The Center of Applied Data Science

Big Data and Hadoop Ecosystems

Ravie Chandren Muniyandi

m.raviechandren@gmail.com

+6012-3249577

June 2019

BIG DATA & HADOOP

The Center of Applied Data Science

Learning Objectives

- What is big data ?
- How Hadoop handle big data ?
- What is the Hadoop framework ?
- Hadoop Ecosystems
- Spark vs Hadoop
- Batch vs Streaming

Contents

1. 5Vs of Big Data
2. Types of Data
3. Introduction to Apache Hadoop
4. Principles of Hadoop
5. Hadoop Ecosystem
6. Use Cases of Apache Spark
7. Big Data Pipelines

Training Schedule

9.00-10.30 a.m. –Introduction & Training

10.30 a.m. – 10.45 a.m. – Morning tea break

10.45 a.m.-12.30pm – Training

12.30 p.m. – 1.30 p.m. – Lunch break/prayers

1.30-3.30 pm – Training

3.30 p.m. – 3.45 p.m. – Afternoon tea break

3.45-5.00 pm – Training

5.00 p.m. – Training Adjourns

The Center of Applied Data Science

What is big data ?

big data

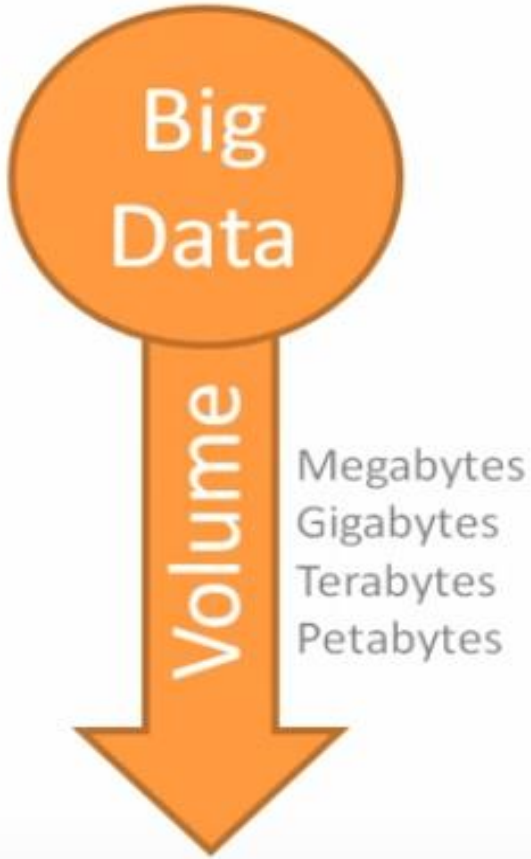
noun COMPUTING

extremely large data sets that may be analysed computationally to reveal patterns, trends, and associations, especially relating to human behaviour and interactions.
"much IT investment is going towards managing and maintaining big data"

Translations, word origin, and more definitions

The Center of Applied Data Science

How Big is Big Data?



The Center of

What is Big Data?

Big Data is high-volume, high-velocity and/or high-variety information assets that demand cost-effective, innovative forms of information processing that enable enhanced insight, decision making, and process automation.

What Is Big Data? - Gartner IT Glossary - Big Data

www.gartner.com/it-glossary/big-data

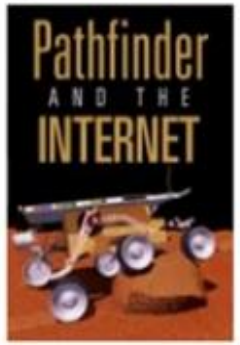
How Big data evolve ?

1994

- 11 million American households are "equipped to ride the information superhighway."
- One of the first known Web purchases takes place: a pepperoni pizza with mushrooms and extra cheese from Pizza Hut.
- President Bill Clinton's White House comes online.
- Yahoo! is created by Stanford University graduate students Jerry Yang and David Filo. They originally named the site "Jerry and David's Guide to the World Wide Web."

1997

- Millions "visit Mars – on the internet" – the Jet Propulsion Lab allows people to watch the Sojourner rover landing and exploration of Mars. The broadcast generates about 40 million to 45 million hits each day.
- Netflix launches as a company that sends DVDs to homes via mail.
- Go Daddy launches as Jomax Technologies.
- Google.com registers as a domain.
- Jorn Barger becomes the first person to use the term "Weblog" to



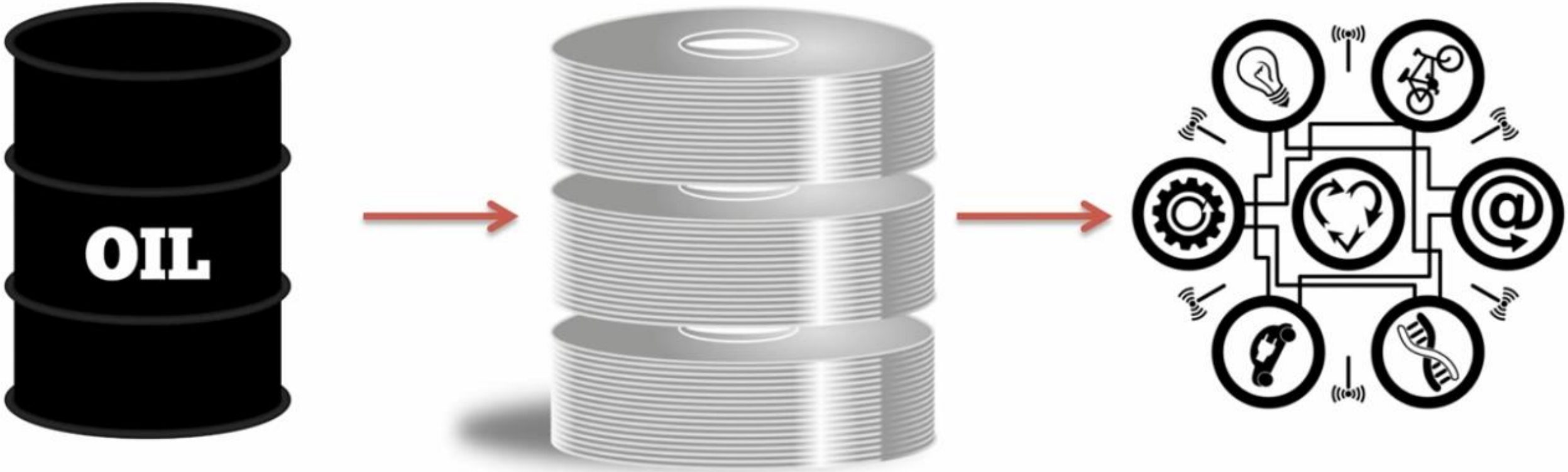
2004

- 11% of American internet users follow the returns on election night online. One-in-ten internet users sign up for political email newsletters and news alerts during the campaign.
- Harvard student Mark Zuckerberg launches thefacebook.com. 1,200 Harvard students sign up within the first 24 hours. Facebook goes on to become the world's biggest social networking site, with over a billion users worldwide.
- Google starts trading on the NASDAQ at $85 a share.

- Social news website Digg launches. Digg users vote to "digg up" links that they like and "bury" down those they don't.
- Mozilla releases Firefox 1.0.
- Massively multiplayer online role-playing game(MMORPG) World of Warcraft launches.

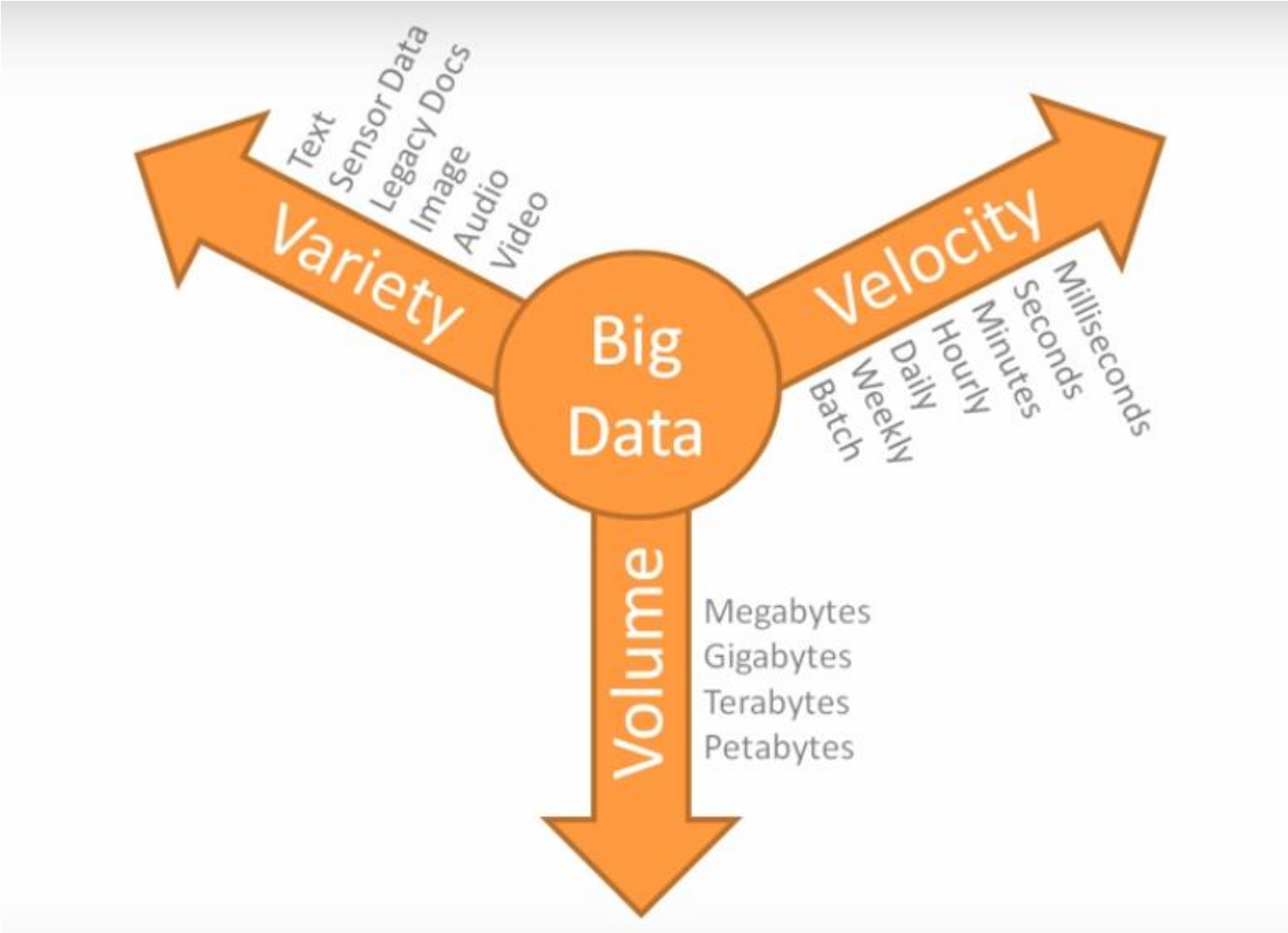
Source - http://www.pewinternet.org/

The Center of Applied



The Center of Applied Data Science

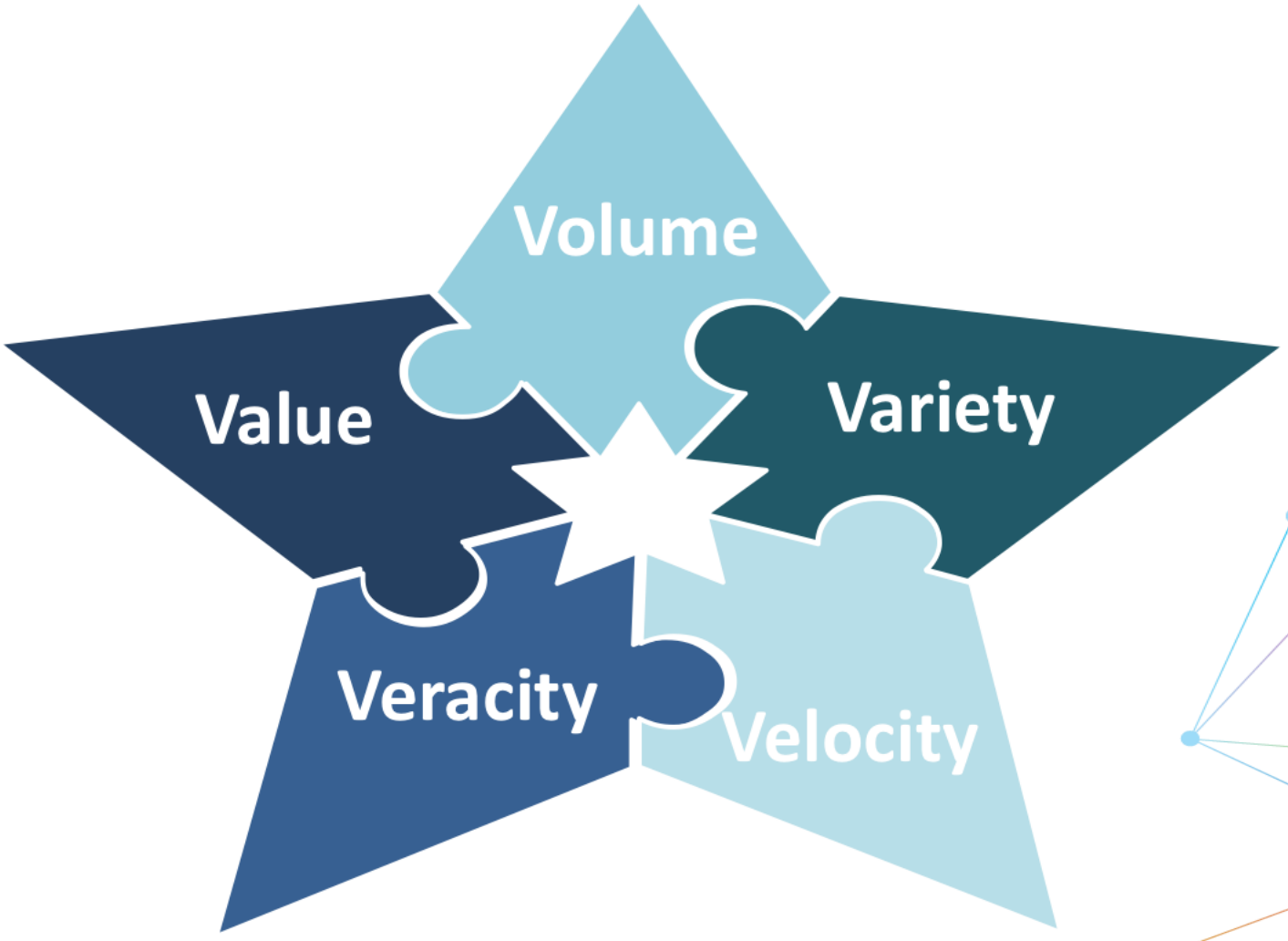
3Vs



The Center of Applied Data Science

5Vs of Big Data

Definition



The Center of Applied Data Science

5Vs of Big Data

Definition

Volume

Vast amounts of data (Zettabytes/ Brontobytes) generated every second.

Variety

Different types of data we can now use (structured, semi-structured and unstructured data).

Velocity

The speed of new data is being generated and being moved around.

5Vs of Big Data

Definition

Veracity

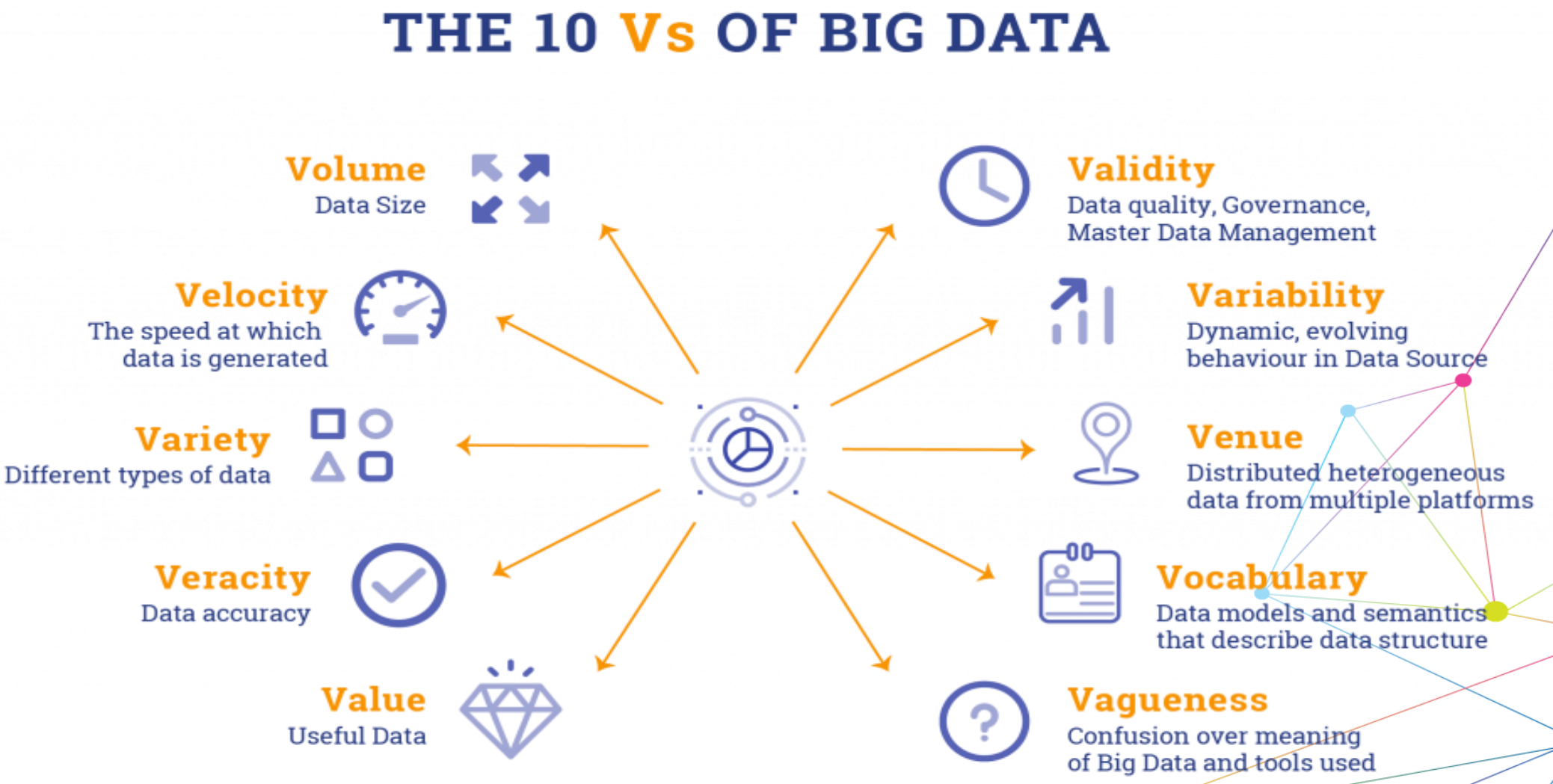
Able to use data regardless of the quality and accuracy.

Value

Value generated from big data.

The Center of Applied Data Science

10Vs



Source: xenonstack.com

The Center of Applied Data Science

Types of Data

Types of Data

Definition

Structured

- Stored in **databases**.
- Organized in rows and columns.
- **Example:** Data received from web logs and sensors.

Semi-Structured

- Data that is not stored in traditional database, but being stored in **certain organizational way**.
- **Example:** NoSQL documents

Unstructured

- Data that **do not have clear format** in storage.
- **Example:** Pictures uploaded online, YouTube videos, Text messages sent to social media

Batch vs Real time Processing

- Batch :
  - Batch data processing is an efficient way of processing high volumes of data is where a group of transactions is collected over a period of time. Data is collected, entered, processed and then the batch results are produced (Hadoop is focused on batch data processing).
  - Batch processing requires separate programs for input, process and output. An example is payroll and billing activities, which usually occur on monthly cycles.
  - E.g. When you're processing archived or historical data to determine consumer trends over the past few years.

