Hadoop performance measuring

By James Baldwin and Martin Callaghan

Contents

1. Cluster setup, tuning and management options
2. Case study
3. Practical
4. Hadoop cluster monitoring tool
5. Alternatives to MapReduce

Sticky Notes

Display the **ORANGE** sticky if need help

Display the **BLUE** sticky if everything is OK

START PPT

Hadoop options

- **Private** or **public** cloud options
  - Azure
  - AWS
  - Google

- Use case: **business requirements**
  - Test business need against that platform

Build/operate your HDP cluster?

- **In-house** operate it in **house**
- **In-house** operate it using a **partner**
- Build using **partner** and operate in **house**
- Build it with **partner** and operate with **partner**

Cluster Size

Sizing the cluster is important, right **balance** of resources will allow you to **optimise** the **environment** for **purpose**, but this is not easy as there are many **complexities** in **tuning** a distributed environment and the use of related **plugins** from the eco-system

Basic setup

Clusters that have more than **three machines** would use a **dedicated** NameNode/JobTracker and other nodes that are called **slave nodes**

The larger cluster, the greater number of **master nodes** are required to **co-ordinate** jobs appropriately, **backup** and **voting**

An example of a Hadoop cluster

NameNode: **Metadata** of filesystem

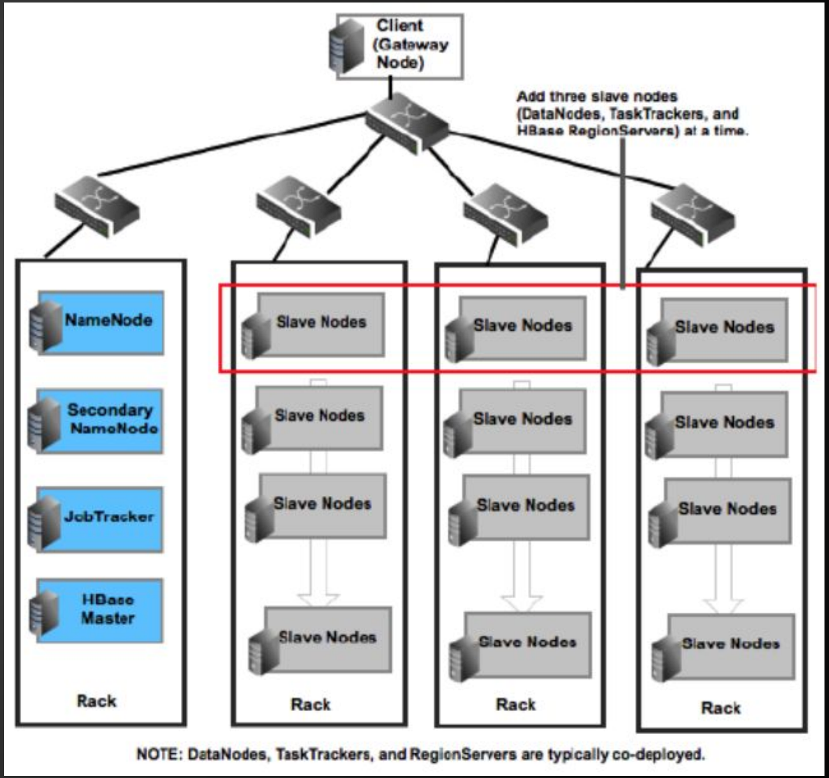
Secondary NameNode:

Connects to the **primary namenode**, **snapshotting** the filesystem **metadata** into local/remote storage.

JobTracker:

Resource management:

Assigning and **coordinating** tasks to **slave nodes**



Balance and performance

- Hadoop cluster: Ideally **little idle** time to **maximise** the cluster

- Hadoop runs **multiple jobs** and **creates new** or **updates** existing **datasets**, which has an **impact** on the **bandwidth**

- Balancing will require a good selection of hardware config AS *"**master** and **slave nodes** where consumption of CPU, memory and disk resources **peak together**" (Hortonworks, 2013)*, **design consideration** will revolve more around **slave nodes**

- A balanced cluster minimise performance bottlenecks and avoids unused capacity

Hadoop: Shared Nothing

- Master and slave nodes **do not share** any **hardware resource**
- If **sharing resource**, means the demand will **not exceed** the **total bandwidth**
- Hadoop jobs demand **max performance**, if **sharing resource** will result in **saturation** and a **bottleneck**
- Shared Nothing has its benefits, when **nodes** are added to the cluster, as nodes **do not share resource**, **new nodes** brings **added** CPU, memory and disk I/O

