APAN PS5400: Managing Data

Week 12: Hadoop-HDFS, MapReduce

Lecturer: François Scharffe

COLUMBIA UNIVERSITY
School of Professional Studies

Last Week

- Reverse engineering Kayak

This week

- Introduction to Hadoop Ecosystem
- HDFS
- MapReduce
- Inverted Spectrum as a data model
- YARN

What is Hadoop

- Hadoop is a massively distributed data management and processing system on clusters of commodity hardware
  - Unlike Cassandra and MongoDB database systems we have studied, Hadoop can do all sorts of processing (not just database operations).
- Apache Hadoop is open source
- Necessitated by the advent of big data and applications based on big data

Basic Principle: Distributing Data + Distributing computation (Moving Computation to Data)

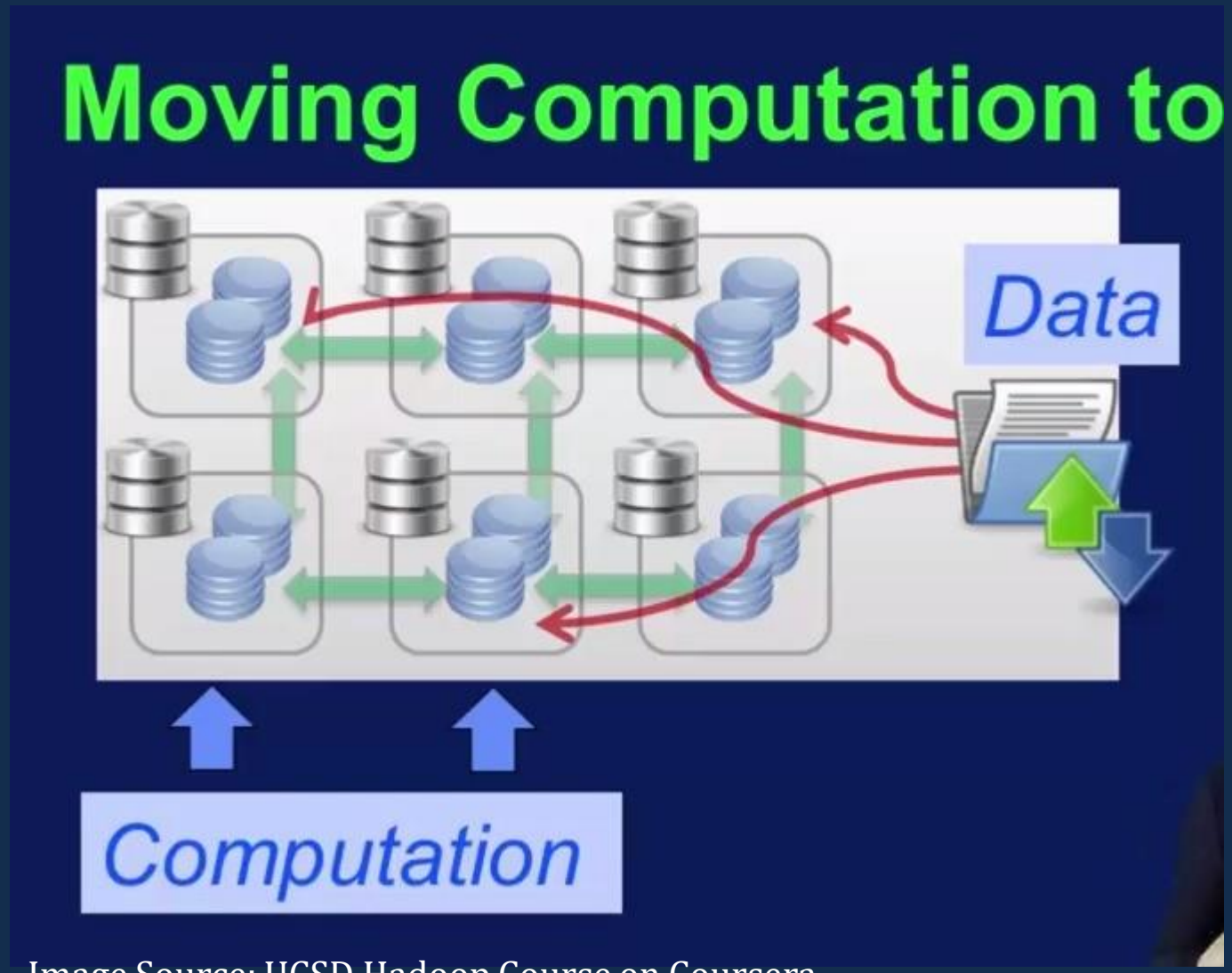
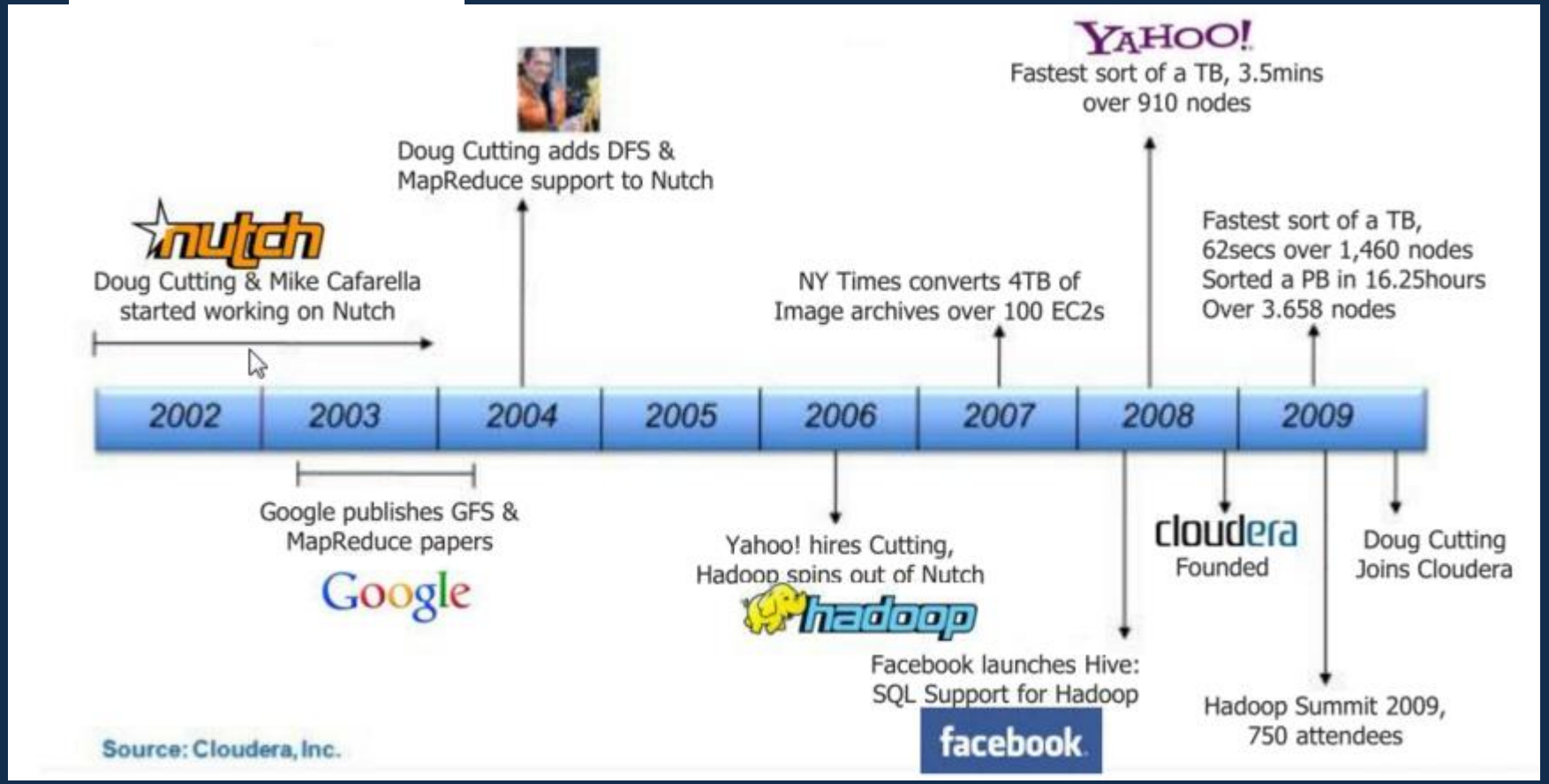


Image Source: UCSD Hadoop Course on Coursera

History & Development



What is Hadoop used for?

