Hadoop Basics

Nagabhushan

Agenda – Day 1

- Introduction to Big Data and Hadoop
- History of Hadoop & Use Cases
- HDFS Storage Architecture
- Hadoop Setup

What is Big Data?

- *3 Vs of Big Data*
  - *Velocity ➔ Speed*
  - *Variety ➔ Different forms of data*
  - *Volume ➔ Size of data*

  - *Hadoop's 4th V ➔ VALUE*

  - *How to store Big Data? ➔ HDFS –*
  - *How to process Big Data? ➔ MapReduce (1.x) / YARN (2.x)*

Hadoop Core

Data Measurement Scale

1 KB	*Kilobyte*	*1000*
1 MB	*Megabyte*	*1000000*
1 GB	*Gigabyte*	*1000000000*
1 TB	*Terabyte*	*1000000000000*
1 PB	*Petabyte*	*1000000000000000*
1 EB	*Exabyte*	*1000000000000000000*
1 ZB	*Zetabyte*	*1000000000000000000000 X 5*
1 YB	*Yotabyte*	*1000000000000000000000000*

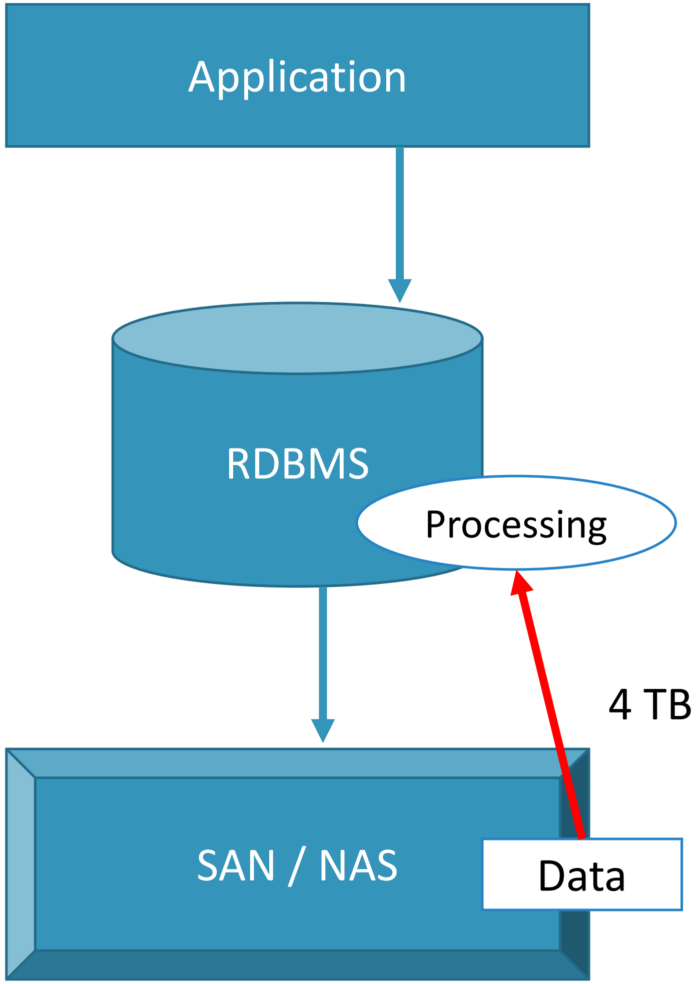
Traditional System

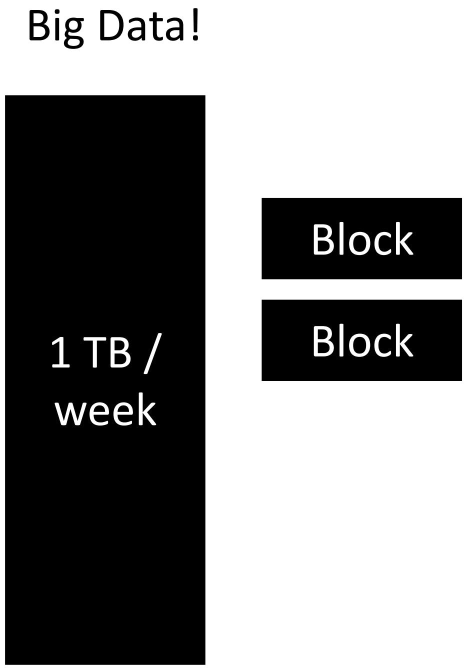
Moving Data to Processing

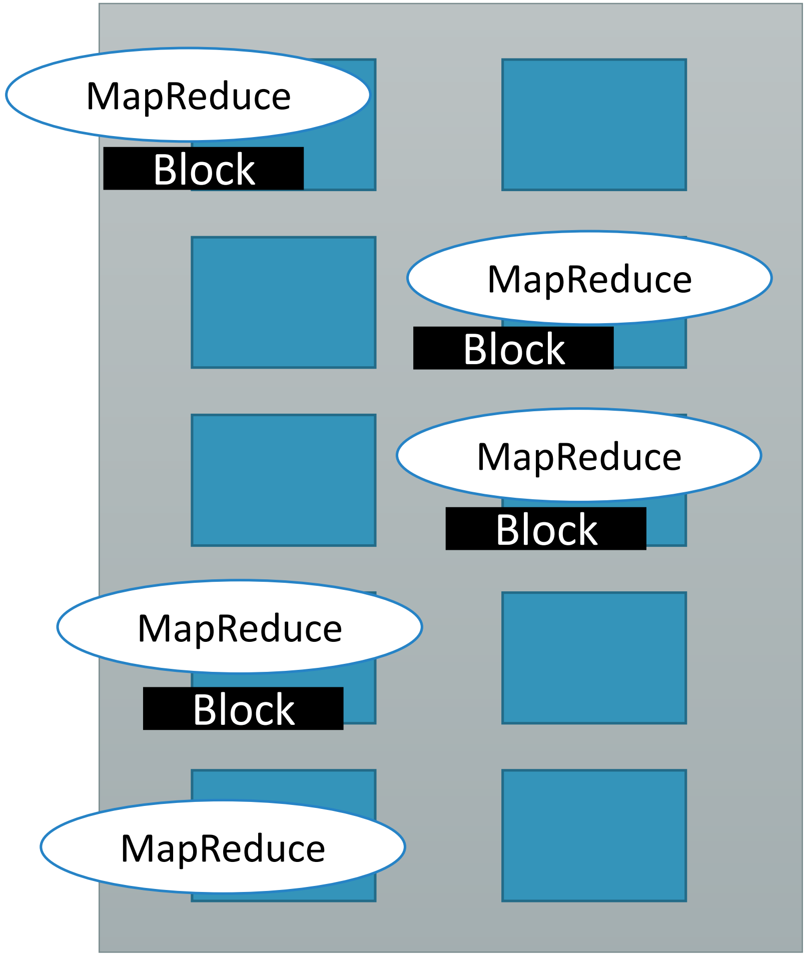
8 cores
64 GB RAM
240 TB Storage

Hadoop System

Moving Processing to Data







Features of Hadoop

- Commodity Hardware
- Open Source
- Fault Tolerance
- Distributed Storage
- Read Only File System ➔ Parallel Processing (free)
- Horizontal Scaling
- Data Locality → Move Processing to Data
- Simplified Programming

