

Apache Hadoop

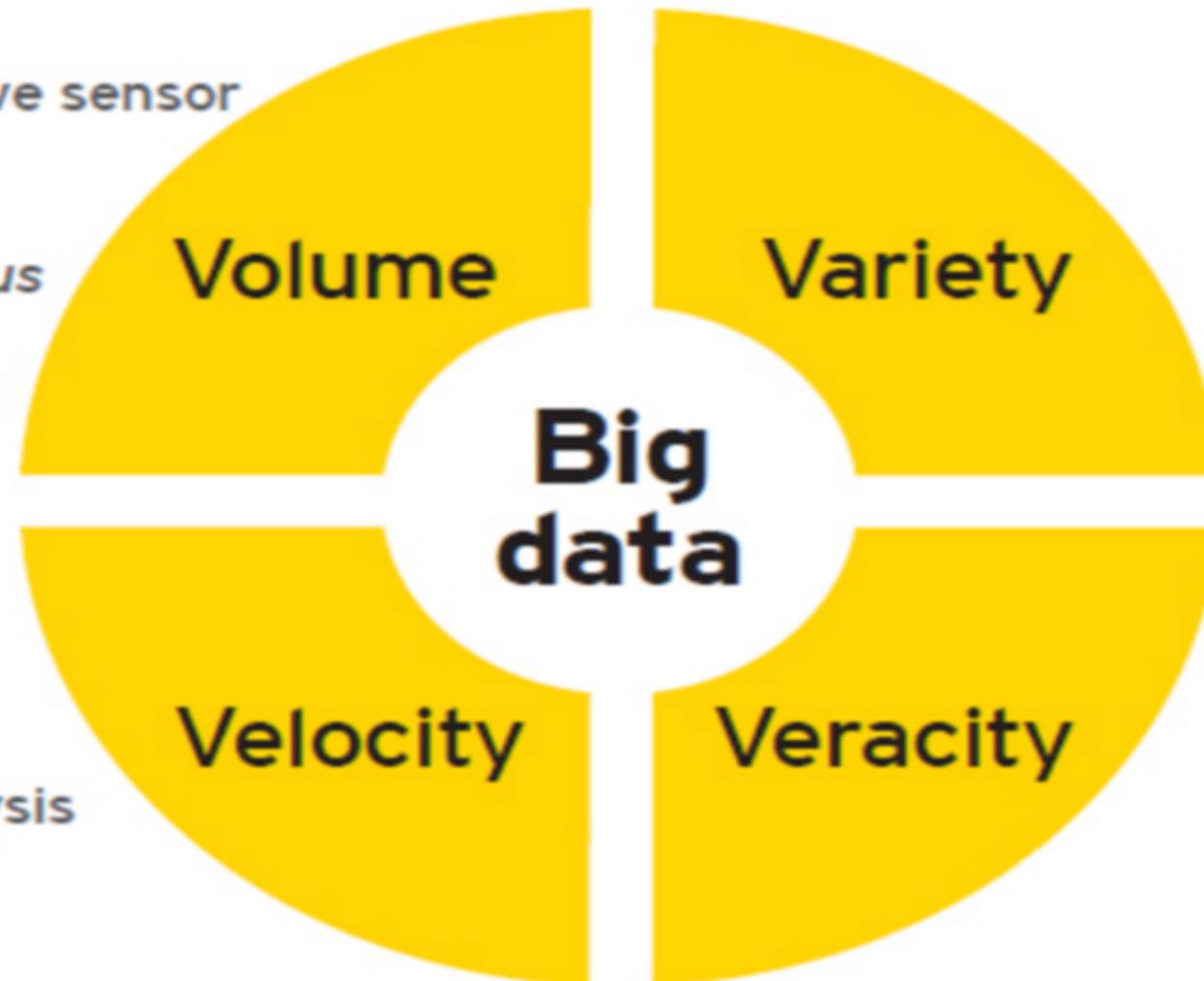
Course Content

- Introduction to Apache Hadoop
- Hadoop Cluster Installation
- Hadoop Distributed File System
- MapReduce and Spark on Yarn
- Hadoop Configuration and Daemon Logs
- Getting Data into HDFS
- Planning Hadoop Cluster
- Installing and Configuring Hive, Impala & Pig
- Hadoop Client including Hue
- Advanced Cluster Configuration
- Hadoop Security
- Managing Resources
- Cluster Maintenance
- Cluster Monitoring and Troubleshooting

Big Data !!

- ▶ Click stream
- ▶ Active/passive sensor
- ▶ Log
- ▶ Event
- ▶ Printed corpus
- ▶ Speech
- ▶ Social media
- ▶ Traditional

- ▶ Speed of generation
- ▶ Rate of analysis



- ▶ Unstructured
- ▶ Semi-structured
- ▶ Structured

- ▶ Untrusted
- ▶ Uncleansed

Volume

40 ZETTABYTES

[43 TRILLION GIGABYTES]

of data will be created by 2020, an increase of 300 times from 2005

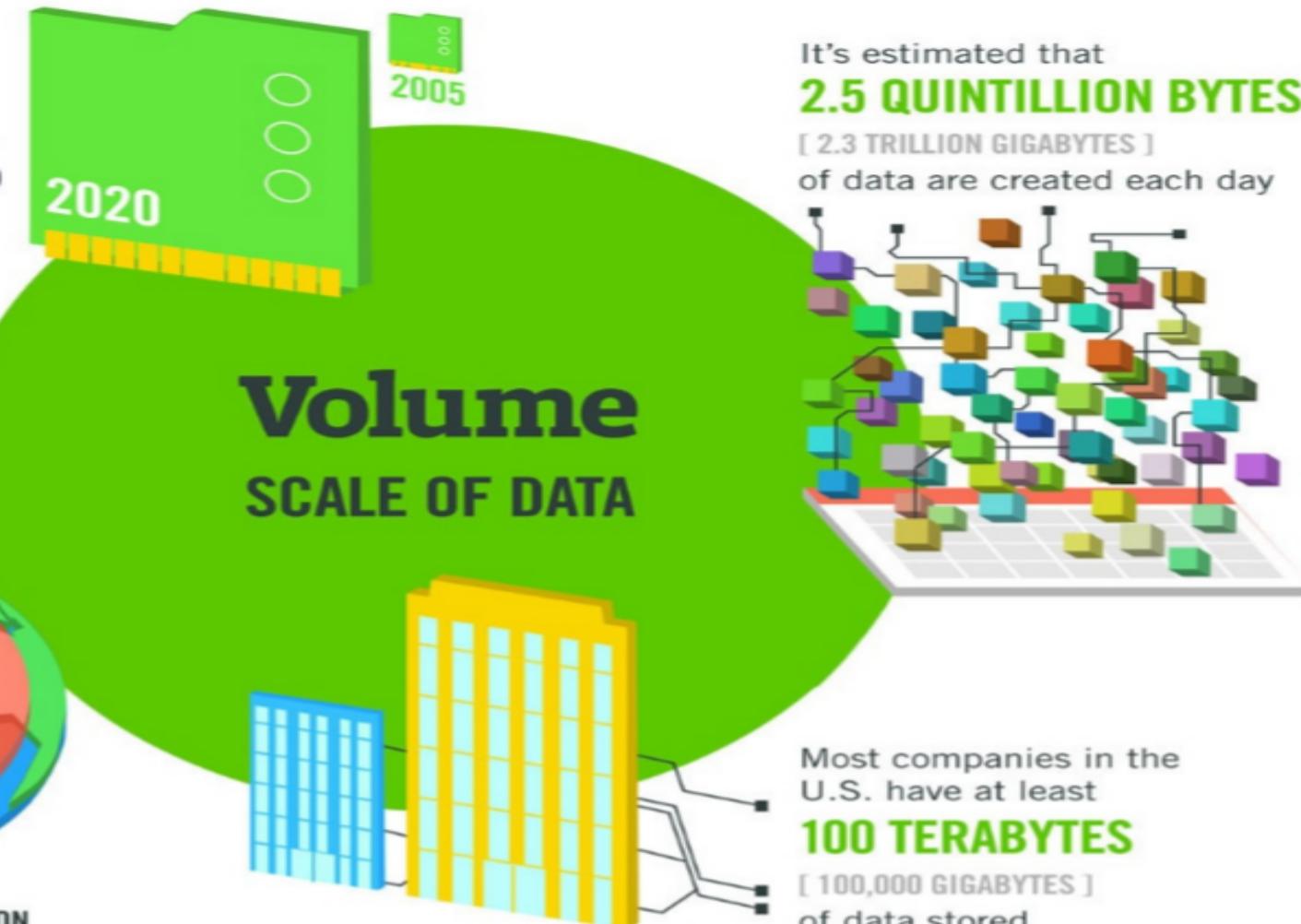


**6 BILLION
PEOPLE**

have cell phones



WORLD POPULATION: 7 BILLION



Velocity

The New York Stock Exchange captures
1 TB OF TRADE INFORMATION during each trading session

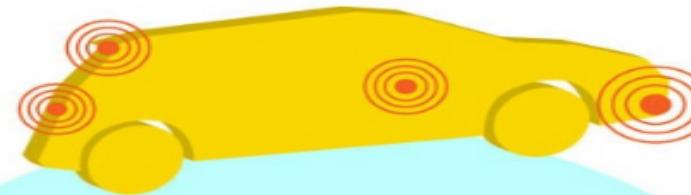


By 2016, it is projected there will be

18.9 BILLION NETWORK CONNECTIONS

– almost 2.5 connections per person on earth

Modern cars have close to
100 SENSORS that monitor items such as fuel level and tire pressure



Velocity ANALYSIS OF STREAMING DATA



Variety

As of 2011, the global size of data in healthcare was estimated to be

150 EXABYTES

[161 BILLION GIGABYTES]



30 BILLION PIECES OF CONTENT

are shared on Facebook every month



Variety
DIFFERENT FORMS OF DATA

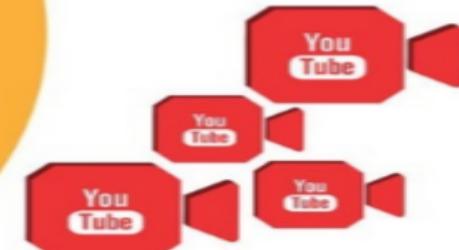


By 2014, it's anticipated there will be

420 MILLION WEARABLE, WIRELESS HEALTH MONITORS

4 BILLION+ HOURS OF VIDEO

are watched on YouTube each month



400 MILLION TWEETS

are sent per day by about 200 million monthly active users

Veracity

**1 IN 3 BUSINESS
LEADERS**

don't trust the information
they use to make decisions



**27% OF
RESPONDENTS**

in one survey were unsure of
how much of their data was
inaccurate

Veracity
**UNCERTAINTY
OF DATA**

Poor data quality costs the US
economy around
\$3.1 TRILLION A YEAR

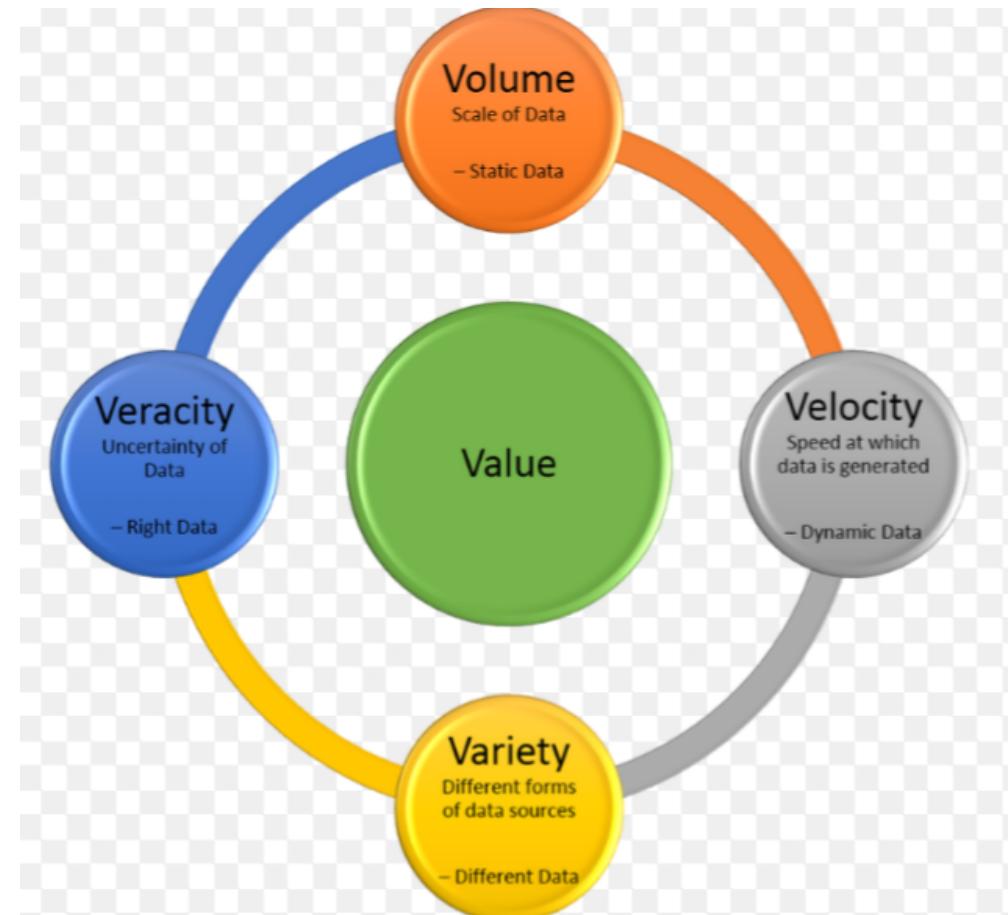


Data is Value !!

Data has many valuable application:

- Marketing analysis
- Product recommendations
- Demand forecasting
- Fraud Detection, etc.

We must **PROCESS** data to extract its Value.



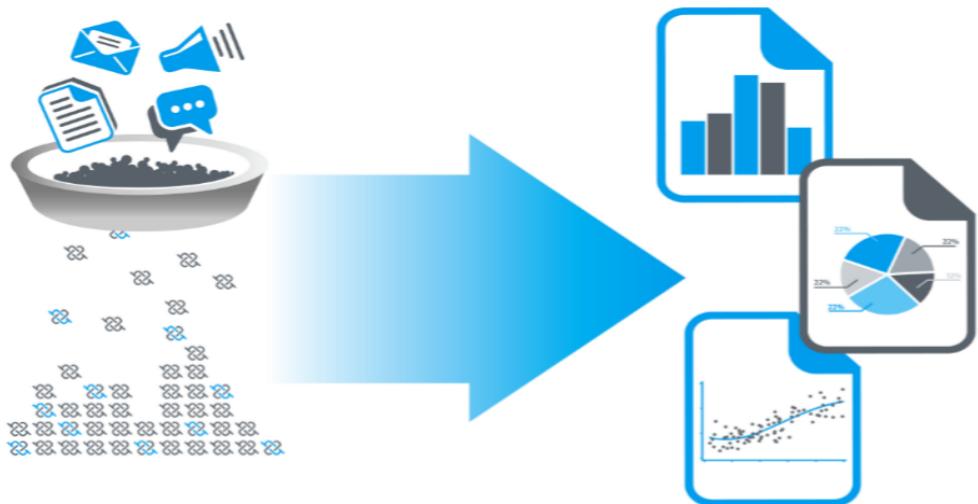
Introduction to Apache Hadoop

- The case for Apache Hadoop
- Why Hadoop ?
- Fundamental Concepts
- Core Hadoop Components

Why Hadoop ?

Major problem to process data:

- Large-scale Data Storage
- Large-scale Data Analysis



Data Storage

- The size and cost of storage has kept pace, Capacity has increased while price has decreased

Year	Capacity (GB)	Cost per GB (USD)
1997	2.1	\$157
2004	200	\$1.05
2014	3,000	\$0.036

Disk Performance

- Disk performance has also increased in last 15 years, but the transfer rate have not kept pace with capacity.

Year	Capacity (GB)	Transfer Rate (MB/s)	Disk Read Time
1997	2.1	16.6	126 seconds
2004	200	56.5	59 minutes
2014	3,000	210	3 hours, 58 minutes

Data Access

- We can process data more quickly, but accessing data is still slow for both reads and writes.
- Limitation is speed of the disk as we cannot process the data until we have read it.



Hadoop's solution can solve
this Bottleneck.

Types of Computing

Monolithic Computing

- Computation has been processor-bound.
- Intense processing on small amount of data.

Limitations:

- More powerful machine, requires more RAM which leads to High Cost.
- Limited scalability.

Distributing Computing

- Modern large-scale processing is distributed across machines.
- Main focus is on distributing the processing workload.
- Separate system for data storage.
- Fast network connection to connect them.

Limitation:

- Doesn't scale with large amount of data, as time spent on copying data is more than processing the data.
- Data processing bottleneck grows worse on adding more compute nodes.

Complexity of Distributed Computing

Complexity involves:

- Scaling up and down is not a smooth process.
- Processing power is spent on transporting data.
- Data synchronization is required during exchange.
- Huge dependency on network and huge Bandwidth demands.
- Partial failure are difficult to handle.
- Error handling often accounts for the majority of codes.



Fundamental Concepts

- Doug Cutting, Mike Cafarella and team took the solution provided by Google (GFS) and started an Open Source Project called HADOOP in 2005 and Doug named it after his son's toy elephant. Now Apache Hadoop is a registered trademark of the Apache Software Foundation.
- Apache Hadoop software library is framework that allows for distributed processing of large data sets across clusters of computers using a simple programming models.

Hadoop Scalability

- Hadoop aims for linear horizontal scalability
 - Cross communication between nodes is minimal.
 - Just add nodes to increase cluster capacity and performance.
- Clusters are built from industry-standard hardware
 - Widely-available and relatively inexpensive servers.
 - You can “scale-out” later when the need arises.

How Hadoop resolves Data Access Bottleneck ?

- Hadoop stores and process data on the same machine, and hence adding nodes to the cluster increases the capacity and performance.
- It also conserves the bandwidth.
- Uses intelligent job scheduling for data locality.

How Hadoop resolves Disk Performance Bottleneck ?

- Hadoop uses multiple disks in parallel.

Ex: The transfer rate of one disk might be 210 MB/seconds and takes almost 4hours to read 3 TB of data. So 1000 such disks in parallel can transfer 210 MB/seconds which takes less than 15 seconds to read 3TB of data.

- Colocated storage and processing makes this solution feasible.

Solution to Complex Processing Code

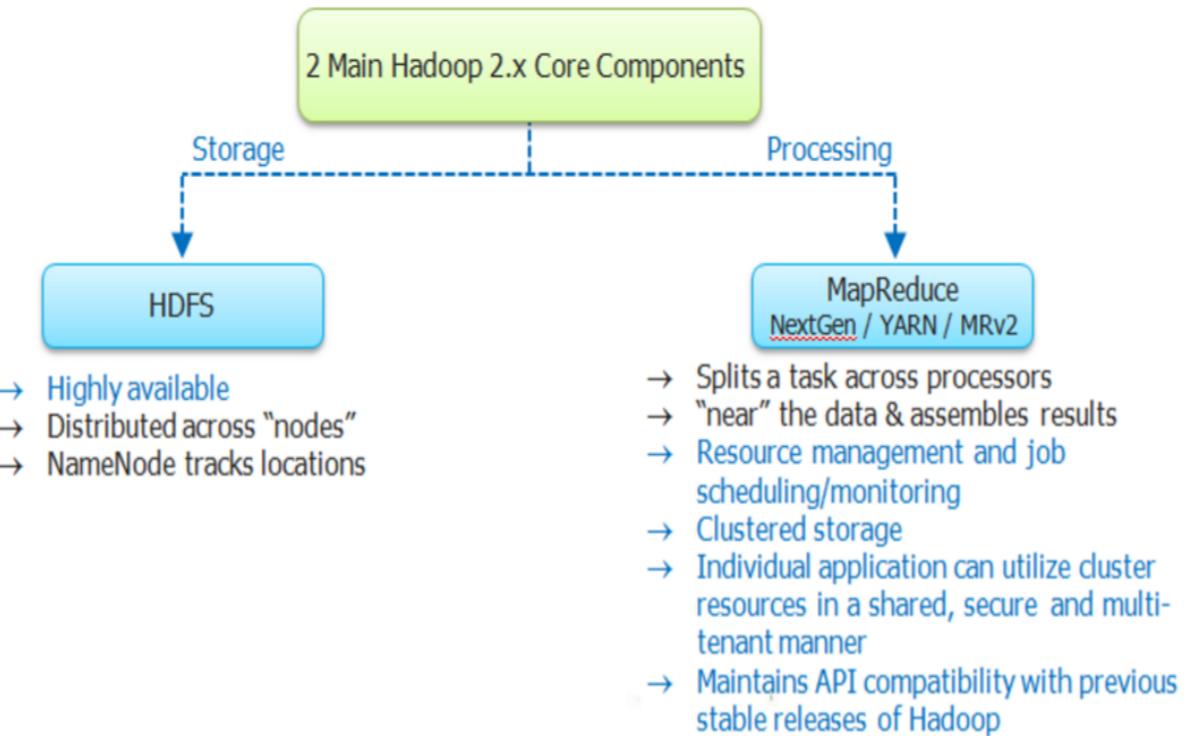
- MapReduce code is typically written in Java and it can be written in nearly any language.
- MapReduce programming model simplifies processing
 - Deals with one record (key-value pair) at a time
 - Complex details are abstracted away

Core Hadoop Components

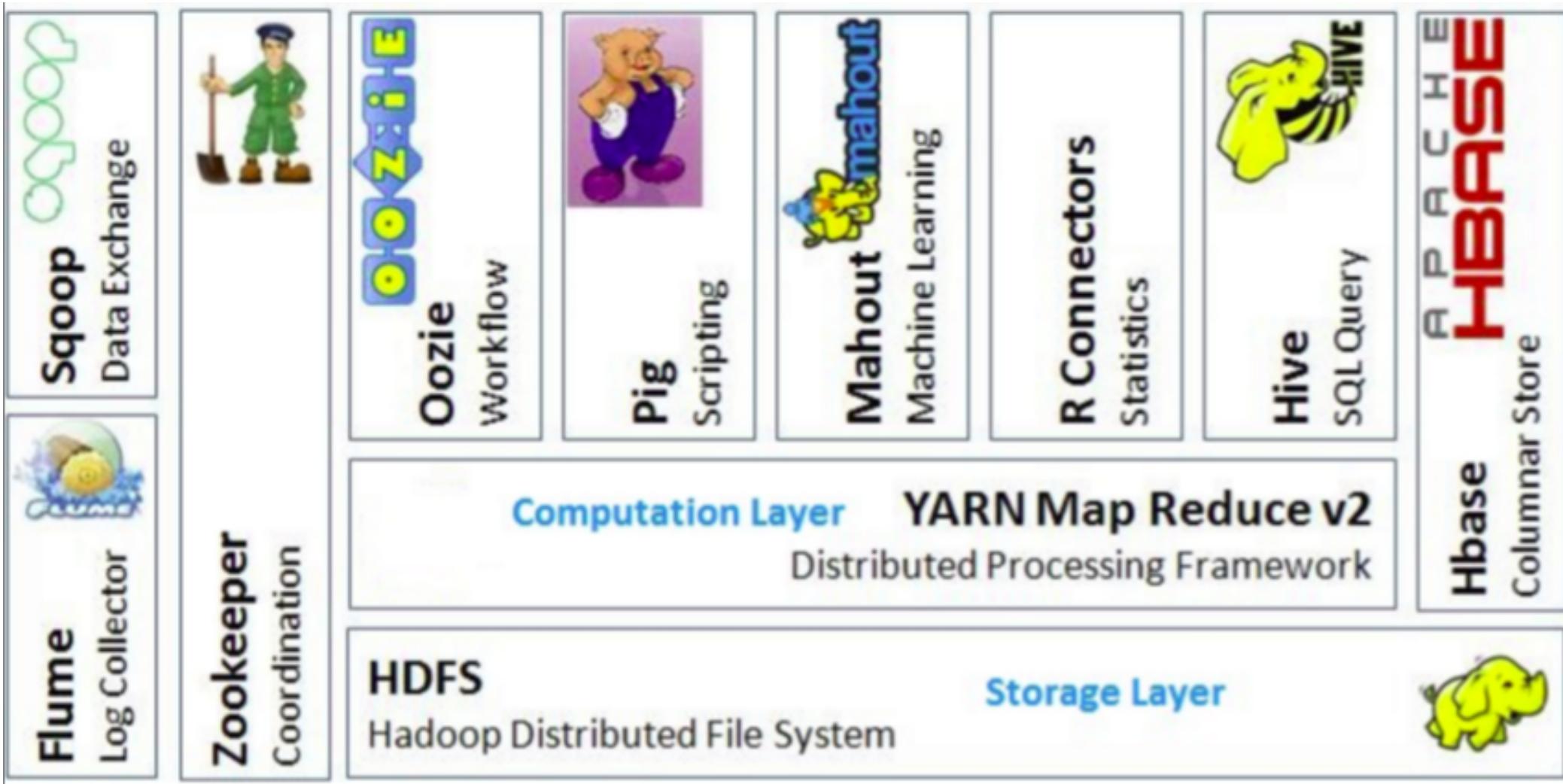
Hadoop is system for large scale data processing.

Hadoop consist of two core components:

- The Hadoop Distributed File System (HDFS) for data storage.
- YARN Framework for job scheduling and resource management, includes MapReduce version 2 plug in for data processing.



Hadoop Ecosystem



Quick Overview

- Four V's of Big Data.
- Data access and processing bottleneck with existing techniques.
- Hadoop eliminates the problem by storing and processing data on the same machine.



Hadoop Cluster Installation

We will cover following topics under this section:

- Rationale for a Cluster Management Solutions
- Cloudera Manager features
- Cloudera Manager Installation
- Hadoop (CDH) Installation

Modes Of Operation

Hadoop Cluster operates in following three Modes:

Local/Standalone Mode

1. Hadoop software runs as a single monolithic java process.
2. HDFS is not utilized in this mode.
3. Local filesystem is used for input and output.
4. Used for debugging purpose.
5. Custom configuration is not required.
6. Standalone Mode is faster than Psedo-Distributed Mode.

Cont..

Pseudo Distributed Mode (Single Node Cluster)

1. Configuration is required (mapred-site.xml, core-site.xml, hdfs-site.xml)
2. Replication factor is one for HDFS.
3. Used for code test in HDFS.
4. All daemon runs on one node.

Cont..

Fully Distributed Mode (Multiple Node Cluster)

1. Suitable for production environment.
2. Data is distributed across multiple nodes.
3. Hadoop daemons runs on different nodes as per the configuration.

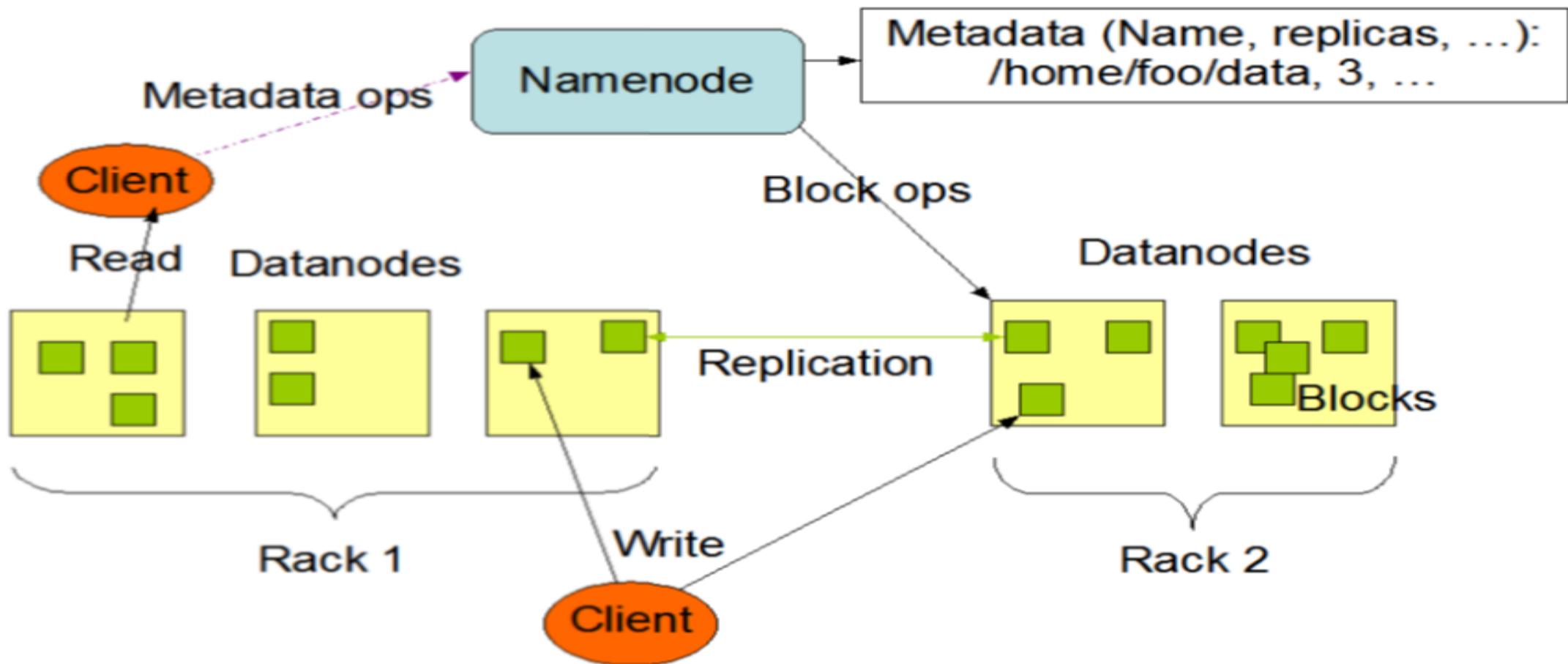
Hadoop Distributed File System (HDFS)

- Hadoop File System was developed using distributed file system design based on GFS.
- HDFS is responsible for storing data on the clusters of machines.
- Data is normally split into blocks of 64MB to 128MB and spread across the cluster.
- Files are stored in redundant fashion to rescue the system from possible data losses in case of failure.

HDFS Features

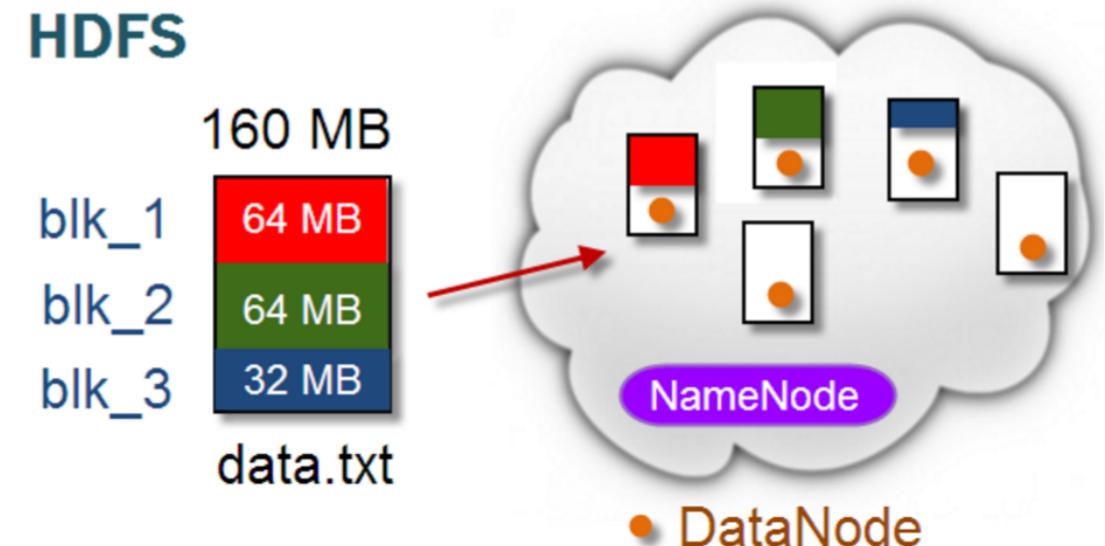
- It is suitable for the distributed storage and processing.
- HDFS provides file permissions and authentication.
- Fault tolerance
- High Performance
- Scalability
- Streaming access to file system data.

HDFS Architecture



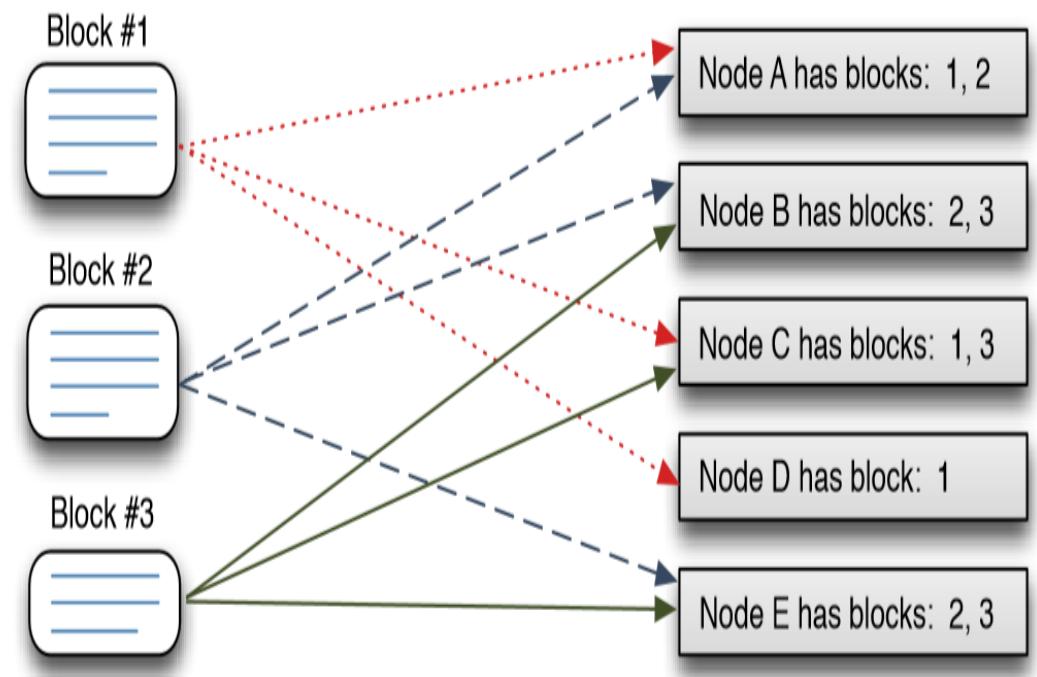
HDFS Blocks

- When a file is added to HDFS, it is split into blocks
- Default block size is 64MB, which is configurable.



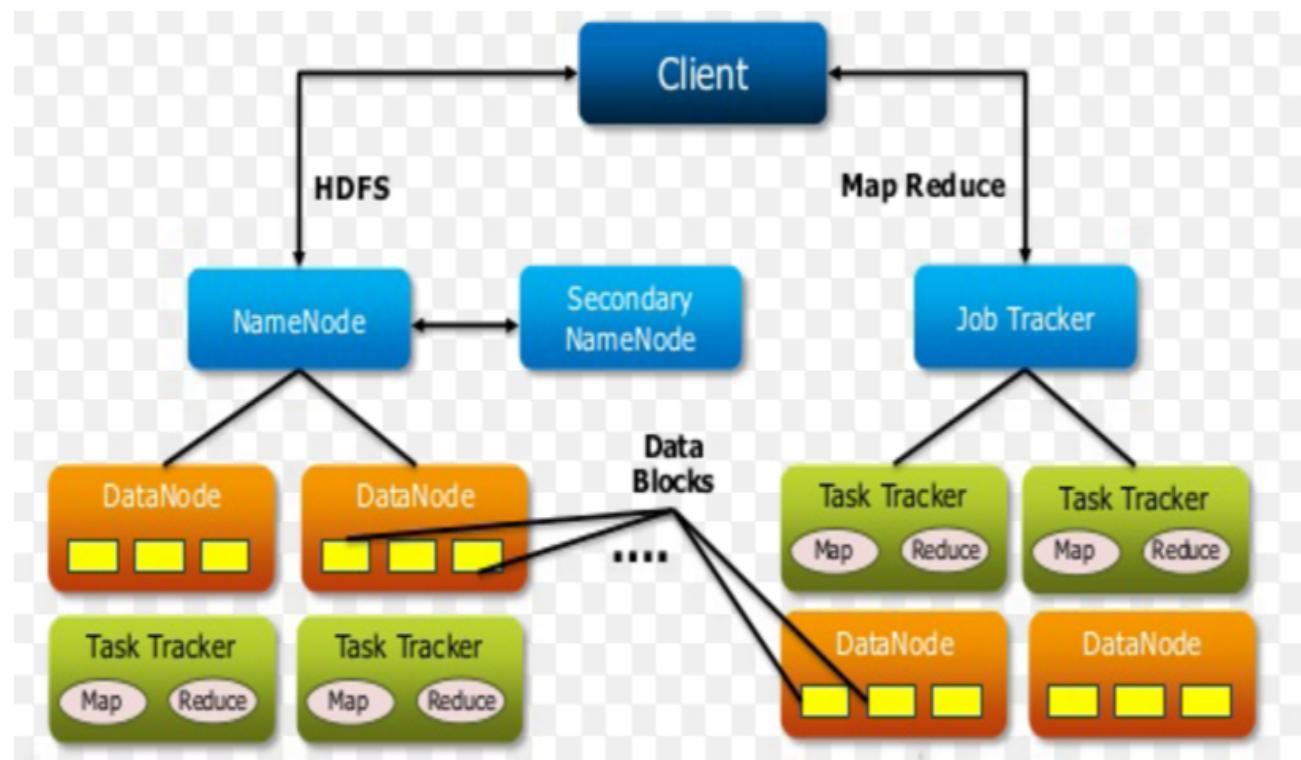
HDFS Replication

- Based on replication factor, blocks are replicated to nodes throughout the cluster.
- Replication increases reliability as data can tolerate loss of all but one replica.
- Replication also enhances the performance as it provides more opportunity for data locality.



