Lecture slides of the course Introduction to Big Data

HADOOP FUNDAMENTALS

(Part I)

Lecturer: Dr. Nguyen Ngoc Thao – MSc. Le Ngoc Thanh

Department of Computer Science, FIT, HCMUS

Ho Chi Minh City, September 2018

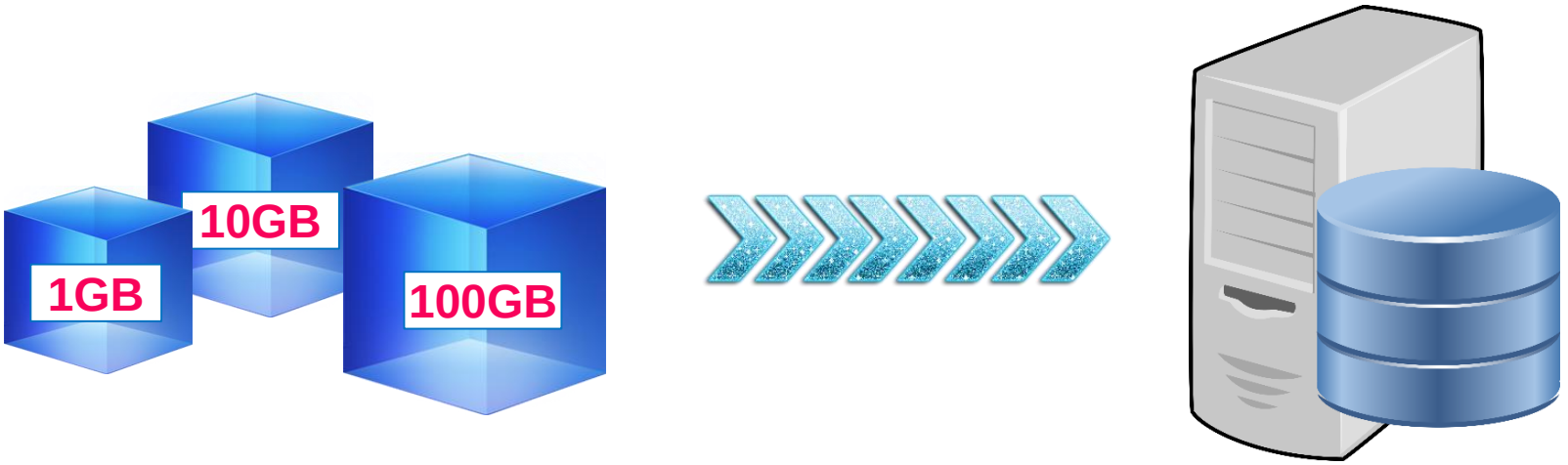
Outline

- What is Apache Hadoop?
  - A Brief History of Hadoop
  - When to Use and not to Use Hadoop
  - Why Learn Hadoop?
- The architecture of Apache Hadoop
  - Pre-Hadoop 2.2. Architecture
  - Hadoop 2.2 Architecture
  - Hadoop 3.0 Architecture

INTRODUCTION TO HADOOP

An essential ecosystem for Big Data

Imagine a scenario where…



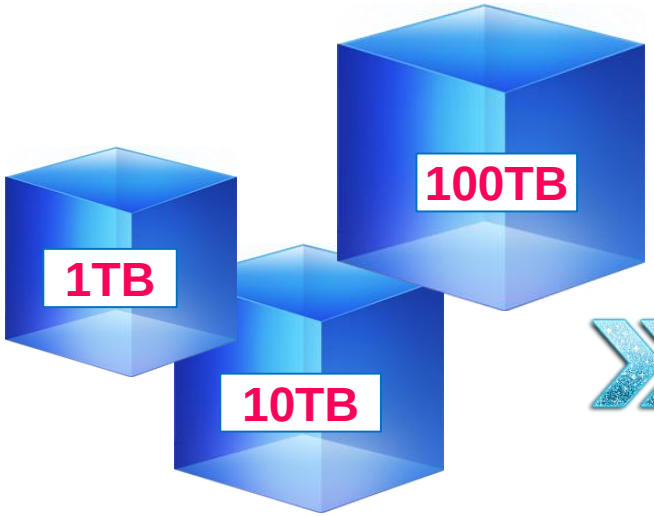
You have 1GB of data that you need to process…. No problem!

Your company starts growing very quickly, and that data grows to 10GB

And then 100GB….

You start to reach the limits of your current desktop computer.

Imagine a scenario where…



So you scale-up by investing in a larger computer, but…

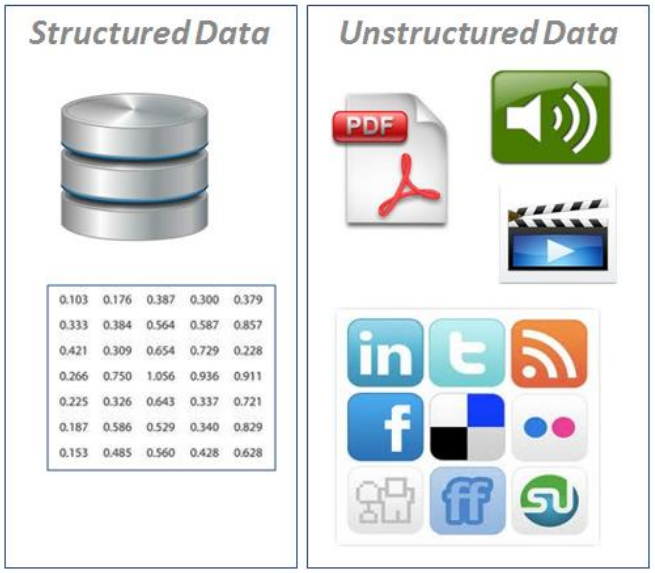
the data increases quickly in a few months

it is required to feed the application with unstructured data

Imagine a scenario where…

You may want

- To derive information from both relational data and unstructured data

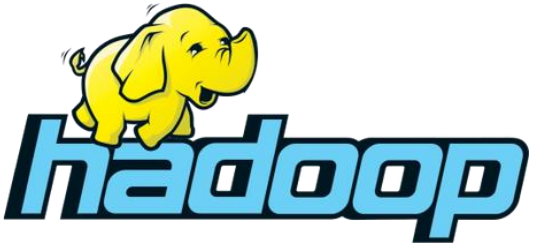


- The derivation is done as soon as possible.

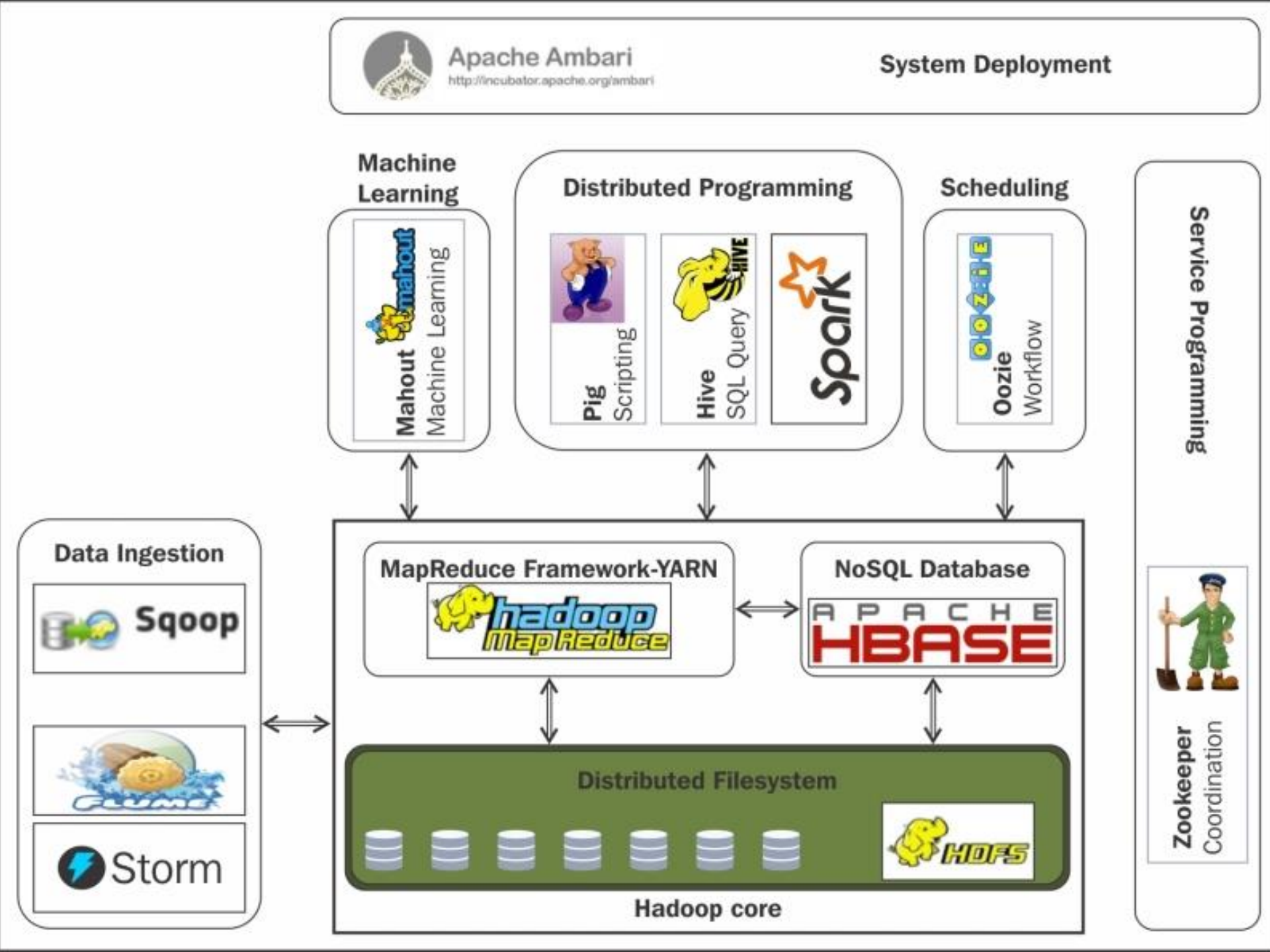
may be your answer!!!

What is Hadoop?

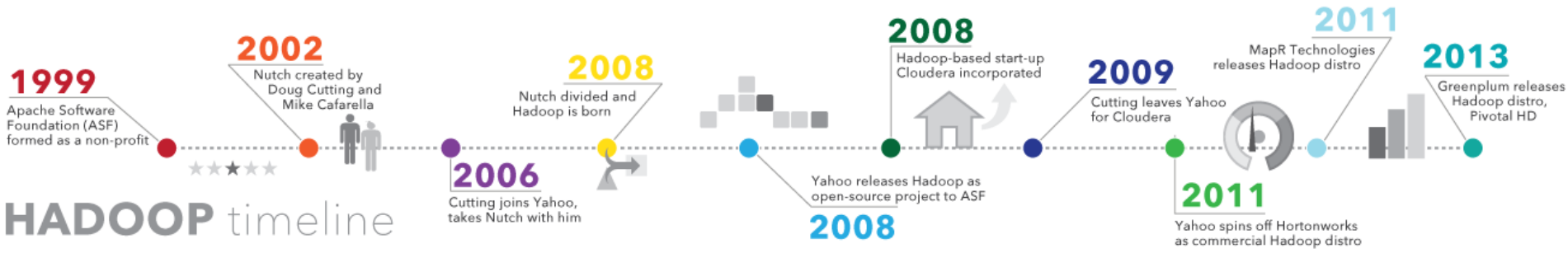
- Open-source software framework maintained by the ASF



- Written in Java
- Optimized for massive amounts of data through distribution
  - A variety of data types (structured, semi-structured, unstructured)
  - Use inexpensive commodity hardware.
- Massive parallel processing with great performance
- Reliability provided through replication


Apache Ambari
http://incubator.apache.org/ambari
System Deployment
Machine Learning
Mahout
Machine Learning
Distributed Programming
Pig
Scripting
Hive
SQL Query
Spark
Scheduling
Oozie
Workflow
Service Programming
Zookeeper
Coordination
Data Ingestion
Sqoop
Flume
Storm
MapReduce Framework-YARN
hadoop
MapReduce
NoSQL Database
APACHE
HBASE
Distributed Filesystem
HDFS
Hadoop core

A brief history of Hadoop



- 2002 – Dough Cutting and Mike Caferella created the Nutch project.
- 2006 – Cutting joined Yahoo and the Nutch project was divided.
  - The web crawler portion remained as Nutch.
  - The distributed computing and processing portion became Hadoop, which was later released as an open-source project.
- 2008 – Yahoo released Hadoop as an open-source project to ASF.

Source: https://www.sas.com/en_us/insights/big-data/hadoop.html

A brief history of Hadoop

- The competition of sorting a terabyte (TB) of data.
- In April 2008, Hadoop sorted an entire TB in 209 seconds by running on a 910-node cluster, beat the previous year's winner of 297 seconds.
- In November 2008, Google's MapReduce implementation sorted 1TB in 68 seconds.
- Then, in April 2009, a team in Yahoo! used Hadoop to sort 1TB in 62 seconds.

Why is Hadoop important?

Storage and processing speed

Store and process huge amounts of disparate data quickly

Flexibility

Store as much data as you want

Decide how to use it later

Computing power

The more computing nodes, the more processing power

Low cost

Open-source framework

Commodity hardware

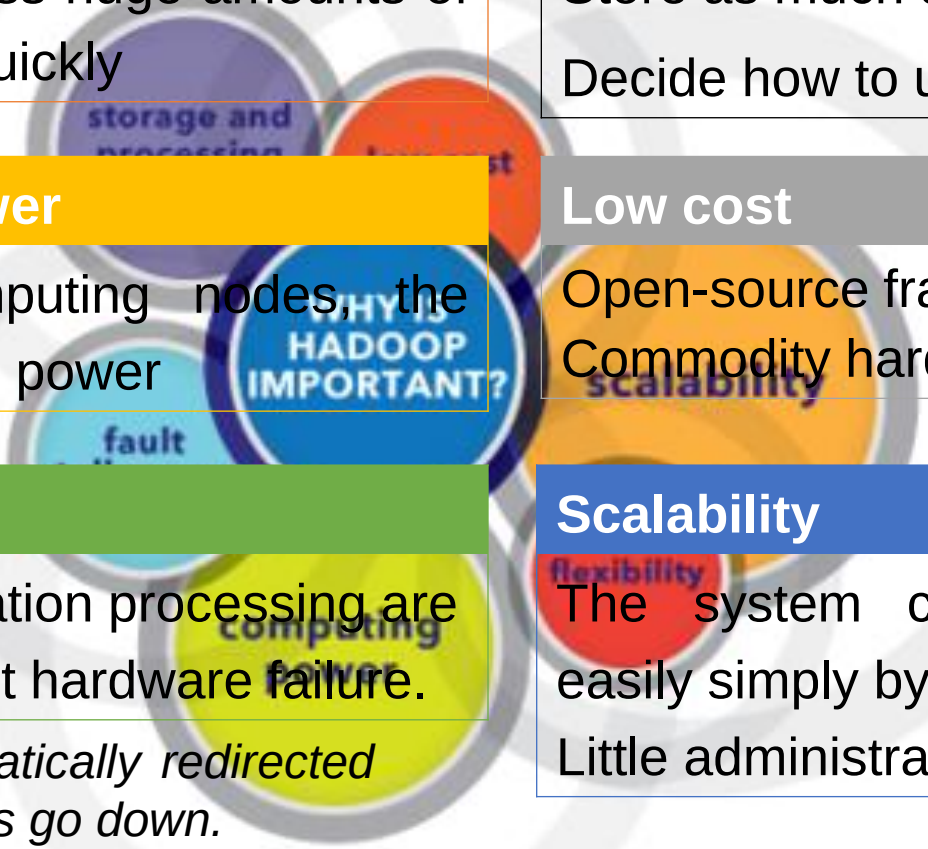
Fault tolerance

Data and application processing are protected against hardware failure.

Jobs are automatically redirected when their nodes go down.
Multiple copies of all data are stored automatically.

Scalability

The system can be expanded easily simply by adding nodes.
Little administration is required.



Consideration when using Hadoop

Not a good match for all problems

Good for simple information requests and problems dividable into independent units. Inefficient for iterative/interactive analytic tasks.

Widely acknowledged talent gap

Difficult to find programmers having sufficient Java skills to be productive with MapReduce.

Data security

Kerberos authentication protocol is a great step toward making Hadoop secure.

