Detailed Role Assignments

- Very Small (Up to 10 workers, No High Availability)
  - 1 Master Node
    - NameNode, YARN Resource Manager, Job History Server, ZooKeeper, Impala StateStore
  - 1 Utility/Edge Node
    - Secondary NameNode, Cloudera Manager, Hive Metastore, HiveServer2, Impala Catalog, Hue, Oozie, Flume, Relational Database, Gateway configurations
  - 3-10 Worker Nodes
    - DataNode, NodeManager, Impalad, Llama
Detailed Role Assignments

- Small (Up to 20 workers, High Availability)
  - 2 Master Nodes
    - NameNode (with JournalNode and FailoverController), YARN Resource Manager, ZooKeeper
    - (1 Node each) Job History Server, Impala StateStore
  - 1 Utility/Edge Node
    - Cloudera Manager, Hive Metastore, HiveServer2, Impala Catalog, Hue, Oozie, Flume, Relational Database, Gateway configurations
    - (requires dedicated spindle) Zookeeper, JournalNode
  - 3-20 Worker Nodes
    - DataNode, NodeManager, Impalad, Llama
Detailed Role Assignments

- Medium (Up to 200 workers, High Availability)
  - 3 Master Nodes
    - (3 Nodes) Zookeeper, JournalNode
    - (2 Nodes each) NameNode (with FailoverController), YARN Resource Manager
    - (1 Node each) Job History Server, Impala StateStore
  - 2 Utility Nodes
    - Node 1: Cloudera Manager, Relational Database
    - Node 2: CM Management Service, Hive Metastore, Catalog Server, Oozie
  - 1+Edge Noded
    - Hue, HiveServer2, Flume, Gateway configuration
  - 50-200 Worker Nodes
    - DataNode, NodeManager, Impalad, Llama
Detailed Role Assignments

- Large (Up to 500 workers, High Availability)
  - 5 Master Nodes
    - (5 Nodes) Zookeeper, JournalNode
    - (2 Nodes) NameNode (with FailoverController)
    - (2 different Nodes) YARN Resource Manager
    - (1 Node each) Job History Server, Impala StateStore
  - 2 Utility Nodes
    - Node 1: Cloudera Manager, Relational Database
    - Node 2: CM Management Service, Hive Metastore, Catalog Server, Oozie
  - 1+Edge Noded
    - Hue, HiveServer2, Flume, Gateway configuration
  - 200-500 Worker Nodes
    - DataNode, NodeManager, Impalad, Llama
# Memory Allocation

<table>
<thead>
<tr>
<th>Item</th>
<th>RAM Allocated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System Overhead</td>
<td>2 GB (minimum)</td>
</tr>
<tr>
<td>DataNode</td>
<td>1-4 GB</td>
</tr>
<tr>
<td>YARN NodeManager</td>
<td>1 GB</td>
</tr>
<tr>
<td>YARN ApplicationManager</td>
<td>1 GB</td>
</tr>
<tr>
<td>YARN Map/Reduce Containers</td>
<td>1-2GB/Container</td>
</tr>
<tr>
<td>HBase RegionServer</td>
<td>4-12 GB</td>
</tr>
<tr>
<td>Impala</td>
<td>128 GB (can be reduced with spill-to-disk)</td>
</tr>
</tbody>
</table>
Apache Hadoop Operations for Production Systems: Configuration

Philip Zeyliger
Agenda

- Mechanics
- Key configurations
- Resource Management
What’s there to configure, anyway?

- On a 100-node cluster, there are likely 400+ (HDFS datanode, Yarn nodemanager, HBase RegionServer, Impalad) processes running. Each has environment variables, config files, and command line options!
- Where your daemons are running are configuration too, but often implicit. Moving from machine A to machine B is a configuration change!
- Most (but not all) settings require a restart to take effect.
- A management tool will help you with “scoping.” Some configurations must be the same globally (e.g., kerberos), some make sense within a service (HDFS Trash), some per-daemon
$ vi /etc/hadoop/conf/hdfs-site.xml

- Configs are key-value pairs, in a straightforward if verbose XML format
- When in doubt, place configuration everywhere, since you might not know whether the client reads it or which daemons read it.
- Dear golly, use configuration management.
Editing a configuration

- Quick demo!