HDFS Without High Availability

- HDFS can be deployed with and without high availability.
- Without high availability, there are three daemons:
 - NameNode
 - Secondary NameNode
 - DataNode



NameNode

- NameNodes maintain the namespace tree for HDFS and a mapping of file blocks to DataNodes where the data is stored.
- A simple HDFS cluster can have only one primary NameNode, supported by a secondary NameNode that periodically compresses the NameNode edits log file that contains a list of HDFS metadata modifications.
- Metadata is stored on disk (`fsimage` file) and read when the Namenode daemon starts up.
- Changes to metadata are made in RAM, which are also written to a log file on disk called `edits`.

Slave Nodes

- Actual content of the files are stored as blocks on the slave nodes.
- Blocks are simply files (blk_XXXXXXX) on the slave nodes underlying filesystem.
- Each block is stored on multiple nodes for redundancy, default value is three.
- Each slave node runs a DataNode daemon that controls access to the blocks and communicates with Namenode.

Secondary Namenode

Secondary Namenode is not a failover Namenode.

- NameNode is responsible for the reliable storage and interactive lookup and modification of the metadata for HDFS.
- To maintain interactive speed, the filesystem metadata is stored in the NameNode's RAM.
- Storing the data reliably necessitates writing it to disk as well.

Cont..

- To ensure that writes do not become a speed bottleneck, instead of storing the current snapshot of the filesystem every time, a list of modifications is continually appended to a log file called the EditLog.
- The Secondary NameNode periodically compacts the EditLog into a “checkpoint;” the EditLog is then cleared.
- Without this compaction process, restarting the NameNode can take a very long time.

Checkpointing the File System Metadata

The Secondary NameNode periodically checkpoints-the NameNode's in-memory file system data.

1. Tells the NameNode to roll its edits file.
2. Retrieves fsimage and edits from the Namenode
3. Loads fsimage into memory and applies the changes from the edits file
4. Creates a new, consolidated fsimage file

Cont..

5. Sends the new fsimage file back to the primary NameNode
6. The NameNode replaces the old fsimage file with the new one, replaces the old edits file with the new one it created in step 1

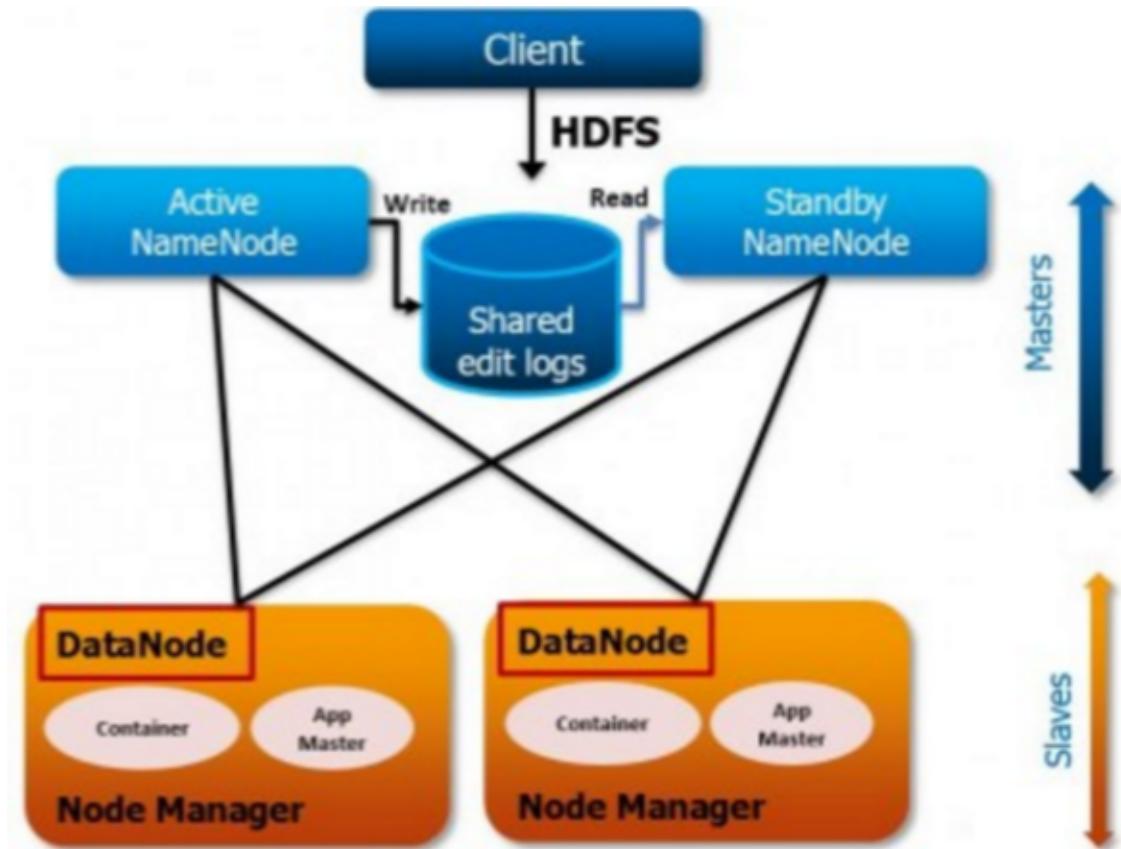
By default, checkpointing occurs once an hour or for every 1,000,000 transactions, whichever occurs sooner

Single Point of Failure

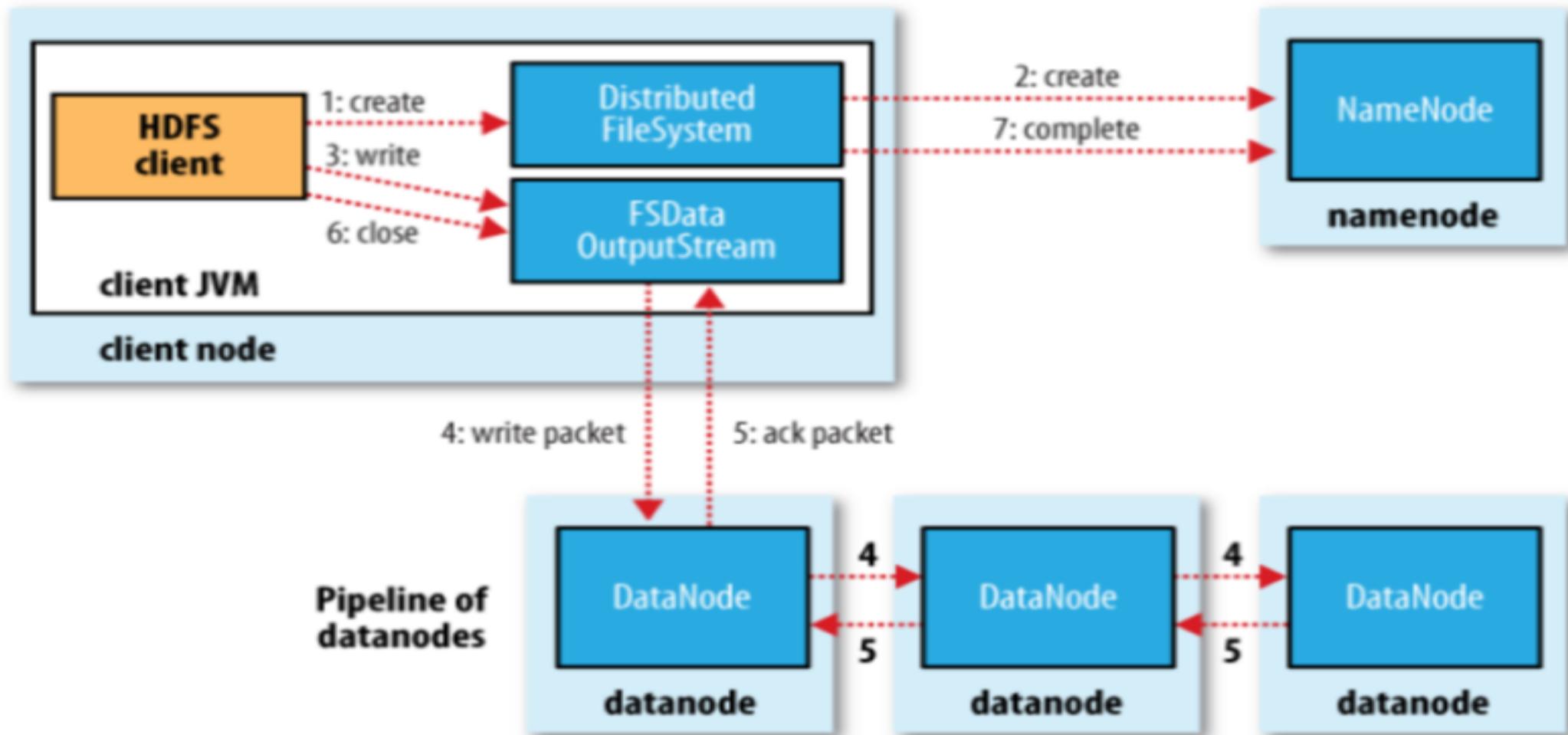
- In HDFS without high availability, NameNode is single point of failure.
- HDFS will be unavailable until NameNode is replaced.

HDFS with High Availability

- To eliminate the Namenode's single point of failure, deploy HDFS with high availability.
- Secondary Namenode is not needed.
- Two Namenodes are configured: one active and one standby
 - Standby Namenode takes over when active Namenode fails.



Writing Files



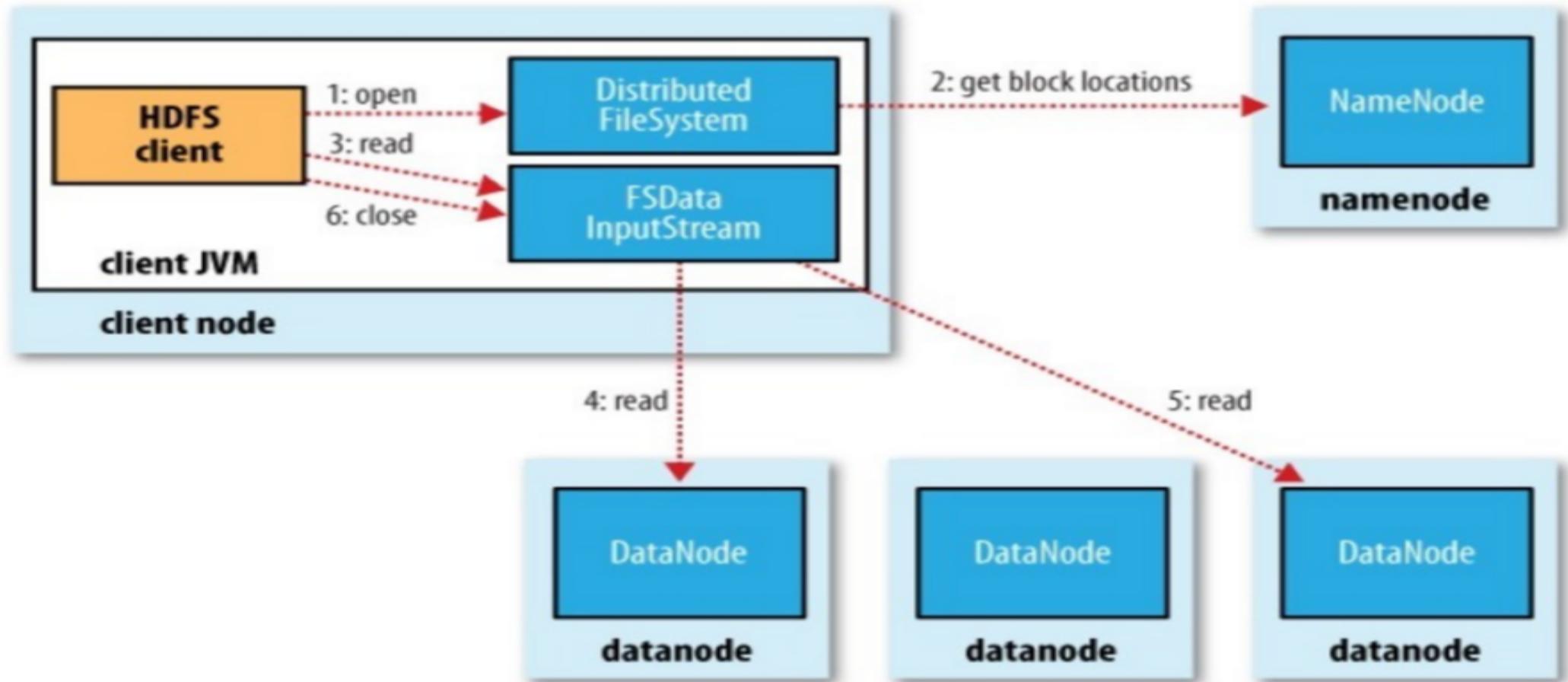
Flow of File Write

1. Client connects to the NameNode
2. NameNode places an entry for the file in its metadata, returns the block name and list of DataNodes to the client
3. Client connects to the first DataNode and starts sending data.
4. As data is received by the first DataNode, it connects to the second and starts sending data
5. Second DataNode similarly connects to the third
6. Ack packets from the pipeline are sent back to the client
7. Client reports to the NameNode when the block is written

In case of Failure:

- If a DataNode in the pipeline fails
 - The pipeline is closed
 - A new pipeline is opened with the two good nodes.
 - The data continues to be written to the two good nodes in the pipeline.
 - The Namenode will realize that the block is under replicated, and will re-replicate it to another Datanode.
- As blocks of data are written, the client calculates the checksum for each block
 - Sent to the DataNode alone with the data
 - Written together with each data block
 - Used to ensure the integrity of the data when it is later read.

Reading Files



Flow of File Read

1. Client connects to the NameNode.
2. NameNode returns the name and location of the first few blocks of the file
3. Client connects to the first of the DataNodes, and reads the block

If the DataNodes fails during the read, the client will seamlessly connect to the next one in the list to read the block.

NameNode Memory Consideration

- The Namenode serves all of its metadata directly from RAM.
- The metadata contains the filename, permissions, owner and group data, list of blocks that make up each file, and current known location of each replica of each block.
- As a base rule of thumb, the Namenode consumes roughly 1 GB for every 1 million blocks.

Overview of HDFS Security

- The Hadoop daemons leverage Kerberos to perform user authentication on all remote procedure calls (RPCs).
- Group resolution is performed on the Hadoop master nodes, NameNode, JobTracker and ResourceManager to guarantee that group membership cannot be manipulated by users.
- Map tasks are run under the user account of the user who submitted the job, ensuring isolation there.
- Authorization mechanisms have been introduced to HDFS and MapReduce to enable more control over user access to data.

Web UI for HDFS

NameNode Web UI can be accessed on port 50070

The screenshot shows the HDFS NameNode Web UI. The top navigation bar includes links for Hadoop, Overview, Datanodes, Snapshot, Startup Progress, and Utilities. The main content area is titled "Overview". It displays several key cluster statistics in a table:

Started:	Mon Mar 31 14:34:46 EDT 2014
Version:	2.3.0-cdh5.0.0-SNAPSHOT, r8e266e052e423af592871e2dfe09d54c03f6a0e8
Compiled:	2014-03-25T23:19Z by jenkins from (no branch)
Cluster ID:	CID-a4ab0c19-9b8b-4ba7-aa6b-8427f58d16f3
Block Pool ID:	BP-1744186959-10.161.98.12-1396289656873

Below the table, the "Summary" section contains the following information:

- Security is off.
- Safemode is off.
- 64 files and directories, 25 blocks = 89 total filesystem object(s).
- Heap Memory used 22.17 MB of 29.48 MB Heap Memory. Max Heap Memory is 61.88 MB.

Resource Manager Web UI can be accessed on port 8088

The screenshot shows the Hadoop Resource Manager Web UI. At the top left is the Hadoop logo. The main title is "All Applications". On the left, there's a sidebar with a "Cluster" section containing links for "About", "Nodes", "Applications" (which is expanded to show "NEW", "NEW_SAVING", "SUBMITTED", "ACCEPTED", "RUNNING", "FINISHED", "FAILED", "KILLED"), "Scheduler", and "Tools". The main content area has two tables under "Cluster Metrics" and "User Metrics for dr.who". Both tables have columns for Apps Submitted, Apps Pending, Apps Running, Apps Completed, Containers Running, Memory Used, Memory Total, Memory Reserved, Vcores Used, Vcores Total, Vcores Reserved, Active Nodes, Decommissioned Nodes, and Lost Nodes. Below the tables is a search bar with "Show 20 entries" and a "Search" button. At the bottom, it says "No data available in table" and "Showing 0 to 0 of 0 entries".

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	Vcores Used	Vcores Total	Vcores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes
0	0	0	0	0	0 B	32 GB	0 B	0	32	0	4	0	0

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Containers Pending	Containers Reserved	Memory Used	Memory Pending	Memory Reserved	Vcores Used	Vcores Pending	Vcores Reserved	Vcores Total
0	0	0	0	0	0	0	0 B	0 B	0 B	0	0	0	0

Show 20 entries

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Running Containers	Allocated CPU Vcores	Allocated Memory MB	Progress

No data available in table

Showing 0 to 0 of 0 entries

JobHistoryServer Web UI can be accessed on port 19888

The screenshot shows the Hadoop JobHistory Server Web UI. At the top left is the Hadoop logo with a yellow elephant icon and the word "hadoop". At the top right, it says "Logged in as: dr.who". The main title is "JobHistory". On the left, there's a sidebar with "Application" expanded, showing "About Jobs" and "Tools". The "Tools" section is highlighted with a grey background. The main content area is titled "Retired Jobs". It features a table header with columns: Submit Time, Start Time, Finish Time, Job ID, Name, User, Queue, State, Maps Total, Maps Completed, Reduces Total, and Reduces Completed. Below the header, a message says "No data available in table". At the bottom, there are links for "Submit Time", "Start Time", "Finish Time", "Job ID", "Name", "User", "Queue", "State", "Maps Total", "Maps Comple", "Reduces Total", and "Reduces Con". At the very bottom, it says "Showing 0 to 0 of 0 entries" and has links for "First", "Previous", "Next", and "Last".

Default Hadoop Ports

	Daemon	Default Port
HDFS	Namenode	50070
	Datanodes	50075
	Secondary Namenode	50090
	Checkpoint Node	50105
MR	JobTracker	50030
	TaskTracker	50060

Using Hadoop File Shell

- Hadoop filesystem can be accessed via command line and HUE browser.
- End users typically access HDFS via `hadoop fs` command.
- Most of the subcommands are similar to corresponding UNIX commands

Note: The directory structure of Hadoop filesystem is different from local filesystem i.e. Linux filesystem.

Accessing HDFS via Command Line

- To list the content of root directory

```
hdfs dfs -ls /
```

- To make a directory named “Hadoop” under root directory

```
hdfs dfs –mkdir /Hadoop
```

- To create a sample text file, use put command that will copy local files to HDFS.

```
hdfs dfs -put sample.txt /Hadoop
```

Cont..

- Use get command to fetch local copy of a file from HDFS

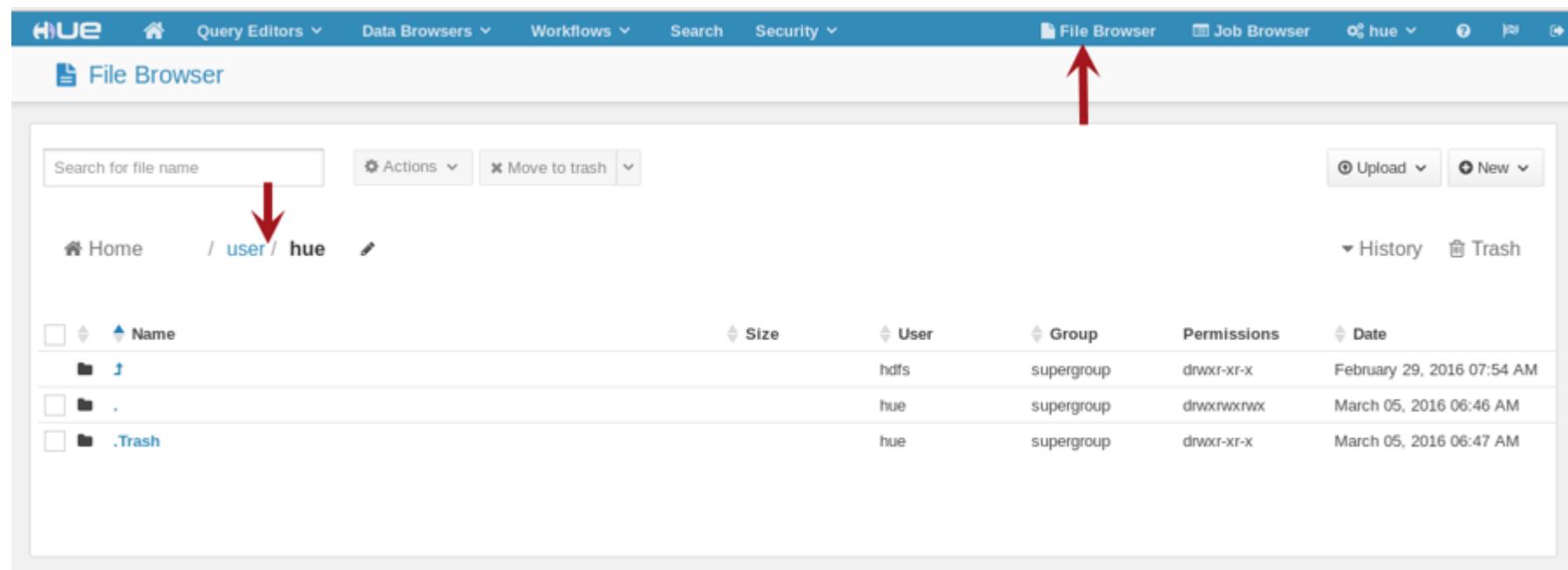
```
hdfs dfs -get /Hadoop/sample.txt
```

- To remove a file

```
hdfs dfs -rm /Hadoop/sample.txt
```

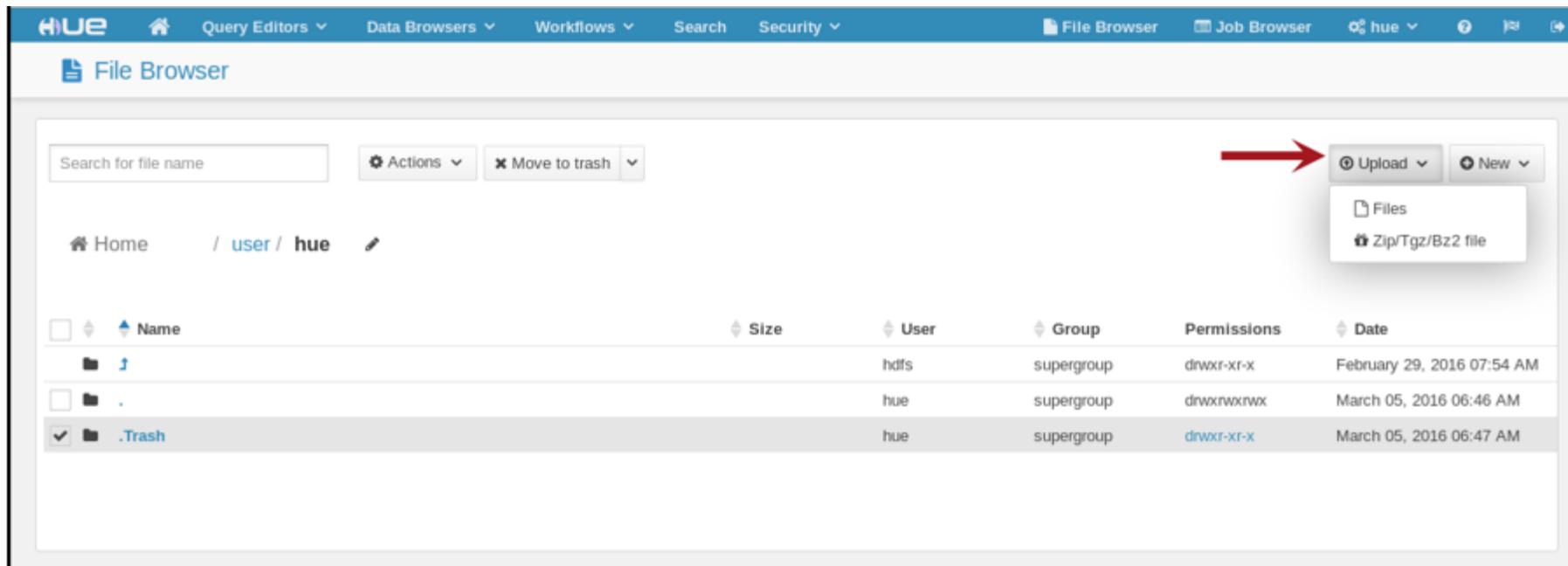
Accessing HDFS via HUE Browser

Navigate to File Browser tab



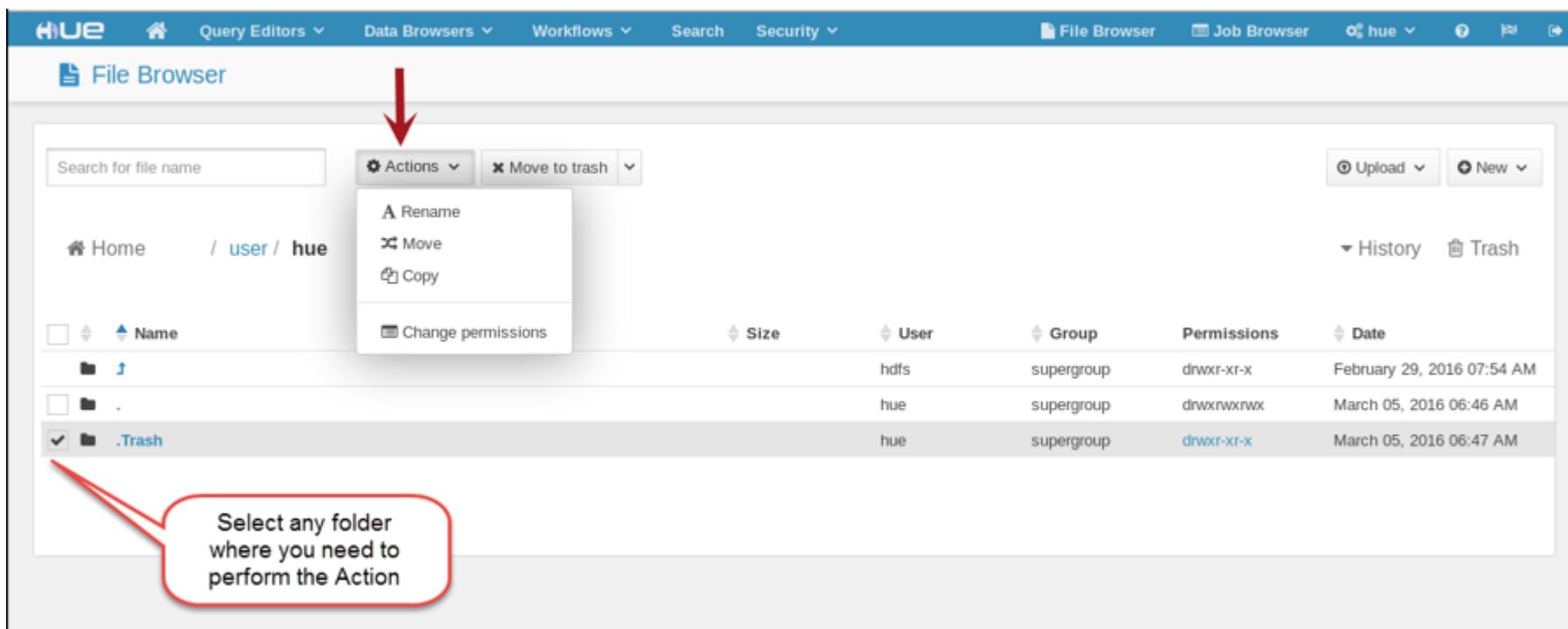
Cont..

To upload file from local filesystem to HDFS



Cont..

To perform COPY, MOVE & RENAME operations



Quick Overview

- HDFS provides fault tolerance with built in data redundancy.
- Block size is configurable.
- The NameNode daemon maintains all HDFS metadata in memory and stores it on disk.
- NameNode is single point of failure if high availability is not enabled.



MapReduce and Spark on YARN

We will cover following topics:

- Role of Computational Framework
- YARN: The cluster resource manager
- MapReduce Concepts
- Apache Spark Concepts
- Running Computational Framework on YARN
- Exploring YARN application through Web UI and shell
- YARN application logs

Role of Computation Framework

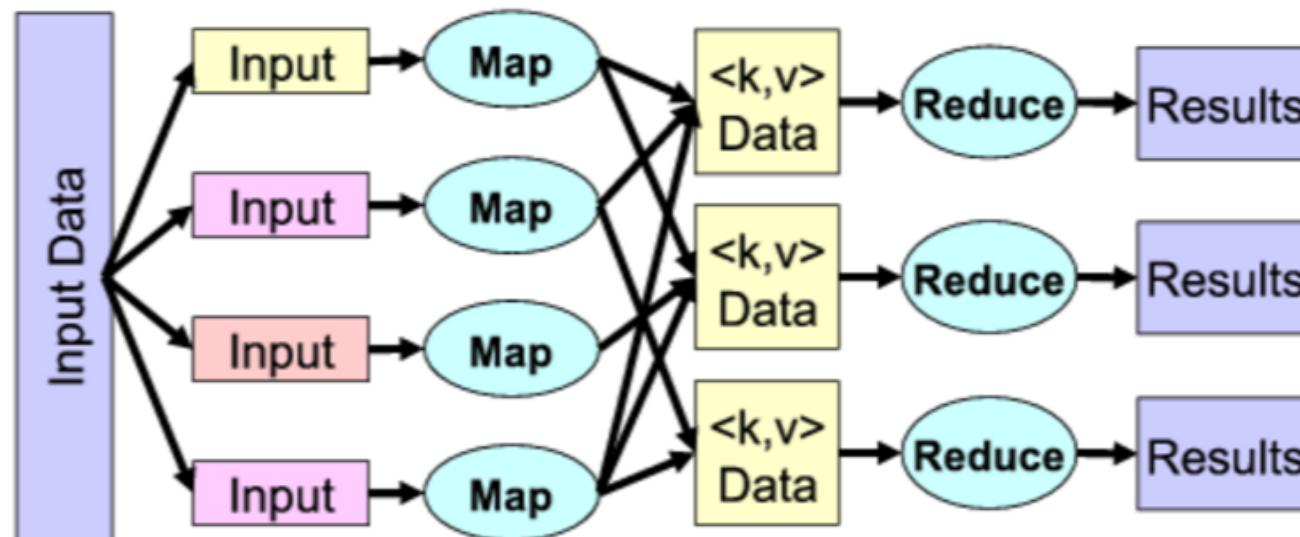
- CDH supports two version of MapReduce computation framework:
 - MRv1 (MapReduce)
 - MRv2 (YARN)
- In CDH 4 cluster, MapReduce service is default MapReduce computation framework.
- In CDH5 cluster, YARN service is the default MapReduce computation framework.

MapReduce

- MapReduce is a processing technique and a program model for distributed computing based on java.
- The MapReduce algorithm contains two important tasks, namely Map and Reduce.
- Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs).
- Reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples.

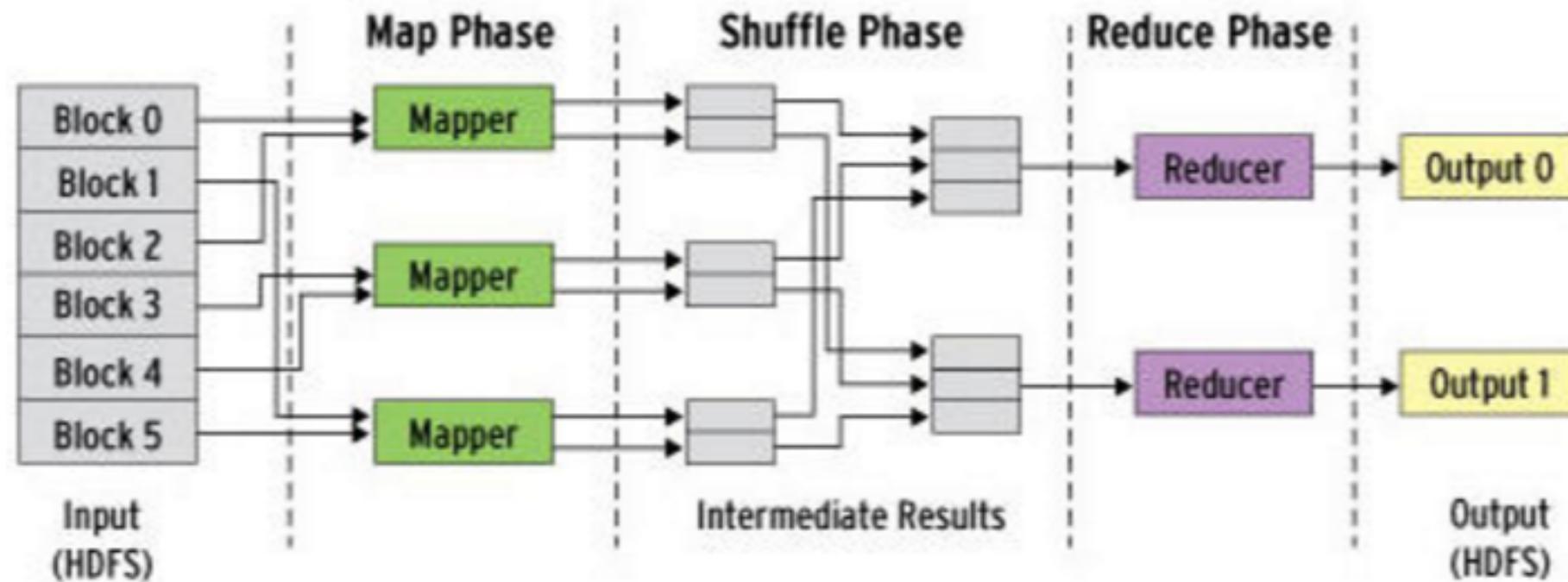
Cont..

- The data processing primitives are called mappers and reducers.
- Scales data processing over multiple computing nodes



Cont..

- In between Map and Reduce is Shuffle and sort



How it works ?

MapReduce works similar to that of Unix pipeline

```
cat /my/log | grep '\.html' | sort | uniq -c > /my/outfile
```

Map

Shuffle
and Sort

Reduce

Roles

- **Map:** Each worker node applies the "map()" function to the local data, and writes the output to a temporary storage. A master node ensures that only one copy of redundant input data is processed.
- **Shuffle:** Worker nodes redistribute data based on the output keys (produced by the "map()" function), such that all data belonging to one key is located on the same worker node.
- **Reduce:** Worker nodes now process each group of output data, per key, in parallel.

MapReduce Computation

- **Prepare the Map() input** – the "MapReduce system" designates Map processors, assigns the input key value $K1$ that each processor would work on, and provides that processor with all the input data associated with that key value.
- **Run the user-provided Map() code** – Map() is run exactly once for each $K1$ key value, generating output organized by key values $K2$.

Cont

- "**Shuffle**" the Map output to the Reduce processors – the MapReduce system designates Reduce processors, assigns the K_2 key value each processor should work on, and provides that processor with all the Map-generated data associated with that key value.
- **Run the user-provided Reduce() code** – Reduce() is run exactly once for each K_2 key value produced by the Map step.
- **Produce the final output** – the MapReduce system collects all the Reduce output, and sorts it by K_2 to produce the final outcome.