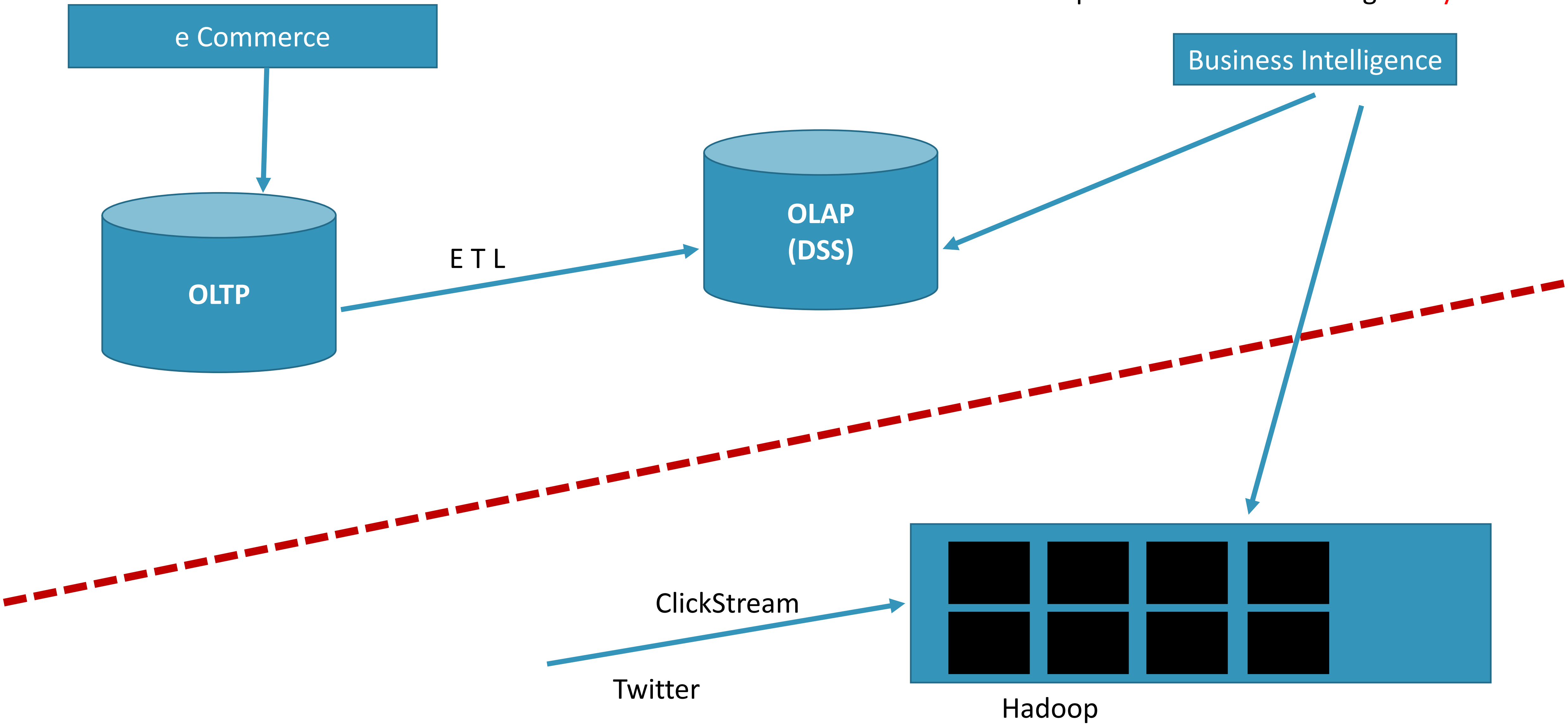
Limitations of Hadoop

- WORM → Write Once Read Many
- Sequential Access to Data – No Random Reads

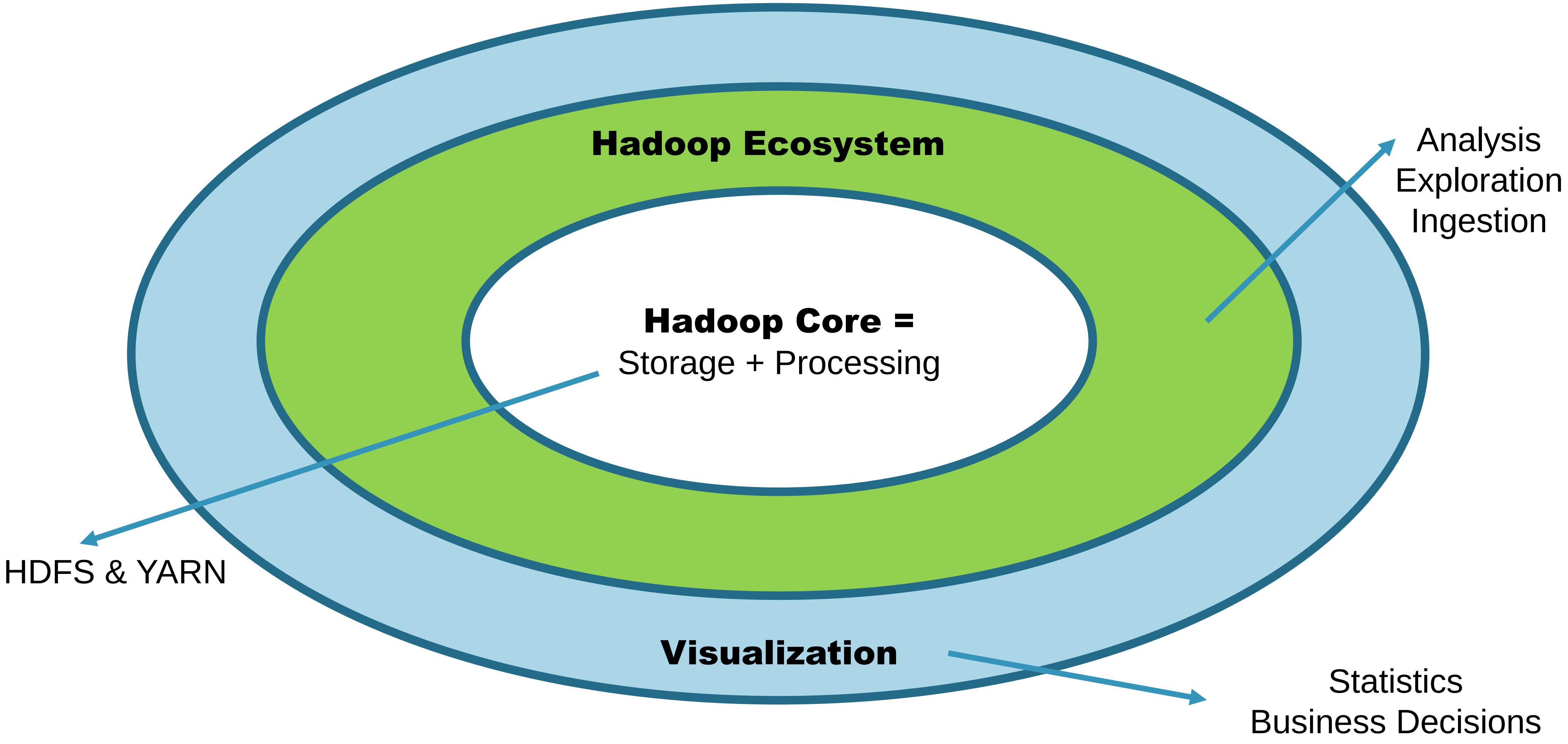
A simple use case!

Business Owner

What products are NOT selling? Why?



Hadoop Overview – Core & Ecosystem



Hadoop Ecosystem

Hadoop Core =
Storage + Processing

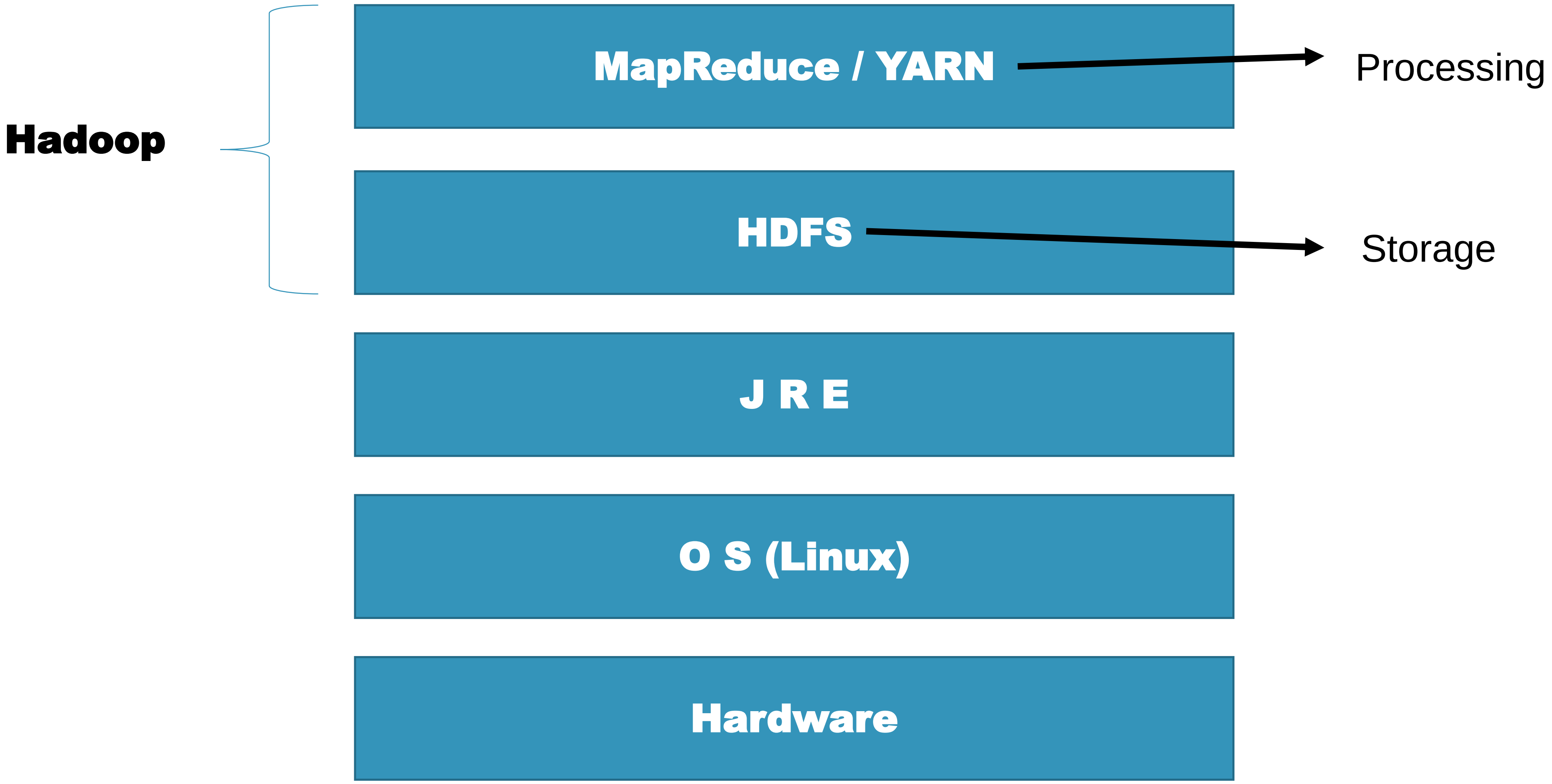
Visualization

HDFS & YARN

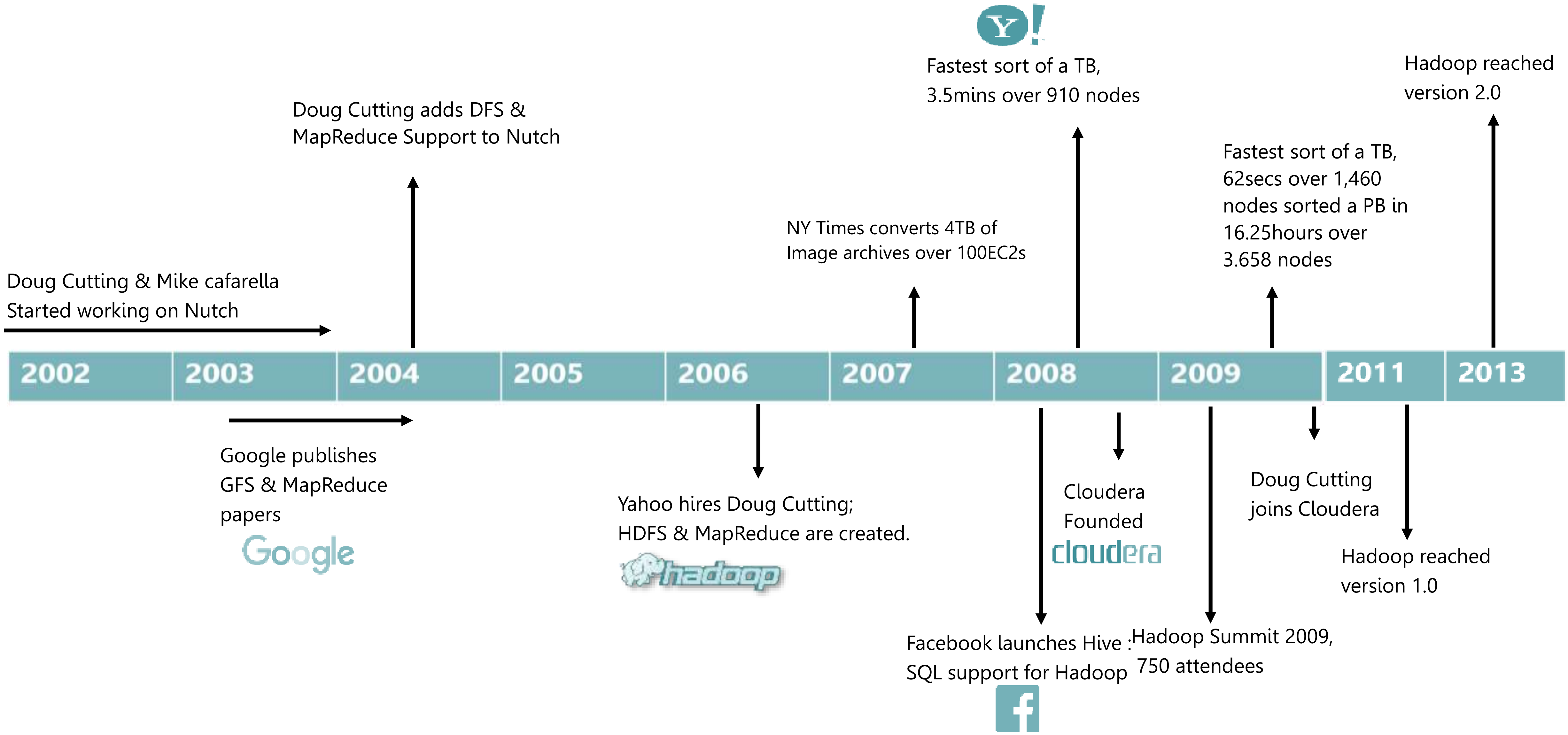
Analysis
Exploration
Ingestion

Statistics
Business Decisions

Hadoop’s Physical Architecture



History of Hadoop



2002 2003 2004 2005 2006 2007 2008 2009 2011 2013

Doug Cutting & Mike cafarella
Started working on Nutch

Google publishes
GFS & MapReduce
papers

Doug Cutting adds DFS &
MapReduce Support to Nutch

Yahoo hires Doug Cutting;
HDFS & MapReduce are created.