Real Time:

- Real time data processing involves a continual input, process and output of data. Data must be processed in a small time period (or near real time).

E.g.

Radar systems

Customer services

When a shopper hits the ATM, the bank balance and transactional data have to be processed instantly, or so close to instantly that the customer doesn't even notice the delay.

Batch Processing Advantages & Disadvantages

Advantage:

- Ideal for processing large volumes of data/transaction for it increases efficiency rather than processing each individually.
- Can be done or processed independently during less-busy times or at a desired designated time.
- Ideal for processing large volumes of data/transaction for it increases efficiency rather than processing each individually.

Disadvantage:

- Disadvantage of batch processing is the time delay between the collection of data (transaction receiving) and getting the result (output in master file) after the batch process.

Distributed Computing

- **Distributed Computing** refers to the use of **distributed systems** to solve computational problems.

- A **Distributed System** consists of multiple **autonomous** computers that communicate through a computer network.

- A problem will be divided into small tasks, each of which is solved by one or more computers, which communicate with each other by passing messages.

- Generally, in case of individual computer failures there are **toleration mechanisms** in place. However, the **cardinality**, **topology** and the **overall structure** of the system is **not known** beforehand and everything is **dynamic**.

- There are two predominant ways of organizing computers in a distributed system. The first is the **client-server architecture**, and the second is the **peer-to-peer architecture**.

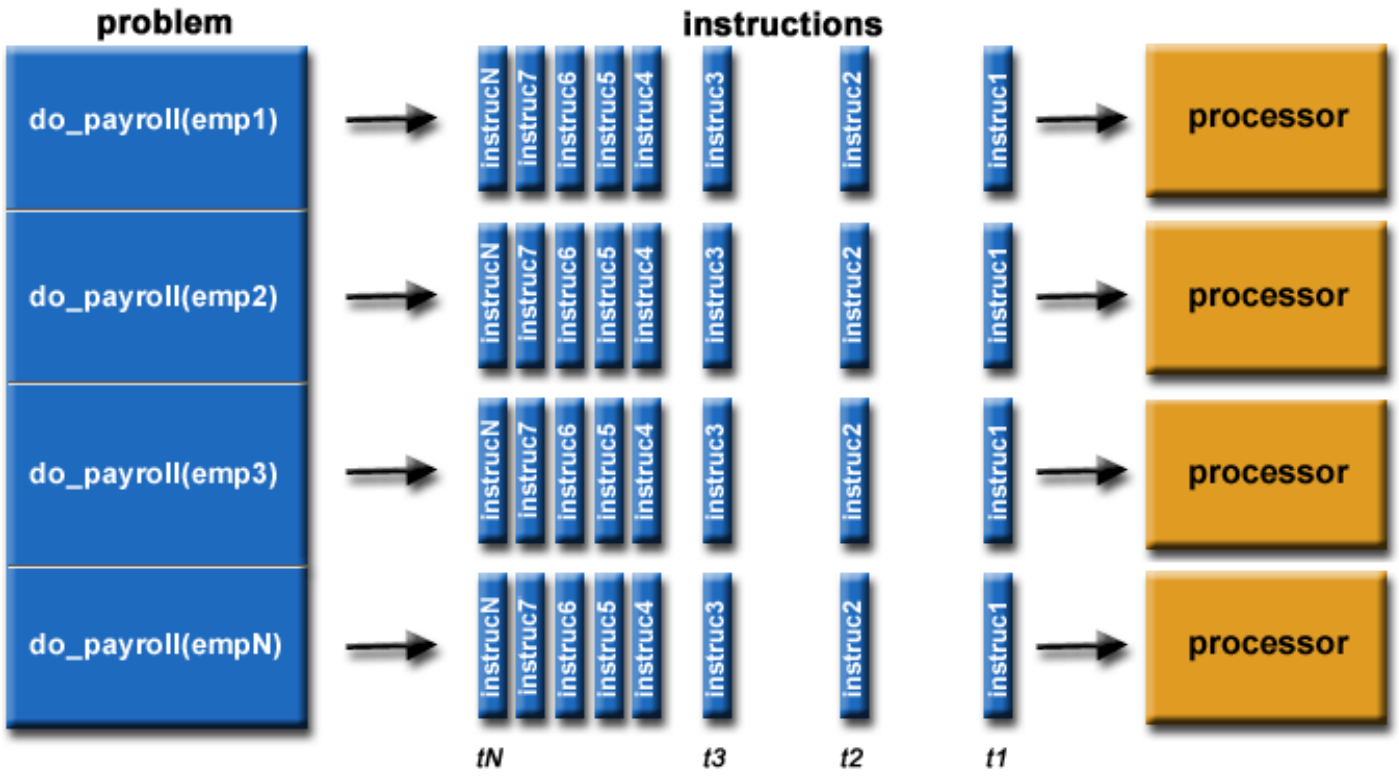
Parallel Computing

- **Parallel Computing** is a form of computation in which many calculation are carried out simultaneously. Large problems can often be divided into smaller **independent** parts, which can then be solved at the same time.

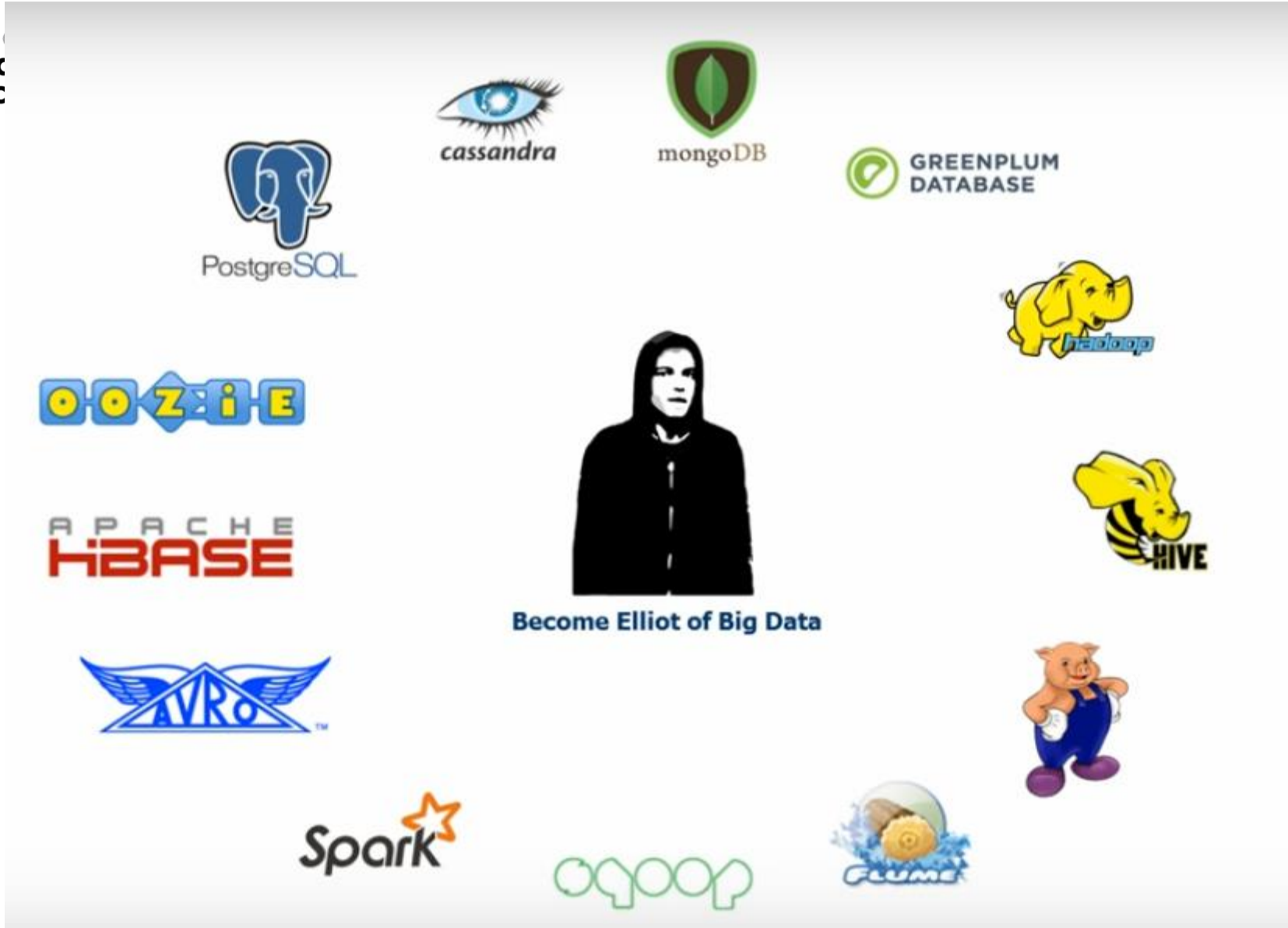
- There are two forms of parallel computing:
  - **Parallelism in Hardware (Uniprocessor):**
    - Parallelism in Uniprocessors
      - Pipelining
      - Superscalar, VLIW, etc.
    - SIMD instructions, vector processing, GPUs
    - Multiprocessor
      - Symmetric shared-memory multiprocessors
      - Distributed memory multiprocessors
      - Chip-multiprocessors a.k.a multi-cores.
  - **Parallelism in software:**
    - instruction-level
    - Data
    - task parallelism.

Parallel Computing

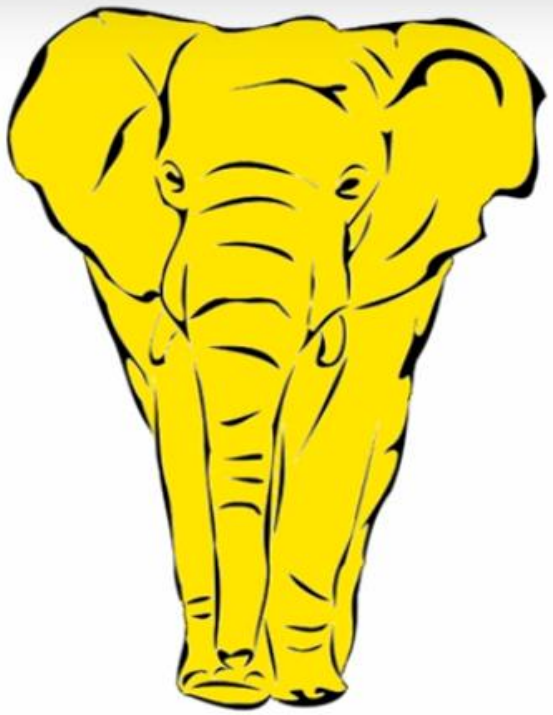
- In the simplest sense, parallel computing is the simultaneous use of multiple compute resources to solve a computational problem:
  - A problem is broken into **discrete** parts that can be solved concurrently
  - Each part is further broken down to a series of instructions
  - Instructions from each part execute simultaneously on different processors
  - An overall control/coordination mechanism is employed



The Center
Applied
Data Sc



The Center of Applied Data Science



Apache Hadoop

Where did it come?

How to solve big data problem?

Part 1 -

Secure | https://research.google.com/archive/gfs.html

Google Research Publications

Google Research Home

The Google File System

Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung

Abstract

We have designed and implemented the Google File System, a scalable distributed file system for large distributed data-intensive applications. It provides fault tolerance while running on inexpensive commodity hardware, and it delivers high aggregate performance to a large number of clients.

While sharing many of the same goals as previous distributed file systems, our design has been driven by observations of our application workloads and technological environment, both current and anticipated, that reflect a marked departure from some earlier file system assumptions. This has led us to reexamine traditional choices and explore radically different design points.

The file system has successfully met our storage needs. It is widely deployed within Google as the storage platform for the generation and processing of data used by our service as well as research and development efforts that require large data sets. The largest cluster to date provides hundreds of terabytes of storage across thousands of disks on over a thousand machines, and it is concurrently accessed by hundreds of clients.

In this paper, we present file system interface extensions designed to support distributed applications, discuss many aspects of our design, and report measurements from both micro-benchmarks and real world use.

Appeared in:
19th ACM Symposium on Operating Systems Principles,
Lake George, NY, October, 2003.

How to solve big data problem?

Part 2 -

Secure | https://research.google.com/archive/mapreduce.html

Google Research Publications

Google Research Home

MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat

Abstract

MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key. Many real world tasks are expressible in this model, as shown in the paper.

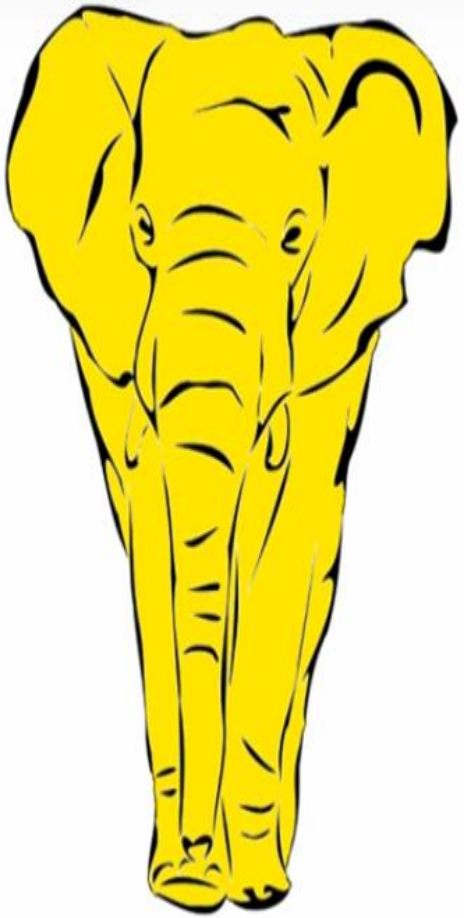
Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. The run-time system takes care of the details of partitioning the input data, scheduling the program's execution across a set of machines, handling machine failures, and managing the required inter-machine communication. This allows programmers without any experience with parallel and distributed systems to easily utilize the resources of a large distributed system.

Our implementation of MapReduce runs on a large cluster of commodity machines and is highly scalable: a typical MapReduce computation processes many terabytes of data on thousands of machines. Programmers find the system easy to use: hundreds of MapReduce programs have been implemented and upwards of one thousand MapReduce jobs are executed on Google's clusters every day.

Appeared in:
OSDI'04: Sixth Symposium on Operating System Design and Implementation,
San Francisco, CA, December, 2004.

Download: PDF Version

The Center of Applied Data Science



Hadoop 1.0.0 (2011)

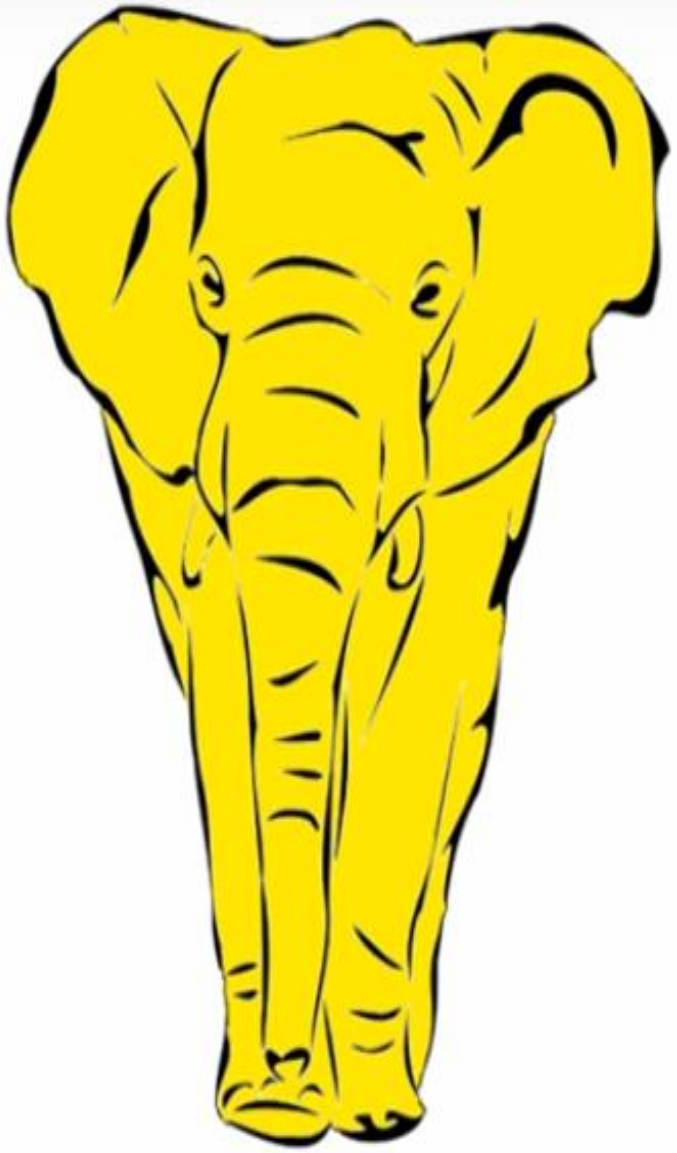
- A distributed file system (HDFS)
- A distributed programming framework (Map Reduce)

Hadoop 2.0.0 (2012)

- A distributed file system (HDFS)
- A distributed programming framework (Map Reduce)
- YARN

Hadoop 3.0.0 alpha2 (2017)

Version	Release date
3.1.2	2019 Feb 6
3.2.0	2019 Jan 16
2.9.2	2018 Nov 19
2.8.5	2018 Sep 15
2.7.7	2018 May 31



Hadoop Core

- A distributed file system (HDFS)
- A distributed programming framework (Map Reduce)
- YARN

Hadoop Ecosystem

- Hive
- Pig
- HBase
- Spark
- Sqoop
- Flume
- Kafka
- Oozie

https://hadoop.apache.org

Apache Hadoop | Download | Documentation | Community | Development | Help | Old site | Apache Software Foundation

Apache Hadoop

The Apache™ Hadoop® project develops open-source software for reliable, scalable, distributed computing.

The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.

Learn more » | Download » | Getting started »

Latest news

Apache Hadoop	2019 May 7

Modules

The project includes these modules:

- **Hadoop Common**: The common utilities that support the other Hadoop modules.
- **Hadoop Distributed File System (HDFS™)**: A

Related projects

Other Hadoop-related projects at Apache include:

- **Ambari™**: A web-based tool for provisioning, managing, and monitoring Apache Hadoop clusters which includes support for Hadoop HDFS, Hadoop

The Center of Applied Data Science

Introduction to Apache Hadoop

Overview

Open source programs and frameworks which can be used as the backbone of the big data operations.

Advantages

- Scalability
- Reliability
- Flexibility

Initiative

Longer time is needed to read the data when the physical storage devices become bigger.

Instead, many smaller storage devices working in parallel are more efficient than a large storage device.