Full-fledged data management and governance

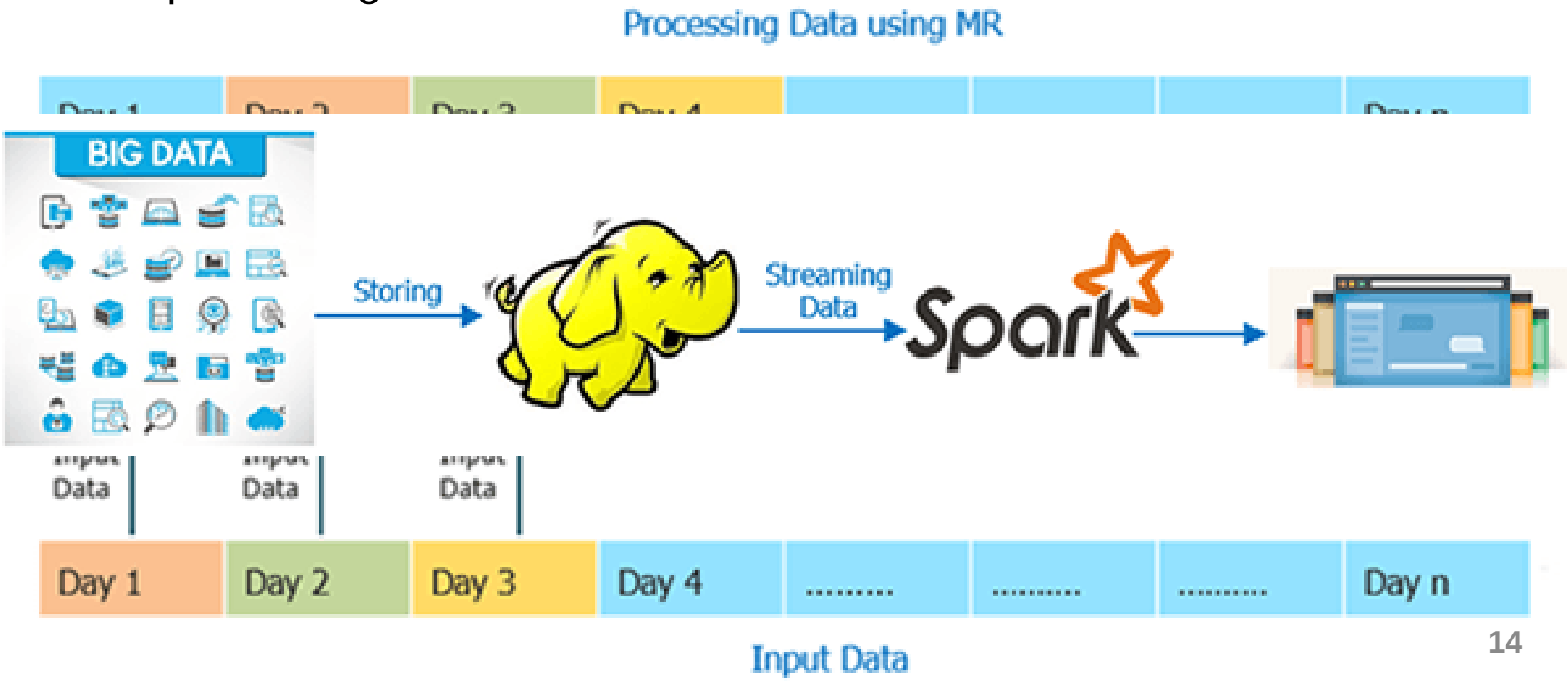
Lack of easy-to-use, full-feature tools for data management, data cleansing, governance and metadata, especially tools for data quality and standardization.

5 Reasons When to and When not to use Hadoop

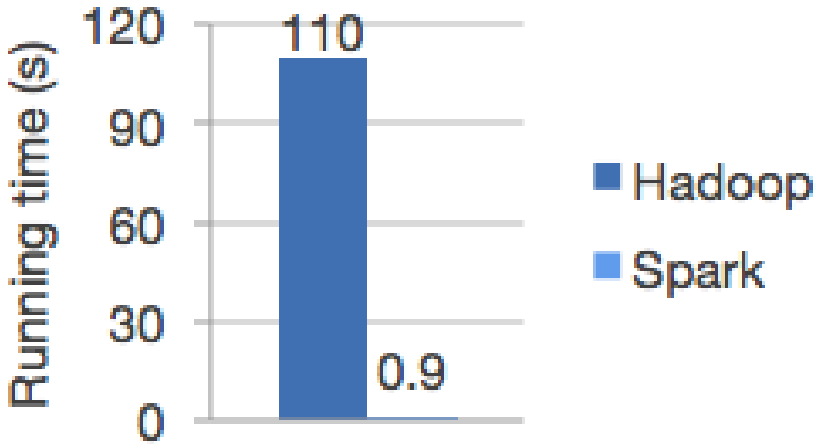
Source: https://www.edureka.co/blog/5+Reasons-when-to-use-and-not-to-use-hadoop/

When not to use Hadoop?

- **Real-time analytics:** Results are expected to come quickly.
  - Hadoop works on batch processing → response time is high
  - Store the Big data in HDFS and mount Spark over it to make the processing real time

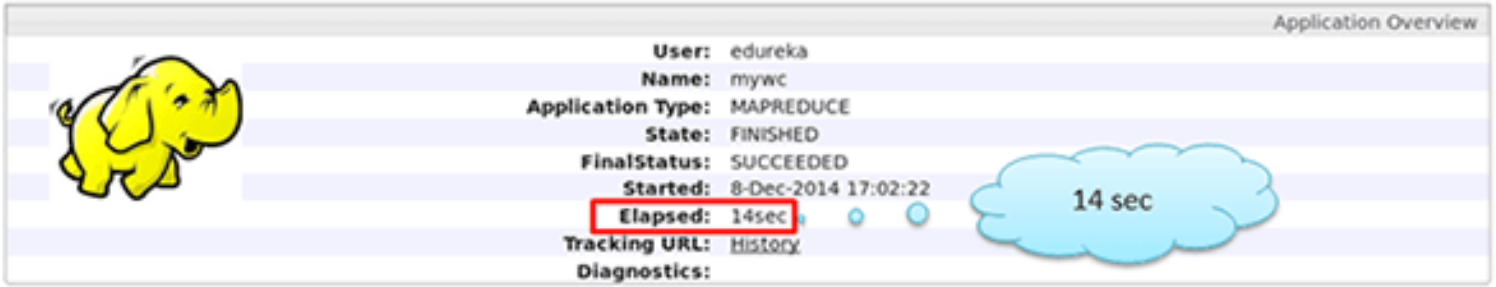


Hadoop vs. Spark



Spark runs programs up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk.

A simple example of line processing in Hadoop and Spark



Logistic regression in Hadoop and Spark


```
14/12/08 04:10:06 INFO spark.SparkHadoopWriter: attempt_201412080410_0005_m_000000_5: Committed
14/12/08 04:10:06 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 5.0 (TID 5) in 296 ms on localhost (1/1)
14/12/08 04:10:06 INFO scheduler.TaskSchedulerImpl: Removed TaskSet 5.0, whose tasks have all completed, from pool
14/12/08 04:10:06 INFO scheduler.DAGScheduler: Stage 5 (saveAsTextFile at <console>:14) finished in 0.279 s
14/12/08 04:10:06 INFO spark.SparkContext: Job finished: saveAsTextFile at <console>:14, took 0.626043309 s
14/12/08 04:10:06 INFO executor.Executor: Finished task 0.0 in stage 5.0 (TID 5). 826 bytes result sent to driver
wordCounts: Unit = ()
```

When not to use Hadoop?

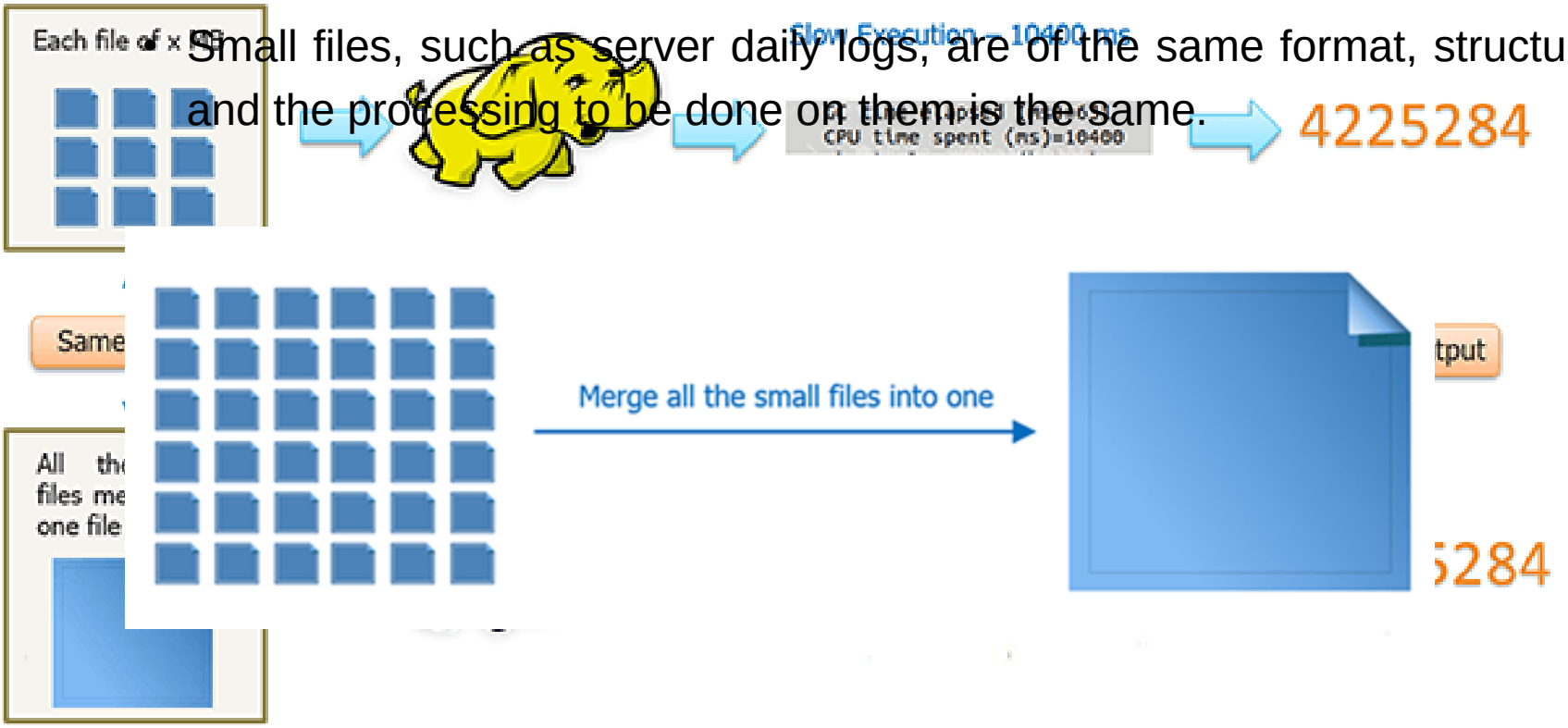
- **Not a replacement for existing infrastructure**
  - Hadoop is not a replacement for existing infrastructures of data processing but Hadoop can be used along with them instead.
  - The data can be stored in HDFS, processed and transformed into structured manageable data, and then sent to relational database technologies for BI, decision support, reporting, etc.

Hadoop is not going to replace your database, but your database is not likely to replace Hadoop either.

Different tools for different jobs, as simple as that.

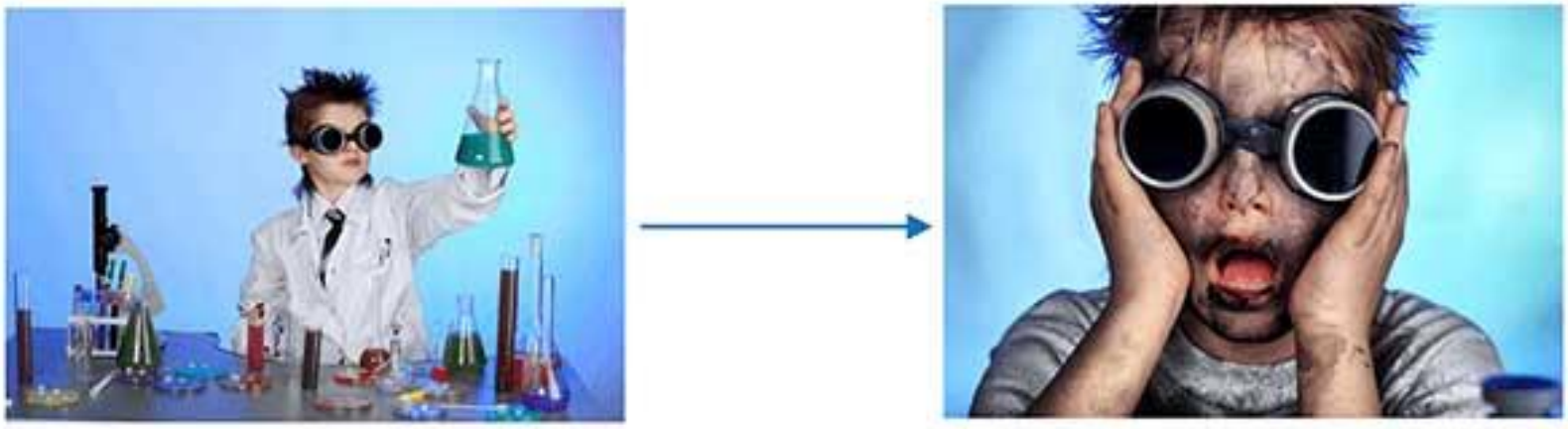
When not to use Hadoop?

- **Multiple smaller datasets**
  - Hadoop is not recommended for small-structured datasets since it can be costlier than other tools (e.g. MS Excel, RDBMS, etc.).
  - Merge all small files into one big file and then run MapReduce on it.
  - Small files, such as server daily logs, are of the same format, structure and the processing to be done on them is the same.



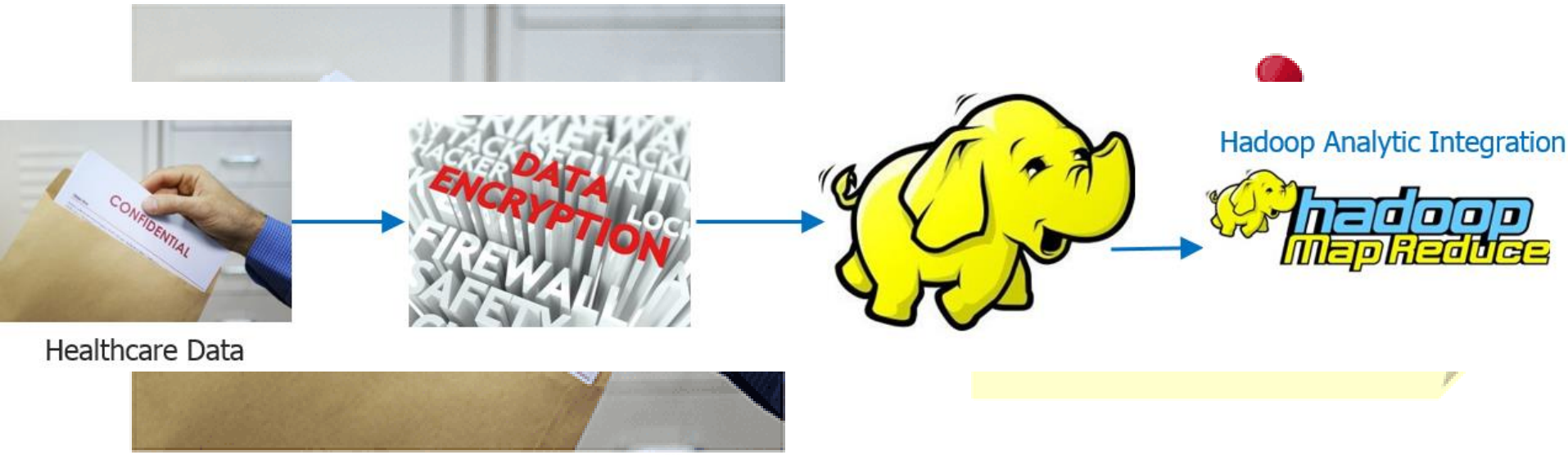
When not to use Hadoop?

- **Novice Hadoopers**
  - Learning Hadoop and its eco-system tools and deciding which technology suits your need is again a different level of complexity.



When not to use Hadoop?

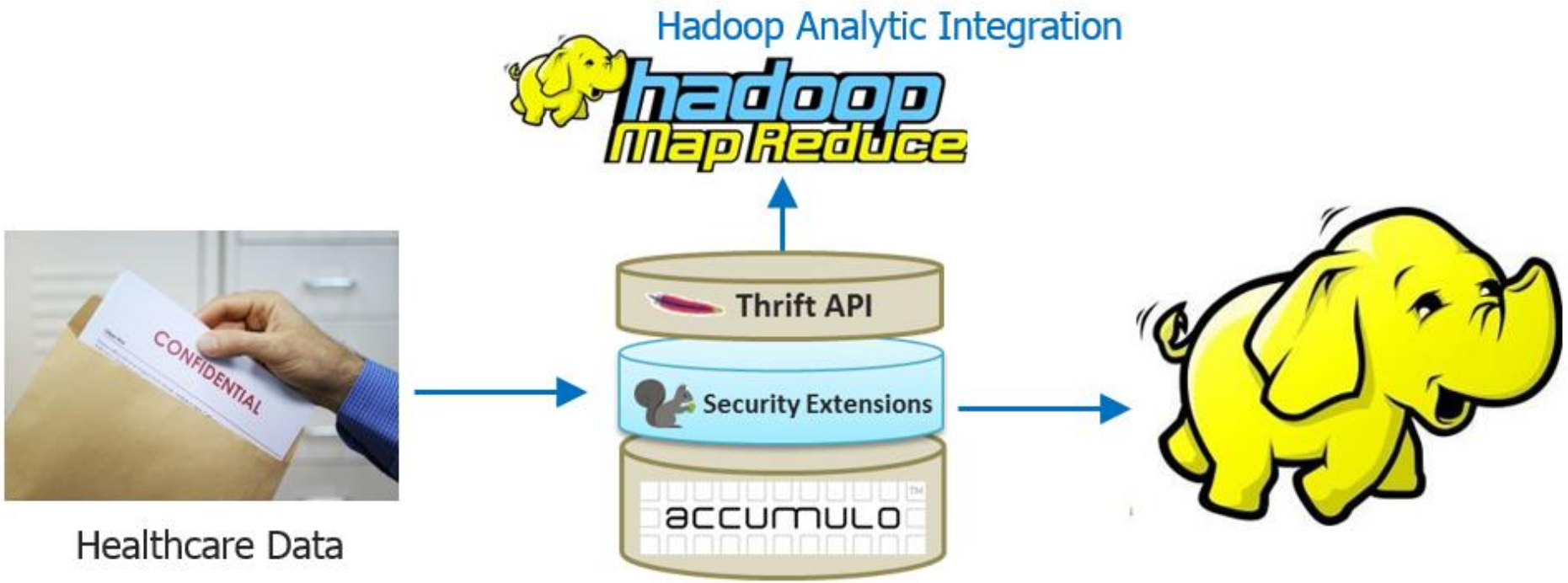
- **Where security is the primary concern**
  - Enterprises dealing with sensitive data are not able to move towards implementing Big data projects and Hadoop quickly.
  - Encrypt the data while moving to Hadoop, then use it for further processing to get relevant insights.



