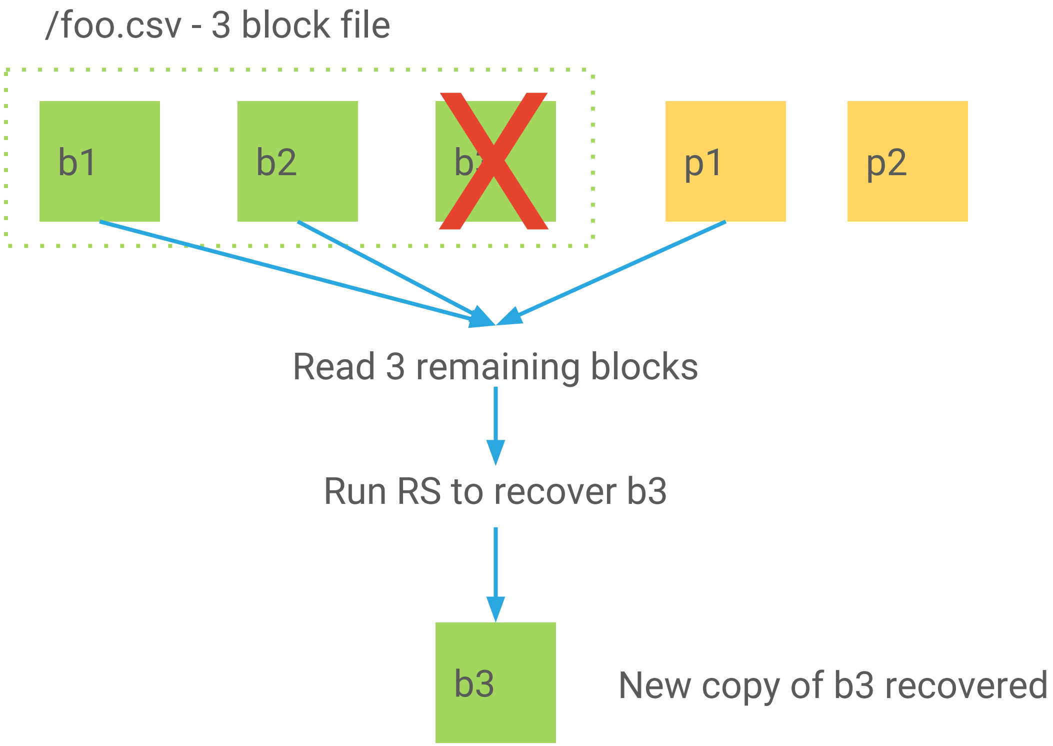
EC Reconstruction

/foo.csv - 3 block file



Reed-Solomon (3,2)

EC Reconstruction



/foo.csv - 3 block file

Read 3 remaining blocks

Run RS to recover b3

New copy of b3 recovered

Reed-Solomon (3,2)

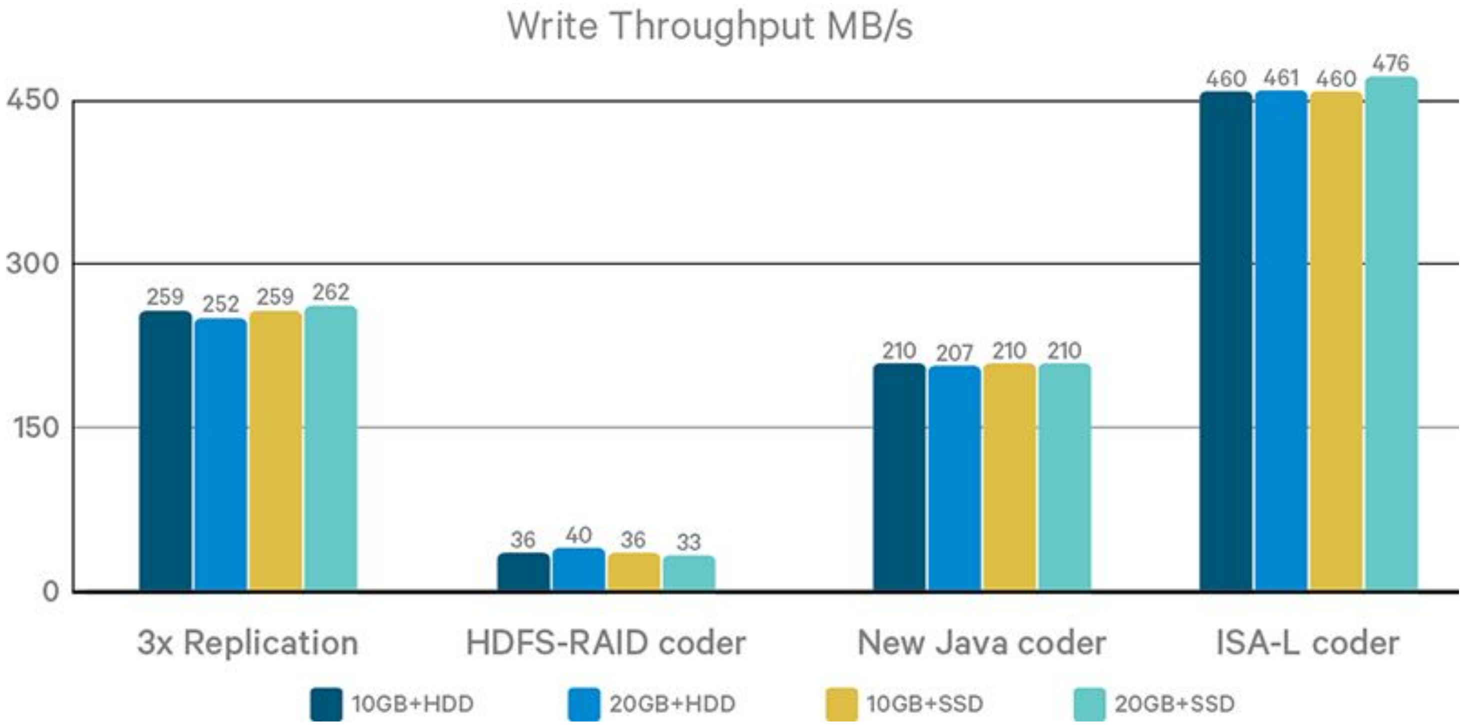
Erasure coding (HDFS-7285)

- Motivation: improve storage efficiency of HDFS
  - **~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