URLs and passwords:

http://tiny.cloudera.com/strata1 (user1/strata2015) (Kerberos)
admin@54.153.123.57 pass: admin321

http://tiny.cloudera.com/strata2 (user2/strata2015)
admin@54.67.90.80 pass: admin321

admin@54.153.83.11 pass: admin321
Step One: edit a config

Step Two: save
Step Three: at top-level, note that restart is needed

Step Four: review changes

Step Five: restart
Show me the files!

- core-site.xml, hdfs-site.xml, dfs_hosts_allow, dfs_hosts_exclude, hbase-site.xml, hive-env.sh, hive-site.xml, hue.ini, mapred-site.xml, oozie-site.xml, yarn-site.xml, zoo.cfg (and so on)
- e.g., /var/run/cloudera-scm-agent/process/*-NAMENODE

Configuration Files:
- core-site.xml
- hadoop-policy.xml
- hdfs-site.xml
- ssl-client.xml
- ssl-server.xml
- cloudera-monitor.properties
- cloudera-stack-monitor.properties
- cloudera_manager_agent_fencer.py
- dfs_hosts_allow.txt
- dfs_hosts_exlude.txt
- event-filter-rules.json
- hadoop-metrics2.properties
- hdfs.keytab
- http-auth-signature-secret
- log4j.properties
- navigator.client.properties
- topology.map
- topology.py

Environment Variables:
- HADOOP_NAMENODE_OPTS=-Xms287309824 -Xmx287309824 -XX:+UseConcMarkSweepGC -XX:-CMSConcurrentMTEnabled -XX:CMSInitiatingOccupancyFraction=70 -XX:+CMSParallelRemarkEnabled -XX:OnOutOfMemoryError="{{AGENT_COMMON_DIR}}/killparent.sh"
- HADOOP_LOGFILE=hadoop-cmf-HDFS-1-NAMENODE-nightly-1.ent.cloud
- HADOOP_AUDIT_LOGGER=INFO,RFAAUDIT
- HADOOP_ROOT_LOGGER=INFO,RFA
- CDH_VERSION=5
- HADOOP_LOG_DIR=/var/log/hadoop-hdfs
- HADOOP_SECURITY_LOGGER=INFO,RFAS
Let’s take a look!

- Demo/Activity: find the files via UI

URLs and passwords:

http://tiny.cloudera.com/strata1 (user1/strata2015) (Kerberos)
admin@54.153.123.57 pass: admin321

http://tiny.cloudera.com/strata2 (user2/strata2015)
admin@54.67.90.80 pass: admin321

admin@54.153.83.11 pass: admin321
How to double-check?

Key Configuration Themes (everyone)

- Security
- Ports
- Heap Sizes
- Local Storage
- JVM (use Oracle JDK 1.7)
- Databases (back them up)
Rack Topology

- Hadoop cares about racks because:
  - Shared failure zone
  - Network bandwidth
- When operating a large cluster, tell Hadoop (by use of a rack locality script) what machines are in which rack.
Networking

- Does your DNS work?
- Like, forwards and backwards?
- For sure?
- Have you really checked?
HDFS Configurations of Note

- **Heap sizes:**
  - Datanodes: linear in # of blocks
  - Namenode: linear in # of blocks and # of files

```sql
select blocks_total, jvm_heap_used_mb
where roletype=DATANODE and hostname RLIKE "hodor-016.*"
```
HDFS Configurations of Note

- Local Data Directories
  - `dfs.datanode.data.dir`: one per spindle; avoid RAID
  - `dfs.namenode.name.dir`: two copies of your metadata better than one
- High Availability
  - Requires more daemons
    - Two failover controllers co-located with namenodes
    - Three journal nodes
Yarn Configurations of Note

- Yarn doles out resources to applications across two axes: memory and CPU
- Define per-NodeManager resources
  - yarn.nodemanager.resource.cpu-vcores
  - yarn.nodemanager.resource.memory-mb
Resource management in YARN

- Fannie and Freddie go in on a large cluster together. Fannie ponies up 75% of the budget; Freddie ponies up 25%.
- When cluster is idle, let Freddie use 100%
- When Fannie has a lot of work, Freddie can only use 25%.
- The “Fair Scheduler” implements “DRF” to share the cluster fairly across CPU and memory resources.
- Configured by an “allocations.xml” file.