"Example Health-care data used by Insurance companies to calculate premium"

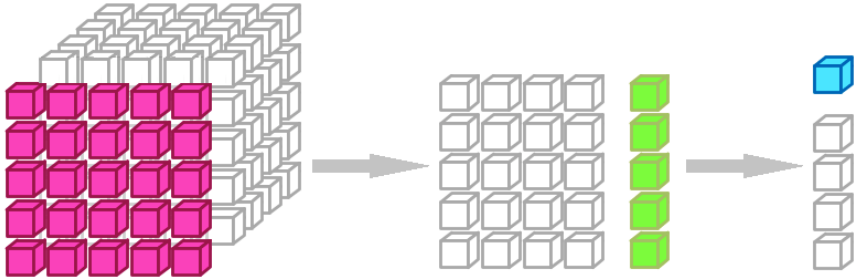
When not to use Hadoop?

- **Where security is the primary concern**
  - Use Apache Accumulo on top of Hadoop.
  - This is a sorted, distributed key/value store that provides robust, scalable data storage and retrieval. Cell-based access control.



In summary, what is Hadoop not for?

- Not for OLTP, not for OLAP/DSS, good for Big Data
  - Hadoop is not a replacement for existing RDBMS technology but complements them.



- Not to process transactions (random access)
- Not good when work cannot be parallelized.
- Not good for low latency data access.
- Not good for processing lots of small files.
- Not good for intensive calculations with little data.

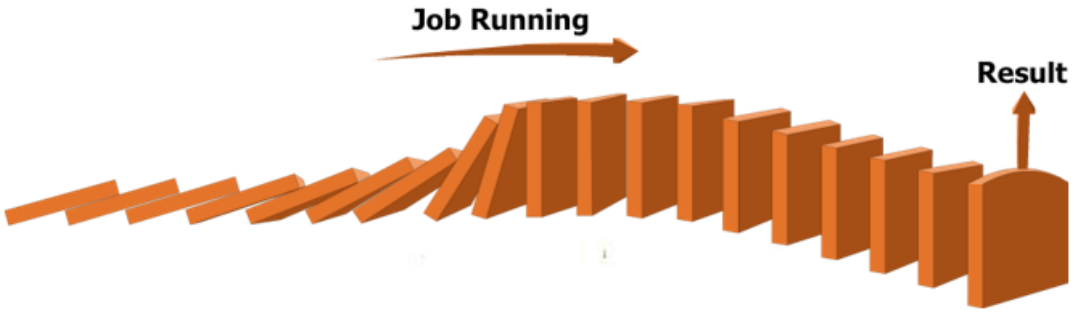
When to use Hadoop?

- **Data size and data diversity**

→ The data set is huge in size i.e. several Terabytes or Petabytes

→ Your have different types of data : structured, semi-structured and unstructured

→ You are not in a hurry for Answers



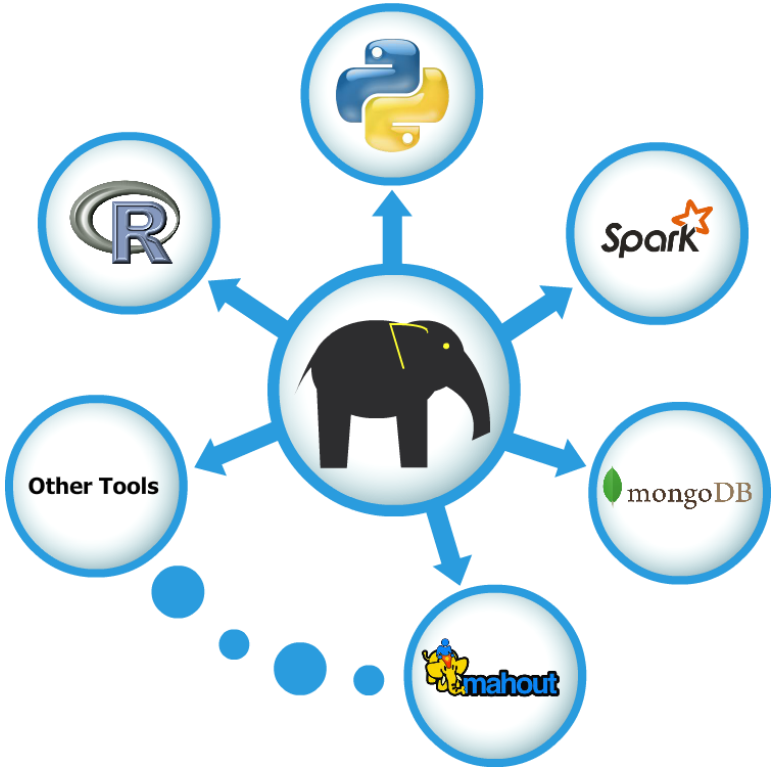
When to use Hadoop?

- **Future planning**
  - Build a small or medium cluster for the data available at present and scale up the cluster in future.

When to use Hadoop?

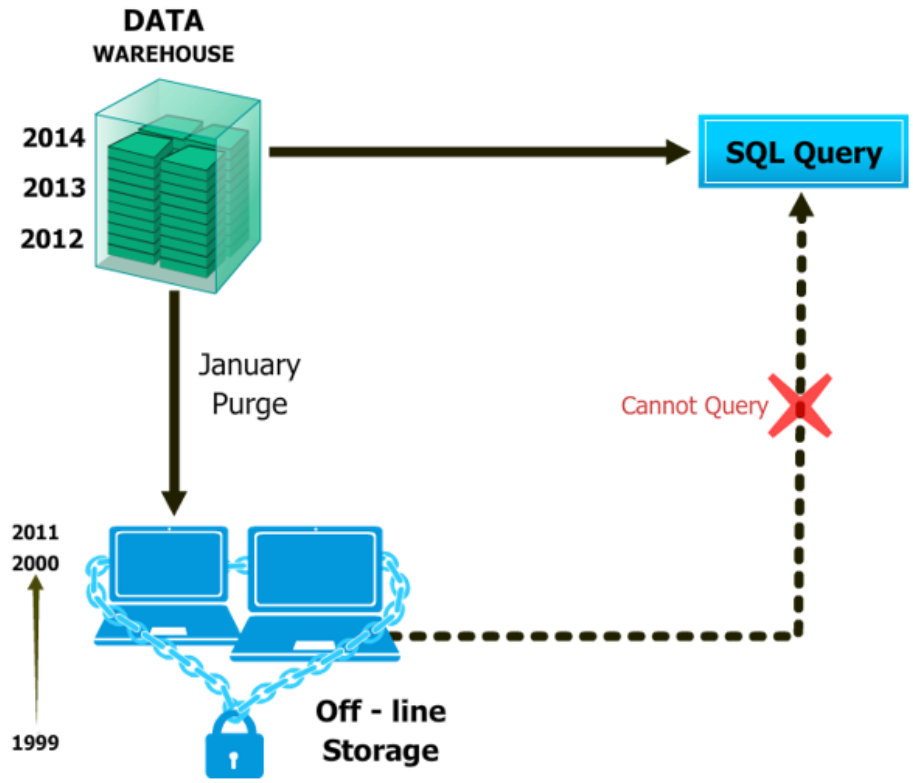
- **Multiple frameworks for Big Data**
  - Hadoop can be integrated with multiple analytic tools to get the best out of it.

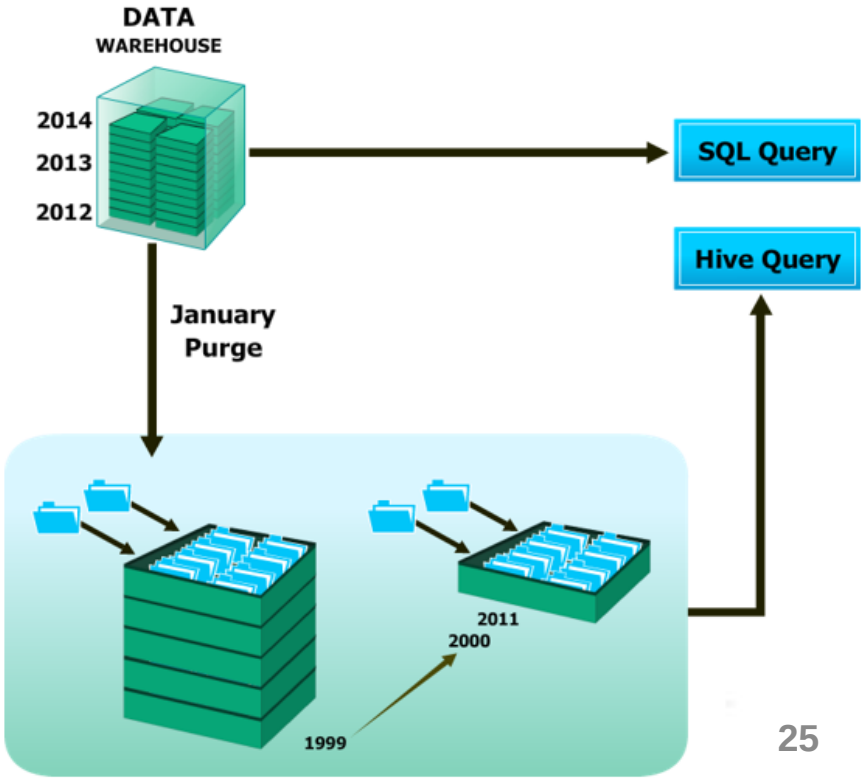
- **Mahout** for Machine-Learning,
- **R and Python** for analytics and visualization
- **Spark** for real time processing,
- **MongoDB and Hbase** for Nosql database
- **Pentaho** for BI, etc.



When To Use Hadoop?

- **Lifetime data availability**
  - With Hadoop's scalability, the stored data can be live and running forever.
  - The cluster size can be increased unlimitedly by adding nodes to it.





Keys to successfully adopting Hadoop

Business users and analysts have access to as much data as possible

Regulatory requirements like data privacy must still be respected.

Results are accessible through standard tools in an organization

Hadoop developers should expose their logic so that results are easily consumed and reusable.

Governance requirements for the data stored in Hadoop

Data audit for both RDBMS and Hadoop are possible.

Keys to successfully adopting Hadoop

Should not try to find an open-ended problem

This kind of problem has neither clearly defined milestones nor measurable business value.

Working with business's leaders

Businesses want to see value from their IT investments, and with Hadoop it may come in a variety of ways.

Examine the perspectives of people and processes that are adopting Hadoop in the organization

Hadoop deployments will be successful when adopters make effort to support data science by fostering experimentation and data exploration

5 Reasons When to and When not to use Hadoop

Source: https://www.edureka.co/blog/5+Reasons-when-to-use-and-not-to-use-hadoop/

Big Data case studies with Hadoop

Log data analysis

Risk Modeling

Data warehouse modernization

Fraud Detection

Social sentiment analysis

Graph analysis

Image classification

The more data, the better decisions and predictions, and then the better outcomes.

Examples of Hadoop in action

中国移动通信
CHINA MOBILE

10x
The Data

1/5
The Cost

The New York Times

4TB in 1.5TB

24Hours

Examples of Hadoop in action

About **200 million** text pages

Use Hadoop to distribute the workload for loading this information into memory.

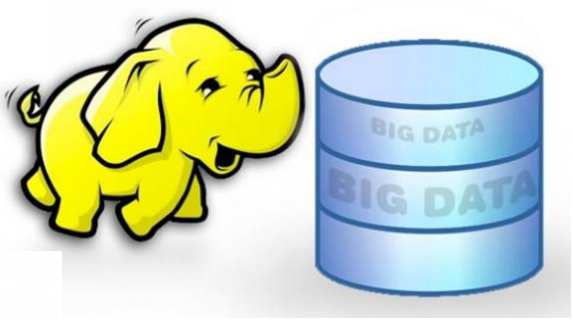
5.5-million-page corpus from the IBM Intranet

A search engine that leverage a variety of open supply platforms and tools, including

- Nutch crawler
- Hadoop MapReduce framework, and
- Lucene indexing engine

Examples of Hadoop in action

- There are many internet or social network companies implementing Hadoop.



Why learn Hadoop?

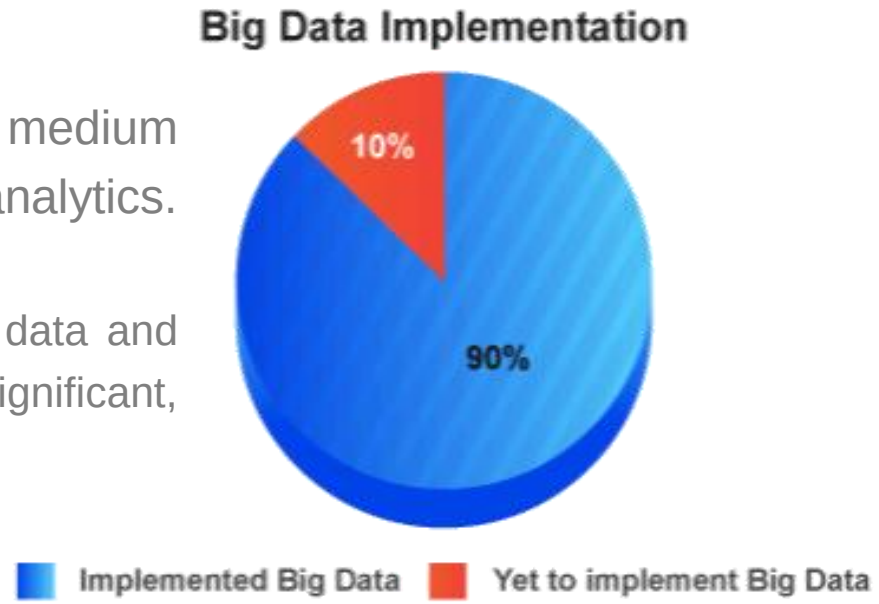
Source: https://www.edureka.co/blog/5-reasons-to-learn-hadoop

Why learn Hadoop?

- **Better career:** Hadoop supports means to ramp up your career and gives you advantages including
  - Accelerated career growth
  - Increased pay package due to Hadoop skill

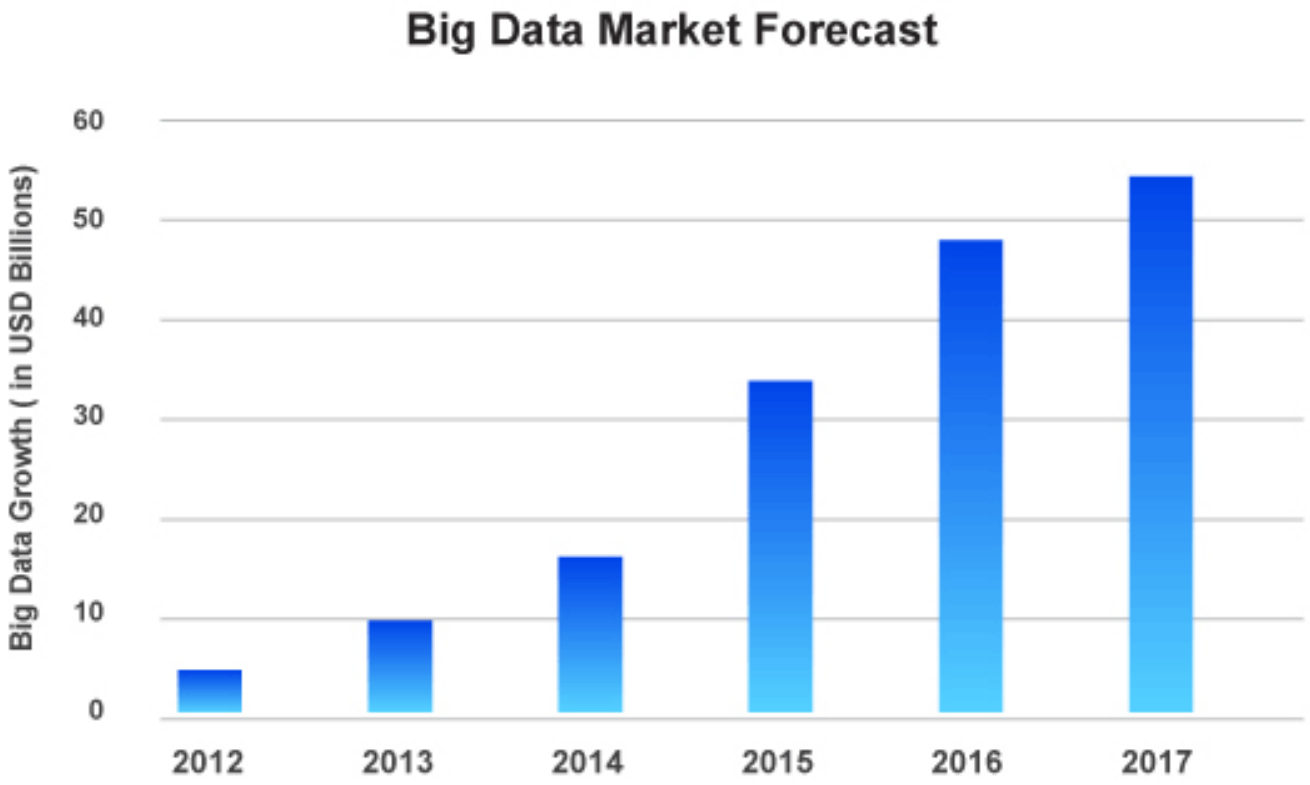
About 90% of global organizations report medium to high levels of investment in big data analytics. (Forbes, 2015).

About 2/3 of respondents report that big data and analytics initiatives have had a significant, measurable impact on revenues.