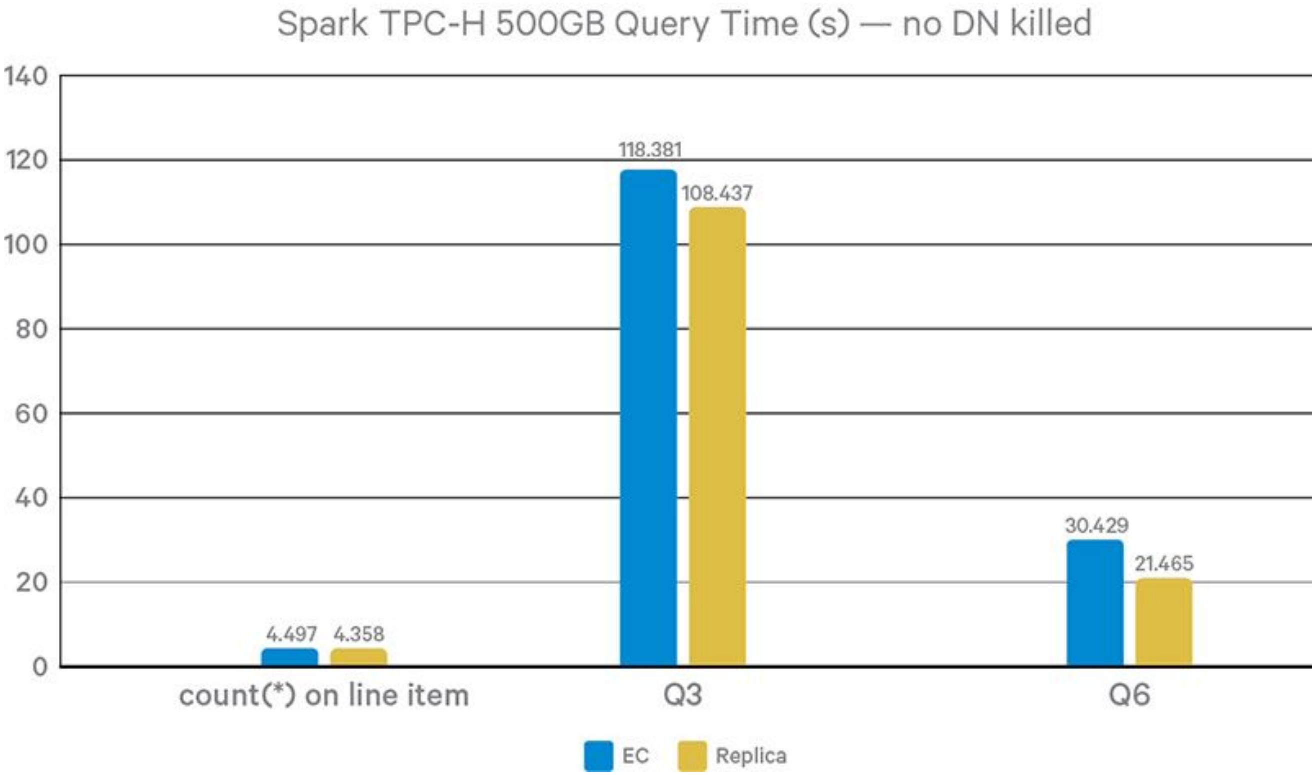
EC implications

- 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+ 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

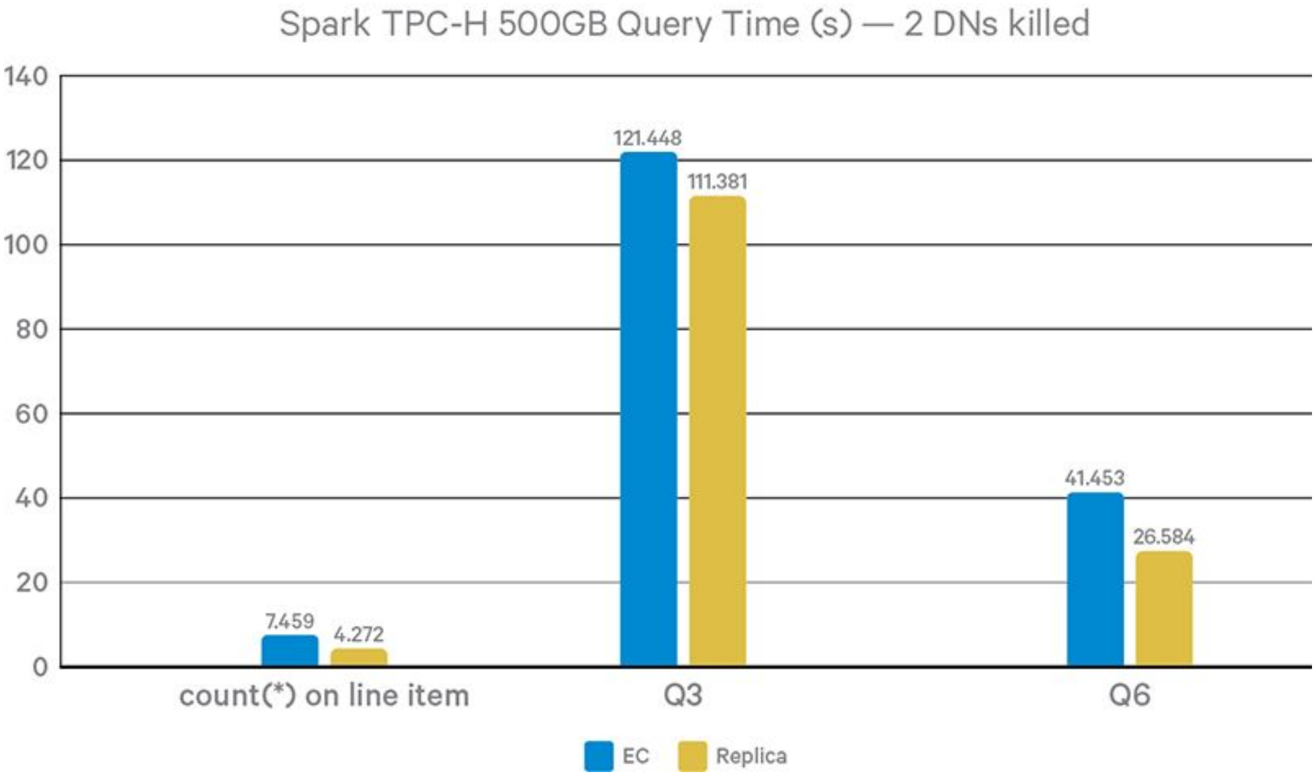


EC performance



cloudera

EC performance



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)

cloudera

Miscellaneous

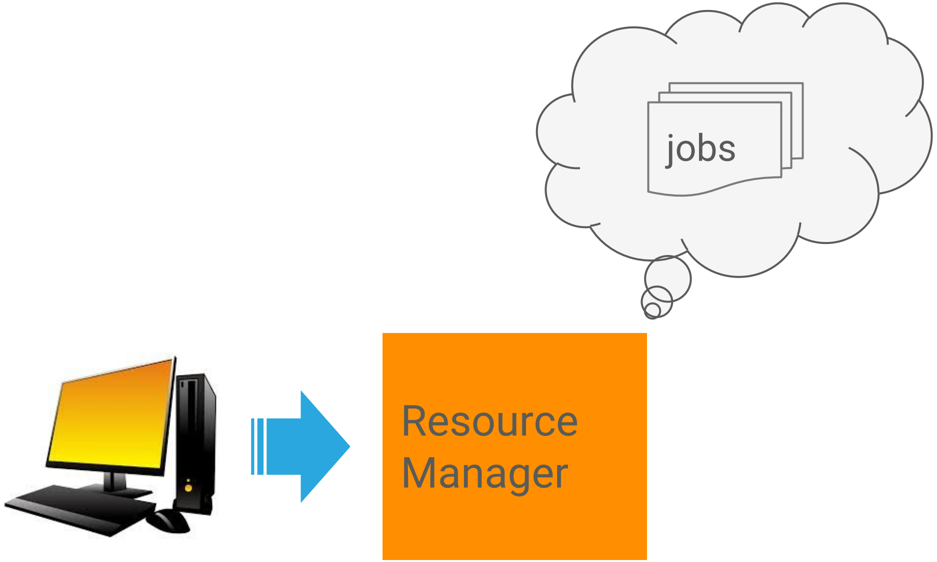
- Supportability improvements
  - Shell script rewrite
  - Intra-DataNode balancer
  - Move default ports out of the ephemeral range
- Support for multiple Standby NameNodes
- Cloud enhancements
  - Support for Microsoft Azure Data Lake and Aliyun OSS
  - S3 consistency and performance improvements
-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