INTRICITY
simplifying complexity

Presented by Jared Hillam, EIM Practice Director

COMPARING HADOOP & SQL

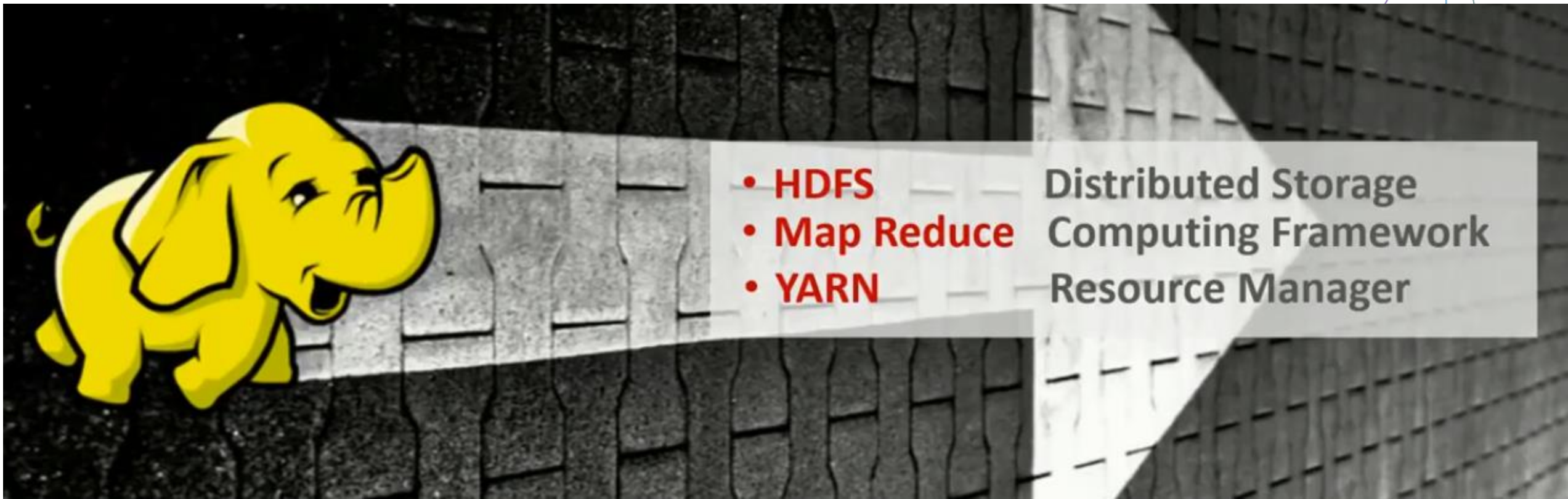
The Center of Applied Data Science

QUIZ 1

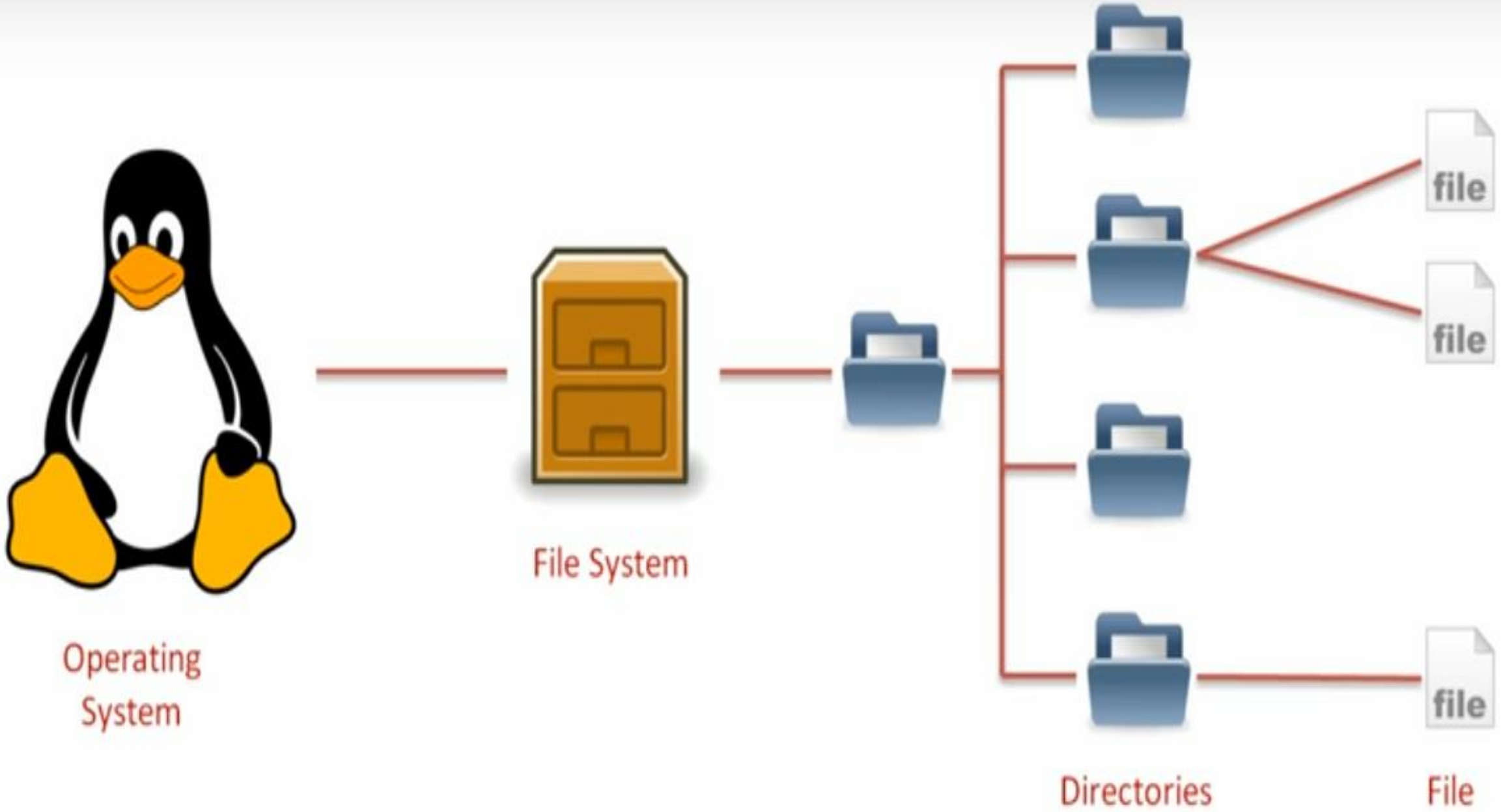
- https://forms.gle/y6uFf7yKBM1PAtUSA

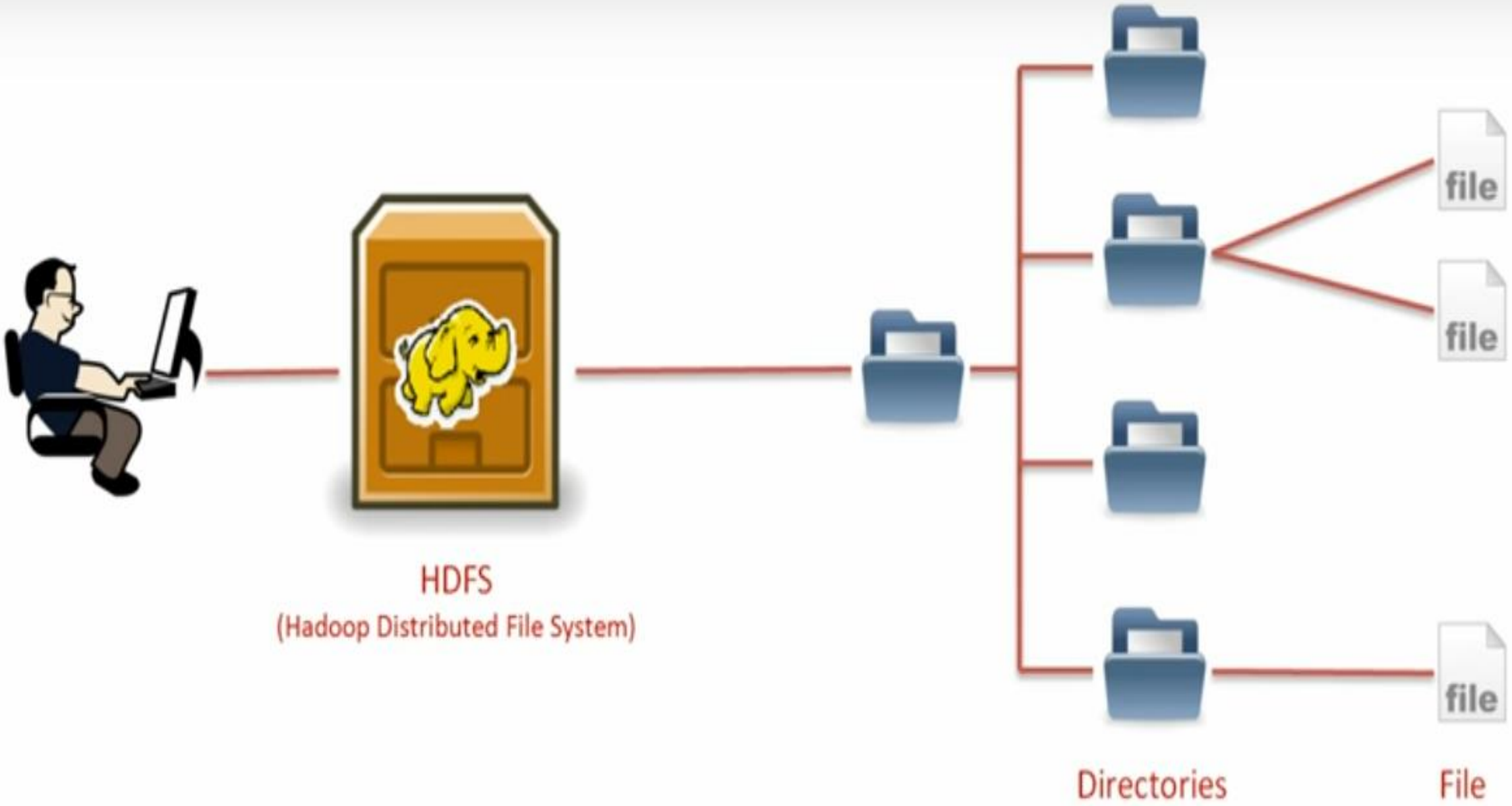
The Center of Applied Data Science

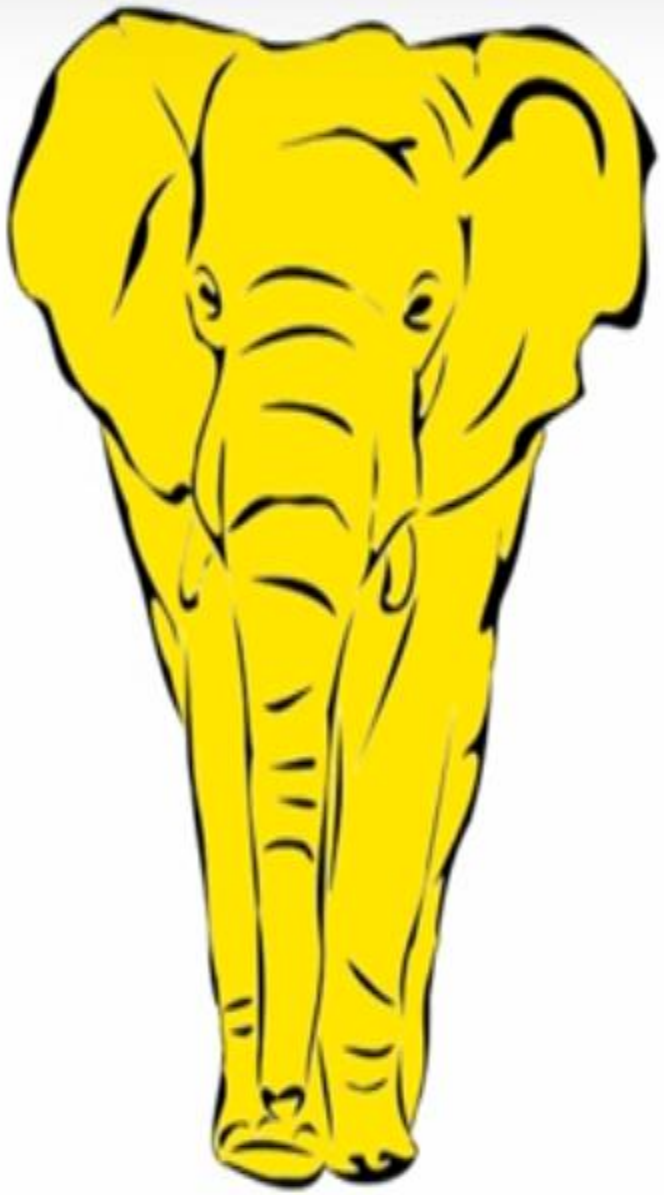
Principles of Hadoop



1. What is HDFS?
2. What can we do with HDFS?
3. What are the distinctive features of HDFS?

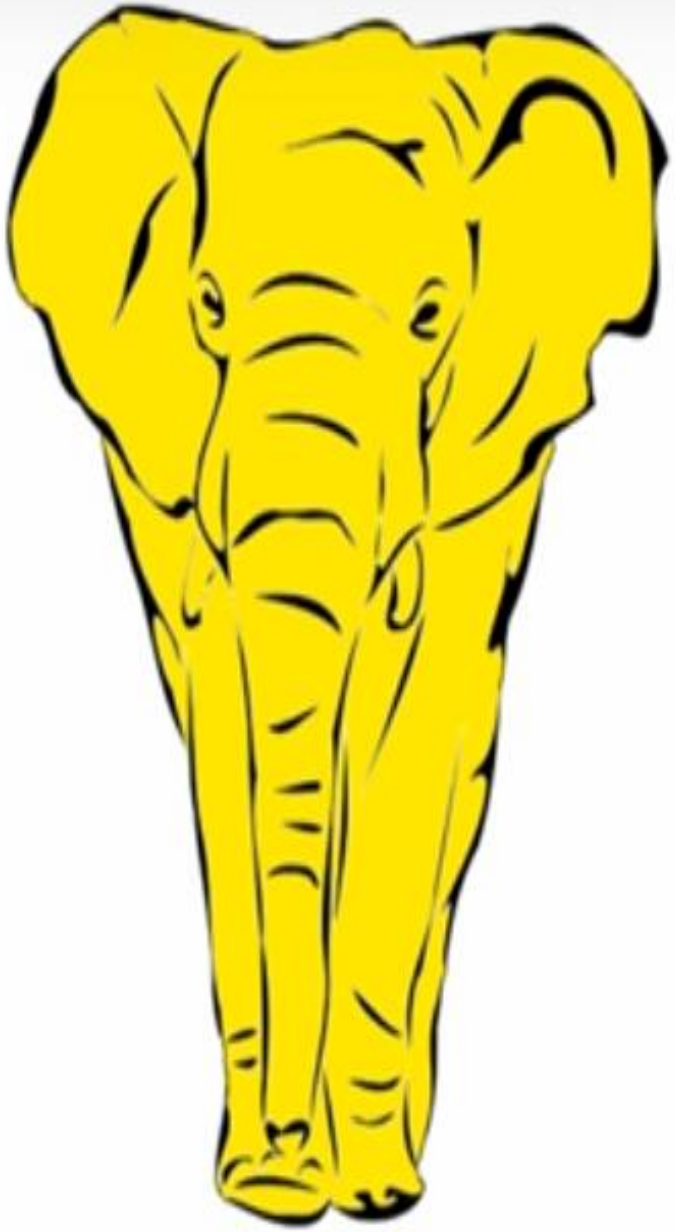

file
file