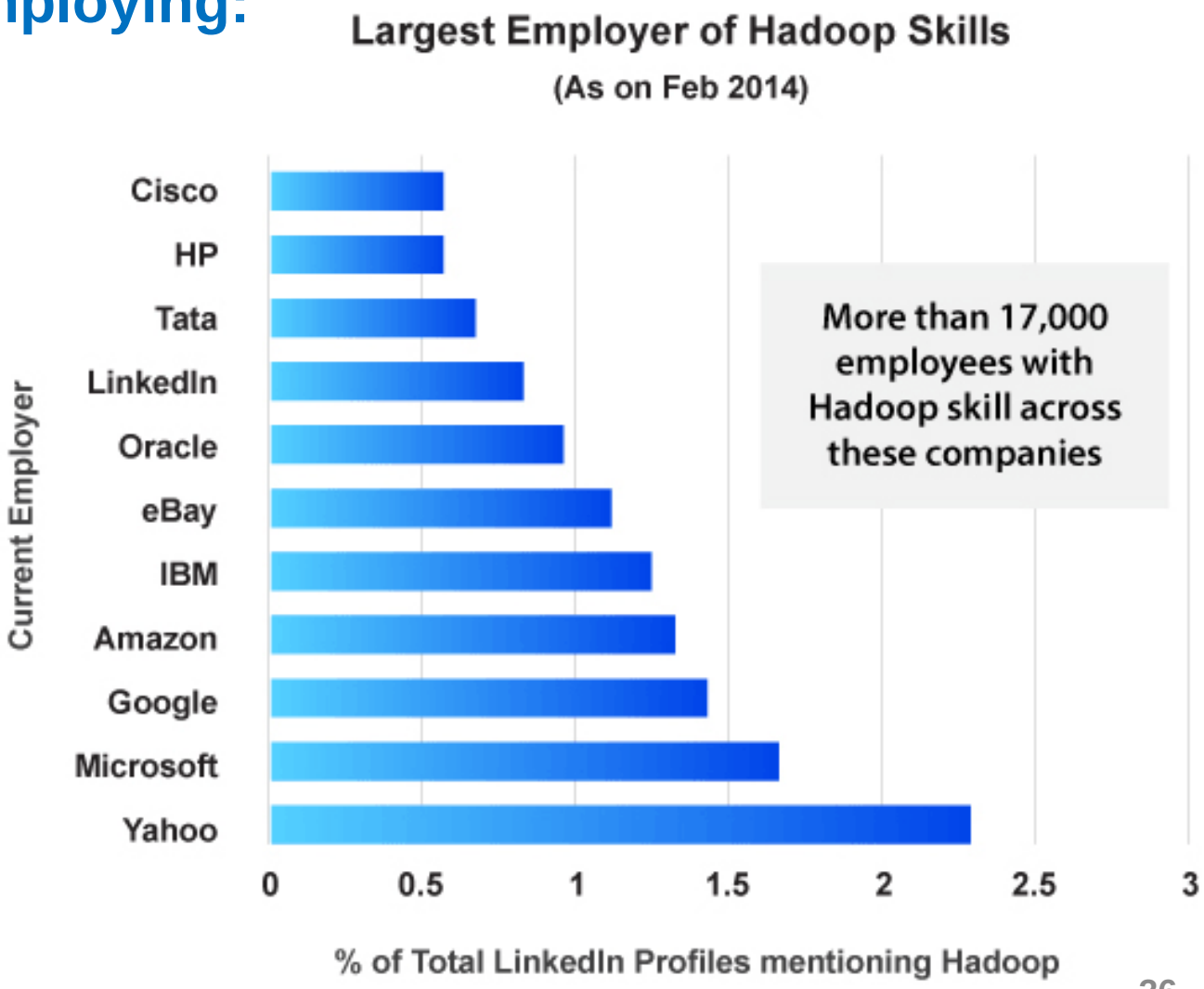
Why learn Hadoop?

- **Better job opportunities:** Hadoop has the potential to improve job prospects whether you are a fresher or an experienced professional.



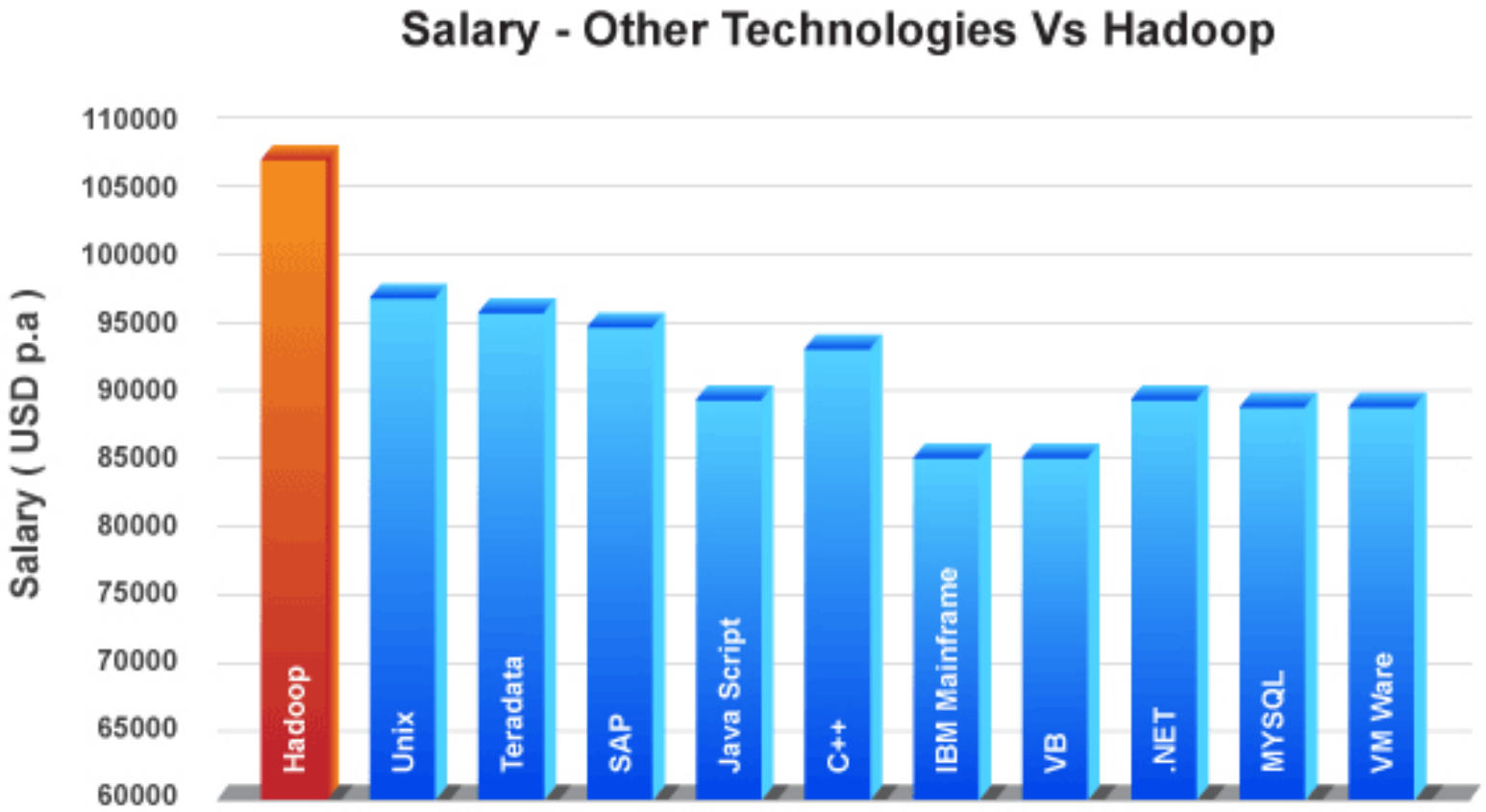
Why learn Hadoop?

- **Look who is employing:**

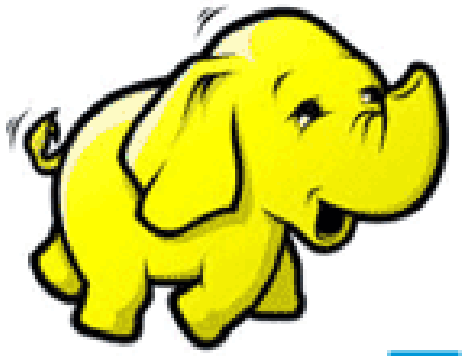


Why learn Hadoop?

- **Big Data and Hadoop equal Big Bucks!**



Top Hadoop Technology companies



KARMASPHERE

Datameer

DELL

amazon web services™

Pivotal

IBM

SUPERMICRO

cloudera

DATASTAX

Zettaset

hadapt

MAPR TECHNOLOGIES

pentaho

Hortonworks

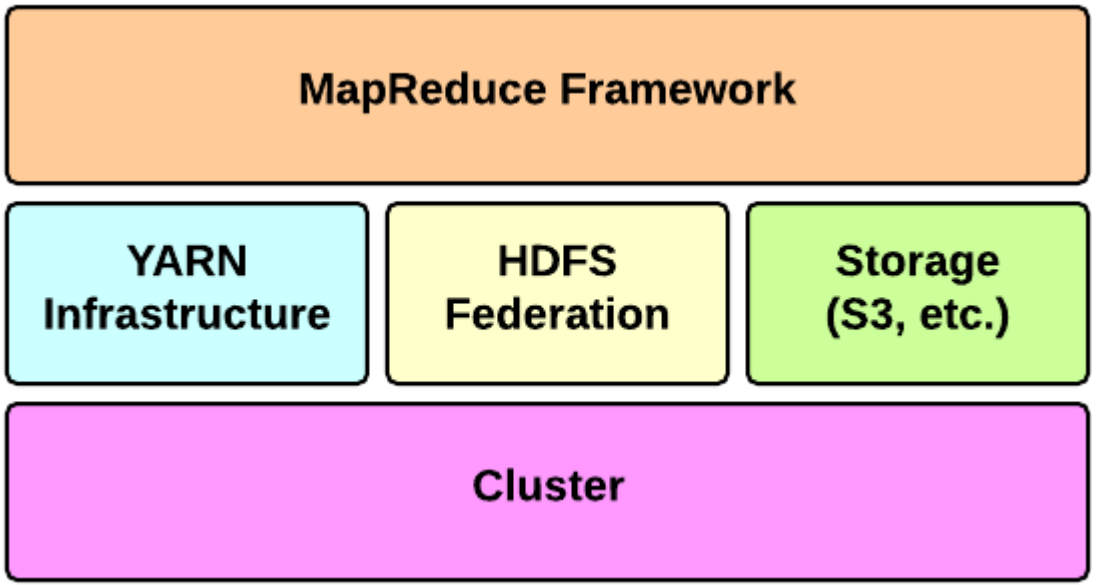
Where to learn Hadoop?

coursera

tutorialspoint
SIMPLYEASYLEARNING

edureka!

cloudera®



HADOOP ARCHITECTURE

HDFS, MapReduce, YARN,…

Terminology review: Node

- A **node** is simply a computer.

Node 1

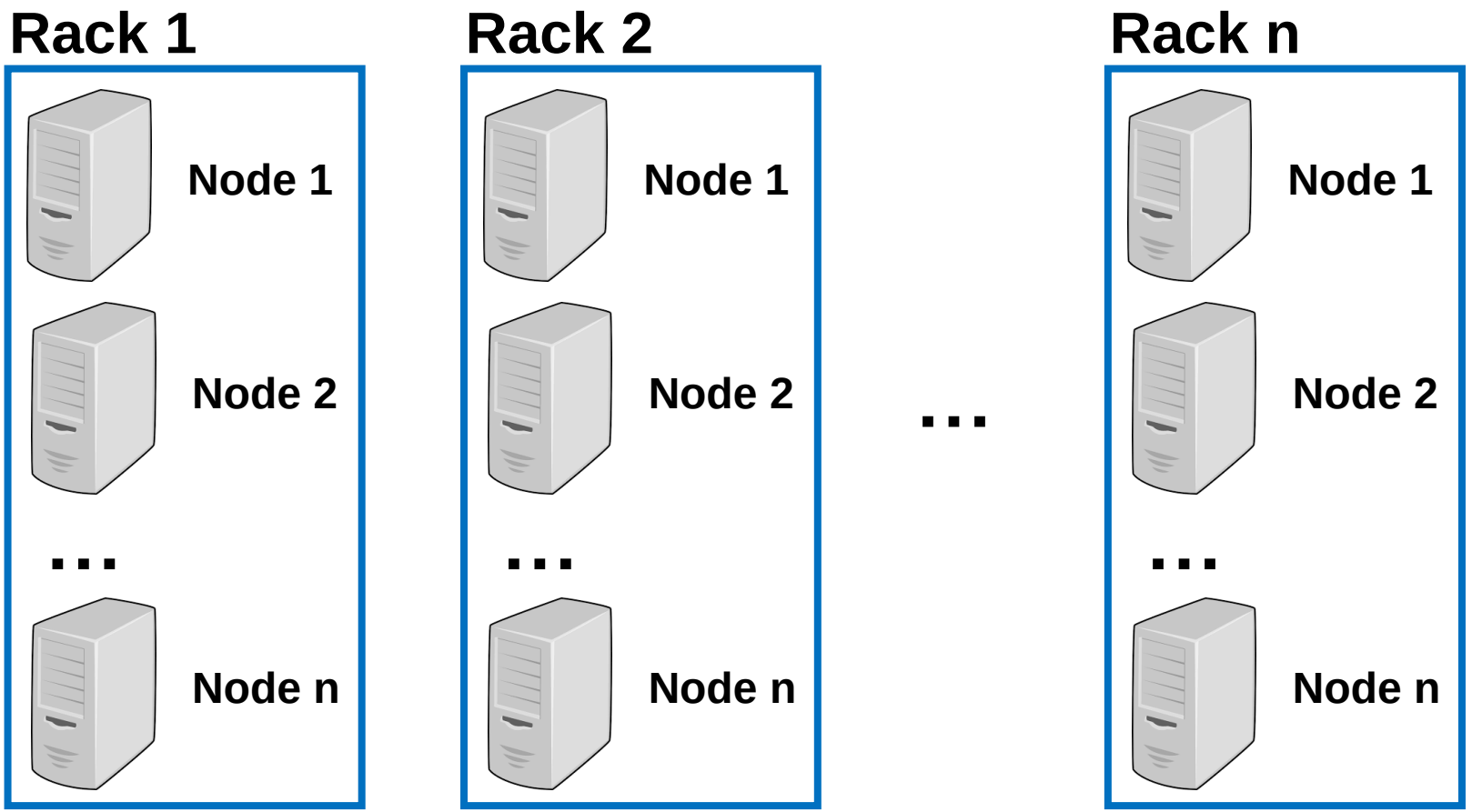
Node 2

...

Node n

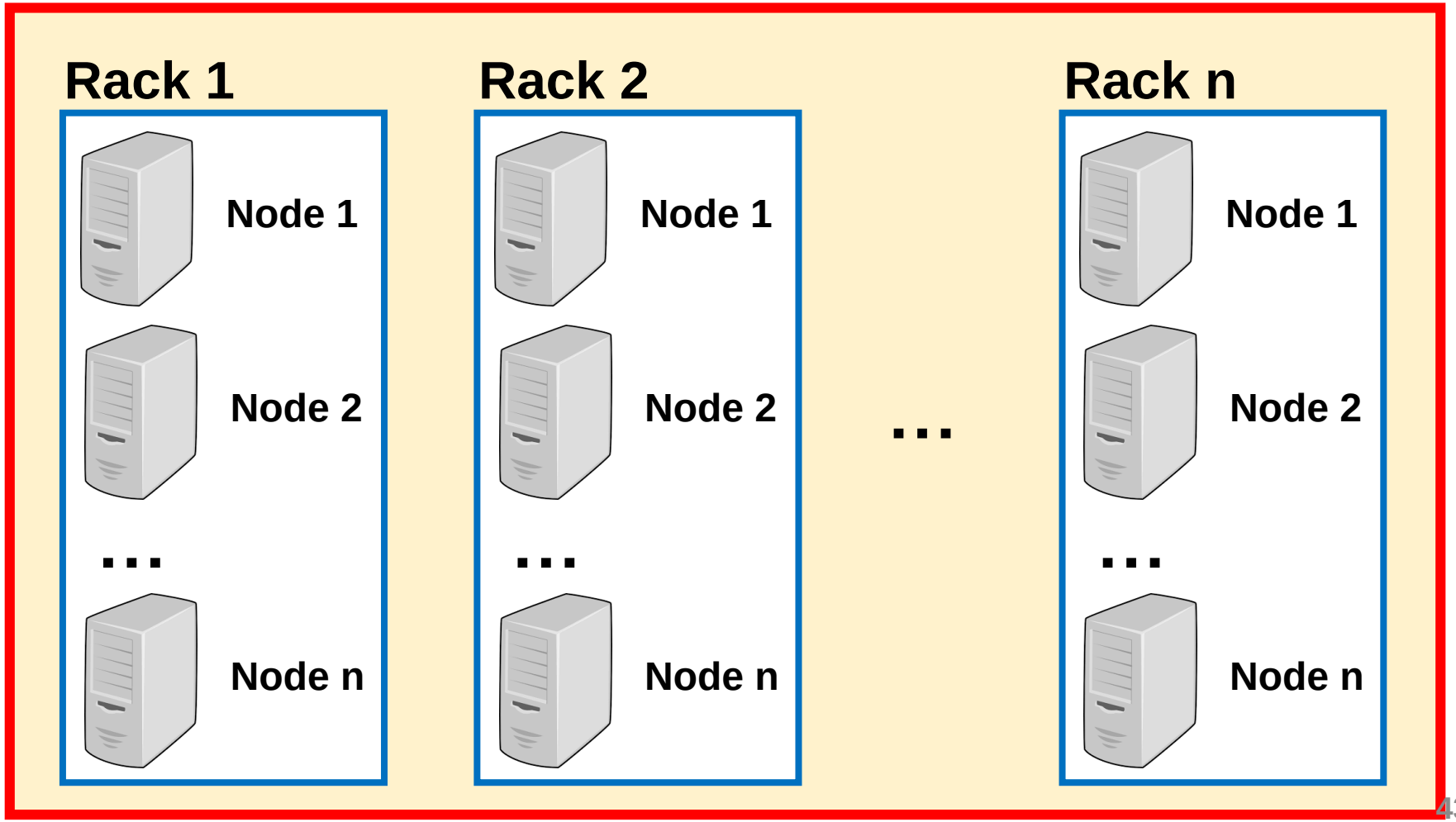
Terminology review: Rack

- A **rack** consists of 30 or 40 nodes physically stored close together and all connected to the same network switch.