Cluster design tradeoffs

- A smaller cluster provides **flexibility** in **decision-making**
- As the **cluster grows** the more **stringent** and **sensitive** the **designing** of the cluster
  - **Cooling** and **power**
  - How things **connect** together in the **network**

- **Easier** to grow **incrementally** on demand, **build** in **standard blocks** to **keep order**
- The **larger** the **cluster**, the more **resilient** it becomes
  - **More nodes** tend to be better, **than** having **more powerful** nodes

Map optimisation

Optimization before the job runs

- Data preparation - **preprocessing**
- **Compress** the file, **break** the file down into different sizes
- Keep the job **simple**

- Write **better code** that has a higher level of **efficiency**
  - Input an instruction to **skip invalid** or **bad data**
- Using a **combiner**
  - Rather than listing a word 50 times, we would just have 50
  - A combiner is common, as it means **less data** passed **across** the **network**

Optimization after job completion

- **Breakdown** chunks into **multiple jobs**, running **lightweight** jobs go **first** and then **dedicate** greater resource to **larger jobs**
- **Compression**: think about its positive and negative aspects are to processing
- **Applications** or **jobs** can run faster when configuring the path to point to HDFS or HTTP file (cached text, archive and jar files) location, as a file will be copied to each data node on a job execution for the mappers, and can be used for quick lookups

Reducer

1) **Subdividing** the data to **prevent** reducer being overloaded
2) To **log** or **debug**, in-order to understand where the workload is causing the reducer to **overload**, there are usually **less nodes** to **reduce** the data, identify these nodes in the log to understand how the **bottlenecks** are occurring
3) **Threshold** limit - **wait** time, **kill** long running, **retrieve** more info and **suspend**
4) Secondary **sort**
5) **Do not** use a **reducer** for jobs that do not require it, such as photo processing may be in binary, reduce may happen once the data has been **preprocessed**

Summary: tuning the cluster

- You can **add** more **nodes** to the cluster, as MapReduce and Hadoop is highly **distributed** and **scalable**
- **Measure**, don't second guess it, **profile** your cluster, you may gain some **insights** that are useful
- **Tune MapReduce**, but may take time to understand how to optimise the job depending on **user experience**

- Revert to the **cloud** as **option** when requiring **extra resources** on-demand, then **decide** how to **optimise** the cluster
- **Do not** just **invest** in **tools** only, but spend time in **load testing**

STOP

PPT

Any Questions

START

Case Study

Case study

- **MapReduce** has become an **increasingly** popular programming paradigm for many **data analytics** applications to process data in **parallel**.

- In many application areas (particularly in **Scientific Computing**), MapReduce has **never** been used and is **not** being **adopted**.

- The **purpose** of this **case study** is to explore some of the reasons why this might be the case.

- **Two** papers, study in **groups**, then **feedback**...

Case study questions

1. Where is the MPI programming paradigm used?
2. What are the differences between the MPI and MapReduce programming paradigms?
3. What Scientific Computing applications have not adopted MapReduce? What do you think the reasons for this might be? Is it worth adopting MapReduce? How difficult it is to adapt MPI to Hadoop?
4. Are there differences in efficiency for the two programming models? Where do they exist?
5. Are there any Scientific Computing application areas that could adopt MapReduce? How might this improve efficiency.

STOP Case Study

Now

START Practical Session

Practical

1. Connect to a Hadoop VM instance via using Putty
2. Start cluster
3. Bit of navigation and check stuff is already in HDFS
4. Run simple wordcount in Java
5. Two Pig scripts
6. Two Hive equivalent
7. Two variations of Hadoop streaming using Python

Through each stage…….

Feedback….