file
Operating System
File System
Directories
File


file
file
file
HDFS
(Hadoop Distributed File System)
Directories
File



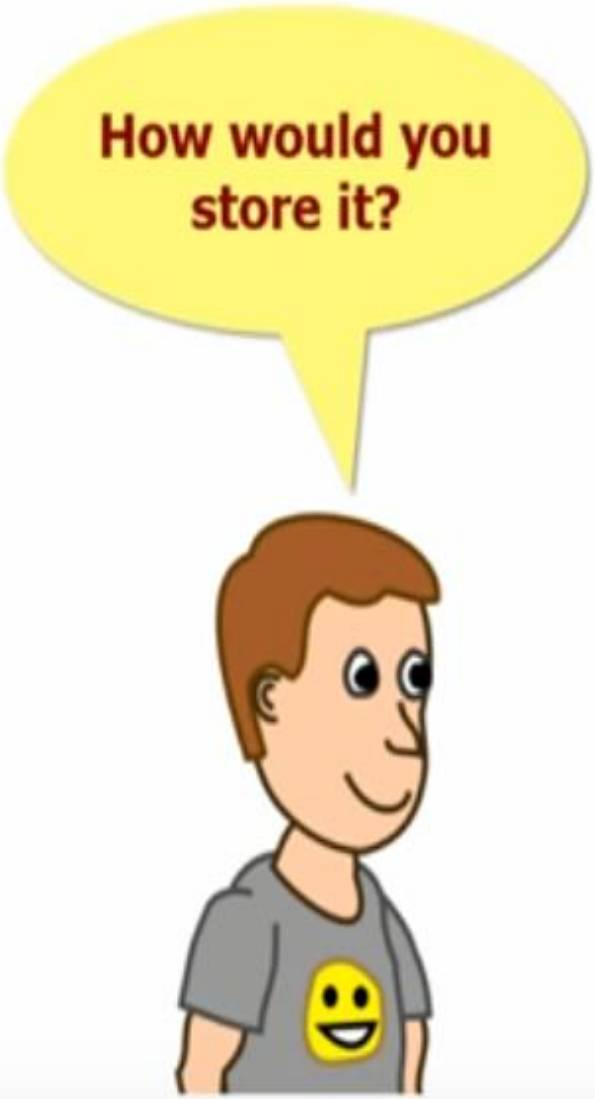
HDFS - Features

1. Distributed
2. Scalable
3. Cost Effective
4. Fault Tolerant
5. High Throughput



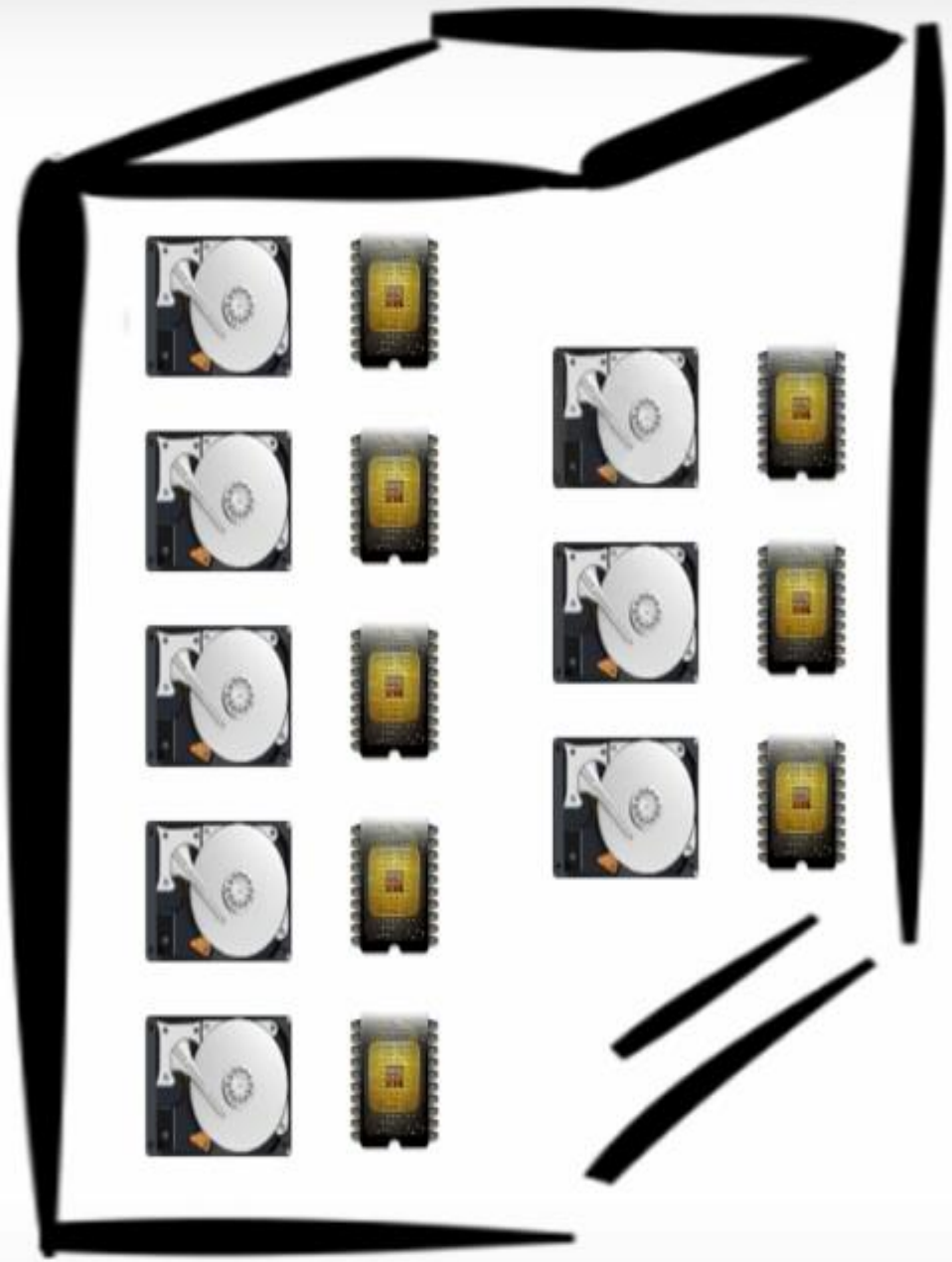
HDFS - Features

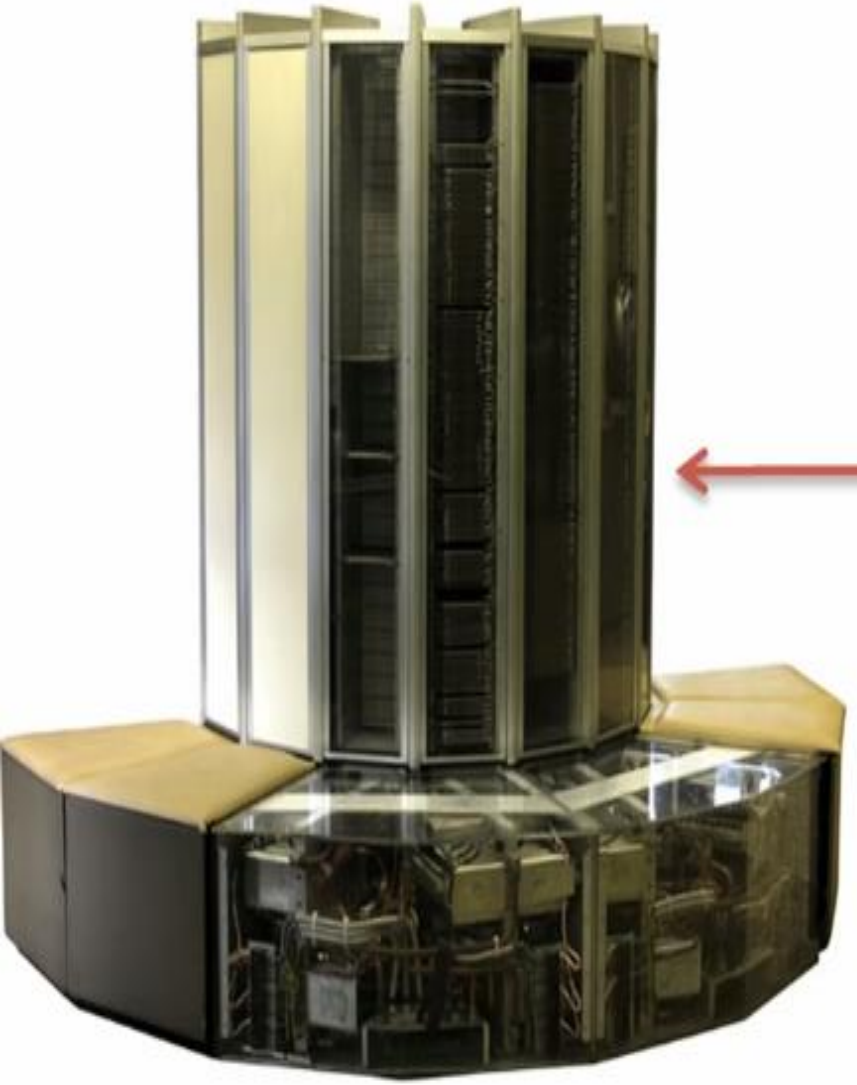
1. Distributed
2. Scalable
3. Cost Effective
4. Fault Tolerant
5. High Throughput

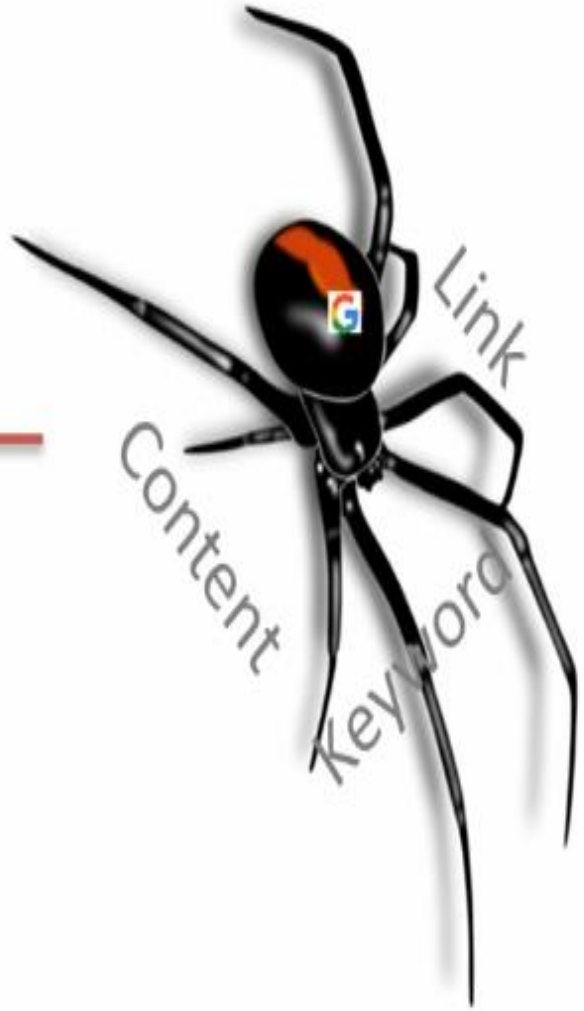

How would you store it?

Buy a bigger one or add another disk.