NY Times converts 4TB of
Image archives over 100EC2s

Fastest sort of a TB,
3.5mins over 910 nodes

Facebook launches Hive :
SQL support for Hadoop

Cloudera
Founded

Hadoop Summit 2009,
750 attendees

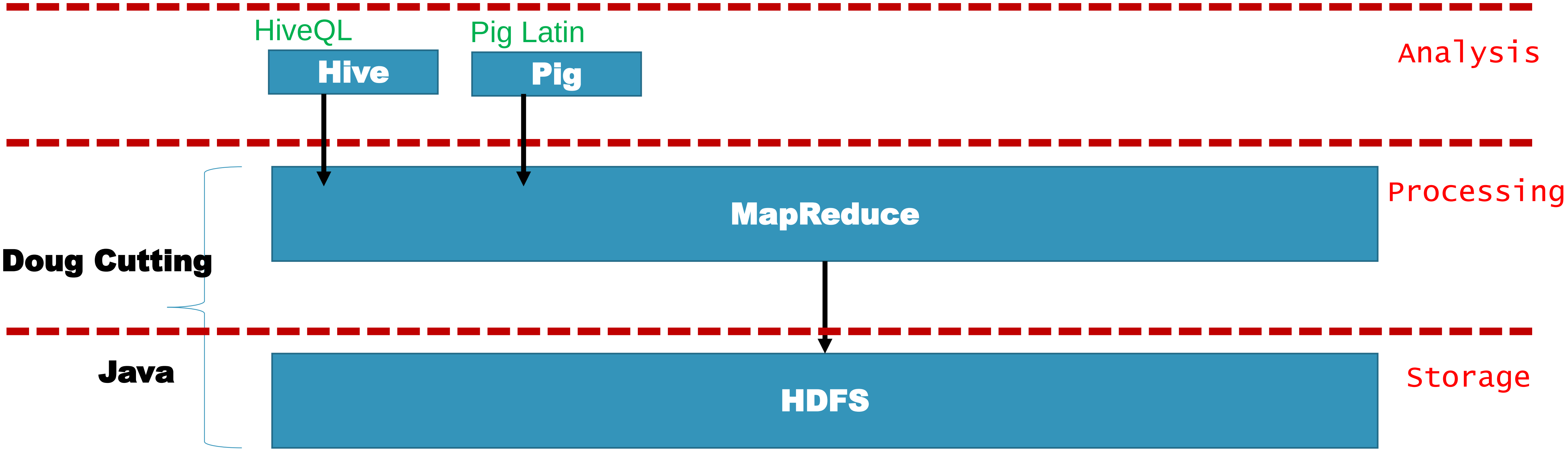
Fastest sort of a TB,
62secs over 1,460
nodes sorted a PB in
16.25hours over
3.658 nodes

Doug Cutting
joins Cloudera

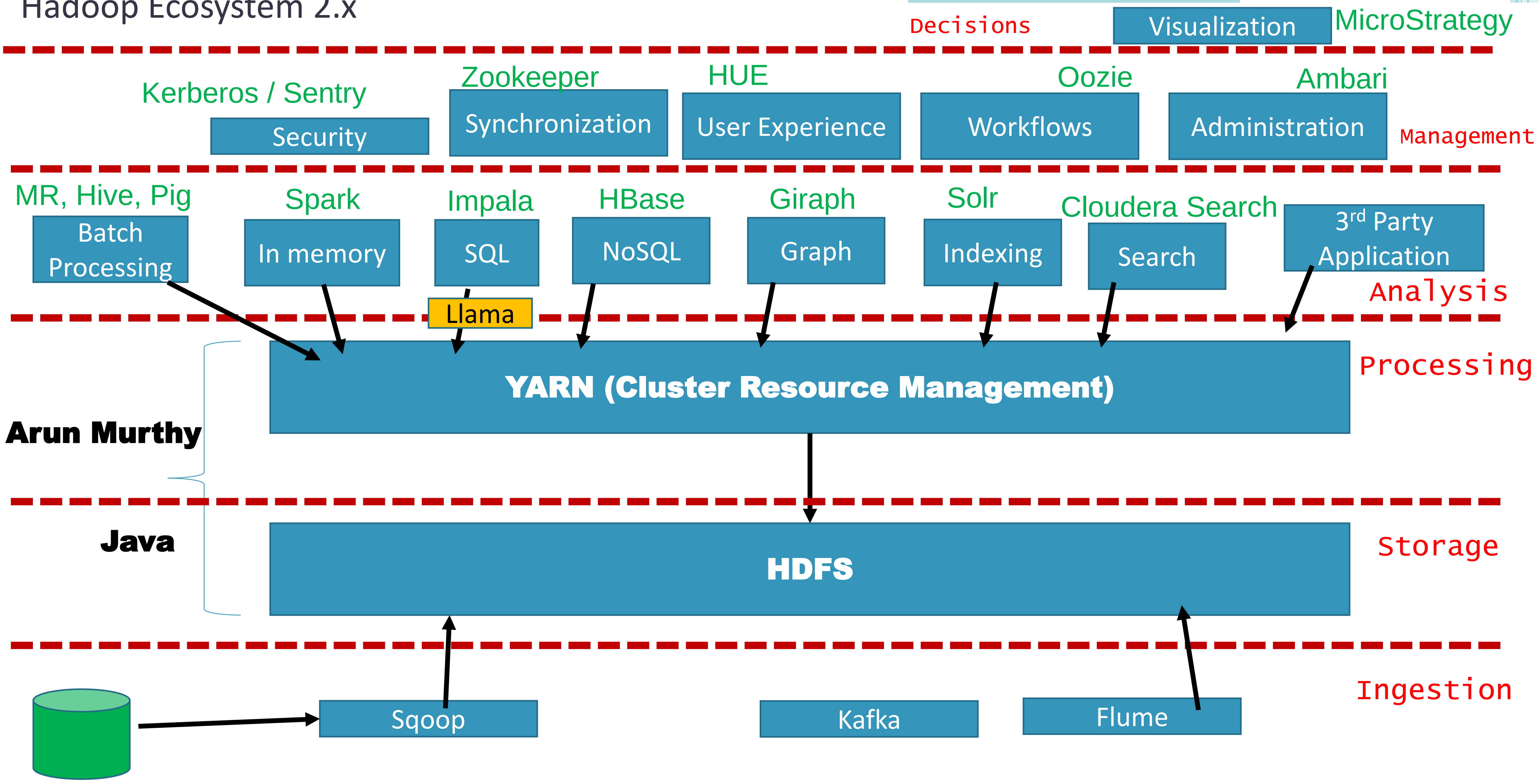
Hadoop reached
version 1.0

Hadoop reached
version 2.0

Hadoop Ecosystem 1.x



Hadoop Ecosystem 2.x



Decisions

Visualization

MicroStrategy

Kerberos / Sentry

Security

Zookeeper

Synchronization

HUE

User Experience

Oozie

Workflows

Ambari

Administration

Management

MR, Hive, Pig

Batch Processing

Spark

In memory

Impala

SQL

Llama

HBase

NoSQL

Giraph

Graph

Solr

Indexing

Cloudera Search

Search

3rd Party Application

Analysis

YARN (Cluster Resource Management)

Processing

Arun Murthy

Java

HDFS

Storage

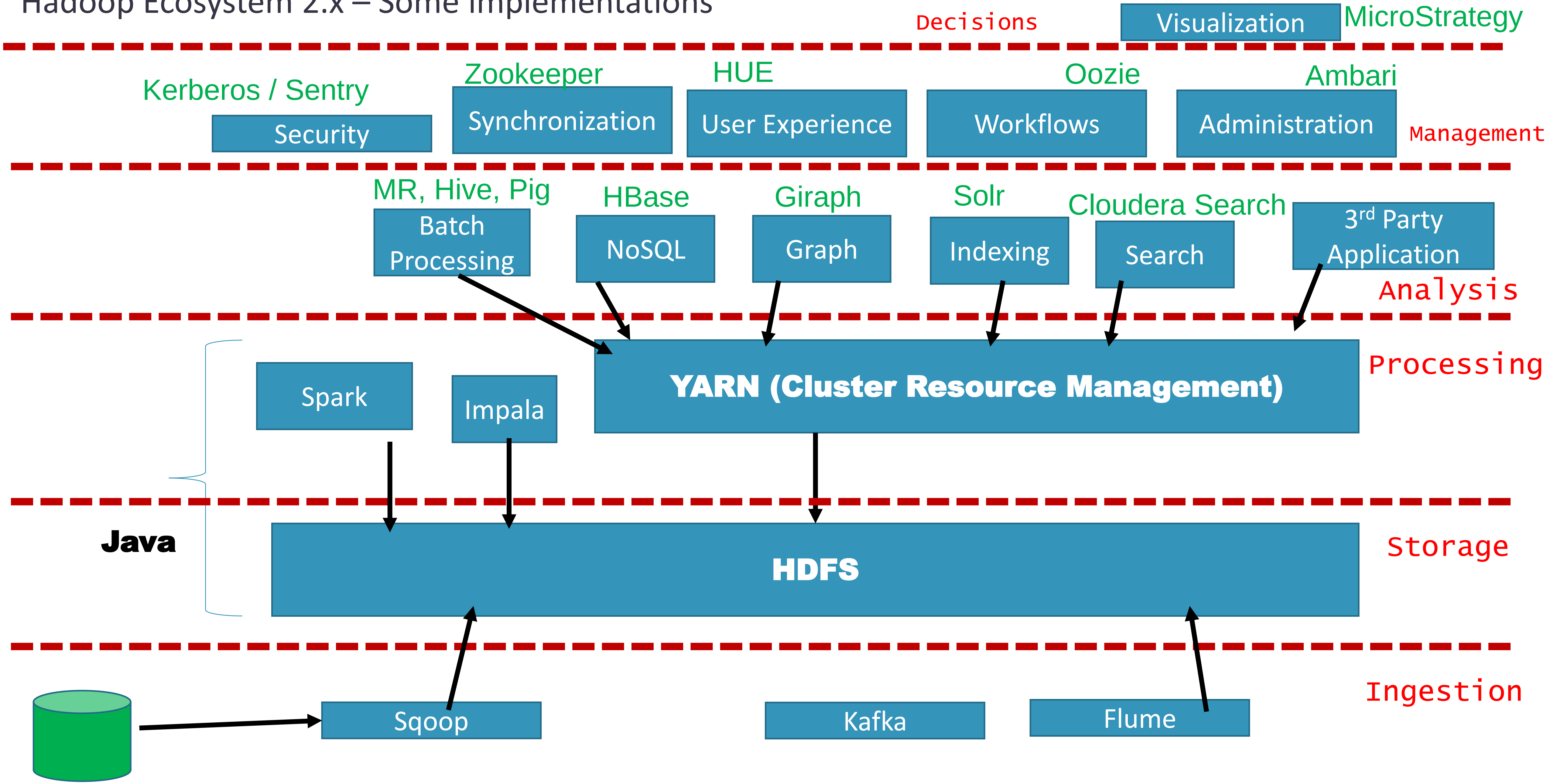
Sqoop

Kafka

Flume

Ingestion

Hadoop Ecosystem 2.x – Some implementations



Commercial Distributions

- Cloudera http://www.cloudera.com/
- Hortonworks http://hortonworks.com/
- MAPR https://www.mapr.com/

HDFS Daemons

Master – Slave Architecture

NameNode – Metadata is in 2 file

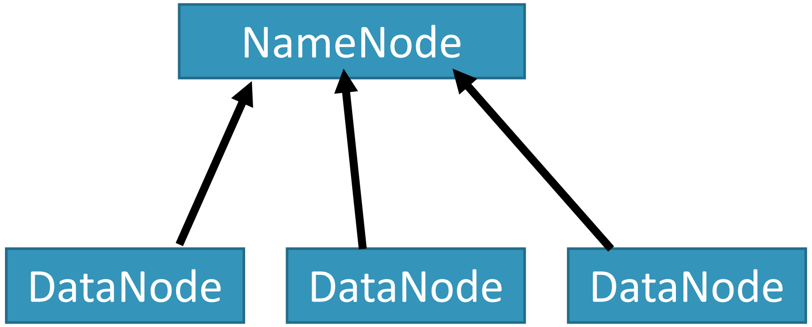
- fsimage
- edit logs (transaction logs)

NameNode

DataNode

Secondary NameNode

NameNode – orchestrates storage
Manages the File System Metadata



DataNode – Blocks are stored
Listen to NameNode’s instructions
Send heartbeats / block report