YARN Features

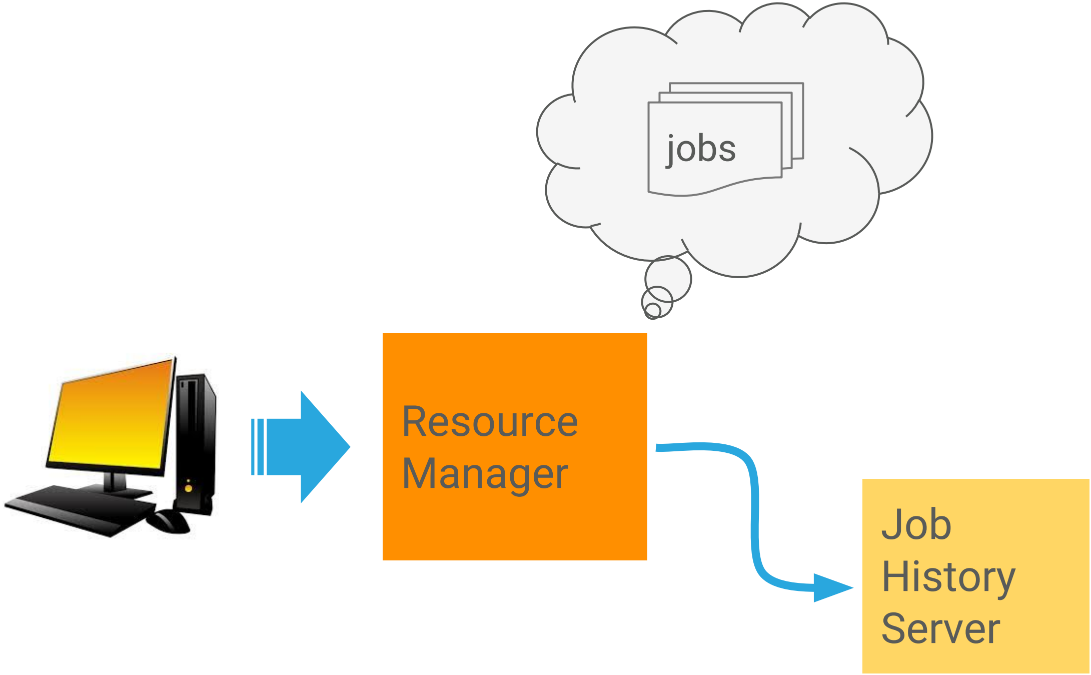
Job History Server

Resource Manager

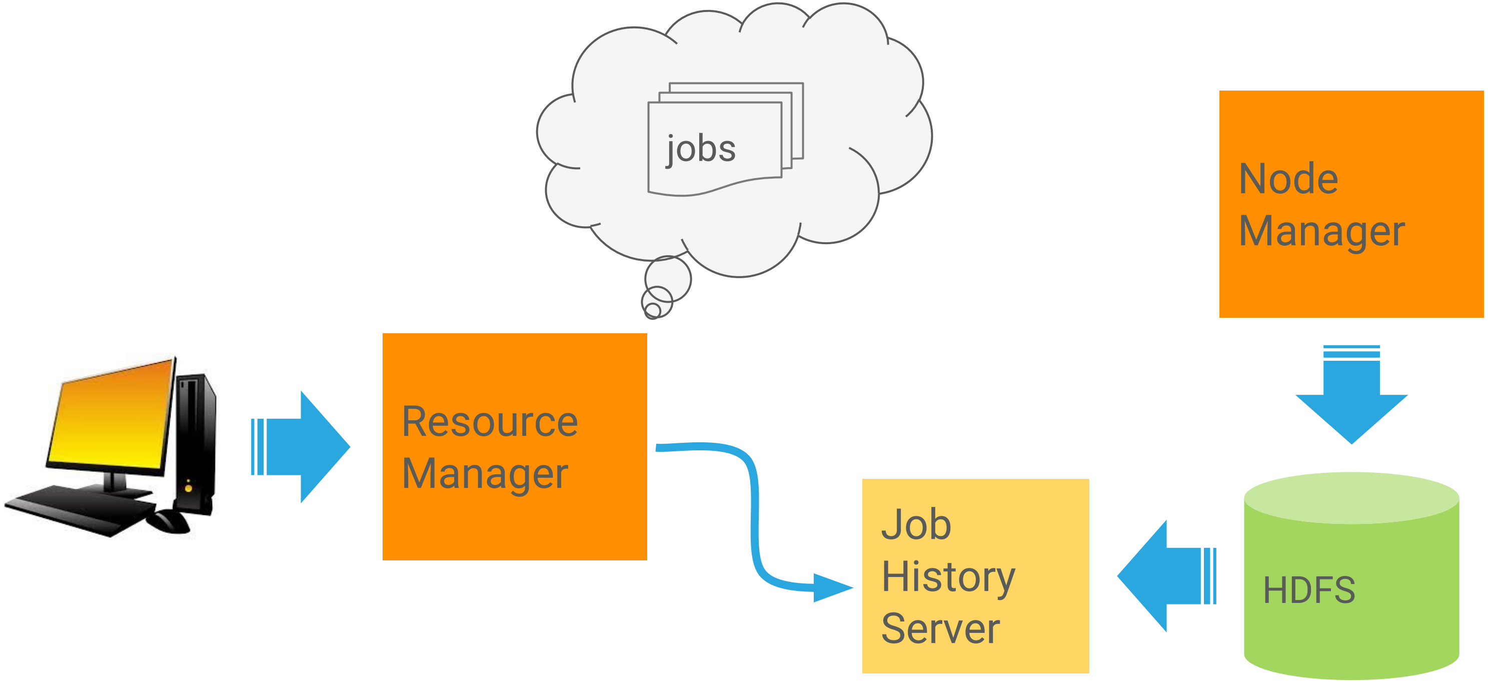
Job History Server



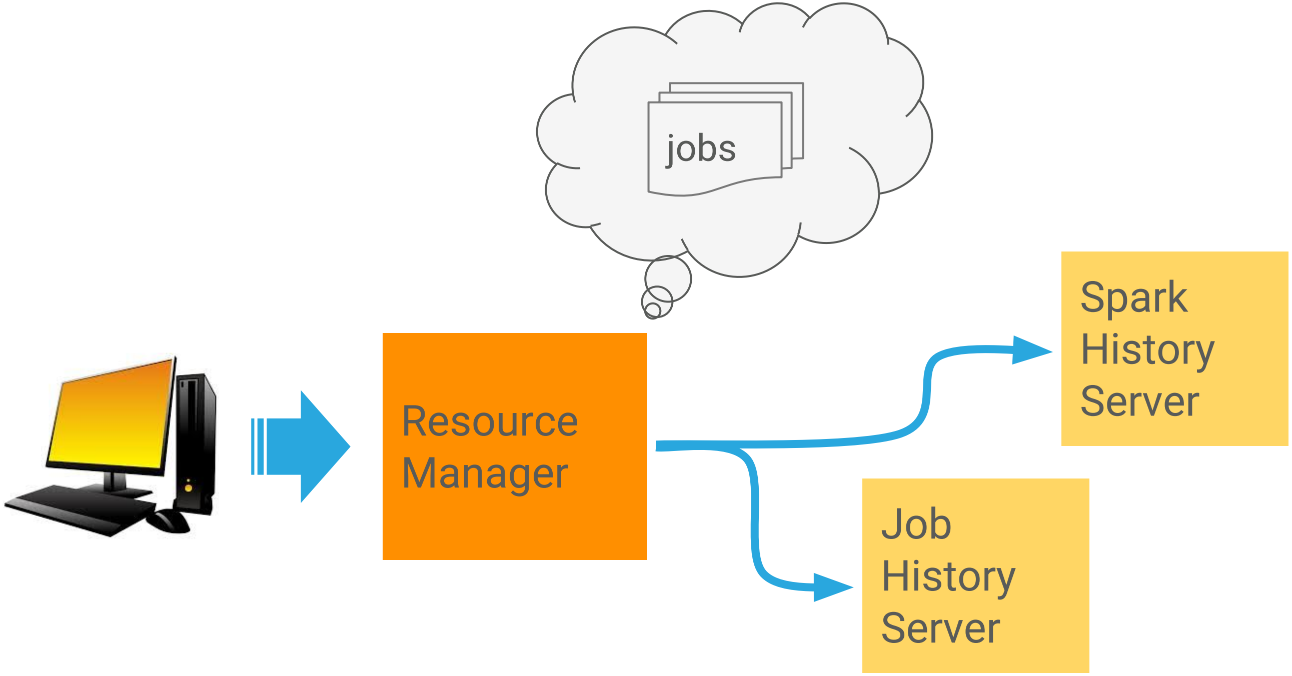
Job History Server



Job History Server