Key Points

- Each mapper process is single input split from HDFS.
- Hadoop passes one record at a time to developer's Mapper code.
- Each record has a key and a value.
- Intermediate data is written by the Mapper to local disk.
- During shuffle and sort phase, all the values associated with the same intermediate key are transferred to the same Reducer.
- Keys and its associated values are passed to Reducer.
- Output from Reducer is written to HDFS.

Features of MapReduce

- Automatic parallelization and distribution.
- Fault tolerance.
- Status and monitoring tools.
- MapReduce programs are usually written in JAVA, can be written in other languages as well.
- MapReduce abstracts all the ‘housekeeping’ away from developer.

MapReduce: Example

WordCount is the “Hello World!” of Hadoop.

Step1: Map

Assume input is set of text files, k is a byte offset and v is the line for that offset.

```
let map (k,v) = foreach word in v: emit  
                (word,1)
```

Cont..

Step2: Input to the Mapper

Some students like to study in morning
Tom plays football in morning

Step3: Intermediate data produced

(some,1), (students,1), (like,1), (study,1), (in,1), (morning,1), (Tom,1),
(plays,1), (football,1), (in,1), (morning,1)

Cont..

Step4: Input to the Reducer

```
(football,[1])  
(in,[1,1])  
(like,[1])  
(morning,[1,1])  
(plays,[1])  
(some,[1])  
(students,[1])  
(study,[1])  
(Tom,[1]),
```

Cont..

Step5: Reduce

k is a word, vals is a list of one's

```
let reduce (k, vals) =  
    sum = 0  
    foreach (v in vals) :  
        sum = sum + v  
    emit (k, sum)
```

Cont..

Step6: Output from Reducer, written to HDFS

```
(football,1)
(in,2)
(like,1)
(morning,2)
(plays,1)
(some,1)
(students,1)
(study,1)
(Tom,1),
```

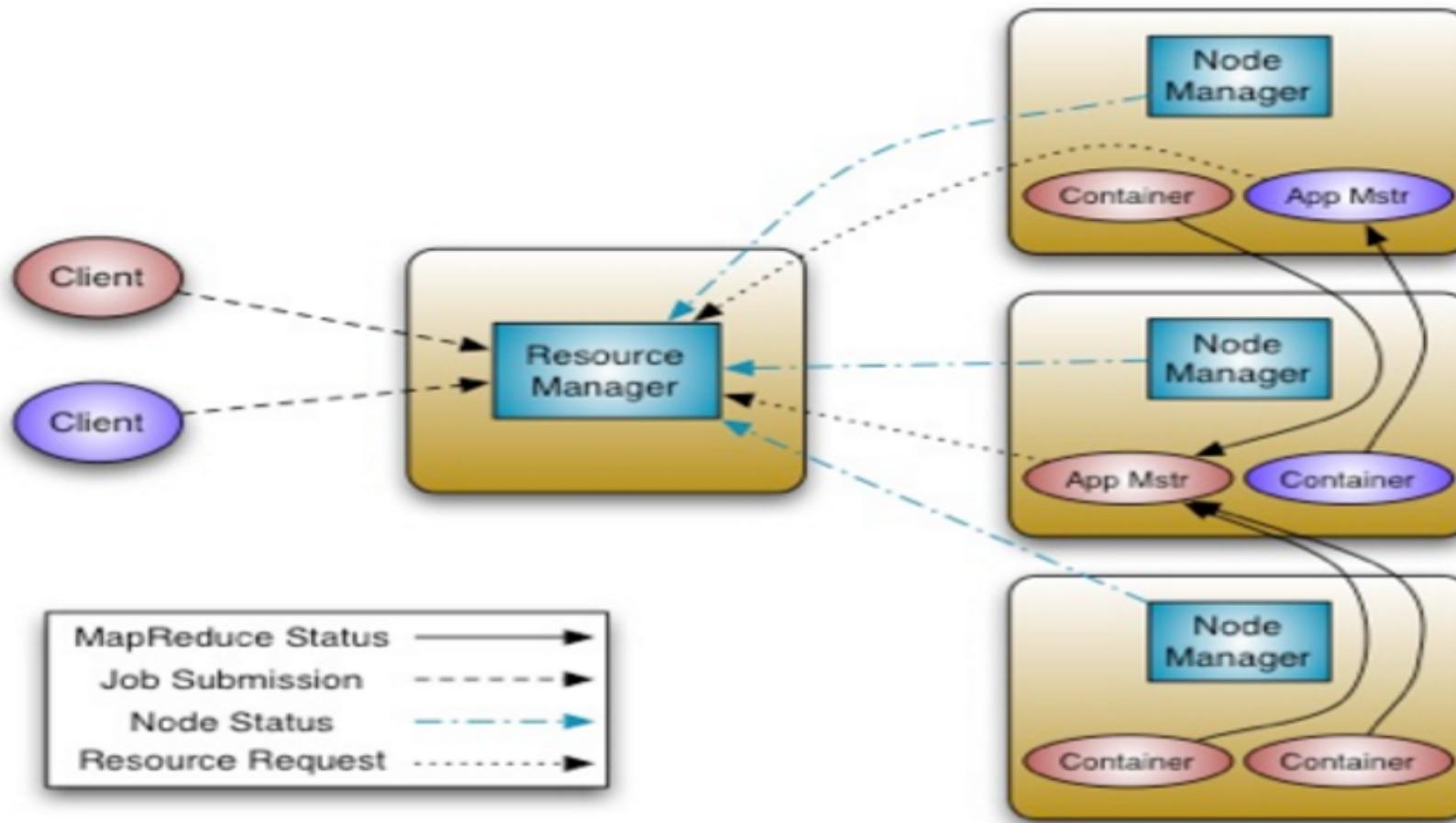
YARN: The Cluster Resource Manager

YARN: Yet Another Resource Negotiator

(Uses ResourceManager/NodeManager architecture)

- The fundamental idea of YARN is to split up the functionalities of resource management and job scheduling/monitoring into separate daemons.

YARN: Architecture



YARN Daemons

Resource Manager: It is responsible for tracking the resources in a cluster, and scheduling applications.

- It works together with per node NodeManager and per-application ApplicationMasters.
- It effectively replaces the functions of the JobTracker.

NodeManager: takes instructions from the ResourceManager and manage resources available on a single node.

- NodeManagers run on slave nodes instead of TaskTracker daemons.

Cont..

ApplicationMasters: are responsible for negotiating resources with the ResourceManager and for working with the NodeManager to start the containers.

- Accepts job-submissions
- Provides service for restarting the ApplicationMaster container on failure.

Cont..

Containers: Allocated by ResourceManger

- A Container grants rights to an application to use a specific amount of resources (memory, cpu etc.) on a specific host.

JobHistoryServer: Responsible for servicing all job history related requests from client. Archives jobs metrics and metadata.

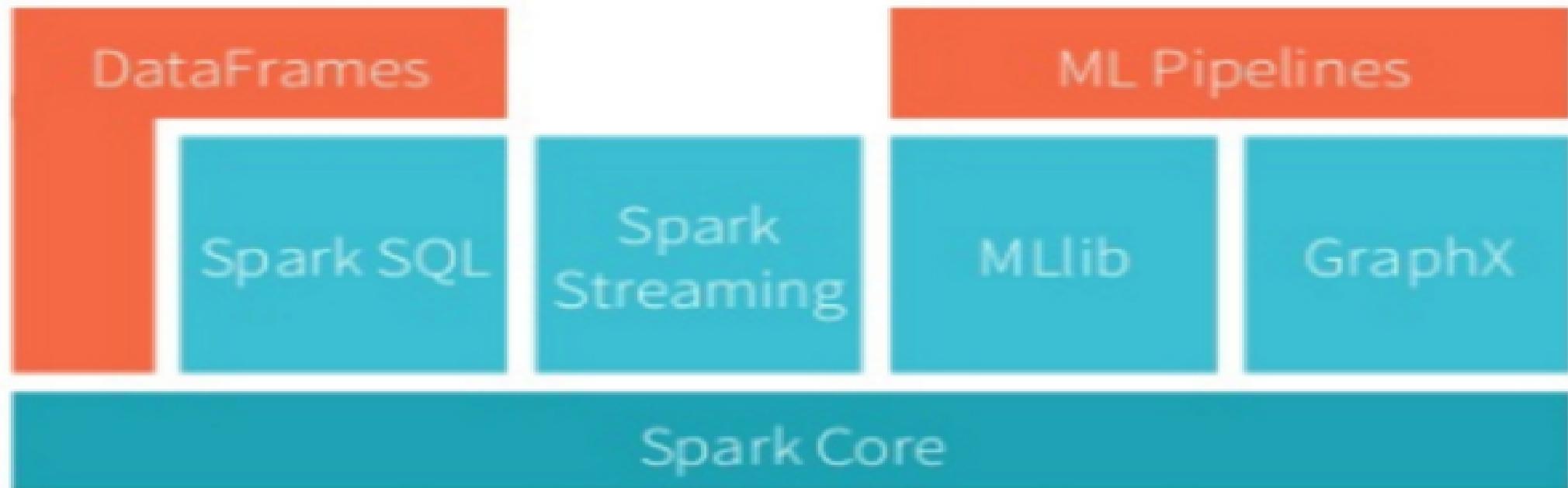
Apache Spark

- Apache Spark is a fast and general engine for large scale data processing.
- Apache Spark is a general-purpose cluster in-memory computing system.
- Originally developed at the University of California, Berkeley's [AMPLab](#), the Spark codebase was later donated to the Apache Software Foundation.

Cont..

- Apache Spark provides programmers with an application programming interface centered on a data structure called the resilient distributed dataset (RDD).
- RDD is a read-only multiset of data items distributed over a cluster of machines, that is maintained in a fault-tolerant way.

Spark Ecosystem



Spark Components

- **Spark Core** is the base of the whole project. It provides distributed task dispatching, scheduling, and basic I/O functionalities.
- **Spark SQL** is a component on top of Spark Core that introduces a new data abstraction called `DataFrames`.
 - Provides support for structured and semi-structured data.
 - Spark SQL provides a domain-specific language to manipulate `DataFrames` in Scala, Java, or Python.

Cont..

- **Spark Streaming** leverages Spark Core's fast scheduling capability to perform streaming analytics.
 - It ingests data in mini-batches and performs RDD transformations on those mini-batches of data.
 - Spark Streaming has support built-in to consume from Kafka, Flume, Twitter, [ZeroMQ](#), Kinesis, and TCP/IP sockets.

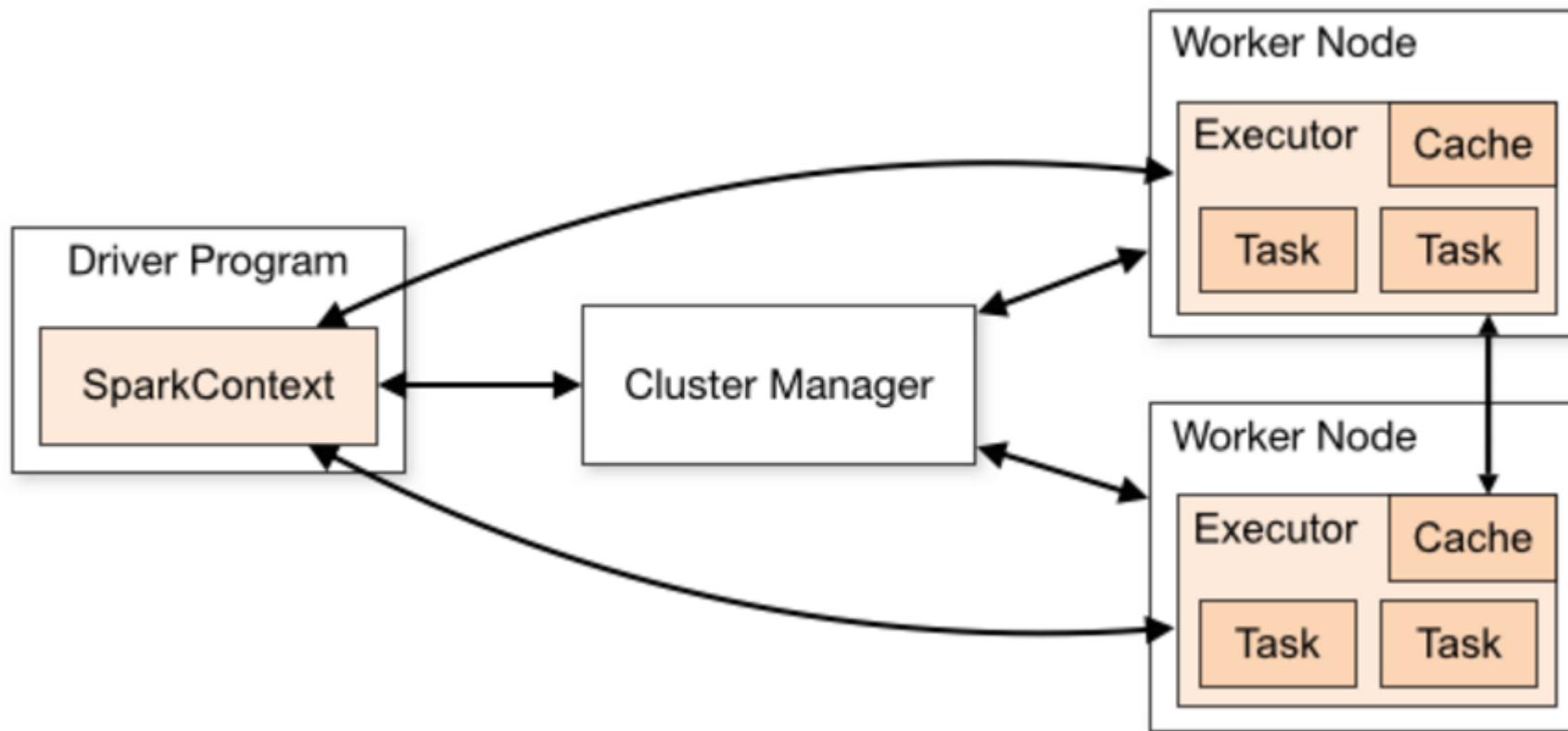
Cont..

- **Spark MLlib** is a distributed machine learning framework on top of Spark Core
 - The distributed memory-based Spark architecture, is as much as nine times as fast as the disk-based implementation used by Apache Mahout

Cont..

- GraphX is a distributed graph processing framework on top of Apache Spark.
 - Because it is based on RDDs, which are immutable, graphs are immutable and thus GraphX is unsuitable for graphs that need to be updated.
 - GraphX can be viewed as being the Spark in-memory version of Apache Giraph, which utilized Hadoop disk-based MapReduce.

Spark Architecture



Components

- Driver Program: The process running the main() function of the application and creating the SparkContext.
- Worker node: Any node that can run application code in the cluster.
- Executor: A process launched for an application on a worker node, that runs tasks and keeps data in memory or disk storage across them. Each application has its own executors.
- Task: A unit of work that will be sent to one executor

Cluster Manager Types

The system currently supports three cluster managers:

- [Standalone](#) – a simple cluster manager included with Spark that makes it easy to set up a cluster.
- [Apache Mesos](#) – a general cluster manager that can also run Hadoop MapReduce and service applications.
- [Hadoop YARN](#) – the resource manager in Hadoop 2.

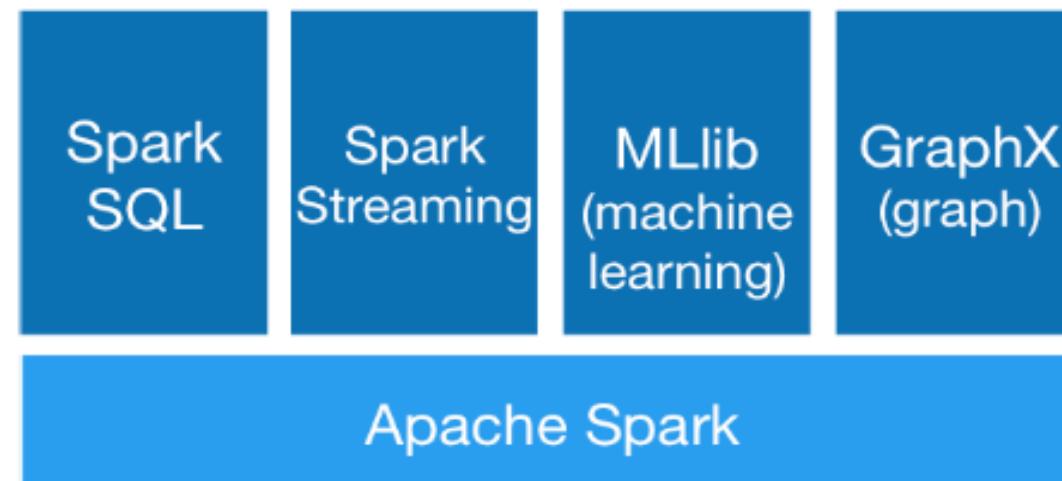
Features

Spark runs on Hadoop, Mesos, standalone, or in the cloud. It can access diverse data sources including HDFS, Cassandra, HBase, and S3.



Cont..

Spark powers a stack of libraries including SQL and DataFrames, [MLlib](#) for machine learning, [GraphX](#), and Spark Streaming.



Cont..

Spark offers over 80 high-level operators that make it easy to build parallel apps. And you can use it interactively from the Scala, Python and R shells.

Python

```
lines = sc.textFile(...)  
lines.filter(lambda s: "ERROR" in s).count()
```

Scala

```
val lines = sc.textFile(...)  
lines.filter(x => x.contains("ERROR")).count()
```

Java

```
JavaRDD<String> lines = sc.textFile(...);  
lines.filter(new Function<String, Boolean>() {  
    Boolean call(String s) {  
        return s.contains("error");  
    }  
}).count();
```

Standalone Programs

Python, Scala, & Java

Interactive Shells

Python & Scala

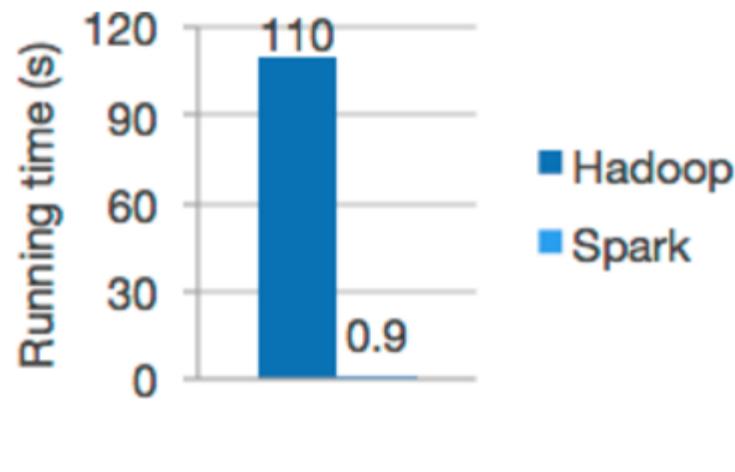
Performance

Java & Scala are faster due to static typing

...but Python is often fine

Cont..

- Run programs up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk.
- Apache Spark has an advanced DAG execution engine that supports cyclic data flow and in-memory computing.



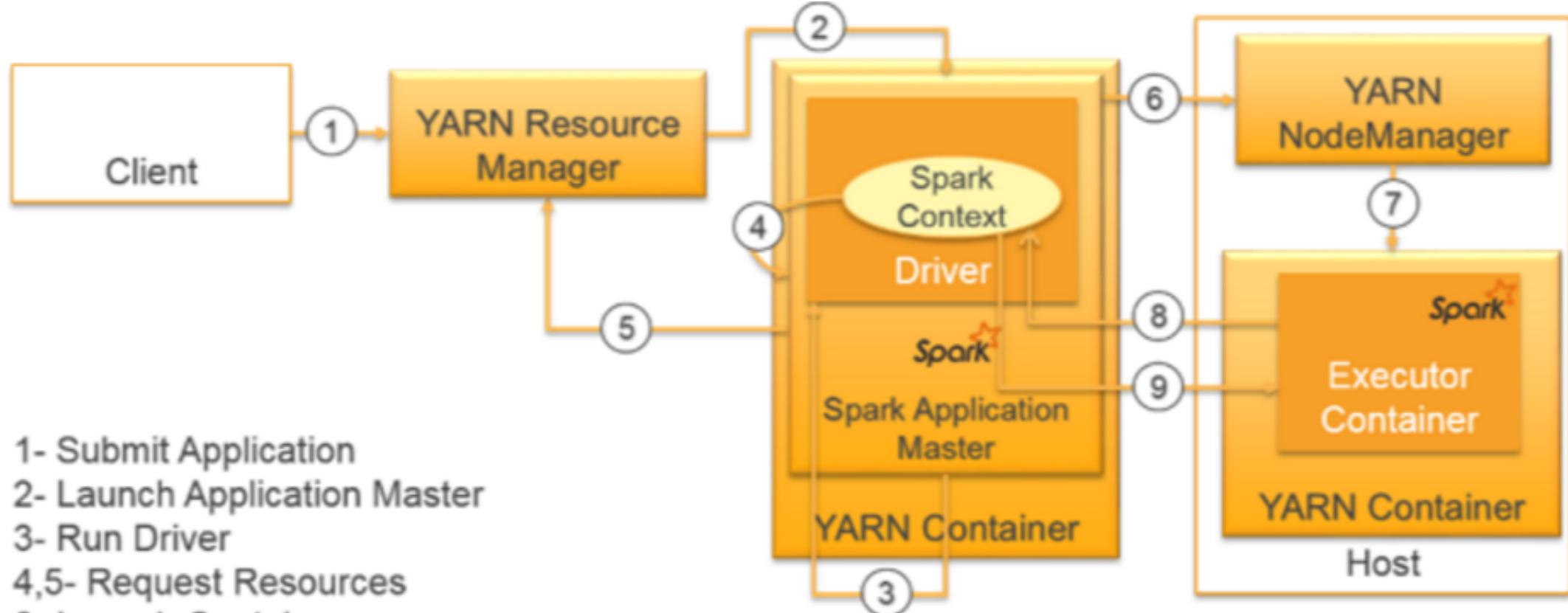
Functionality

- Spark application run as independent set of processes on a cluster, coordinated by `SparkContext` object in your main program (called driver program)
- `SparkContext` can connect to several types of cluster managers (either Spark's own standalone cluster manager, Mesos or YARN), which allocates resources across application.

Cont..

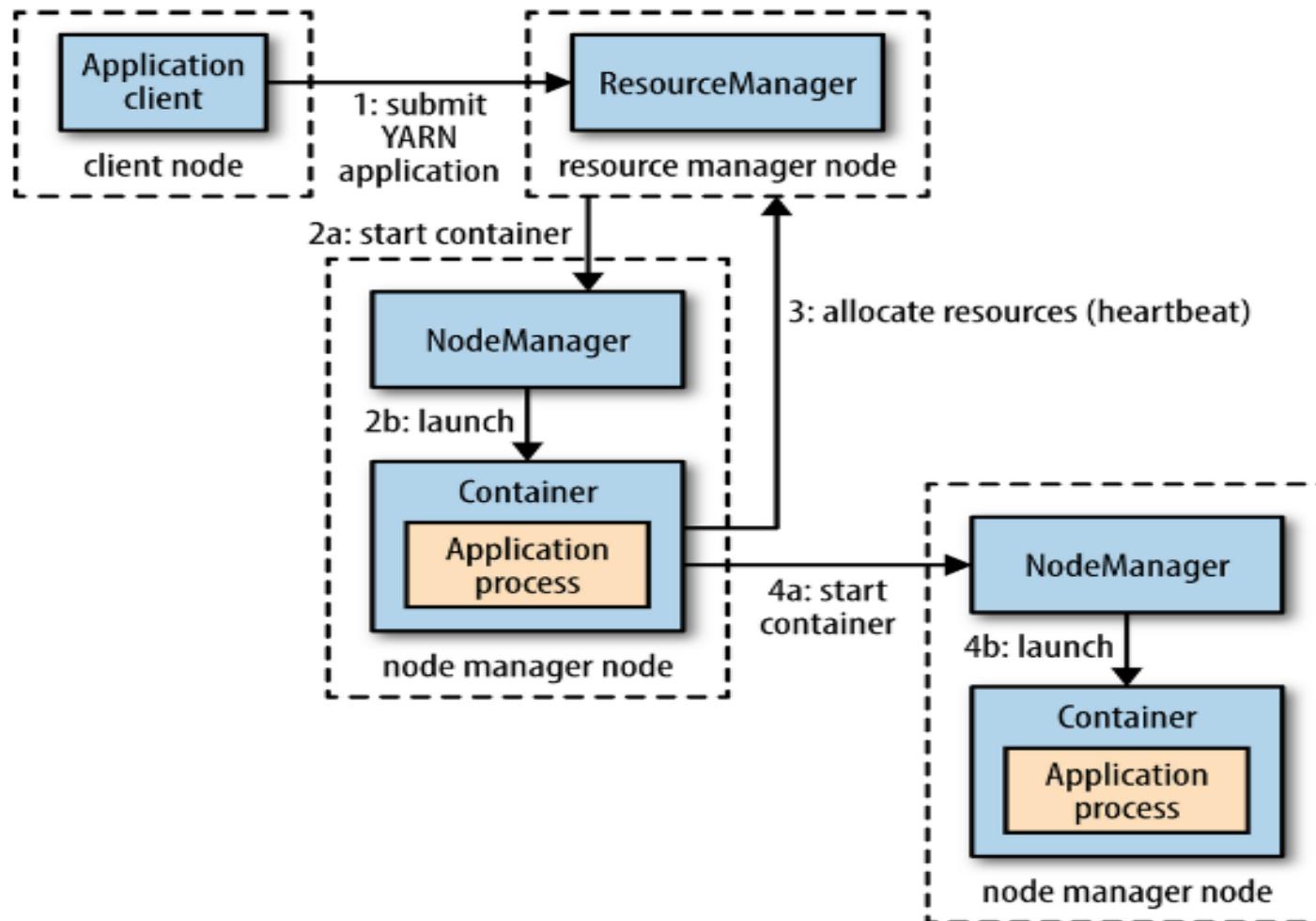
- Spark acquires executors on nodes in the cluster, which are processes that run computations and store data for your application.
- It sends your application code (defined by JAR or Python files passed to `SparkContext`) to the executors
- Finally, `SparkContext` sends tasks to the executors to run.

Spark on YARN



- 1- Submit Application
- 2- Launch Application Master
- 3- Run Driver
- 4,5- Request Resources
- 6- Launch Containers
- 7- Launch Spark Executors
- 8- Register with the Driver
- 9- Launch Tasks

Running Computational Framework on YARN



Working

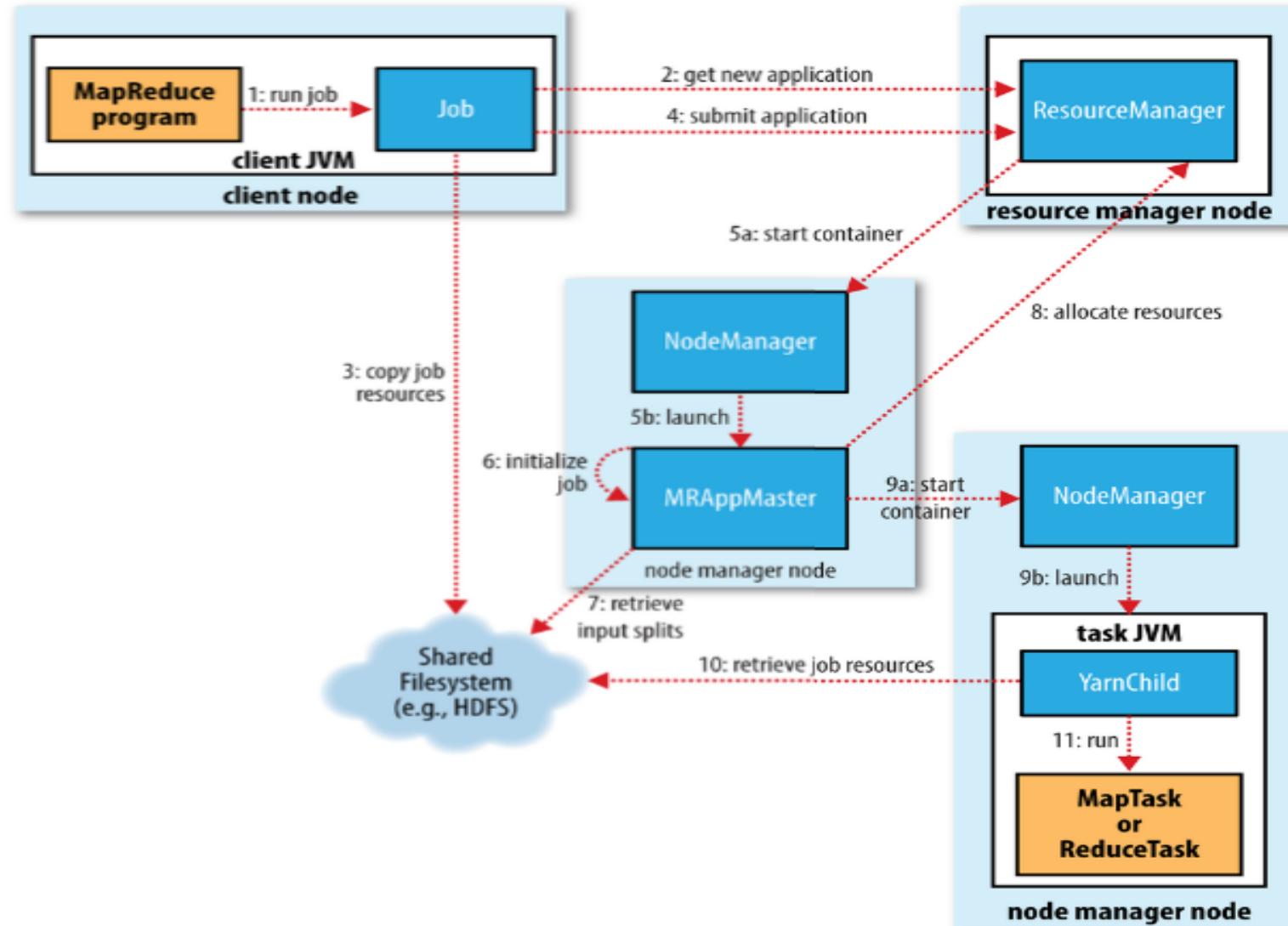
Step1: A client contacts the resource manager and asks it to run an application master process.

Step2: The resource manager then finds a node manager that can launch the application master in a container.

Step3: Request more containers from the resource managers.

Step4: Runs distributed computation

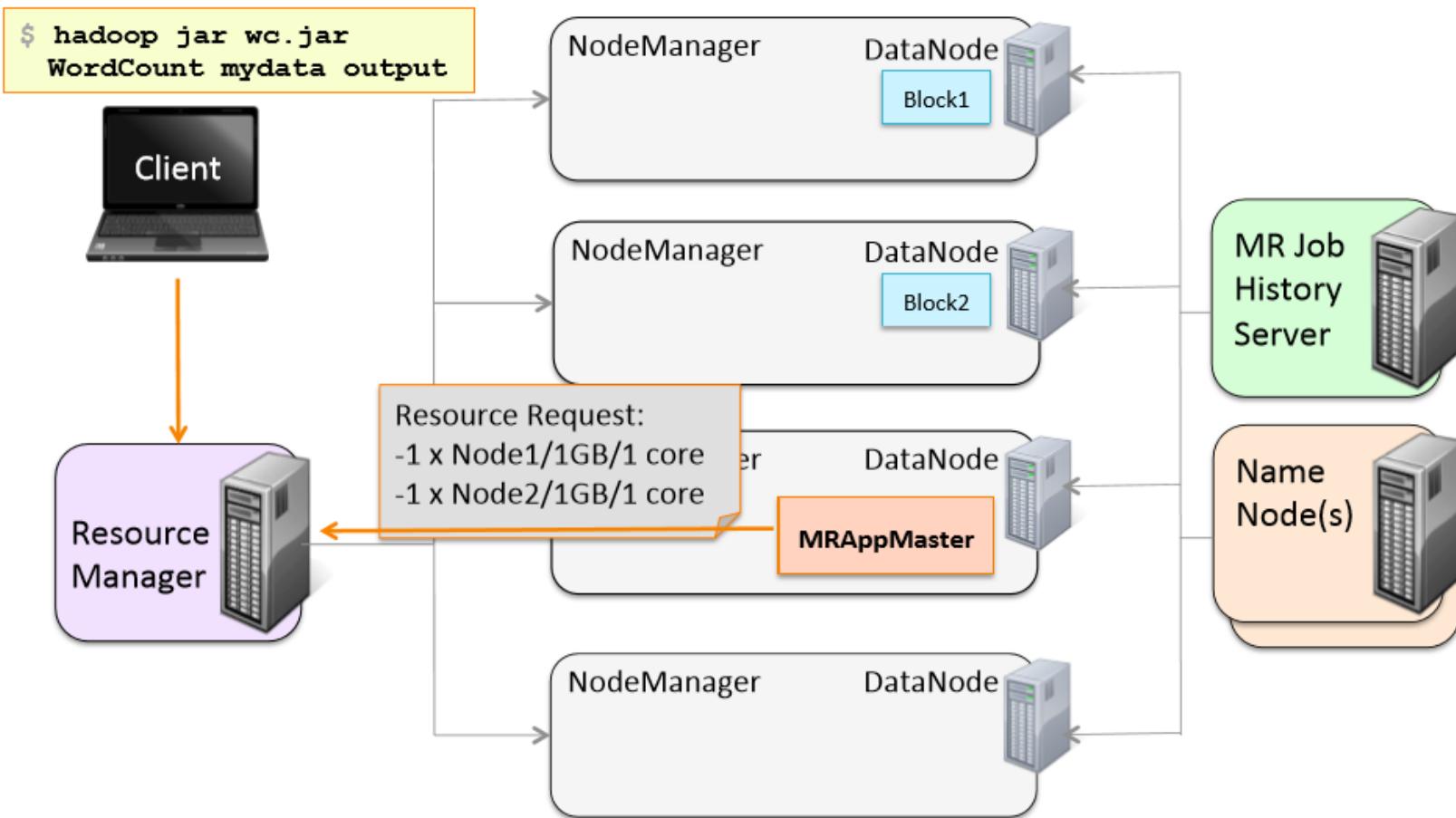
How Hadoop runs a Job ?



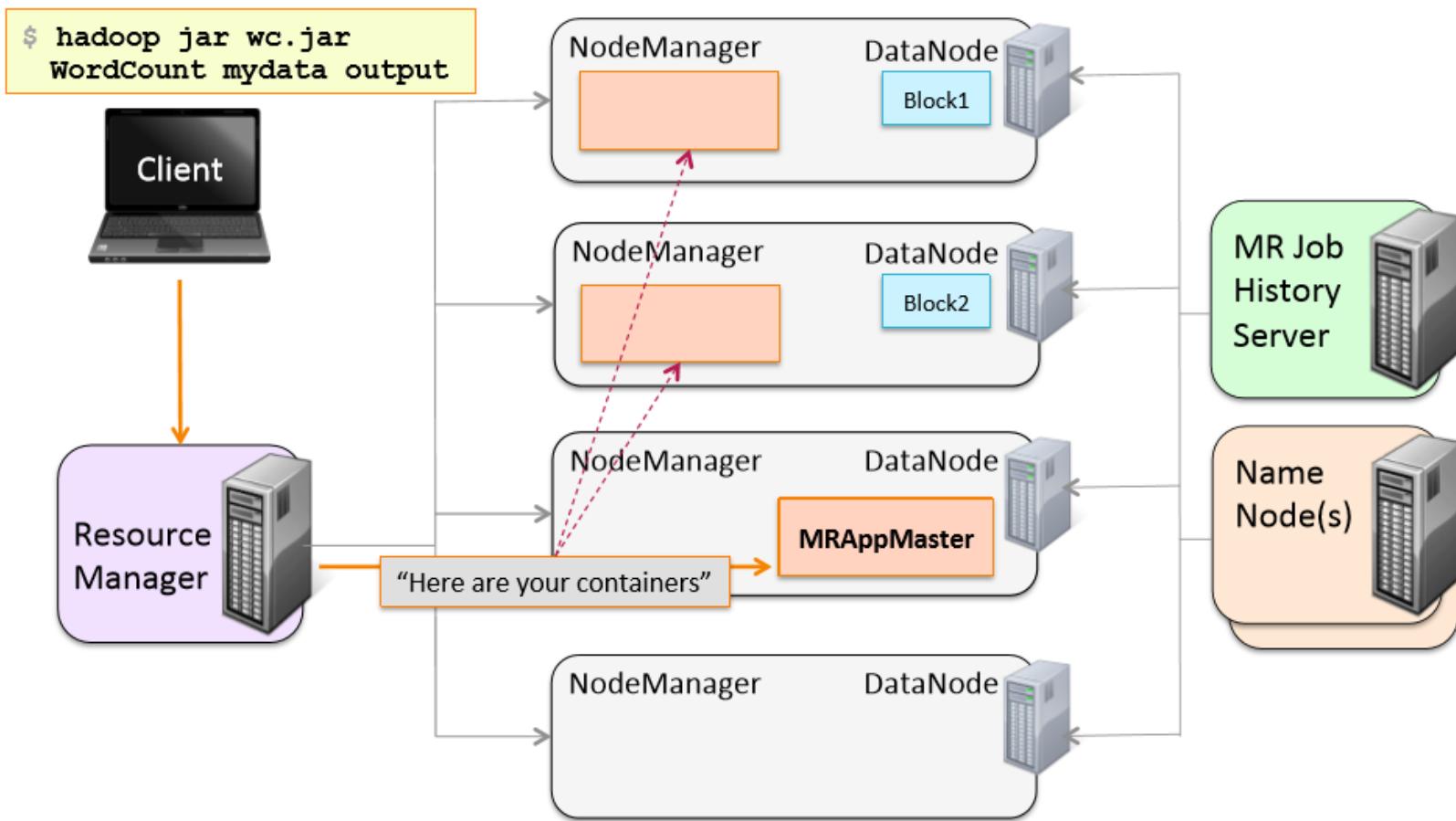
Role of each Component

- The client, which submits the MapReduce job.
- The YARN resource manager, which coordinates the allocation of compute resources on the cluster.
- The YARN node managers, which launch and monitor the compute containers on machines in the cluster.
- The MapReduce application master, which coordinates the tasks running the MapReduce job.
- The application master and the MapReduce tasks run in containers that are scheduled by the resource manager and managed by the node managers.
- The distributed filesystem, which is used for sharing job files between the other entities.