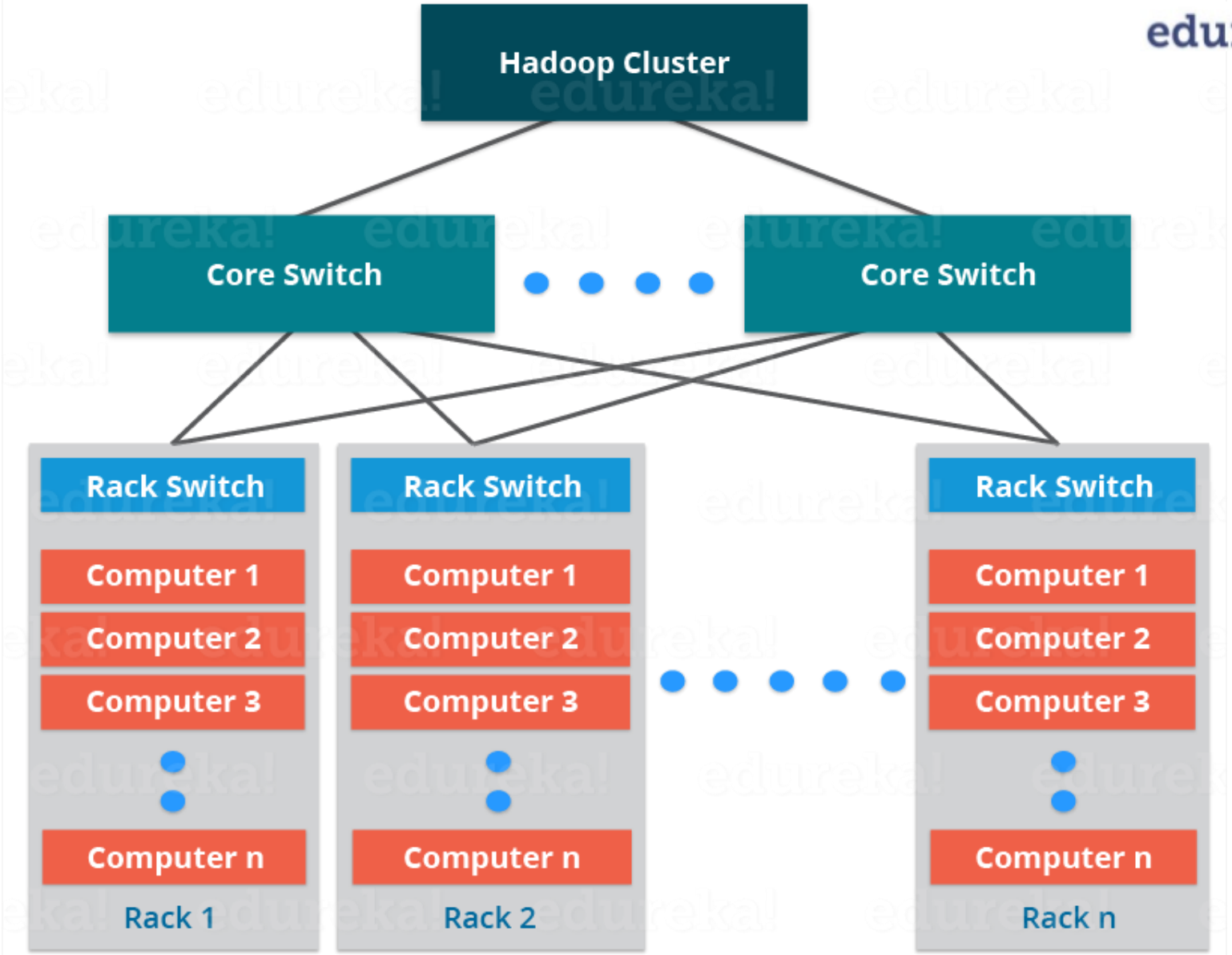


Terminology review: Cluster

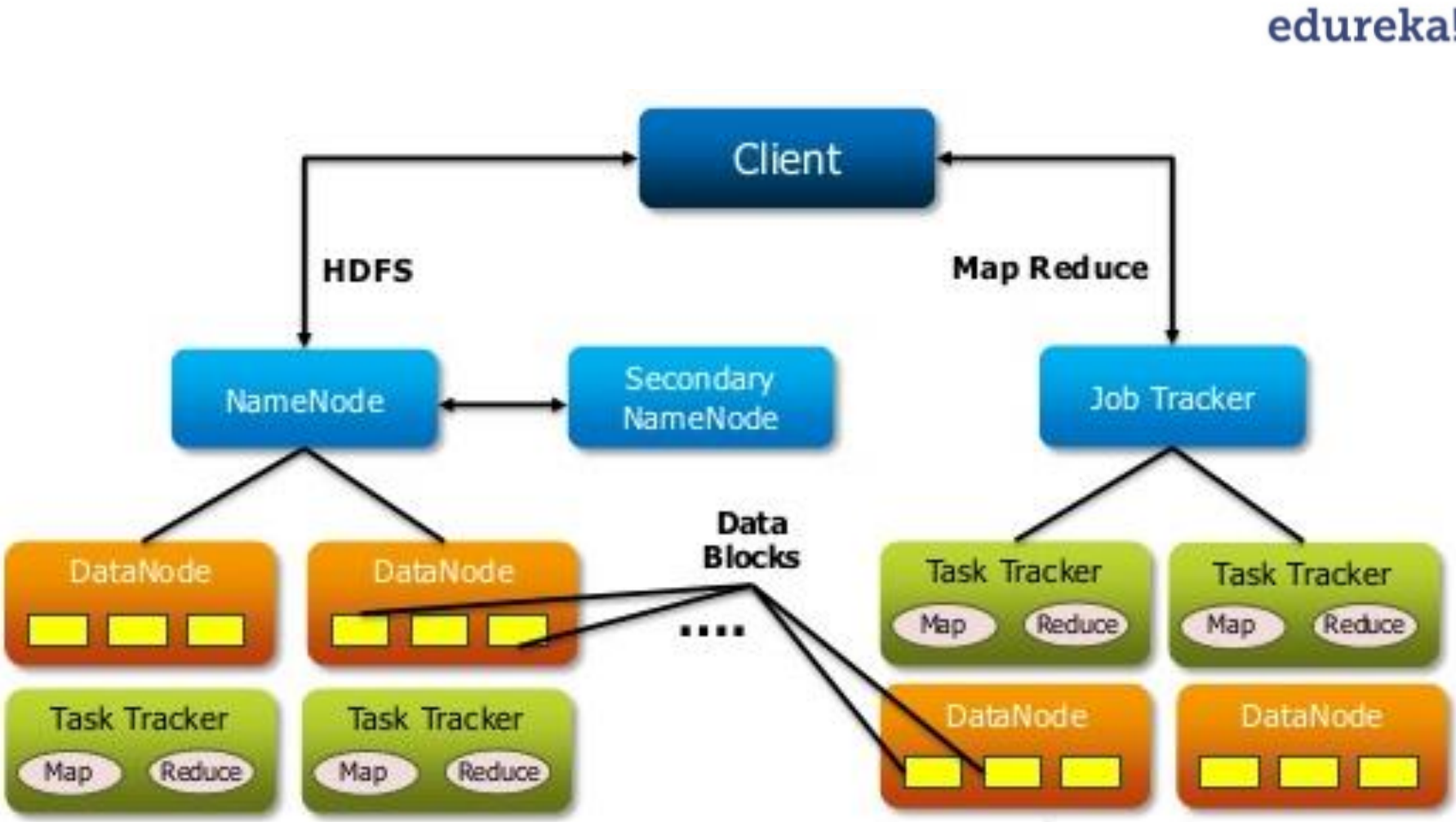
- A **Hadoop cluster** is a collection of racks.



Terminology review: Cluster



Pre-Hadoop 2.2 architecture

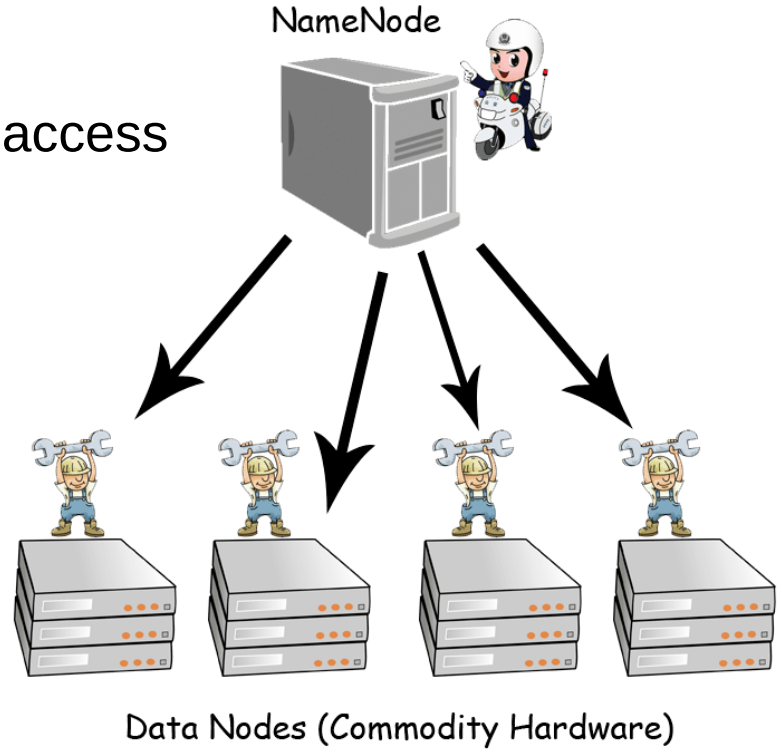


Pre-Hadoop 2.2 architecture

- Distributed Filesystem
  - Mainly **Hadoop Distributed Filesystem** (HDFS)
  - Other filesystems are also supported: IBM Spectrum Scale, Amazon S3, etc.
  - Designed to store large files as multiple blocks on multiple servers
  - Data is automatically re-replicated on need.
- MapReduce Engine
  - Framework for performing calculations on the data stored in the distributed filesystem
  - Built-in resource manager and scheduler

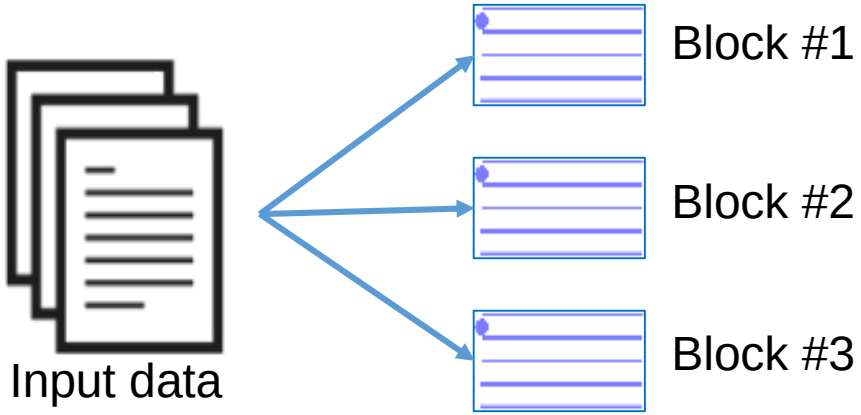
Hadoop distributed filesystem

- Run on top of the existing filesystem.
  - Tolerate high component failure rate through replication of the data
  - Not POSIX compliant.
- Work best with very large files
  - Designed for streaming or sequential access rather than random access



HDFS file blocks

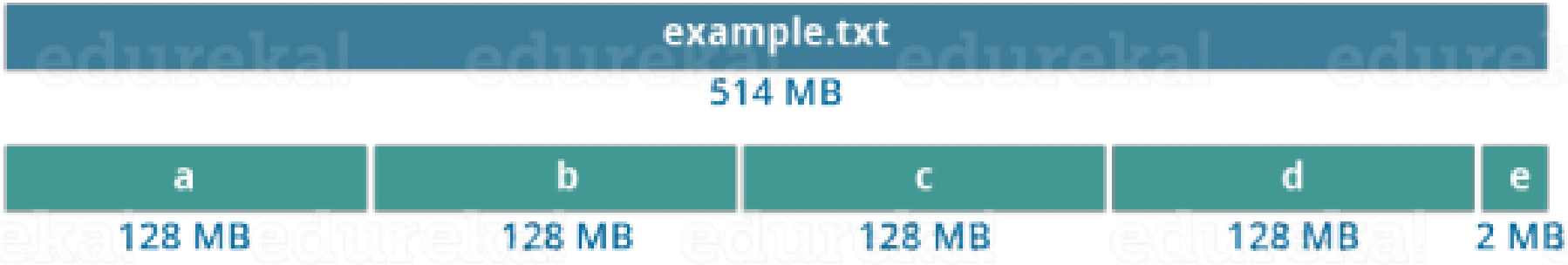
- Hadoop uses blocks to store a file or parts of a file.



- A HDFS block is a file on the underlying filesystem.
  - Block size: commonly 64MBs (Hadoop) or 128MBs (IBM BigInsights)
- Advantages of using blocks
  - Fixed in size → calculate how many can fit on a disk easily
  - Blocks can be spread over multiple nodes → a file can be larger than any single disk in the cluster.

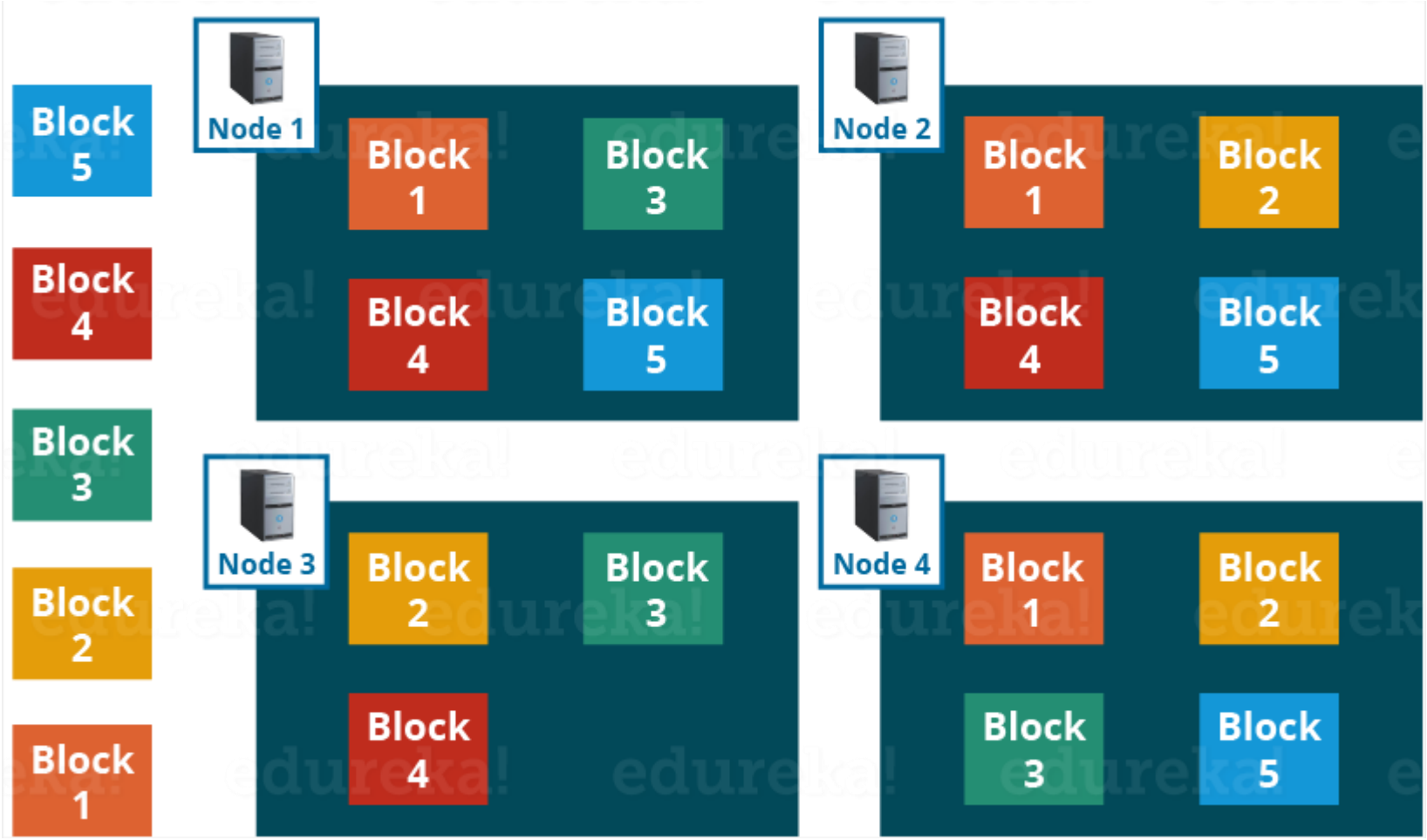
HDFS file blocks

- HDFS blocks also do not waste space.
  - If a file is not an even multiple of the block size, the block containing the remainder does not occupy the space of an entire block.



