Managing the System Lifecycle and Configuration of Apache Hadoop and Other Distributed Systems

Philip Zeyliger
Cloudera
Velocity 2011
6/14/2011
$whoami
philip

• Engineer at Cloudera (since ‘08)
• Past life working on Megastore at Google (CIDR ‘11 for paper; adapts BigTable for user-facing apps)

@philz42
philip@cloudera.com
$whoareyou?

- CTOs?
- Architects?
- Data Scientists?
- Devs & Engineers?
- Devops?
- Managers?
- Sysadmins?

- Hadoop novices?
- Hadoop experts?
## Agenda

<table>
<thead>
<tr>
<th>Background</th>
<th>Why should we care?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position</td>
<td>What’s different about distributed systems?</td>
</tr>
<tr>
<td>Implementation</td>
<td>One possible solution, with details.</td>
</tr>
<tr>
<td>Hadoop</td>
<td>Tips, tricks, and best practices for managing Apache Hadoop specifically.</td>
</tr>
</tbody>
</table>

*meta: interrupt me; ask me questions; etc.*
Background:
Why Apache Hadoop?
We’re storing a lot more data.
Dramatic Changes in Enterprise Data Management

Big Data
• Any kind
• From any source
• At scale

Hard Problems
• Deep analysis
• Exhaustive and detailed
• Run sophisticated algorithms in parallel and generate useful results quickly
Exploding Data Volumes

• Online
  • Web-ready devices
  • Social media
  • Digital content
  • Smart grids

• Enterprise
  • Transactions
  • R&D data
  • Operational (control) data

Source: An IDC White Paper - sponsored by EMC. As the Economy Contracts, the Digital Universe Expands. May 2009

Digital universe grew by 62% last year to 800K petabytes and will grow to 1.2 “zetabytes” this year

2,500 exabytes of new information in 2012 with Internet as primary driver
Hadoop is a good way to deal with this.
What is Apache Hadoop?

Open Source Storage and Processing Engine

- **Consolidates Everything**
  - Move complex and relational data into a single repository

- **Stores Inexpensively**
  - Keep raw data always available
  - Use commodity hardware

- **Processes at the Source**
  - Eliminate ETL bottlenecks
  - Mine data first, govern later
Processes across your cluster...

Maintains state on local FS

Services

Single Master (special)
Multi-Master
Utility
Common
Processes across your cluster...

HDFS is the Distributed File System
Namenode: holds the metadata
Datanodes: hold the blocks
**Processes across your cluster...**

- **HDFS**
  - Namenode
  - DataNode
- **MapReduce**
  - JobTracker
  - TaskTracker
- **ZooKeeper**
  - ZK Server
- **HBase**
  - Region Server

---

**MapReduce is the computation scheduler.**

**JobTracker is the scheduler.**

**TaskTrackers shepherd the jobs.**

- **Single Master (special!)**
  - Multi-Master
  - Utility
  - Common

**Maintains state on local FS**

---

Copyright 2011 Cloudera Inc. All rights reserved.
Processes across your cluster...

HDFS is the Distributed File System.
- Namenode: holds the metadata
- Datanodes: hold the blocks

MapReduce is the compute scheduler.
- JobTracker: is the scheduler.
- TaskTrackers: shepherd the jobs.

HBase is the distributed key-value store.
- Masters assign regions.
- RegionServers serve them.

Maintains state on local FS

Single Master (special)
Multi-Master
Utility
Common
Note the growth...

- When Cloudera started, we were just worrying about HDFS and MapReduce
- Now HBase is just as important
- Next year?
Not that we’re surprised...

...if “the datacenter is the computer,” it better have a toolchain and ecosystem.
Background

Cloud Computing
Background

Cloud Computing

Datacenter Computing
Datacenter Computing implies Distributed Systems
What are distributed systems?

“collection of independent computers that appear to the users of the system as a single computer.” (Tanenbaum)

The core contribution of Map/Reduce is that the user doesn’t have to be an expert in distributed systems to use it.
Hadoop separates distributed system fault-tolerance code from application logic.
Desirable Properties

• Fault-tolerant
• Highly Available
• Recoverable
• Consistent
• Scalable
• Predictably Performant (linearly?)
• Secure

Most systems don’t give you all of these all the time, but it’s nice to dream.
Many Flavors

- p2p (BitTorrent)
- # clients >> # servers >> 1 (email, chat)
- Hierarchical (DNS)
- and more...