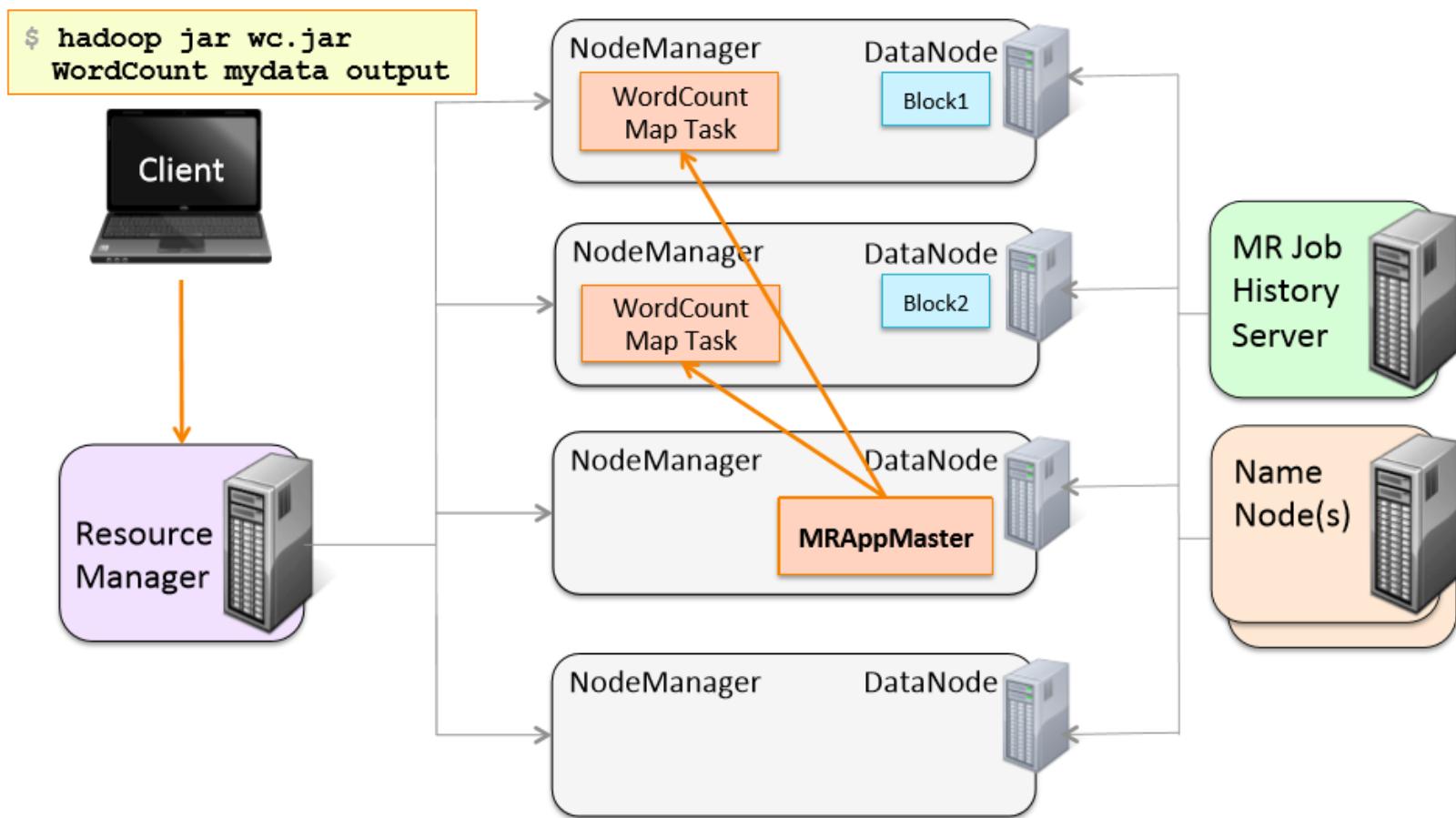
Map Tasks



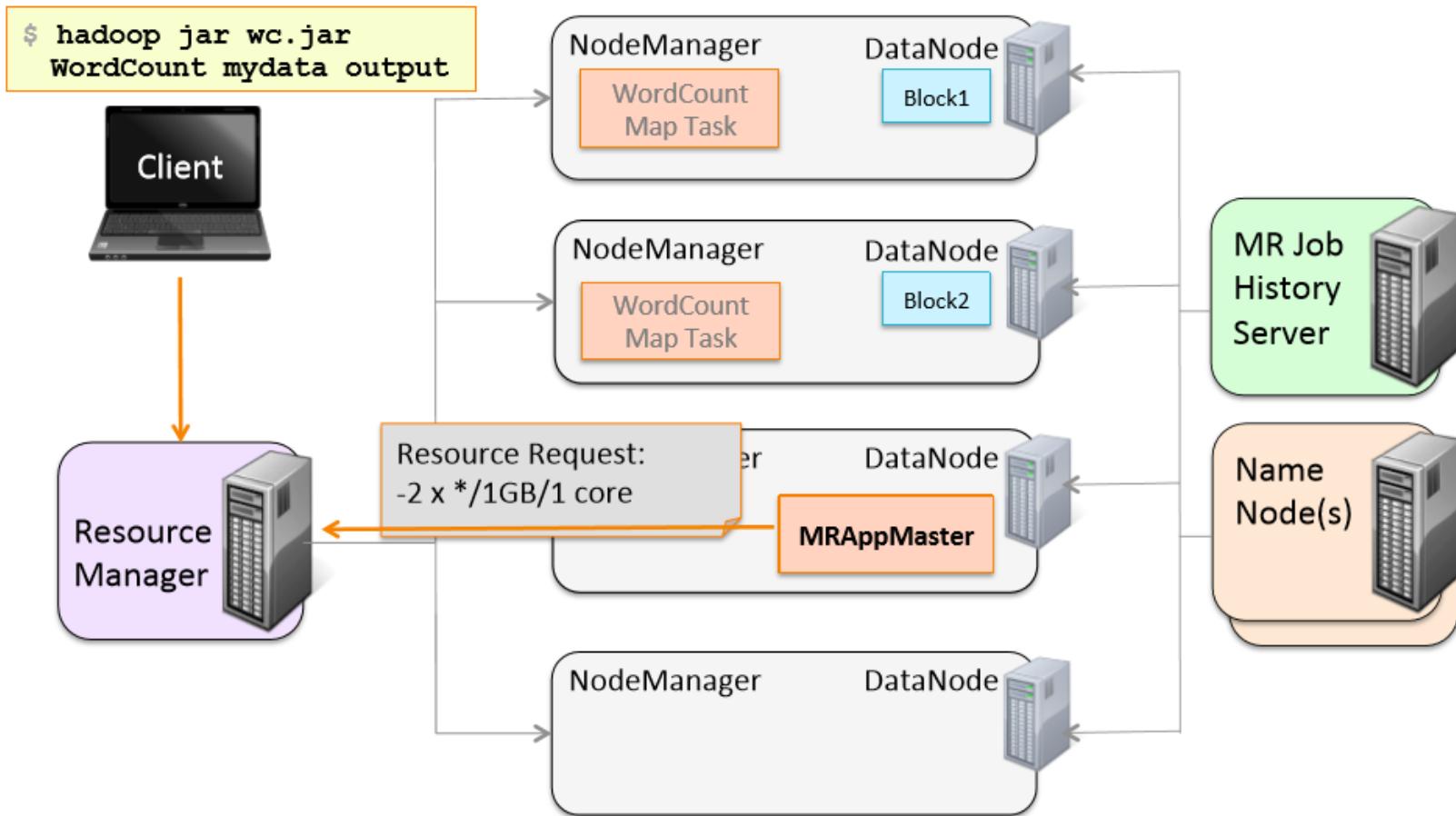
Allocating Container for Map Task



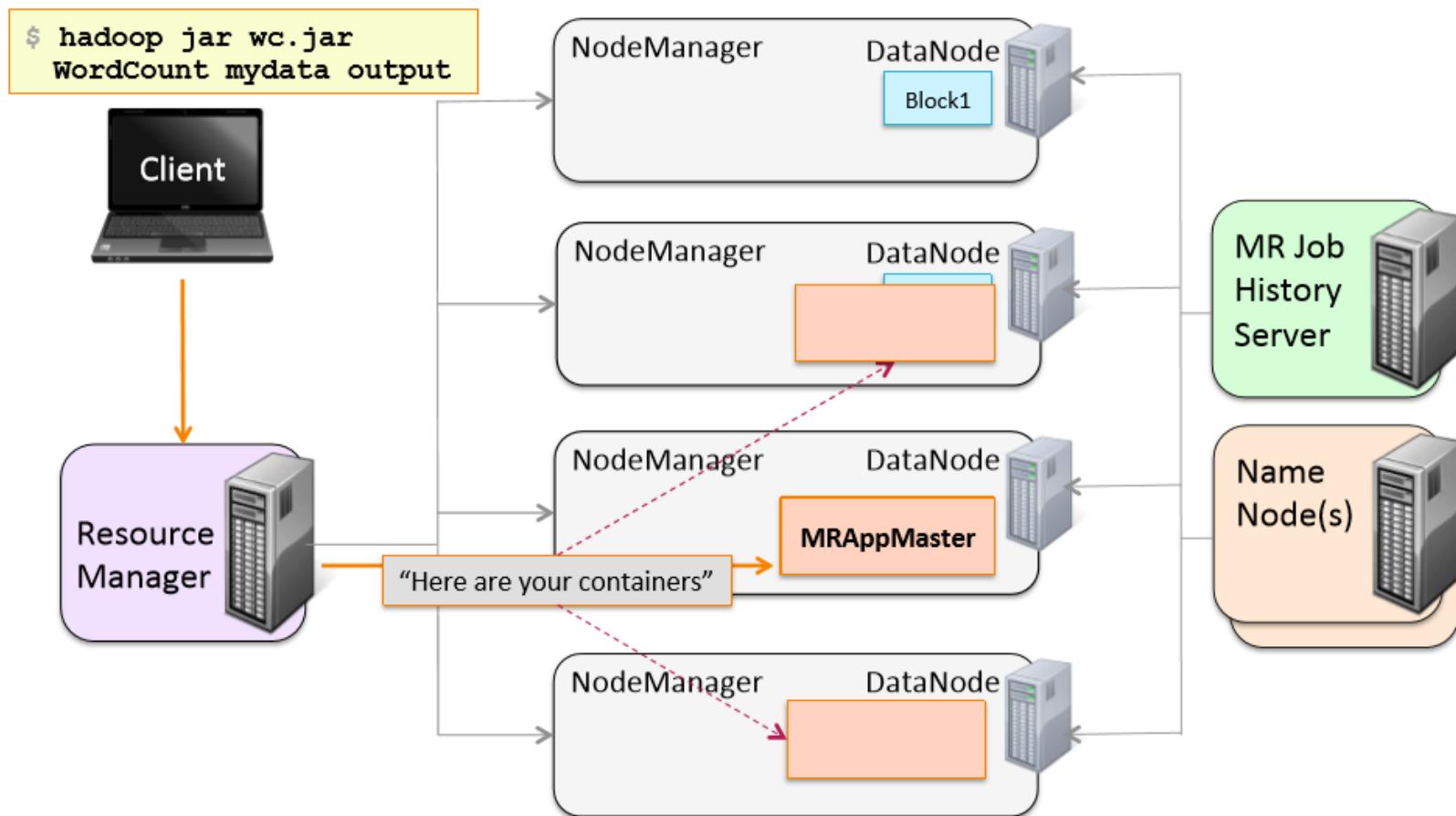
Executing Map Task



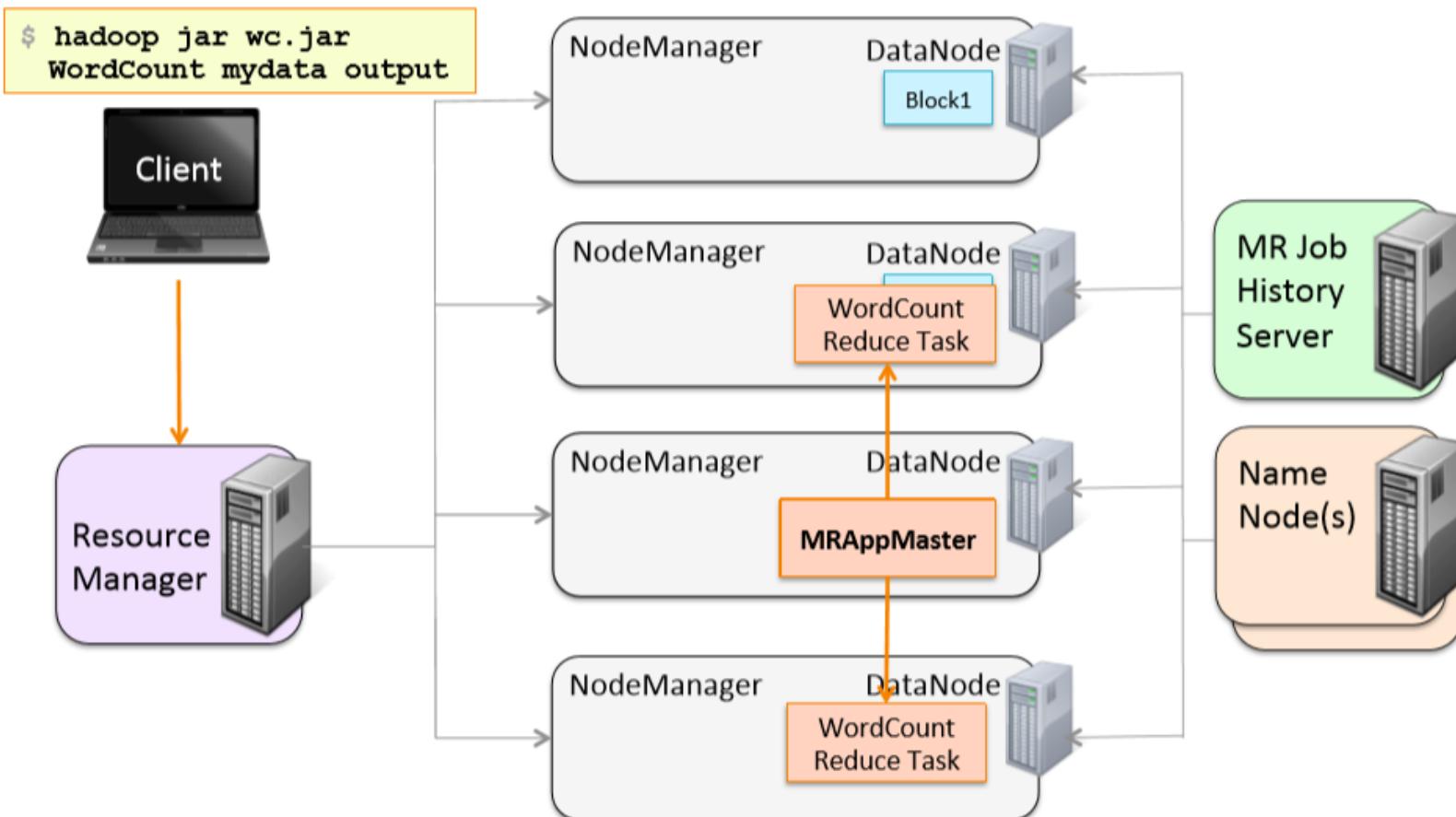
Reduce Tasks



Allocating container for Reduce Task



Executing Reduce Task



What if something Fails?

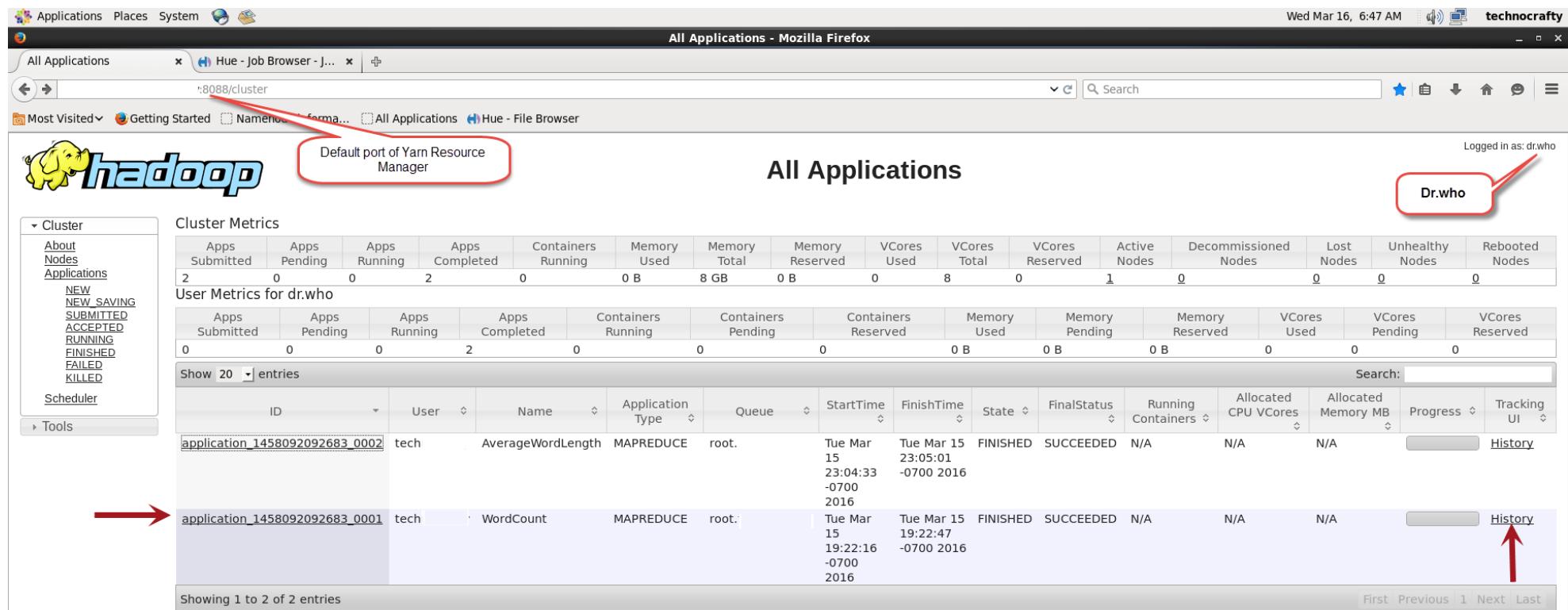
- The ApplicationMaster will reattempt tasks that complete with exceptions or stop responding (4 times by default)
- Application with too many failed tasks are considered failed.
- If ApplicationMaster stops sending heartbeats, the ResourceManager will reattempt the whole application (2 times by default)
- If ApplicationMaster Job Recovery optional setting is set to false, all the task will re-run. If set to true, only incomplete tasks will rerun.

Cont..

- If NodeManager stops sending heartbeats to ResourceManager, it is removed from list of active nodes.
- If the ApplicationMaster node fails, it will be treated as a failed application.
- No application or tasks can be launched if ResourceManager is unavailable.

Exploring YARN Application Through Web UI

The resource manager can be access at default port 8088



Application Details

The screenshot shows the Hadoop Application Details page. On the left, there's a sidebar with 'Cluster' and 'Tools' sections. The main area displays application details for a 'WordCount' job.

User: techi
Name: WordCount
Application Type: MAPREDUCE
Application Tags:
State: FINISHED
FinalStatus: SUCCEEDED
Started: Tue Mar 15 19:22:16 -0700 2016
Elapsed: 31sec
Tracking URL: [History](#) ←
Diagnostics:

Total Resource Preempted: <memory:0, vCores:0>
Total Number of Non-AM Containers Preempted: 0
Total Number of AM Containers Preempted: 0
Resource Preempted from Current Attempt: <memory:0, vCores:0>
Number of Non-AM Containers Preempted from Current Attempt: 0
Aggregate Resource Allocation: 95623 MB-seconds, 56 core-seconds

A red arrow points to the 'History' link under the Tracking URL. A red rounded rectangle highlights the 'Tracking URL' section, with the text 'Select the Node HTTP Address to Open Node Manager UI' positioned next to it.

Container Information

The screenshot shows the Hadoop NodeManager interface. The left sidebar has a tree view with 'ResourceManager' expanded, 'NodeManager' selected, and 'Node Information' expanded. Under 'Node Information', 'List of Applications' is highlighted with a red arrow and a callout bubble. Another red arrow points from the 'List of Containers' link to a callout bubble. The main content area displays various metrics and status information:

Total Vmem allocated for Containers	16.80 GB	NodeManager information
Vmem enforcement enabled	false	
Total Pmem allocated for Container	8 GB	
Pmem enforcement enabled	true	
Total VCores allocated for Containers	8	
NodeHealthyStatus	true	
LastNodeHealthTime	Wed Mar 16 06:54:41 PDT 2016	
NodeHealthReport		
Node Manager Version:	2.6.0-cdh5.5.0 from fd21232cef7b8c1f536965897ce20f50b83ee7b2 by jenkins source checksum db52b8a74b1a7e55c309ec5fbcd7ca on 2015-11-09T20:43Z	
Hadoop Version:	2.6.0-cdh5.5.0 from fd21232cef7b8c1f536965897ce20f50b83ee7b2 by jenkins source checksum 98e07176d1787150a6a9c087627562c on 2015-11-09T20:37Z	

To check the application you have submitted

To check the allocation of containers by Resource Manager on selected node for current running application

JobHistory Information



MapReduce Job job_1458092092683_0001

Application

Job

- [Overview](#)
- [Counters](#)
- [Configuration](#)
- [Map tasks](#)
- [Reduce tasks](#)

Tools

Navigate to each sub tab's below and explore the metrics

Job Overview				
Job Name:	WordCount	User Name:	tech	
Queue:	root.	State:	SUCCEEDED	
Uberized:	false	Submitted:	Tue Mar 15 19:22:16 PDT 2016	
Started:	Tue Mar 15 19:22:26 PDT 2016	Finished:	Tue Mar 15 19:22:47 PDT 2016	
Elapsed:	20sec	Diagnostics:		
Average Map Time	7sec	Average Shuffle Time	6sec	
Average Merge Time	0sec	Average Reduce Time	1sec	

ApplicationMaster

Attempt Number	Start Time	Node	Logs
1	Tue Mar 15 19:22:21 PDT 2016	8042	logs

Task Type	Total	Complete
Map	1	1
Reduce	1	1

Map Task



Logged in as: dr.who

Map Tasks for job_1458092092683_0001

Application		Task						Successful Attempt		
Job		Name	State	Start Time	Finish Time	Elapsed Time	Start Time	Finish Time	Elapsed Time	
		task_1458092092683_0001_m_000000	SUCCEEDED	Tue Mar 15 19:22:28 -0700 2016	Tue Mar 15 19:22:36 -0700 2016	7sec	Tue Mar 15 19:22:28 -0700 2016	Tue Mar 15 19:22:36 -0700 2016	7sec	
Tools		ID	State	Start Time	Finish Time	Elapsed	Start Time	Finish Time	Elapsed	

Showing 1 to 1 of 1 entries

First Previous | 1 Next Last

Reduce Task



Reduce Tasks for job_1458092092683_0001

Task										Successful Attempt			
Name	State	Start Time	Finish Time	Elapsed Time	Start Time	Shuffle Finish Time	Merge Finish Time	Finish Time	Elapsed Time	Shuffle			
_1458092092683_0001_r_000000	SUCCEEDED	Tue Mar 15 19:22:38 -0700 2016	Tue Mar 15 19:22:47 -0700 2016	8sec	Tue Mar 15 19:22:38 -0700 2016	Tue Mar 15 19:22:45 -0700 2016	Tue Mar 15 19:22:45 -0700 2016	Tue Mar 15 19:22:47 -0700 2016	6sec				
		Start Time	Finish Time	Elapsed	Start Time	Shuffle Time	Merge Time	Finish Time	Elapsed				
Showing 1 to 1 of 1 entries										F			

Number of part file generated =1

YARN Application Logs

The screenshot shows the YARN Application Overview page. On the left, there's a sidebar with 'Application' and 'Job' sections containing links like 'Overview', 'Counters', 'Configuration', 'Map tasks', 'Reduce tasks', and 'Tools'. A red arrow points from the 'Overview' link in the Job section to a callout bubble that says 'Navigate to each sub tab's below and explore the metrics'. The main area is titled 'Job Overview' and contains details for a 'WordCount' job submitted by 'tech' from 'root' queue. It shows the job state as 'SUCCEEDED', submission time as 'Tue Mar 15 19:22:16 PDT 2016', and completion time as 'Tue Mar 15 19:22:47 PDT 2016'. The elapsed time was '20sec'. Below this, under 'Diagnostics', are metrics for map, shuffle, merge, and reduce times. At the bottom, there's a table for the 'ApplicationMaster' with columns for Attempt Number, Start Time, Node, and Logs. The first attempt started at 'Tue Mar 15 19:22:21 PDT 2016' on node '8042' with logs at 'logs'. A red arrow points to the 'Logs' column. Another red arrow points to the 'Attempt Number' column of the ApplicationMaster table.

Job Name: WordCount
User Name: tech
Queue: root.
State: SUCCEEDED
Uberized: false
Submitted: Tue Mar 15 19:22:16 PDT 2016
Started: Tue Mar 15 19:22:26 PDT 2016
Finished: Tue Mar 15 19:22:47 PDT 2016
Elapsed: 20sec
Diagnostics:
Average Map Time 7sec
Average Shuffle Time 6sec
Average Merge Time 0sec
Average Reduce Time 1sec

ApplicationMaster				
Attempt Number	Start Time	Node	Logs	
1	Tue Mar 15 19:22:21 PDT 2016	8042	logs	

Task Type	Total	Complete
Map	1	1
Reduce	1	1

Quick Overview

- Map-Reduce deals with one record (key-value pair) at a time.
- YARN framework is for job scheduling and resource management.
- MapReduce version 1 framework is for data processing.
- MapReduce job has three phase.
- Spark in-memory computing system.



Configuring HDFS High Availability

- An HDFS HA cluster is configured with two NameNodes - an Active NameNode and a Standby NameNode.
- Only one NameNode can be active at any point in time.
- HDFS High Availability depends on maintaining a log of all namespace modifications in a location available to both NameNodes, so that in the event of a failure the Standby NameNode has up-to-date information about the edits and location of blocks in the cluster.

Cont..

There are two implementations available for maintaining the copies of the edit logs:

- High Availability using Quorum-based Storage
- High Availability using an NFS-mounted shared edits directory

Enabling High Availability with Quorum-based Storage

- From the **Services** tab, select your HDFS service.
- Click the **Instances** tab.
- Click **Enable High Availability**
- The next screen shows the hosts that are eligible to run a Standby NameNode and the JournalNodes.
 - Select **Enable High Availability with Quorum-based Storage** as the High Availability Type.
 - Select the host where you want the Standby NameNode to be set up.
- Enter a directory location for the JournalNode edits directory into the fields for each JournalNode host.
- Click **Continue**

Enabling High Availability using NFS Shared Edits Directory

- From the **Services** tab, select your HDFS service.
- Click the **Instances** tab.
- Click **Enable High Availability**
- The next screen shows the hosts that are eligible to run a Standby NameNode.

Cont..

- Select **Enable High Availability with NFS shared edits directory** as the High Availability Type.
- Select the host where you want the Standby NameNode to be installed, and click **Continue**.
- Confirm or enter the directories to be used as the name directories for the NameNode.
- Enter the absolute path of the local directory, on each NameNode host, that is mounted to the remote shared edits directory.
- Click **Continue** to proceed

Enabling Automatic Failover

To enable Automatic Failover:

- From the **Services** tab, select your HDFS service.
- Click the **Instances** tab.
- Click **Enable Automatic Failover...**
- Confirm that you want to take this action. This will stop the NameNodes for the Nameservice, create and configure Failover Controllers for each NameNode, initialize the High Availability state in ZooKeeper, and start the NameNodes and Failover Controllers.

Manually Failing Over Your Cluster

To perform a manual failover:

- From the **Services** tab, select your HDFS service.
- Click the **Instances** tab.
- Click **Manual Failover...**
- From the pop-up, select the NameNode that should be made active, then click **Manual Failover**.
- When all the steps have been completed, click **Finish**.

Running the Balancer

- The Balancer usually shows a status of **N/A** on the **HDFS > Status** tab because the Balancer runs only temporarily.

To run the Balancer:

- On the **HDFS > Status** tab, choose **Rebalance** on the **Actions** menu.
- Click **Rebalance** that appears in the next screen to confirm. If you see a **Finished** status, the Balancer successfully ran and completed.

The Balancer role is normally added (by default) when the HDFS service is installed. If it has not been added, you must add a Balancer role instance in order to rebalance HDFS and to see the **Rebalance** action.

Hadoop Daemon Logs

- Exist on all machines running at least one Hadoop Daemon.
- Each daemon has .log and .out file
- The .out files are only written when daemons are started.
- Once daemon started successfully, the .out files are truncated.
- All log messages can be found in .log file, including daemon startup message that are sent to .out

Cont..

The .log and .out file names are constructed as follows:

hadoop-<user-running-Hadoop>-<daemon>-<hostname>.log

where

<user-running-Hadoop> is the user running Hadoop daemon i.e. ‘hadoop’

<daemon> name of the daemon these logs are associated i.e. ‘namenode’ ,
‘datanode’ etc

<hostname> machine on which daemon are running

Cont..

- By default, the .log files are rotated daily by log4j.
- Configurable with /etc/hadoop/conf/log4j.properties
- Administrator should review these logs for cluster specific error and warning to identify any daemon running incorrectly.

Note

The Namenode and Secondary Namenode logs should not be deleted more frequently than `fs.checkpoint.period`.

In the event of a Secondary Namenode edits log compaction failure, logs from the Namenode and Secondary Namenode should be available for diagnostics.

Configuring the YARN Service

To add the YARN service and configure it to have a higher priority than MRv1, do the following:

- Click the **Services** tab, then choose **All Services**.
- From the **Actions** menu, select **Add a Service**. A list of possible services are displayed. You can add one type of service at a time.
- Follow the instructions in the Add Service wizard to add the service.
- Stop the YARN service.
- Go to the YARN service page (by selecting the YARN service from the **Services** menu or from the **All Services** page).

Cont..

- Go to the **Configuration** tab, and search for "Alternatives Priority".
- Change the priority value to be higher than MRv1.
- Save your configuration change.
- Start the YARN service.
- Redeploy the client configuration to have the changes take effect; (from the YARN Status page, **Actions** menu, click **Deploy Client Configuration**).

Cont..

When adding the YARN service, the **Add Service** wizard automatically creates a job history directory for HBase.

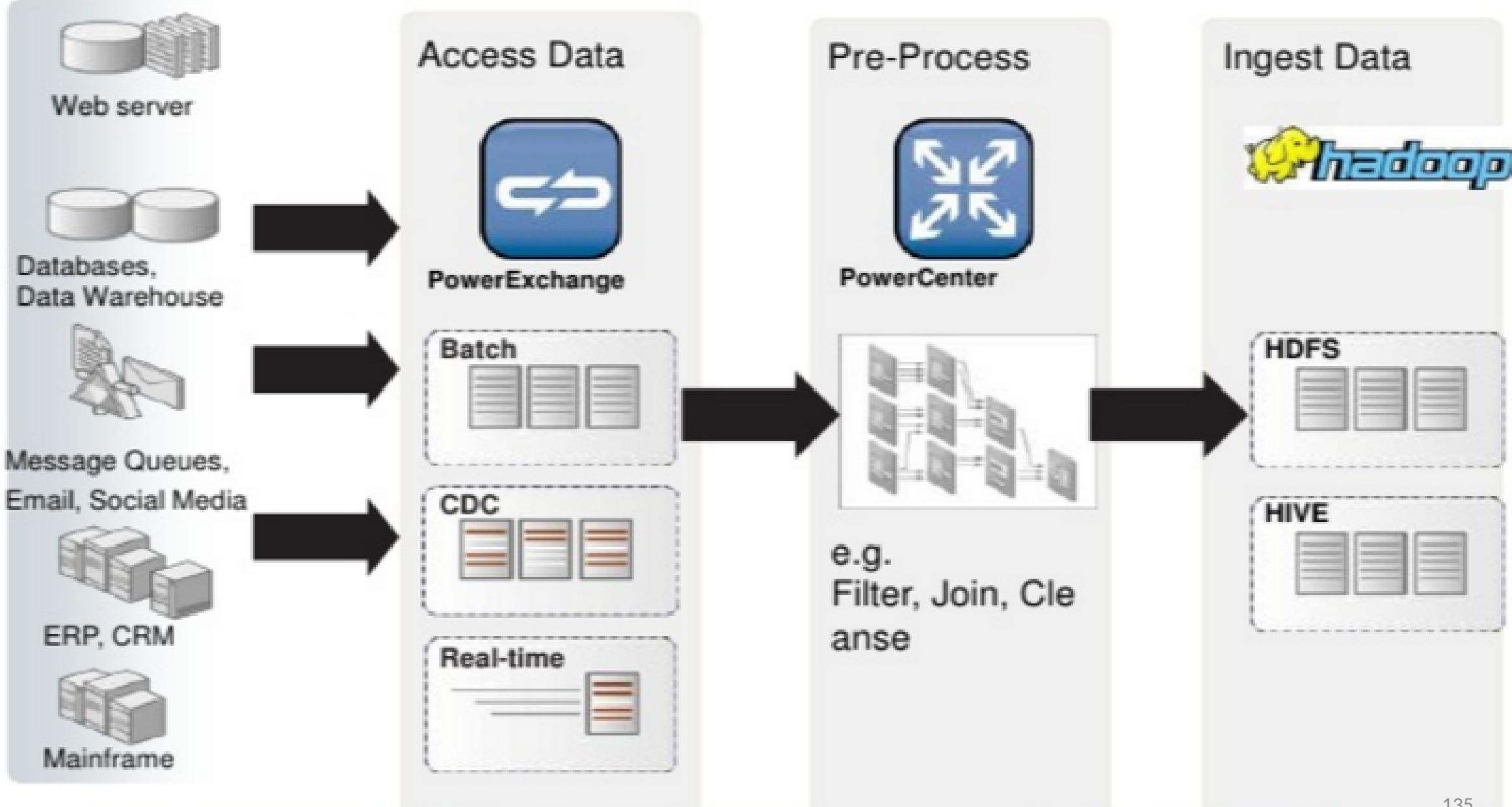
If you quit the **Add Service** wizard or it does not finish, you can create the directory outside the wizard by doing these steps:

- Choose **Create Job History Dir** from the **Actions** menu in the **YARN > Status** tab.
- Click **Create Job History Dir** again to confirm.