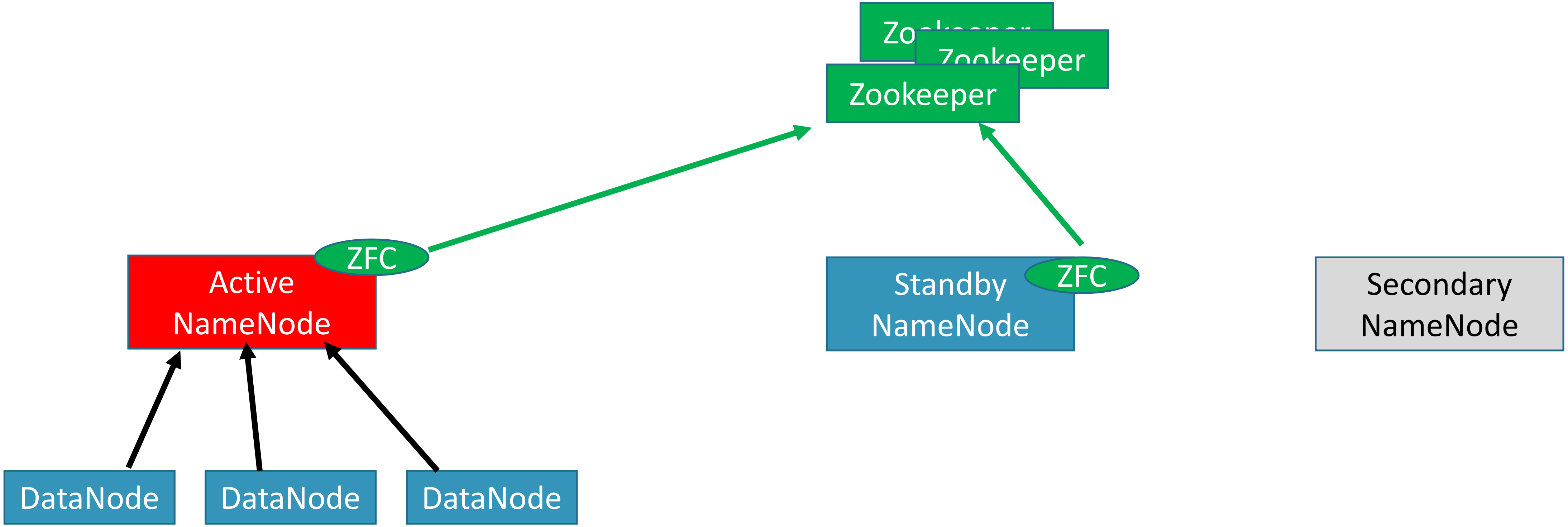
Checkpoint Node
– Regularly backup the metadata