<table>
<thead>
<tr>
<th>Name</th>
<th>Weight</th>
<th>%</th>
<th>Virtual Cores Min / Max</th>
<th>Memory Min / Max</th>
<th>Max Running Apps</th>
<th>Scheduling Policy</th>
</tr>
</thead>
<tbody>
<tr>
<td>root</td>
<td>1</td>
<td>100.0%</td>
<td>-/-</td>
<td>-/-</td>
<td>-</td>
<td>DRF</td>
</tr>
<tr>
<td>fannie</td>
<td>3</td>
<td>75.0%</td>
<td>-/-</td>
<td>-/-</td>
<td>-</td>
<td>DRF</td>
</tr>
<tr>
<td>freddie</td>
<td>1</td>
<td>25.0%</td>
<td>-/-</td>
<td>-/-</td>
<td>-</td>
<td>DRF</td>
</tr>
</tbody>
</table>
Quick Demo/Activity

- Look at heap sizes

URLs and passwords:

http://tiny.cloudera.com/strata1 (user1/strata2015)
admin@54.153.123.57 pass: admin321

http://tiny.cloudera.com/strata2 (user2/strata2015)
admin@54.67.90.80 pass: admin321

admin@54.153.83.11 pass: admin321
Activity

What is the current HDFS block size?

URLs and passwords:

http://tiny.cloudera.com/strata1 (user1/strata2015) (H)  
admin@54.153.123.57 pass: admin321

http://tiny.cloudera.com/strata2 (user2/strata2015)  
admin@54.67.90.80 pass: admin321

admin@54.153.83.11 pass: admin321
Hokey-Pokey Break (You put your right CPU in... )
Apache Hadoop Operations for Production Systems: Troubleshooting

Kathleen Ting
Troubleshooting

Managing Hadoop Clusters
Troubleshooting Hadoop Systems
Debugging Hadoop Applications
Troubleshooting

Managing Hadoop Clusters
Troubleshooting Hadoop Systems
Debugging Hadoop Applications
Understanding Normal

- Establish normal
  - Boring logs are good

- So that you can detect abnormal
  - Find anomalies and outliers
  - Compare performance

- And then isolate root causes
  - Who are the suspects
  - Pull the thread by interrogating
  - Correlate across different subsystems
Basic tools

- What happened?
  - Logs
- What state are we in now?
  - Metrics
  - Thread stacks
- What happens when I do this?
  - Tracing
  - Dumps
- Is it alive?
  - listings
  - Canaries
- Is it OK?
  - fsck / hbck
Diagnostics for single machines

Analysis and Tools via @brendangregg
## Diagnosis Tools

<table>
<thead>
<tr>
<th></th>
<th>HW/Kernel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logs</td>
<td>/log, /var/log, dmesg</td>
</tr>
<tr>
<td>Metrics</td>
<td>/proc, top, iotop, sar, vmstat, netstat</td>
</tr>
<tr>
<td>Tracing</td>
<td>Strace, tcpdump</td>
</tr>
<tr>
<td>Dumps</td>
<td>Core dumps</td>
</tr>
<tr>
<td>Liveness</td>
<td>ping</td>
</tr>
<tr>
<td></td>
<td>Corrupt fsck</td>
</tr>
</tbody>
</table>
Diagnostics for the JVM

- Most Hadoop services run in the Java VM, intros new java subsystems
  - Just in time compiler
  - Threads
  - Garbage collection
- Dumping Current threads
  - jstacks (any threads stuck or blocked?)
- JVM Settings:
  - Enabling GC Logs: `-XX:+PrintGCDateStamps -XX:+PrintGCDetails`
  - Enabling OOME Heap dumps: `-XX:+HeapDumpOnOutOfMemoryError`
- Metrics on GC
  - Get gc counts and info with `jstat`
- Memory Usage dumps
  - Create heap dump: `jmap -dump:live,format=b,file=heap.bin <pid>`
  - View heap dump from crash or jmap with `jhat`
## Diagnosis Tools