Getting Data Into HDFS

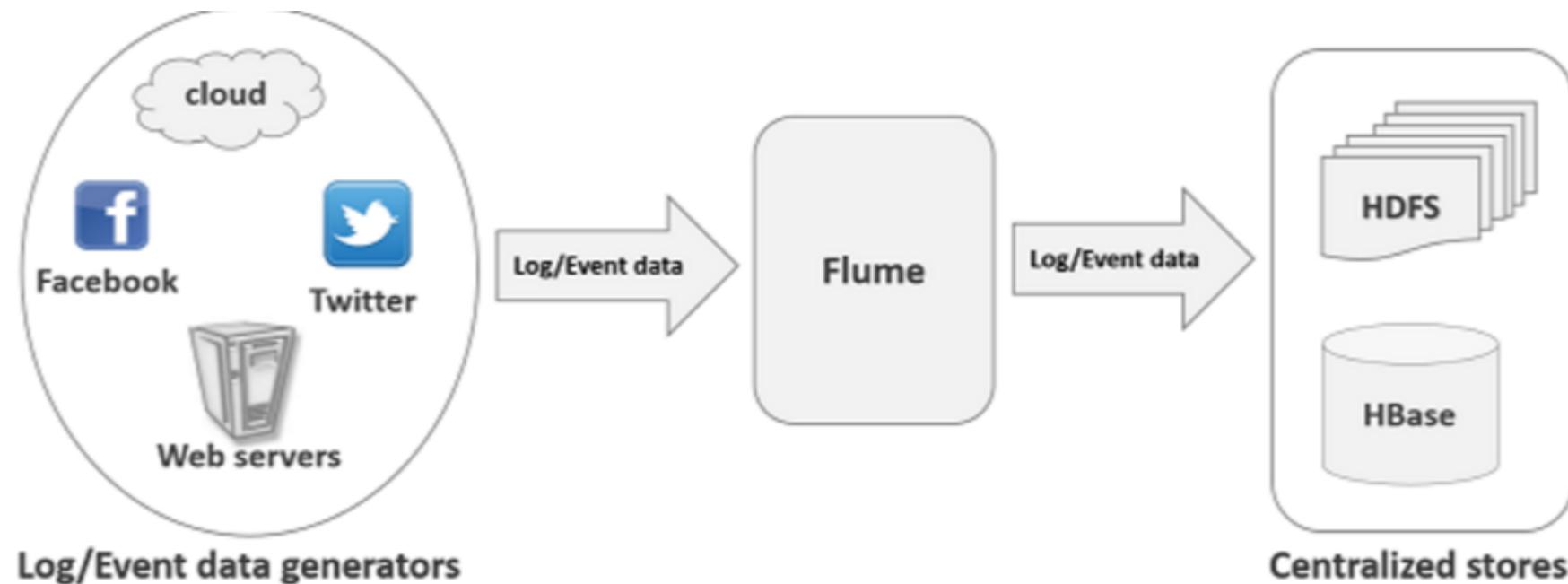
We will cover following topics:

- Ingesting Data From External Sources With Flume
- Ingesting Data From Relational Databases With Sqoop
- REST Interfaces
- Best Practices for Importing Data



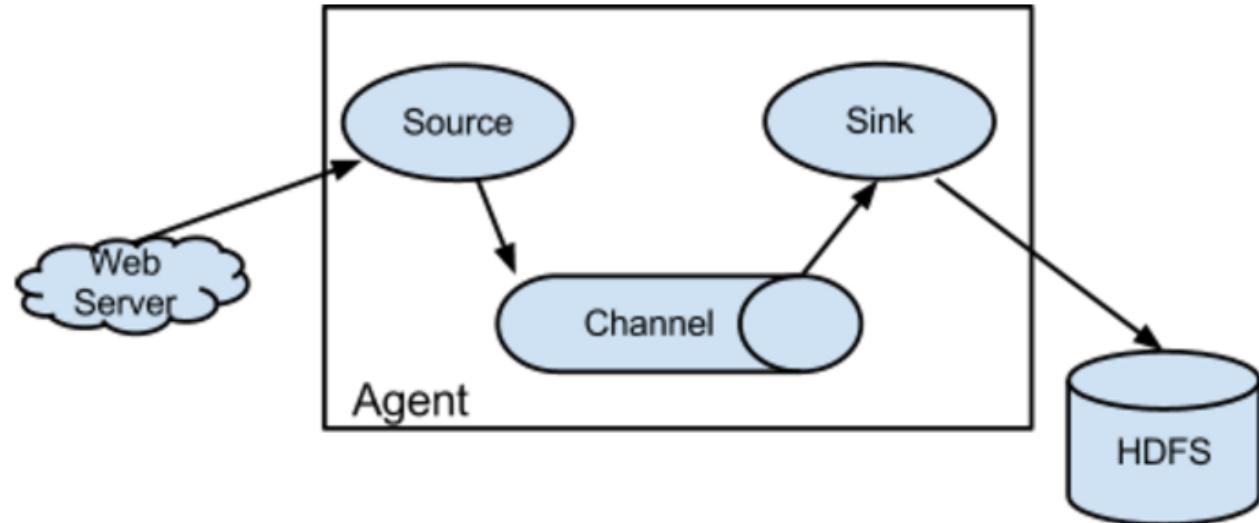
Ingesting Data From External Sources

With Flume



Flume Basics

- Flume is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of log data.
- Simple and flexible architecture based on streaming data flows.
- Robust and fault tolerant.



Features of Flume

- Flume ingests log data from multiple web servers into a centralized store (HDFS, HBase) efficiently.
- Using Flume, we can get the data from multiple servers immediately into Hadoop.
- Along with the log files, Flume is also used to import huge volumes of event data produced by social networking sites like Facebook and Twitter, and e-commerce websites like Amazon and Flipkart.
- Flume supports a large set of sources and destinations types.
- Flume supports multi-hop flows, fan-in fan-out flows, contextual routing, etc.
- Flume can be scaled horizontally

How it works ?

- **Data generators** (such as Facebook, Twitter) generate data which gets collected by individual Flume **agents**.
- **Data collector** (which is also an agent) collects the data from the agents which is aggregated and pushed into a centralized store such as HDFS or HBase.



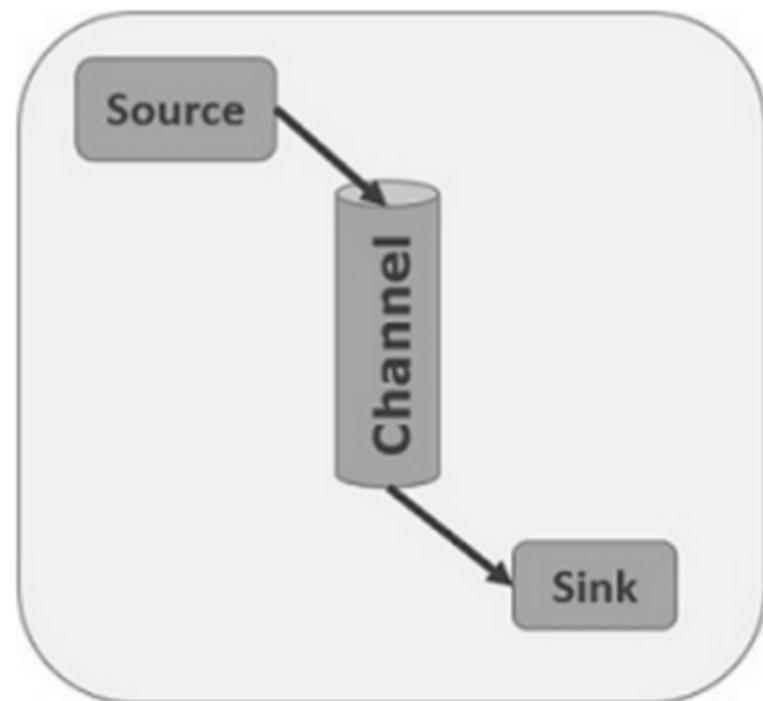
Flume Event

- An **event** is the basic unit of the data transported inside **Flume**.
- It contains a payload of byte array that is to be transported from the source to the destination accompanied by optional headers.



Flume Agent

- An **agent** is an independent daemon process (JVM) in Flume.
- It receives the data (events) from clients or other agents and forwards it to its next destination (sink or agent).
- Flume may have more than one agent



Flume Agent

Main Components of Flume Agent

- **Source:** It receives data from the data generators and transfers it to one or more channels in the form of Flume events.

Example – Avro source, Thrift source, twitter 1% source etc.

- **Channel:** It is a transient store which receives the events from the source and buffers them till they are consumed by sinks. It acts as a bridge between the sources and the sinks.

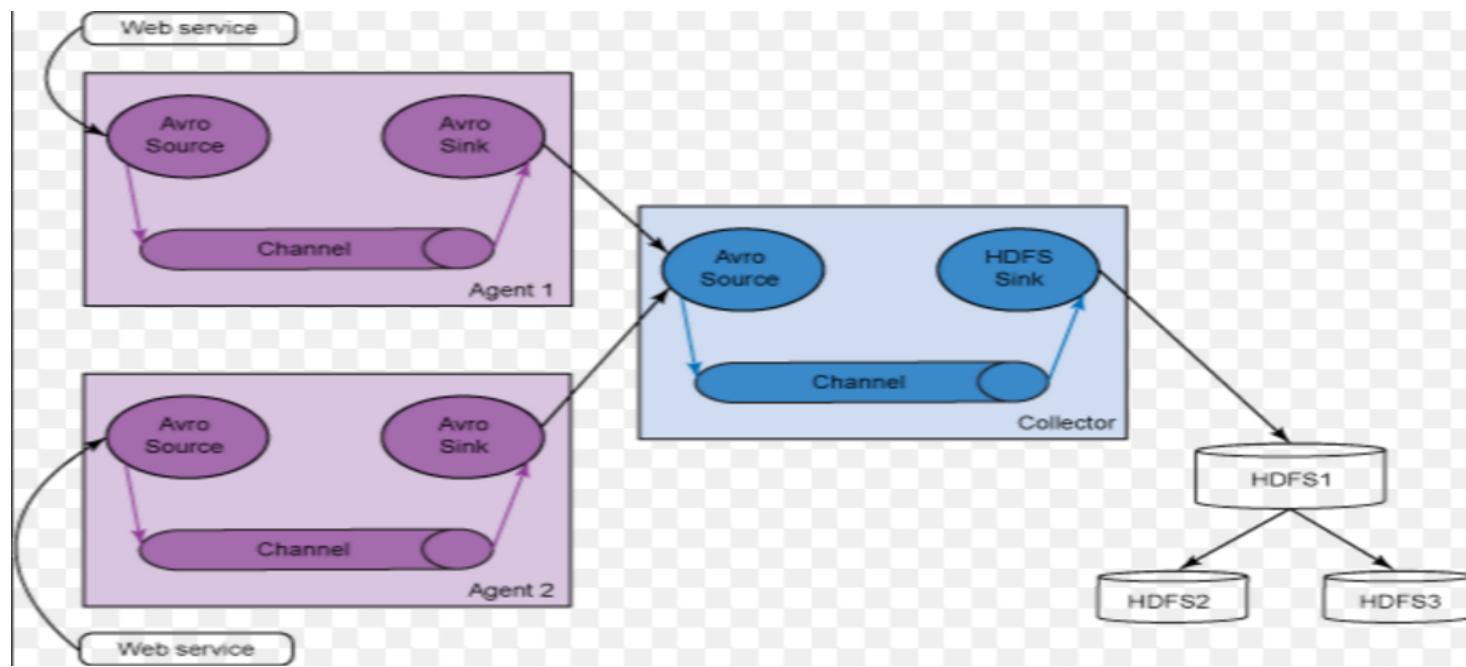
Example – JDBC channel, File system channel, Memory channel, etc.

- **Sink:** It stores the data into centralized stores like HBase and HDFS. It consumes the data (events) from the channels and delivers it to the destination.

Example – HDFS sink

Flume Data Flow

Multi-hop Flow: Within Flume, there can be multiple agents and before reaching the final destination, an event may travel through more than one agent.

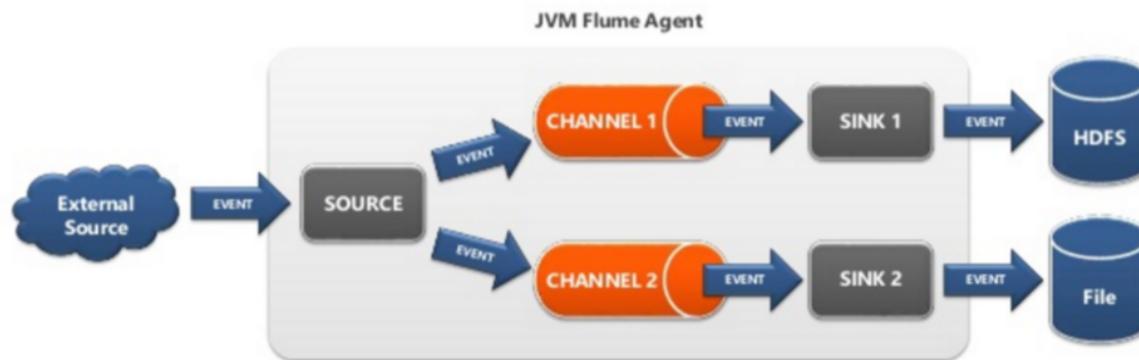


Cont..

Fan-out Flow: Dataflow from one source to multiple channels.

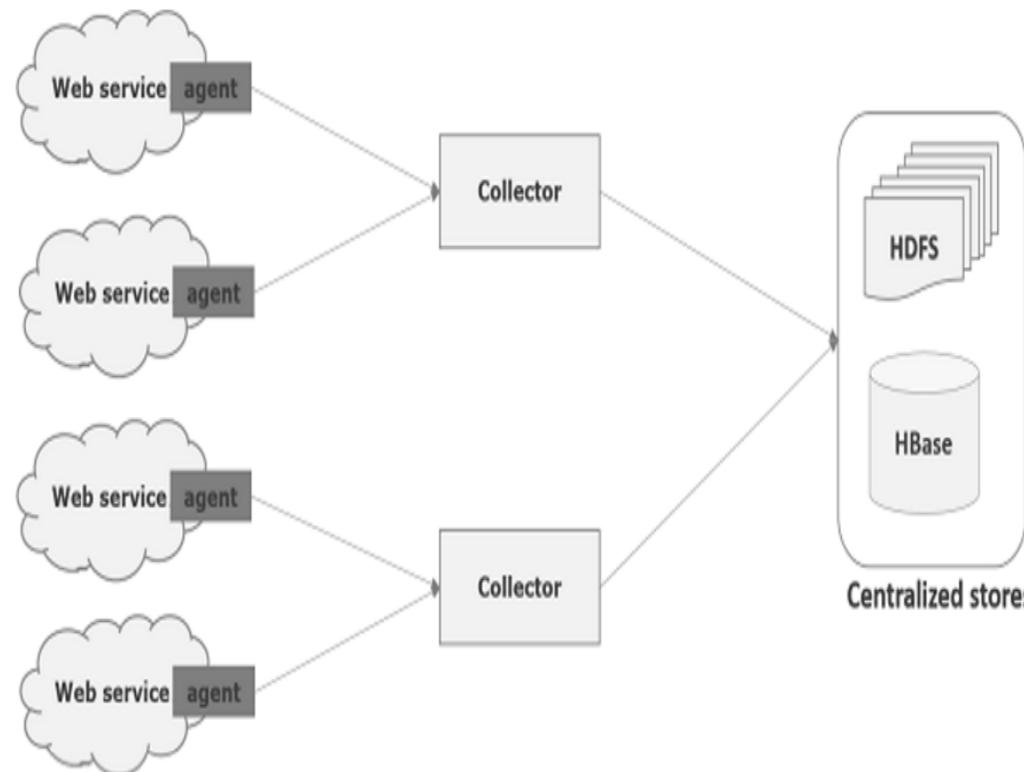
It is of two type-

1. **Replicating** – The data flow where the data will be replicated in all the configured channels.
2. **Multiplexing** – The data flow where the data will be sent to a selected channel which is mentioned in the header of the event.



Cont..

Fan-in Flow: The data flow in which the data will be transferred from many sources to one channel is known as **fan-in flow**.



Failure Handling

- For each event, two transactions take place: one at the sender and one at the receiver.
- The sender sends events to the receiver. Soon after receiving the data, the receiver commits its own transaction and sends a “received” signal to the sender. After receiving the signal, the sender commits its transaction.
- Sender will not commit its transaction till it receives a signal from the receiver.

Installing, Configuring and Starting Flume

- To install,run
 - sudo yum install flume-ng
 - sudo yum install flume-ng-agent
- Configure Agent Nodes on the machine generating the data
 - The /etc/flume-ng/conf/flume-conf.properties file define source, sink ,channels and flow within an agent
 - Start by copying the template file /etc/flume-ng/conf/flume-conf.properties.template

Cont..

- To start Flume
 - Run `sudo service flume-ng-agent start` to start a single Flume agent on a host.
 - Run `flume-ng - -name <agent_name>` from the command line if you need to start multiple flume agent on the same host.

Flume - Configuration

In the Flume configuration file, we need to –

- Name the components of the current agent.
- Describe/Configure the source.
- Describe/Configure the sink.
- Describe/Configure the channel.
- Bind the source and the sink to the channel.

Cont..

- Naming the component

```
agent_name.sources = source_name  
agent_name.sinks = sink_name  
agent_name.channels = channel_name
```

- To List few of them

Sources	Channels	Sinks
Avro Source	Memory Channel	HDFS Sink
Thrift Source	JDBC Channel	Hive Sink
Exec Source	Kafka Channel	Logger Sink
JMS Source	File Channel	Avro Sink
Spooling Directory Source	Spillable Memory Channel	Thrift Sink

Cont..

- Describing the source: Each source will have a separate list of properties.

```
agent_name.sources. source_name.type = value  
agent_name.sources. source_name.property2 = value  
agent_name.sources. source_name.property3 = value
```

- Describing the sink: Each sink will also have a separate list of properties.

```
agent_name.sinks. sink_name.type = value  
agent_name.sinks. sink_name.property2 = value  
agent_name.sinks. sink_name.property3 = value
```

Cont..

- Describing the channel: Flume provides various channels to transfer data between sources and sinks.

```
agent_name.channels.channel_name.type = value  
agent_name.channels.channel_name.property2 = value  
agent_name.channels.channel_name.property3 = value
```

- Binding the Source and Sink to the Channel

```
agent_name.sources.source_name.channels = channel_name  
agent_name.sinks.sink_name.channels = channel_name
```

Ingesting Data from Relational Databases

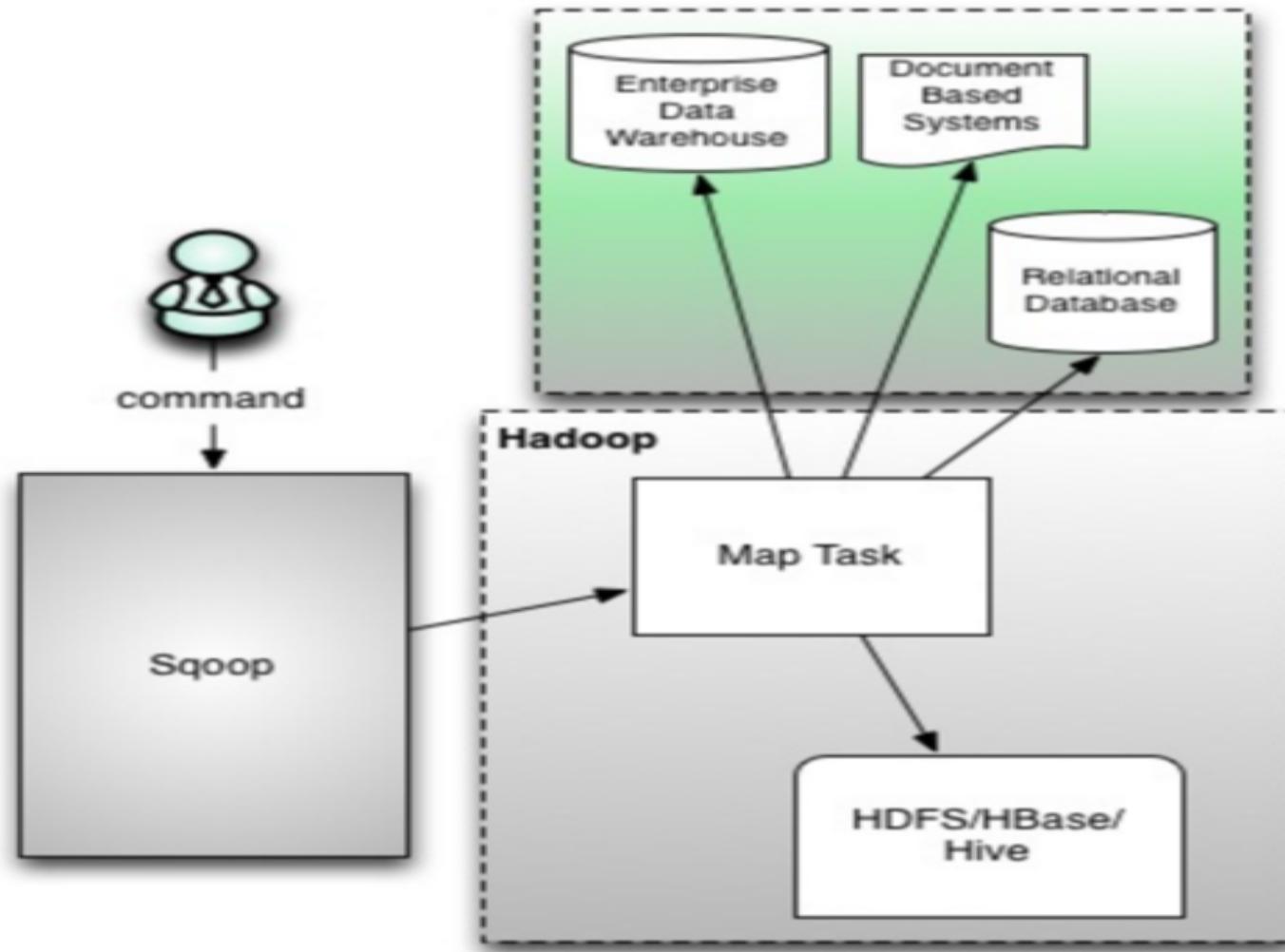
With Sqoop



Sqoop

- Sqoop is a tool designed to transfer data between Hadoop and relational database servers.
- It is used to import data from relational databases such as MySQL, Oracle to Hadoop HDFS, and export from Hadoop file system to relational databases

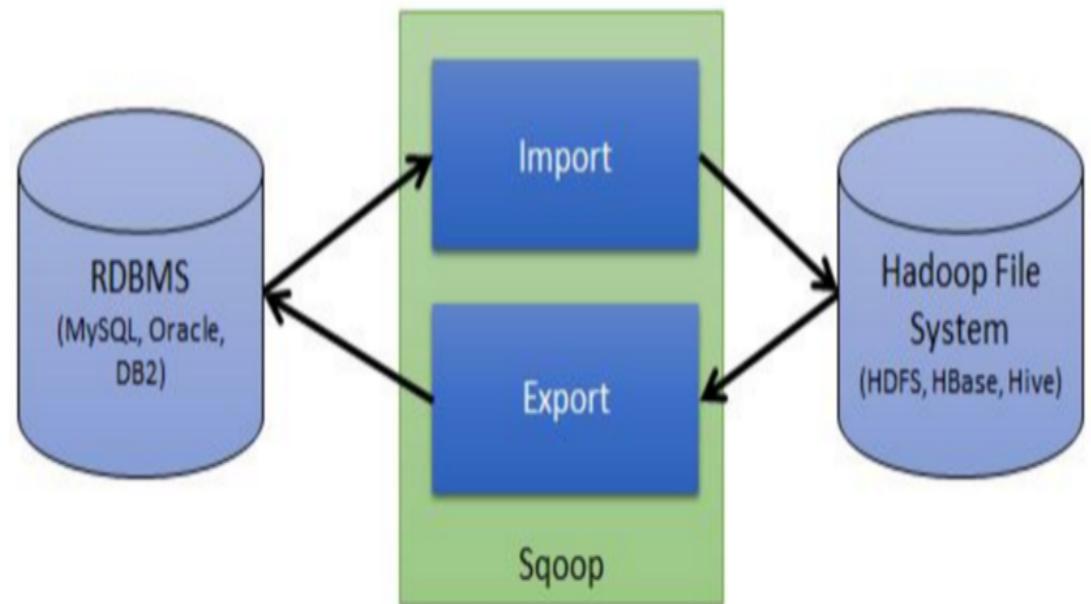
Sqoop Architecture



How Sqoop Works?

Sqoop Import

- The import tool imports individual tables from RDBMS to HDFS.
- Each row in a table is treated as a record in HDFS.
- All records are stored as text data in text files or as binary data in Avro and Sequence files.



Cont..

Sqoop Export

- The export tool exports a set of files from HDFS back to an RDBMS.
- The files given as input to Sqoop contain records, which are called as rows in table.
- Those are read and parsed into a set of records and delimited with user-specified delimiter.

Features of Sqoop

Sqoop allow to :

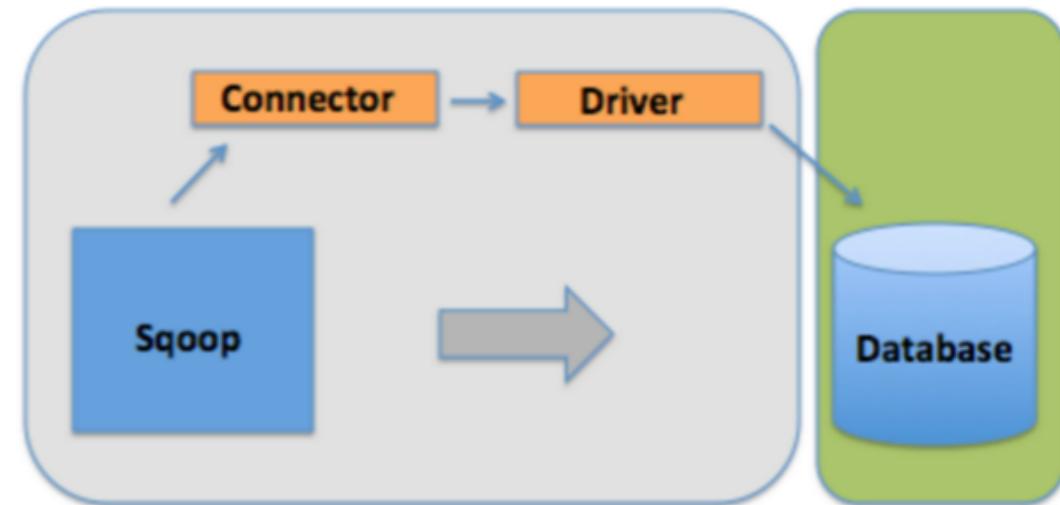
- Import one table
- Import complete database
- Import selected tables
- Import selected columns from a particular table
- Filter out certain rows from certain table etc
- It can create Hive table based on the imported data
- Export data from HDFS to a database table

Cont..

- Sqoop uses Map reduce to fetch data from RDBMS and stores that on HDFS.
- By default it uses four mappers but this value is configurable.

Sqoop Connectors

Types of Connector: Common (JDBC) and Direct (vendor specific)



Common Connectors

MySQL

PostgreSQL

Oracle

SQL Server

DB2

Generic

Direct Connectors

MySQL

PostgreSQL

Oracle

Teradata

And others

Commands

- To list available commands

```
sqoop help
```

Available commands:

codegen	Generate code to interact with database records
create-hive-table	Import a table definition into Hive
eval	Evaluate a SQL statement and display the results
export	Export an HDFS directory to a database table
help	List available commands
import	Import a table from a database to HDFS
import-all-tables	Import tables from a database to HDFS
import-mainframe	Import datasets from a mainframe server to HDFS
job	Work with saved jobs
list-databases	List available databases on a server
list-tables	List available tables in a database
merge	Merge results of incremental imports
metastore	Run a standalone Sqoop metastore
version	Display version information

Cont..

- To list the database

```
sqoop list-databases --username fred -P --connect jdbc:mysql://dbserver.example.com/
```

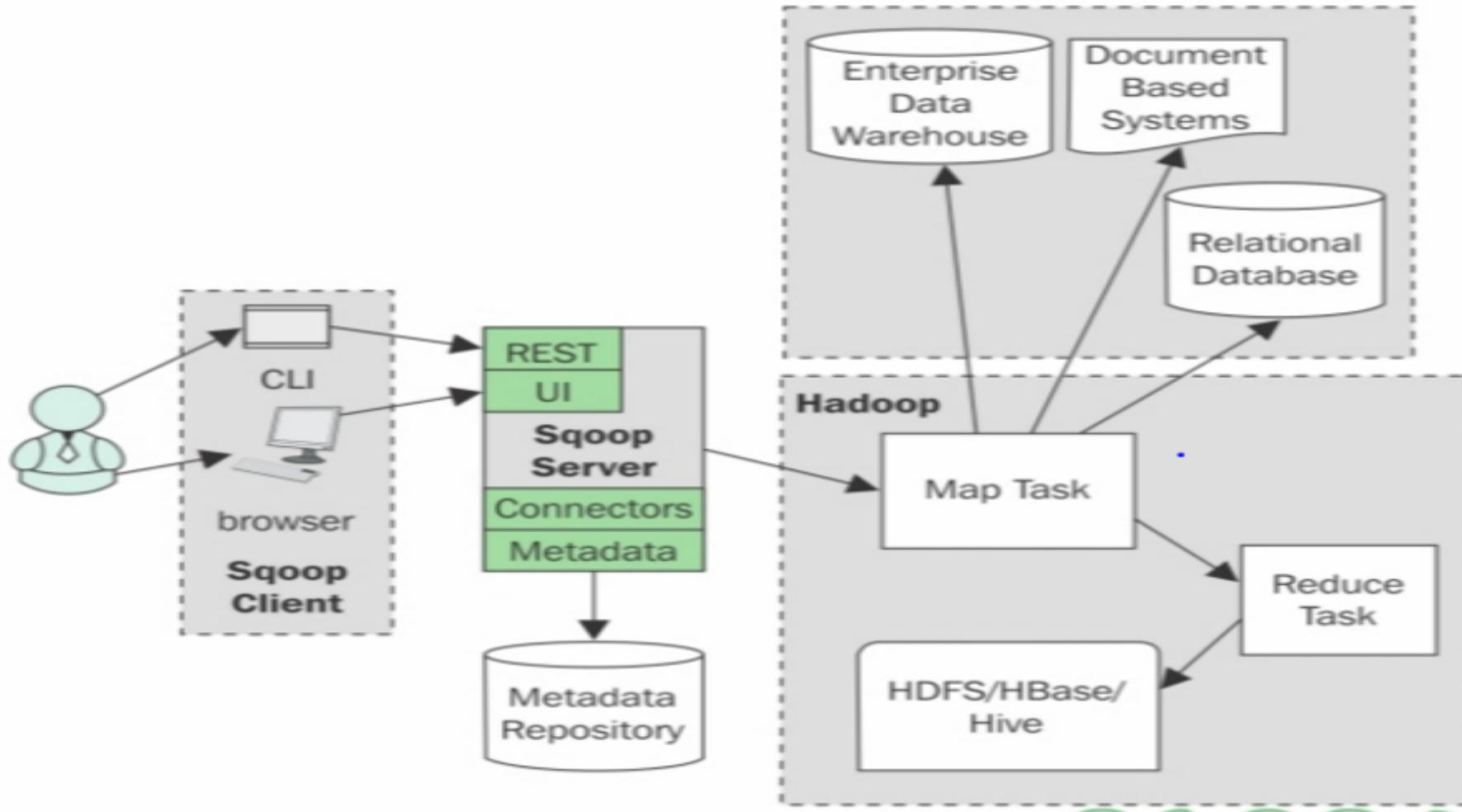
- To list all table in ‘student’ database

```
sqoop list-tables --username fred -P --connect jdbc:mysql://dbserver.example.com/student
```

- To import all table from ‘student’ database

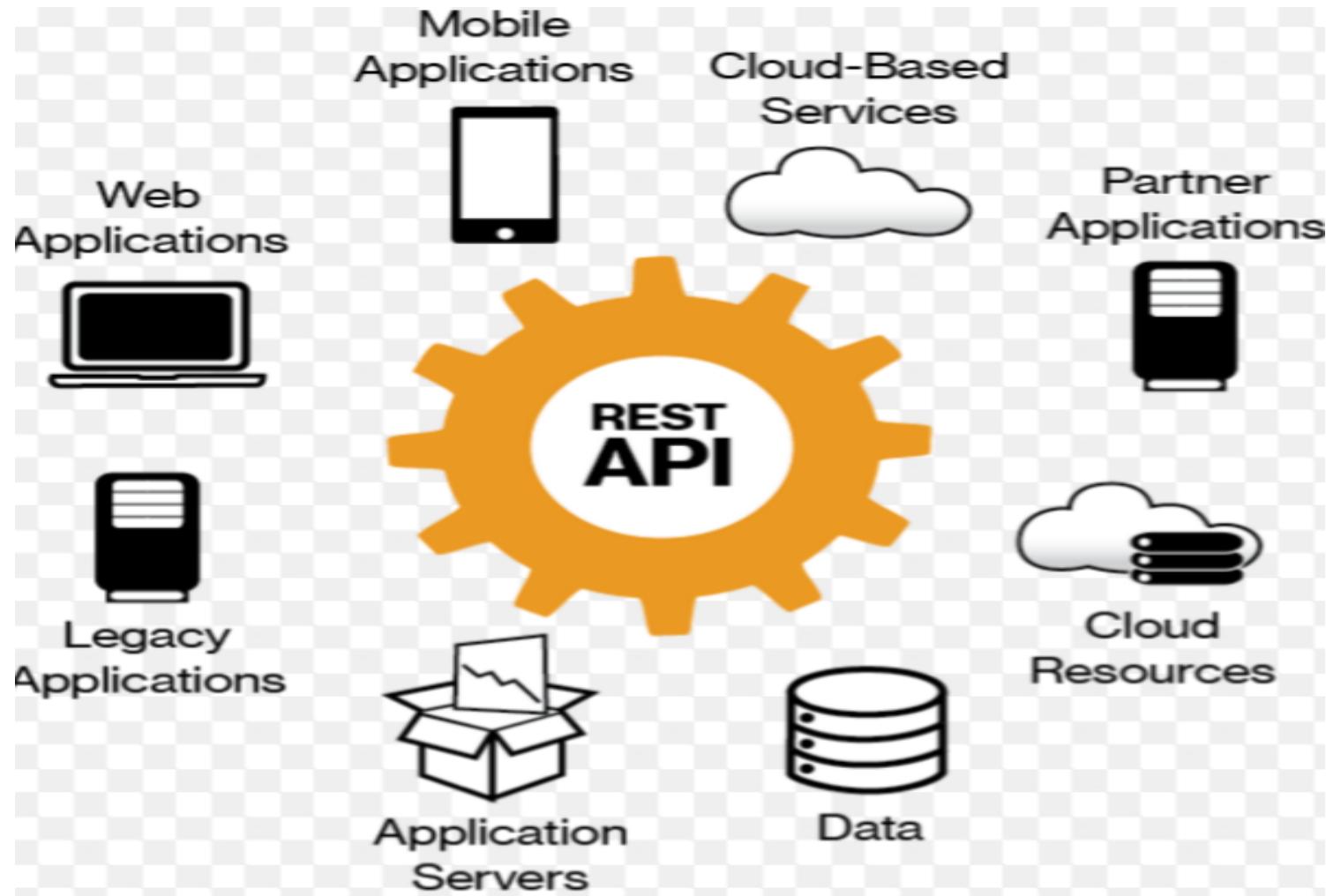
```
sqoop import-all-tables --username fred --password derf --connect jdbc:mysql://dbserver.example.com/student
```

Sqoop2 – Sqoop as a Service



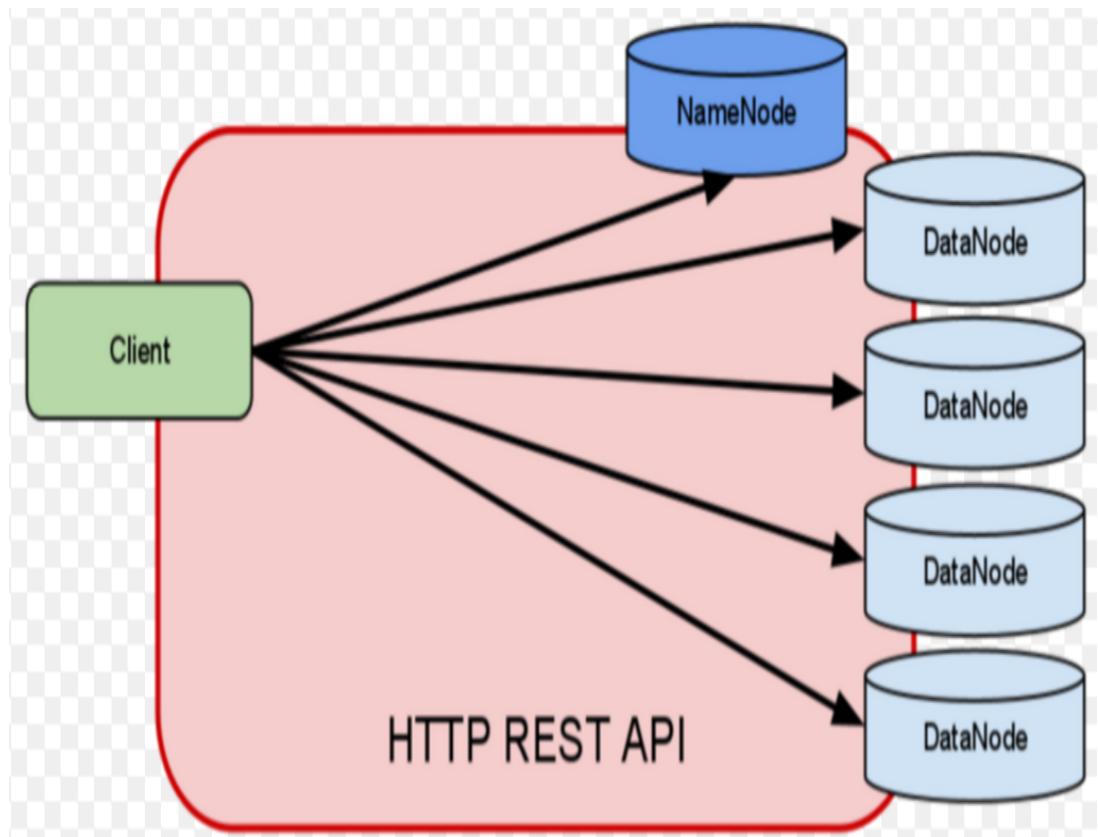
Feature	Sqoop 1	Sqoop 2
Connectors for all major RDBMS	Supported.	<p>Not supported.</p> <p>Workaround: Use the generic JDBC Connector which has been tested on the following databases: Microsoft SQL Server, PostgreSQL, MySQL and Oracle.</p> <p>This connector should work on any other JDBC compliant database. However, performance might not be comparable to that of specialized connectors in Sqoop.</p>
Kerberos Security Integration	Supported.	Supported.
Data transfer from RDBMS to Hive or HBase	Supported.	<p>Not supported.</p> <p>Workaround: Follow this two-step approach.</p> <ol style="list-style-type: none"> 1. Import data from RDBMS into HDFS 2. Load data into Hive or HBase manually using appropriate tools and commands such as the <code>LOAD DATA</code> statement in Hive
Data transfer from Hive or HBase to RDBMS	<p>Not supported.</p> <p>Workaround: Follow this two-step approach.</p> <ol style="list-style-type: none"> 1. Extract data from Hive or HBase into HDFS (either as a text or Avro file) 2. Use Sqoop to export output of previous step to RDBMS 	<p>Not supported.</p> <p>Follow the same workaround as for Sqoop 1.</p>

REST Interfaces



WebHDFS

- Provides HTTP/HTTPS REST interface to HDFS
- Supports both read and writes from/to HDFS
- Can be accessed from within a program or script.
- Can be used via command-line tool such as curl or wget.