Secondary NameNode

dfs.namenode.checkpoint.period=3600
dfs.namenode.checkpoint.txns=1000000

Hadoop 1.x → NameNode is Single Point of Failure

HDFS 2 – Introduces NameNode High Availability

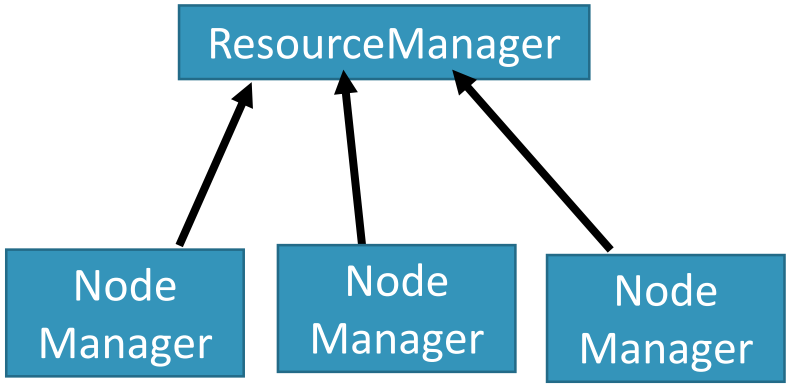


YARN Daemons

Master – Slave Architecture

ResourceManager
NodeManager
JobHistoryServer

ResourceManager – Scheduling Jobs



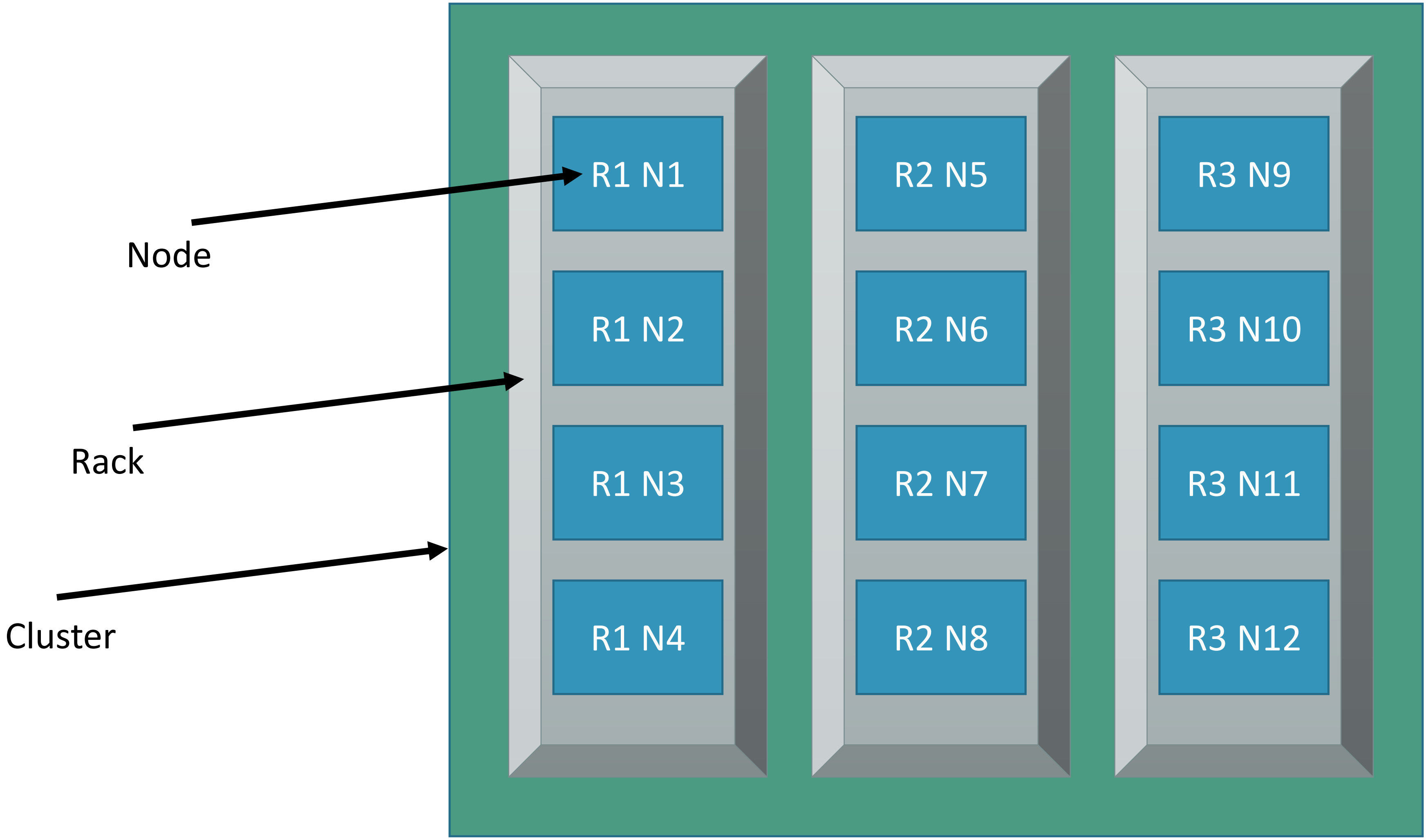
NodeManager – Execution of jobs

JobHistoryServer

Retired Jobs

MRAppMaster → Monitoring jobs

Hadoop Terminologies



Node

Rack

Cluster

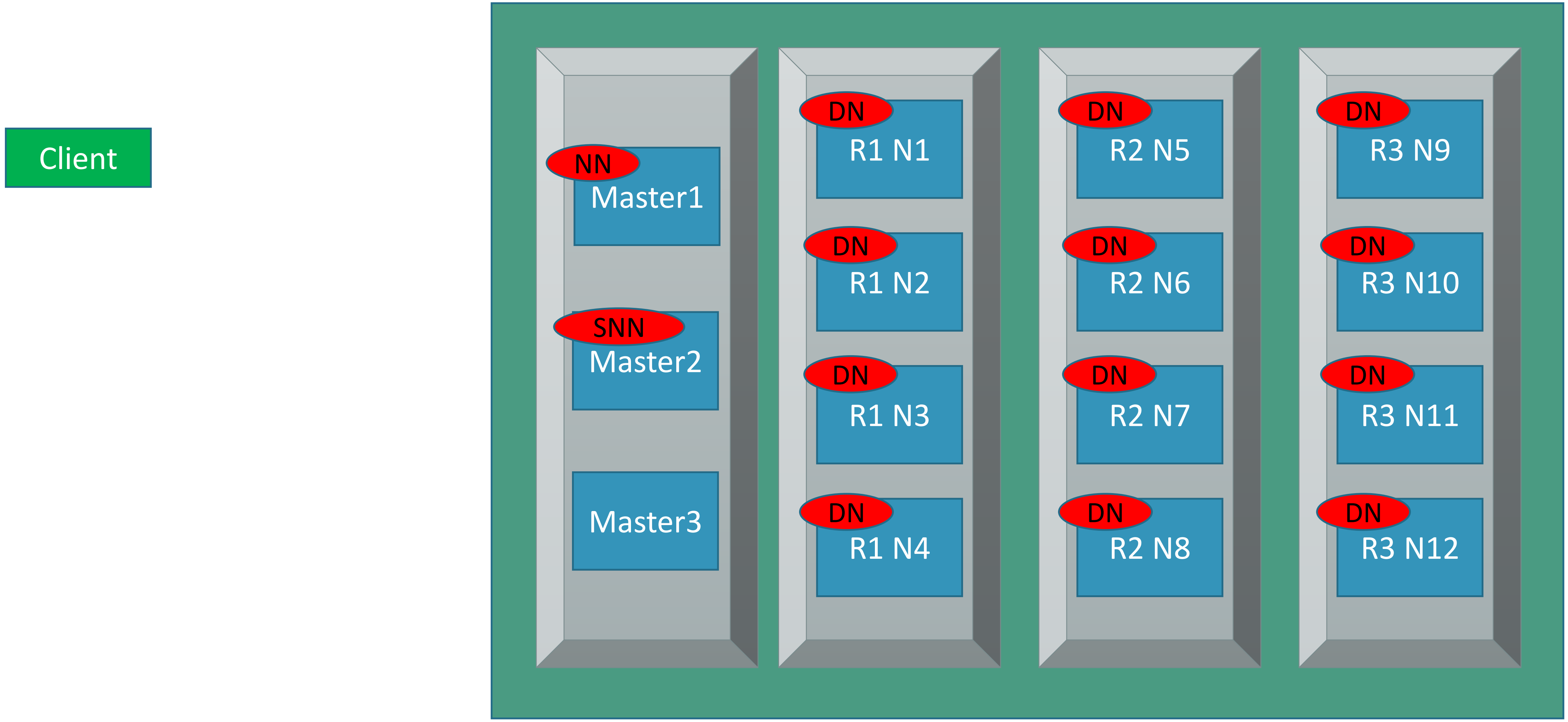
R1 N1 R2 N5 R3 N9

R1 N2 R2 N6 R3 N10

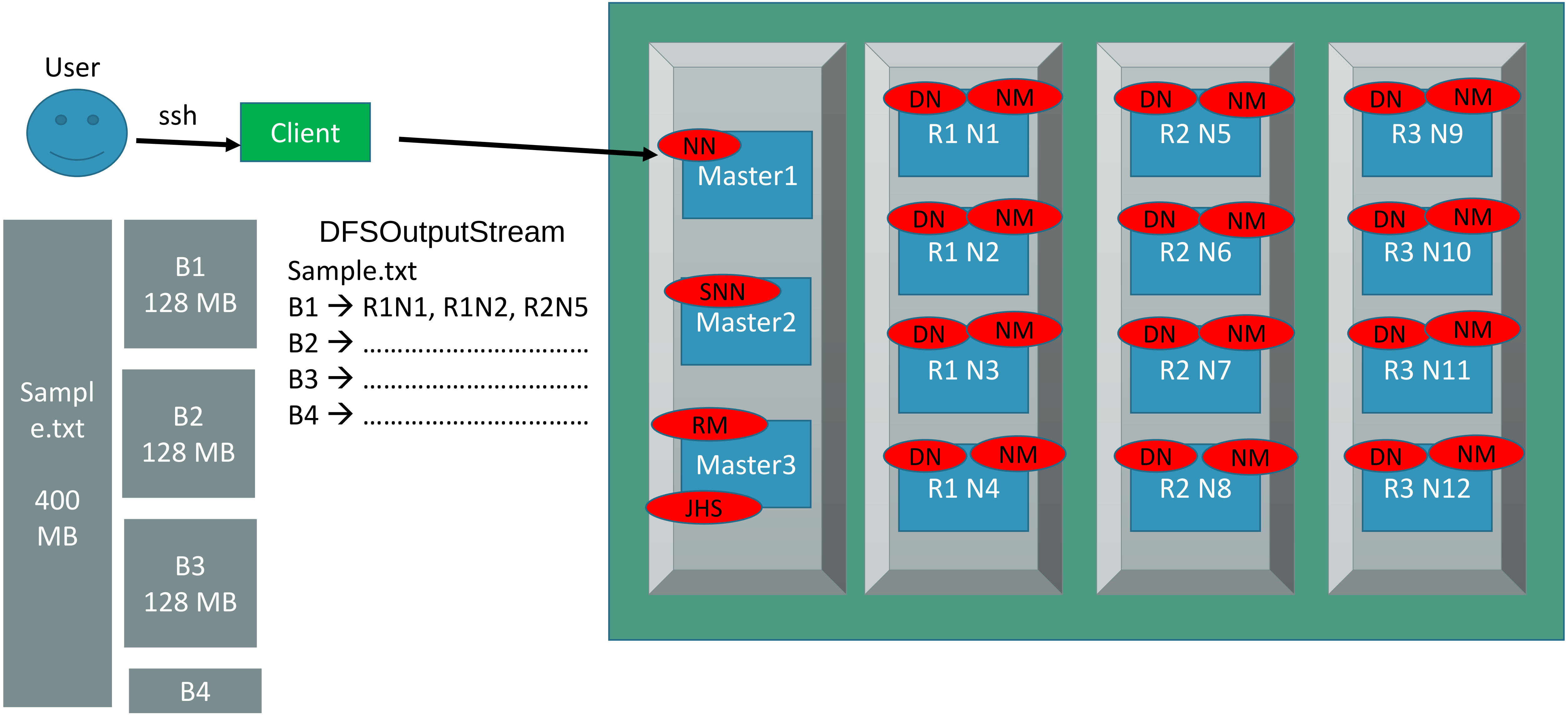
R1 N3 R2 N7 R3 N11

R1 N4 R2 N8 R3 N12

HDFS Daemons Distributed Over a cluster



How to write a File to the cluster



User

ssh

Client

Sampl
e.txt

400
MB

B1
128 MB

B2
128 MB

B3
128 MB

B4

DFSOutputStream

Sample.txt
B1 → R1N1, R1N2, R2N5
B2 → ………………………….
B3 → ………………………….
B4 → ………………………….

Default Hadoop Configuration

core-default.xml
hdfs-default.xml
mapred-default.xml
yarn-default.xml

→

```
dfs.replication=3
dfs.blocksize= 134217728 = 128 MB
dfs.heartbeat.interval=3
dfs.namenode.stale.datanode.interval=30000
```

Customized Hadoop Configuration

core-site.xml
hdfs-site.xml
mapred-site.xml
yarn-site.xml

→

```
dfs.replication=1
```

$HADOOP_HOME/etc/hadoop ➔ Hadoop’s conf dir

Hadoop Setup

Hardware

In premise

Virtualization

Cloud – AWS / GCP

Hadoop Flavor

Cloudera

Apache

Hortonworks

MAPR

O S Flavor

RHEL

SUSE

Fedora

Ubuntu

CentOS

……….

Mode of Deployment

Standalone Mode

Pseudo Distributed Mode

Fully Distributed Mode

- Standalone Operation By default, Hadoop is configured to run in a non-distributed mode, as a single Java process. This is useful for debugging

- Pseudo-Distributed Operation: Hadoop can also be run on a single-node in a pseudo-distributed mode where each Hadoop daemon runs in a separate Java process

- Fully Distributed Mode – Multi Node cluster, a typical production environment where each daemons would be distributed on many nodes

Hadoop Setup Steps

- Pre-Requisites
  - JDK
  - ssh (Passphraseless)

- Download and unpack Hadoop packages
- Customize Hadoop
  - core-site.xml
  - hdfs-site.xml
  - mapred-site.xml
  - yarn-site.xml
  - hadoop-env.sh
- Format the NameNode / DataNode
- Start Hadoop Services

Web UI Ports

NameNode – http://localhost:50070

ResourceManager – http://localhost:8088

HistoryServer – http://localhost:19888

Secondary NameNode - http://localhost:50090

DataNode - http://localhost:50075

Agenda – Day 2

- Multi Node Setup
- Working with HDFS & File System commands
- MapReduce & YARN
- Data Ingestion → Sqoop

Hadoop Setup – Multi Node Demo

Hardware

In premise
Virtualization
Cloud – AWS / GCP

O S Flavor

RHEL
SUSE
Fedora
Ubuntu
CentOS
……….

Hadoop Flavor

Cloudera
Apache
Hortonworks
MAPR

Mode of Deployment

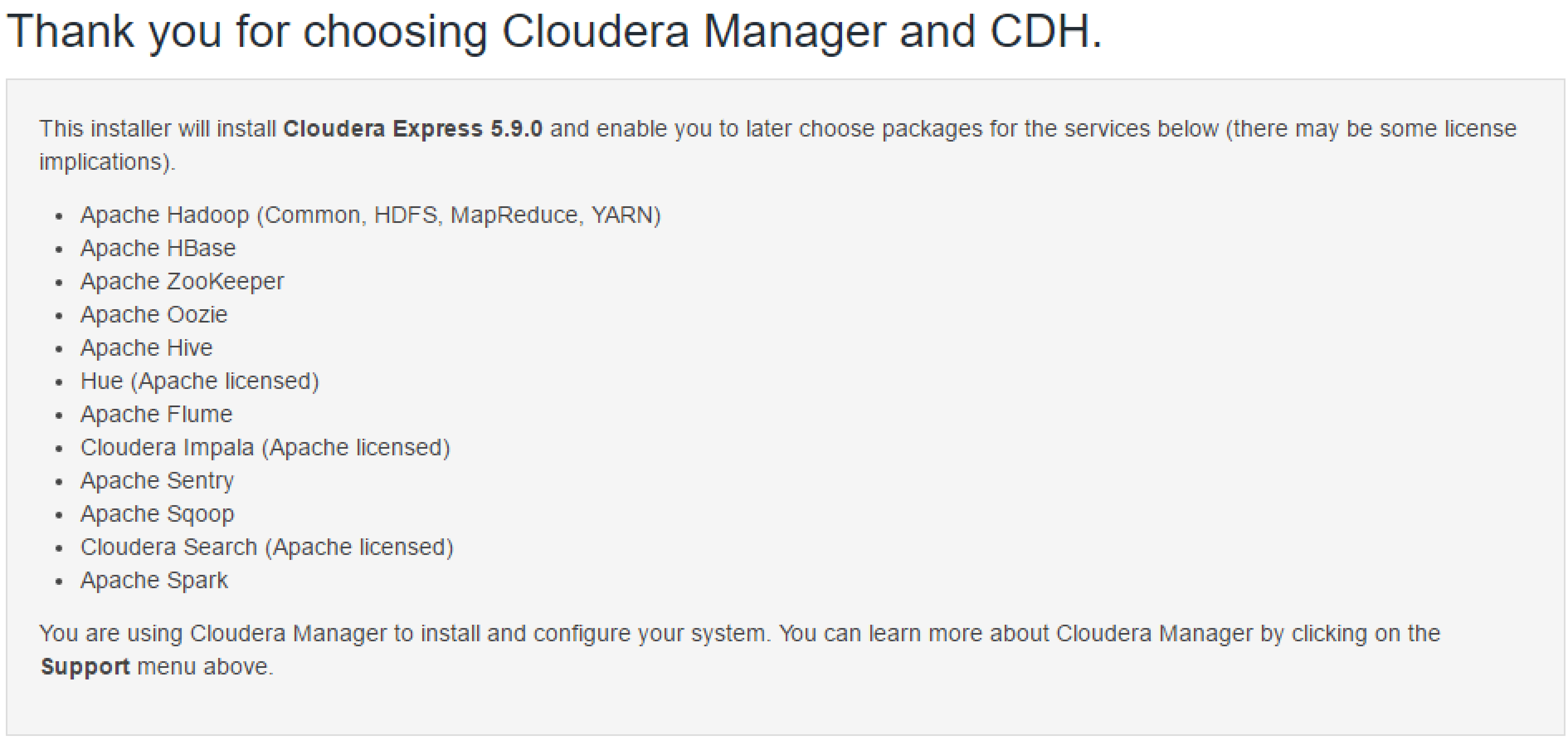
Standalone Mode
Pseudo Distributed Mode
Fully Distributed Mode

Hadoop Setup Steps (AWS)

- Login to AWS with credentials and get the required keys (*.pem) – AWS specific ssh key
- As we would work on Windows, to connect to the remote machine (AWS), we need a ssh client (Ex: PuTTY)
- While connecting via PuTTY, the *.pem is not recognized, we need PuTTYgen to convert *.pem to *.ppk
- Refer https://docs.aws.amazon.com/console/ec2/instances/connect/putty

1. Subscribe for 3 EC2 instances on https://aws.amazon.com/ of AMI "Ubuntu Server 14.04 LTS"
2. Connect to one of the instance and download the Cloudera installer using the command:

```
wget http://archive.cloudera.com/cm5/installer/latest/cloudera-manager-installer.bin
```



Thank you for choosing Cloudera Manager and CDH.

This installer will install **Cloudera Express 5.9.0** and enable you to later choose packages for the services below (there may be some license implications).

- Apache Hadoop (Common, HDFS, MapReduce, YARN)
- Apache HBase
- Apache ZooKeeper
- Apache Oozie
- Apache Hive
- Hue (Apache licensed)
- Apache Flume
- Cloudera Impala (Apache licensed)
- Apache Sentry
- Apache Sqoop
- Cloudera Search (Apache licensed)
- Apache Spark

You are using Cloudera Manager to install and configure your system. You can learn more about Cloudera Manager by clicking on the **Support** menu above.