Searching

Log processing

Recommendation systems

Analytics

Video and image analysis

Data retention

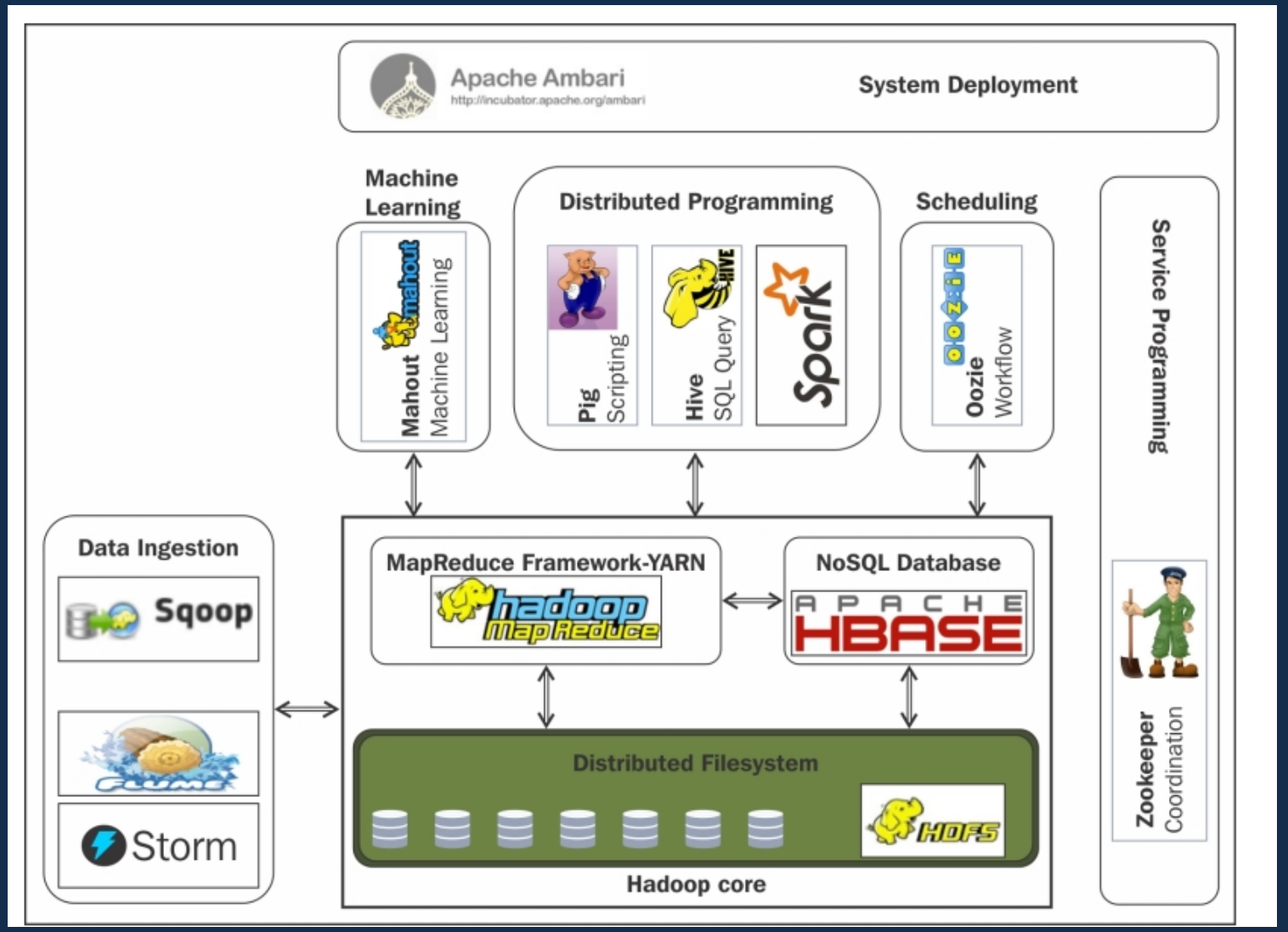
History & Development

There are many, many Hadoop distributions

Amazon Web Services
Apache Bigtop
Cascading
Cloudera
Cloudspace
Datameer
Data Mine Lab
Datasalt
DataStax
DataTorrent
Ndisco
Debian
Emblocsoft
Hortonworks
HStreaming
IBM
Impetus
Jaspersoft
Karmasphere
Apache Mahout
Nutch
NGDATA
Pentaho
Pervasive Software
Pivotal
Syncsort
Talend
Think Big
Tresata
Vmware
WANdisco