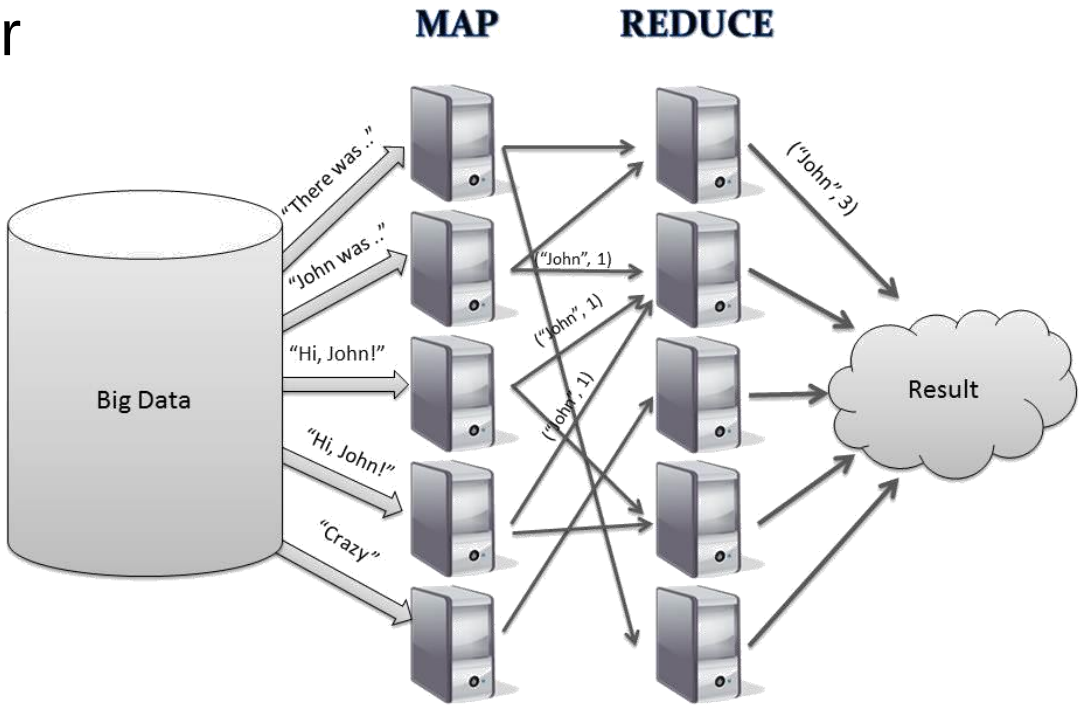
HDFS file blocks

- HDFS blocks fit well with replication, allowing HDFS to be fault tolerant and available on commodity hardware.



Hadoop MapReduce

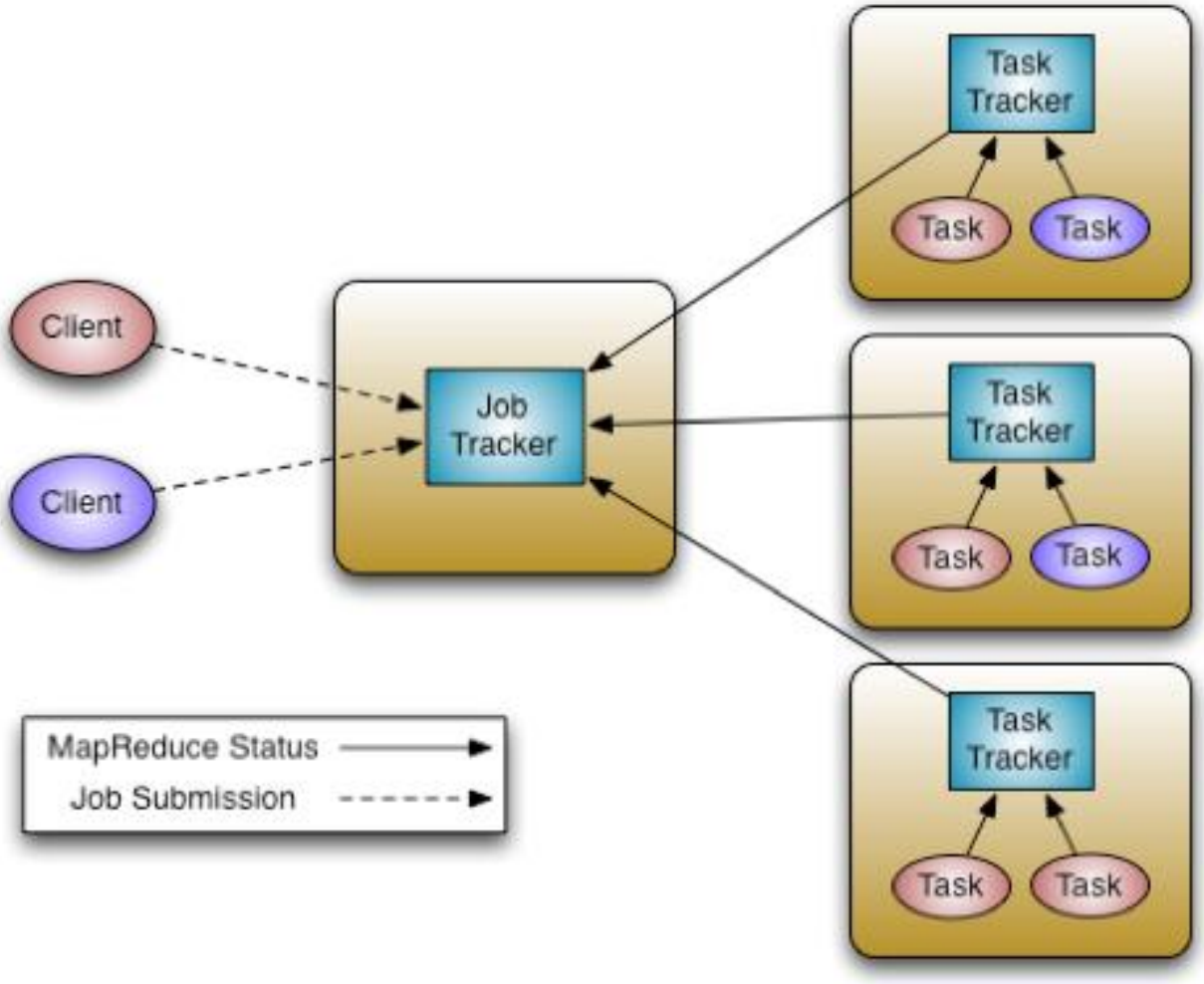
- Programming model for data processing



- Process huge datasets for certain kinds of distributable problems using a large number of nodes.
  - Scalability up to 100s or 1000s of computers, each with several processor cores

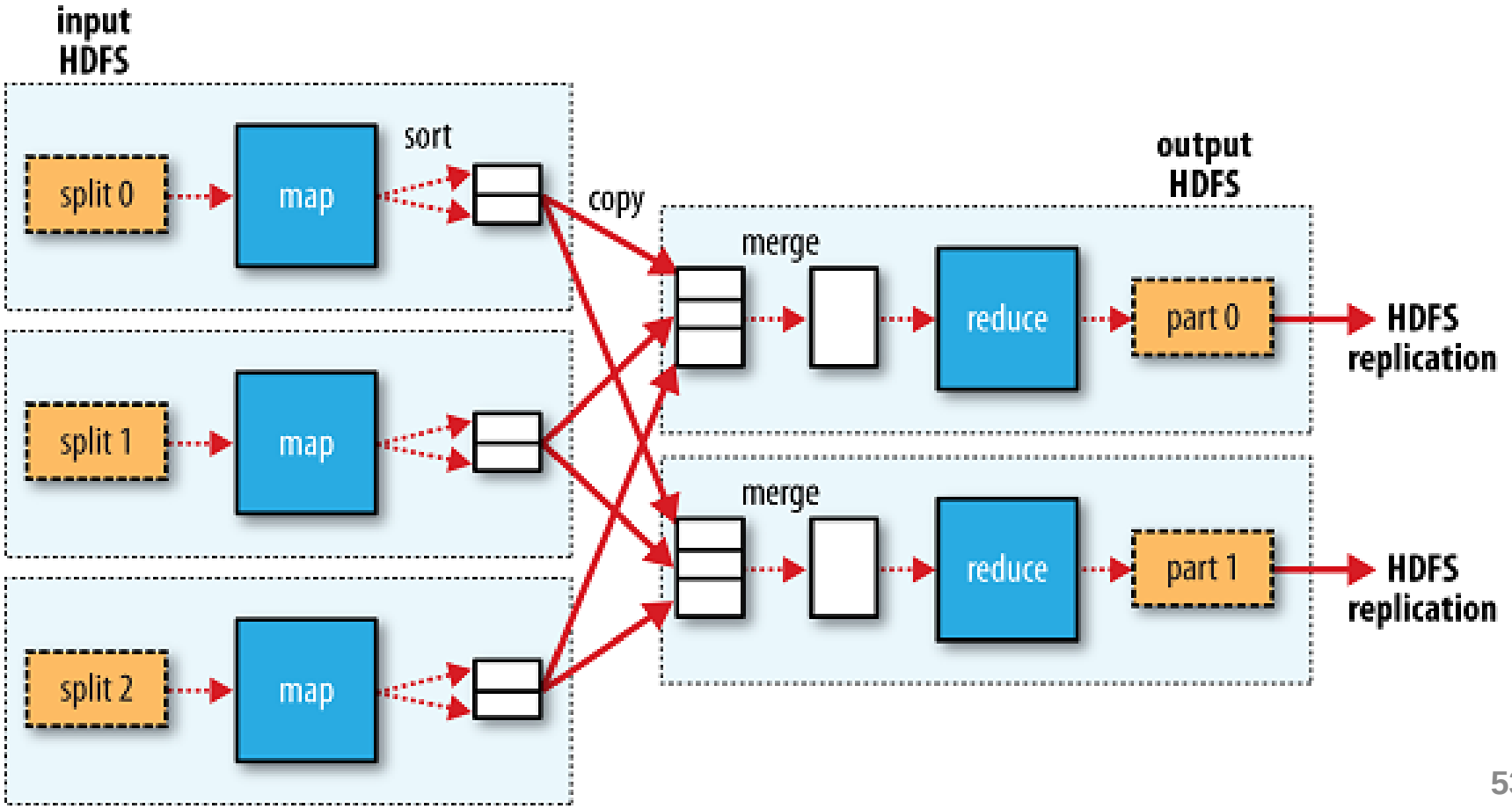
Hadoop MapReduce

- The workload is divided into multiple independent tasks and scheduled across cluster nodes.



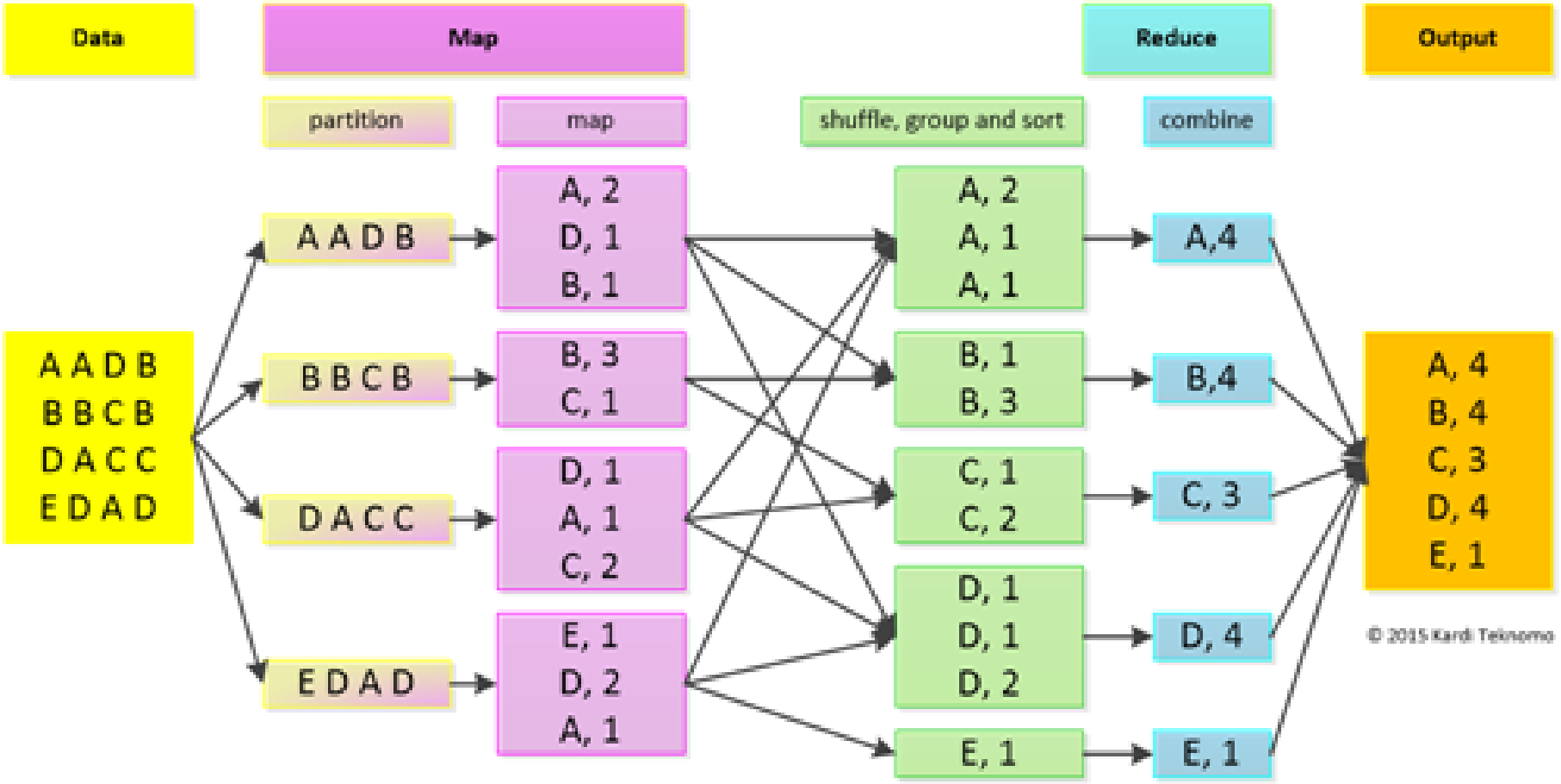
Hadoop MapReduce

- A MapReduce program consists of Map and Reduce tasks.
  - Work performed by each task is done in isolation from one another.

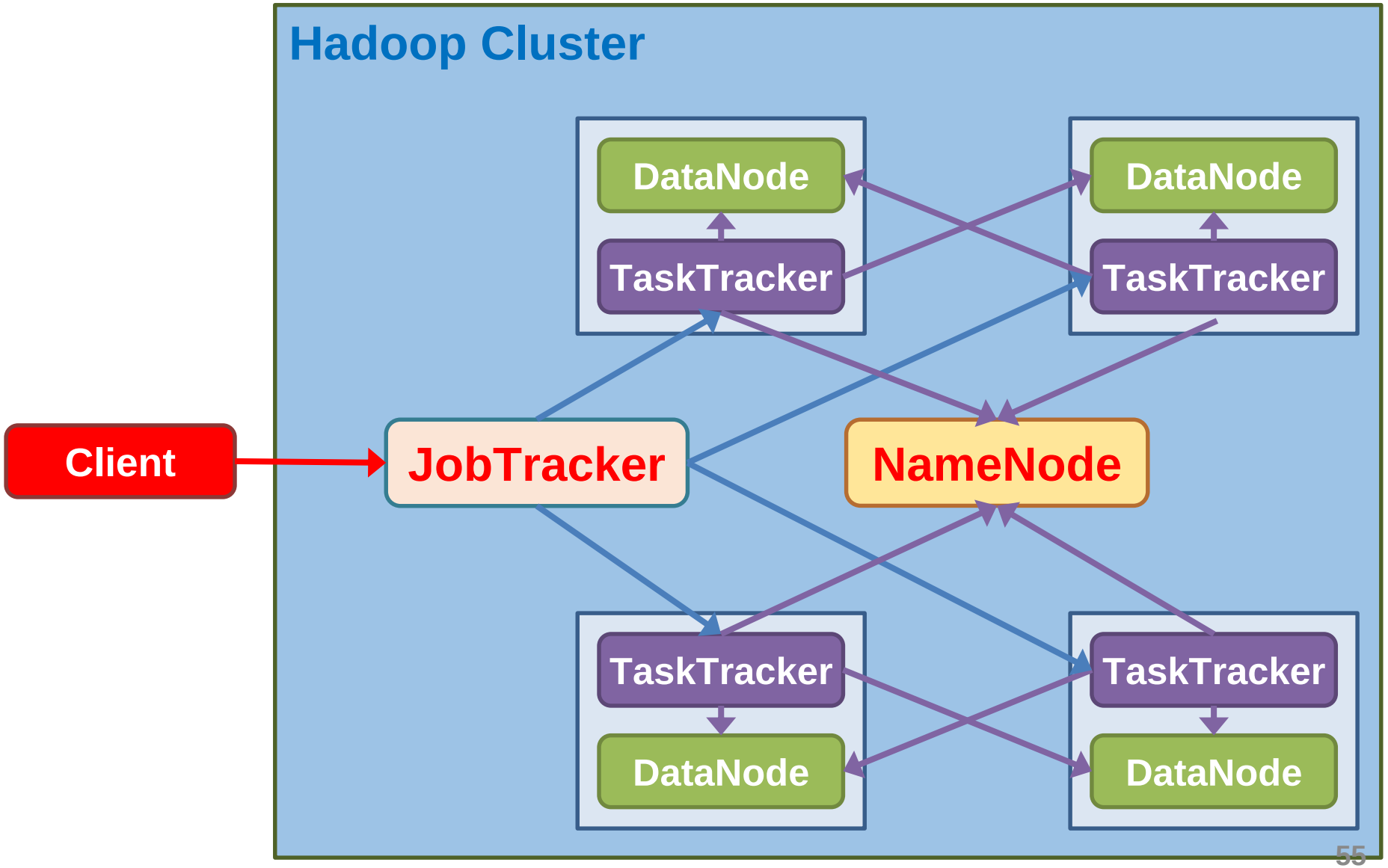


Hadoop MapReduce: An example

- Count the number of occurrences of each letter



Types of nodes



Types of nodes: NameNode

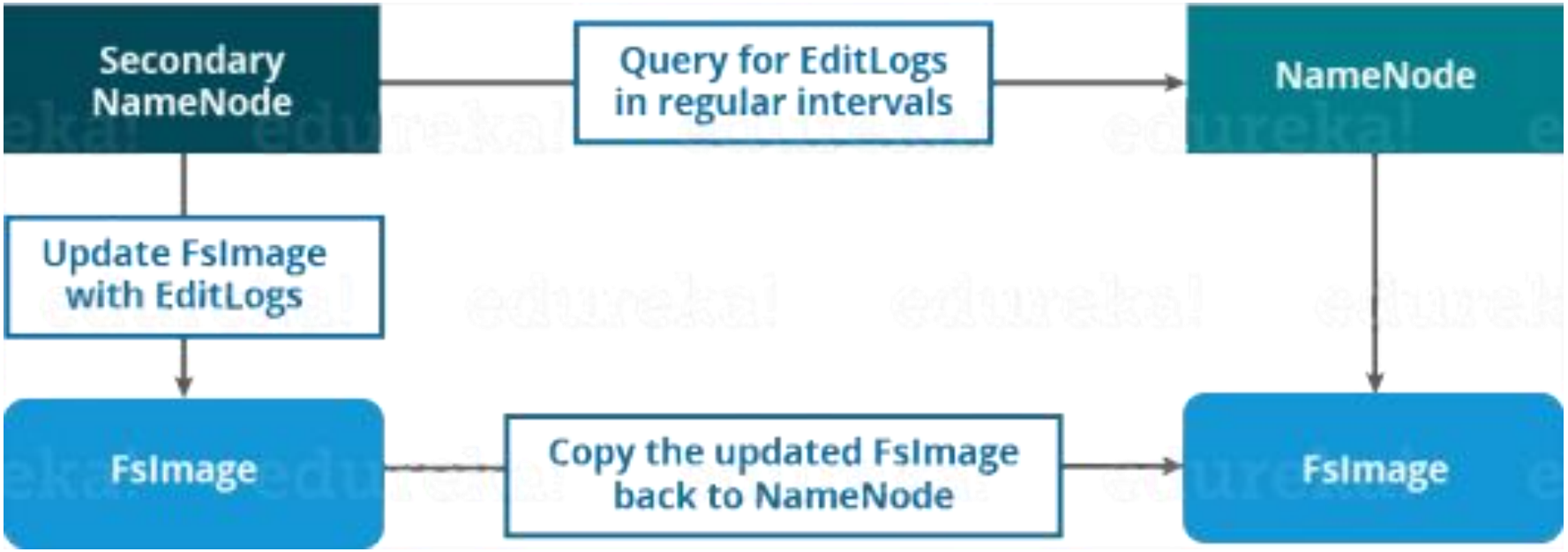
- **Only one per Hadoop cluster**
  - Best enterprise hardware for maximum RAM and reliability
- Manage the filesystem namespace and metadata for a file
  - Data blocks are not stored on the NameNode.
  - Data does not go through the NameNode while reading/writing.
- **Single point of failure**
  - Good idea to mirror the NameNode (e.g. local disk, NFS mount, etc.)
  - Do not use inexpensive or commodity hardware

Types of nodes: DataNode

- **Many per Hadoop cluster**
- Blocks from different files can be stored on the same node.
- Manage data blocks and serves them to clients
- Periodically report to NameNode the list of blocks it stores.
- Suitable for inexpensive/commodity hardware

Types of nodes: Secondary NameNode

- Work concurrently with the primary NameNode as a helper daemon, yet **not a backup NameNode**
- Housekeeping, backup of NameNode metadata
  - Performs regular checkpoints in HDFS.