- within-datacenter
Easier to do these days...

- Rent some machines from Amazon, Rackspace, Softlayer, and a dozen other folks...

- Meanwhile, getting harder to just buy a bigger, faster machine. (Clock speeds aren’t increasing quite as much any more.)
Easy Easier Distributed Systems

• Stateless
• e.g., caches and mirrors
Stateful Distributed Systems

• Local Disk used; harder to move.
• If you screw up a database, people have time to come after you...
• Lifecycle issues are more interesting here.
• HDFS is stateful
Evolution of Distributed Systems in your Data Center

Homebrew Distributed Systems

Other People’s Distributed Systems

Lots of Distributed Systems

your web tier?

Hadoop?
<table>
<thead>
<tr>
<th>Agenda</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Background</strong></td>
<td>Why should we care?</td>
</tr>
<tr>
<td><strong>Position</strong></td>
<td>What’s different about distributed systems?</td>
</tr>
<tr>
<td><strong>Implementation</strong></td>
<td>One possible solution, with details.</td>
</tr>
<tr>
<td><strong>Hadoop</strong></td>
<td>Tips, tricks, and best practices for managing Apache Hadoop specifically.</td>
</tr>
</tbody>
</table>
My Position

Short version: your existing tools don’t generalize to distributed systems.

Distributed systems (especially those that run on 20+ machines) are sufficiently different that traditional approaches to configuration and lifecycle management aren’t appropriate.
Lifecycle

- Pesky host stuff
- Install
- Config
- Start/Stop
We have lots of existing tools!
Pesky host stuff

- Do something sensible: PXE, Cobbler, Kickstart, etc.
- Our customers often don’t have a choice: datacenter ops teams mandate something.
## Install

<table>
<thead>
<tr>
<th>Method</th>
<th>Command(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traditional</td>
<td><code>tar -xzvf &amp;&amp; ./configure &amp;&amp; make install</code></td>
</tr>
<tr>
<td>Package Based</td>
<td>rpm, dpkg, zypper, msi</td>
</tr>
<tr>
<td>File Transfer</td>
<td>rsync, sftp</td>
</tr>
<tr>
<td>Installer</td>
<td><code>./install.sh</code></td>
</tr>
</tbody>
</table>
## Configure

<table>
<thead>
<tr>
<th>Editors</th>
<th>vi, emacs, etc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Transfer/Sync</td>
<td>rsync, scp, ftp</td>
</tr>
<tr>
<td>Configuration Management</td>
<td>puppet, chef, cfengine, bcfg2</td>
</tr>
<tr>
<td>Source Control</td>
<td>git, svn, cvs, perforce, rcs</td>
</tr>
<tr>
<td>Start/Stop Tools</td>
<td></td>
</tr>
<tr>
<td>--------------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td><strong>Supervision</strong></td>
<td>daemontools, supervisord, monit</td>
</tr>
<tr>
<td><strong>Start-up Scripts</strong></td>
<td>init.d (Sys V), Solaris Service Management Facility, Windows “Services”, launchd, runit, systemd, upstart</td>
</tr>
<tr>
<td><strong>Unix process control</strong></td>
<td>bash</td>
</tr>
<tr>
<td><strong>Application Containers</strong></td>
<td>Tomcat, JBoss, ...</td>
</tr>
</tbody>
</table>
Long live the Unix toolchain!
Q: With all those tools, why are we having this conversation?
Q: With all those tools, why are we having this conversation?

A: Distributed systems are different.
Where are fork(2) and execve (2) in distributed systems, anyway?
Differences between traditional services and distributed services?

- Partial failure
  If you have 500 machines, and a datanode (one of 498) gives up the ghost at 4am, you ought to:
  (a) Wake up when the pager rings; start frantically recovering it.
  (b) Dismiss the page, and go back to sleep.
  (c) Find out tomorrow that you should fix it next week.

(But if two racks die...)
More differences:

Configuration Divergence

- Cloudera has measured significant configuration divergence within our customers. Also, binary version divergence.
- Latent, hard to discover bugs.
Non-config Config

• Where is the list of machines participating in your system? What other meta-configuration is there?
Dev, Stage, Prod, really?