https://wiki.apache.org/hadoop/Distributions%20and%20Commercial%20Support

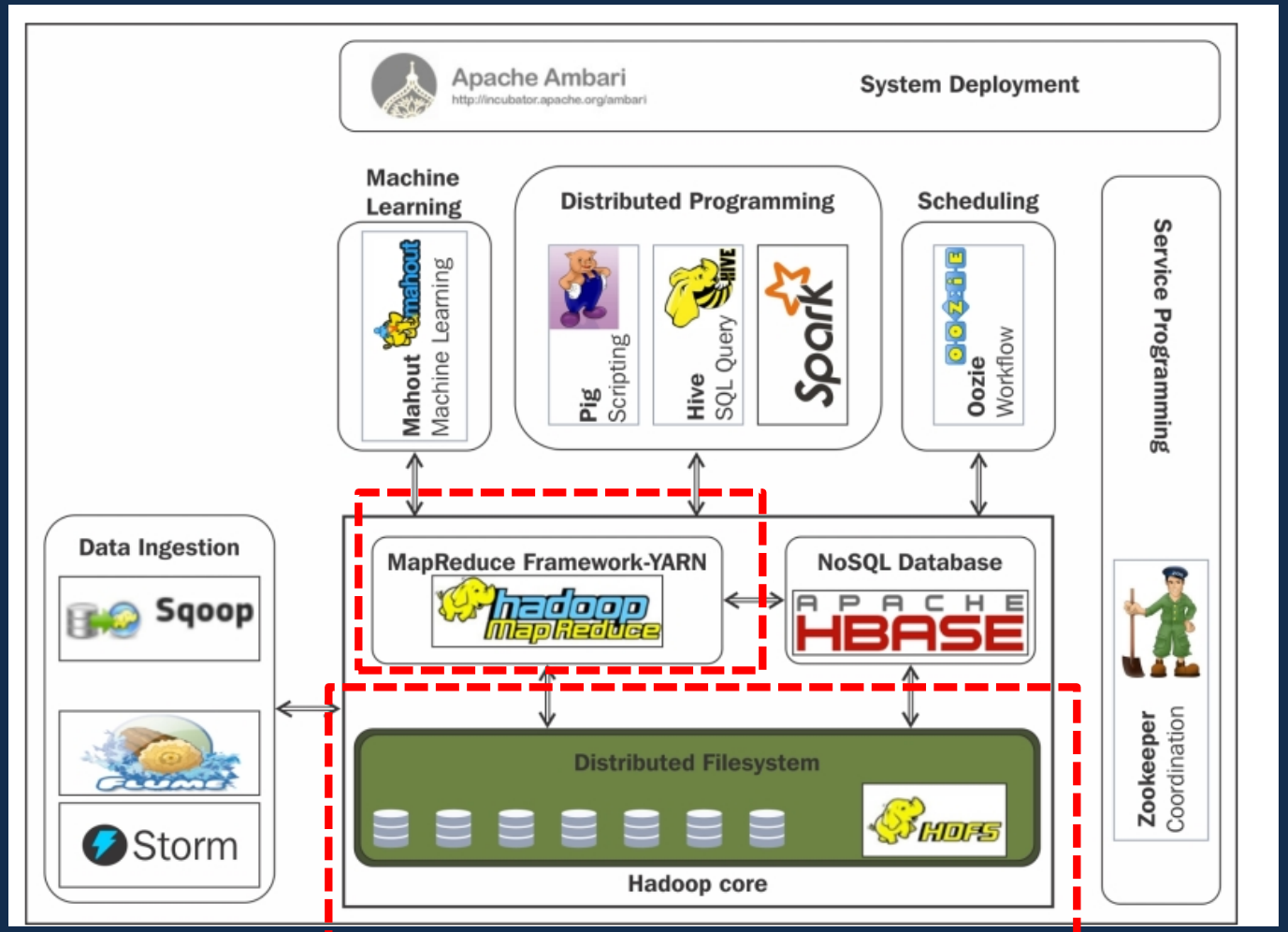
Key terms / key concepts



From Swizec , ***Hadoop Essentials***

Key terms / key concepts

Foundation

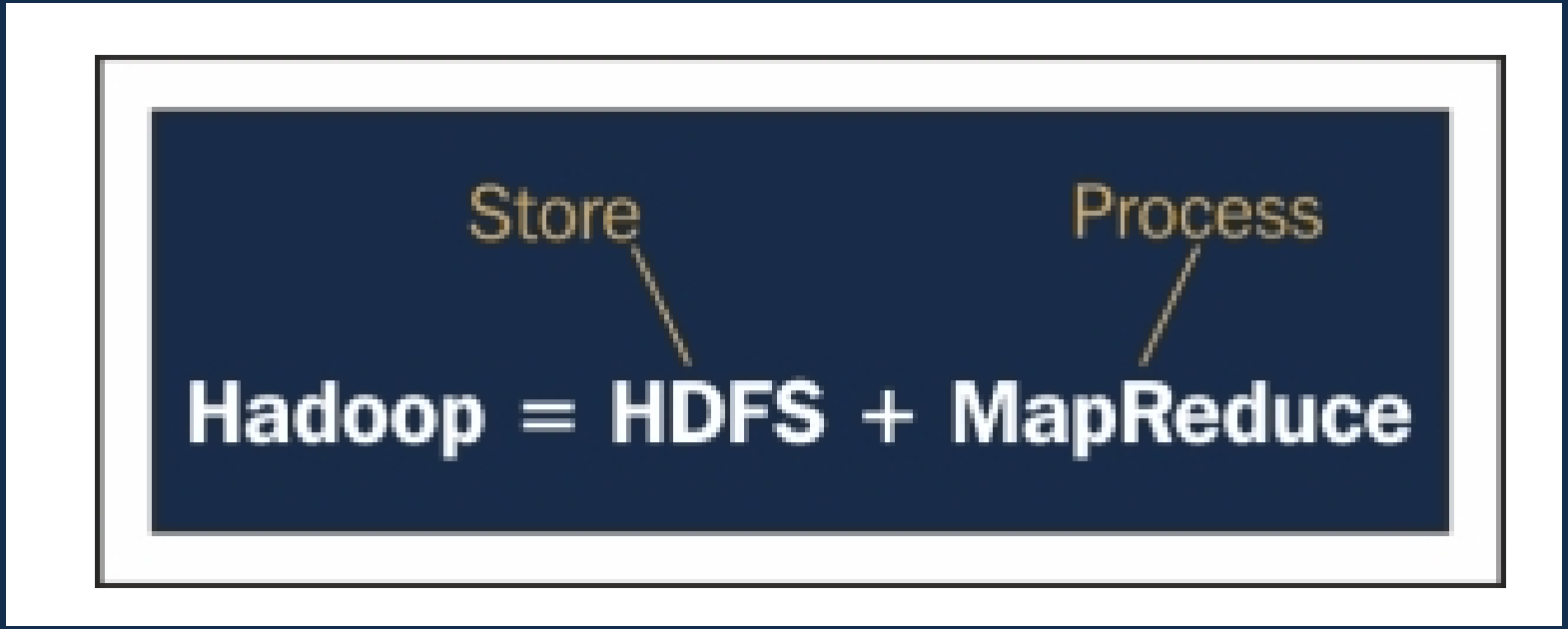


From Swizec , ***Hadoop Essentials***

Foundation

- HDFS (Hadoop Distributed File System)
- MapReduce
- YARN (Introduced in Hadoop 2)
- Hadoop Common (contains libraries used by the other Hadoop components)