<table>
<thead>
<tr>
<th></th>
<th>HW/Kernel</th>
<th>JVM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Logs</strong></td>
<td>/log, /var/log dmesg</td>
<td>Enable gc logging</td>
</tr>
<tr>
<td><strong>Metrics</strong></td>
<td>/proc, top, iotop, sar, vmstat, netstat</td>
<td>jstat</td>
</tr>
<tr>
<td><strong>Tracing</strong></td>
<td>Strace, tcpdump</td>
<td>Debugger, jdb, jstack</td>
</tr>
<tr>
<td><strong>Dumps</strong></td>
<td>Core dumps</td>
<td>jmap</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Enable OOME dump</td>
</tr>
<tr>
<td><strong>Liveness</strong></td>
<td>ping</td>
<td>Jps</td>
</tr>
<tr>
<td><strong>Corrupt</strong></td>
<td>fsck</td>
<td></td>
</tr>
</tbody>
</table>

© Cloudera, Inc. All rights reserved.
Tools for lots of machines

- Other systems: httpd, sas, custom apps etc

- Machine
  - Hadoop Daemons
  - JVM
  - Linux
  - Disk, CPU, Mem
## Tools for lots of machines

<table>
<thead>
<tr>
<th>Machine</th>
<th>Hadoop Daemons</th>
<th>JVM</th>
<th>Linux</th>
<th>Disk, CPU, Mem</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Ganglia*, *Nagios*

Other systems: httpd, sas, custom apps etc
Ganglia

- Collects and aggregates metrics from machines
- Good for what’s going on right now and describing normal perf
- Organized by physical resource (CPU, Mem, Disk)
  - Good defaults
  - Good for pin pointing machines
  - Good for seeing overall utilization
  - Uses RRDTool under the covers
- Some data scalability limitations, lossy over time.
- Dynamic for new machines, requires config for new metrics
Graphite

- Popular alternative to ganglia
- Can handle the scale of metrics coming in
- Similar to ganglia, but uses its own RRD database.
- More aimed at dynamic metrics (as opposed to statically defined metrics)
Nagios

- Provides alerts from canaries and basic health checks for services on machines
- Organized by Service (httpd, dns, etc)
- Defacto standard for service monitoring

- Lacks distributed system know how
  - Requires bespoke setup for service slaves and masters
  - Lacks details with multi-tenant services or short-lived jobs
Tools for the Hadoop Stack