- If you have a 500-node prod cluster, do you have a staging cluster?
- Is it 500 nodes?
- Does it run your mission-critical jobs?

(Most folks we know have much smaller dev clusters; catches some issues, but not all of them.)
Is it really up?

• Maybe it’s a network partition?
• Maybe it’s just overloaded and hence timing out?
• Maybe...
Shell scripting != panacea

- Do you always check your error codes in shell scripts? Ok, fine, set -e. Do they recover correctly?
- Can you tell whether you did something on 98 machines or 99?
Multi-machine coördination is hard

- Cluster execution is fundamentally async.
- You want to do things in parallel, because serial is too slow.
- “Graceful” restart is actually a rolling restart.
And yet a lot is the same...

- Dependencies
-Configs
- Versions and version compatibility
- Configuration tracking and auditing
So, what do we need?
What can we conclude?
We need...

- Something analogous to packaging, but that understands multiple machines.
- Something to be used by users, not developers of the underlying system.
- One config instead of many.
- Start, stop, and workflows (e.g., rolling upgrades).
Why not machine images?

• Because there’s data on the local disk!
• Heavy-weight.
• Hard to do rolling upgrades and small config changes.
Why not config management tools?

- Excellent for the base OS (e.g., installing Java).
- Aren’t aware of Hadoop.
- Get tripped up on stuff like correct shutdown.
- Make assumptions about machines running N services, not services running on M machines.
So... we need something new.

• The right granularity is still user-land.
• Doesn’t make sense to restrict to one part of the lifecycle. Tackle at least config & supervision/start/stop; preferably deployment too.
• Facilitate higher-order operations.
# Agenda

<table>
<thead>
<tr>
<th>Background</th>
<th>Why should we care?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Position</td>
<td>What’s different about distributed systems?</td>
</tr>
<tr>
<td>Implementation</td>
<td>One possible solution, with details.</td>
</tr>
<tr>
<td>Hadoop</td>
<td>Tips, tricks, and best practices for managing Apache Hadoop specifically.</td>
</tr>
</tbody>
</table>
Implementation

- Cloudera Service and Configuration Manager (“SCM”)
- Available for Cloudera customers to help them manage their Hadoop deployments
Motivations

• Every Cloudera customer deploys Hadoop slightly differently.
• Many problems are preventable; e.g., inappropriate configs.
• Several tasks are error-prone, multi-step processes.
• Hadoop’s learning curve is steep.
• Configurations piecemeal instead of global.
Key Use cases

- Changing a config
- Gracefully shutting down HBase
- Adding nodes to the cluster
Top-level
Config

Dependencies!
Drill-down and Detail
## SCM Vocabulary

<table>
<thead>
<tr>
<th>Class</th>
<th>Instance</th>
<th>Definition</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Type</td>
<td>Service Instance</td>
<td>a managed system, typically distributed</td>
<td>HDFS, MR, HBase</td>
</tr>
<tr>
<td>Role Type</td>
<td>Role Instance</td>
<td>A component of a service that runs on a single machine.</td>
<td>NameNode, DataNode, RegionServer</td>
</tr>
<tr>
<td>Host</td>
<td></td>
<td>a unique datacenter machine</td>
<td>m003.sf.example.com</td>
</tr>
</tbody>
</table>
“casting”

- “casting” is the process of assigning roles to machines.
- In SCM, “casting” is static: the user tells the system what machine does what. (You could also imagine a “scheduler” which dynamically adjusts what goes where.)
What’s it do?

• Process Supervision
  What processes are running where?

• Configuration Management
  What configs the different processes should have; validation; dependency management.

• Commands and workflows
  (format, rebalance, graceful shutdown, etc.)
How’s it work?

- Agents ask server “what should I be doing?”
- Server keeps track of who’s doing what.

<table>
<thead>
<tr>
<th>Data Model</th>
<th>Admin Console/ Web Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Health Calculations</td>
<td>Workflow Execution</td>
</tr>
<tr>
<td>Configuration Generation</td>
<td>Validations</td>
</tr>
<tr>
<td>Heartbeat Server</td>
<td>Security, Audit, etc.</td>
</tr>
</tbody>
</table>

Process Supervision

Basic Stats Collection

Heartbeating

Server (one of these)

Agent (many of these)
Does it work?

• Yep.
A bit more detail...