Job History Server



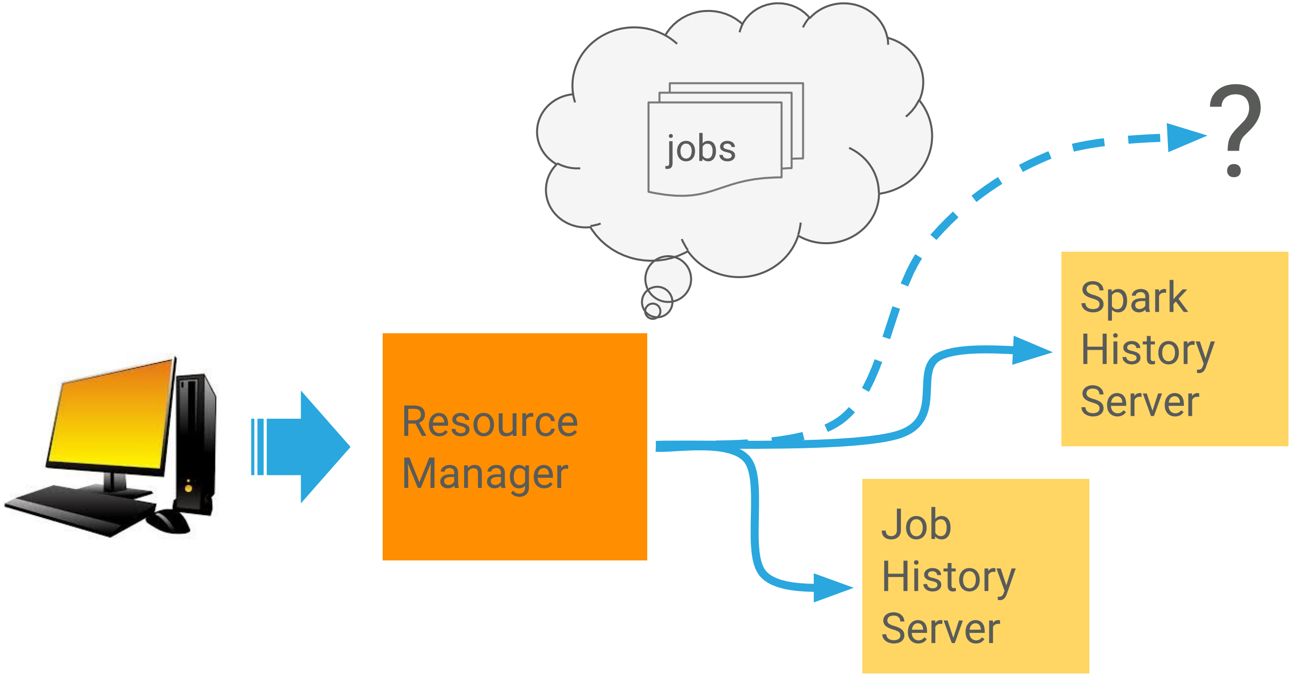
jobs

Resource Manager

Spark History Server

Job History Server

Job History Server



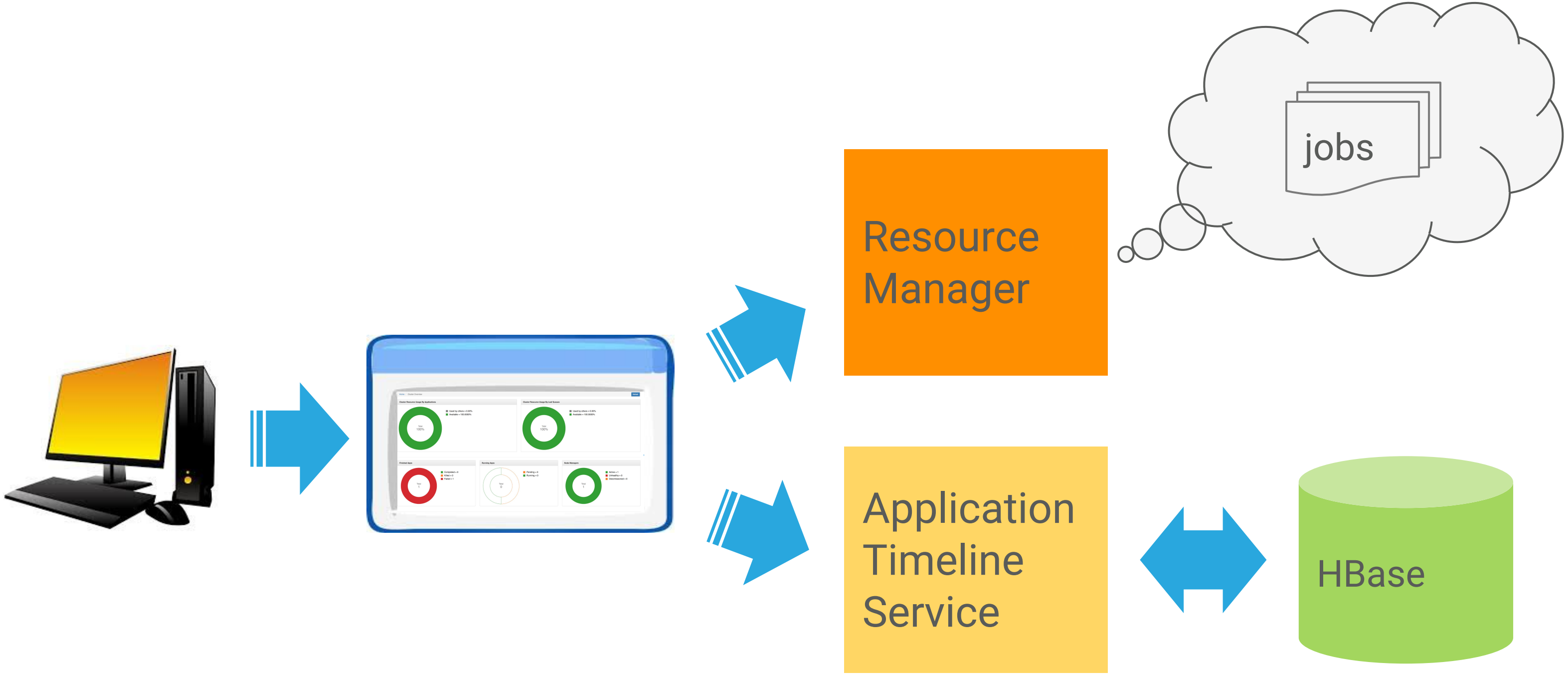
cloudera

© Cloudera, Inc. All rights reserved.