STOP   Practical

Now

START PPT

Last hour

1. Apache Ambari
2. Alternatives to MapReduce
3. BioPig

Background

Petabytes of Data

Big Data

MapReduce Jobs daily

amongst other operations

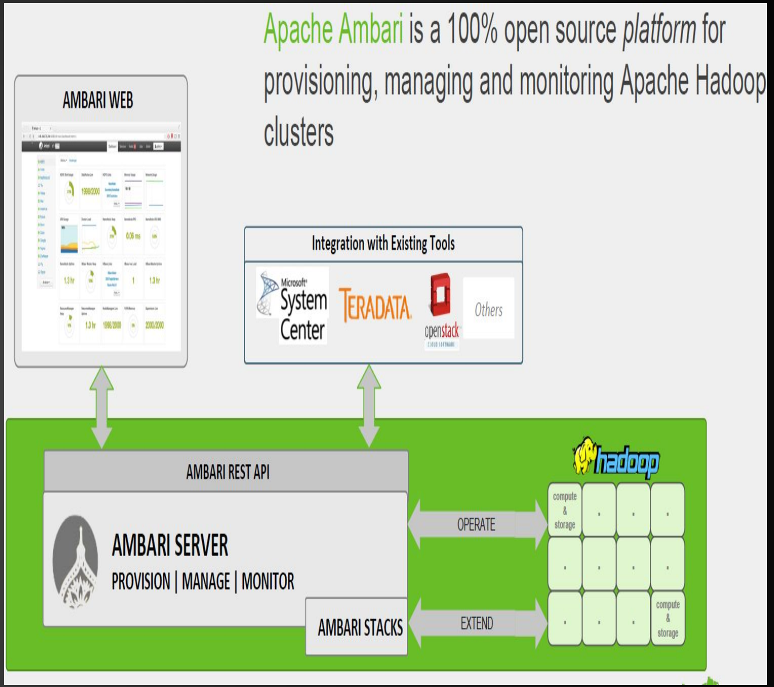
Multiple machine Hadoop clusters

Before

- Jobs running in cluster overnight
  - Cluster crash occurs
  - Limited view of data
- **Multiple jobs** running, how would one **manage** the use of **different plugins** across the cluster

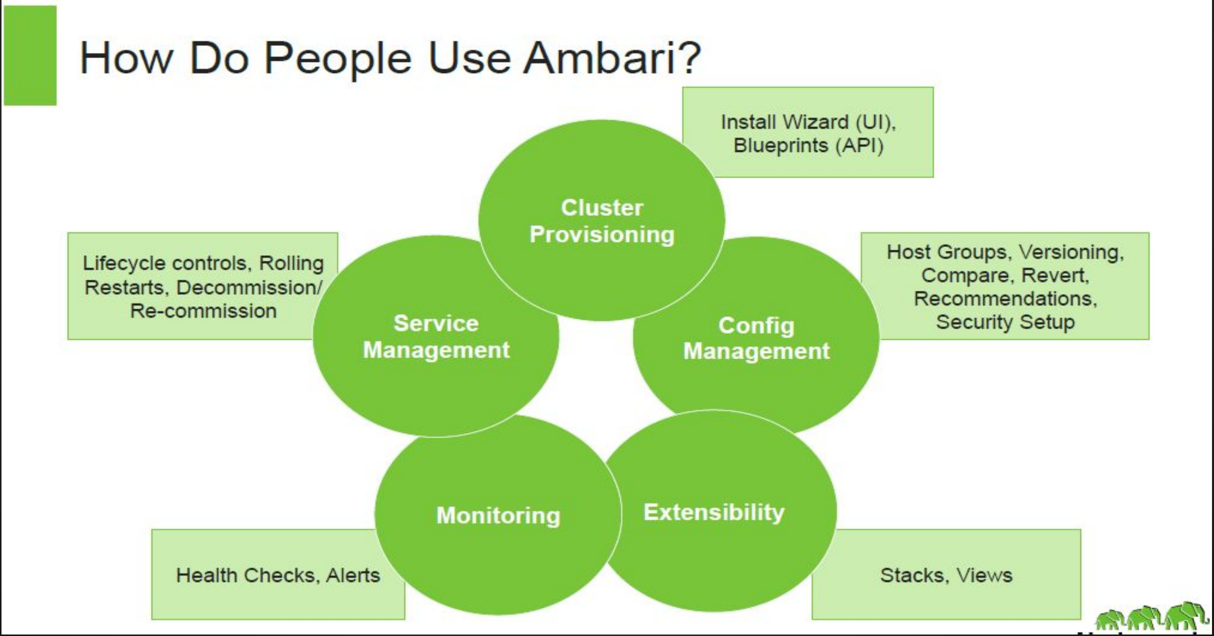
After

- Apache came up with **Ambari**
- Ambari **manages**, **monitors** and operates the the Hadoop cluster
    - **Growth** of cluster
- Ambari **integrates** into the Hadoop ecosystem, being at **central** point of **operational control**
- Wide **support** for Ambari
- Ambari has an **REST API**, same as web interface actions



How can Ambari be used?