Working with HDFS

http://hadoop.apache.org/docs/r2.7.3/hadoop-project-dist/hadoop-common/FileSystemShell.html

Read about HDFS Metadata directories from the document within the VM
/home/user1/Downloads/.05_Programs/15_Misc/HDFS_Metadata_Directories.pdf

Introduction to MapReduce

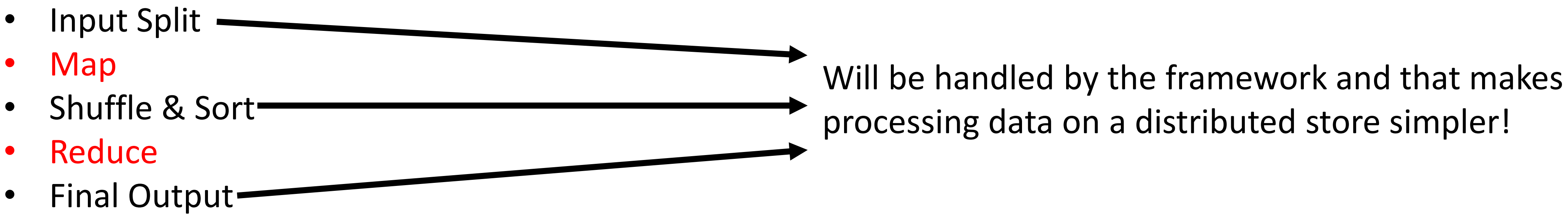
MapReduce works on (Key, Value) pairs

Example: (Welcome, 1)
Welcome → Key
1 → Value

Sort → Key

- Processing Engine of Hadoop
- Works in 2 phases (Developer)
  - Map
  - Reduce

- The Framework has following phases



  - Input Split
  - Map
  - Shuffle & Sort
  - Reduce
  - Final Output

Will be handled by the framework and that makes processing data on a distributed store simpler!

In Map phase – parse, transform, extract
In Reduce phase – statistics, aggregations

MapReduce WordCount Program

Problem Statement: Count the occurrence of each word in the file

/Sample/SampleFile.txt

```
Welcome to Hadoop
Learning Hadoop is fun
Hadoop Hadoop Hadoop is the buzz
```

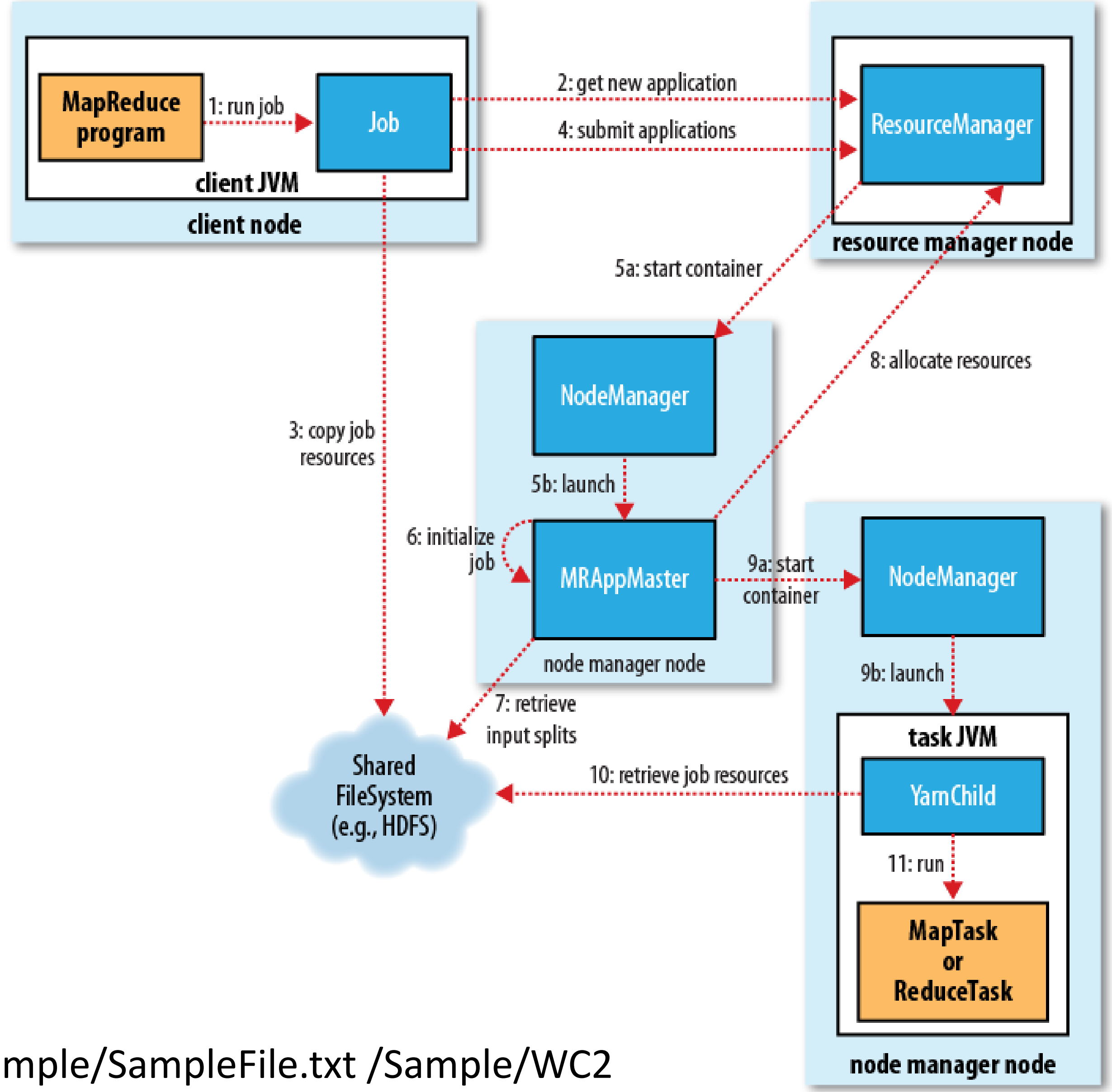
Expected Output

```
Hadoop, 5
Learning, 1
Welcome, 1
buzz, 1
fun, 1
is, 2
the, 1
to, 1
```

Line to word → Map
Count the words → Reduce

The Anatomy of a Job Run on YARN

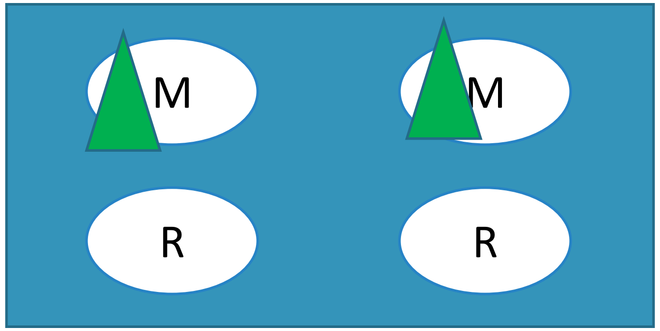
ResourceManager – 1 / cluster
MRAppMaster – 1 / job
YarnChild – 1 / task



yarn jar /home/user1/Documents/wc.jar WordCount /Sample/SampleFile.txt /Sample/WC2

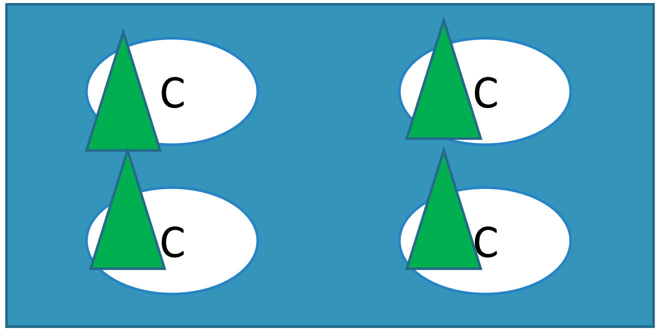
Limitations of Hadoop 1.x

- NameNode is the Single Point of Failure
- JobTracker is the Single Point of Failure
- JobTracker is overburdened → Scheduling + Monitoring
- TaskTracker(s) have slots (2 map slots and 2 reduce slots by default)
  - mapreduce.tasktracker.map.tasks.maximum=2
  - mapreduce.tasktracker.reduce.tasks.maximum=2
- Inefficient Resource Utilization at the cluster level
- Scalability is at a threshold of 4000+ nodes
- The TaskTracker(s) were only capable of Map/Reduce
- FIFO scheduler on JobTracker

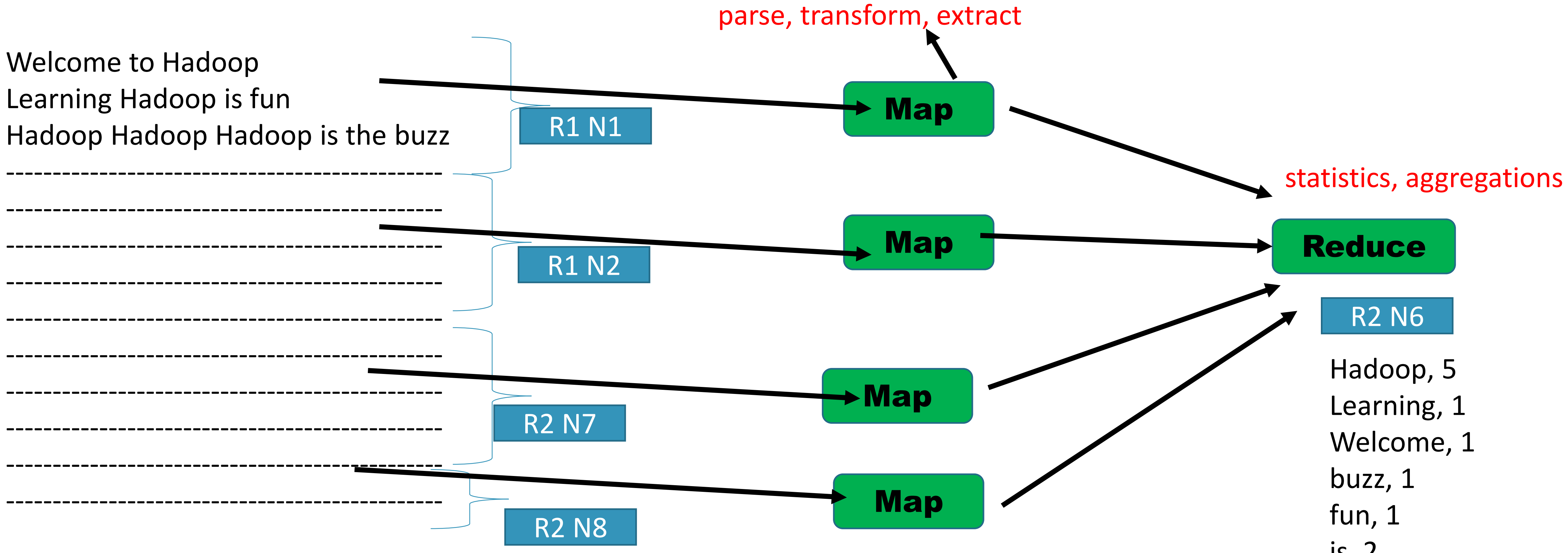


Features of Hadoop 2.x

- NameNode HA → Active NameNode + Standby NameNode
- Resource Manager HA → Active ResourceManager + Standby ResourceManager
- JobTracker is now ResourceManager (1 / cluster) and Application Master (1 / job)
- TaskTracker(s) are NodeManagers and have generic containers capable of running applications beyond Map/Reduce
- ResourceManager + Application Manager + Application Master + History Server = YARN (Yet Another Resource Negotiator) is a new cluster resource management layer which promises efficient cluster resource utilization even at a scale of 25000+ nodes in a cluster
- Capacity Scheduler
- Multi Tenancy