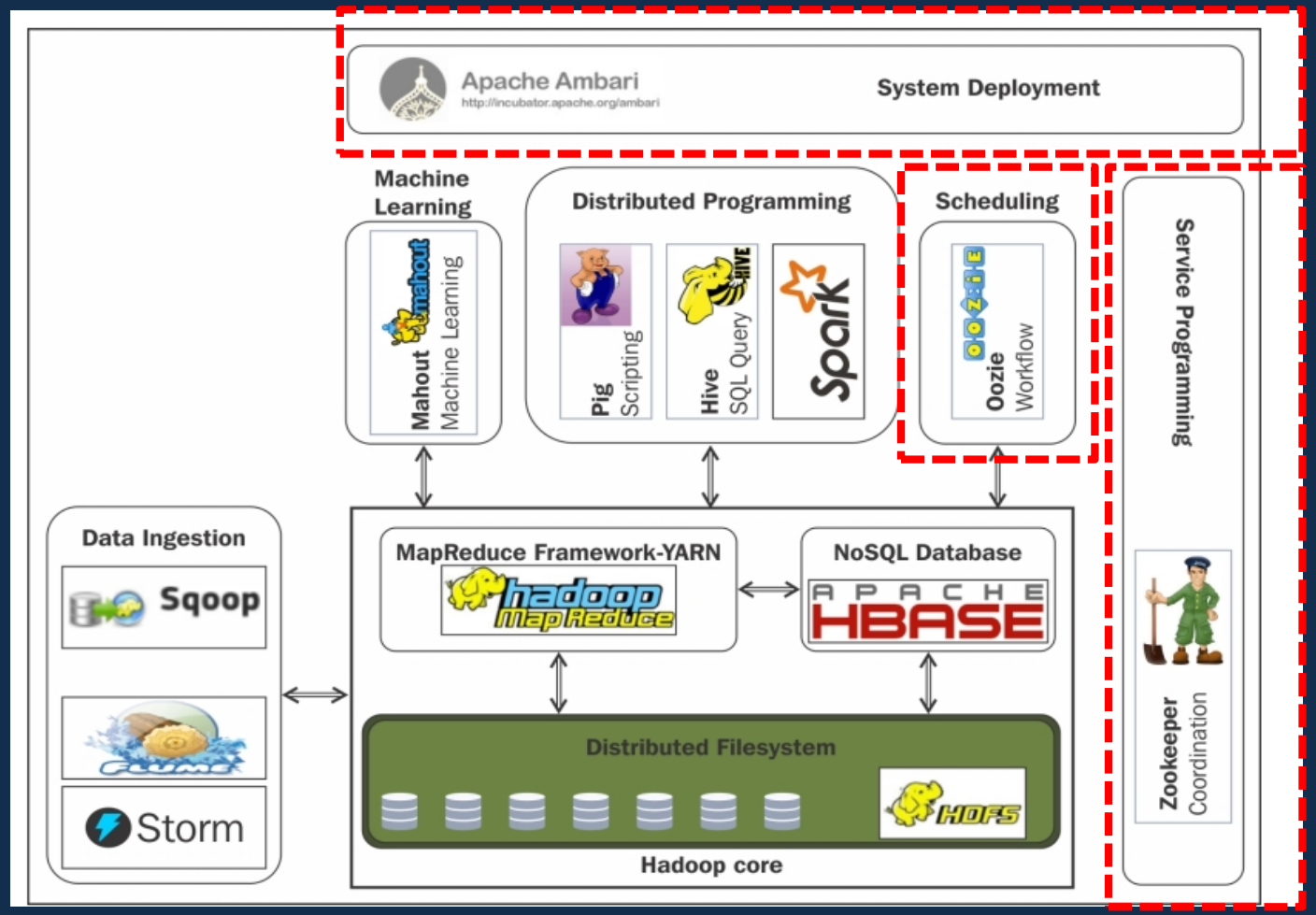
Key terms / key concepts



YARN is more about resource management

From Swizec , ***Hadoop Essentials***

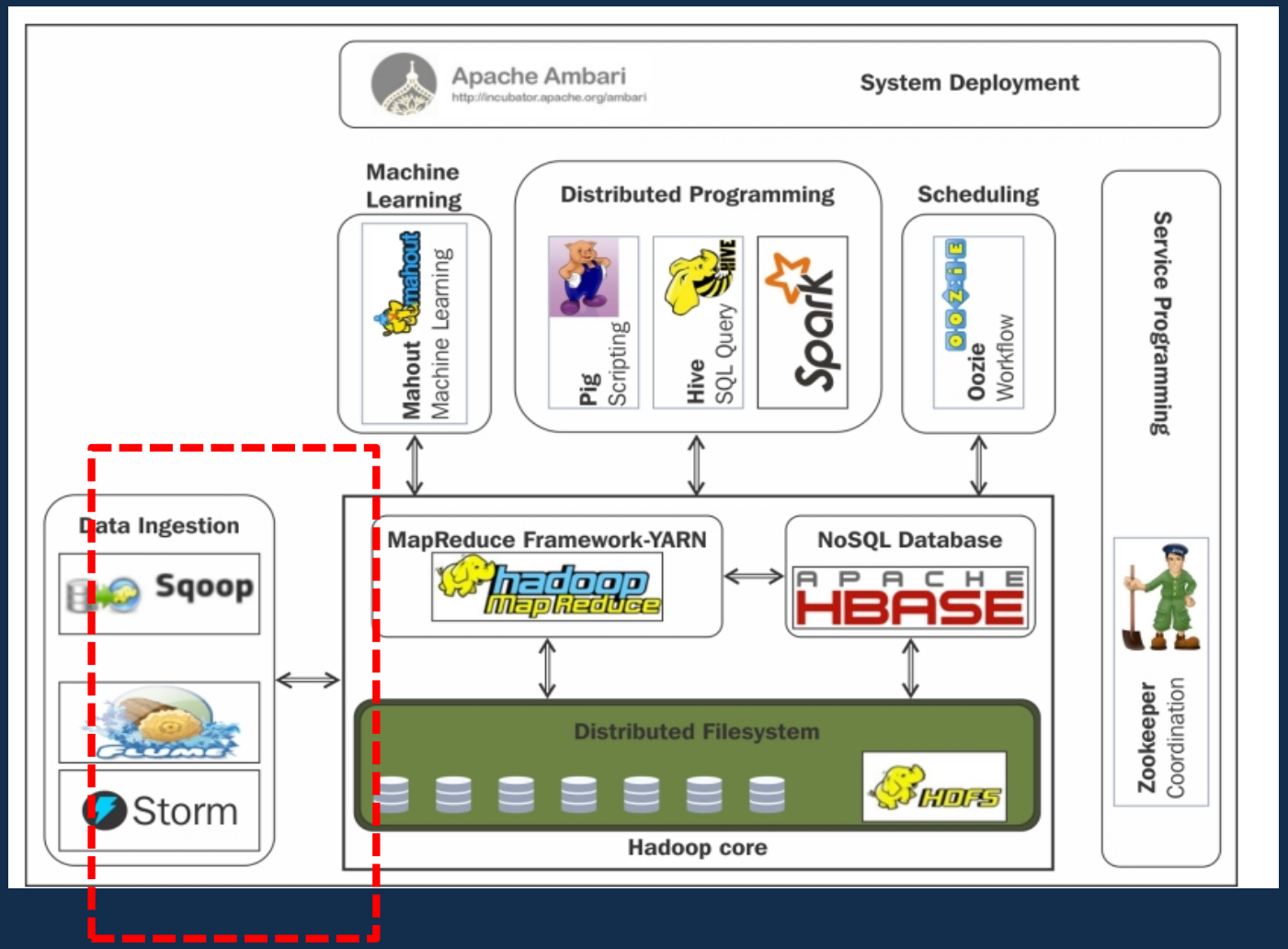
Managing Hadoop



- Ambari
- Zookeeper
- Oozie

From Swizec , ***Hadoop Essentials***

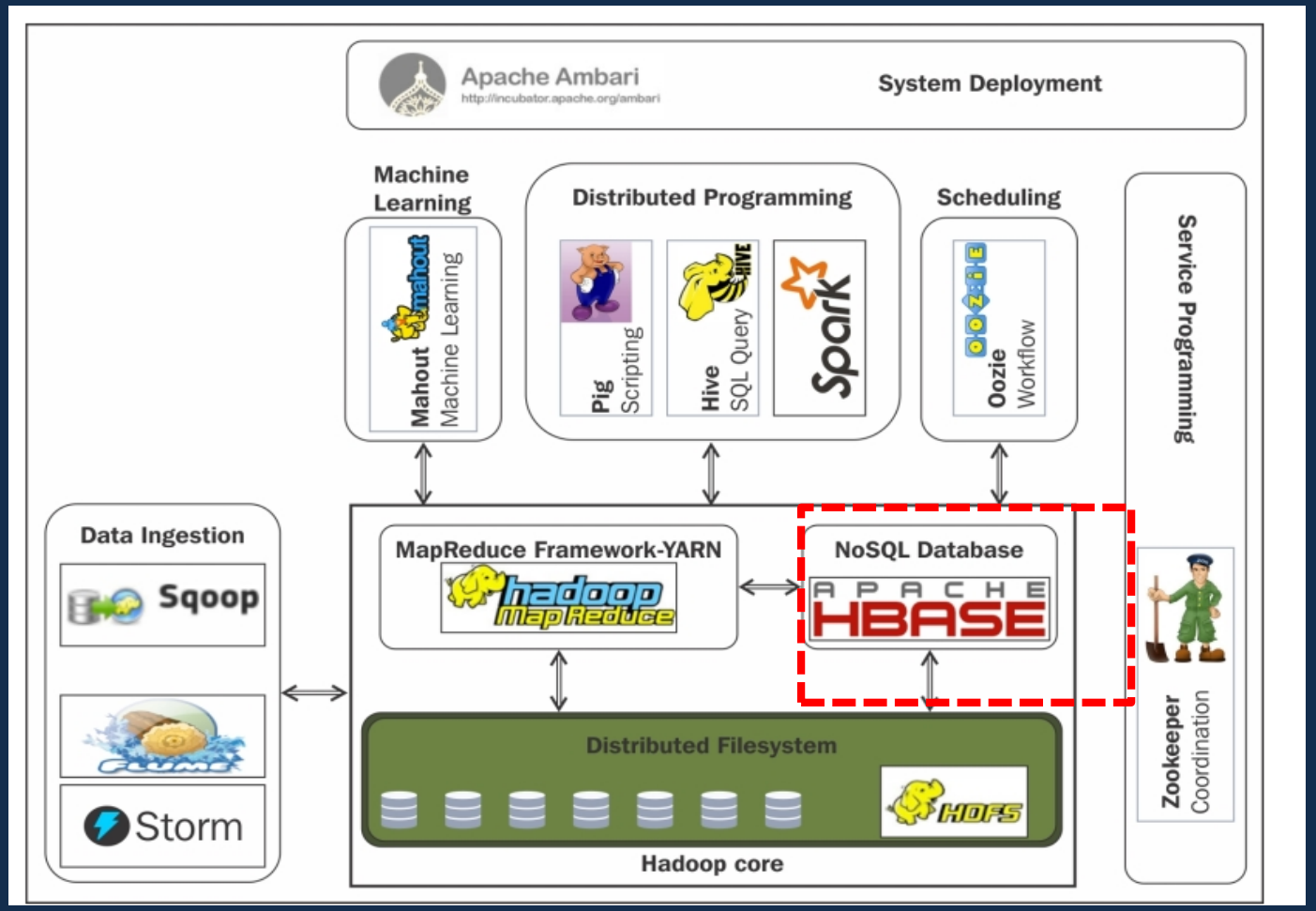
Getting data into Hadoop



- Flume
- Sqoop
- Storm

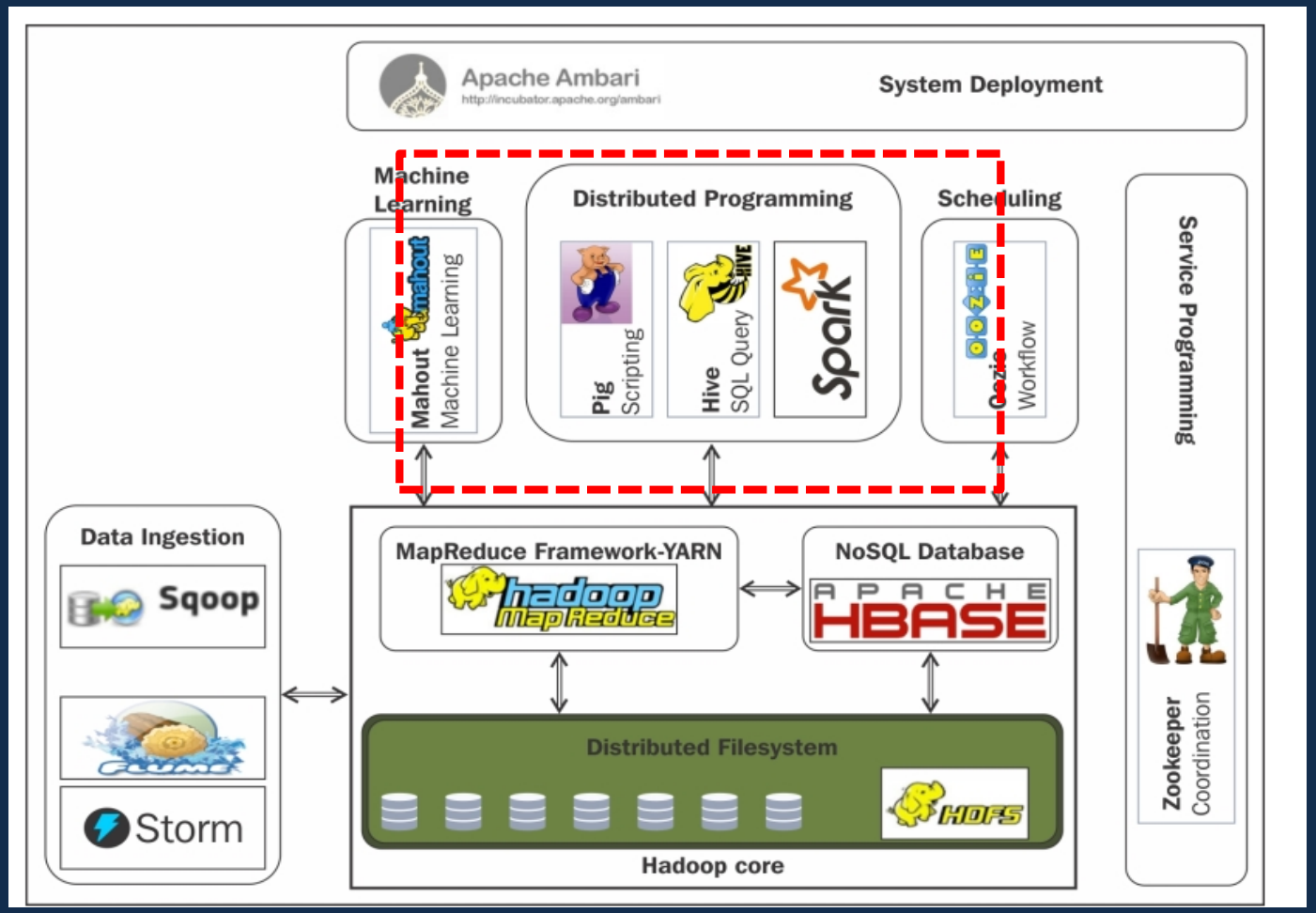
From Swizec , ***Hadoop Essentials***

Hadoop's database



From Swizec , ***Hadoop Essentials***

Getting insight out of Hadoop



- Pig
- Hive
- Mahout

From Swizec , ***Hadoop Essentials***

HDFS

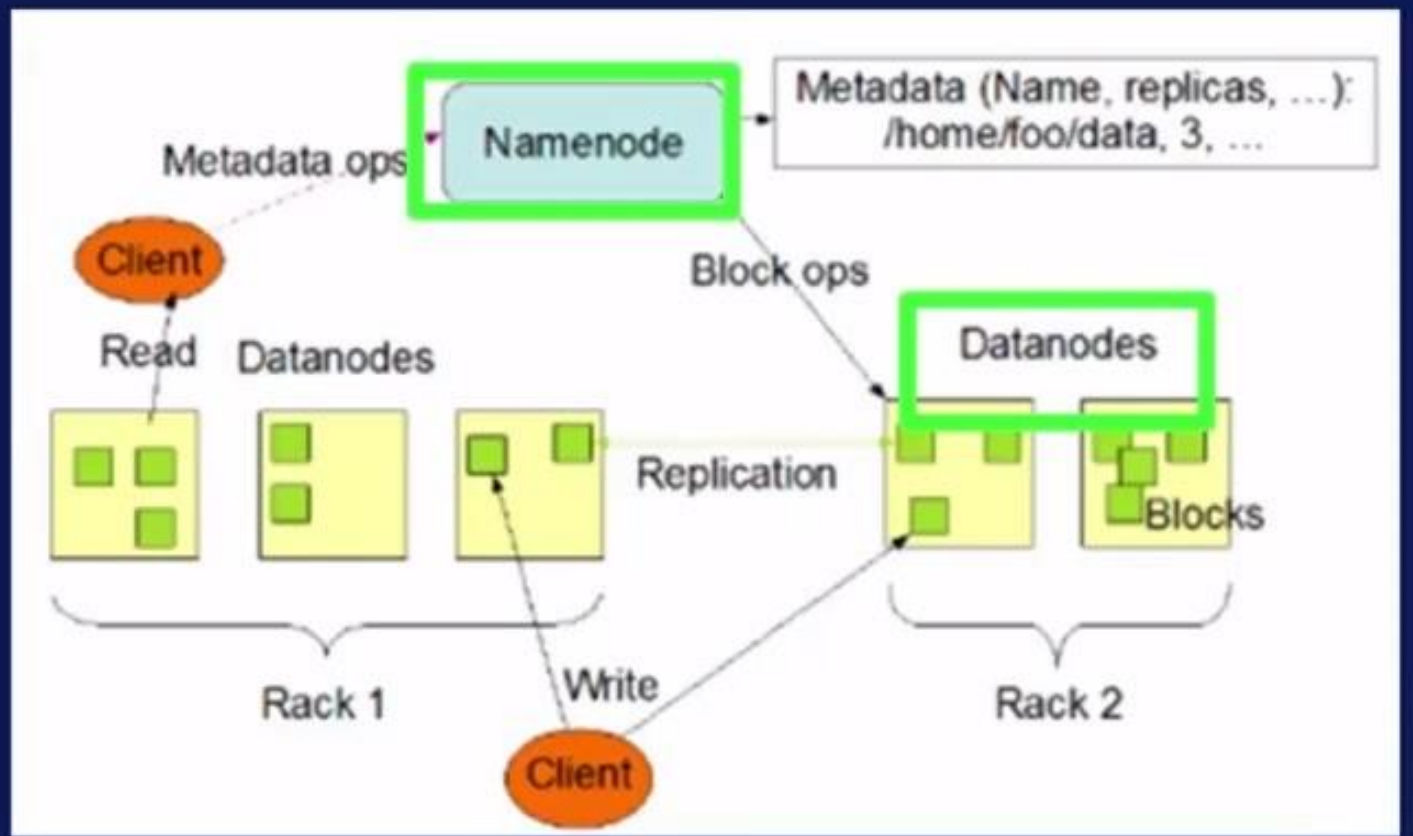


Image Source: UCSD Hadoop Course on Coursera

- Data is divided into blocks of the same size.
- Each block is replicated (default 3)