“Ambari **simplifies** the **operation** and **hides** the **complexity** of Hadoop, making Hadoop work as a **single**, **cohesive** data **platform**.” (Hortonworks, 2013)



Cluster provisioning: Blueprints

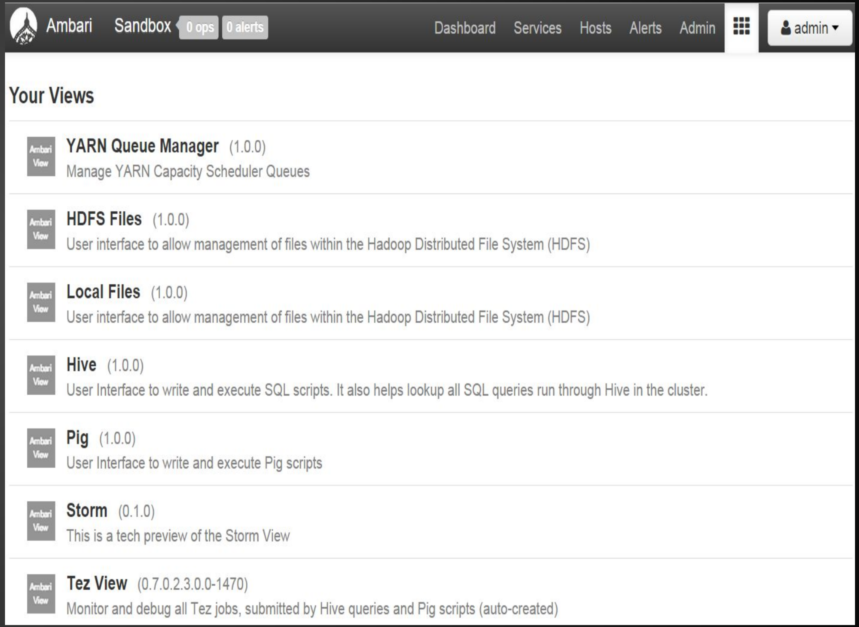
To use Ambari for **automating** cluster installations

To share best practices on layout and cluster configuration

Extensibility: Views

Ambari is growing and adding a secondary layer called views

- Views may eventually replace bundles with HDP like HUE
  - Ambari views for **file browser**, **Hive** queries and for **PIG** scripting, spin up **separate** views instances put **session aware balancer** in front of them



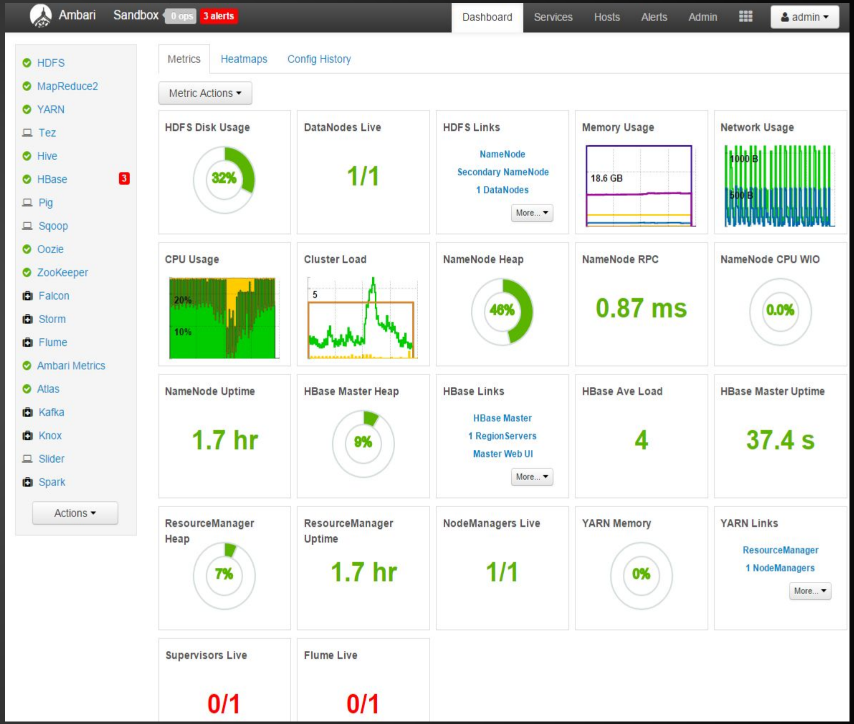
Extensibility and Config management: Stack and Versioning