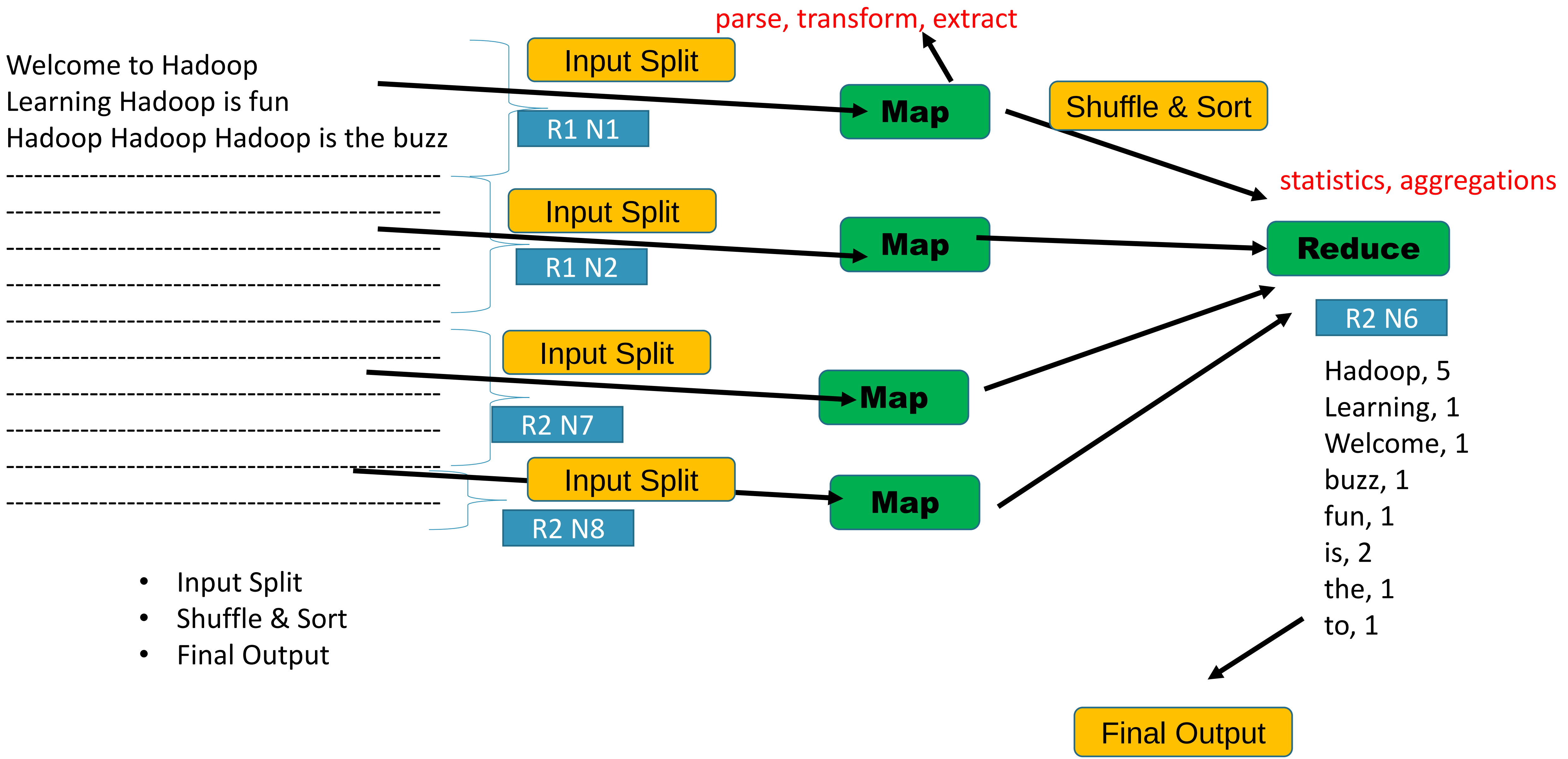
MapReduce Detailed



Welcome to Hadoop
Learning Hadoop is fun
Hadoop Hadoop Hadoop is the buzz

Hadoop, 5
Learning, 1
Welcome, 1
buzz, 1
fun, 1
is, 2
the, 1
to, 1

MapReduce Detailed Steps



Welcome to Hadoop
Learning Hadoop is fun
Hadoop Hadoop Hadoop is the buzz

Hadoop, 5
Learning, 1
Welcome, 1
buzz, 1
fun, 1
is, 2
the, 1
to, 1

- Input Split
- Shuffle & Sort
- Final Output

MapReduce API Overview

/home/user1/HadoopInstallations/hadoop-2.7.1/share/doc/hadoop/api/index.html

Map Signature

```
Mapper Class {
        map (key, value, context) {
                ----------------------------------------
                ----------------------------------------
                ----------------------------------------
                   Logic for Transformation
                ----------------------------------------
                ----------------------------------------
                ----------------------------------------
}
}
```

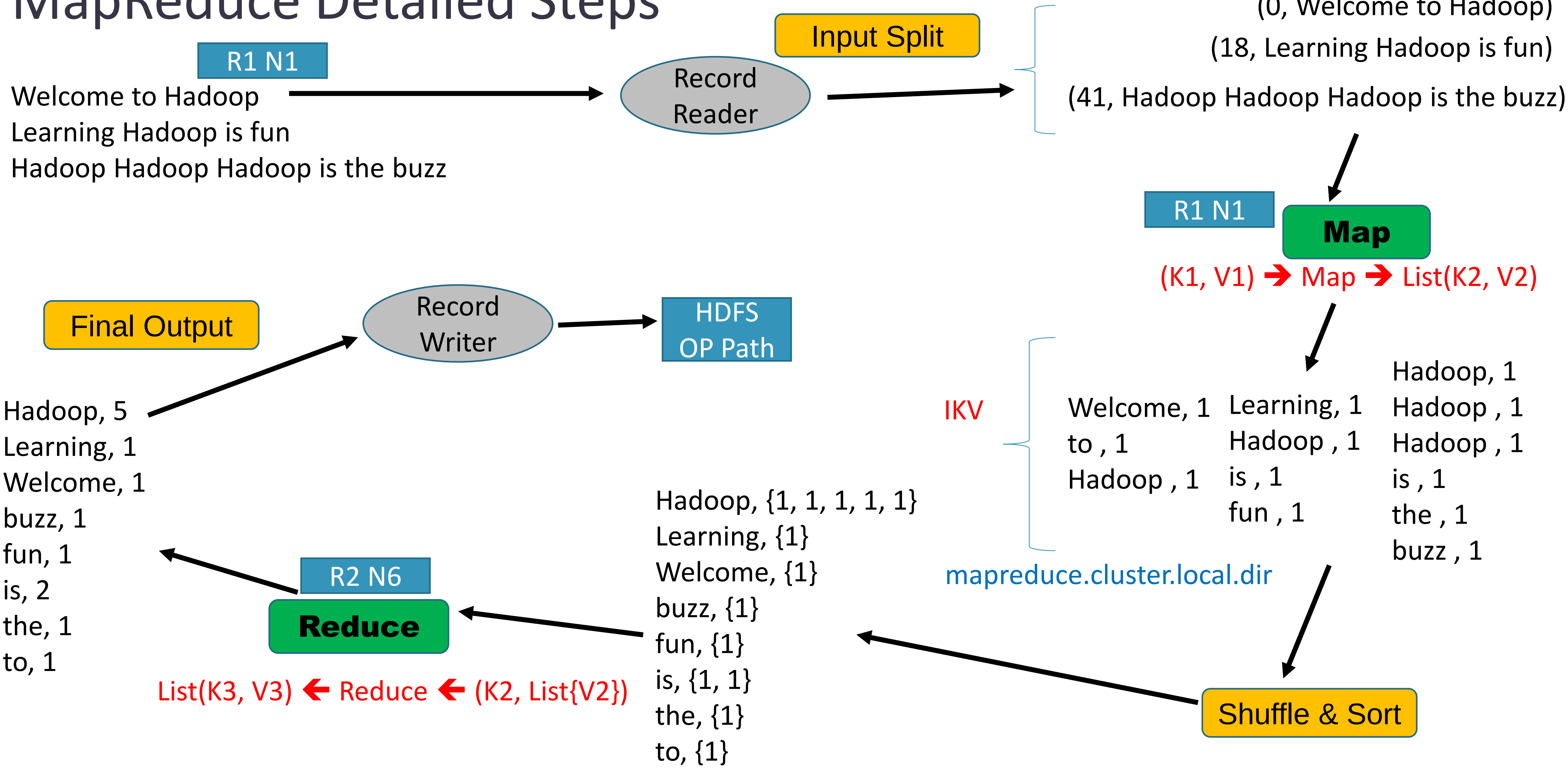
(K1, V1) ➔ Map ➔ List(K2, V2)

Reduce Signature

```
Reducer Class {
        reduce (key, list{values}, context) {
                ----------------------------------------
                ----------------------------------------
                ----------------------------------------
                   Logic for Aggregation
                ----------------------------------------
                ----------------------------------------
                ----------------------------------------
}
}
```

(K2, List{V2}) ➔ Reduce ➔ List(K3, V3)

MapReduce Detailed Steps



R1 N1

Welcome to Hadoop
Learning Hadoop is fun
Hadoop Hadoop Hadoop is the buzz

Record Reader

Input Split

(0, Welcome to Hadoop)
(18, Learning Hadoop is fun)
(41, Hadoop Hadoop Hadoop is the buzz)

R1 N1

Map

(K1, V1) ➔ Map ➔ List(K2, V2)

IKV

Welcome, 1
to , 1
Hadoop , 1

Learning, 1
Hadoop , 1
is , 1
fun , 1

Hadoop, 1
Hadoop , 1
Hadoop , 1
is , 1
the , 1
buzz , 1

mapreduce.cluster.local.dir

Shuffle & Sort

Hadoop, {1, 1, 1, 1, 1}
Learning, {1}
Welcome, {1}
buzz, {1}
fun, {1}
is, {1, 1}
the, {1}
to, {1}

R2 N6

Reduce

List(K3, V3) 🡐 Reduce 🡐 (K2, List{V2})

Hadoop, 5
Learning, 1
Welcome, 1
buzz, 1
fun, 1
is, 2
the, 1
to, 1

Final Output

Record Writer

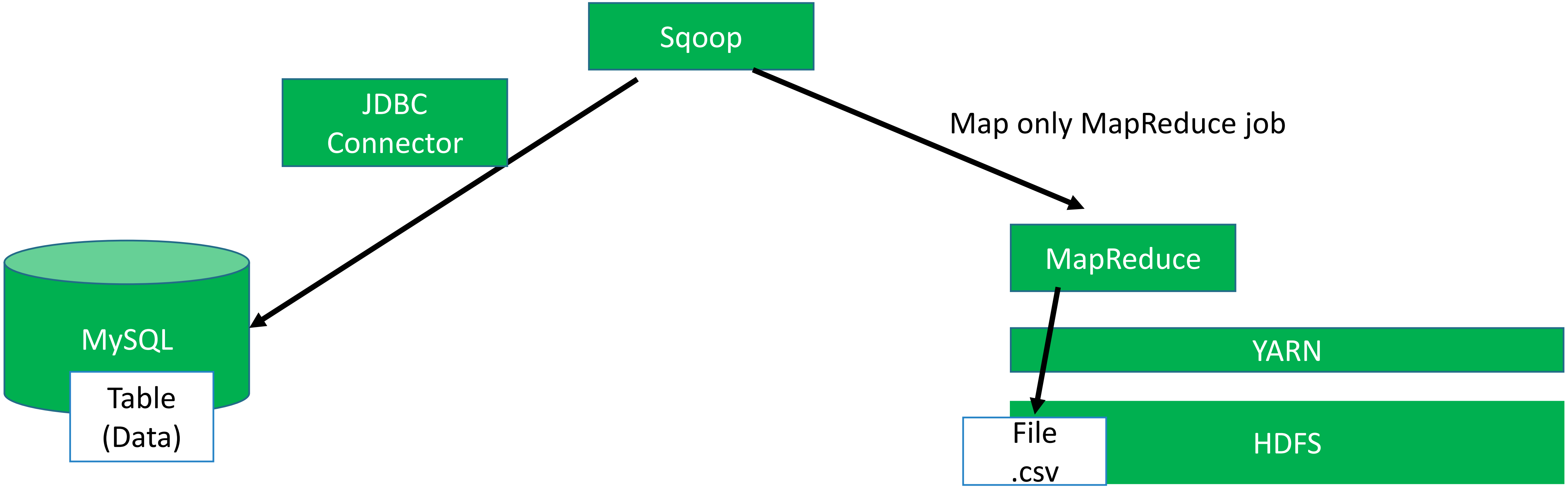
HDFS OP Path

Apache Sqoop

http://sqoop.apache.org/

Apache Sqoop is a tool designed for efficiently transferring bulk data between Apache Hadoop and structured data stores such as relational databases