- Use packages to install the Hadoop binaries. Disk is cheap; install everything everywhere.
- Configurations are transient on the managed machines; sent as part of heartbeat. Avoids config creep.
- “Commands” manipulate the state of the system
Models & Observations
America’s Next Top Model?

• Processes are:
  • executable + arguments
  • environment
  • config directory

• But also...
  • log locations, directories used, ports open
Observations

• Observe the state of the system
• Compare with the model
• Note if there are differences
it all adds up to packaging+init.d
for distributed systems
Is the approach general?

- Our sample size is > 3, so, probably.
- Idiosyncrasies have definitely cropped up.
Related Work

• Google’s cluster manager
• procfile & foreman
• LinkedIn’s glu

Distributed supervision was actually the easy part; making the Hadoop stuff work was the complex part.
Monitoring

- Cloudera’s Activity Monitor -- like “top” for Hadoop Map/Reduce
### Cluster Activity

Activity Monitor


![Activity Monitor Graph](image)

### Activities

<table>
<thead>
<tr>
<th>Name</th>
<th>Id</th>
<th>Type</th>
<th>Status</th>
<th>User</th>
<th>Map Input Records</th>
<th>Reducer Output Records</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activity Monitor Test</td>
<td>job_201106082045_0006</td>
<td>Map/Reduce</td>
<td>✔️</td>
<td>henry</td>
<td>124456</td>
<td>901325</td>
</tr>
<tr>
<td>Activity Monitor Test</td>
<td>job_201106082045_0005</td>
<td>Map/Reduce</td>
<td>✗</td>
<td>henry</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PigLatin:Script2-hadoop.pig</td>
<td>pig_74926352-a43d-3af6-a0e8-733885a95029</td>
<td>Pig</td>
<td>✔️</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PIEstimator</td>
<td>job_201106082045_0001</td>
<td>Map/Reduce</td>
<td>✔️</td>
<td>henry</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>PIEstimator</td>
<td>job_201106021435_0004</td>
<td>Map/Reduce</td>
<td>✔️</td>
<td>nightly</td>
<td>100</td>
<td>0</td>
</tr>
</tbody>
</table>
Resource Accounting

![Disk Usage by Users, Number of Files, Bytes, Raw Bytes](image)
BTW, if this part is interesting...

we’re hiring!

philip@cloudera.com
<table>
<thead>
<tr>
<th>Agenda</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Background</strong></td>
<td>Why should we care?</td>
</tr>
<tr>
<td><strong>Position</strong></td>
<td>What’s different about distributed systems?</td>
</tr>
<tr>
<td><strong>Implementation</strong></td>
<td>One possible solution, with details.</td>
</tr>
<tr>
<td><strong>Hadoop</strong></td>
<td>Tips, tricks, and best practices for managing Apache Hadoop specifically.</td>
</tr>
</tbody>
</table>
Planning

• How big?
• You can grow more later
• Storage capacity is a good first approximation (don’t forget typical 3x replication factor)
• How much data are you ingesting per day?
Casting

- For small clusters (<10 nodes), you can locate multiple “masters” on one machine.
- For bigger clusters, separate them.
Hardware & Networking

- NameNode and JobTracker often on beefier hardware
- Configure disks as JBOD
- Gigabit Ethernet
- Top of rack switches
- Avoid virtualization
A Typical Look...

- 5-4000 commodity servers
  (8-core, 24GB RAM, 4-12 TB, gig-E)
- 2-level network architecture
- 20-40 nodes per rack
Software

- CentOS 5/RHEL 5 is the most common
- Oracle JDK (avoid 1.6.0u18)
OS Config

- Mount noatime
- Tweak ulimit
- Tweak swappiness
Software Install

- I highly recommend Cloudera’s Distribution including Apache Hadoop, version 3 (CDH3).
- .rpm and .deb packages; yum install and go!
Why CDH?

- **Open source** – 100% Apache licensed and free for download
- **Simplified** – Component versions & dependencies managed for you
- **Integrated** – All components & functions interoperate through standard API’s
- **Reliable** – Patched with fixes from future releases to improve stability
- **Supported** – Employs project founders and committers for >90% of components
Configs

• Use source control
• For simplicity, people tend to use the same config on all machines.
• XML files. core-site.xml, hdfs-site.xml, and mapred-site.xml. Also, hadoop-env.sh.
• Monitor for divergence
## Important Configs: where’s the data?

