High Availability
HDFS

Matt Foley
Hortonworks, Inc.
mfoley@hortonworks.com
**Matt Foley - Background**

**• MTS at Hortonworks Inc.**
  – HDFS contributor, part of original ~25 in Yahoo! spin-out of Hortonworks
  – Currently managing engineering infrastructure for Hortonworks
  – My team also provides Build Engineering infrastructure services to ASF, for Hadoop core and several related projects within Apache
  – Formerly, led software development for back end of Yahoo Mail for three years – 20,000 servers with 30 PB of data under management, 400M active users
  – Did startups in Storage Management and Log Management

**• Apache Hadoop, ASF**
  – Committer and PMC member, Hadoop core
  – Release Manager – Hadoop-1.0
In 2006, Yahoo! was a very early adopter of Hadoop, and became the principle contributor to it.

Over time, invested 40K+ servers and 170PB storage in Hadoop

Over 1000 active users run 5M+ Map/Reduce jobs per month

In 2011, Yahoo! spun off ~25 engineers into Hortonworks, a company focused on advancing open source Apache Hadoop for the broader market (http://www.wired.com/wiredenterprise/2011/10/how-yahoo-spawned-hadoop)
Agenda

• Overview of HDFS architecture
• Hadoop “ecosystem”
• Hadoop 2.0

• High Availability
• What has been the HDFS record?
  – reliability
  – availability
• HDFS-HA
What is Hadoop?

- **Hadoop - Open Source Apache Project**
  - Framework for reliably storing & processing petabytes of data using *commodity* hardware and storage

- **Scalable solution**
  - Computation capacity
  - Storage capacity
  - I/O bandwidth

- **Core components**
  - HDFS: Hadoop Distributed File System - distributes data
  - Map/Reduce - distributes application processing and control

- **Move computation to data and not the other way**

- **Written in Java**

- **Runs on**
  - Linux, Windows, Solaris, and Mac OS/X
Typically in 2- or 3-level architecture
- Nodes are commodity Linux servers
- 20 - 40 nodes/rack
- Uplink from rack is 10 or 2x10 gigabit
- Rack-internal is 1 or 2x1 gigabit all-to-all

“Flat fabric” 10Gbit network architectures being planned at growing number of sites
Hadoop Distributed File System (HDFS)

• One PB-scale file system for the entire cluster
  – Managed by a single Namenode
  – Files are written, read, renamed, deleted, but append-only
  – Optimized for streaming reads of large files

• Files are broken into uniform sized blocks
  – Blocks are typically 128 MB (nominal – no wasted space)
  – Replicated to several Datanodes, for reliability
  – Exposes block placement so that computation can be migrated to data

• Client library directly reads data from Data Nodes
  – Bandwidth scales linearly with the number of nodes
  – System is topology-aware
  – Array of block locations is available to clients
HDFS Diagram

Hierarchical Namespace
File Name ➔ BlockIDs

Heartbeats & Block Reports

Block ID ➔ Block Locations

Block ID ➔ Data

Horizontally Scale IO and Storage
Block Placement

• Default is 3 replicas, but settable

• Blocks are placed (writes are pipelined):
  – First replica on the local node or a random node on local rack
  – Second replica on a remote rack
  – Third replica on a node on same remote rack
  – Other replicas randomly placed

• Clients read from closest replica
  – System is topology-aware

• Block placement policy is pluggable
Block Correctness

• **Data is checked with CRC32**

• **File Creation**
  – Client computes block checksums
  – DataNode stores the checksums

• **File access**
  – Client retrieves the data and checksum from DataNode
  – If Validation fails, Client tries other replicas

• **Periodic validation by DataNode**
  – Background DataBlockScanner task
HDFS Data Reliability

1. replicate

2. copy

3. blockReceived

Bad/lost block replica

Namespace State

Block Map

Namenode

Datanodes

b1  b2  b3  b4

b1  b2  b3  b5

b1  b2  b3  b5

b6  b5  b4
Active Data Management

- Continuous replica maintenance
- End-to-end checksums
- Periodic checksum verification
- Decommissioning nodes for service
- Balancing storage utilization
Other Hadoop Ecosystem Components

- **Core Apache Hadoop**
  - Zookeeper (Coordination)
  - HBase (Columnar NoSQL Store)
  - HCatalog (Table & Schema Management)
  - MapReduce (Distributed Programming Framework)
  - HDFS (Hadoop Distributed File System)

- **Related Hadoop Projects**
  - Hive (SQL)
  - Pig (Data Flow)

---

*Architecting the Future of Big Data*
Agenda

• Overview of HDFS architecture
• Hadoop “ecosystem”
• Hadoop 2.0

• High Availability
• What has been the HDFS record?
  – reliability
  – availability
• HDFS-HA
Hadoop 2.0

• Developed on Hadoop branch 0.23
• Highlights:
  – HDFS Namenode HA
  – HDFS Namenode Federation
  – Next-Generation MapReduce architecture (aka YARN)
  – Performance
HDFS Federation in v2.0

- Improved scalability and isolation
- Clear separation of Namespace and Block Storage
MapReduce2 - YARN

Diagram showing the relationship between Client, Resource Manager, Node Manager, App Mstr, and Container.
Agenda