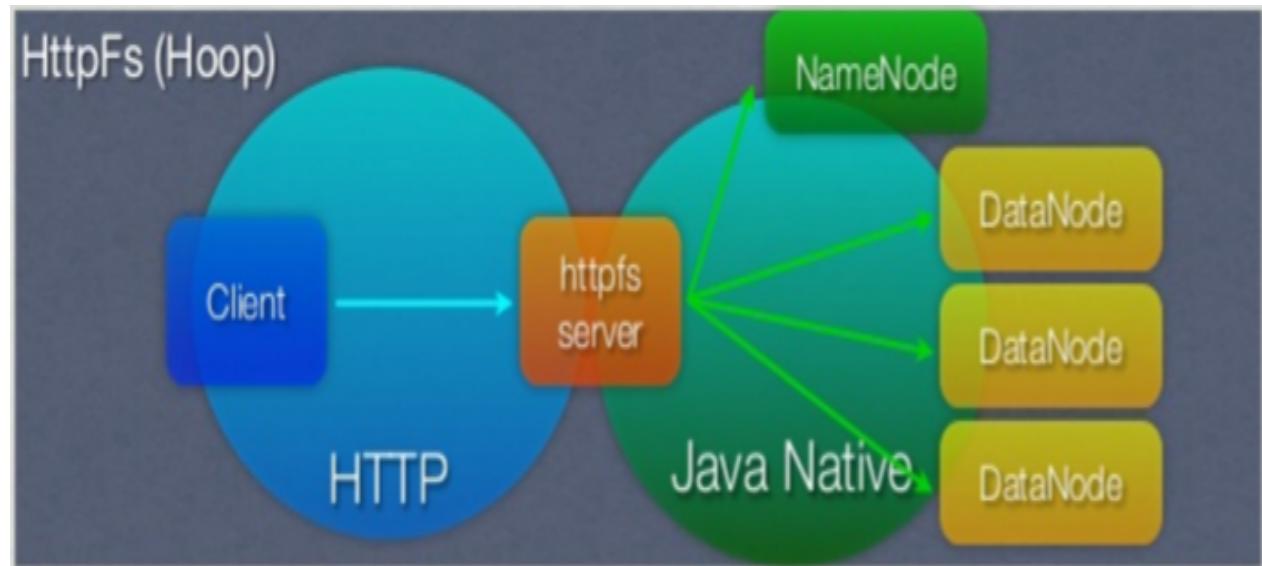


Cont..

- Simple to deploy by just making changes to Hadoop configuration
- Requires client access to every datanode in the cluster
- Does not support HDFS HA deployment.

HttpFS

- Provides an HTTP/HHTTPS REST interface to HDFS.
- More complex than WebHDFS to deploy.
- Required client access to HttpFS server only.
- Supports HDFS HA deployment.



Best Practices for Importing Data

- Import data into temporary directory
- After file is completely written, move it to target directory.
- Create sqoop jobs to perform the import.
- Make use of incremental imports for large datasets.

Quick Overview

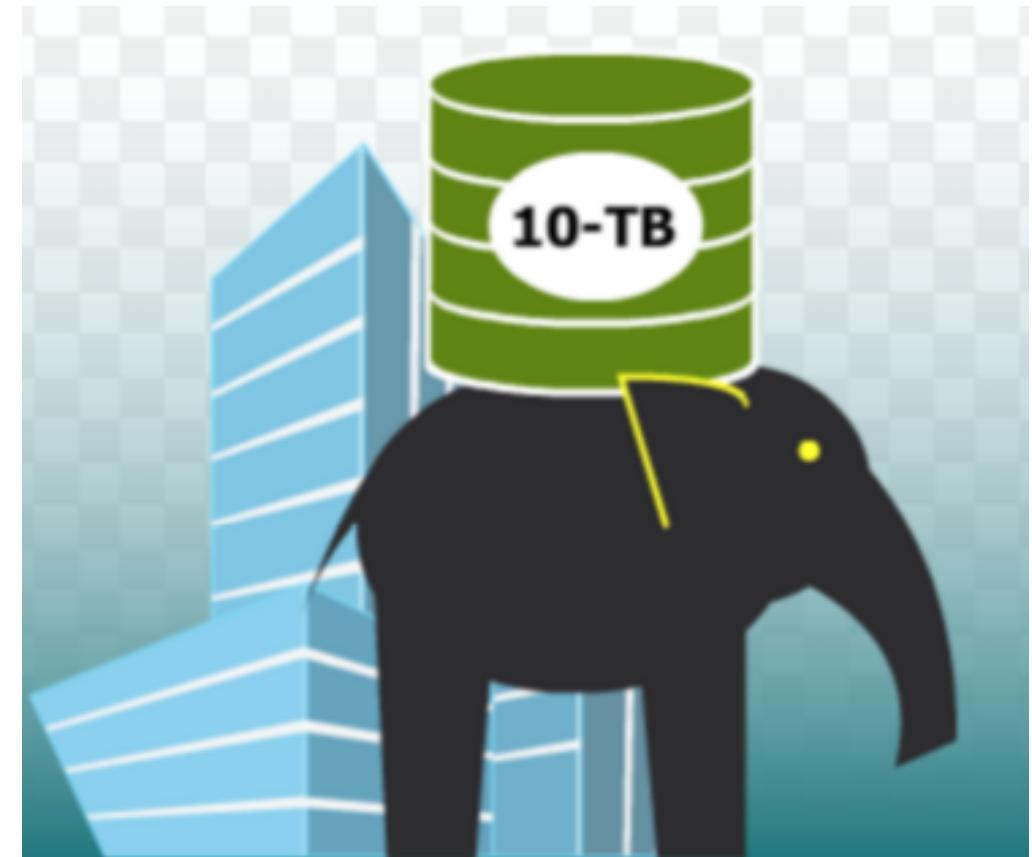
- Flume uses the terms source, sink and channel to describe its actors.
- Using Sqoop, you can import data from a relational database into HDFS.
- REST interface is available for accessing HDFS.



Planning Your Hadoop Cluster

We will cover following topics:

- General Planning Considerations
- Choosing the Right Hardware
- Virtualization Options
- Network Considerations
- Configuring Nodes

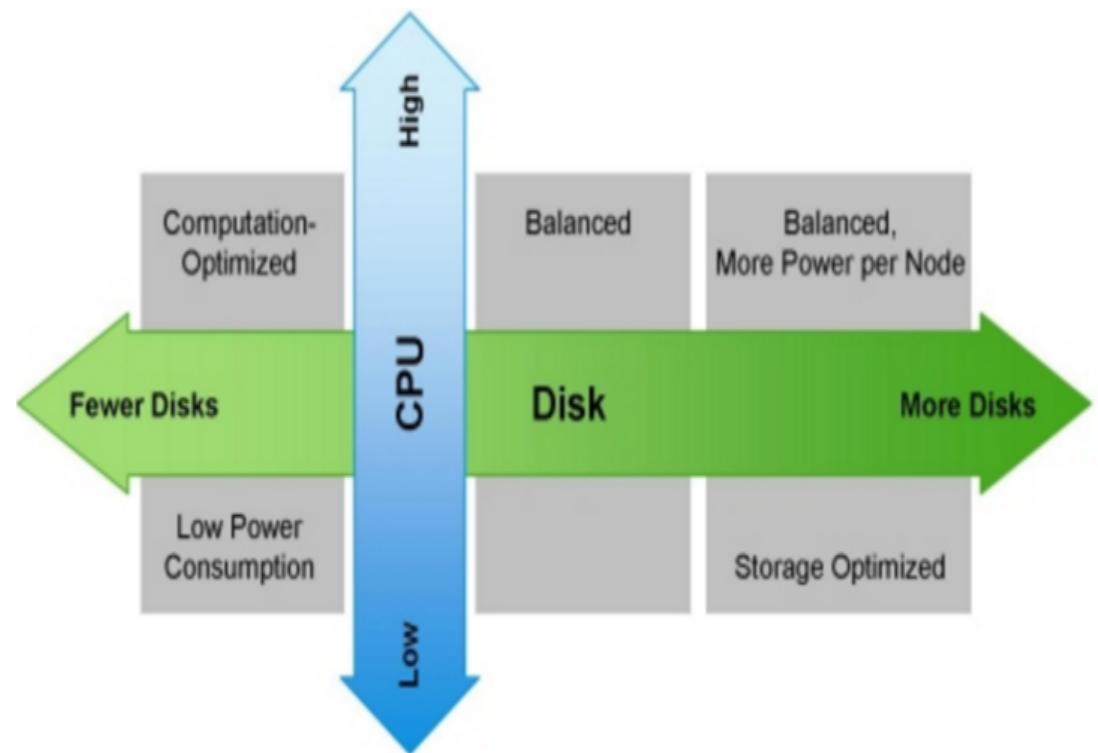


Essentials to Keep in Mind!!

Select the most suitable configuration depending upon need.

Key things:

1. Proper storage and computing hardware.
2. Designing the interconnected network
3. Installing and configuring the operating system



Why workload matters?

MapReduce job will either encounter a bottleneck reading data from disk or from the network (known as an IO-bound job) or in processing data (CPU-bound).

IO-bound job:

- Indexing
- Grouping
- Data importing and exporting
- Data movement and transformation

Cont..

CPU-bound workloads:

- Clustering/Classification
- Complex text mining
- Natural-language processing
- Feature extraction

Planning Considerations

- In a Hadoop cluster, different types of nodes may require different hardware configurations.
- Hadoop cluster sizes may have different configuration requirements.
- Nodes in a Hadoop cluster are interconnected through network devices such as switches and routers.
- Data will be transferred from one node to another over the network during different phases of a MapReduce job.

Cont..

- Network segmentation caused by device failures can greatly degrade the cluster performance.
- Highly available and resilient network architecture is crucial for a Hadoop cluster.
- Operating system configuration is needed such as configuring users, groups, and system security, such as firewalls and SELinux.
- Requires Hadoop dependency software, Java, and some optional tools that can improve cluster management efficiency

Storage Capacity

- Basing your cluster growth on storage capacity is often a good method.
- Estimate the size of the cluster depending on amount of data to be processed.

Example to Estimate Cluster size

Daily data input	100 GB	<i>Storage space used by daily data input = daily data input * replication factor = 300 GB</i>
HDFS replication factor	3	
Monthly growth	5%	<i>Monthly volume = (300 * 30) + 5% = 9450 GB</i> <i>After one year = 9450 * (1 + 0.05)^12 = 16971 GB</i>
Intermediate MapReduce data	25%	<i>Dedicated space = HDD size * (1 - Non HDFS reserved space per disk / 100 + Intermediate MapReduce data / 100)</i> <i>= 4 * (1 - (0.25 + 0.30)) = 1.8 TB (which is the node capacity)</i>
Non HDFS reserved space per disk	30%	
Size of a hard drive disk	4 TB	
Number of DataNodes needed to process:		
<i>Whole first month data = 9.450 / 1800 ~= 6 nodes</i>		
<i>The 12th month data = 16.971 / 1800 ~= 10 nodes</i>		
<i>Whole year data = 157.938 / 1800 ~= 88 nodes</i>		

Choosing The Right Hardware

- Hadoop cluster contains two types of nodes: a master node and a slave node.
- Slave node comprises of Datanodes and NodeManager daemon.
- Master nodes runs Namenode, Secondary Namenode (standby namenode) and resource manager daemon.
- Selecting appropriate hardware for these compute and storage nodes can maximize the efficiency of a Hadoop cluster.

Cont..

- The NameNode role is responsible for coordinating data storage on the cluster.
- The Standby NameNode should not be co-located on the NameNode machine for clusters and will run on hardware identical to that of the NameNode.
- Customers should opt for redundant power and enterprise-grade disks in RAID 1 or 10 configurations for Namenode and Standby Namenode.

Recommended Configurations

For DataNode/TaskTrackers in a balanced Hadoop cluster:

- 2-24 1-4TB hard disks in a JBOD (Just a Bunch Of Disks) configuration
- 2 quad-/hex-/octo-core CPUs, running at least 2-2.5GHz
- 64-512GB of RAM
- Bonded Gigabit Ethernet or 10Gigabit Ethernet (the more storage density, the higher the network throughput needed)

Cont..

For NameNode/JobTracker/Standy NameNode nodes:

- 4–6 1TB hard disks in a JBOD configuration (1 for the OS, 2 for the FS image [RAID 1], 1 for Apache ZooKeeper, and 1 for Journal node)
- 2 quad-/hex-/octo-core CPUs, running at least 2-2.5GHz
- 64-128GB of RAM
- Bonded Gigabit Ethernet or 10Gigabit Ethernet

Cont..

- NameNode RAM requirement is directly proportional to the number of data blocks in the cluster.
- A good rule of thumb is to assume 1GB of NameNode memory for every 1 million blocks stored in the distributed file system.

RAM Considerations

- Slave node configuration specifies the amount of memory and number of cores that Map tasks, Reduce tasks and Application Masters can use on that node.
- Typically each map and reduce task requires 2-4 GB of RAM and Application Master takes 1GB of RAM.
- Slave nodes should not be using virtual memory.
- Ensure you have sufficient RAM to run all the task and additional overhead for Datanode and Nodemanager daemon and operating system.

Cont..

Remember as general rule of thumb:

Total number of tasks = Number of physical processor cores – 1

Equip your slave nodes with as much RAM as you can.

Slave Nodes – Disk Considerations

- Hadoop requires at least two locations for storing it's files: mapred.local.dir, where MapReduce stores intermediary files, and dfs.data.dir, where HDFS stores the HDFS data.
- Cloudera recommends to configure them across the same set of partitions to maximize disk-level parallelism.
- The sizing guide for HDFS is very simple: each file has a default replication factor of 3 and you need to leave approximately 25% of the disk space for intermediate shuffle files.

Cont..

- As a rule of thumb, you need 4x times the raw size of the data you will store in the HDFS.
- You can run Hadoop MapReduce with only 5-10% of the disk space left, the performance will be compromised due to fragmentation.
- Disk performance can be up to 77% slower due to fragmentation and other issues compared to the “empty disk”.
- With a disk more than 80% full you also run the risk of running out of disk space on an individual mount.

Cont..

- In general, more spindle (disks) is better.
- Use 3.5" disk which are faster, cheaper and has higher capacity than 2.5" disk.
- 7200 RPM SATA/SATA II drives are fine
- 8 x 1.5TB drives is likely to be better than 6 x 2 TB drives
- Mechanical hard drives currently provides a significantly better cost/performance ratio than solid-state drives (SSDs)

Why not RAID?

Slave nodes do not benefit from using “Redundant Array of Inexpensive Disks” storage because:

- HDFS provides built-in redundancy by replicating blocks across multiple nodes.
- RAID 0 is actually slower than the JBOD configuration used by HDFS.
- RAID 0 read and write operation are limited by speed of slowest disk in the RAID array.
- Disk operations on JBOD are independent, so the average speed is greater than that of the slowest disk.

Virtualization Options

- A virtualized private cloud uses a group of hardware on same hypervisor such as vSphere [by Vmware], XenServer [by Citrix], KVM [by Red Hat] or Hyper-V [by Microsoft].
- Virtualization works by running a hypervisor either in a host OS or directly on bare metal, replacing the host OS entirely.
- Virtual machines (VMs) can be deployed within a hypervisor and have access to whatever hardware resources are allocated to them by the hypervisor.

Pros

Virtualizing Hadoop brings benefits such as:

- Rapid provisioning
- High Availability
- Elasticity
- Multi-tenancy

Cons

Virtualization will possess some drawbacks like:

- Network contention
- Only one volume if allocated
- No way to guarantee rack placement of nodes
- Possible to have all three replicas of block on same physical host

For cloud deployment, Cloudera partners with Amazon to support Cloudera Enterprise on AWS.

Network Considerations

- Hadoop is very bandwidth-intensive! Often, all nodes are communicating with each other at the same time
- Use dedicated switches for your Hadoop cluster
- Nodes are connected to a top-of-rack switch
- Nodes should be connected at a minimum speed of 1Gb/sec
- For clusters where large amounts of intermediate data is generated, consider 10Gb/sec connections
 - Expensive – Alternative: bond two 1Gb/sec connections to each node

Cont..

- Racks are interconnected via core switches
- Core switches should connect to top-of-rack switches at 10Gb/ sec or faster
- Beware of oversubscription in top-of-rack and core switches
- Consider bonded Ethernet to mitigate against failure
- Consider redundant top-of-rack and core switches

Operating System Recommendations

- **CentOS**: geared towards servers rather than individual workstations
 - Conservative about package versions
 - Very widely used in production
 - **RedHat Enterprise Linux (RHEL)**: RedHat-supported analog to CentOS
 - Includes support contracts, for a price
- In production, we often see a mixture of RHEL and CentOS machines
- Often RHEL on master nodes, CentOS on slaves

Cont..

- **Fedora Core:** geared towards individual workstations
 - Includes newer versions of software, at the expense of some stability – Recommends server-based, rather than workstation-based, Linux distributions
- **Ubuntu:** Very popular distribution, based on Debian
 - Both desktop and server versions available
 - Try to use an LTS (Long Term Support) version
- **SuSE:** popular distribution, especially in Europe
 - Cloudera provides CDH packages for SuSE
- **Solaris, OpenSolaris:** not commonly seen in production clusters

Configuring Nodes

Configure dependencies before deploying CDH:

- Enabling NTP
- Configuring Network Names
- Disabling SELinux
- Disabling the Firewall

Enabling NTP

CDH requires that you configure the Network Time Protocol (NTP) service on each machine in your cluster.

Perform the following steps on each node in your cluster

- Install NTP
- Open the /etc/ntp.conf file and add NTP servers
- Configure the NTP service to run at reboot
- Start the NTP service
- Synchronize the node
- Synchronize the system clock

Configuring Network Names

To ensure the members of the cluster can communicate with each other, do the following:

- Set the hostname of each system to a unique name.
- Make sure /etc/hosts file on each system contains the IP addresses and fully-qualified domain names (FQDN) of all the members of the cluster.
- Make sure the /etc/sysconfig/network file on each system contains the hostname.
- Check that this system is consistently identified to the network.

Disabling SELinux

- Security-Enhanced Linux (SELinux) allows you to set access control through policies.
- Must disable SELinux on each host before you deploy CDH on your cluster.

To disable SELinux, perform the following steps on each host:

- Check the SELinux state.
- Open the /etc/selinux/config file
- Change the line SELINUX=enforcing to SELINUX=permissive
- Save and close the file
- Restart your system

Disabling the Firewall

To disable the firewall on each host in your cluster, perform the following steps on each host:

- Save the existing iptables rule set

- Disable iptables

```
chkconfig iptables off
```

OR

```
/etc/init.d/iptables stop
```

Quick Overview

- Hadoop nodes are typically disk and network I/O bound.
- Use hostnames, not IP addresses to identify nodes while configuring Hadoop.
- Forward and reverse lookup should work while configuring cluster.
- Each node in cluster requires its own configuration files.



Hive, Impala and Pig



Hive

- Initially Hive was developed by Facebook.
- Apache Hive is an open source project run by volunteers at the Apache Software Foundation.
- Apache Hive data warehouse software facilitates reading, writing, and managing large datasets residing in distributed storage using SQL.

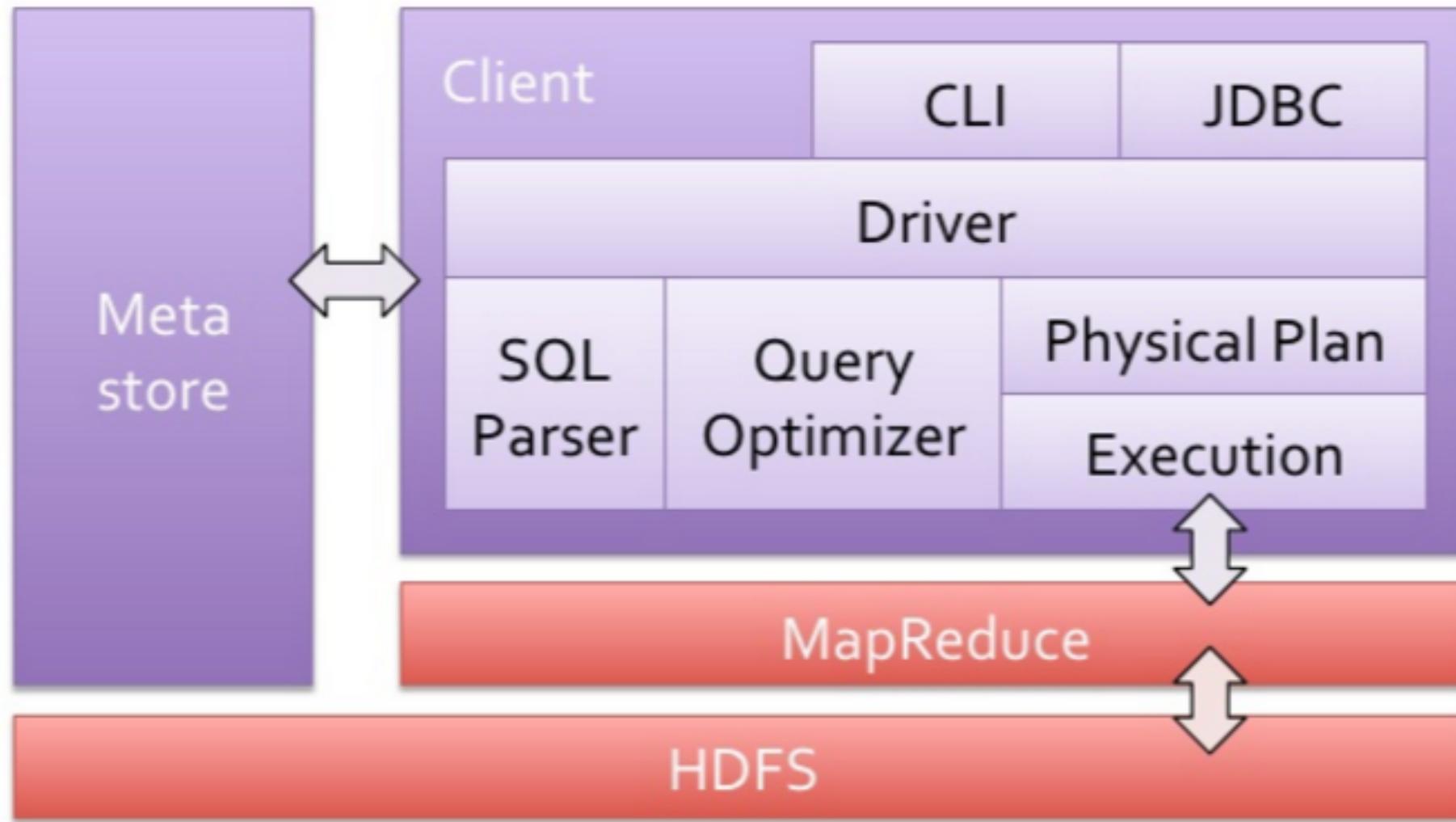
Features of Hive

- It stores schema in a database and processed data into HDFS.
- It is designed for OLAP.
- It is familiar, fast, scalable, and extensible.
- Hive provides a way to query data in HDFS using a SQL-like language.
- Hive turns HiveQL queries into standard MapReduce jobs.

Hive is Not

- A relational database
- A design for Online Transaction Processing (OLTP)
- A language for real-time queries and row-level updates

Architecture of Hive



Components

Meta Store: Hive chooses respective database servers to store the schema or Metadata of tables, databases, columns in a table, their data types, and HDFS mapping.

- A ‘Table’ in Hive represents an HDFS directory.
- By default, Hive uses a Metastore on the user’s local machine.
- A shared Metastore is a database in an RDBMS such as MySQL.

Cont..

HiveQL Process Engine: HiveQL is similar to SQL for querying on schema info on the Metastore.

It is one of the replacements of traditional approach for MapReduce program.

Instead of writing MapReduce program in Java, we can write a query for MapReduce job and process it.

Cont..

Execution Engine: The conjunction part of HiveQL process Engine and MapReduce is Hive Execution Engine.

Execution engine processes the query and generates results as same as MapReduce results.

HDFS or Hbase: Hadoop distributed file system or HBASE are the data storage techniques to store data into file system.

How it works?

Step1: Execute Query

The Hive interface such as Command Line or Web UI sends query to Driver (any database driver such as JDBC, ODBC, etc.) to execute.

Step2: Get Plan

The driver takes the help of query compiler that parses the query to check the syntax and query plan or the requirement of query.

Cont..

Step3: Get Metadata

The compiler sends metadata request to Metastore (any database).

Step4: Send Metadata

Metastore sends metadata as a response to the compiler.

Step5: Send Plan

The compiler checks the requirement and resends the plan to the driver. Up to here, the parsing and compiling of a query is complete.

Cont..

Step6: Execute Plan

The driver sends the execute plan to the execution engine.

Step7: Execute Job

Internally, the process of execution job is a MapReduce job.

Meanwhile in execution, the execution engine can execute metadata operations with Metastore.

Cont..

Step8: Fetch Result

The execution engine receives the results from Data nodes.

Step9: Send Results

The execution engine sends those resultant values to the driver.

The driver sends the results to Hive Interfaces.

Installing and Configuring Hive

- Hive runs on users's machine and not on Hadoop Cluster itself.
 - User can set up Hive with no System Administrator input.
 - If users will be running JDBC-based clients, you can run Hive as a service on a centrally-available machine.
-
- To install Hive, run sudo yum install hive
 - Configure the client so Hive can access the Hadoop cluster.

Configuring the client

- If the client has core-site.xml, yarn-site.xml and mapred-site.xml configured to access the Hadoop cluster, Hive will use the configuration from those files.
- If not, configure fs.defaultFS and yarn.resourcemanager.address in /etc/hive/conf/hive-site.xml

Creating and Configuring a Shared Metastore

Step1: Create a database in RDBMS

```
create database metastore;
```

Step2: Create a database user with appropriate privileges.

```
use metastore;
create user 'hiveuser' @ '%' identified by 'password';
revoke all privileges, grant option from 'hiveuser' @ '%';
grant select, insert, update, delete, lock tables, execute, create,
alter on metastore.* to 'hiveuser' @ '%';
```

Cont..

Step3: Run the schematool utility to import the Hive metastore schema into the database.

```
sudo /usr/lib/hive/bin/schematool –dbType mysql  
–initSchema
```

Step4: Revoke the ALTER and CREATE privileges from the database user

```
Revoke alter, create on metastore.* from ‘hiveuser’ @  
‘%’;
```

Cont..

Step5: Install and start the Hive Metastore service

```
sudo yum install hive-metastore  
sudo service hive-metastore start
```

Step6: Modify `hive-site.xml` on client systems to refer to the shared Metastore.

hive-site.xml properties for Shared Metastore

Property
javax.jdo.option.ConnectionURL
javax.jdo.option.ConnectionDriverName
javax.jdo.option.ConnectionUserName
javax.jdo.option.ConnectionPassword
datanucleus.autoCreateSchema
datanucleus.fixedDatastore
datanucleus.autoStartMechanism
hive.metastore.uris

Impala

- Impala is the open source, native analytic database for Apache Hadoop.
- Impala is a MPP (Massive Parallel Processing) SQL query engine for processing huge volumes of data that is stored in Hadoop cluster.
- Impala can read almost all the file formats such as Parquet, Avro, RCFile used by Hadoop.

Cont..

- Impala uses the same metadata, SQL syntax (Hive SQL), ODBC driver, and user interface (Hue Beeswax) as Apache Hive, providing a familiar and unified platform for batch-oriented or real-time queries.
- Unlike Apache Hive, Impala is not based on MapReduce algorithms.
- Impala queries run on a additional set of daemons that run on the Hadoop cluster, referred to as ‘Impala Server’.

Advantages of Impala

- Using impala, you can process data that is stored in HDFS at lightning-fast speed with traditional SQL knowledge.
- Data transformation and data movement is not required for data stored on Hadoop, while working with Impala.
- Using Impala, you can access the data that is stored in HDFS, HBase, and Amazon s3 without the knowledge of Java (MapReduce jobs). You can access them with a basic idea of SQL queries.

Cont..

- Impala is pioneering the use of the Parquet file format, a columnar storage layout that is optimized for large-scale queries typical in data warehouse scenarios.

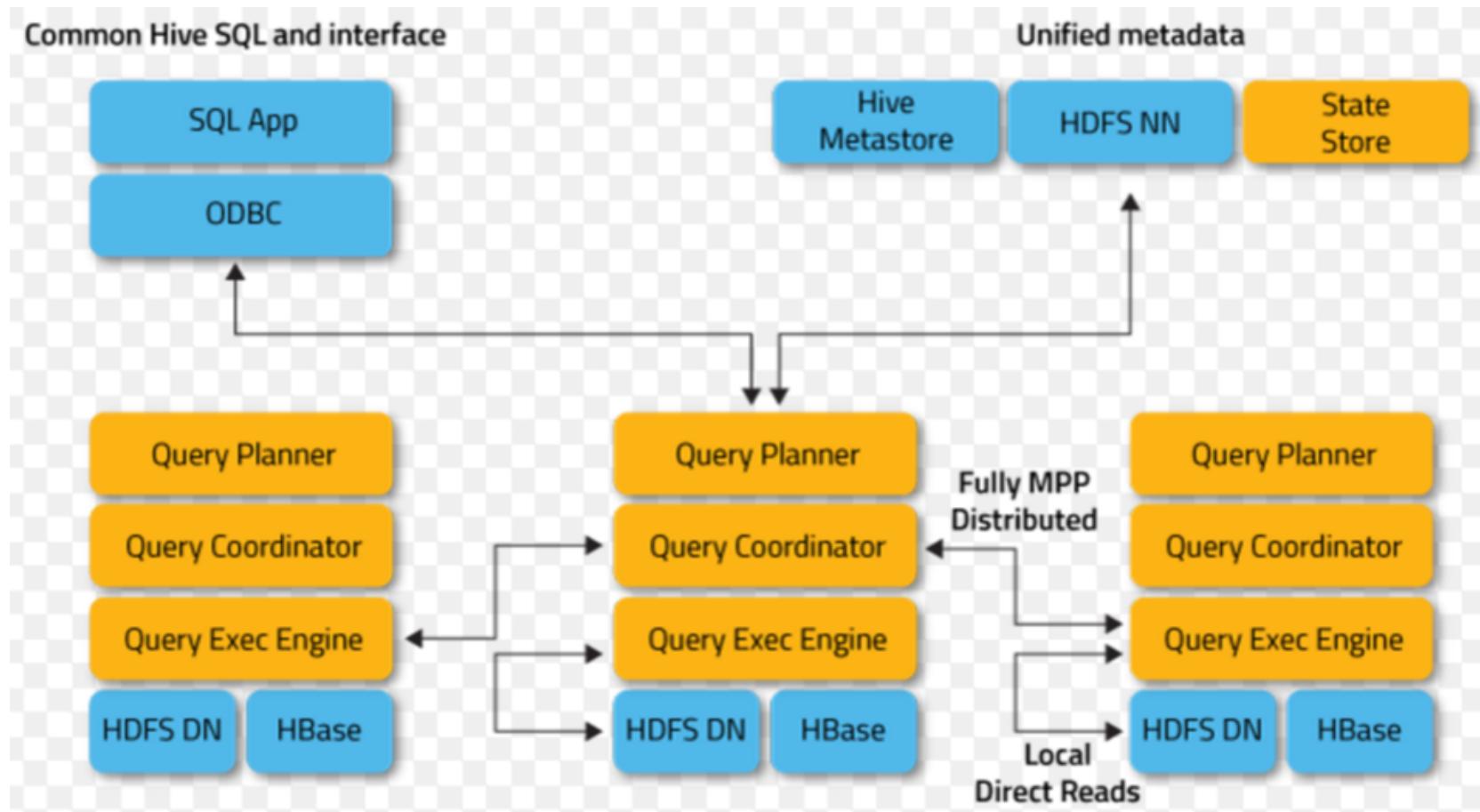
Features of Impala

- Impala is available freely as open source under the Apache license.
- Impala supports in-memory data processing.
- You can access data using Impala using SQL-like queries.
- Impala provides faster access for the data in HDFS when compared to other SQL engines.
- You can integrate Impala with business intelligence tools like Tableau, Pentaho, Micro strategy, and Zoom data.
- Impala supports various file formats such as, LZO, Sequence File, Avro, RCFile, and Parquet.
- Impala uses metadata, ODBC driver, and SQL syntax from Apache Hive.

Drawbacks of Impala

- Impala does not provide any support for Serialization and Deserialization.
- Impala can only read text files, not custom binary files.
- Whenever new records/files are added to the data directory in HDFS, the table needs to be refreshed.

Impala Architecture



Main Components

- **Impala Daemon (Impalad)**

Runs on each node where Impala is installed. It accepts the queries from various interfaces like impala shell, hue browser, etc.... and processes them.

- **Impala Statestore**

Is responsible for checking the health of each *Impalad*

- **Impala Metastore**

The important details such as table & column information & table definitions are stored in a centralized database known as a meta store.

Impala Interfaces

- **Impala-shell**

Start the impala shell by typing the command `impala-shell`.

- **Hue interface**

Logging to the HUE browser and access the editor to execute impala queries.

- **ODBC/JDBC driver**

Connect to impala through programming languages that supports these drivers and build applications.

Deploying Impala

- Impala server should reside on each Datanode host.
- One Impala state store server and one Impala Catalog server on the cluster.
 - Co-locate with Namenode, Secondary NameNode, or other server.

Installing, Configuring and Starting Impala

1. Install the impala software
2. Configure HDFS for optimal Impala performance.
3. Restart all the Datanodes
4. If necessary, configure hive-site.xml for shared Hive Metastore.
5. Copy hive-site.xml, core-site.xml, hdfs-site.xml and log4j.properties to /etc/impala/conf on all the host that will run Impala server.
6. Configure startup options in /etc/default/impala on all the hosts that will run impalad, Impala state store server or the Impala Catalog server.
7. Start the Impala server, Impala state store server or the Impala Catalog server.