- Each replica is stored in a different Datanode (some of which may be in different Racks)
- Namenode contains all the information about which blocks are in which Datanode

- NameNode is Master node, but no MasterNode among the DataNodes
- A write request can go to any DataNode
- That node makes the write and then sends the data to its replicas.
- A read request can go to any of the replicas
- Unlike Cassandra, the replica doesn't make consistency check with any of the other replicas
- What if that block on that node is corrupted (so data is not valid)?
- NameNode can have a CheckSum of the data (digital signature) and check the returned data against the CheckSum.

Write/Read in HDFS

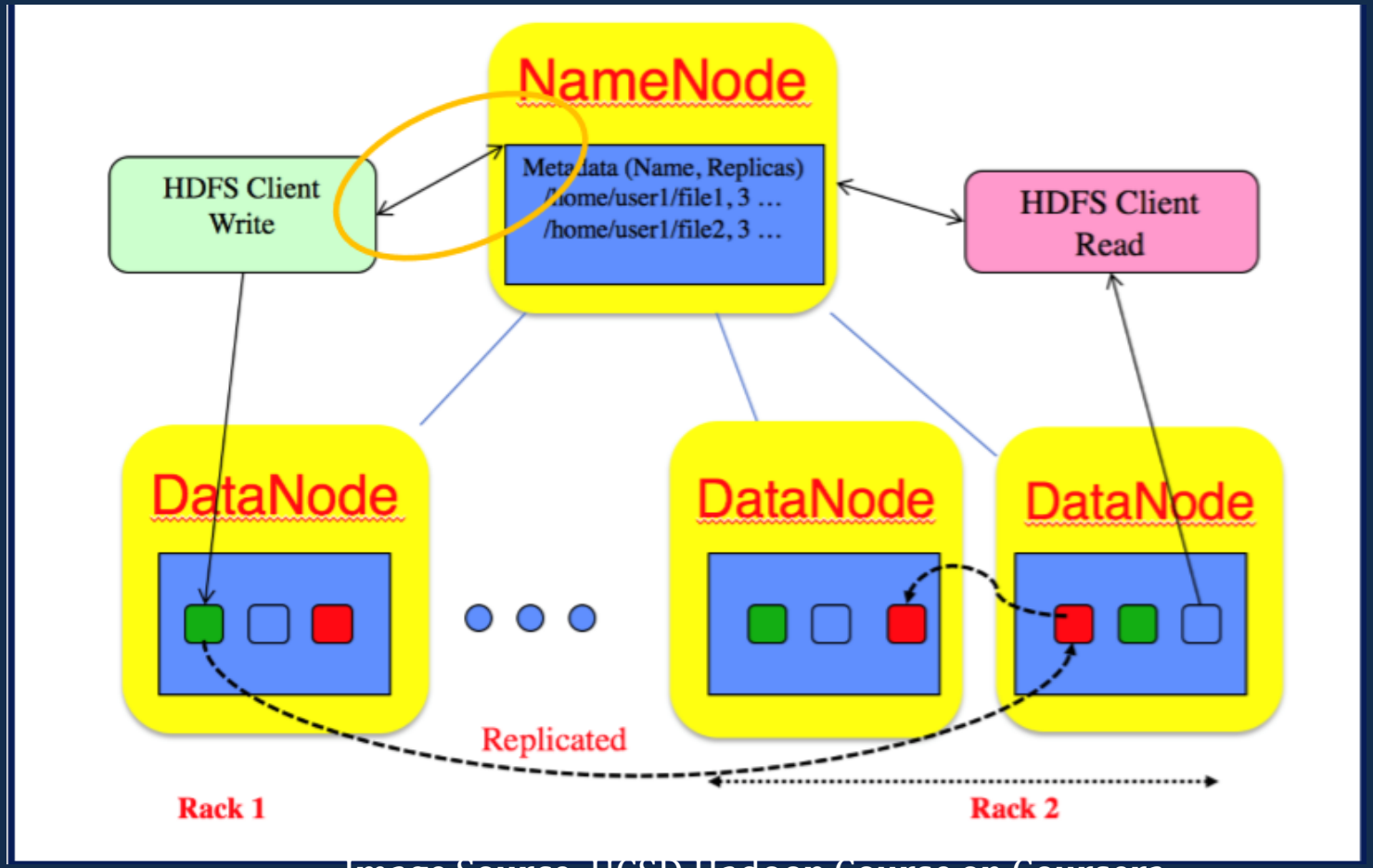


Image Source: UCSD Hadoop Course on Coursera

Definitions

A Cluster is a collection of Data Centers.
A Data Center is a collection of Racks.
A Rack is a collection of Servers.
A Server contains 256 virtual nodes (or vnodes) by default.
A vnode is the data storage layer within a server.

Inverted Index

To understand the most common example of MapReduce you need to understand **the Inverted Index** data model:

An **inverted index** is an **index** data structure storing a mapping from content, such as words or numbers, to its locations in a document or a set of documents. ... A word-level **inverted index** additionally contains the positions of each word within a document.

Doc 1
I did enact Julius Caesar: I was killed i' the Capitol; Brutus killed me.

Doc 2
So let it be with Caesar. The noble Brutus hath told you Caesar was ambitious:

term	docID
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

⟹

term	docID
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2

⟹

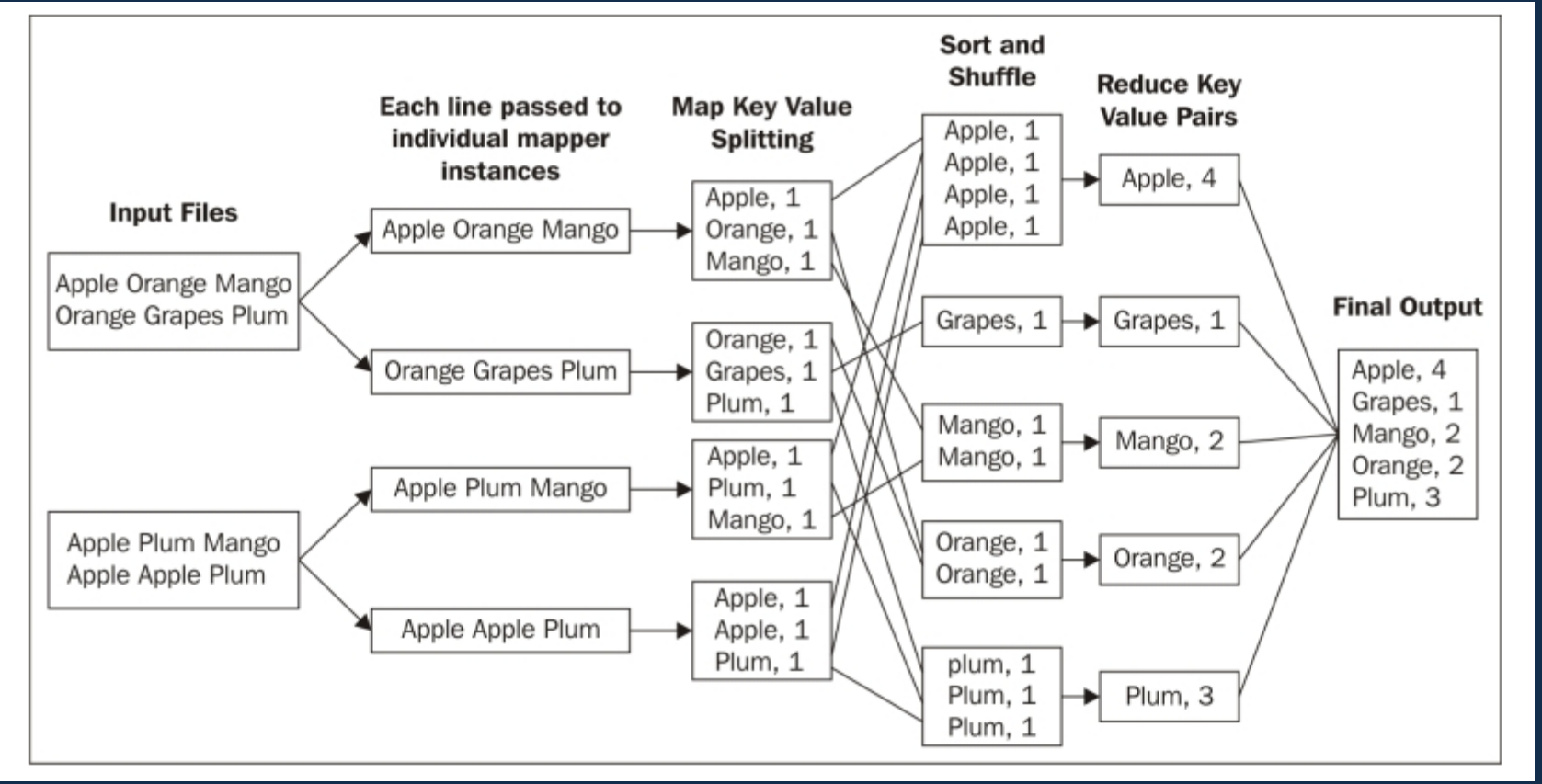
term	doc. freq.	→	postings lists
ambitious	1	→	2
be	1	→	2
brutus	2	→	1 → 2
capitol	1	→	1
caesar	2	→	1 → 2
did	1	→	1
enact	1	→	1
hath	1	→	2
I	1	→	1
i'	1	→	1
it	1	→	2
julius	1	→	1
killed	1	→	1
let	1	→	2
me	1	→	1
noble	1	→	2
so	1	→	2
the	2	→	1 → 2
told	1	→	2
you	1	→	2
was	2	→	1 → 2
with	1	→	2

► **Figure 1.3** Building an index by sorting and grouping. The sequence of terms in each document, tagged by their documentID (left) is sorted alphabetically (middle). Instances of the same term are then grouped by word and then by documentID. The terms and documentIDs are then separated out (right). The dictionary stores the terms, and has a pointer to the postings list for each term. It commonly also stores other summary information such as, here, the document frequency of each term. We use this information for improving query time efficiency and, later, for weighting in ranked retrieval models. Each postings list stores the list of documents in which a term occurs, and may store other information such as the term frequency (the frequency of each term in each document) or the position(s) of the term in each document.

MapReduce: Map and Reduce

- Basic idea: Parallelizing processing on different chunks of data stored on different nodes.
  - The processing function is cloned and 'mapped' to each data chunk (e.g., a web document)
  - For example, the processing function keeps a tally of the number of occurrences of every word in that document.
- Reduce: Consolidating the results of each mapping function into a common unit
  - The final output might be a tally of the occurrence of each word in all the documents
- Intermediate units: shuffling and sorting the results produced by each mapping function
  - E.g. all words from all the documents that start with 'a' to 'g' go to one intermediate unit, all words that start with 'h' to 'p' go to another, and so on, and then they are sorted alphabetically.

MapReduce



From Swizec , ***Hadoop Essentials***

MapReduce Continued

- Computation occurs where the data resides
  - No need to move data to a global processing unit.
- Works only if the task is highly parallelizable
- Any function can be used for map
  - The result has to be something that can easily be reduced
  - For your assignment, the Reduce function assembles a list of anchors for each target web page
- Lot of overhead distributing map functions and reduce functions, synchronizing the different processes, assigning nodes for intermediate stages. (Hence, the need to introduce a resource manager component like YARN)