• Overview of HDFS architecture
• Hadoop “ecosystem”
• Hadoop 2.0

• High Availability
• What has been the HDFS record?
  – reliability
  – availability
• HDFS-HA
Current HDFS Reliability & Availability

• **Block store – extremely high**
  – Block replicas stored in native FS on multiple nodes
    – Transparently ensure that blocks stay replicated
    – Serve from closest available replica
    – A lost node with 12 TB can be re-replicated in 7 minutes
    – A single lost disk of 1TB can be re-replicated in 30 seconds
  – With standard 3x replication, probability of data loss due to normal rates of server and disk failure is infinitesimally small
    – even assuming very casual approach to parts replacement
    – In study of 2009 data, lost 19 blocks out of 329M on 20,000 nodes, due to software bugs that have since been fixed.
Current HDFS Reliability & Availability

• **Meta-data store**
  – Single NameNode stores state
  – Journaling and snapshot management to assure data persistence, to multiple local and NFS (HA) stores
  – But SPOF with manual switch-over on failure

• **How well did it work?**
  – 18 month study of 25 clusters had 22 NN failures
    – Only 8 of them would have been helped with HA
  – Impacted availability, but never durability.
HA: Approach and Terminology

• Initial goal is Active-Standby
  – *Active Namenode*: actively serves read/write operations from clients
  – *Standby Namenode*: waits, becomes active when Active Namenode fails
    – Could serve read operations

• Standby’s State may be cold, warm or hot
  – *Cold*: Standby has zero state (e.g. started after the Active is declared dead.
  – *Warm*: Standby has partial state:
    – has loaded fsImage & editLogs but has not received any block reports
    – has loaded fsImage and rolled logs and all block reports
  – *Hot Standby*: Standby has all most of the Active’s state and start immediately
High Level Use Cases

• **Planned downtime**
  – Upgrades
  – Config changes
  – Main reason for downtime

• **Unplanned downtime**
  – Hardware failure
  – Server unresponsive
  – Software failures
  – Occurs infrequently

• **Supported failures**
  – Single hardware failure
    – Double hardware failure not supported
  – Some software failures
    – Same software failure affects both active and standby
Deployment Models

• Single Namenode configuration; no failover
• Active and Standby with manual failover
  – Standby could be cold/warm/hot
• Active and Standby with automatic failover
  – Hot standby

• See HDFS-1623 for detailed use cases
Design

• Failover control outside Namenode
• Parallel Block reports to Active and Standby (Hot failover)
• Shared or non-shared Namenode state
• Fencing of shared resources/data
  – Datanodes
  – Shared Namenode state (if any)
• Client failover
  – IP Failover
  – Smart clients (e.g. Zookeeper for coordination)
**Failover Control Outside Namenode**

- **Failover Controller**
  - outside Namenode

- **Daemon manages resources**
  - All resources modeled uniformly
  - Resources – OS, HW, Network etc.
  - Namenode is just another resource

- **Heartbeat with other nodes**

- **Quorum based leader election**
  - Zookeeper for co-ordination and Quorum

- **Fencing during split brain**
  - Prevents data corruption
HA Namenode with ZooKeeper

FailoverController
Active

ZK

Heartbeat

ZK

Heartbeat

FailoverController
Standby

Monitor Health of NN. OS, HW

Cmds

NN
Active

NN
Standby

Monitor Health of NN. OS, HW

DN

DN

DN
Sharing the Namenode’s Persistent State

*medium term – 6 month timeframe*

### Shared Storage Approach

**NN Active**

**NN Standby**

- Shared NN state with single writer (fencing)

**Edit Logs**

**Direct stream to Standby NN**
Sharing the Namenode’s Persistent State

*long term*

Store NN journal and checkpointed image on Datanodes
Hadoop 2.0 “Availability” (in the field)

• Requires LOTS of testing

• In small-scale test (500-800 nodes) 2Q2012

• Ramping up over rest of year, with full range of application testing

• Expected to be in production at multiple sites by end/2012
Credits

For major contributions to Hadoop technology, and help with this presentation:

• **Sanjay Radia and Suresh Srinivas, Hortonworks**
  – Architect and Team Lead, HDFS
  – HA and Federation

• **Owen O’Malley, Hortonworks**
  – Hadoop lead Architect
  – Security, Map/Reduce

• **Arun Murthy, Hortonworks**
  – Architect and Team Lead, Map/Reduce
  – M/R2, YARN, etc.

• **Rob Chansler, Yahoo!**
  – Team Lead, HDFS
  – Analysis of Data Availability and Durability
Help getting started

• **Apache Hadoop Projects**

• **Apache Hadoop Email lists:**
  – common-user@hadoop.apache.org
  – hdfs-user@hadoop.apache.org
  – mapreduce-user@hadoop.apache.org

• **O’Reilly Books**
  – Hadoop, The Definitive Guide
  – HBase, The Definitive Guide

• **Hortonworks, Inc.**
  – Installable Data Platform distribution (100% OSS, conforming to Apache releases)
  – Training and Certification programs

• **Hadoop Summit 2012 (June 13-14, San Jose)**
Thanks for Listening!

Questions?