Other systems: httpd, sas, custom apps etc
Tools for the Hadoop Stack

User interface: HUE

Languages/APIs: Hive, Pig, Crunch, Kite, Mahout

Batch processing: MapReduce

Interactive SQL: Impala

Search: Solr

Interactive SQL: Impala

Batch processing: MapReduce

Resource Management: YARN

Random Access Storage: HBase

File Storage: HDFS

Coordination: zookeeper; Security: Sentry

Coordination: zookeeper; Security: Sentry

Event ingest: Flume, Kafka

DB Import Export: Sqoop

Other systems: httpd, sas, custom apps etc

Ganglia, Nagios

JVM

Linux

Disk, CPU, Mem

Ganglia*, Nagios*

Random Access Storage: HBase

Random Access Storage: HBase

File Storage: HDFS

File Storage: HDFS

Languages/APIs: Hive, Pig, Crunch, Kite, Mahout

Batch processing: MapReduce

Interactive SQL: Impala

Search: Solr

Random Access Storage: HBase

File Storage: HDFS

Coordination: zookeeper; Security: Sentry

Event ingest: Flume, Kafka

DB Import Export: Sqoop

Other systems: httpd, sas, custom apps etc

Ganglia, Nagios

JVM

Linux

Disk, CPU, Mem

Ganglia*, Nagios*

Random Access Storage: HBase

Random Access Storage: HBase

File Storage: HDFS

File Storage: HDFS

Languages/APIs: Hive, Pig, Crunch, Kite, Mahout

Batch processing: MapReduce

Interactive SQL: Impala

Search: Solr

Random Access Storage: HBase

File Storage: HDFS

Coordination: zookeeper; Security: Sentry

Event ingest: Flume, Kafka

DB Import Export: Sqoop

Other systems: httpd, sas, custom apps etc

Ganglia, Nagios

JVM

Linux

Disk, CPU, Mem

Ganglia*, Nagios*

Random Access Storage: HBase

Random Access Storage: HBase

File Storage: HDFS

File Storage: HDFS

Languages/APIs: Hive, Pig, Crunch, Kite, Mahout

Batch processing: MapReduce

Interactive SQL: Impala

Search: Solr

Random Access Storage: HBase

File Storage: HDFS

Coordination: zookeeper; Security: Sentry

Event ingest: Flume, Kafka

DB Import Export: Sqoop

Other systems: httpd, sas, custom apps etc

Ganglia, Nagios

JVM

Linux

Disk, CPU, Mem

Ganglia*, Nagios*

Random Access Storage: HBase

Random Access Storage: HBase

File Storage: HDFS

File Storage: HDFS

Languages/APIs: Hive, Pig, Crunch, Kite, Mahout

Batch processing: MapReduce

Interactive SQL: Impala

Search: Solr

Random Access Storage: HBase

File Storage: HDFS

Coordination: zookeeper; Security: Sentry

Event ingest: Flume, Kafka

DB Import Export: Sqoop

Other systems: httpd, sas, custom apps etc

Ganglia, Nagios

JVM

Linux

Disk, CPU, Mem

Ganglia*, Nagios*

Random Access Storage: HBase

Random Access Storage: HBase

File Storage: HDFS

File Storage: HDFS

Languages/APIs: Hive, Pig, Crunch, Kite, Mahout

Batch processing: MapReduce

Interactive SQL: Impala

Search: Solr

Random Access Storage: HBase

File Storage: HDFS

Coordination: zookeeper; Security: Sentry

Event ingest: Flume, Kafka

DB Import Export: Sqoop

Other systems: httpd, sas, custom apps etc

Ganglia, Nagios

JVM

Linux

Disk, CPU, Mem

Ganglia*, Nagios*

Random Access Storage: HBase

Random Access Storage: HBase

File Storage: HDFS

File Storage: HDFS

Languages/APIs: Hive, Pig, Crunch, Kite, Mahout

Batch processing: MapReduce

Interactive SQL: Impala

Search: Solr

Random Access Storage: HBase

File Storage: HDFS

Coordination: zookeeper; Security: Sentry

Event ingest: Flume, Kafka

DB Import Export: Sqoop

Other systems: httpd, sas, custom apps etc

Ganglia, Nagios

JVM

Linux

Disk, CPU, Mem

Ganglia*, Nagios*

Random Access Storage: HBase

Random Access Storage: HBase

File Storage: HDFS

File Storage: HDFS

Languages/APIs: Hive, Pig, Crunch, Kite, Mahout

Batch processing: MapReduce

Interactive SQL: Impala

Search: Solr

Random Access Storage: HBase

File Storage: HDFS

Coordination: zookeeper; Security: Sentry

Event ingest: Flume, Kafka

DB Import Export: Sqoop

Other systems: httpd, sas, custom apps etc

Ganglia, Nagios

JVM

Linux

Disk, CPU, Mem

Ganglia*, Nagios*

Random Access Storage: HBase

Random Access Storage: HBase

File Storage: HDFS

File Storage: HDFS

Languages/APIs: Hive, Pig, Crunch, Kite, Mahout

Batch processing: MapReduce

Interactive SQL: Impala

Search: Solr

Random Access Storage: HBase

File Storage: HDFS

Coordination: zookeeper; Security: Sentry

Event ingest: Flume, Kafka

DB Import Export: Sqoop

Other systems: httpd, sas, custom apps etc

Ganglia, Nagios

JVM

Linux

Disk, CPU, Mem

Ganglia*, Nagios*

Random Access Storage: HBase

Random Access Storage: HBase

File Storage: HDFS

File Storage: HDFS

Languages/APIs: Hive, Pig, Crunch, Kite, Mahout