- Wizard based
- Install services
- Upgrades
  - Migration
- Across the cluster

Service	Version	Status	Description
HDFS	2.7.1.2.3	Installed	Apache Hadoop Distributed File System
MapReduce2	2.7.1.2.3	Installed	Apache Hadoop NextGen MapReduce (YARN)
YARN	2.7.1.2.3	Installed	Apache Hadoop NextGen MapReduce (YARN)
Tez	0.7.0.2.3	Installed	Tez is the next generation Hadoop Query Processing framework written on top of YARN.
Hive	1.2.1.2.3	Installed	Data warehouse system for ad-hoc queries & analysis of large datasets and table & storage management service
HBase	1.1.1.2.3	Installed	A Non-relational distributed database, plus Phoenix, a high performance SQL layer for low latency applications.
Pig	0.15.0.2.3	Installed	Scripting platform for analyzing large datasets
Sqoop	1.4.6.2.3	Installed	Tool for transferring bulk data between Apache Hadoop and structured data stores such as relational databases
Oozie	4.2.0.2.3	Installed	System for workflow coordination and execution of Apache Hadoop jobs. This also includes the installation of the optional Oozie Web Console which relies on and will install the ExtJS Library.
ZooKeeper	3.4.6.2.3	Installed	Centralized service which provides highly reliable distributed coordination
Falcon	0.6.1.2.3	Installed	Data management and processing platform
Storm	0.10.0	Installed	Apache Hadoop Stream processing framework
Flume	1.5.2.2.3	Installed	A distributed service for collecting, aggregating, and moving large amounts of streaming data into HDFS
Accumulo	1.7.0.2.3	Add Service	Robust, scalable, high performance distributed key/value store.
Ambari Metrics	0.1.0	Installed	A system for metrics collection that provides storage and retrieval capability for metrics collected from the cluster
Atlas	0.5.0.2.3	Installed	Atlas Metadata and Governance platform
Kafka	0.8.2.2.3	Installed	A high-throughput distributed messaging system
Kerberos	1.10.3-10	Add Service	A computer network authentication protocol which works on the basis of 'tickets' to allow nodes communicating over a non-secure network to prove their identity to one another in a secure manner.
Knox	0.6.0.2.3	Installed	Provides a single point of authentication and access for Apache Hadoop services in a cluster
Mahout	0.9.0.2.3	Add Service	Project of the Apache Software Foundation to produce free implementations of distributed or otherwise scalable machine learning algorithms focused primarily in the areas of collaborative filtering, clustering and classification
Ranger	0.5.0.2.3	Add Service	Comprehensive security for Hadoop
Ranger KMS	0.5.0.2.3	Add Service	Key Management Server
Slider	0.80.0.2.3	Installed	A framework for deploying, managing and monitoring existing distributed applications on YARN.
Spark	1.3.1.2.3	Installed	Apache Spark is a fast and general engine for large-scale data processing.

Monitoring

- **Cluster management**
  - Performance
- **Ambari metrics (refer to materials)**
- **Ambari alerts**
  - **Pre-configured** by default
  - Health **alerts** are **centrally managed**
  - Modifying alerts: **threshold** and **return message**
  - Alert **groups** and **notifications**



Summary

- If you need to **manage** a Hadoop cluster, **Ambari** a good option
- Well **supported** by the community
- Use of **wizard** makes it **user friendly**
- **Easier** to **manage** the cluster, helping to **perform** at its **optimum**
- Considered as the **future**, possibly **replacing** HUE

Alternatives to MapReduce

BioPig

BioPig

Plugin for Apache Hadoop MapReduce and utilises PIG language

Most bioinformatics analysis tools do not support parallelization

Data growth has made most bioinformatics analytic tools obsolete

- Fail to scale with data
- Too much time and memory

BioPig benefits

3 major advantages

- Ease of use
- Scales with large data
- Portable

Accelerates data-intensive bioinformatics analysis

BioPig limitations

Slower than MPI

- Due to both the latency of Hadoop's initialization
- Generic MapReduce algorithm not optimised for specific problems
    - Time spent on data analysis exceeds cost of startup latency

Alternatives

- IBM's Symphony
- Spark

Been developed to reduce Hadoop's startup latency and other issues

Refer to MPI slides

Think about this

“WHETHER AN ALGORITHM IS ‘AMENABLE’ TO MAPREDUCE IS A RELATIVE JUDGMENT THAT IS ONLY MEANINGFUL IN THE CONTEXT OF AN ALTERNATIVE.”

“OF COURSE IT MAKES SENSE TO USE THE RIGHT TOOL FOR THE JOB, BUT WE MUST ALSO RECOGNIZE THE COST ASSOCIATED WITH SWITCHING TOOLS—IN SOFTWARE ENGINEERING TERMS.”

As a result, the rate of growth of sequence data is now outpacing the underlying advances in storage and compute technologies (Moore’s law)

STOP PPT

Any questions

Before I Shut Down

Feedback

Write a positive comment about today on the **ORANGE** sticky

Use the **BLUE** sticky if there is something that could be improved or you'd like to know more about

References

Hortonwork. (2013). http://hortonworks.com/wp-content/uploads/downloads/2013/06/Hortonworks.ClusterConfigGuide.1.0.pdf