Installing Software

- Install the Impala Server on all hosts running DataNodes:

```
sudo yum install impala-server
```

- Install the Impala State Store server on a single host:

```
sudo yum install impala-state-store
```

- Install the Impala Catalog Server on the same host as the Impala State Store server

```
sudo yum install impala-catalog
```

Cont..

- Install the Impala shell on clients

```
sudo yum install impala-shell
```

HDFS Configuration for Impala

<code>dfs.client.read.shortcircuit</code> (<code>hdfs-site.xml</code>)	Allows daemons to read directly off their host's disks instead of having to open a socket to talk to DataNodes. Required value: <code>true</code> .
<code>dfs.domain.socket.path</code> (<code>hdfs-site.xml</code>)	Short-circuit reads use a UNIX domain socket, which requires a path. Recommended value: <code>/var/run/hadoop-hdfs/dn._PORT</code> .
<code>dfs.client.file-block-storage-locations.timeout.millis</code> (<code>hdfs-site.xml</code>)	Timeout on a call to get the locations of required file blocks. Recommended value: <code>10000</code> .
<code>dfs.datanode.hdfs-blocks-metadata.enabled</code> (<code>hdfs-site.xml</code>)	Expose the disk on a datanode on which a block resides. Recommended value: <code>true</code> .

Changes to Impala Startup Options

In /etc/default/impala, modify below configurations:

- IMPALA_STATE_STORE_HOST

Hostname of the host running the Impala State Store Server

- IMPALA_CATALOG_SERVICE_HOST

Hostname of the host running the Impala Catalog Server

- IMPALA_SERVER_ARGS

Ex: -mem_limit=70%, -mem_limit=31GB

Start the services

- Impala Server

```
sudo service impala-server start
```

- Impala State Store Server

```
sudo service impala-state-store start
```

- Impala Catalog Server

```
sudo service impala-catalog start
```

Pig

- Apache Pig is an abstraction over MapReduce.
- Apache Pig is a data flow language.
- It is a tool/platform which is used to analyse larger sets of data representing them as data flows.
- Pig provides a high-level language known as **Pig Latin**.
- Pig Latin is a **procedural** language.
- Scripts written in Pig Latin, are internally converted to Map and Reduce tasks.
- Easy to write complex tasks such as joins of multiple datasets.

Why we need Apache Pig?

- Apache Pig is a boon for programmers who are not good at JAVA.
- Apache Pig uses multi-query approach, reduces the length of code.
- Pig Latin is SQL-like language.
- Apache Pig provides nested data types like tuples, bags and maps.

Features of Pig

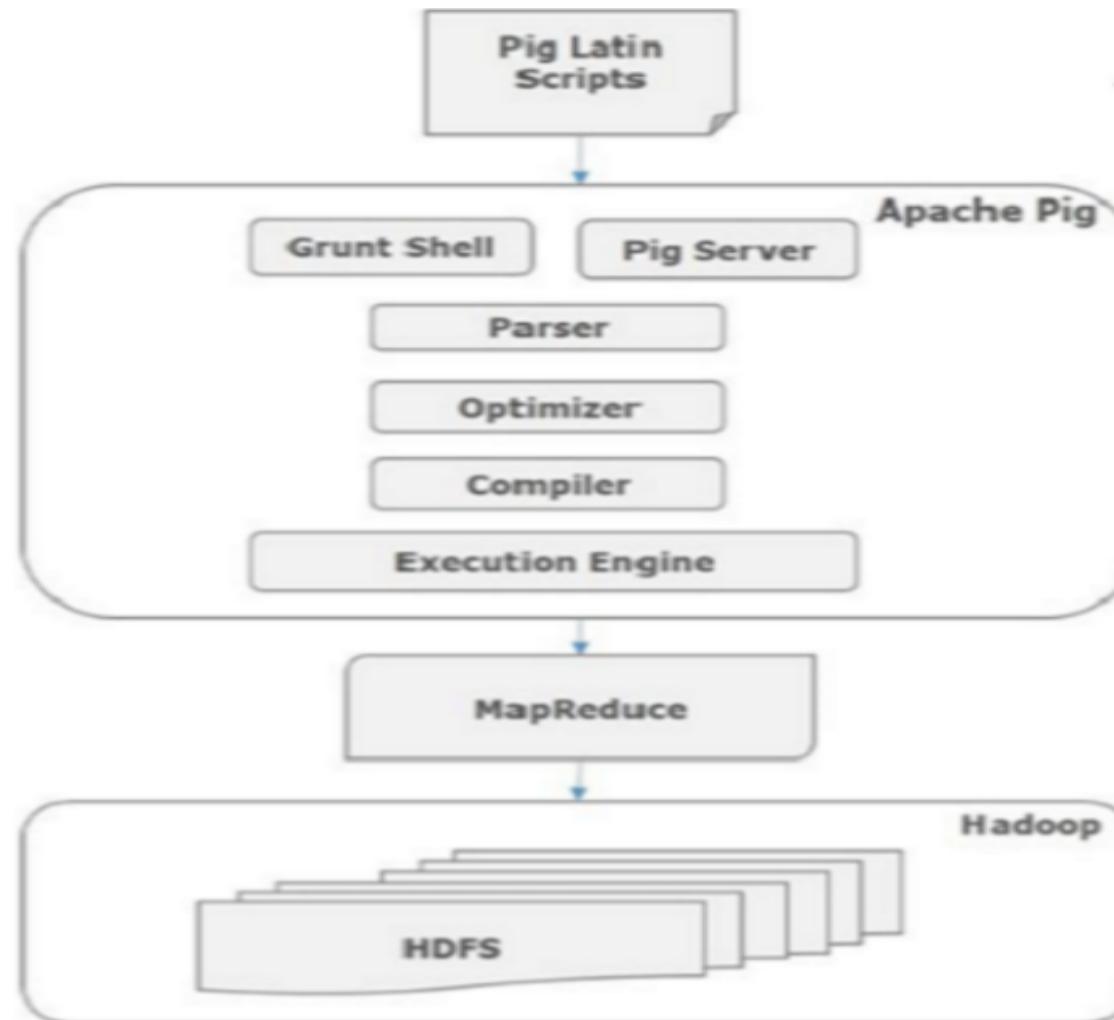
- Rich set of operators
- Ease of programming
- Apache Pig optimize their execution automatically.
- Extensibility: user can develop their own function to read, process and write data.
- Facility to create User Defined Functions.
- Handles all kind of Data

Application of Pig

Apache Pig is used –

- To process huge data sources such as web logs.
- To perform data processing for search platforms.
- To process time sensitive data loads.

Apache Pig- Architecture



Components

Parser:

- Pig Scripts are handled by the Parser,
- It checks the syntax of the script, does type checking, and other miscellaneous checks.
- The output of the parser will be a DAG (directed acyclic graph), which represents the Pig Latin statements and logical operators.

Cont..

Compiler: The compiler compiles the optimized logical plan into a series of MapReduce jobs.

Execution engine: The MapReduce jobs are submitted to Hadoop in a sorted order. Finally, these MapReduce jobs are executed on Hadoop producing the desired results.

Pig Latin Data Model

Atom

- Any single value in Pig Latin, irrespective of their data, type is known as an **Atom**.

Tuple

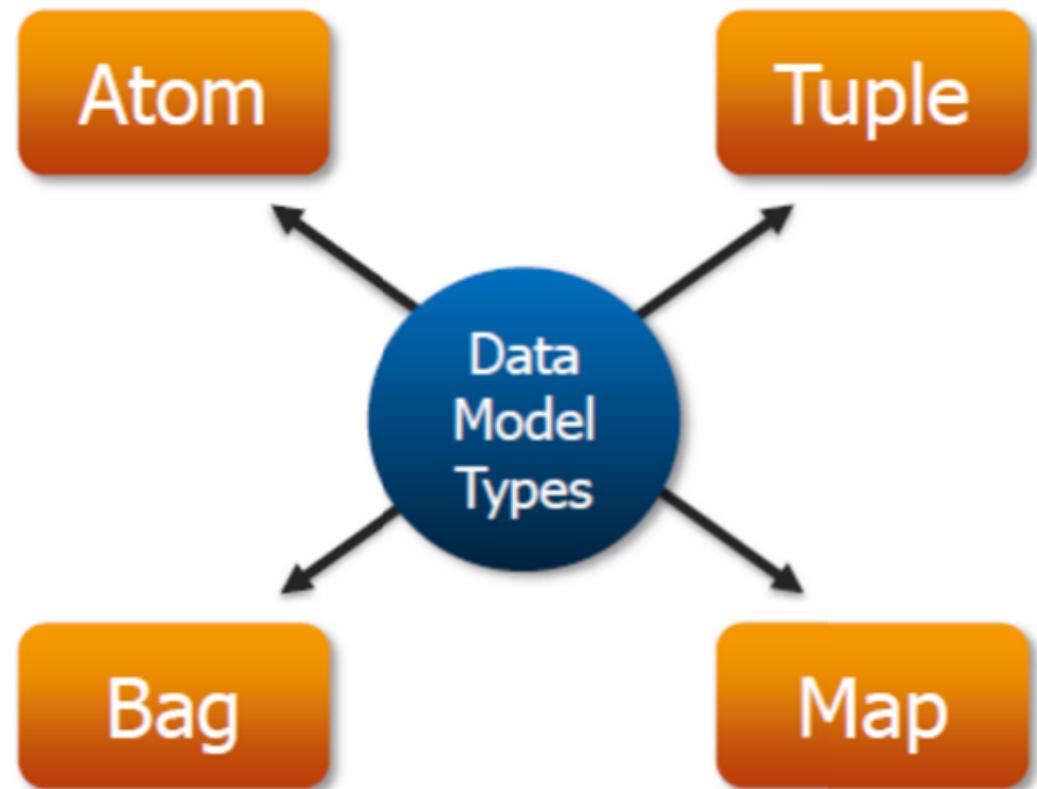
- A record that is formed by an ordered set of fields is known as a tuple.
- A tuple is similar to a row in a table of RDBMS

Bag

- A bag is an unordered set of tuples.

Map

- A map (or data map) is a set of key-value pairs.



Installing Pig

- Pig runs as a client-side application
- To install, run

```
sudo yum install pig
```

- Configure the client so Pig can access the Hadoop Cluster by modifying the configuration properties in core-site.xml, yarn-site.xml and mapred-site.xml.

Apache Pig Execution Mechanisms

Pig scripts can be executed in three ways:

- **Interactive Mode** (Grunt shell) – In this shell, you can enter the Pig Latin statements and get the output (using Dump operator).
- **Batch Mode** (Script) – You can run Apache Pig in Batch mode by writing the Pig Latin script in a single file with .pig extension.
- **Embedded Mode** (UDF) – Apache Pig provides the provision of defining our own functions (**User Defined Functions**) in programming languages such as Java, and using them in our script.

Quick Overview

- Hive provides a way to query data in HDFS using SQL-like language.
- Hive turns HiveQL into standard MapReduce jobs.
- Impala uses same shared Metastore that Hive uses.
- Pig is another high-level abstraction on top of MapReduce.



Hadoop Clients Including Hue

We will cover following under this section:

- What are Hadoop Clients?
- Installing and Configuring Hadoop Clients
- Installing and Configuring Hue
- Hue Authentication and Authorization

Hadoop Client

A Hadoop client requires

- Hadoop API to access services running on Hadoop Cluster.
- Configurations to connect to Hadoop components.

Examples

Command line clients

- The Hadoop shell
- The Pig shell
- The sqoop command-line interface

Server Daemons

- Flume agent
- Centralized sqoop and Hive servers
- Oozie

Installing Hadoop Clients

1. Hadoop Shell and MapReduce

```
sudo yum install hadoop-client
```

2. Hive Shell

```
sudo yum install hive
```

3.Pig Shell

```
sudo yum install pig
```

Cont..

4. Sqoop shell

```
sudo yum install sqoop
```

5. Flume Agent

```
sudo yum install flume-ng-agent
```

6. Centralized Hive Server

```
sudo yum install hive-server2
```

Cont..

7. Centralized Sqoop Server

```
sudo yum install sqoop2
```

8. Oozie

```
sudo yum install oozie
```

Hadoop client installers automatically include any required Hadoop API as dependencies if they are not already installed.

Configuring Hadoop Client

- Clients must be configured to identify the Hadoop cluster mainly Namenode location and ResourceManager location.
- Copy the Hadoop configuration files to clients.
- For client specific configurations, update the corresponding properties files accordingly.

Example: hive-site.xml, pig-properties, flume-conf.properties etc

Cont..

- Client configuration files are generated automatically by Cloudera Manager based on the services and roles you have installed.
- Cloudera Manager creates zip files that contain the relevant configuration files with the settings for your services.
- Cloudera Manager deploys these configurations automatically when you install your cluster, when you add a service on a host, or when you add a Gateway role on a host.

Example: The MapReduce client configuration zip file contains copies of core-site.xml, Hadoop-env.sh, hdfs-site.xml, log4j.properties and mapred-site.xml

Viewing the Client Configuration Files

- Click the **Services** tab in the Cloudera Manager Admin Console.
- Click the **Client Configuration URLs** button. This opens a popup window with links to the configuration zip files that have been created for the services you have installed: HDFS, MapReduce, YARN, and Hbase.
- Click a link to initiate a download of the configuration zip file to your local system.

Installing and Configuring Hue

- Install the hue-common package on the machine where you will run the Hue Server

```
sudo yum install hue hue-server
```

- To enable communication between the Hue server and CDH components, make changes to your CDH configuration files in /etc/hadoop/conf

Cont..

- For security, specify the secret key that is used for secure hashing in the session store.
 - Open the hue.ini configuration file and enter random characters under desktop section

```
[desktop] secret_key=qpbdxoewsqlkhztybvfidtvwekftusgdlofbcfghaswuicmqp
```

- Start the hue server

```
sudo service hue start
```

Cont..

- Access Hue from a browser

http://<hue_server>:8888

Hue Authentication And Authorization

- The first user to log in to Hue automatically receives superuser privileges.
- Superuser can be added or removed.
- The first user's login credentials are stored in the Hue database
- Administrator configures Hue to access the LDAP directory
- Administrator syncs users and groups from LDAP to the Hue database.
- Hue authenticates users by accessing credentials from the Hue database.

Quick Overview

- Hadoop Client accesses functionality within a Hadoop cluster using Hadoop API.
- Clients must be configured to identify Hadoop cluster.
- HUE provides many capabilities such as browsing files and MapReduce jobs, submitting Hive and Impala queries, checking the status of Oozie workflow.



Advanced Cluster Configuration

We will cover following:

- Advanced Configuration parameters
- Configuring Hadoop Ports
- Configuring HDFS for Rack Awareness
- Configuring HDFS high Availability

Advanced Configuration parameters

Advanced configurations fall into following categories:

- Optimization and performance tuning
- Capacity management
- Access Control

Properties

For NameNode:

1. The number of threads the NameNode uses to handle RPC request from Datanode i.e. "**dfs.namenode.handler.count**" parameter in hdfs-site.xml. Default value is 10, ideally it should be set as per cluster need.

Cont..

For DataNode:

1. The number of volumes allowed to fail before the DataNode take itself offline i.e. "**dfs.datanode.failed.volumes.tolerated**" parameter in hdfs-site.xml. Default value is 0

2. The maximum amount of memory a DataNode can use for caching i.e. "**dfs.datanode.max.locked.memory**" parameter in hdfs-site.xml. Default value is 0.

Cont..

For Trash

1. When a file is deleted, it is placed in a .Trash directory in the user's home directory, rather than being immediately deleted. It is purged from HDFS after the number of minutes specified.

2. Edit the parameter '**fs.trash.interval**' in core-site.xml file. Default value is set to 0, recommended is 1440.

Cont..

For file Read and Write Buffering:

1. The parameter '**io.file.buffer.size**' in core-site.xml file, determines how much data is buffered during the read and write operation.
2. It should be a power of 2 of hardware page size.
3. Default value is 4096, recommended value is 64KB.

Cont..

Resource Allocation: Memory

1. The amount of RAM, available on the host for YARN managed task can be configured using '**yarn.nodemanager.resource.memory-mb**' parameter in `yarn-site.xml`
2. Default value set to 8192
3. Recommendation: the amount of RAM on the host minus the amount of memory needed for non-YARN-managed work.

Cont..

Resource Allocation: CPU

1. The '**yarn.nodemanager.resource.cpu-vcores**' parameter in `yarn-site.xml` decides the number of cores on the host for yarn managed tasks.
2. Default value is set to 8
3. Recommendation: the number of physical cores on the host minus 1.

Cont..

Resource Allocation: Container Memory

1. Minimum amount of memory to allocate for a container can be set using **yarn.scheduler.minimumallocation-mb** (yarn-site.xml) property.
2. Default value is 1024, can be increased upto 4096 depending upon need.
3. Amount of memory for Map or Reduce task can be set using **mapreduce.map.memory.mb** and **mapreduce.reduce.memory.mb** (mapred-site.xml) properties.
4. Amount of memory for Application Master can be configured using **yarn.app.mapreduce.am. resource.mb** (mapred-site.xml) property.

Cont..

Container Heap Size

1. **yarn.app.mapreduce.am.command-opts** (mapred-site.xml)

Java options passed to the ApplicationMaster.

Default is -Xmx1024m (1GB of heap space). Recommendation: -Xmx768m

2. **mapreduce.map.java.opts mapreduce.reduce.java.opts** (mapred-site.xml)

Java option passed to Mapper and Reducers.

Default is -Xmx200m (200MB of heap space). Recommendation: increase to a value from 1GB to 4GB

Note

- Adjust both the Java heap size properties and the resource allocation properties.
- Set the Java heap size for Mappers and Reducers to 75% of **mapreduce.map.memory.mb** and **mapreduce.reduce.memory.mb**

File Compression

- Specify list of compression codecs that Hadoop can use for file compression under **io.compression.codecs** (core-site.xml) property.
- To determine whether intermediate data from Mappers should be compressed before transfer across the network, set **mapreduce.map.output.compress** (mapred-site.xml) property to true.
- The compression codec used to compress intermediate data from Mappers can be set to org.apache.hadoop.io.compress.SnappyCodec under property **mapreduce.map.output.compress.codec** (mapred-site.xml).

Optimizing Shuffle and Sort

1. **mapreduce.task.io.sort.mb** (mapred-site.xml)

- The size of the buffer in RAM on the Mapper to which the Mapper initially stores its Key/Value pairs before they are written to disk.
- Default: 100MB. Recommendation: 256MB, if the child heap size is 1GB

2. **mapreduce.task.io.sort.factor** (mapred-site.xml)

- The number of streams to merge at when sorting files.
- Default: 10. Recommendation: 64. Used by Mappers and Reducers

Configuring Hadoop Ports

- Cloudera Manager, CDH components, managed services, and third-party components use the ports.
- Before you deploy Cloudera Manager, CDH, and managed services, and third-party components make sure these ports are open on each system.
- If you are using a firewall, such as iptables, and cannot open all the listed ports, you will need to disable the firewall completely to ensure full functionality.

Web UI Ports

	Daemon	Default Port	Configuration parameter
HDFS	NameNode	50070	dfs.namenode.http-address
	DataNode	50075	dfs.datanode.http.address
	Secondary NameNode	50090	dfs.namenode.secondary.http-address
YARN	ResourceManager	8088	yarn.resourcemanager.webapp.address
	NodeManager	8042	yarn.nodemanager.webapp.address
	JobHistoryServer	19888	mapreduce.jobhistory.webapp.address

Default Ports for Administrator

Daemon	Default Port	Configuration Parameter	Used for
NameNode	8020	fs.defaultFS	Filesystem metadata operations
DataNode	50010	dfs.datanode.address	DFS data transfer
	50020	dfs.datanode.ipc.address	Block metadata operations and recovery
ResourceManager	8032	yarn.resourcemanager.address	Application submission; used by clients
	8033	yarn.resourcemanager.admin.address	Administration RPC server port; used by the yarn rmadmin client
	8030	yarn.resourcemanager.scheduler.address	Scheduler RPC port; used by ApplicationMasters
	8031	yarn.resourcemanager.resource-tracker.address	Resource tracker RPC port; used by NodeManagers

Cont..

Daemon	Default Port	Configuration Parameter	Used for
NodeManager	8040	yarn.nodemanager.localizer.address	Resource localization RPC port
JobHistoryServer	10020	mapreduce.jobhistory.address	JobHistoryServer RPC port; used by clients to query job history.
ShuffleHandler (MapReduce's auxiliary service)	13562	mapreduce.shuffle.port	ShuffleHandler's HTTP port; used for serving Mapper outputs

Configuring HDFS for Rack Awareness

- Hadoop components are rack-aware.
- For example, HDFS block placement will use rack awareness for fault tolerance by placing one block replica on a different rack.
- If your cluster contains more than 10 hosts, Cloudera recommends that you specify the rack for each host.
- Cloudera Manager includes internal rack awareness scripts, but you must specify the racks where the hosts in your cluster are located.

Cont..

- Cloudera Manager supports nested rack specifications.
- For example, you could specify the rack /rack3, or /group5/rack3 to indicate the third rack in the fifth group.
- All hosts in a cluster must have the same number of path components in their rack specifications.

Specifying Rack for Hosts

To specify racks for hosts:

- Click the **Hosts** tab.
- Check the checkboxes next to the host(s) for a particular rack, such as all hosts for /rack123
- Click **Actions for Selected (*n*) > Assign Rack**, where *n* is the number of selected hosts.
- Enter a rack name or ID that starts with a slash /, such as /rack123 and click **Confirm**
- Optionally restart affected services. Rack assignments are not automatically updated for running services.

Configuring HDFS High Availability

Background:

- The NameNode was a single point of failure (SPOF) in an HDFS cluster.
- In the case of an unplanned event such as a machine crash, the cluster would be unavailable until an operator restarted the NameNode.
- Planned maintenance events such as software or hardware upgrades on the NameNode machine would result in periods of cluster downtime.

Cont..

- The HDFS HA feature addresses the problems by providing the option of running two NameNodes in the same cluster, in an Active/Passive configuration.
- Unlike the Secondary NameNode, the Standby NameNode is hot standby, allowing a fast failover to a new NameNode.

HDFS High Availability Architecture

- An HDFS HA cluster is configured with two NameNodes - an Active NameNode and a Standby NameNode.
- At any point in time, one of the NameNodes is in an Active state, and the other is in a Standby state.
- The Active NameNode is responsible for all client operations in the cluster, while the Standby is simply acting as a slave, maintaining enough state to provide a fast failover if necessary.

Cont..

- Clients only contact the Active NameNode.
- DataNodes sends heartbeat to both NameNode.
- Active NameNode writes its metadata to a quorum of JournalNodes.
- Standby NameNode reads from the JournalNodes to remain in sync with the Active NameNode

How to Configure HDFS HA?

There are two implementations available for maintaining the copies of the edit logs:

- High Availability using Quorum-based Storage
- High Availability using an NFS-mounted shared edits directory

Cont..

- Quorum-based Storage relies upon a set of JournalNodes, each of which maintains a local edits directory that logs the modifications to the namespace metadata.
- Other alternative is to use a NFS-mounted shared edits directory (typically a remote Filer) to which both the Active and Standby NameNodes have read/write access.
- Once you have enabled High Availability, you can enable Automatic Failover, which will automatically failover to the Standby NameNode in case the Active NameNode fails.

Quorum Based Storage

- **Quorum-based Storage** refers to the HA implementation that uses Quorum Journal Manager (QJM).
- For the Standby node to keep its state synchronized with the Active node, both nodes communicate with a group of separate daemons called **JournalNodes**.
- The Standby node is capable of reading the edits from the JournalNodes, and is constantly watching them for changes to the edit log.

Cont..

- As the Standby Node sees the edits, it applies them to its own namespace.
- In the event of a failover, the Standby will ensure that it has read all of the edits from the JournalNodes before promoting itself to the Active state.
- Ensures that the namespace state is fully synchronized before a failover occurs.

Cont..

- In order to provide a fast failover, it is also necessary that the Standby node has up-to-date information regarding the location of blocks in the cluster.
- To achieve this, the DataNodes are configured with the location of both NameNodes, and they send block location information and heartbeats to both.

Cont..

- For correct operation of an HA cluster only one of the NameNodes should be active at a time.
- Otherwise, the namespace state would quickly diverge between the two, risking data loss or other incorrect results.
- In order to ensure this property and prevent the so-called "split-brain scenario," the JournalNodes will only ever allow a single NameNode to be a writer at a time.
- During a failover, the NameNode which is to become active will simply take over the role of writing to the JournalNodes.

Initial Deployment

1. Install and Start the JournalNodes

```
sudo yum install hadoop-hdfs-journalnode
```

2. Create the shared edits directory on the JournalNode

3. Start the JournalNode

```
sudo service hadoop-hdfs-journalnode start
```

Cont.

4. Initialize the shared edits directory

```
hdfs namenode -initializeSharedEdits
```

5. Start the Primary NameNodes

```
sudo service hadoop-hdfs-namenode start
```

6. Run hdfs haadmin –getServiceState command to verify that existing Namenode is not active yet

```
sudo -u hdfs hdfs haadmin –getServiceState nn1
```

Cont..

7. Install the second NameNode and Bootstrap it

```
sudo -u hdfs hdfs namenode -bootstrapStandby  
sudo service hadoop-hdfs-namenode start
```

- Starting the standby NameNode with the `-bootstrapStandby` option copies over the content of the primary NameNode's metadata directories to the standby NameNode.

8. Run `hdfs haadmin –getServiceState` command to verify that existing Namenode is not active yet

```
sudo -u hdfs hdfs haadmin –getServiceState nn2
```

Cont..

10. Install the ZooKeeper failover Controller on the same host as the NameNode

```
sudo yum install --assumeyes hadoop-hdfs-zkfc
```

11. Initialize the current NameNode state in Zookeeper

```
sudo -u hdfs hdfs zkfc -formatZK
```

Cont..

- Start the ZooKeeper Failover Controller

```
sudo service start hadoop-hdfs-zkfc
```

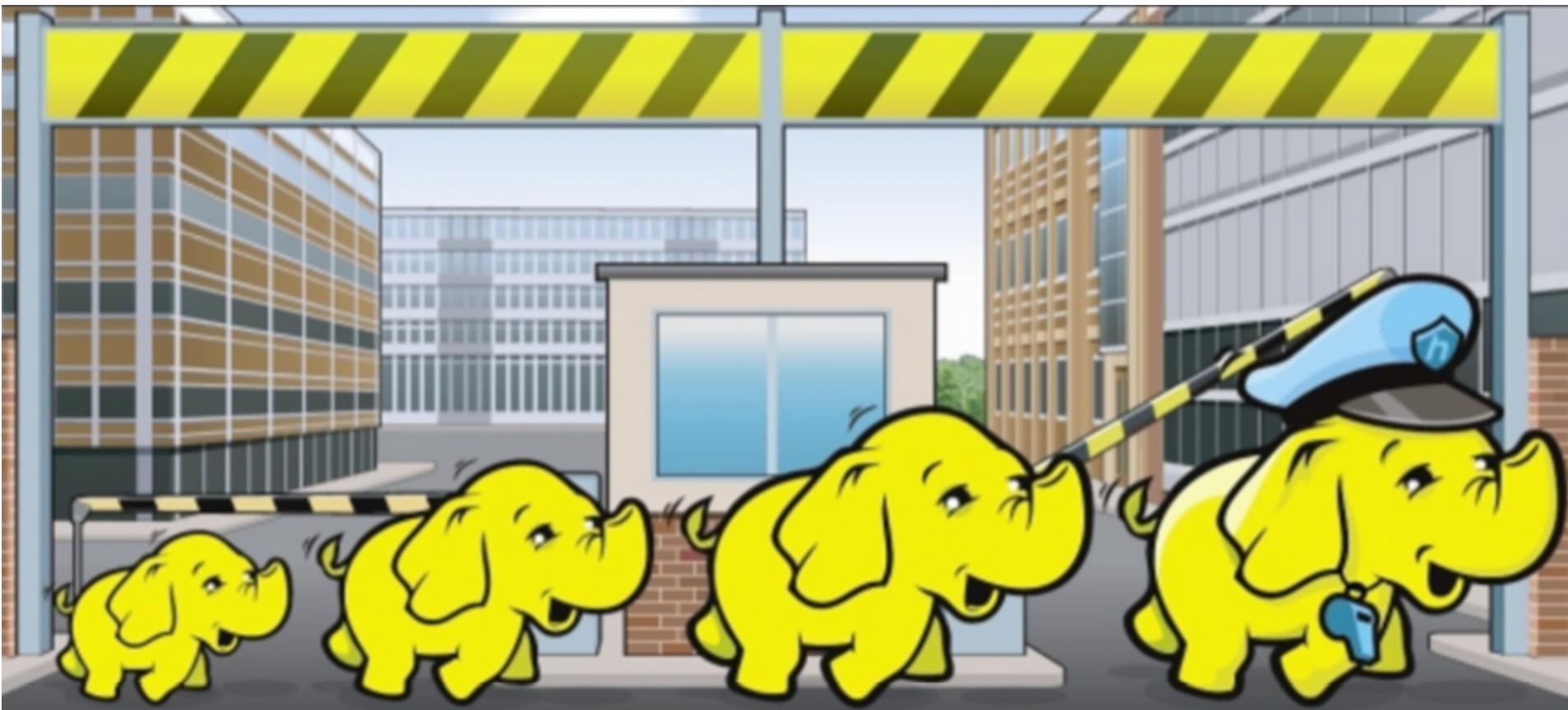
- Restart the Datanodes and the Yarn and MapReduce daemons.
- Restart the Active NameNode and then check the NameNode service state to test automatic failover.

Quick Overview

- HDFS is ‘rack aware’.
- Default ports for Hadoop administration.
- How to avoid “spilt-brain syndrome”.
- How to fence the Namenode
- Manual and automatic failover.
- Zookeeper failover controller runs on each Namenode machine.



Hadoop Security



Why Hadoop Security is Important?

- Big data that resides within a Hadoop environment can contain sensitive financial data in the form of credit card and bank account numbers.
- It may also contain proprietary corporate information and personally identifiable information (PII) such as the names, addresses and social security numbers of clients, customers and employees.
- Due to the sensitive nature of all of this data and the damage that can be done should it fall into the wrong hands, it is imperative that it be protected from unauthorized access.

Challenges

- Ensuring proper authentication of users who access Hadoop.
- Ensuring that authorized Hadoop users can only access the data that they are entitled to access.
- Ensuring that data access histories for all users are recorded in accordance with compliance regulations and for other important purposes.
- Ensuring the protection of data—both at rest and in transit—through enterprise-grade encryption.

Hadoop's Security System Concepts

Perimeter

Guarding access to the cluster itself

Technical Concepts:
Authentication
Network isolation

Access

Defining what users and applications can do with data

Technical Concepts:
Permissions
Authorization

Visibility

Reporting on where data came from and how it's being used

Technical Concepts:
Auditing
Lineage

Data

Protecting data in the cluster from unauthorized visibility

Technical Concepts:
Encryption, Tokenization,
Data masking

Perimeter

- Enables isolation of the Hadoop cluster using gateway and properly configured firewall rules.
- Authentication:
 - Confirming the identification of participant
 - Checking the credentials
- Kerberos authentication

Access

- Allow system administrators to control access to Hadoop data using role-based authorization.
- Fine-grained access control for data stored in HDFS
- Resource-level access control for YARN
- Coarser-grained service level access control for MapReduce Operations
- Table and column family level access control for HBase data
- Table level access control for Apache Hive data sets

Visibility

Allows you to track Hadoop activity using Native Auditing (audit logs) including

- Access requests
- Data processing operations
- Data changes

Cloudera Navigator – centralized audit capabilities.

Data

- Provides the mechanisms for encrypting data in flight
- Requires the use of partner solutions for encrypting data at rest, data discovery, and data masking.

Supports the following wire encryption methods:

- SSL for PHD Components
- RPC encryption
- Data Transfer Protocol

Types of Hadoop Security

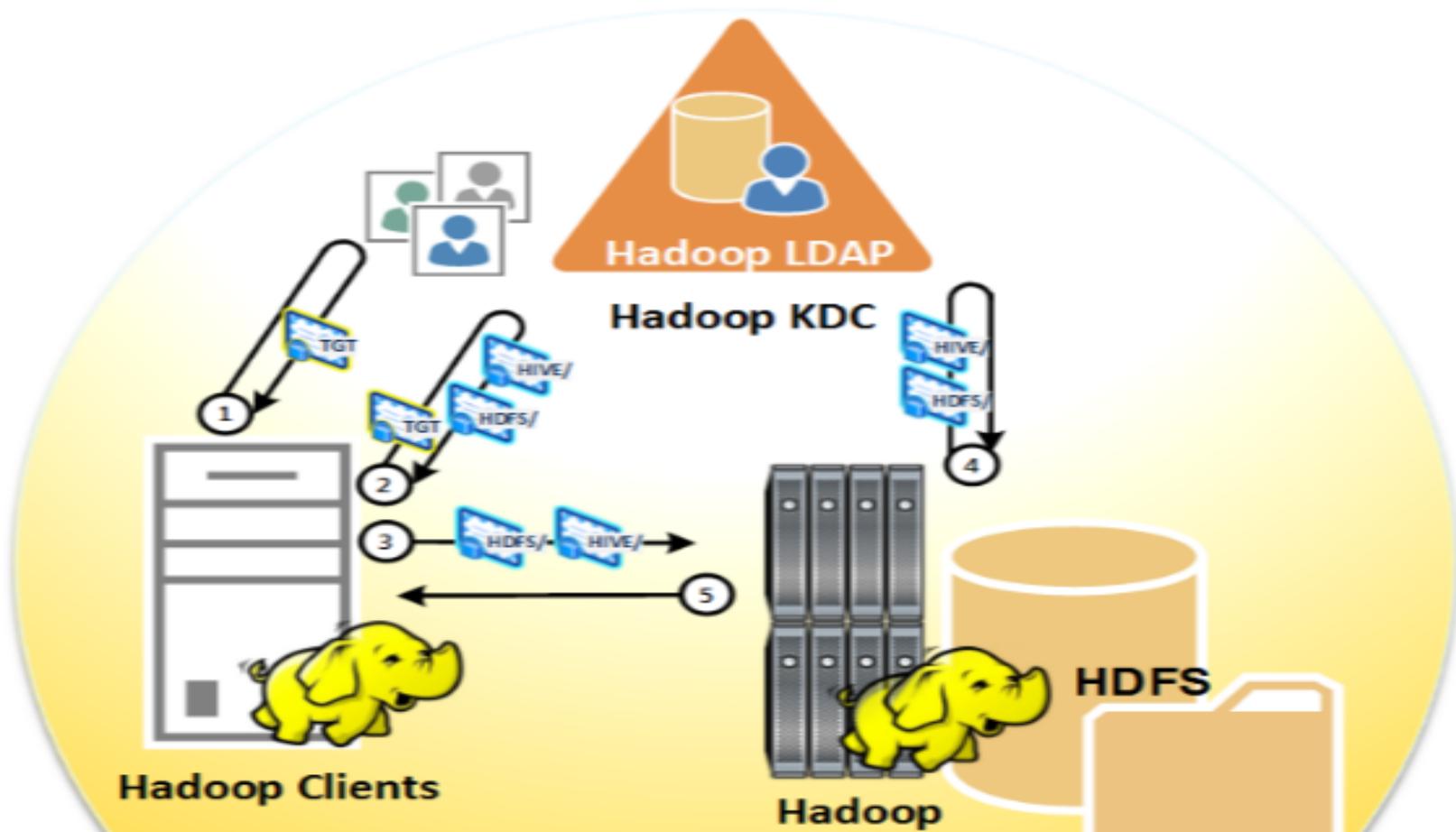
Type	Example
Access	Physical (lock and key), Virtual (Firewalls, VLANS)
Authentication	Logins – verify users are who they say they are
Authorization	Permissions – verify what a user can do
Encryption at Rest	Data protection for files on disk
Encryption in transport	Data protection on the wire
Auditing	Keep track of who accessed what
Policy / Procedure	Protect against Human Error & Social Engineering

What Kerberos is and How it works ?

- Kerberos is a secure method for authenticating a request for a service in a computer network.
- Kerberos was a three-headed dog who guarded the gates of Hades.
- Kerberos lets a user request an encrypted "ticket" from an authentication process that can then be used to request a particular service from a server.
- The user's password does not have to pass through the network.

Working

- To access a server on another computer, it requires a Kerberos "ticket" before it will process your request.
- To get your ticket, you first request authentication from the Authentication Server (AS).
- The Authentication Server creates a "session key". The session key is effectively a "ticket-granting ticket."
- Next send your ticket-granting ticket to a ticket-granting server (TGS).
- The TGS returns the ticket that can be sent to the server for the requested service.
- The service either rejects the ticket or accepts it and performs the service.