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



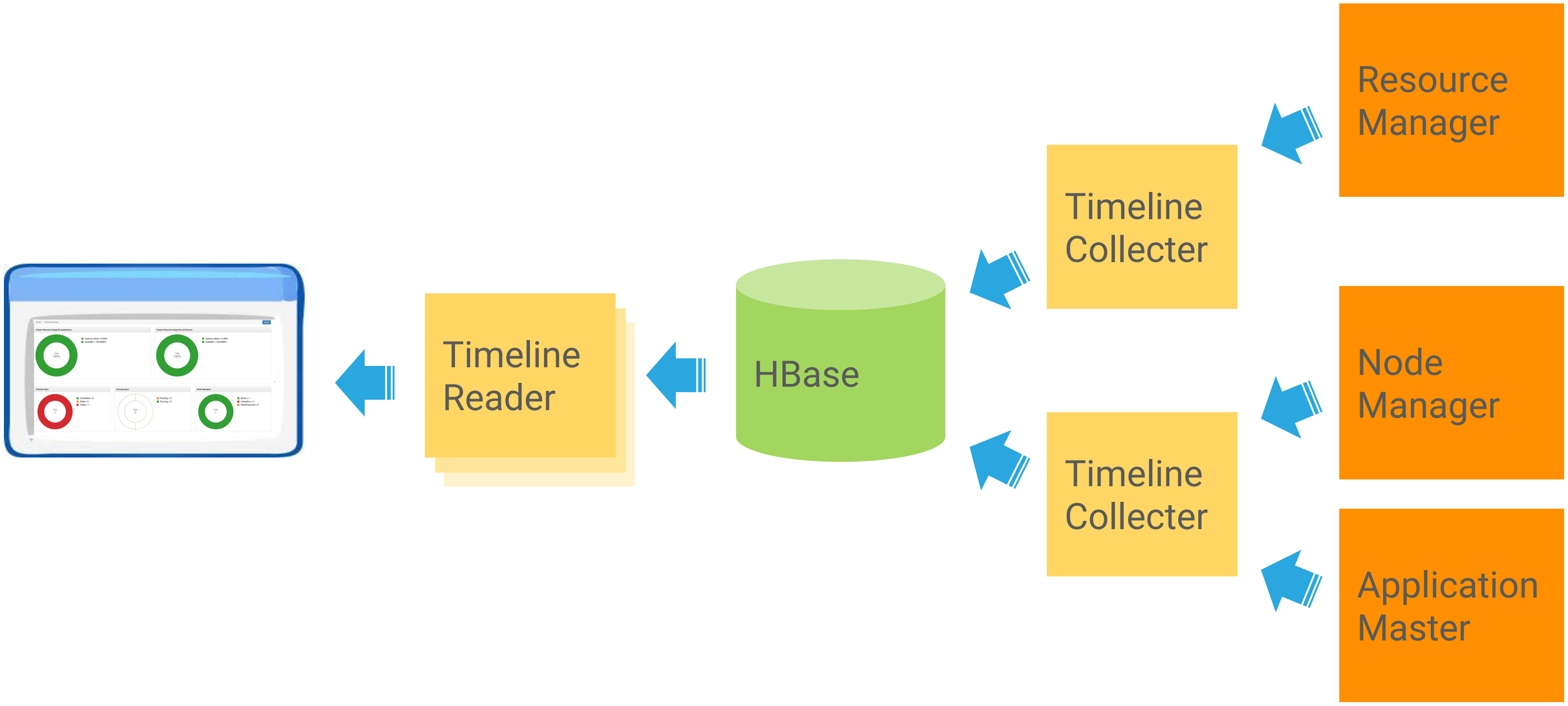
jobs

Resource Manager

Application Timeline Service

HBase

Application Timeline Service v2



Timeline Reader

HBase

Timeline Collecter

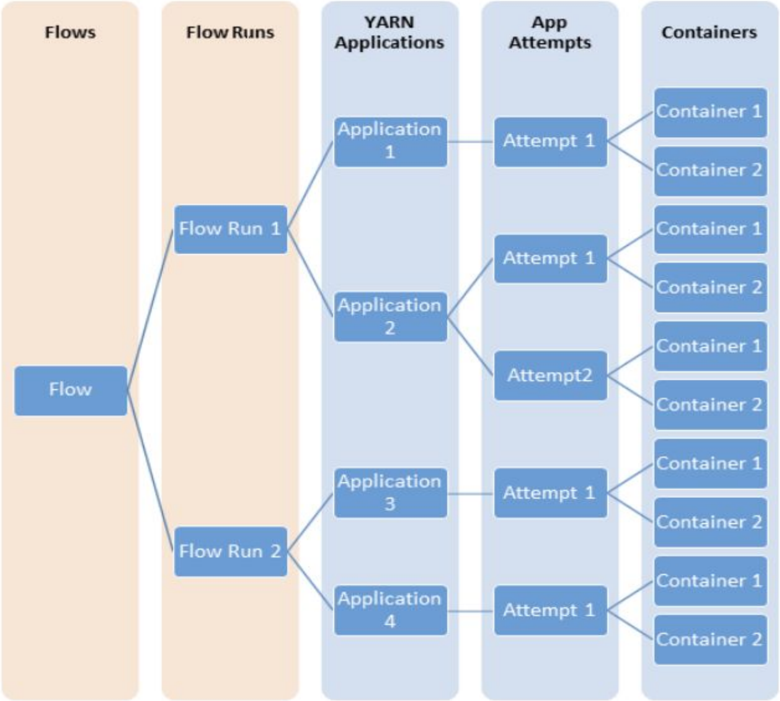
Timeline Collecter

Resource Manager

Node Manager

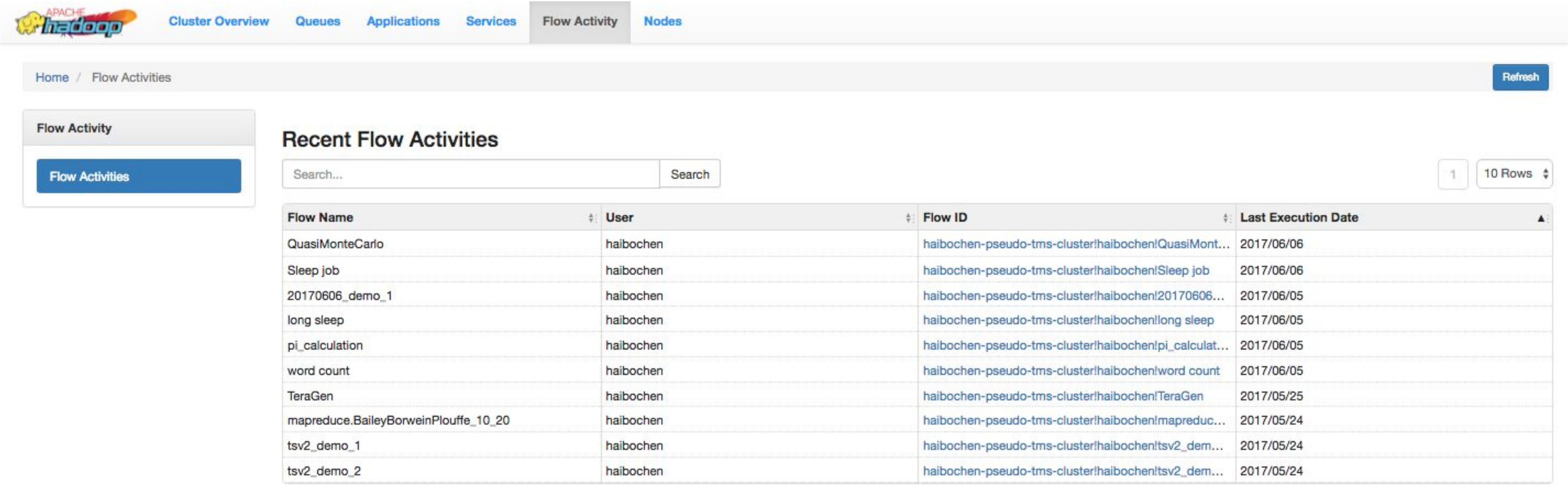
Application Master

Application Timeline Service v2 Flows



cloudera

Application Timeline Service v2 Flows

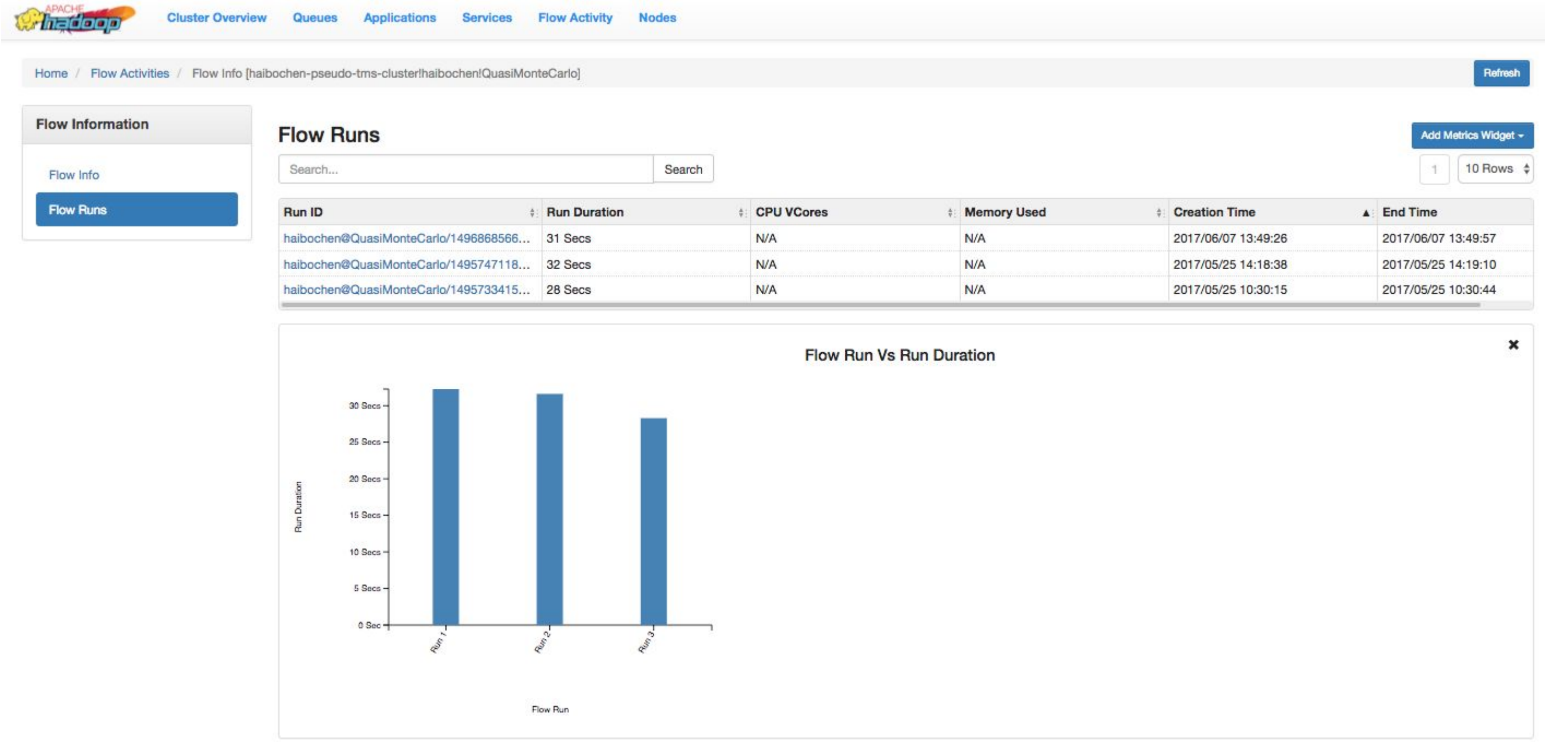


Recent Flow Activities

Flow Name	User	Flow ID	Last Execution Date
QuasiMonteCarlo	haibochen	haibochen-pseudo-tms-cluster!haibochen!QuasiMont...	2017/06/06
Sleep job	haibochen	haibochen-pseudo-tms-cluster!haibochen!Sleep job	2017/06/06
20170606_demo_1	haibochen	haibochen-pseudo-tms-cluster!haibochen!20170606...	2017/06/05
long sleep	haibochen	haibochen-pseudo-tms-cluster!haibochen!long sleep	2017/06/05
pi_calculation	haibochen	haibochen-pseudo-tms-cluster!haibochen!pi_calculat...	2017/06/05
word count	haibochen	haibochen-pseudo-tms-cluster!haibochen!word count	2017/06/05
TeraGen	haibochen	haibochen-pseudo-tms-cluster!haibochen!TeraGen	2017/05/25
mapreduce.BaileyBorweinPlouffe_10_20	haibochen	haibochen-pseudo-tms-cluster!haibochen!mapreduc...	2017/05/24
tsv2_demo_1	haibochen	haibochen-pseudo-tms-cluster!haibochen!tsv2_dem...	2017/05/24
tsv2_demo_2	haibochen	haibochen-pseudo-tms-cluster!haibochen!tsv2_dem...	2017/05/24

cloudera

Application Timeline Service v2 Flows



Home / Flow Activities / Flow Info [haibochen-pseudo-tms-cluster!haibochen!QuasiMonteCarlo]