Types of nodes: JobTracker

- **One per Hadoop cluster**
- Manage MapReduce jobs in the cluster
- Receive job request submitted by client, schedule and monitor MapReduce jobs on the appropriate TaskTrackers
  - Attempt to direct a task to the TaskTracker where the data resides

Types of nodes: TaskTracker

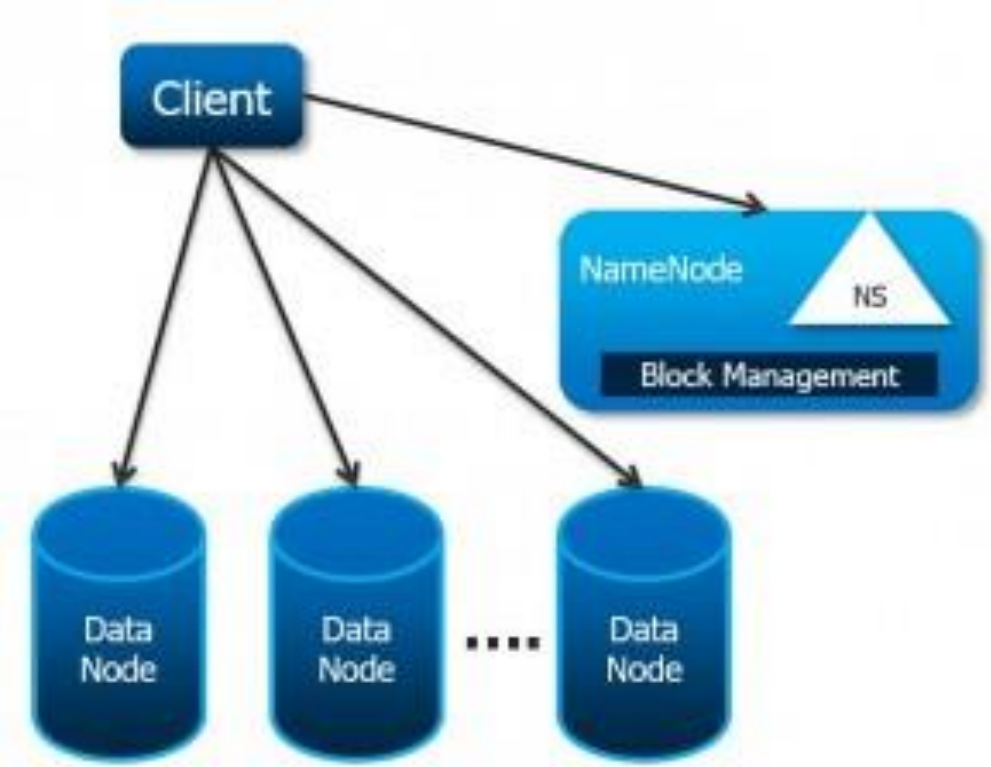
- **Many per Hadoop cluster**
- Execute a Map/Reduce task in Java Virtual Machine (JVM)
  - The number of slots used to run tasks is required to be set.
- Read blocks from DataNode and communicate with the JobTracker via heartbeat messages

Hadoop 2.2 architecture

- **HDFS Federation:** Horizontal scalability of NameNode
- **NameNode High Availability:** NameNode is no longer a Single Point of Failure
- **YARN:** A new framework for cluster resource management
  - Support data processing of size TBs or PBs using Non-MapReduce applications (e.g. MPI, GIRAPH)
  - Splits up the JobTracker into two separate daemons: a global Resource Manager and per-application ApplicationMaster
- Additional features: Capacity Scheduler (Enable Multi-tenancy support in Hadoop), Data Snapshot, Support for Windows, NFS access, etc.

HDFS Federation

- The maximum number of files a Hadoop Cluster can store is limited by the NameNode memory (typically 50–100M files).

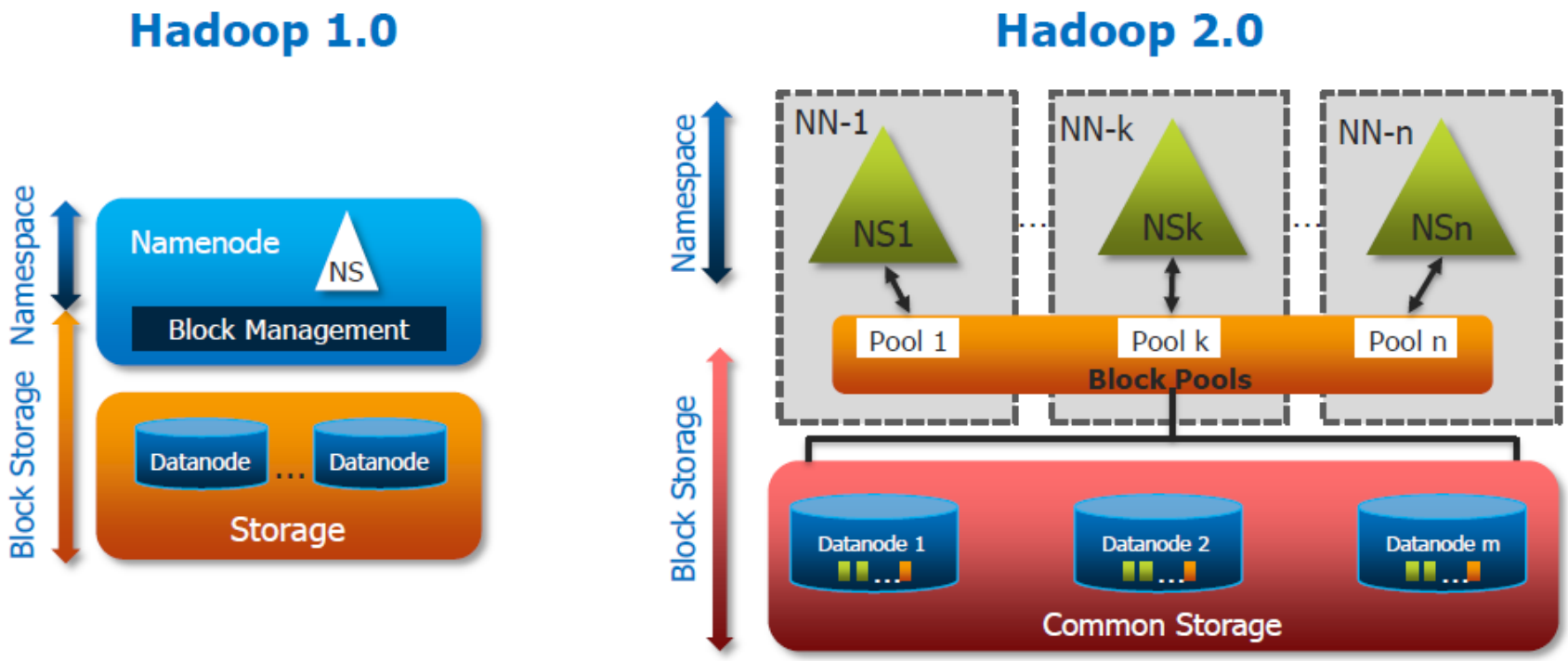


Challenges:

- Meta is stored in NameNode memory
- Bottleneck after ~4000 nodes
- Results in cascading failures

HDFS Federation

- Multiple NameNodes, act independently, share DataNodes
- Each NameNode has its own namespace and control over its own set of files.



HDFS High Availability

- In the Pre-Hadoop 2.2 architecture, NameNode failure makes the Hadoop Cluster inaccessible.
- Hadoop Administrators need to manually recover the NameNode using Secondary NameNode.

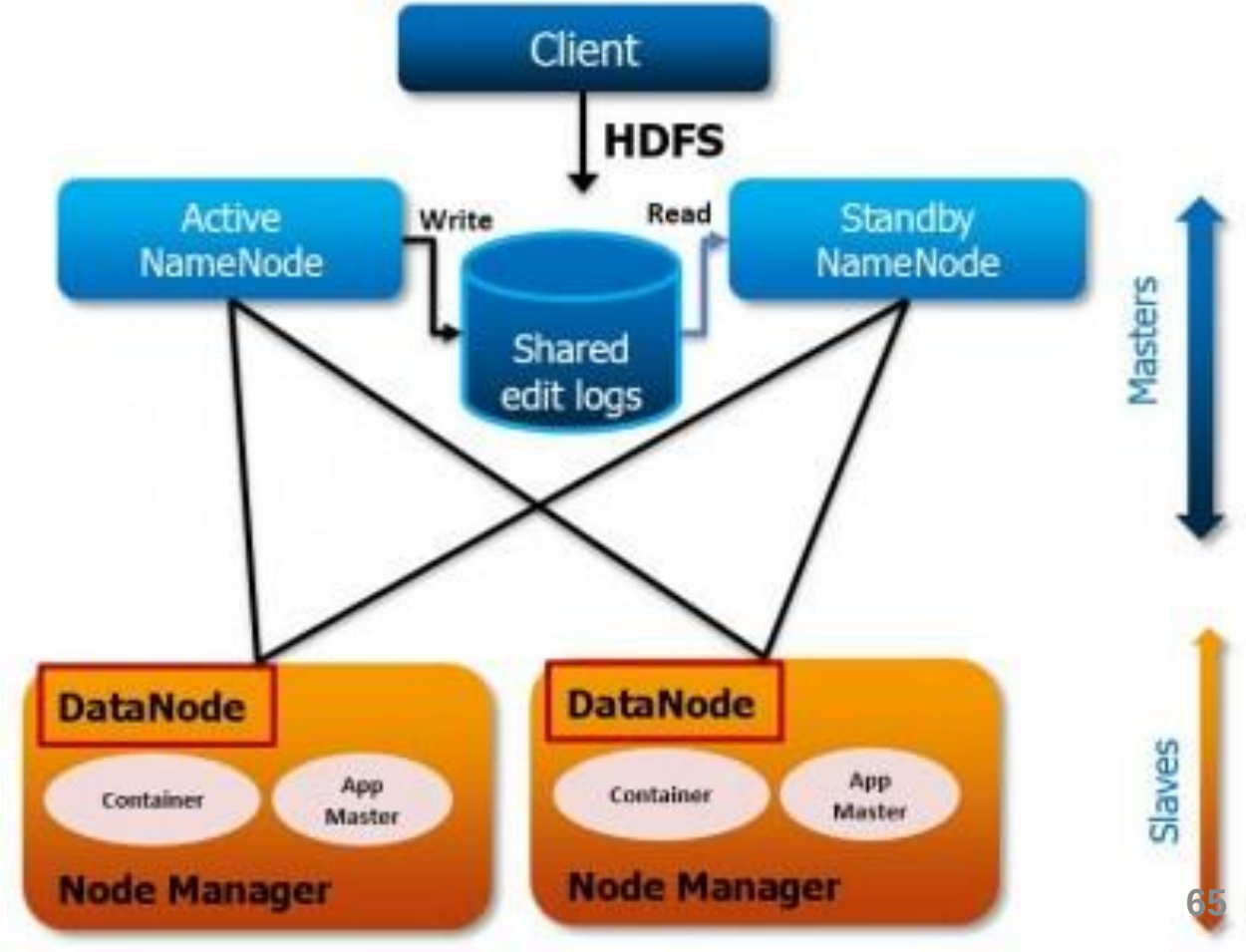
Secondary NameNode:

- "Not a hot standby" for the NameNode
- Connects to NameNode regularly
- Housekeeping, backup of NameNode metadata
- Saved metadata can build a failed NameNode

HDFS High Availability

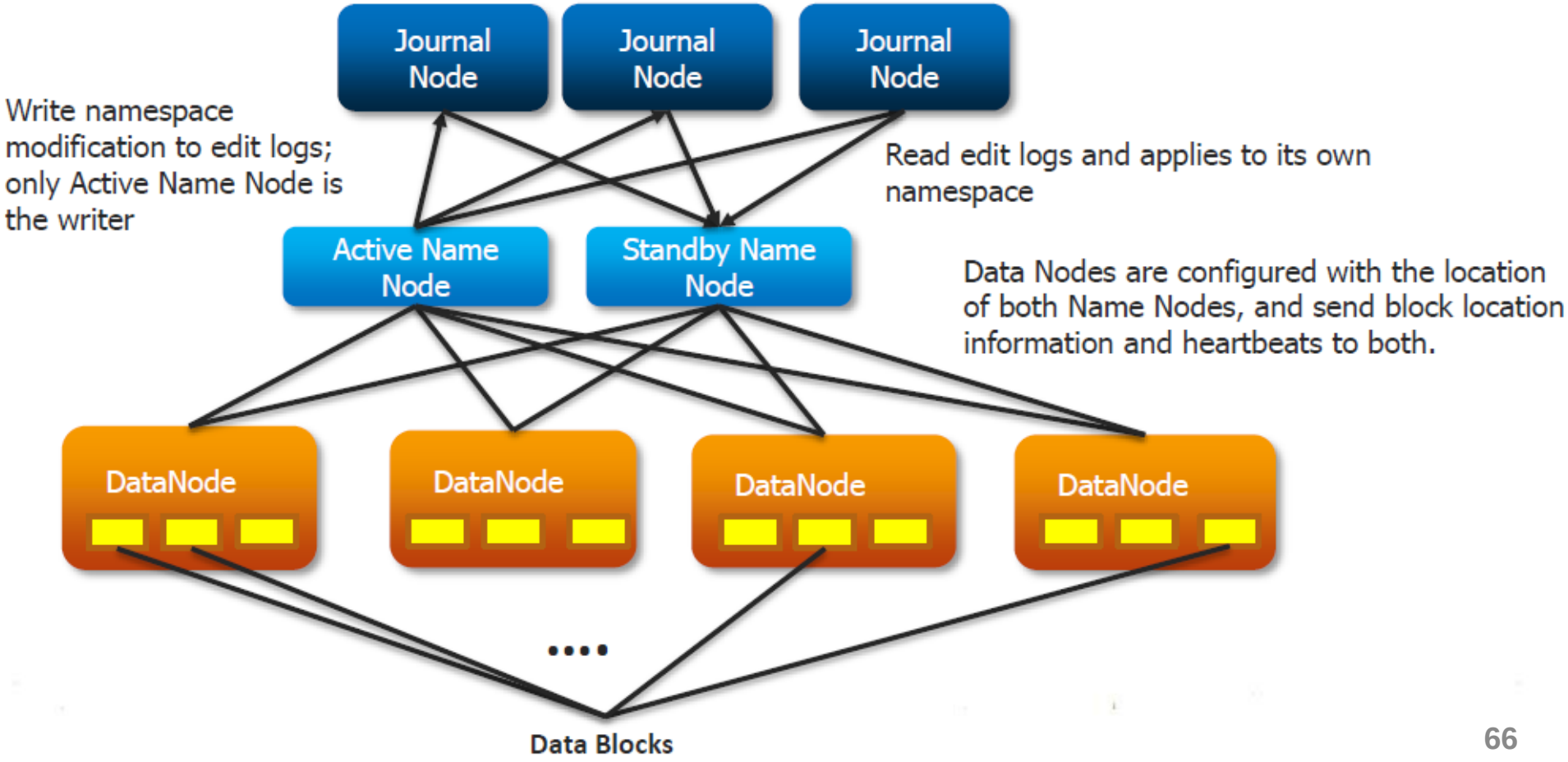
- The **Standby NameNode** will automatically take over as the Active NameNode when a failover occurs.

It keeps synchronized with the Active NameNode through the Shared edit logs.