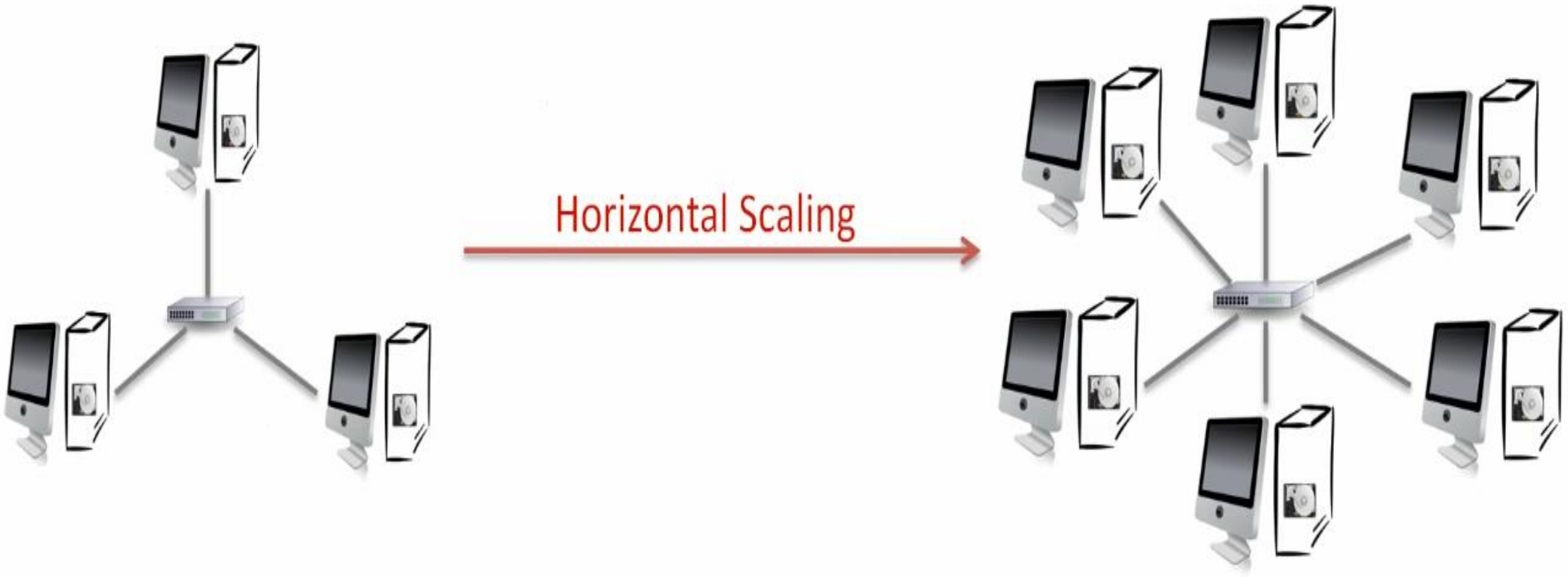
Vertical Scaling





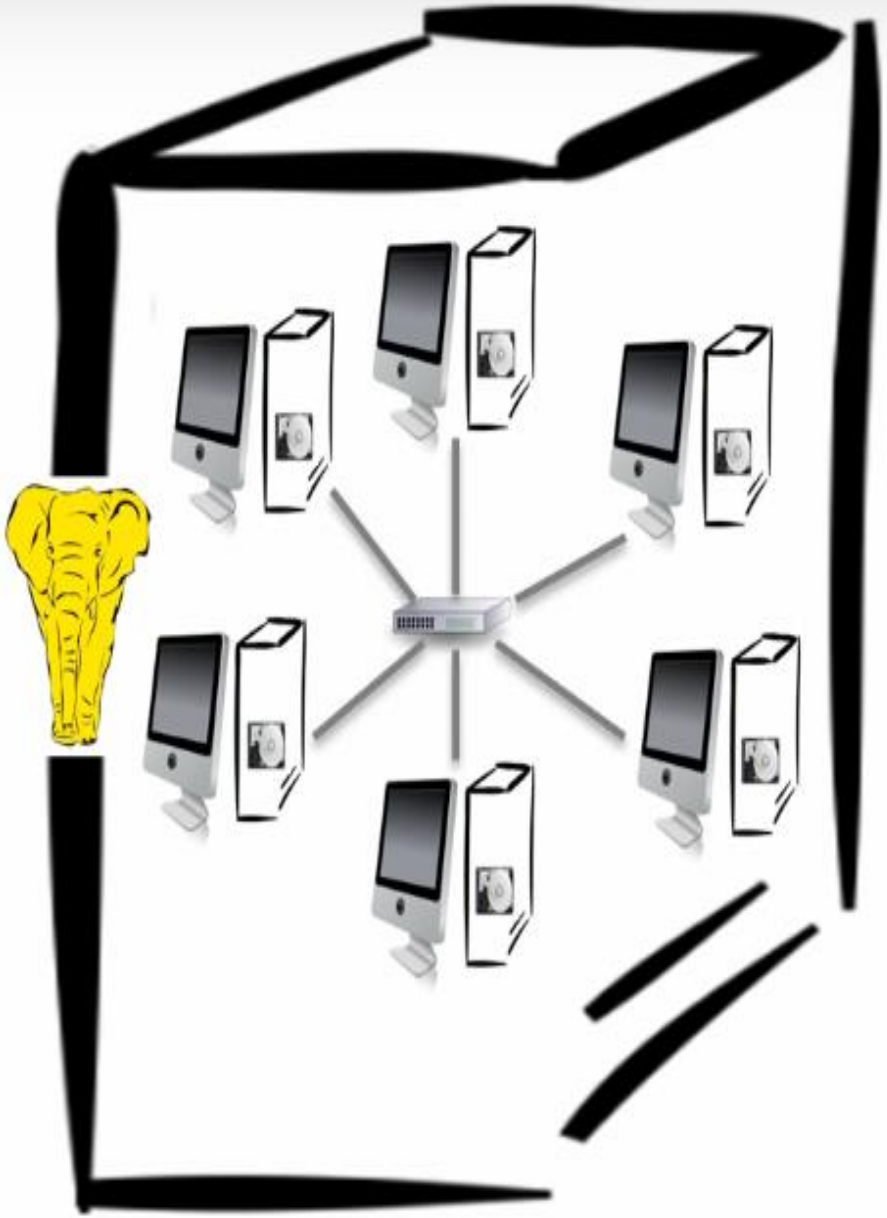

Link
Content
Keyword

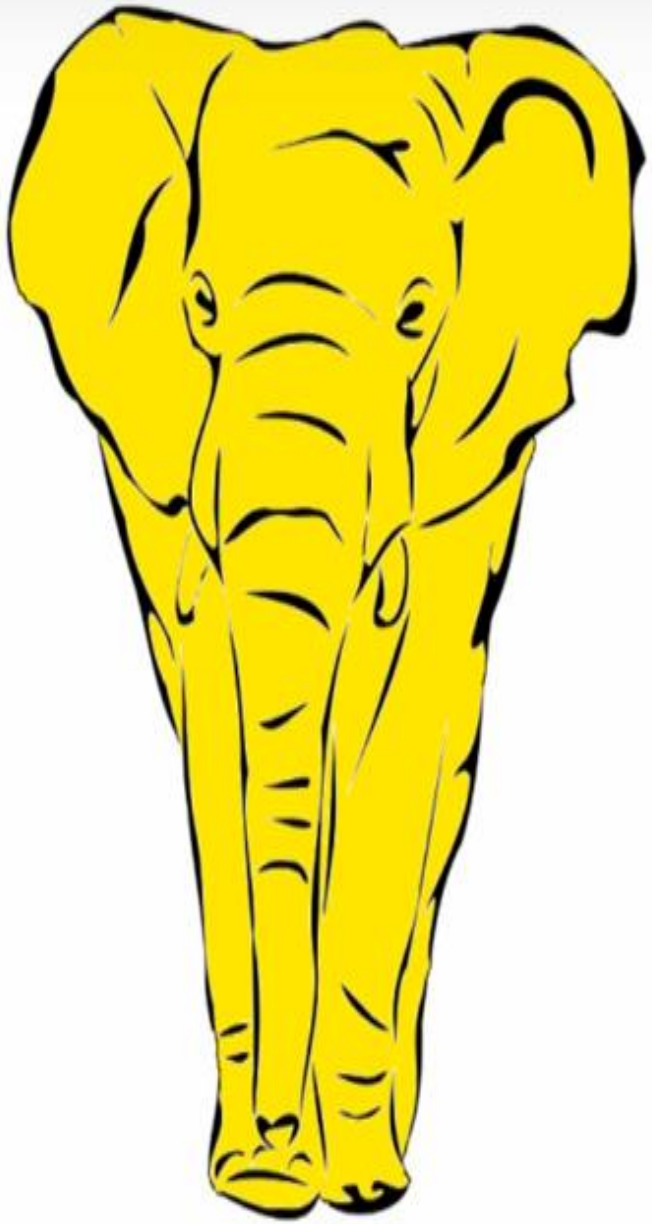
WWW


Horizontal Scaling

Store your data

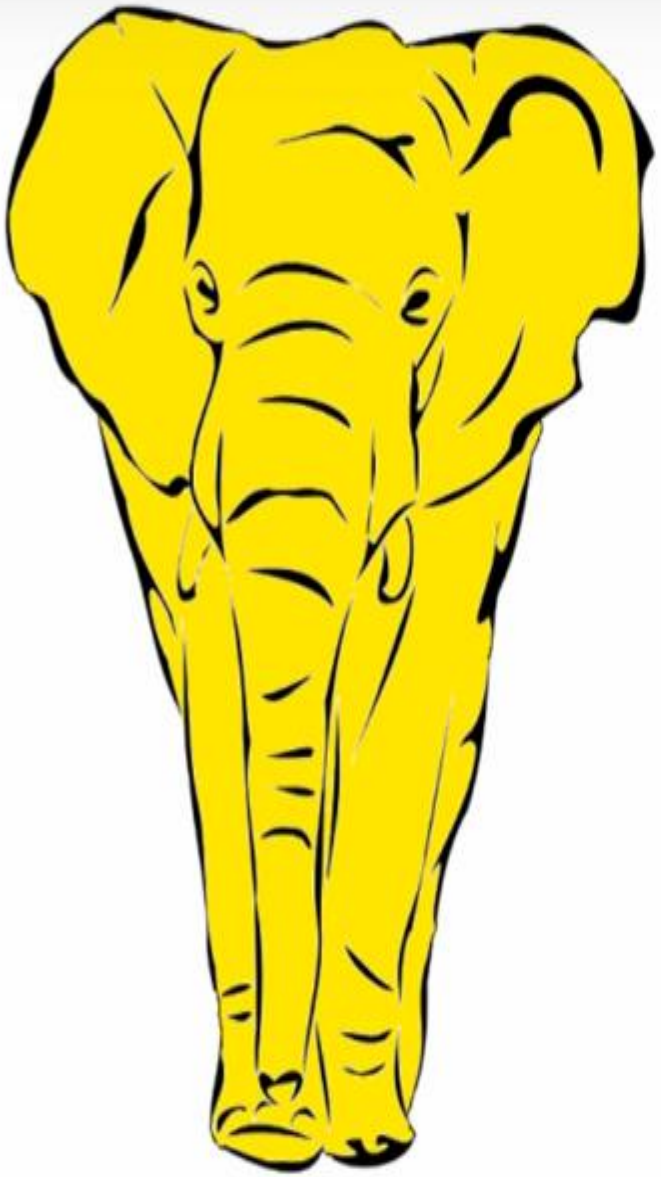
Access it like a single large computer





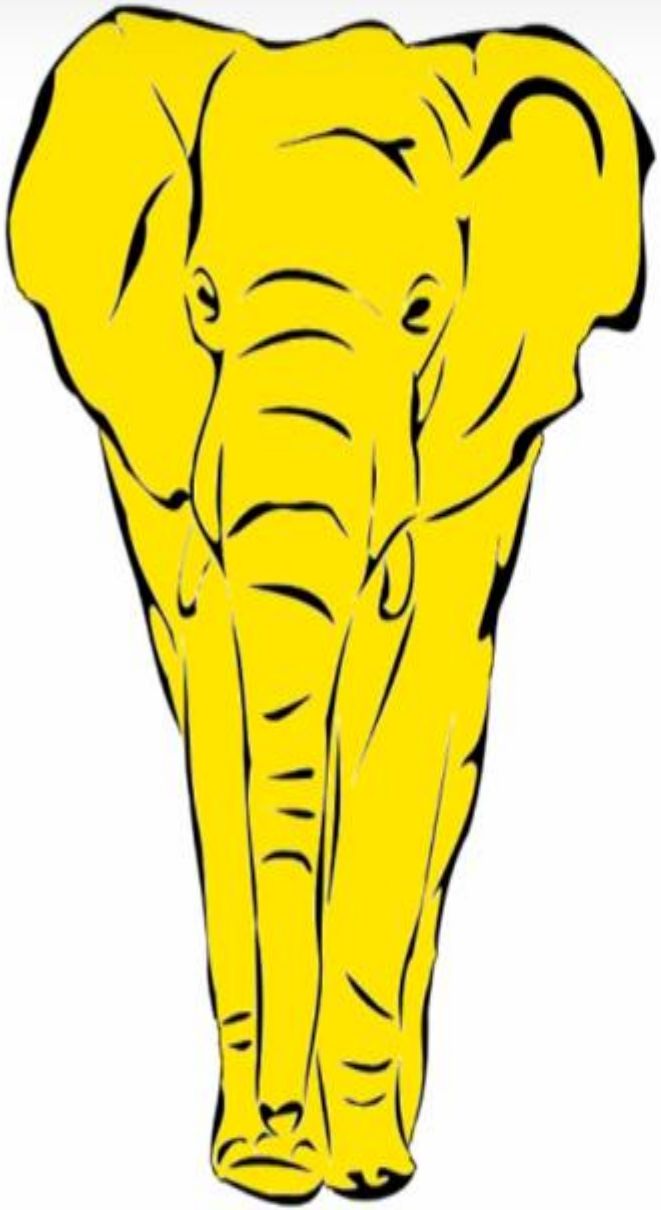
HDFS - Features

1. Distributed
2. Scalable
3. Cost Effective
4. Fault Tolerant
5. High Throughput



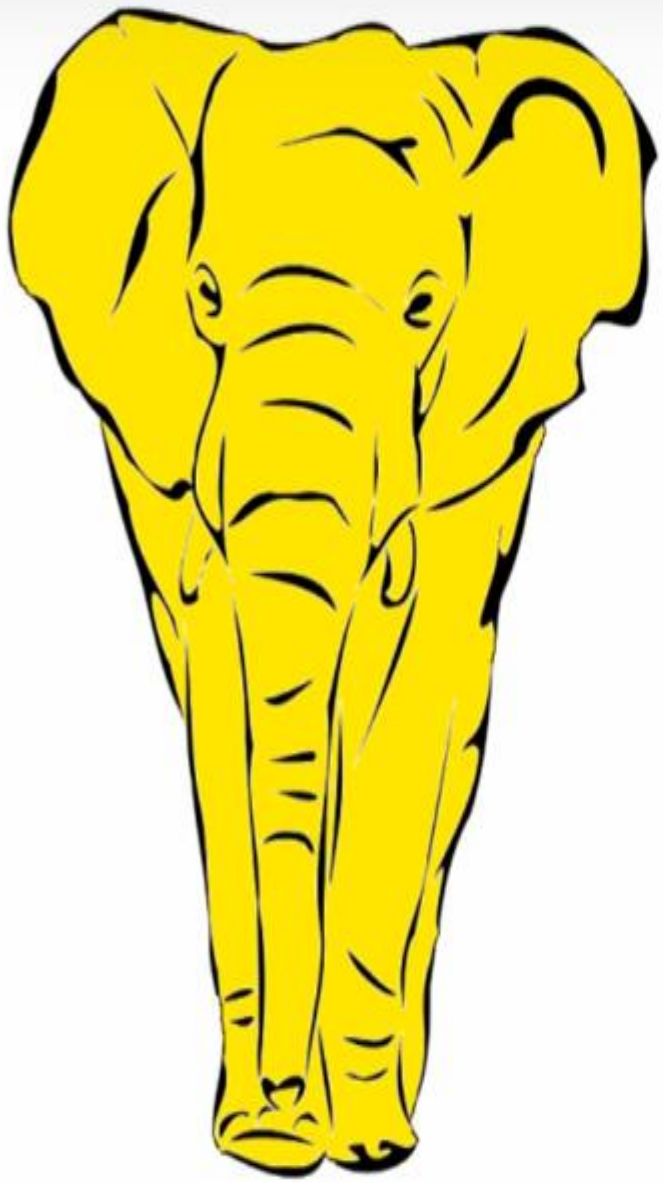
HDFS - Features

1. Distributed
2. Scalable
3. Cost Effective
4. Fault Tolerant
5. High Throughput



HDFS - Features

1. Distributed
2. Scalable
3. Cost Effective
4. Fault Tolerant
5. High Throughput



HDFS - Features

1. Distributed
2. Scalable
3. Cost Effective
4. Fault Tolerant
5. High Throughput

Performance Measurement

Minimize this.
Lower is better.

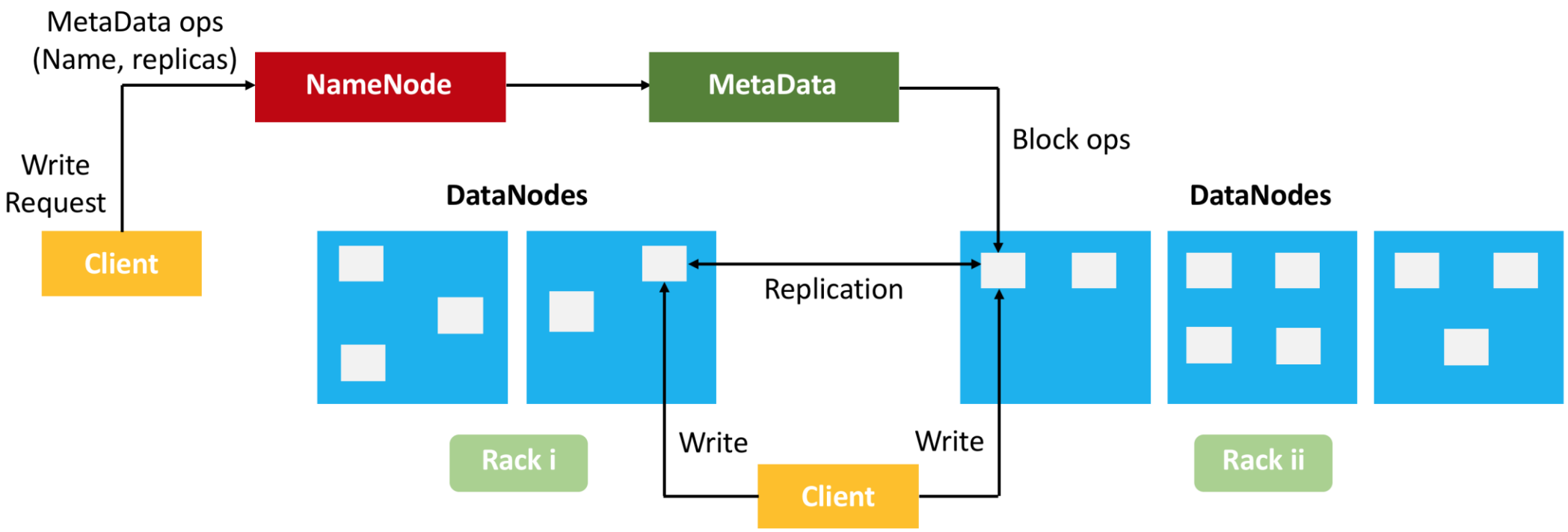
1. **Latency -** Time to get the first record
2. **Throughput -** Number of records processed per unit of time.

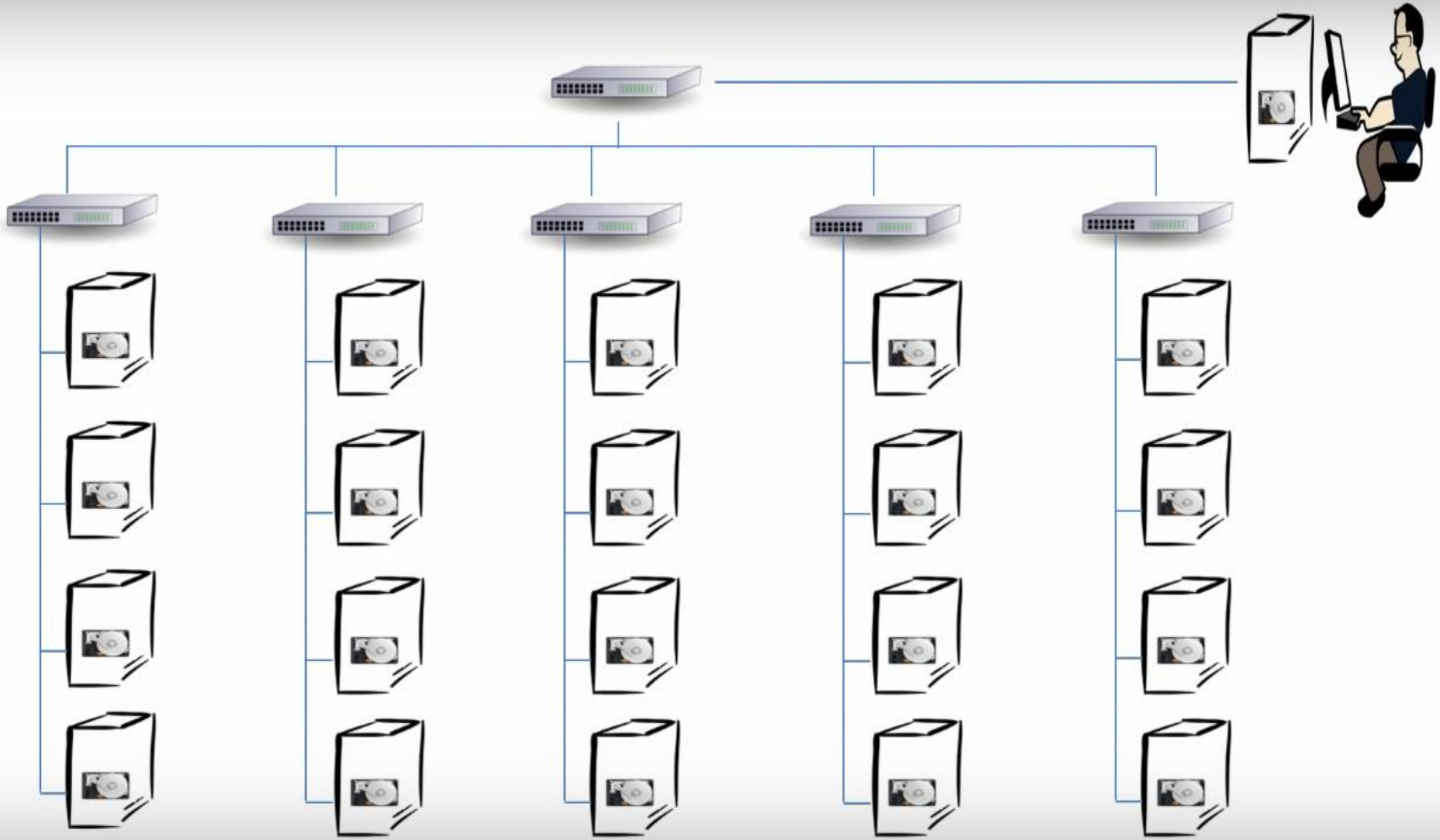
Maximize this.
Higher is better.

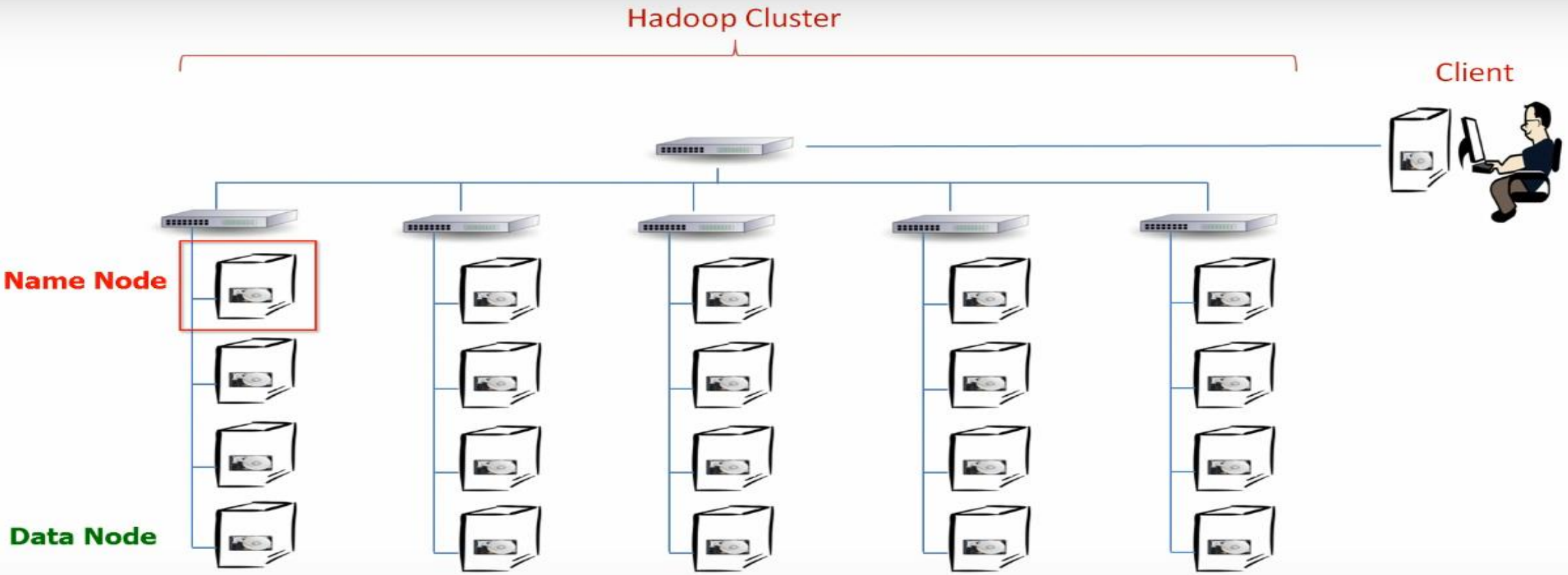
Overview

- **Reliable** architecture to store very large files in Hadoop cluster.
- **Store less number of large files** rather than huge number of small files.
- **Fault tolerance**.
- **High throughput** by providing data access in parallel.