- 1 User obtains a Ticket Granting Ticket (TGT)
- 2 User uses TGT to request a Service Ticket for Hadoop Service (HDFS/HIVE)
- 3 User connects to Hadoop Service providing the Service Ticket for authentication
- 4 User authenticated using the Service Ticket & Service Key
- 5 Results returned from Hadoop Service

Kerberos Terminology

Authentication Server:

A server that issues tickets for a desired service which are in turn given to users for access to the service.

Key Distribution Center (KDC):

A service that issues Kerberos tickets, usually run on the same host as the Ticket-granting Server (TGS).

Cont..

- **Keytab:**

A file that includes an unencrypted list of principals and their keys.

- **kinit:**

The kinit command allows a principal who has already logged in to obtain and cache the initial Ticket-granting Ticket (TGT).

Cont..

- **Principal**

The principal name or principal is the unique name of a user or service allowed to authenticate using Kerberos.

- **Realm**

A network that uses Kerberos, composed of one or more servers called KDCs and a potentially large number of clients.

- **Ticket**

A temporary set of electronic credentials that verify the identity of a client for a particular service.

Cont..

- **Ticket-granting Server (TGS)**

A server that issues tickets for a desired service which are in turn given to users for access to the service. The TGS usually runs on the same host as the KDC.

- **Ticket-granting Ticket (TGT)**

A special ticket that allows the client to obtain additional tickets without applying for them from the KDC.

Securing Hadoop Cluster With Kerberos

Pre-requisites:

1. Working Hadoop Cluster
2. Working Kerberos KDC server
3. Kerberos client libraries installed on all Hadoop nodes.

Configuring Hadoop Security

Many specifics depend on:

- Version of Hadoop and related program
- Type of Kerberos server used (Active directory or MIT)
- Operating system and distribution.

Kerberos Setup using MIT

Setup involves following steps:

- Install CDH 5.
- Verify User Accounts and Groups in CDH 5 Due to Security.
- If you are Using AES-256 Encryption, install the JCE Policy File.
- Create and Deploy the Kerberos Principals and Keytab Files.
- Shut Down the Cluster.
- Enable Hadoop security.

Cont..

- Configure secure HDFS.
- Set Variables for Secure DataNodes.
- Start up the NameNode.
- Start up a DataNode.
- Set the Sticky Bit on HDFS Directories.
- Start up the Secondary NameNode (if used).
- Configure Either MRv1 Security or YARN Security

Active Directory Integration

- Active directory is used to manage user accounts for a Microsoft Windows Network.
- It is very tedious job to create Kerberos principal for every Hadoop user and hence many organization prefer AD.
- Recommended Approach to integrate with AD is
 - Run a local MIT Kerberos KDC
 - Create all service principal in this realm excluding user accounts.
 - Setup one-way cross realm trust between MIT Kerberos and AD, Hadoop's KDC will then accept user account from AD.

Setting up Security using Cloudera Manager

If you do not use Cloudera Manager to implement Hadoop security:

- You must manually create and deploy the Kerberos principals and keytabs on every host in your cluster.
- If you have a large number of hosts, this can be a time-consuming and error-prone process.
- After creating and deploying keytab, you must also manually configure properties in the core-site.xml, hdfs-site.xml, mapred-site.xml files on every host in cluster to enable security in HDFS and MapReduce.

Cont..

- You must manually configure properties in the oozie-site.xml and hue.ini files on cluster in order to enable and configure Hadoop security in Oozie and Hue.

Cloudera Manager enables you to automate all of those manual tasks!!

Cont...

Cloudera Manager can:

- Automatically create and deploy keytab file for the hdfs user and a keytab for the mapred user on every host in your cluster.
- Creates keytab files for oozie and hue users on selected hosts.
- Automatically configures appropriate properties in all the configuration files.
- Once all the appropriate configuration changes have been made, it will automatically start all the Hadoop daemons.

Other Security Concepts

- Sentry is the next step in enterprise-grade big data security and delivers fine-grained authorization to data stored in Apache Hadoop.
- Open Source, Fine-Grained Access Control for Apache Hive and Cloudera Impala
- Sentry introduces the granularity required to secure access to data for the majority of SQL and BI tools.

Cont..

- Role-Based Administration-

Database administrators can unlock key role-based access control (RBAC) requirements and define what users and applications can do with data within a server, database, table, or view.

- Data Classification-

Content producers and owners can intersperse sensitive data with non-sensitive data in the same data set.

Cont..

- Improved Regulatory Compliance-

Business teams can leverage the power of Hadoop while aligning with regulatory mandates like HIPAA, SOX, and PCI.

- Expanded User Base-

Extends the power of Hadoop and makes it suitable for new industries, organizations, and enterprise usage.

Key features of Sentry

- Fine-grained authorization for Hive and Impala
- Specify security for SERVER, DATABASE, TABLE, and VIEW
- SELECT privileges on VIEW, TABLE
- INSERT privilege on TABLE
- TRANSFORM privilege on SERVER
- ALL privilege on SERVER, DATABASE, TABLE, and VIEW

Cont..

- ALL privilege needed to create and modify schema within scope
- Separate authorization policies per database/schema
- Supported in HiveServer2 and Impala 1.1; available with CDH 4.3
- Supports current Hive metastore architecture
- 100% Apache licensed, 100% open source

Benefits of Sentry

- Precise Data Access

Right resources have the proper and relevant permissions to data,

- Easier Administration

Simplify administration by granting sets of permissions based on functional roles within a Hive or Impala database.

- Compliance Assurance

Ensures usage and data compliance for regulation and governance policies.

Cont..

- Broad Usage

foundations of concurrency, authentication, and authorization provided by Hive, Impala, and Sentry.

- Multi-User Application

Build multi-user applications on top of Hive and Impala by segregating access to data sets

- Reuse and Extensibility

Build on existing systems like the Hive metastore

Quick Overview

- Kerberos provides strong authentication for both clients and servers.
- Kerberos server is neither part of nor provided by Hadoop.
- Hadoop does not provide data encryption on disk.
- Apache Sentry provides role based security.



Managing Resources

We will cover following:

- Configuring cgroups with Static Service Pools
- The Fair Scheduler
- Configuring Dynamic Resource Pools
- YARN Memory and CPU Settings
- Impala Query Scheduling

Resource Management

Resource management helps ensure predictable behavior by defining the impact of different services on cluster resources.

The goals of resource management features are to:

- Guarantee completion in a reasonable time frame for critical workloads.
- Support reasonable cluster scheduling between groups of users based on fair allocation of resources per group.
- Prevent users from depriving other users access to the cluster.

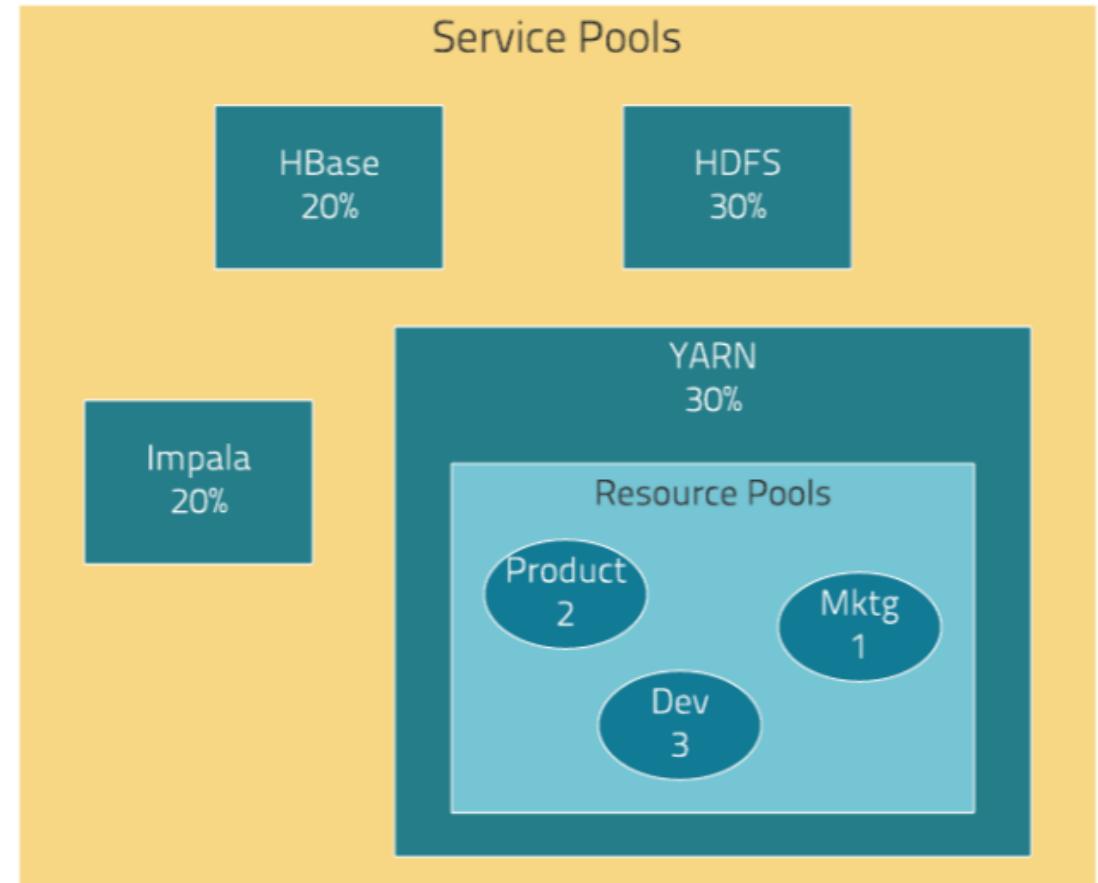
Features: Static Allocation

- Cloudera Manager 4 introduced the ability to partition resources.
- Allows you to set configuration properties that were enforced by Linux control groups (Linux cgroups).
- In Cloudera Manager 5, the ability to statically allocate resources using cgroups is configurable through a single static service pool wizard.
- Allows you to allocate services a percentage of total resources and the wizard configures the cgroups.

Static Service Pool

Static service pools isolate the services in your cluster from one another, so that load on one service has a bounded impact on other services.

Services are allocated a static percentage of total resources—CPU, memory, and I/O weight—which are not shared with other services



Cont..

- Static service pools are implemented per role group within a cluster, using Linux control groups (cgroups) and cooperative memory limits (for example, Java maximum heap sizes).
- Static service pools can be used to control access to resources by HBase, HDFS, Impala, MapReduce, Solr, Spark, YARN, and add-on services.
- Static service pools are not enabled by default.

Schedulers

A scheduler is responsible for deciding which tasks get to run and where and when to run them.

The MapReduce and YARN computation frameworks support the following schedulers:

- FIFO – allocates resources based on arrival time.
- Fair – allocates resources to weighted pools, with fair sharing within each pool.
- Capacity – allocated resources to pool, with FIFO scheduling within each pool.

Fair Scheduler

- The Fair Scheduler controls how resources are allocated to **pools** (or **queues**) and how jobs are assigned to pools.
- Jobs can also be explicitly submitted to pools; to submit an job to a specific pool, you specify the mapreduce.job.queuename property.
- Fair Scheduler configuration is maintained in two files: yarn-site.xml and fair-scheduler.xml

Fair Scheduler Preemption

- Enable the Fair Scheduler to preempt applications in other pools if a pool's fair or minimum share is not met for some period of time.
- When you create a pool, you can specify whether preemption is allowed and whether a specific pool can be pre-empted.
- Fair scheduler preemption is controlled by several properties.

Enable Preemption

- Select **Clusters > Cluster name > Dynamic Resource Pool Configuration.**
- Click **Default Settings.**
- Click the **Enable Fair Scheduler Preemption** link
- Select the **ResourceManager Default Group** checkbox.
- Click **Save Changes** to commit the changes.
- Click **Restart Stale Services.**
- Click **Restart Now.**
- Click **Finish.**

Enable Preemption for a Pool

- Select **Clusters > Cluster name > Dynamic Resource Pool Configuration**
- In a pool row, click **Edit**.
- Click the **Preemption** tab.
- Select or clear **Allow Preemption From**.
- Click **Save**.
- Click **Refresh Dynamic Resource Pools**.

Dynamic Resource Pools

- A **dynamic resource pool** is a named configuration of resources and a policy for scheduling the resources among YARN applications and Impala queries running in the pool.
- Dynamic resource pools allow you to schedule and allocate resources to YARN applications and Impala queries based on a user's access to specific pools and the resources available to those pools.
- Dynamic resource pools have ACLs that restrict who can submit work to and administer them.

Cont..

- A **configuration set** defines the allocation of resources across pools that may be active at a given time.
- A **scheduling rule** defines when a configuration set is active.
- Resource pools can be nested, with sub-pools restricted by the settings of their parent pool.

YARN Memory and CPU Settings

Cloudera recommends following estimated value for YARN resource allocation:

- 10-20% of RAM for Linux and its daemon services
- At least 16 GB RAM for an Impalad process
- No more than 12-16 GB RAM for an HBase RegionServer process
- For vcore, consider the number of concurrent processes or tasks each service runs.
- For the operating system, start with a count of two.

Example: Resource Demand for worker node with 24 vcores and 256GB of memory

Service	vcores	Memory (MB)
Operating system	2	
YARN NodeManager	1	
HDFS DataNode	1	1,024
Impala Daemon	1	16,348
HBase RegionServer	0	0
Solr Server	0	0
Cloudera Manager agent	1	1,024
Task overhead	0	52,429
YARN containers	18	137,830
Total	24	262,144

Cont..

- Configure YARN to use the remaining resources for its supervisory processes and task containers.

Property	Description	Default
yarn.nodemanager.resource.cpu-vcores	Number of virtual CPU cores that can be allocated for containers.	8
yarn.nodemanager.resource.memory-mb	Amount of physical memory, in MB, that can be allocated for containers.	8 GB

Cont..

The number of vcores allocated to the NodeManager should be the lesser of either:

- (total vcores) – (number of vcores reserved for non-YARN use), or
- $2 \times (\text{number of physical disks used for DataNode storage})$

The amount of RAM allotted to a NodeManager for spawning containers should be the difference between a node's physical RAM minus all non-YARN memory demand.

Sizing Resource Manager

Property	Description	Default
yarn.scheduler.minimum-allocation-vcores	The smallest number of virtual CPU cores that can be requested for a container.	1
yarn.scheduler.maximum-allocation-vcores	The largest number of virtual CPU cores that can be requested for a container.	32
yarn.scheduler.increment-allocation-vcores	If using the Fair Scheduler, virtual core requests are rounded up to the nearest multiple of this number.	1
yarn.scheduler.minimum-allocation-mb	The smallest amount of physical memory, in MB, that can be requested for a container.	1 GB
yarn.scheduler.maximum-allocation-mb	The largest amount of physical memory, in MB, that can be requested for a container.	64 GB
yarn.scheduler.increment-allocation-mb	If using the Fair Scheduler, memory requests are rounded up to the nearest multiple of this number.	512 MB

Configuring YARN Settings

Property	Description	Default
mapreduce.map.memory.mb	The amount of physical memory, in MB, allocated for each map task of a job.	1 GB
mapreduce.map.java.opts.max.heap	The maximum Java heap size, in bytes, of the map processes.	800 MB
mapreduce.map.cpu.vcores	The number of virtual CPU cores allocated for each map task of a job.	1
mapreduce.reduce.memory.mb	The amount of physical memory, in MB, allocated for each reduce task of a job.	1 GB
mapreduce.reduce.java.opts.max.heap	The maximum Java heap size, in bytes, of the reduce processes.	800 MB
mapreduce.reduce.cpu.vcores	The number of virtual CPU cores for each reduce task of a job.	1
yarn.app.mapreduce.am.resource.mb	The physical memory requirement, in MB, for the ApplicationMaster.	1 GB
ApplicationMaster Java maximum heap size	The maximum heap size, in bytes, of the Java MapReduce ApplicationMaster. Exposed in Cloudera Manager as part of the YARN service configuration. This value is folded into the property <code>yarn.app.mapreduce.am.command-opts</code> .	800 MB
yarn.app.mapreduce.am.resource.cpu-vcores	The virtual CPU cores requirement for the ApplicationMaster.	1

Defining Containers

With YARN worker resources configured, you can determine how many containers best support a MapReduce application, based on job type and system resources.

Container Formula

Property	Value
mapreduce.job.maps	$\text{MIN}(\text{yarn.nodemanager.resource.memory-mb} / \text{mapreduce.map.memory.mb}, \text{yarn.nodemanager.resource.cpu-vcores} / \text{mapreduce.map.cpu.vcores}, \text{number of physical drives} \times \text{workload factor}) \times \text{number of worker nodes}$
mapreduce.job.reduces	$\text{MIN}(\text{yarn.nodemanager.resource.memory-mb} / \text{mapreduce.reduce.memory.mb}, \text{yarn.nodemanager.resource.cpu-vcores} / \text{mapreduce.reduce.cpu.vcores}, \# \text{ of physical drives} \times \text{workload factor}) \times \# \text{ of worker nodes}$

Impala Resource Management

- Impala supports two types of resource management: independent and integrated.
- Independent resource management is supported for CDH 4 and CDH 5 and is implemented by admission control.
- Integrated resource management is supported for CDH 5 and is implemented by YARN and Llama.

Admission Control

- Admission control is an Impala feature that imposes limits on
 - concurrent SQL queries
 - the maximum number of queries that are queued (waiting)
 - the amount of time query might wait before returning with an error
- Avoids resource usage spikes and out-of-memory conditions on busy CDH clusters.

Llama

- Impala estimates the resources required by the query on each host of the cluster, and requests the resources from YARN.
- Requests from Impala to YARN go through an intermediary service called Llama.
- When the resource requests are granted, Impala starts the query and places all relevant execution threads into the cgroup containers and sets up the memory limit on each host.

Cont..

- During query processing, as the need for additional resources arises, Llama can "expand" already-requested resources, to avoid over-allocating at the start of the query.
- After a query is finished, Llama caches the resources (for example, leaving memory allocated).
- Caching mechanism avoids the latency involved in making a whole new set of resource requests for each query.

Impala Query Scheduling

- The admission control system is decentralized, embedded in each Impala daemon.
- Each Impala daemon makes its own decisions whether to allow each query to run immediately or to queue it for a less-busy time.
- The limit on the number of concurrent queries is a "soft" one.
- To achieve high throughput, Impala makes quick decisions at the host level about which queued queries to dispatch.

Cont..

- To avoid a large backlog of queued requests, set an upper limit on the size of the queue for queries that are delayed.
- When the number of queued queries exceeds this limit, further queries are cancelled rather than being queued.
- Configure a timeout period, after which queued queries are cancelled, to avoid indefinite waits

Cluster Maintenance

We will cover following topics:

- Checking HDFS Status
- Copying Data Between Clusters
- Adding and Removing Cluster Nodes
- Rebalancing the Cluster
- Directory Snapshots
- Cluster Upgrading

Checking HDFS Status

- Use hdfs fsck command to detect any missing or corrupt data blocks, don't attempt to repair error.
- hdfs fsck can also be used to list all files, block location and all racks

```
hdfs fsck /
```

```
hdfs fsck / -files
```

```
hdfs fsck / -files -blocks
```

```
hdfs fsck / -files –blocks -location
```

```
hdfs fsck / -files –blocks –location -racks
```

Cont..

- While running hdfs fsck command with –move option will move the corrupted files to /lost+found directory.
- Corrupted file refers to a file where all replicas of block are missing.
- Also –delete option will delete the corrupted files.

Cont..

hdfs dfsadmin command provides number of administrative features:

- List information about HDFS on a per-datanode basis

```
hdfs dfsadmin -report
```

- Re-read the dfs.hosts and dfs.hosts.exclude files

```
hdfs dfsadmin -refreshNodes
```

dfsadmin Usage

- With dfsadmin command, we can manually set the filesystem to ‘safe mode’

```
hdfs dfsadmin –safemode enter  
hdfs dfsadmin –safemode leave
```

- The Namenode starts up in safe mode and has following effects:
 - No changes can be made to metadata i.e. becomes read-only
 - Does not replicate or delete blocks.
 - Leaves the safe mode when the configured minimum percentage of blocks satisfy the minimum replication condition.

Cont..

- It can also block until the safe mode is exited

```
hdfs dfsadmin –safemode wait
```

- While in safe mode, can also save Namenode metadata to disk and resets the edit log

```
hdfs dfsadmin –saveNamespace
```

Copying Data Between Clusters

- Hadoop clusters are loaded with terabytes of data. It will take forever to transfer terabytes of data from one cluster to another.
- Distributed or parallel copying of data can be a good solution for this and that is what Distcp does.
- Distcp runs map reduce job to transfer your data from one cluster to another.
- It expands a list of files and directories into map tasks, each of which copies a partition of the files specified in the source list.

Distcp Command

- The distributed copy command, distcp is a general utility for copying large data sets between distributed filesystems within and across clusters.
- The distcp command submits a regular MapReduce job that performs a file-by-file copy.
- Files which are copied previously will be skipped
- The only check for duplicate files is that the file's name, size and checksum are identical.

Distcp Examples

- Run ‘hadoop distcp’ command to get all the built-in options.
- To copy data from one cluster to another

```
hadoop distcp hdfs://cluster1_nn:8020/path/to/src  
hdfs://cluster2_nn:8020/path/to/dest
```

- To copy data with same cluster

```
hadoop distcp /path/to/src /path/to/dest
```

HFTP Protocol

- The HFTP protocol allows you to use FTP resources in an HTTP request.
- When copying with distcp across different versions of CDH, use hftp:// for the source file system and hdfs:// for the destination file system, run distcp from the destination cluster.
- The default port for HFTP is 50070 and the default port for HDFS is 8020.

Example

- Source URI:

hftp://namenode-location:50070/basePath

- Destination URI:

hdfs://nameservice-id/basePath or hdfs://namenode-location

basePath in both examples refers to the directory you want to copy

Note

- HFTP is a read-only protocol and can only be used for the source cluster, not the destination.
- HFTP cannot be used when copying with distcp from an insecure cluster to a secure cluster.

Adding and Removing Cluster Nodes

To add nodes to the cluster:

1. Add the names of nodes to the ‘include’ file, the file referred by dfs.hosts (and yarn.resourcemanager.nodes.include-path if that has been used)
2. Update your rack awareness script with new information.
3. Update the NameNode with this new information

```
hdfs dfsadmin -refreshNodes
```

Cont..

4. Update the ResourceManager with new information

```
hdfs rmadmin -refreshNodes
```

5. Start the new DataNodes and NodeManagers.
6. Check that the new DataNodes and NodeManagers appear in the Web UI

Cont..

To remove nodes from the Cluster

1. Add the names of nodes to the ‘exclude’ file, the file referred by dfs.hosts.exclude (and yarn.resourcemanager.nodes.exclude-path if that has been used)
2. Update the ResourceManager with the revised set of Nodes

```
yarn rmadmin -refreshNodes
```

Cont..

3. Update the NameNode with this new information

```
hdfs dfsadmin -refreshNodes
```

The NameNode UI will show the admin state change to “Decommission In Progress” for affected DataNodes.

When all the DataNodes report their state as “Decommissioned”, all blocks will have been replicated somewhere else.

Cont..

4. Shutdown the decommissioned nodes
5. Remove the nodes from the ‘include’ and ‘exclude’ files and update the Namenode.

Rebalancing the Cluster

- HDFS data might not always be placed uniformly across DataNodes.
- Common reason is addition of new DataNodes to an existing cluster.
- HDFS provides a balancer utility that analyzes block placement and balances data across the DataNodes.
- It moves blocks until the cluster is deemed to be balanced, which means that the utilization of every DataNode differs from the utilization of the cluster by no more than a given threshold percentage.

Cont..

- The balancer does not balance between individual volumes on a single DataNode.
- The HDFS balancer utility is implemented by the Balancer role.
- The Balancer role usually shows a health of **None** on the HDFS Instances tab because it does not run continuously.

Cont..

- A node is under-utilized if its utilization is less than (average utilization – threshold)
- A node is over-utilized if its utilization is more than (average utilization + threshold)

Running the Balancer

- Go to the HDFS service.
- Select **Actions > Rebalance**.
- Click **Rebalance** that appears in the next screen to confirm.
- If you see a **Finished** status, the Balancer ran successfully.

Configuring the Balancer Threshold

- The Balancer has a default threshold of 10%, which ensures that disk usage on each DataNode differs from the overall usage in the cluster by no more than 10%.
- For example, if overall usage across all the DataNodes in the cluster is 40% of the cluster's total disk-storage capacity, the script ensures that DataNode disk usage is between 30% and 50% of the DataNode disk-storage capacity.

Cont..

To change the threshold:

- Go to the HDFS service.
- Click the **Configuration** tab.
- Expand the **Balancer Default Group** category.
- Set the **Rebalancing Threshold** property.
- Click **Save Changes** to commit the changes.

Directory Snapshots

- HDFS Snapshots are read-only point-in-time copies of the file system.
- Snapshots can be taken on a subtree of the file system or the entire file system.
- Some common use cases of snapshots are
 1. Data backup
 2. Protection against user errors and
 3. Disaster recovery

Cont..

- Snapshots can be taken on any directory once the directory has been set as snapshottable.
- A snapshottable directory is able to accommodate 65,536 simultaneous snapshots.
- There is no limit on the number of snapshottable directories.
- Administrators may set any directory to be snapshottable.
- If there are snapshots in a snapshottable directory, the directory can be neither deleted nor renamed before all the snapshots are deleted.

Cont..

- Nested snapshottable directories are currently not allowed.
- In other words, a directory cannot be set to snapshottable if one of its ancestors/descendants is a snapshottable directory.

Allow/Disallow Snapshots

- Allowing snapshots of a directory to be created.

```
hdfs dfsadmin –allowSnapshot <path>
```

- Disallow snapshot of a directory

```
hdfs dfsadmin –disallowSnapshot <path>
```

Create/Delete Snapshots

- Create a snapshot of a snapshottable directory.

```
hdfs dfs –createSnapshot <path> [<snapshotName>]
```

- Delete a snapshot of from a snapshottable directory.

```
hdfs dfs –deleteSnapshot <path> [<snapshotName>]
```

Managing HDFS Directory Snapshots

From the HDFS Browse tab you can:

- Designate HDFS directories to be "snapshottable" so snapshots can be created for those directories.
- Initiate immediate (unscheduled) snapshots of a table.
- View the list of saved snapshots currently being maintained. These may include one-off immediate snapshots, as well as scheduled policy-based snapshots.
- Delete a saved snapshot.
- Restore an HDFS directory or file from a saved snapshot.
- Restore an HDFS directory or file from a saved snapshot to a new directory or file (Restore As)

Enabling HDFS Snapshots

HDFS directories must be enabled for snapshots in order for snapshots to be created. You cannot specify a directory as part of a snapshot policy unless it has been enabled for snapshotting.

To enable a HDFS directory for snapshots:

- From the **Clusters** tab, select your CDH 5 HDFS service.
- Go to the **Browse** tab.
- Verify the Snapshottable Path and click **Enable Snapshots**.
- When the command has finished, a **Take Snapshot** button appears. You may need to refresh the page to see the new state.

Upgrade Instructions

General Steps:

1. Stop the YARN cluster
2. Stop the HDFS cluster
3. Install the New version of Hadoop
4. Start the NameNode with the `-upgrade` option
5. Monitor the HDFS cluster until it reports that the upgrade is complete
6. Start the YARN cluster

Cont..

- If you encounter any problem, you can rollback an unfinalized upgrade by stopping the cluster and restarting with –rollback option
- Once the upgraded cluster is running for few days with no problems, finalize the upgrade.

```
hdfs dfsadmin -finalizeUpgrade
```

Cluster Monitoring and Troubleshooting

We will cover following topics:

- Cloudera Manager Monitoring Features
- Monitoring Hadoop Clusters
- Troubleshooting Hadoop Clusters
- Common Misconfigurations

Cloudera Manager Monitoring Features

Monitor Services:

- Service monitoring lets you view the results of health checks at both the service and role instance level.
- Various types of metrics are displayed in charts that help with problem diagnosis.
- Health checks include advice about actions you can take if the health of a component becomes concerning or bad.
- You can also view the history of actions performed on a service or role, and can view an Audit log of configuration changes.

Cont..

Monitor Hosts:

Host monitoring lets you view information pertaining to all the hosts on your cluster like:

- which hosts are up or down
- current resident and virtual memory consumption for a host
- what role instances are running on a host
- which hosts are assigned to different racks, and so on.

Cont..

Monitor Activities:

Activity monitoring lets you see who's running what activities on the cluster, both at the current time and through views of historical activity, and provides many statistics — both in tabular displays and charts — about the resources used by individual jobs.

Cont..

Events:

The Event Server aggregates relevant Hadoop events and makes them available for alerting and for searching, giving you a view into the history of all relevant events that occur cluster-wide.

Cont..

Alerts:

- You can configure Cloudera Manager to generate alerts from a variety of events.
- You can configure thresholds for certain types of events, enable and disable them, and configure alert notifications by email or via SNMP trap for critical events.
- You can also suppress alerts temporarily for individual roles, services, hosts, or even the entire cluster to allow system maintenance/troubleshooting without generating excessive alert traffic.

Cont..

Audit Events:

Audits pages display audit events such as creating a role or service, making configuration revisions for a role or service, decommissioning and recommissioning hosts, and running commands.

Chart Time-Series Data:

Cloudera Manager enables you to search metric data, create charts of the data, group (facet) the data, and save those charts to user-defined views.

Cont..

Logs:

Cloudera Manager provides access to logs in a variety of ways that take into account the current context you are viewing.

For example, when monitoring a service, you can easily click a single link to view the log entries related to that specific service, through the same user interface.

When viewing information about a user's activity, you can easily view the relevant log entries that occurred on the hosts used by the job while the job was running.

Cont..

Reports:

Reports provide an historical view into disk utilization by user, user group, and by directory.

- You can manage your HDFS directories as well, including searching and setting quotas.
- You can also view cluster job activity user, group, or job ID. These reports are aggregated over selected time periods (hourly, daily, weekly, and so on) and can be exported as XLS or CSV files.

Monitoring Hadoop Cluster

- Use monitoring tool to warn you of potential or actual problems on individual machines in the cluster.
- With Cloudera Manager, no extra configuration are required for Hadoop cluster monitoring.
- You can integrate cluster monitoring into many existing monitoring tools:
 - JMX broadcasts
 - Metrics sinks

Things to Monitor

1. Hadoop Daemon
2. Disk and Disk Partition
3. CPU usage
4. Swap on all nodes
5. Network transfer speed
6. HDFS health
7. Log file growth

Performance Metrics To Monitor

Hadoop metrics can be broken down into four broad categories:

- HDFS metrics
- MapReduce counters
- YARN metrics
- ZooKeeper metrics

HDFS Metrics

HDFS metrics can be further decomposed into two categories:

- NameNode metrics
- DataNode metrics

NameNode metrics can be divided into two groups:

- NameNode-emitted metrics
- NameNode JVM metrics

NameNode-emitted Metrics

Metric to alert on:

- CapacityRemaining: the total available capacity remaining across the entire HDFS cluster.
- MissingBlocks: Corrupt or missing blocks can point to an unhealthy cluster.

Cont..

- NumDeadDataNodes: When the NameNode does not hear from a DataNode for 30 seconds, that DataNode is marked as “stale.”

The “stale” node is then marked to “dead” if it fails to communicate with Namenode for next 10mins.

- UnderReplicatedBlocks: Hadoop’s replication factor is configurable on a per-client or per-file basis.

If you see a large, sudden spike in the number of under-replicated blocks, it is likely that a DataNode has died.

NameNode JVM metrics

- NameNode runs in the Java Virtual Machine (JVM), it relies on Java garbage collection processes to free up memory.
- Use the CMS (ConcurrentMarkSweep) garbage collector.
- ConcurrentMarkSweep collections free up unused memory in the old generation of the heap

DataNode metrics

DataNode metrics are host-level metrics specific to a particular DataNode.

Metric to track:

- Remaining disk space
- Number of failed storage volume (dfs.datanode.failed.volume.tolerated)

Collecting HDFS metrics

NameNode and DataNodes emit metrics over an HTTP interface as well as via JMX.

Different ways to collect metrics:

- Collecting NameNode metrics via API
- Collecting DataNode metrics via API
- Collecting HDFS metrics via JMX

NameNode HTTP API

- The NameNode offers a summary of health and performance metrics through an easy-to-use web UI.
- By default, the UI is accessible via port 50070, so point a web browser at:
`http://<namenodehost>:50070`
- To see all the metrics, point your browser to `http://<namenodehost>:50070/jmx` which will result in JSON output.

Cont..

Most of the NameNode metrics can be found under Mbean

Hadoop:name=FSNamesystem, service=Namenode

Hadoop:name=FSNamesystemState, service=Namenode

DataNode HTTP API

- DataNodes expose all of their metrics on port 50075
- A high-level overview of the health of your DataNodes is available in the NameNode dashboard, under the **Datanodes** tab.

MapReduce Counters

The MapReduce framework exposes a number of counters to track statistics on MapReduce job execution.

MapReduce counters can be broken down into four categories:

- job counters
- task counters
- custom counters
- file system counters

Collecting MapReduce Counters

- MapReduce counters provide information on MapReduce task execution, like CPU time and memory used.
- The ResourceManager also exposes all MapReduce counters for each job.
- To access MapReduce counters on your ResourceManager, navigate to <http://<resourcemanagerhost>:8088>

YARN Metrics

YARN metrics can be grouped into three distinct categories:

- Cluster metrics
- Application metrics
- NodeManager metrics

Cluster Metrics

Provide a high-level view of YARN application execution.

Metric to track:

- Number of unhealthy nodes
- Number of currently active nodes
- Number of lost nodes
- Number of failed applications
- Total amount of memory

Application Metrics

Provide detailed information on the execution of individual YARN applications.

Measures progress which gives a real-time execution window of a YARN application.

NodeManager Metrics

- NodeManager metrics provide resource information at the individual node level.
- Number of containers that failed to launch on particular NodeManager.
- If the NodeManager's disk is full, any container it launches will fail. Similarly, if the NodeManager's heap is set too low, containers won't launch.

Collecting YARN metrics

- YARN metrics are also exposed via an HTTP API.
- YARN exposes all of its metrics on port 8088, via the jmx endpoint.

Third-Party Tools

- Cloudera Manager, offer users a unified platform for Hadoop administration and management.
- Provides tools for the collection and visualization of Hadoop metrics, as well as tools for common troubleshooting tasks.
- To connect to the Cloudera Manager dashboard, point your browser to <ClouderaHost>:7180 and login with default user admin and password admin.

ZooKeeper Metrics

It is responsible for ensuring the availability of the HDFS NameNode and YARN's ResourceManager.

ZooKeeper exposes metrics via MBeans as well as through a command line interface.

Metrics to track:

- Number of active followers
- Amount of time it takes to respond to a client request
- Number of clients connected to ZooKeeper

Collecting ZooKeeper metrics

- ZooKeeper randomizes its JMX port on each run.
- Using JMX with an MBean browser like JConsole or Jmxterm, you can collect all of the metrics.

Troubleshooting Hadoop Cluster

Approach for Troubleshooting:

1. Validate environment information including Version installed.
2. Validate Hadoop Cluster is healthy and running.
3. Identify any misconfiguration on the cluster.
4. Identify the User(s) under which you will access the HDFS.
5. SSH into cluster as above identified USER(s)

Cont..

6. Issue basic commands on HDFS to verify that you have the authority to execute them.
7. Verify the ports are open on which Hadoop services are running.
8. Validate any network issues, delays and naming conventions.
9. Run sample word-count program to validate MapReduce functionality.
10. Ensure availability of sufficient resources to avoid exhaustion.

Common Misconfigurations

- 35% of Cloudera support tickets are due to misconfigurations.
- Most common error are:
 - Map/Reduce Task out of memory
 - Not able to Place Enough Replicas
 - Removal of file

Java Heap Space Out of Memory

Map/Reduce job fails due to:

- Bad coding of Mapper or Reducer
- Map/Reduce task run out of memory
- Memory leak in the code

Cont..

To fix above error, follow below:

- Increase size of RAM allocated in mapreduce.map.java.opts and mapreduce.reduce.java.opts.
- Ensure mapreduce.task.io.sort.mb is smaller than RAM allocated in mapreduce.map.java.opts
- Recode Mapper or Reducer program.

Not Able To Place Enough Replicas

Error:WARN org.apache.hadoop.hdfs.server.namenode.FSNamesystem: Not able to place enough replicas.

Reason:

- Fewer DataNodes available than the replication factor of the blocks
- DataNodes do not have enough transfer threads
- Default is 256 threads to manage connections
- Note: yes, the configuration option is misspelled!

Cont..

To fix the error:

- Increase `dfs.datanode.max.transfer.threads` to 4096
- Check replication factor

Removal of File

- User cannot recover an accidentally deleted file from trash.
- Check whether trash is enabled or not.
- Check the trash interval.
- Set `fs.trash.interval` to 1440 or higher value.

Thank You!