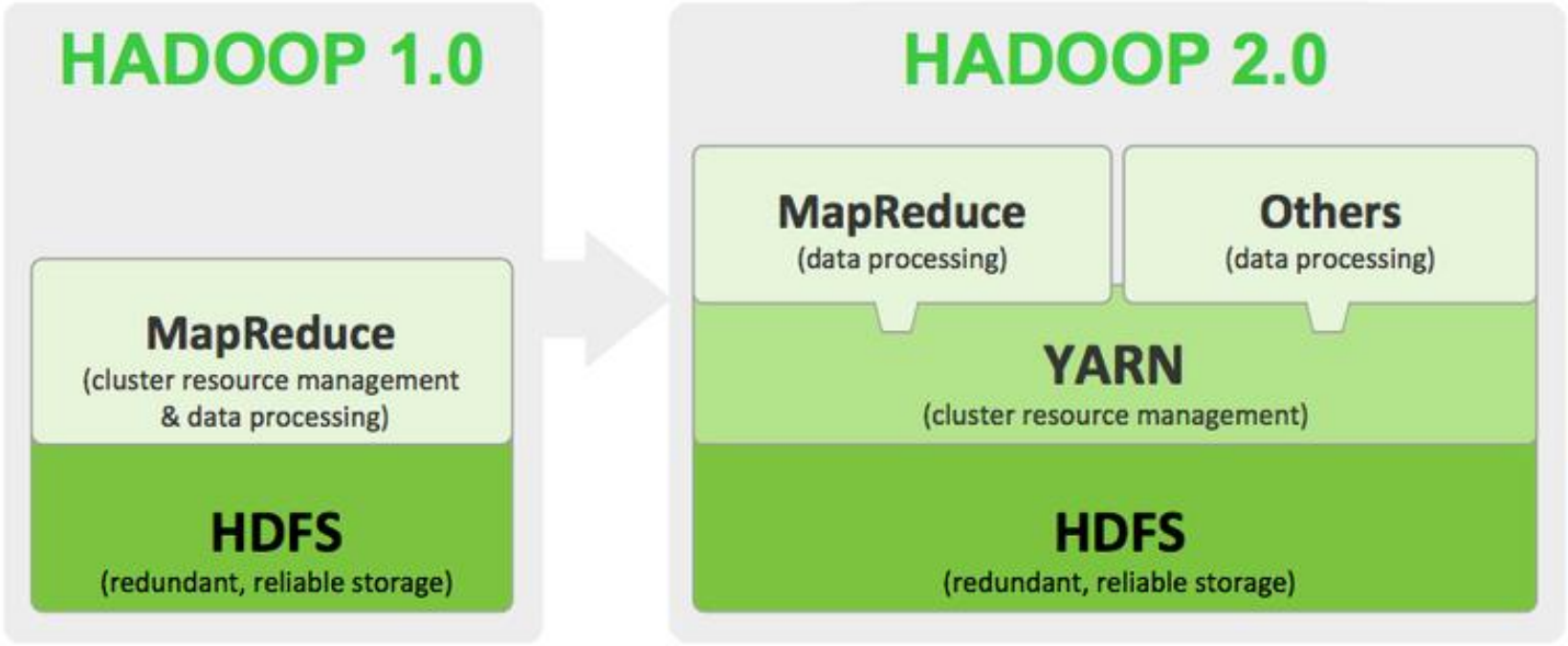
HDFS High Availability

- Shared edit logs are maintained by techniques such as NFS filer and Quorum journal manager (QJM).



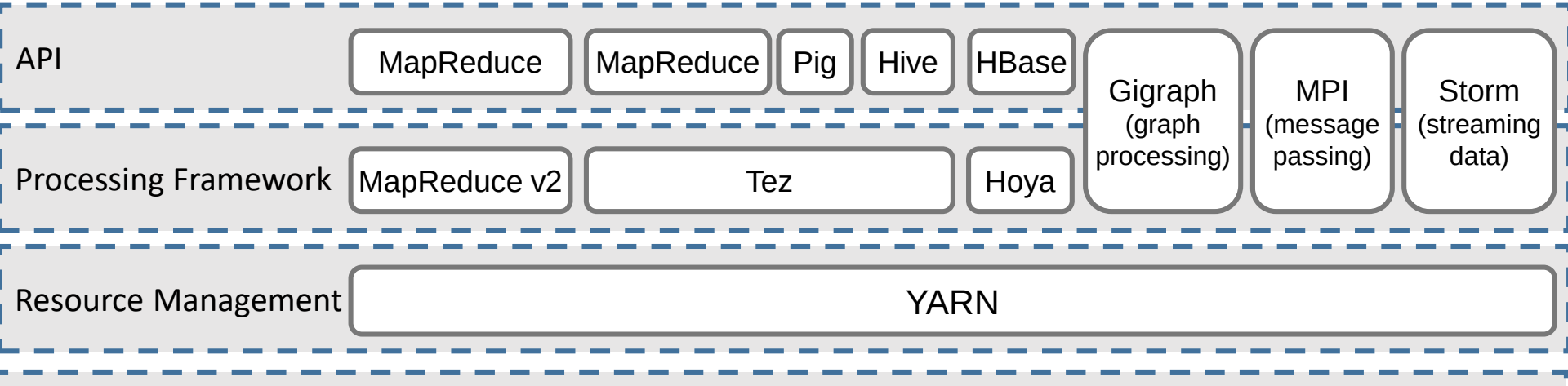
YARN

- **YARN** (Yet Another Resource Negotiator) – Cluster resource manager and scheduler external to any framework
- Not a requirement to run YARN with Hadoop 2.2.

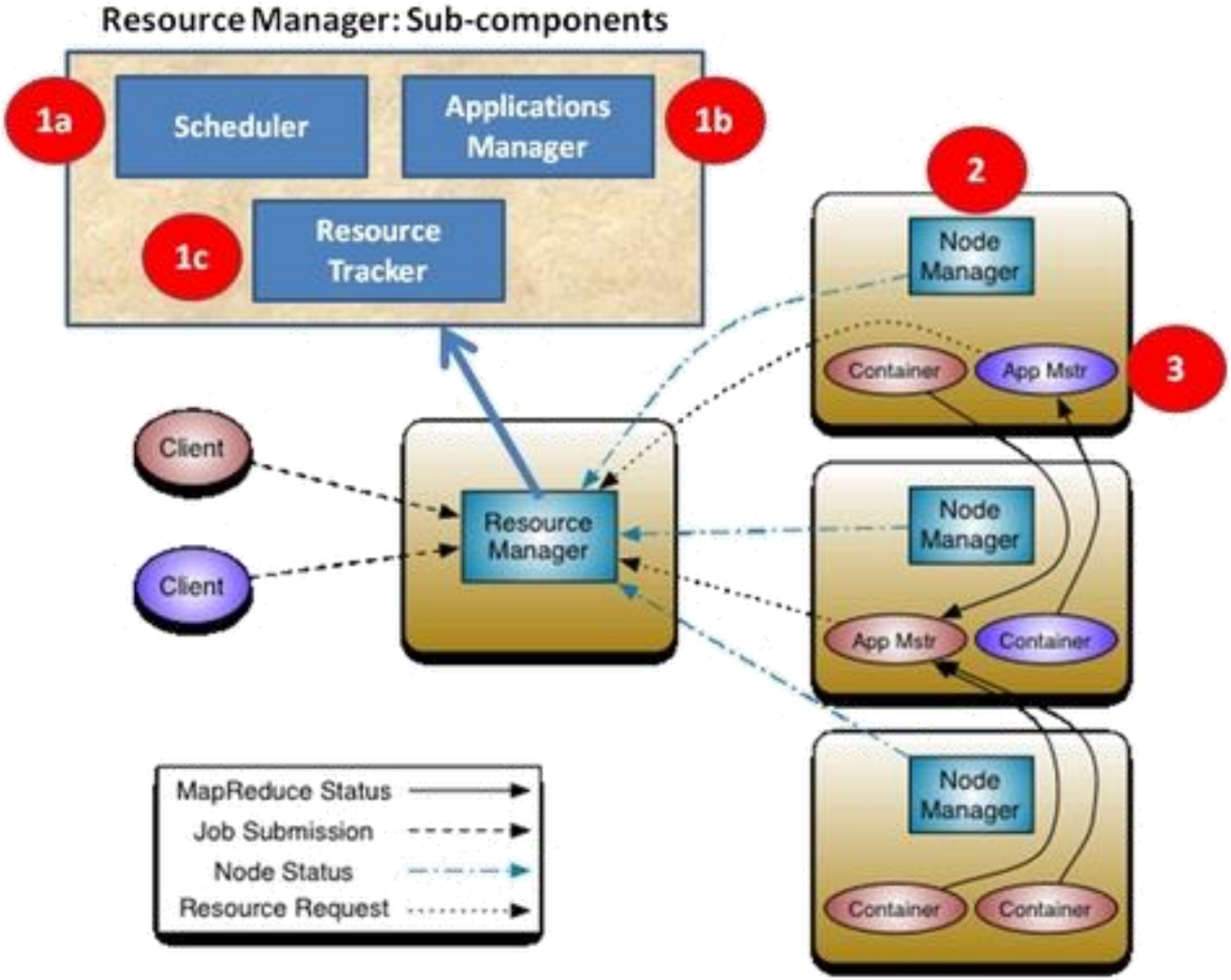


YARN

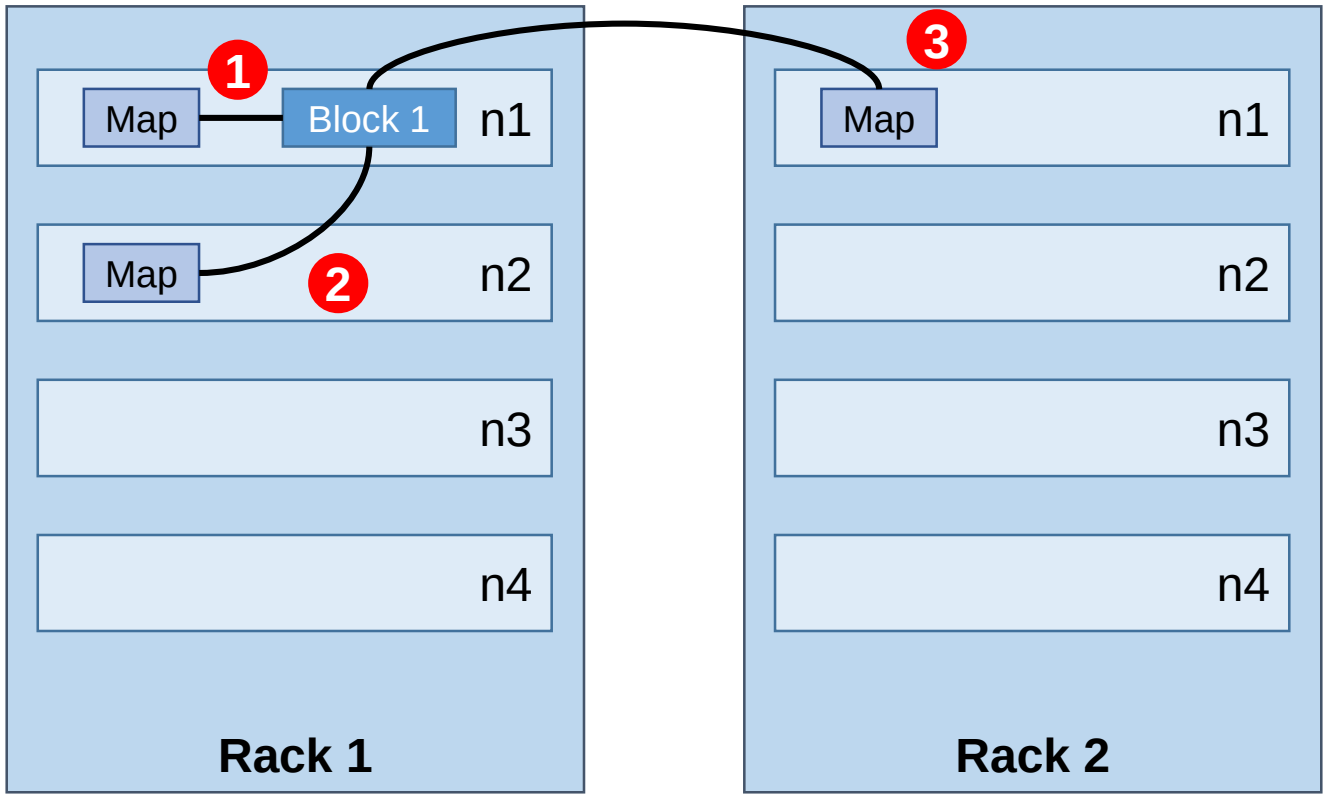
- Generic scheduling and resource management
  - With YARN, Hadoop can support more than just MapReduce, and hence more than just batch processing.
- More efficient scheduling and workload management.
  - No more balancing between Map slots and Reduce slots



An overview of YARN



Topology awareness



The administrator defines the topology in the *topology.script.file.name* property in *core-site.xml*

1 – 3 Best to worst options

Hadoop 3.0 architecture

- Apache Hadoop 3.0 incorporates a number of significant enhancements over the previous major release line (hadoop-2.x).
- 03 September, 2016: Release 3.0.0-alpha1 available
- 25 January, 2017: Release 3.0.0-alpha2 available

Hadoop 3.0 architecture

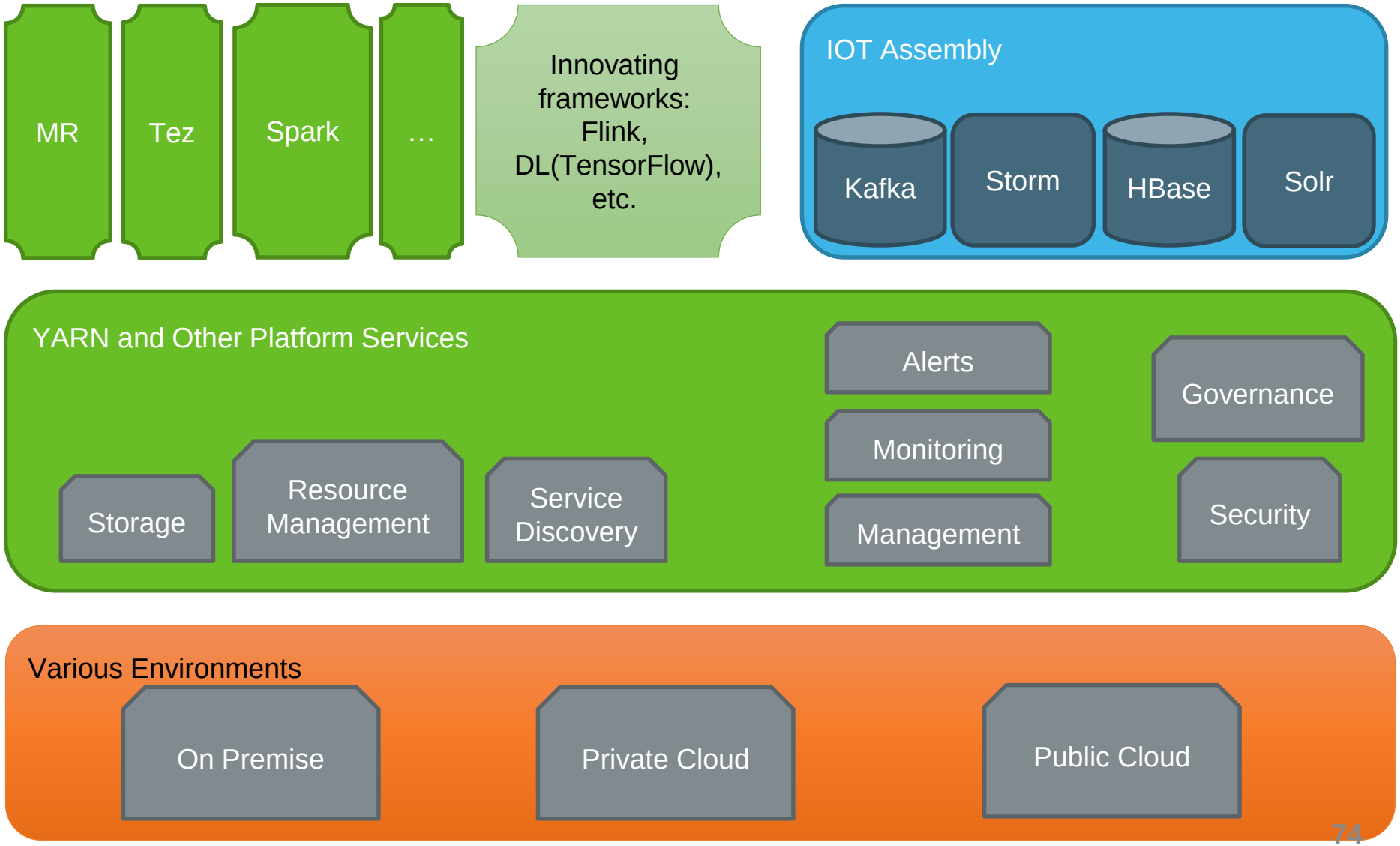
- Minimum required Java version increased to Java 8.
- Support for erasure encoding in HDFS
  - Store data with significant space savings compared to replication
- YARN Timeline Service v.2
  - Improve scalability and reliability of Timeline Service, and enhance usability by introducing flows and aggregation
- Shell script rewrite
  - Fix many long-standing bugs and include some new features

Hadoop 3.0 architecture

- **Support for multiple Standby NameNodes**
  - This improves the fault tolerance of HDFS.
- New default ports for several services
  - The default ports for NameNode, Secondary NameNode, DataNode, and KMS have been moved out of the Linux ephemeral range (32768-61000).
- Intra-DataNode Balancer
  - Address the intra-node skew that can occur when disks are added or replaced.

Source: https://blog.cloudera.com/blog/2016/09/getting-to-know-the-apache-hadoop-3-alpha/

Hadoop 3.0 Architecture



Batch processing vs. Stream processing

Batch-only framework

Hybrid framework

Stream-only framework

- **Batch processing** involves operating over a large, static dataset and returning the result at a later time when the computation is complete.
- **Stream processing** involves computing over data as it enters the system.

THE END