HDFS Architecture

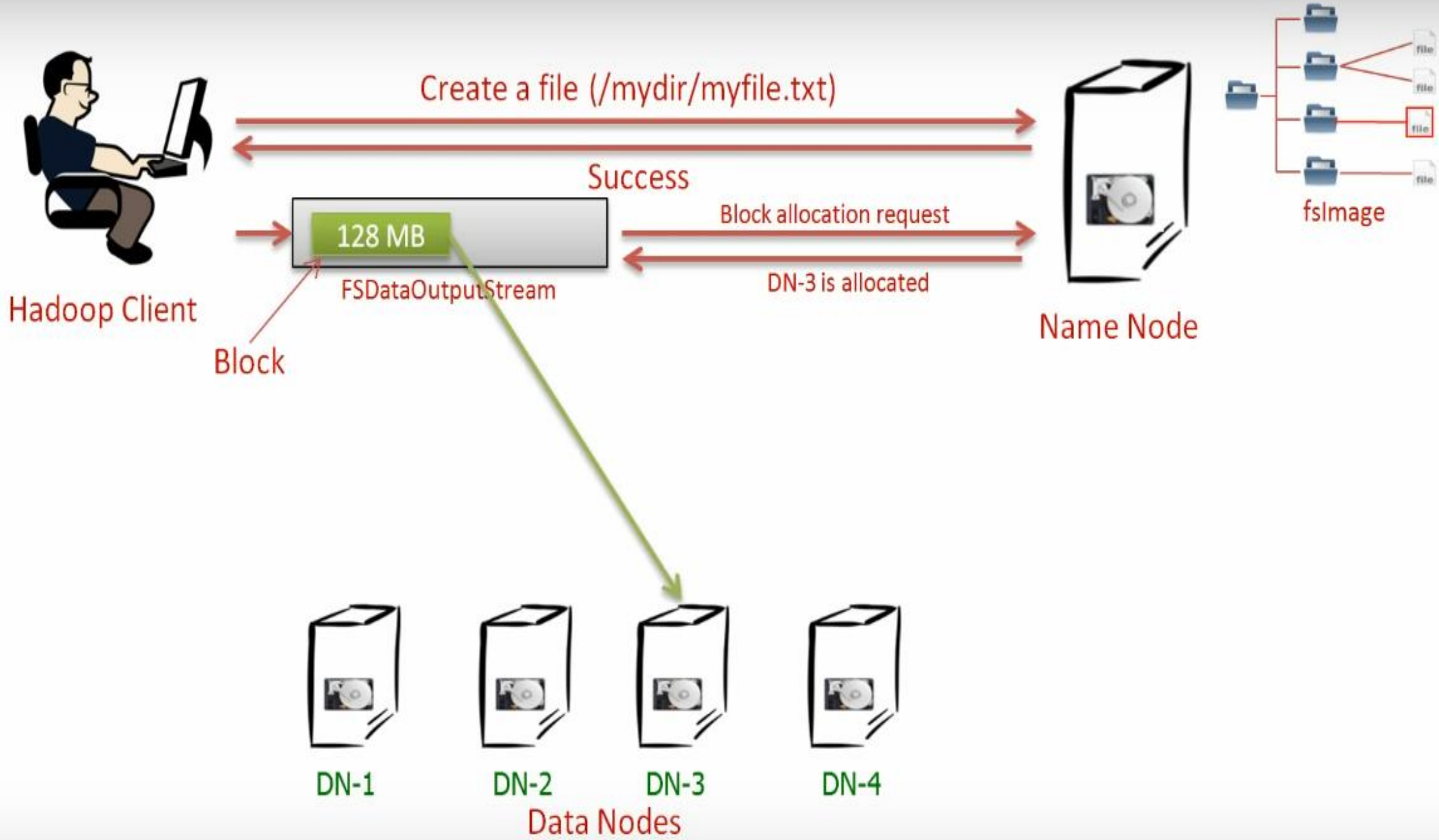


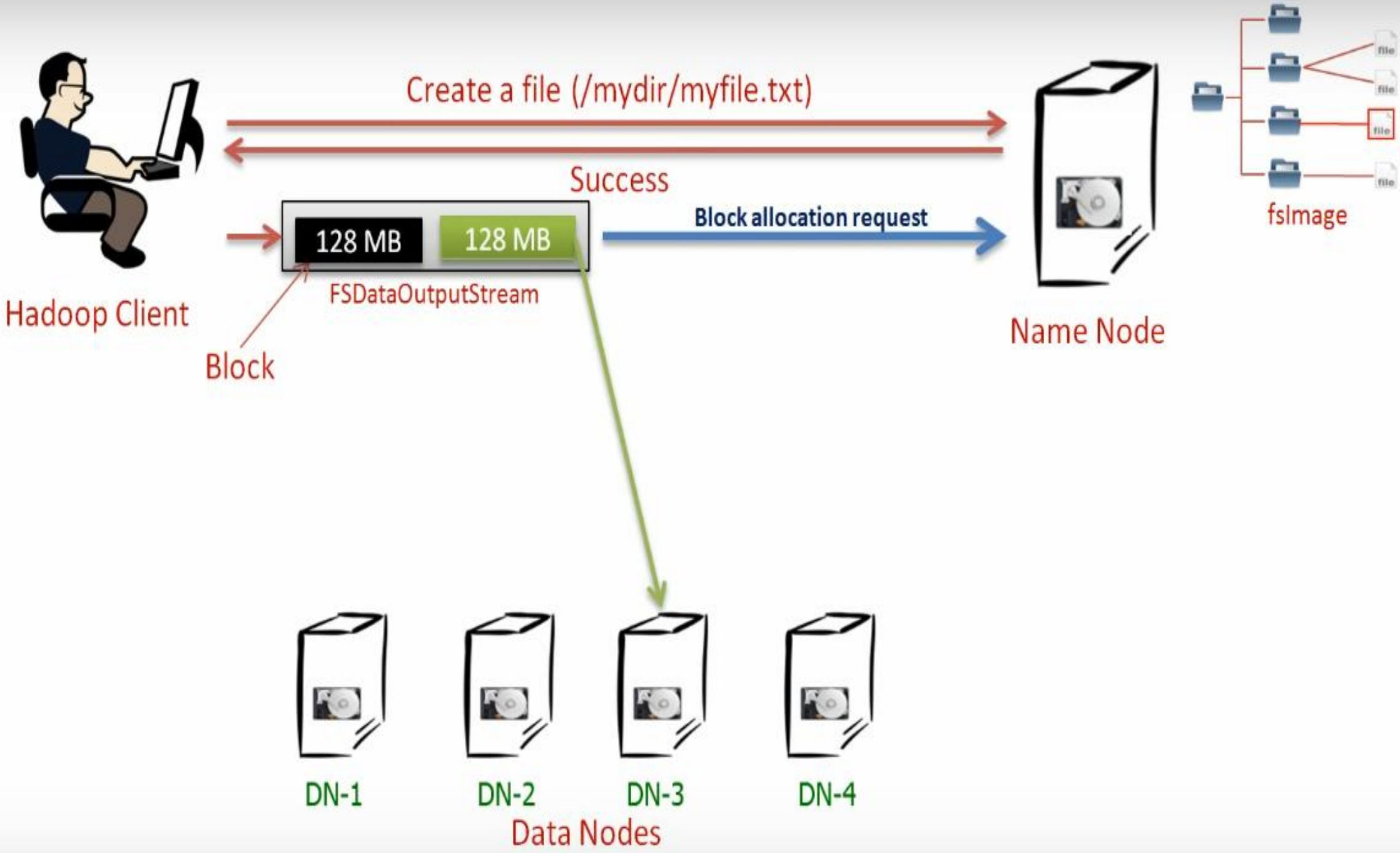


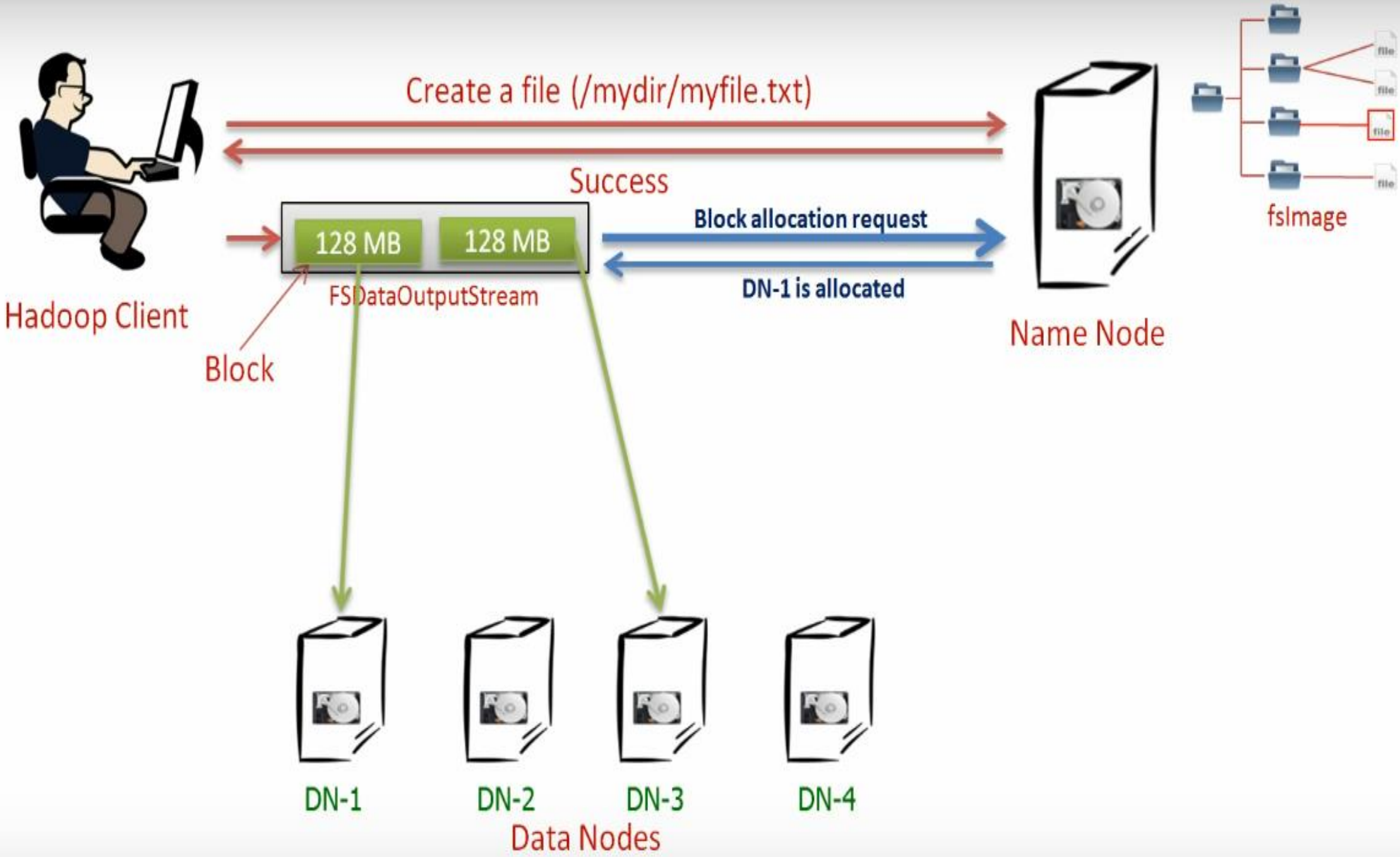


Name Node – Manages file system namespace.

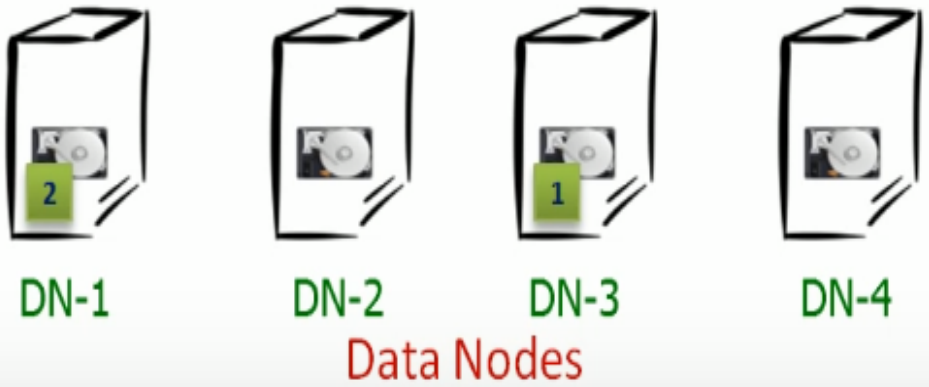
Data Node – Manages data


Create a file (/mydir/myfile.txt)
Success
128 MB
Block allocation request
DN-3 is allocated
FSDataOutputStream
Hadoop Client
Block
Name Node
fsImage
DN-1
DN-2
DN-3
DN-4
Data Nodes


Create a file (/mydir/myfile.txt)
Success
Block allocation request
128 MB
128 MB
FSDataOutputStream
Hadoop Client
Block
Name Node
fsImage
DN-1
DN-2
DN-3
DN-4
Data Nodes


Create a file (/mydir/myfile.txt)
Success
Block allocation request
DN-1 is allocated
128 MB
128 MB
FSDataOutputStream
Hadoop Client
Block
Name Node
fsImage
DN-1
DN-2
DN-3
DN-4
Data Nodes

Create a file (/mydir/myfile.txt)
Success
File Close
Hadoop Client
Name Node
fsImage
file
file
file
file

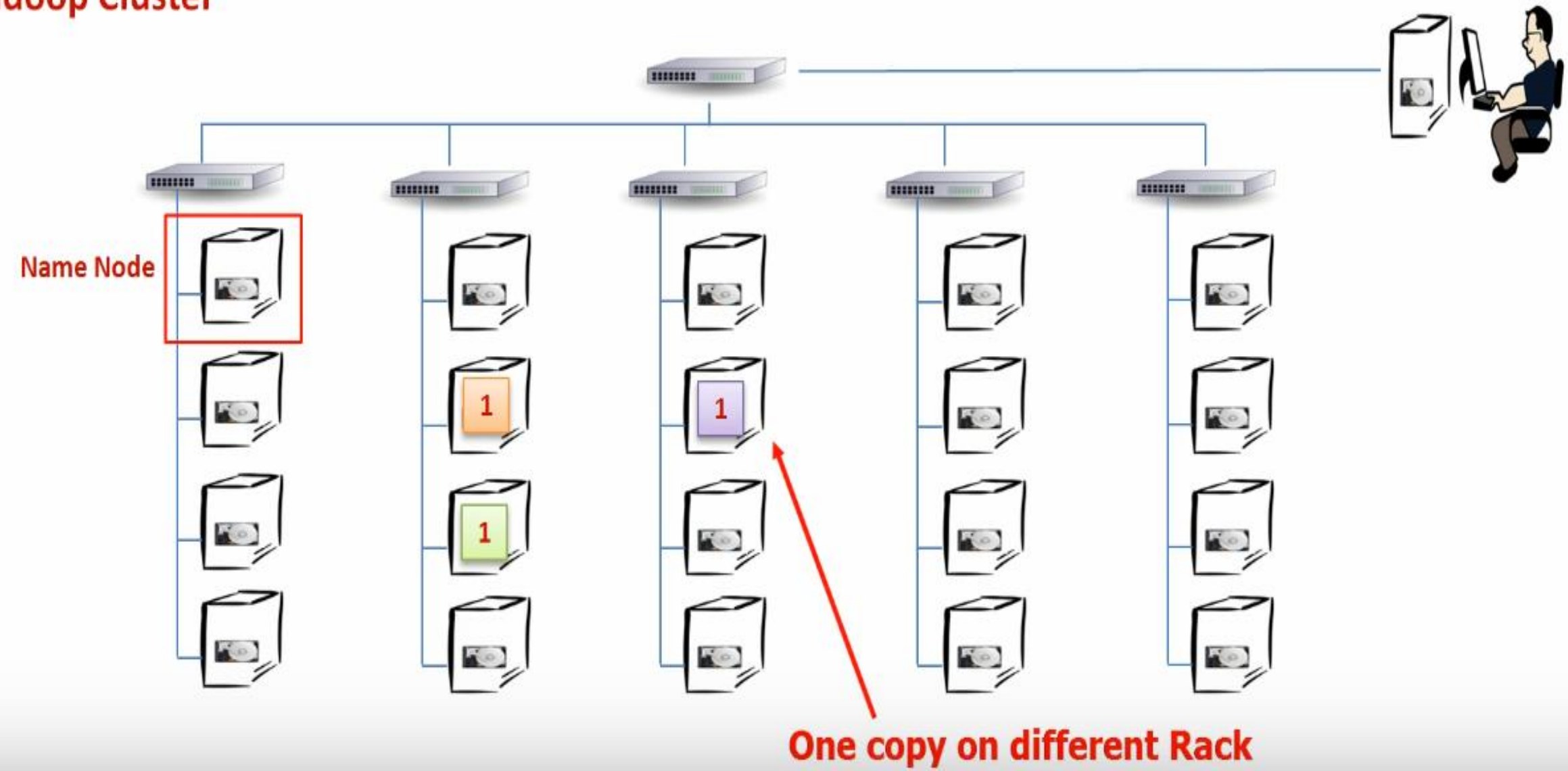

2
1
DN-1
DN-2
DN-3
DN-4
Data Nodes

- Master/Slave architecture
  - Single Name Node (Master)
  - One or more Data Nodes (Slaves)
- NN manages FS namespace
- All client interactions start with NN
- DN stores file data as Blocks
- DN sends heartbeat and block report to NN
- File is broken into Blocks and stored on DN
- NN maintains file to block mapping, location, order of blocks and other metadata.
- Default block size is 128 MB
- You can change block size for a file
- Client directly interacts with DN for reading/writing blocks
- Client buffers data locally to provide streaming read/write
- NN and DN can be installed on single machine to create a single node cluster for learning