<table>
<thead>
<tr>
<th>Config</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dfs.name.dir</td>
<td>NameNode. Typically two volumes + NFS (mounted correctly).</td>
</tr>
<tr>
<td>dfs.data.dir</td>
<td>DataNodes. One directory per physical disk.</td>
</tr>
<tr>
<td>fs.checkpoint.dir</td>
<td>Secondary NameNode dir.</td>
</tr>
<tr>
<td>mapred.local.dir</td>
<td>MapReduce. One directory per physical disk.</td>
</tr>
</tbody>
</table>
## Important Configs: Sizing

<table>
<thead>
<tr>
<th>Config</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dfs.block.size</td>
<td>Block size for HDFS. We recommend 128MB.</td>
</tr>
<tr>
<td>mapred.tasktracker.map.tasks.maximum</td>
<td>Max number of maps per machine.</td>
</tr>
<tr>
<td>mapred.tasktracker.reduce.tasks.maximum</td>
<td>Max number of reduces per machine.</td>
</tr>
</tbody>
</table>
Common Issues

- DNS. Hadoop requires it.
- Java. Oracle JVM is the way to go.
- Secondary name node not checkpointing.
Monitoring

- Hadoop exposes a bit via HTTP /metrics, JMX, gnaglia.
- Cloudera’s Activity Monitor is like “top” for your cluster.
- Sick is worse than dead.
More?

- Cloudera training for sysadmins (and managers) (and developers) much better than me.
- Lots of training around Hadoop Summit at end of June (in San Jose). (Training available world-wide.)
- http://www.cloudera.com/hadoop-training/
Training Schedule

Use code “velocity” for 10% off

<table>
<thead>
<tr>
<th>Date</th>
<th>Course</th>
<th>Registration Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>June 27-28</td>
<td>Developer</td>
<td><a href="http://www.eventbrite.com/event/1639045429">http://www.eventbrite.com/event/1639045429</a></td>
</tr>
<tr>
<td>June 28</td>
<td>Essentials for Managers</td>
<td><a href="http://www.eventbrite.com/event/1639127675">http://www.eventbrite.com/event/1639127675</a></td>
</tr>
<tr>
<td>June 30</td>
<td>HBase</td>
<td><a href="http://www.eventbrite.com/event/164080669">http://www.eventbrite.com/event/164080669</a></td>
</tr>
<tr>
<td>June 30-July 1</td>
<td>Hive and Pig</td>
<td><a href="http://www.eventbrite.com/event/1640792655">http://www.eventbrite.com/event/1640792655</a></td>
</tr>
<tr>
<td>June 30-July 1</td>
<td>Administrator</td>
<td><a href="http://www.eventbrite.com/event/1640780619">http://www.eventbrite.com/event/1640780619</a></td>
</tr>
<tr>
<td>Background</td>
<td>Why should we care?</td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>------------------</td>
<td></td>
</tr>
<tr>
<td>Position</td>
<td>What’s different about distributed systems?</td>
<td></td>
</tr>
<tr>
<td>Implementation</td>
<td>One possible solution, with details.</td>
<td></td>
</tr>
<tr>
<td>Hadoop</td>
<td>Tips, tricks, and best practices for managing Apache Hadoop specifically.</td>
<td></td>
</tr>
</tbody>
</table>
Thanks!, Questions?, & Quick Feedback:

Good

Bad

And, of course, don’t hesitate to e-mail philip@cloudera.com
Backup Slides

because I don’t go anywhere
without some extra pictures
Patterns and Anti-patterns

- Web-based status pages
- Well-defined configs. Centralized configs.
- RPC Definitions, compatibility.
- Monitoring

- Dev-centric approach. (Defaults good for dev, but not prod.)
- Lack of client/server separation.
HDFS Data Storage

NameNode

158MB

/logs/weblog.txt

DN 1

DN 2

DN 3

DN 4

Wednesday, June 15, 2011
HDFS Write Path

1: create
2: create
3: write
4: write packet
5: ack packet
6: close
7: complete

client JVM
client node

Distributed FileSystem

NameNode

FSDataOutputStream

datanode
dataNode
dataNode
dataNode

pipeline of datanodes
Putting it together...