Sqoop Import → From RDMBS to HDFS
Sqoop Export → From HDFS to RDBMS



Agenda – Day 3

- Advanced MapReduce
- Data Ingestion → Flume
- Hive
- Advanced Hive

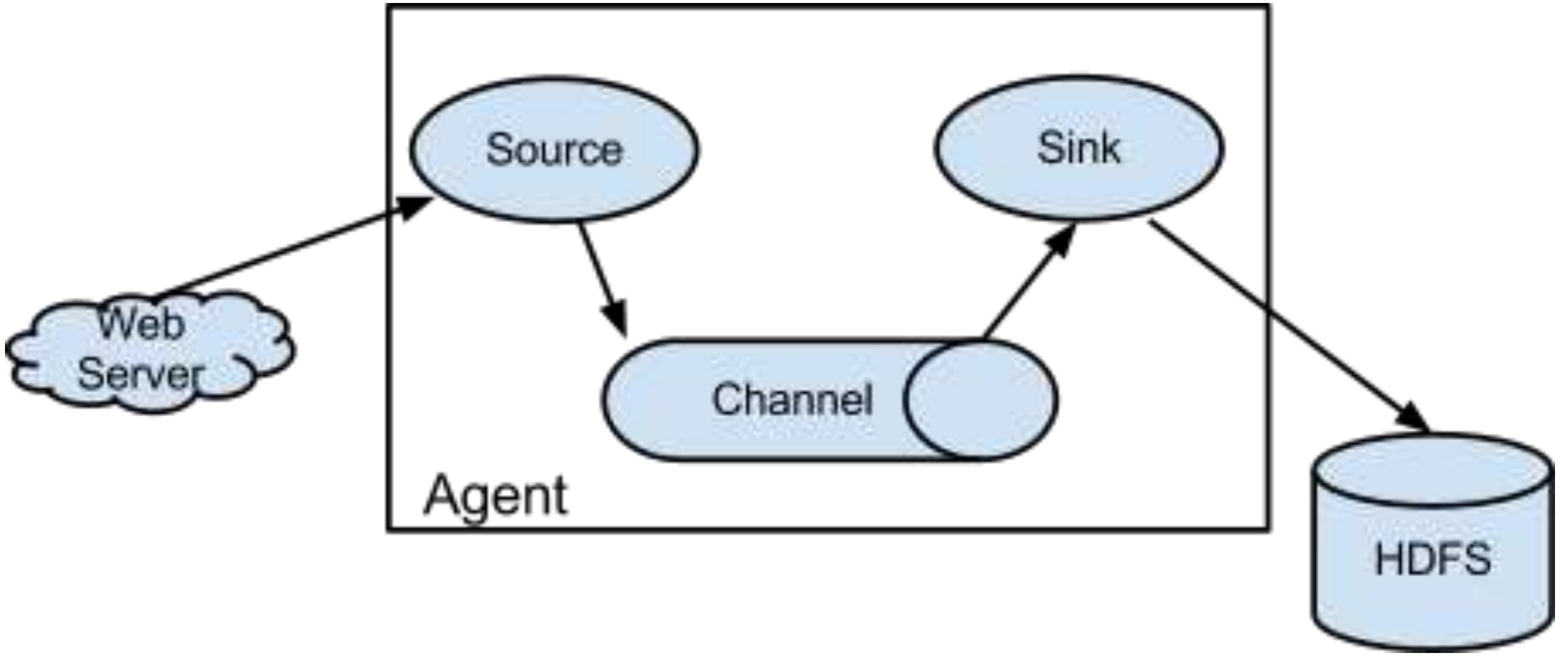
- Advanced MapReduce
    - Combiners & Partitioners
    - Chaining Multiple Maps
    - Joining datasets in MR
    - Working with images, sequence files
    - MapReduce datatypes

Apache Flume

http://flume.apache.org/

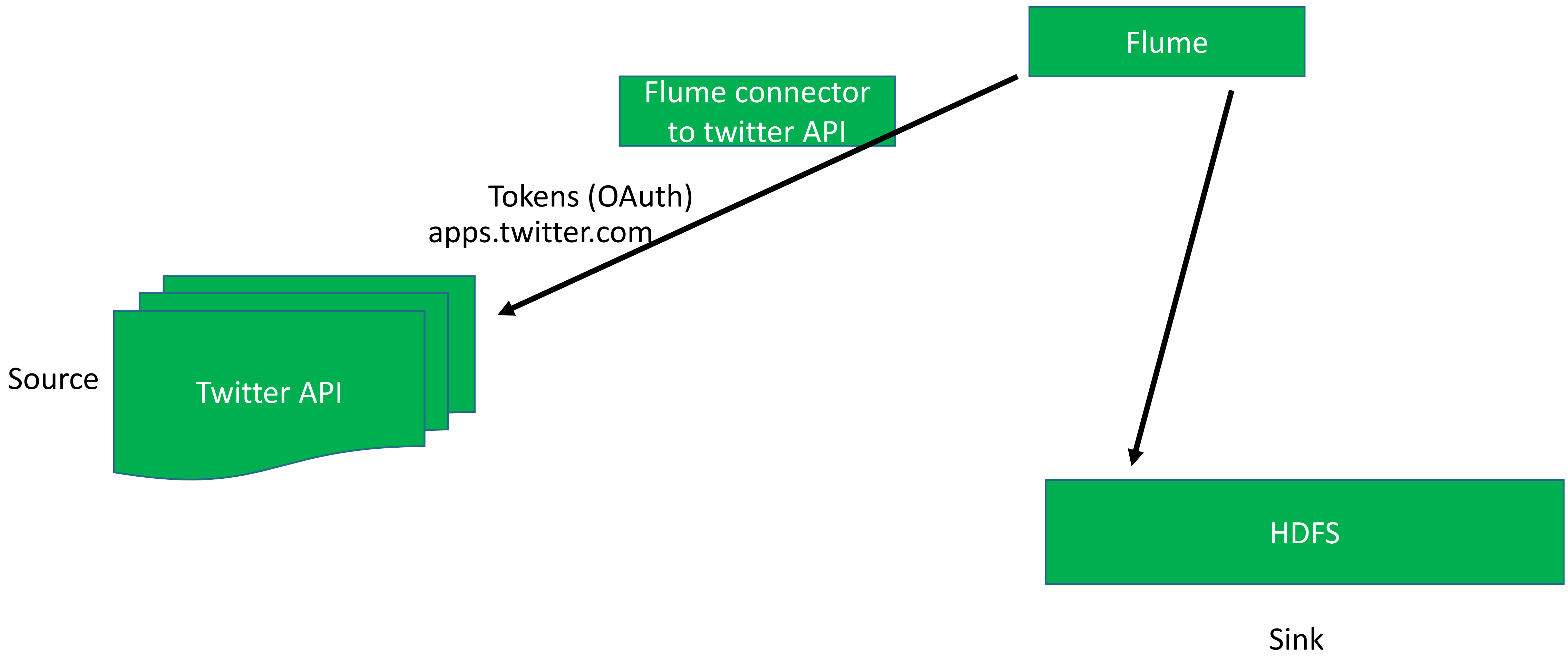
Flume is a distributed service for efficiently collecting and moving large amounts of log data

Example:



http://www.cloudera.com/documentation/kafka/latest.html

Apache Flume - Twitter

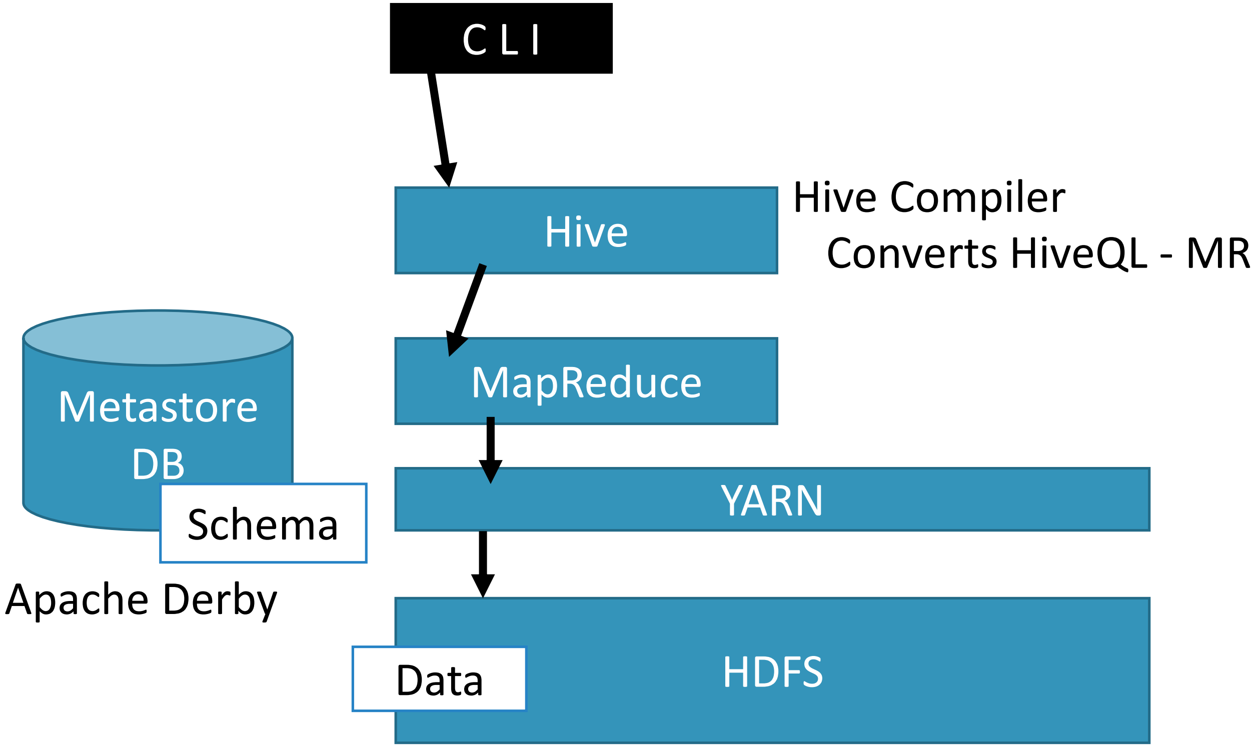


Apache Hive

http://hive.apache.org/

- The Apache Hive data warehouse software facilitates reading managing large datasets residing in distributed storage using SQL
- Structure can be projected onto data already in storage → Schema on Read

/user/hive/warehouse → Hive’s default warehouse directory



Hive Tables

- Managed Tables → Default → Hive manages data & table
- External Tables → External Keyword → Hive manages table

HDFS → Hadoop File System (Read Only)

MapR FS → MapR Distribution (RW)

KUDU → Cloudera Distribution (RW)

Further Reading

http://hadoop.apache.org/

http://hortonworks.com/blog/impala-vs-hive-performance-benchmark/

https://cwiki.apache.org/confluence/display/Hive/LanguageManual