Hadoop Cluster

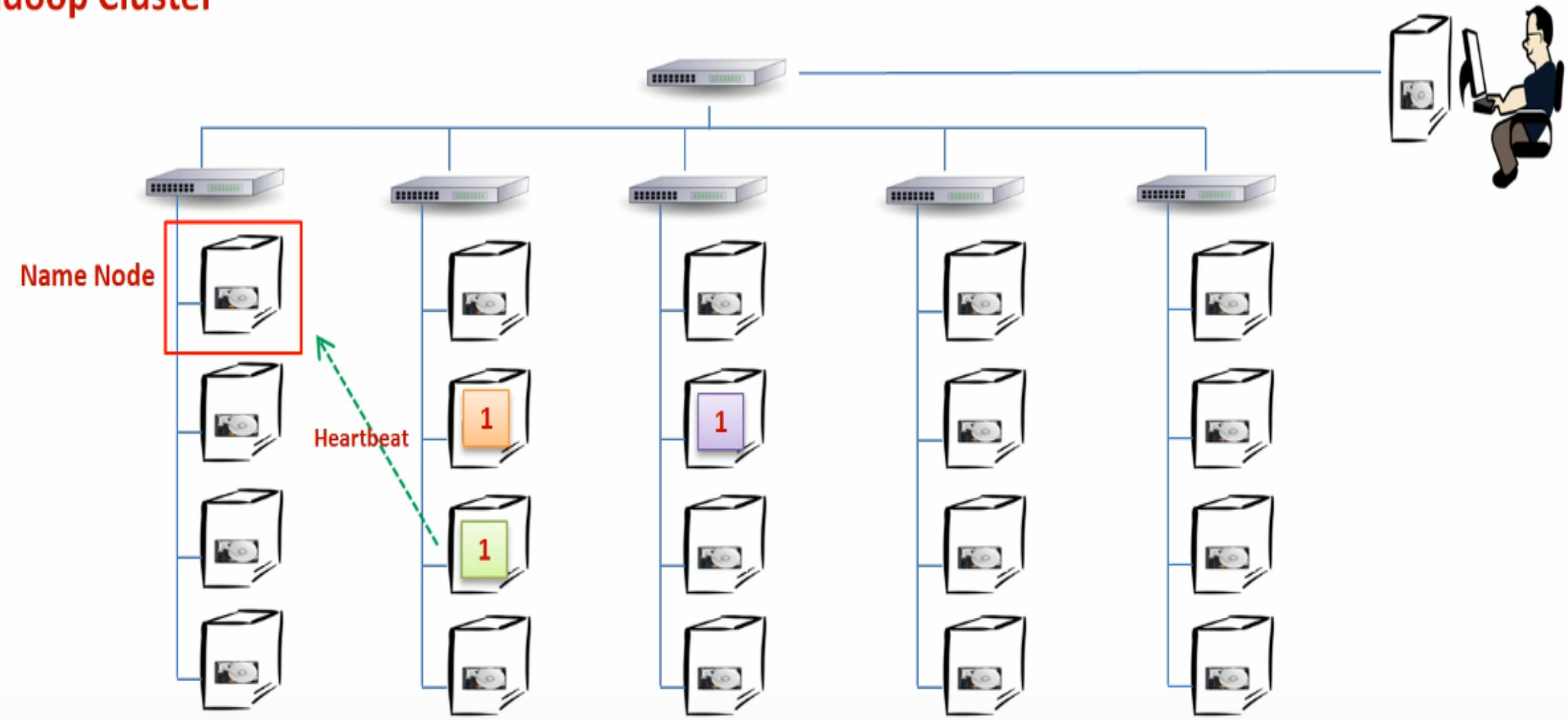
Replication Factor = 3

Rack Aware



Hadoop Cluster

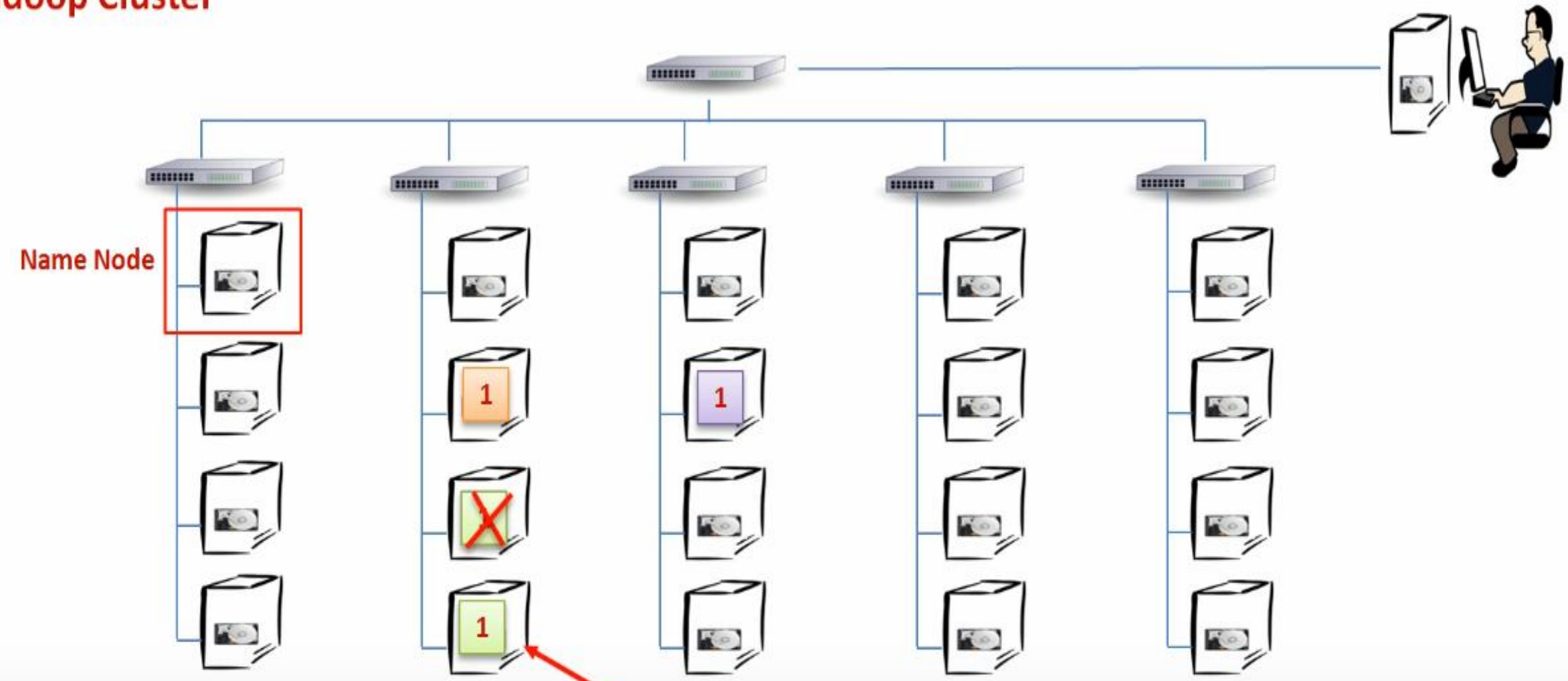
Replication Factor = 3
Rack Aware



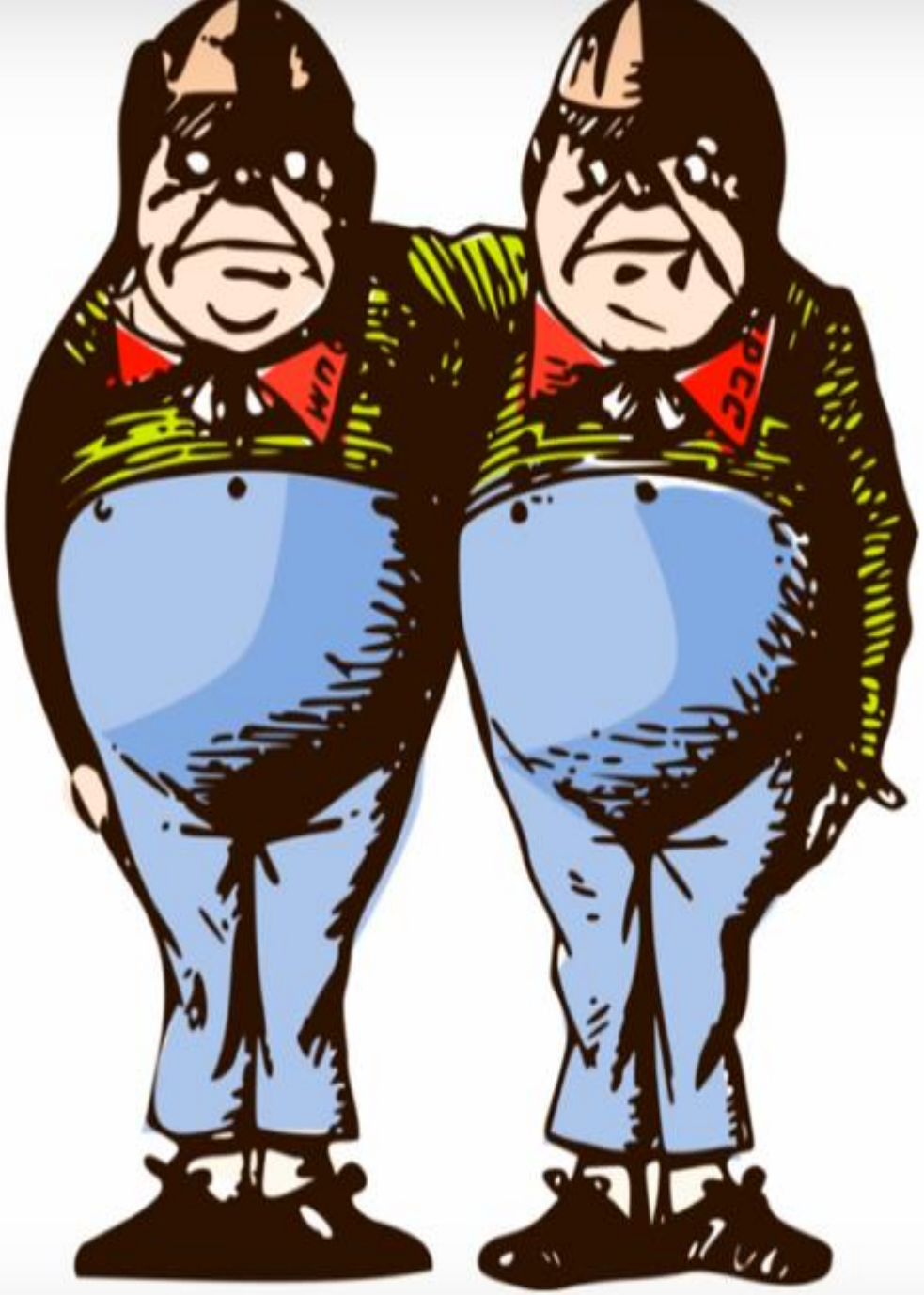
Hadoop Cluster

Replication Factor = 3

Rack Aware



Re-replicated to 3 due to one failure

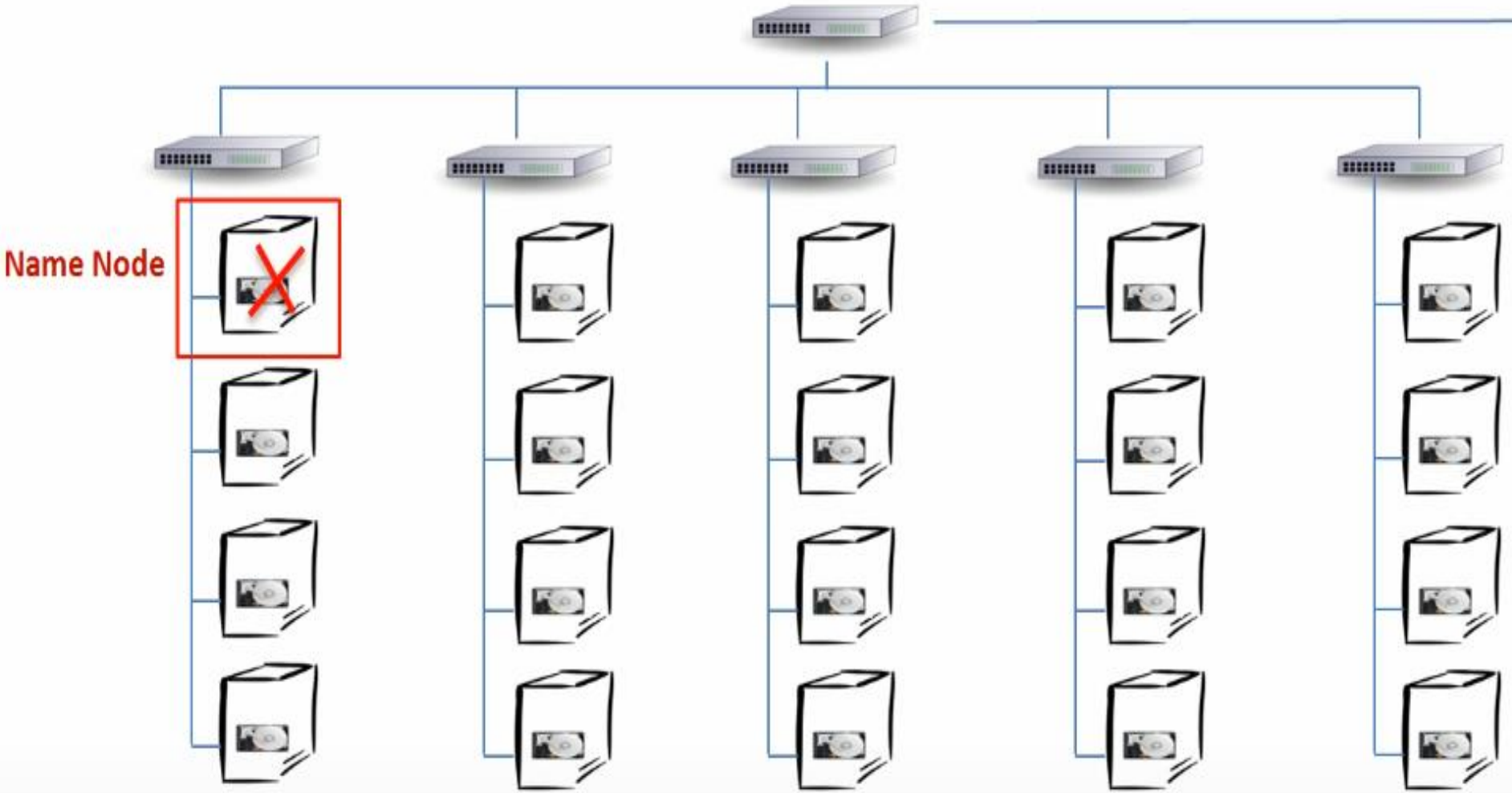


Replication is expensive

- Storage Policies – Save storage cost
- Erasure coding – an alternative to replication

Hadoop Cluster

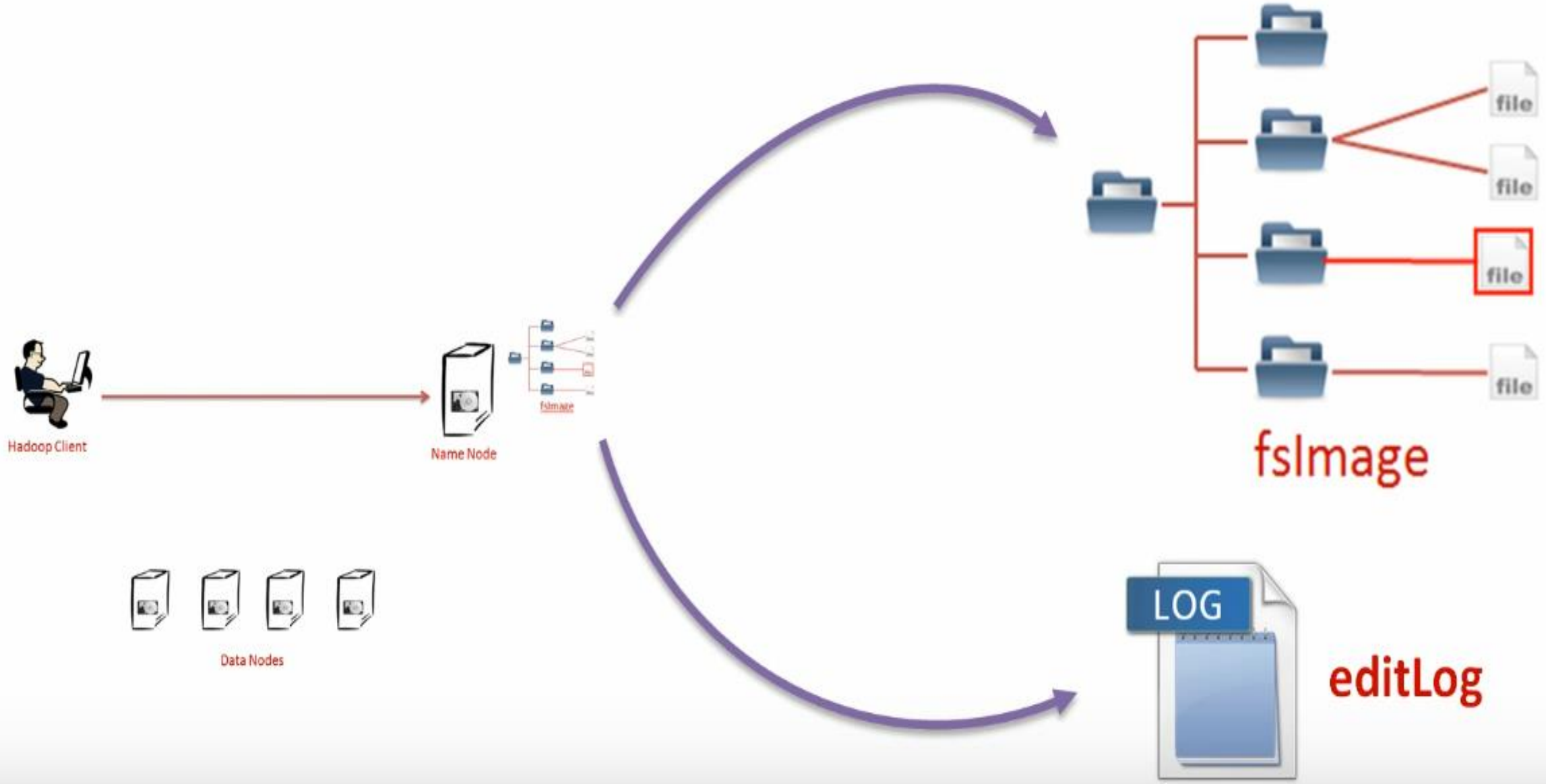
Oops!



How to protect against NN Failure?

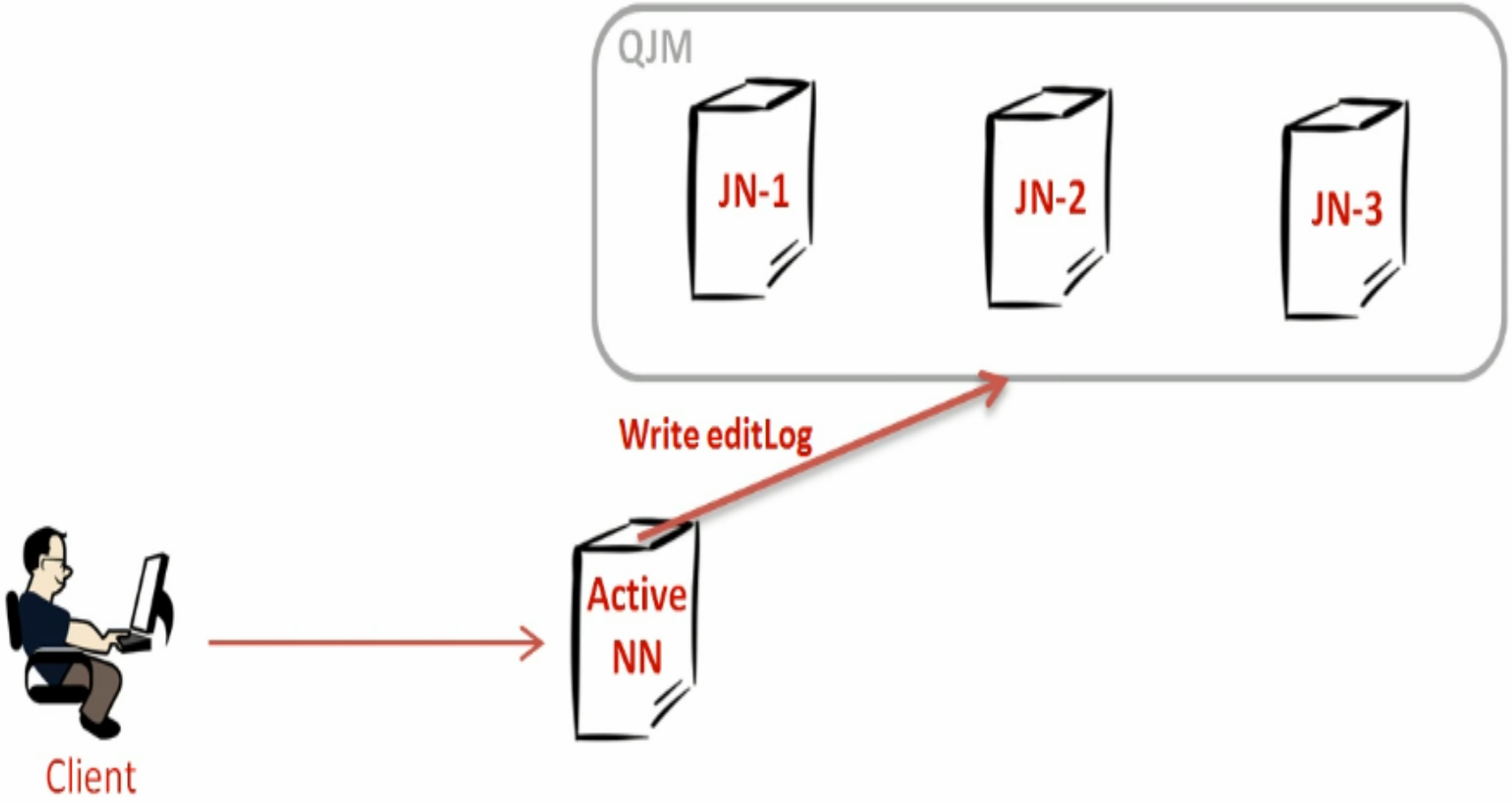
Backup following things

- HDFS Namespace information
- Standby Name node


Hadoop Client
Name Node
fsImage
Data Nodes
file
file
file
file
fsImage
LOG
editLog

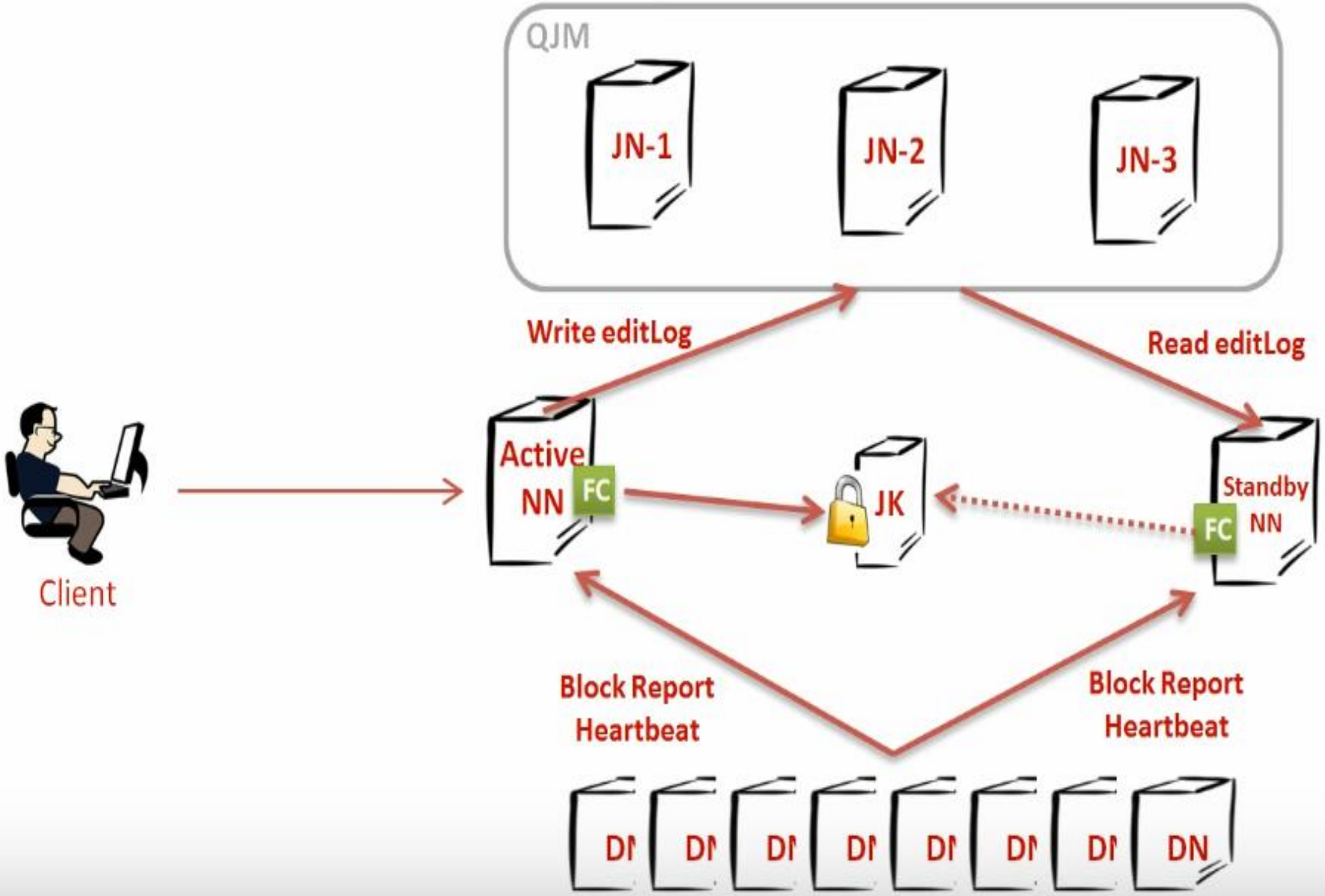
HA using QJM

(Quorum journal manager)