Flow Information

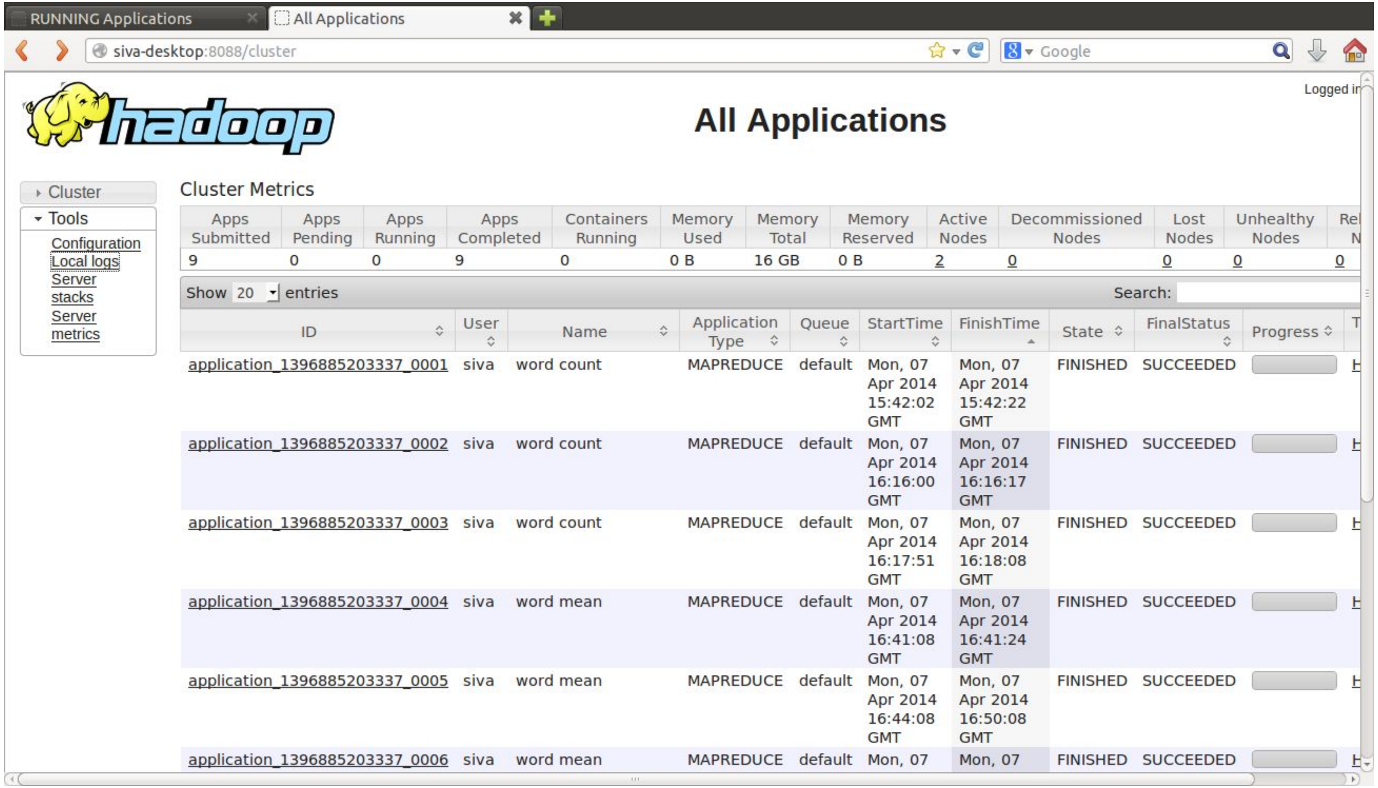
Flow Info

Flow Runs

Flow Runs

Run ID	Run Duration	CPU VCores	Memory Used	Creation Time	End Time
haibochen@QuasiMonteCarlo/1496868566...	31 Secs	N/A	N/A	2017/06/07 13:49:26	2017/06/07 13:49:57
haibochen@QuasiMonteCarlo/1495747118...	32 Secs	N/A	N/A	2017/05/25 14:18:38	2017/05/25 14:19:10
haibochen@QuasiMonteCarlo/1495733415...	28 Secs	N/A	N/A	2017/05/25 10:30:15	2017/05/25 10:30:44

Old YARN UI



All Applications

Cluster Metrics

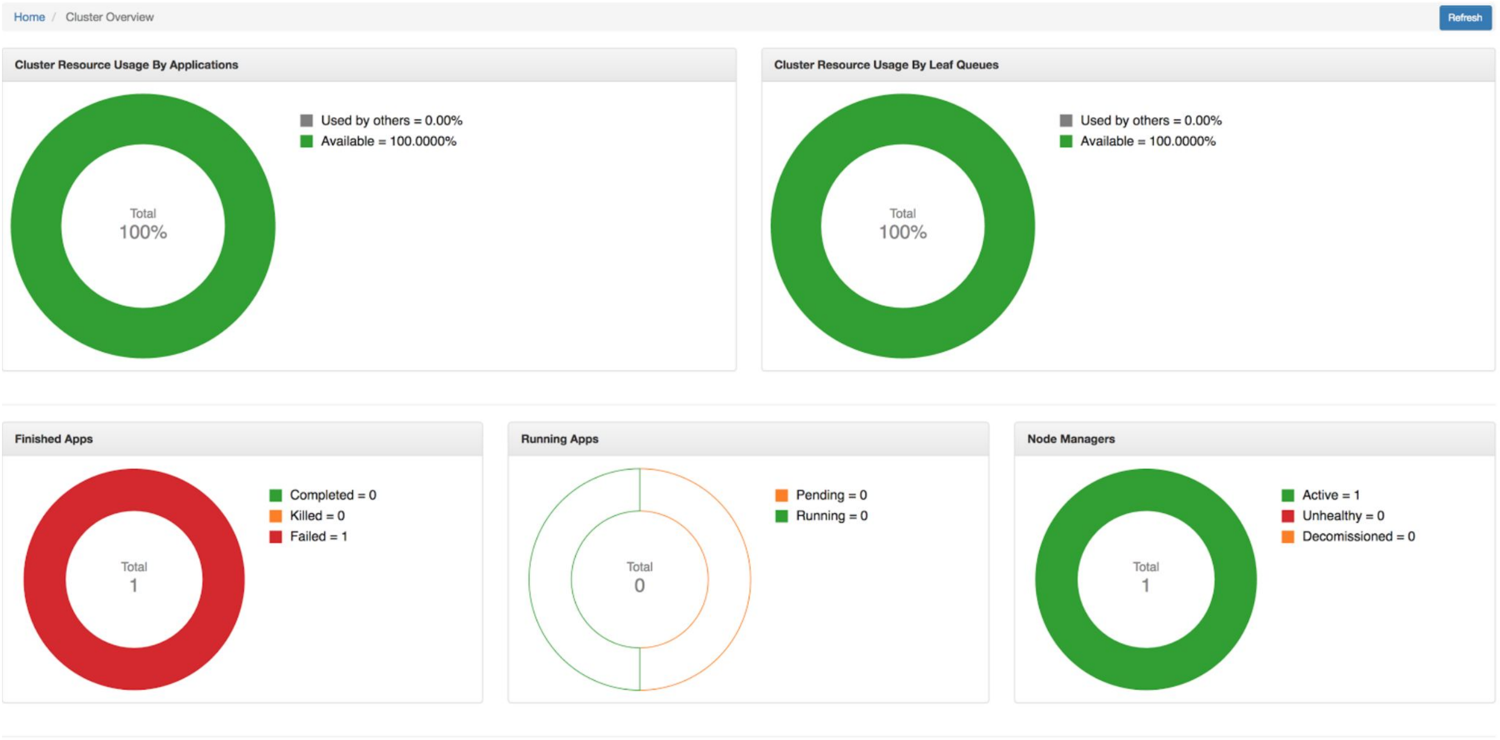
Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Re N
9	0	0	9	0	0 B	16 GB	0 B	2	0	0	0	0

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	T
application_1396885203337_0001	siva	word count	MAPREDUCE	default	Mon, 07 Apr 2014 15:42:02 GMT	Mon, 07 Apr 2014 15:42:22 GMT	FINISHED	SUCCEEDED		H
application_1396885203337_0002	siva	word count	MAPREDUCE	default	Mon, 07 Apr 2014 16:16:00 GMT	Mon, 07 Apr 2014 16:16:17 GMT	FINISHED	SUCCEEDED		H
application_1396885203337_0003	siva	word count	MAPREDUCE	default	Mon, 07 Apr 2014 16:17:51 GMT	Mon, 07 Apr 2014 16:18:08 GMT	FINISHED	SUCCEEDED		H
application_1396885203337_0004	siva	word mean	MAPREDUCE	default	Mon, 07 Apr 2014 16:41:08 GMT	Mon, 07 Apr 2014 16:41:24 GMT	FINISHED	SUCCEEDED		H
application_1396885203337_0005	siva	word mean	MAPREDUCE	default	Mon, 07 Apr 2014 16:44:08 GMT	Mon, 07 Apr 2014 16:50:08 GMT	FINISHED	SUCCEEDED		H
application_1396885203337_0006	siva	word mean	MAPREDUCE	default	Mon, 07	Mon, 07	FINISHED	SUCCEEDED		H