YARN (Yet Another Resource Negotiator) : ResourceManager

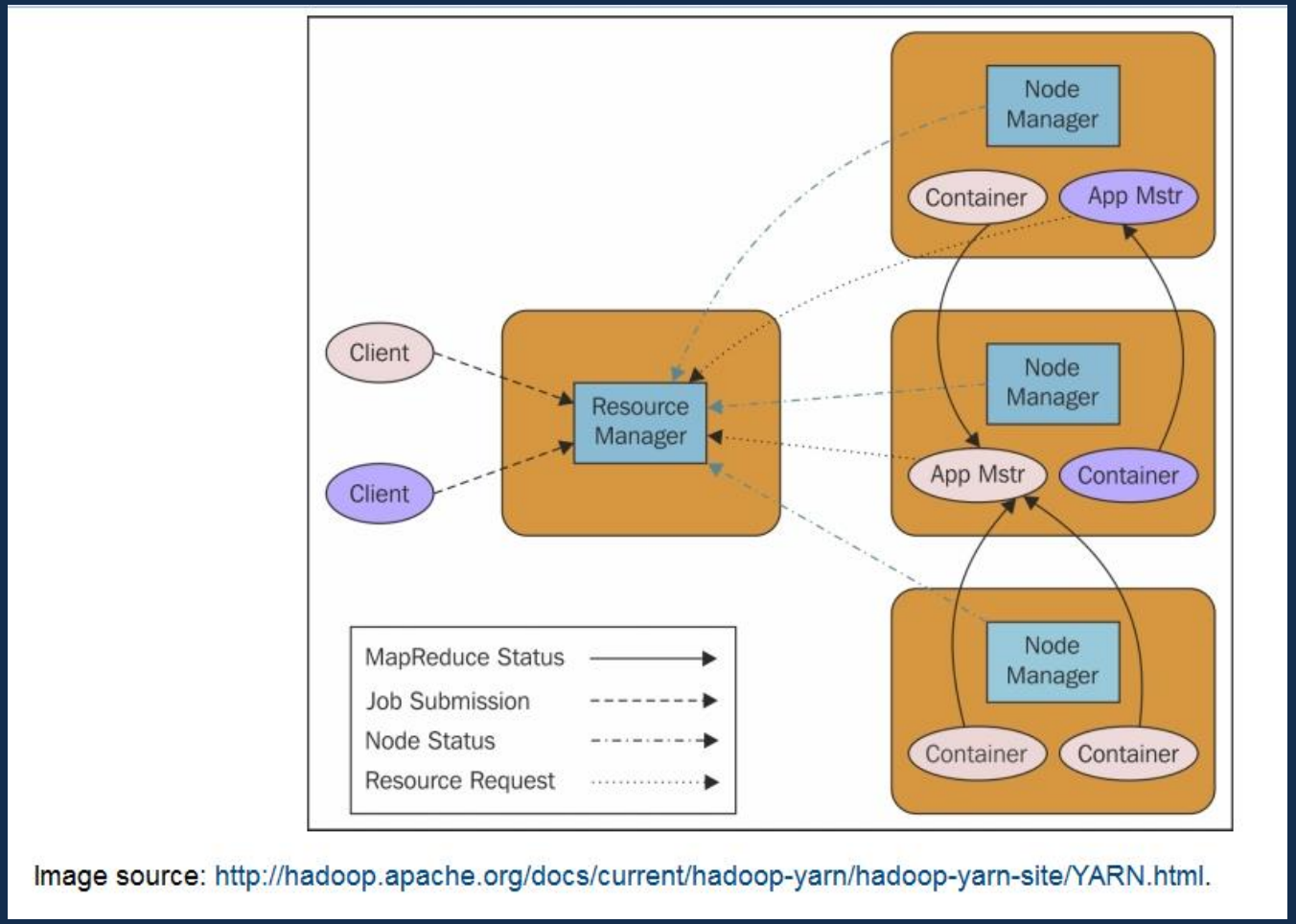


Image source: http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html.

ResourceManager is the master process manager responsible for resource management among the applications in the system. ResourceManager has a scheduler, which allocates the resources to the applications and resource availability which ResourceManager gets from containers that provide information such as memory, disk, CPU, network, and so on.

YARN : NodeManager

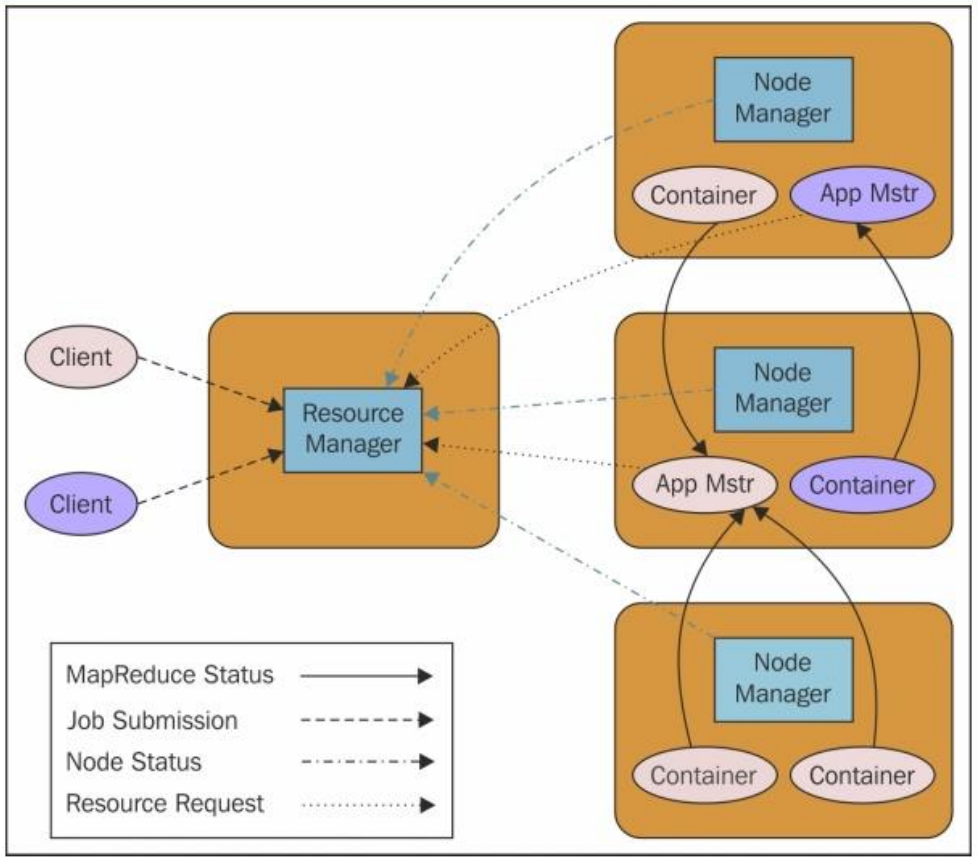


Image source: http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html.

NodeManager is present in all the nodes. It is responsible for containers, authentication, monitoring resource usage, and reports the information to ResourceManager. Similar to TaskTracker, NodeManager sends heartbeats to ResourceManager.

YARN : ApplicationMaster

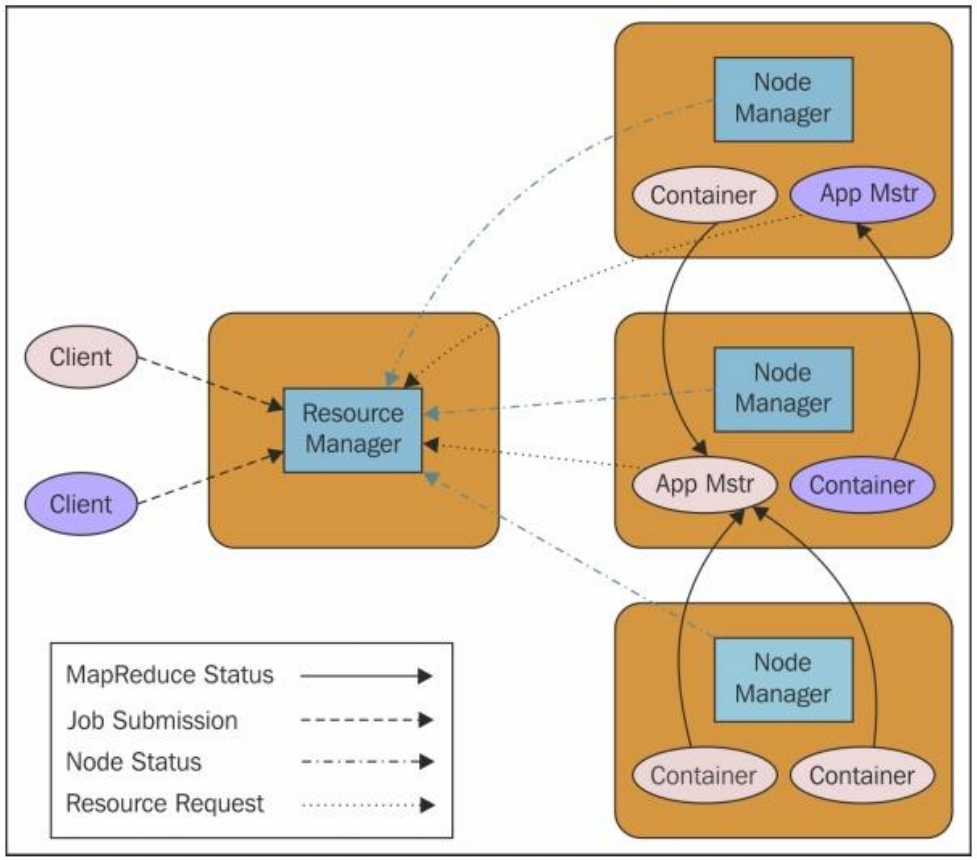


Image source: http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html.

ApplicationMaster is present for each application, responsible for managing each and every instance of applications that runs within YARN. ApplicationMaster coordinates with ResourceManager for the negotiation of the resources and coordinates with the NodeManager to monitor the execution and resource consumption of containers, such as resource allocations of CPU, memory, and so on.

Recap of this week

- Introduction to Hadoop Ecosystem
- HDFS
- MapReduce
- Inverted Spectrum as a data model
- YARN

Next week

- Installing Cloudera Hadoop on Google Cloud
- Interacting with Hadoop components on your instance of Hadoop on Google Cloud
- Discussion of Project 3
- Pig
- Hive
- Flume