Batch processing: MapReduce

Interactive SQL: Impala

Search: Solr

Random Access Storage: HBase

File Storage: HDFS

Coordination: zookeeper; Security: Sentry

Event ingest: Flume, Kafka

DB Import Export: Sqoop

Other systems: httpd, sas, custom apps etc

Ganglia, Nagios

JVM

Linux

Disk, CPU, Mem

Ganglia*, Nagios*

Random Access Storage: HBase

Random Access Storage: HBase

File Storage: HDFS

File Storage: HDFS

Languages/APIs: Hive, Pig, Crunch, Kite, Mahout

Batch processing: MapReduce

Interactive SQL: Impala

Search: Solr

Random Access Storage: HBase

File Storage: HDFS

Coordination: zookeeper; Security: Sentry

Event ingest: Flume, Kafka

DB Import Export: Sqoop

Other systems: httpd, sas, custom apps etc

Ganglia, Nagios

JVM

Linux

Disk, CPU, Mem

Ganglia*, Nagios*

Random Access Storage: HBase

Random Access Storage: HBase

File Storage: HDFS

File Storage: HDFS

Languages/APIs: Hive, Pig, Crunch, Kite, Mahout

Batch processing: MapReduce

Interactive SQL: Impala

Search: Solr

Random Access Storage: HBase

File Storage: HDFS

Coordination: zookeeper; Security: Sentry

Event ingest: Flume, Kafka

DB Import Export: Sqoop

Other systems: httpd, sas, custom apps etc

Ganglia, Nagios

JVM

Linux

Disk, CPU, Mem
Diagnostics for the Hadoop Stack

- Single client call can trigger many RPCs spanning many machines
- Systems are evolving quickly
- A failure on one daemon, by design, does not cause failure of the entire service
- Logs:
  - Each service’s master and slaves have their own logs: /var/log/
  - There are lot of logs and they change frequently
- Metrics:
  - Each daemon offers metrics, often aggregated at masters
- Tracing:
  - Htrace (integrated into Hadoop, HBase, recently in Apache Incubator)
- Liveness:
  - Canaries, service/daemon web uis
## Diagnosis Tools

<table>
<thead>
<tr>
<th></th>
<th>HW/Kernel</th>
<th>JVM</th>
<th>Hadoop Stack</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Logs</strong></td>
<td>/log, /var/log dmesg</td>
<td>Enable gc logging</td>
<td>*/var/log/hadoop</td>
</tr>
<tr>
<td><strong>Metrics</strong></td>
<td>/proc, top, iotop, sar, vmstat, netstat</td>
<td>jstat</td>
<td>*/stacks, */jmx,</td>
</tr>
<tr>
<td><strong>Tracing</strong></td>
<td>Strace, tcpdump</td>
<td>Debugger, jdb, jstack</td>
<td>htrace</td>
</tr>
<tr>
<td><strong>Dumps</strong></td>
<td>Core dumps</td>
<td>jmap</td>
<td>*/dump</td>
</tr>
<tr>
<td><strong>Liveness</strong></td>
<td>ping</td>
<td>Jps</td>
<td>Web UI</td>
</tr>
<tr>
<td><strong>Corrupt</strong></td>
<td>fsck</td>
<td></td>
<td>HDFS fsck, HBase hbck</td>
</tr>
</tbody>
</table>
Tools for the Hadoop Stack

User interface: HUE

Languages/APIs: Hive, Pig, Crunch, Kite, Mahout
Batch processing: MapReduce
In Mem processing: Spark
Interactive SQL: Impala
Search: Solr

Random Access Storage: HBase
File Storage: HDFS

Resource Management: YARN

Coordination: zookeeper; Security: Sentry

Event ingest: Flume, Kafka
DB Import Export: Sqoop

Other systems: httpd, sas, custom apps etc

JVM
Linux
Disk, CPU, Mem
Ganglia, Nagios

Ganglia*, Nagios*

© Cloudera, Inc. All rights reserved.
Ganglia and Nagios are not enough

- Fault tolerant distributed system masks many problems (by design!)
  - Some failures are not critical – failure condition more complicated

- Lacks distributed system know how
  - Requires bespoke setup for service slaves and masters
  - Lacks details with multitenant services or short-lived jobs

- Hadoop services are logically dependent on each other
  - Need to correlate metrics across different service and machines
  - Need to correlate logs from different services and machines.
  - Young systems where Logs are changing frequently
  - What about all the logs?
  - What of all these metrics do we really need?

- Some data scalability limitations, lossy over time
  - What about full fidelity?
OpenTSDB (Time Series Database)
- Efficiently stores metric data into HBase
- Keeps data at full fidelity
- Keep as much data as your HBase instance can handle.

• Free and Open Source
Cloudera Manager

- Extracts hardware, OS, and Hadoop service metrics specifically relevant to Hadoop and its related services.
  - Operation Latencies
  - # of disk seeks
  - HDFS data written
  - Java GC time
  - Network IO
- Provides
  - Cluster preflight checks
  - Basic host checks
  - Regular health checks
- Uses LevelDB for underlying metrics storage
- Provides distributed log search
- Monitors logs for known issues
- Point and click for useful utils (lsif, jstack, jmap)
- Free