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



Cluster Resource Usage By Applications

Total 100%

- ■ Used by others = 0.00%
- 🟩 Available = 100.0000%

Cluster Resource Usage By Leaf Queues

Total 100%

- ■ Used by others = 0.00%
- 🟩 Available = 100.0000%

Finished Apps

Total 1

- 🟩 Completed = 0
- 🟧 Killed = 0
- 🟥 Failed = 1

Running Apps

Total 0

- 🟧 Pending = 0
- 🟩 Running = 0

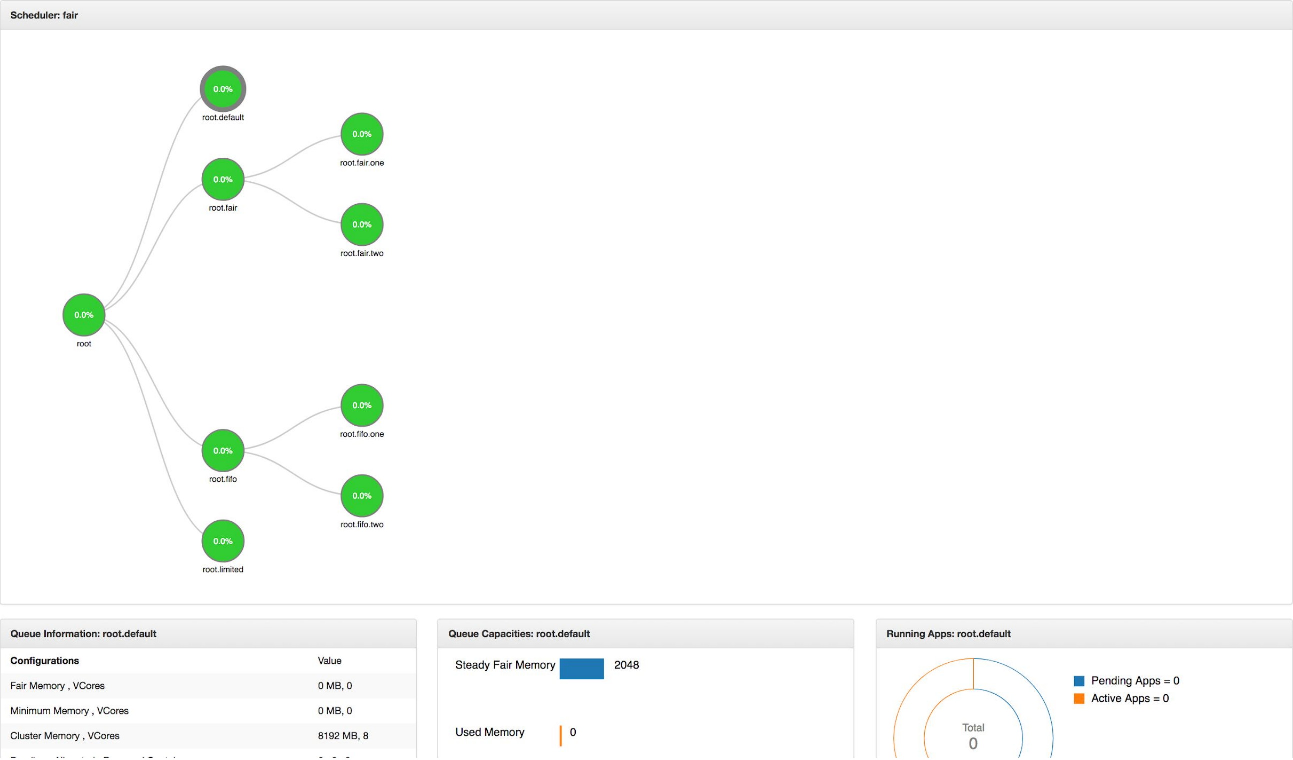
Node Managers

Total 1

- 🟩 Active = 1
- 🟥 Unhealthy = 0
- 🟧 Decomissioned = 0

cloudera

New YARN UI: Queues



Scheduler: fair

Queue Information: root.default

Configurations	Value
Fair Memory , VCores	0 MB, 0
Minimum Memory , VCores	0 MB, 0
Cluster Memory , VCores	8192 MB, 8

Queue Capacities: root.default

Steady Fair Memory 2048

Used Memory 0

Running Apps: root.default

🟦 Pending Apps = 0

🟧 Active Apps = 0

cloudera

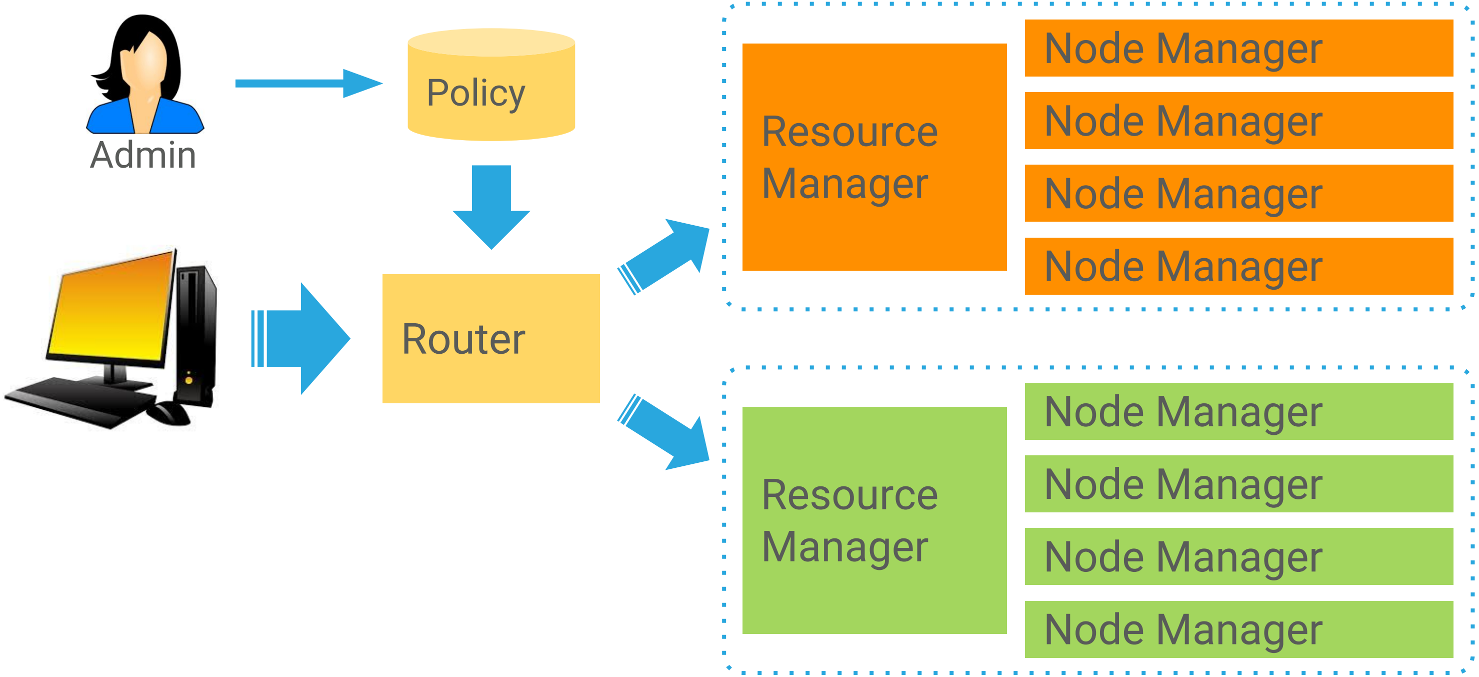
Resource Types

- 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

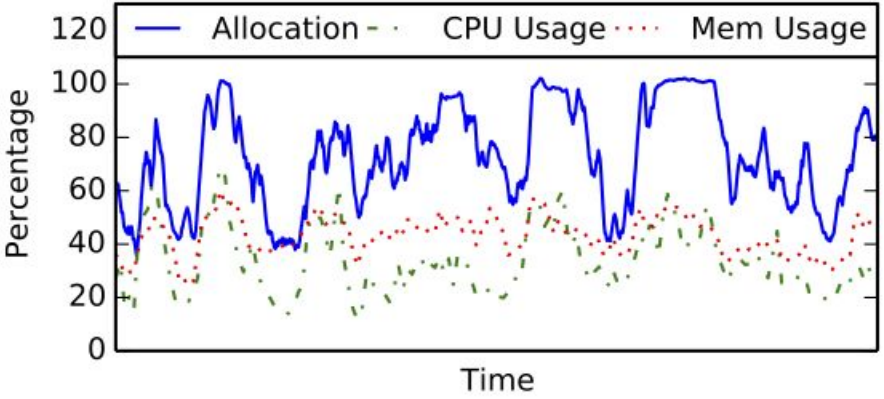


Opportunistic Containers

- 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



cloudera

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: ATSv2
    - 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