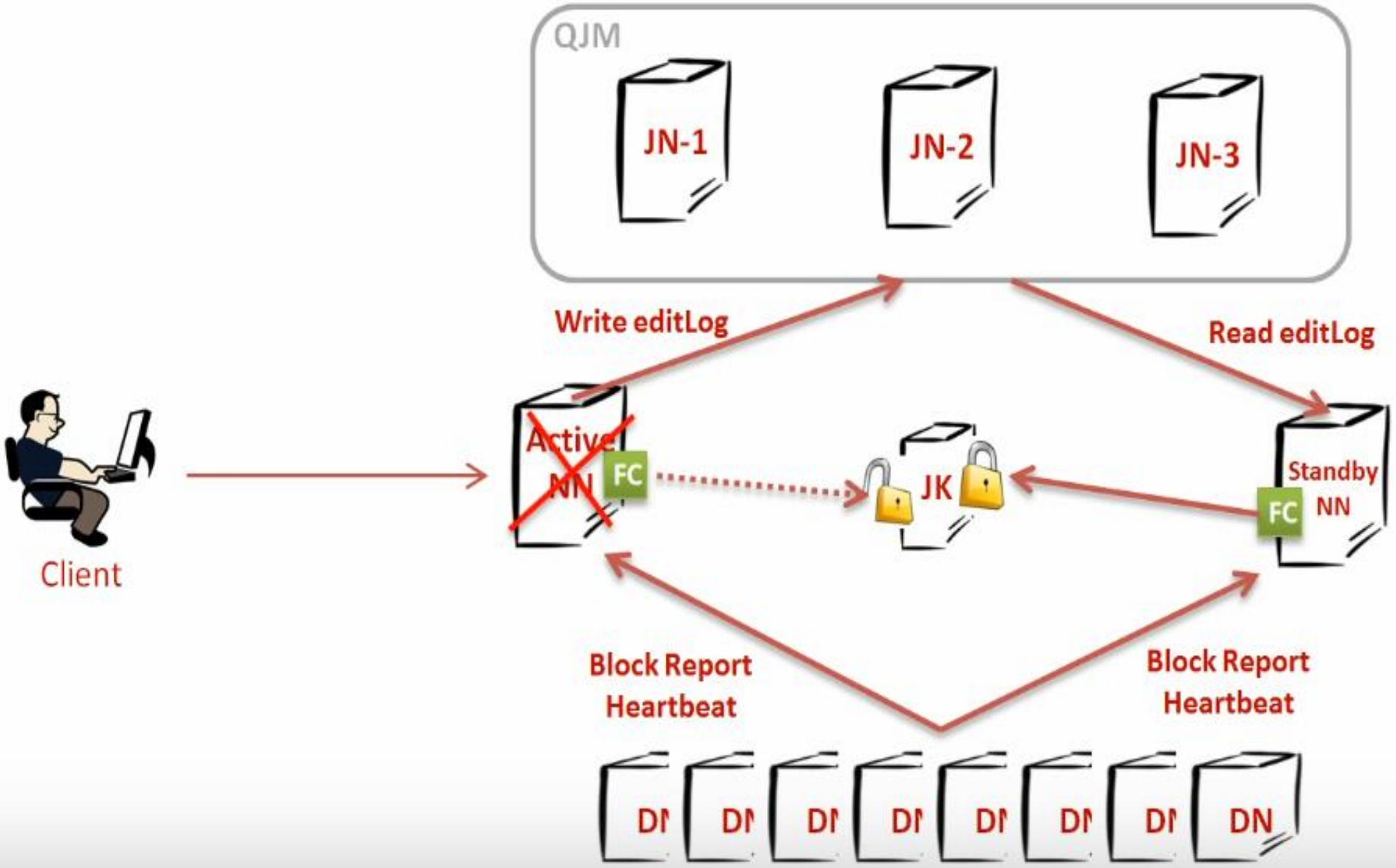
HA using QJM

(Quorum journal manager)



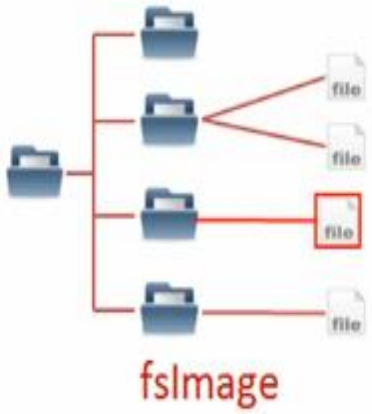
HA using QJM

(Quorum journal manager)



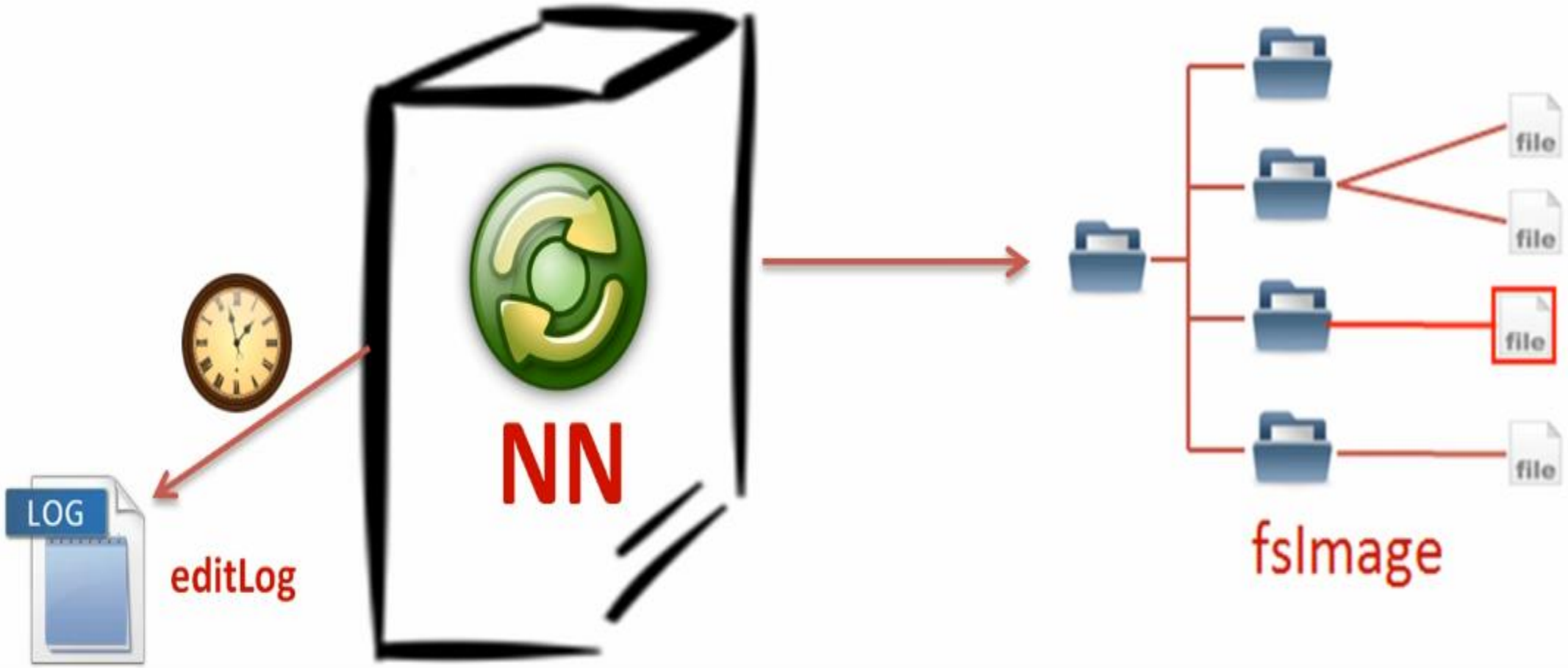
Secondary Name Node

- fsImage

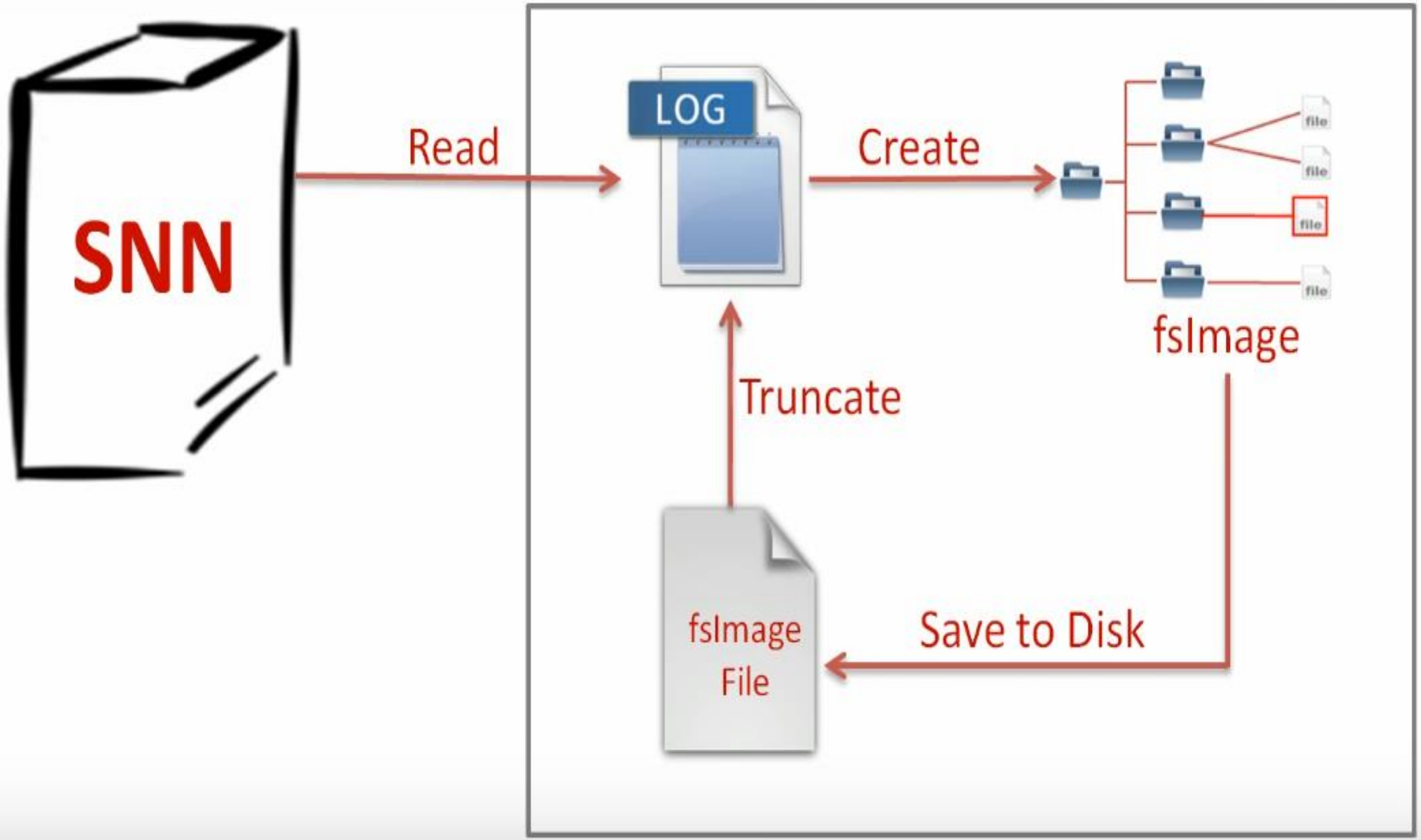


- editLog

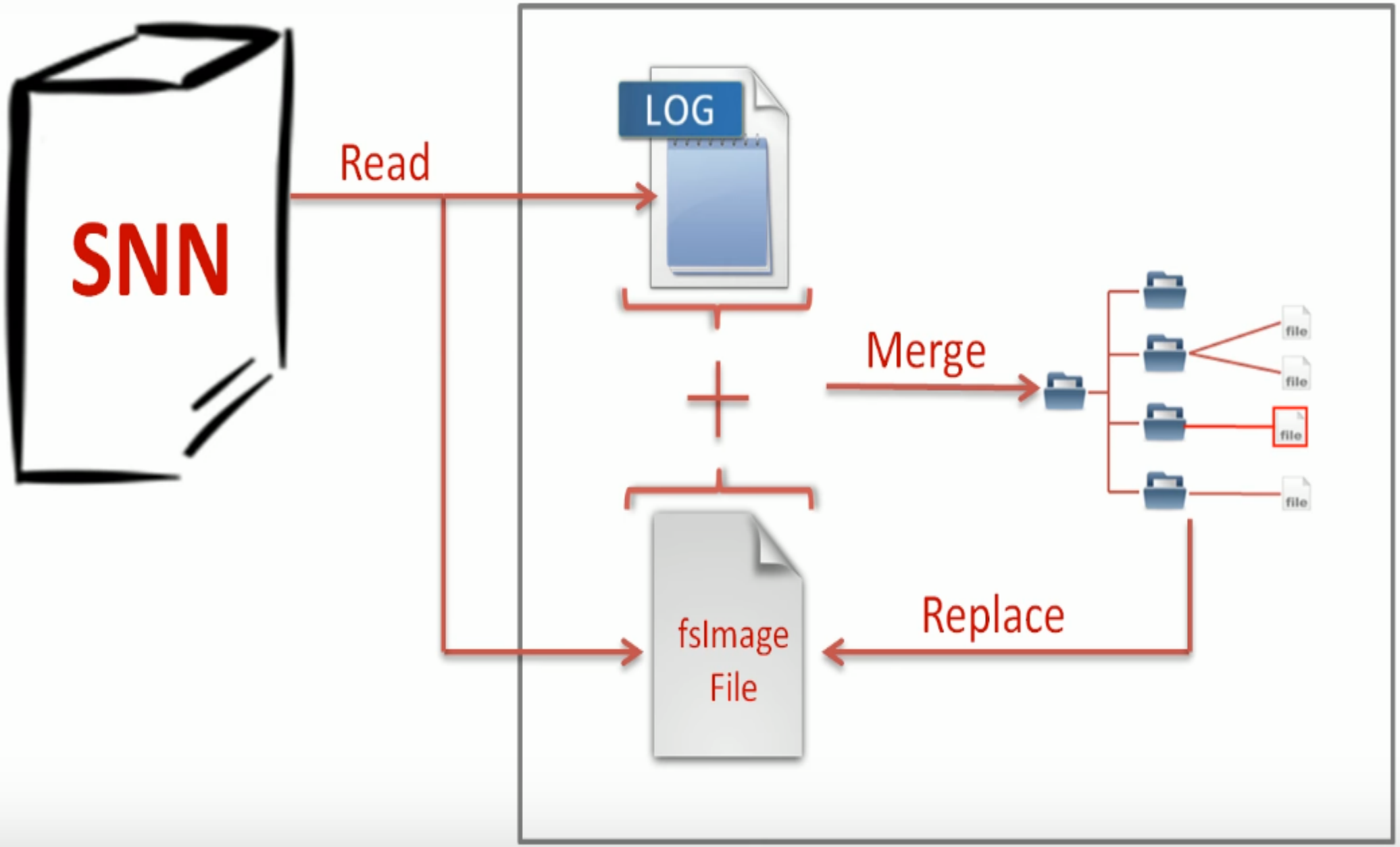
What if you restart NN?



Secondary NN Checkpoint (H1)

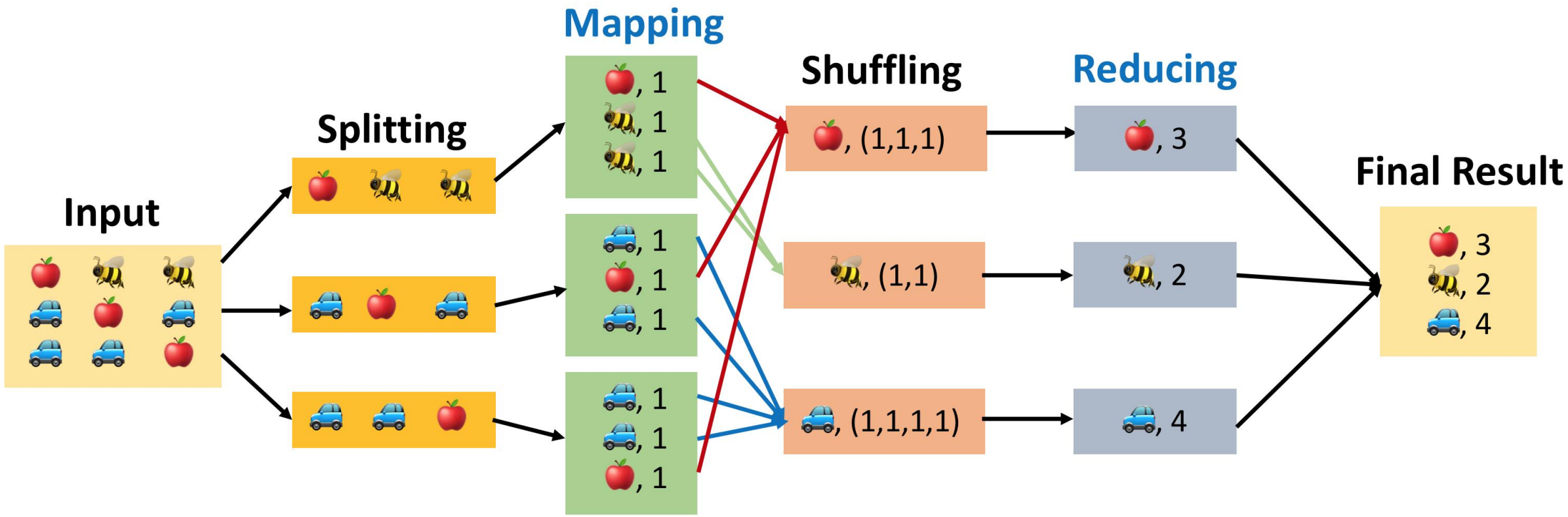


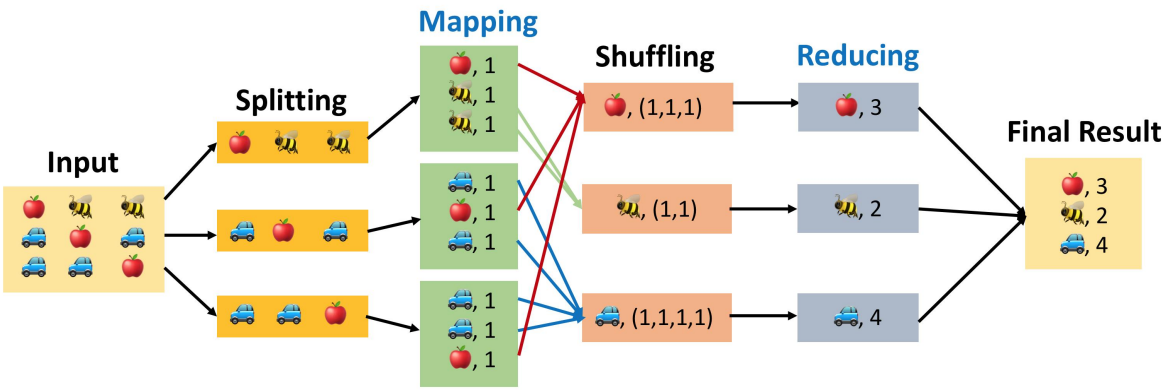
Secondary NN Checkpoint (H2)



Principles of Hadoop

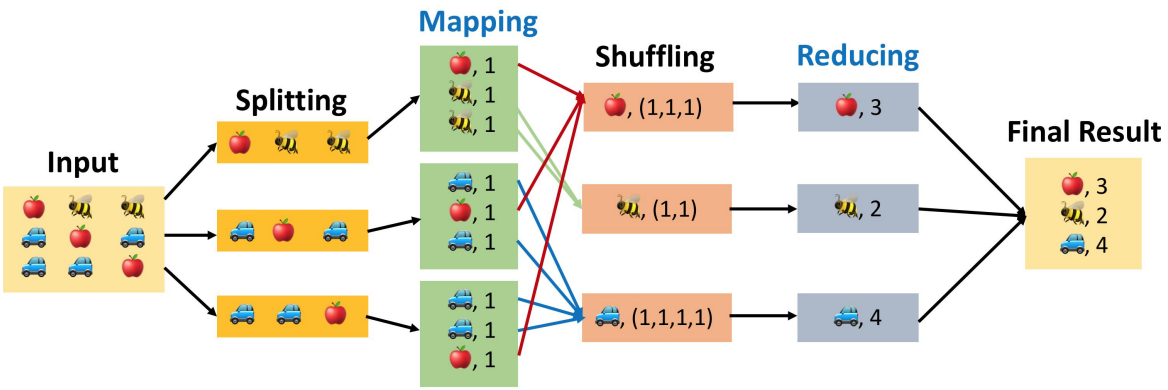
MapReduce





Terminologies

Job	Complete process from input to final output
Task	A part of the job executed on a slice of data
JobTracker	Master node to manage the jobs and resources
TaskTracker	Agent deployed in each machine to run MapReduce

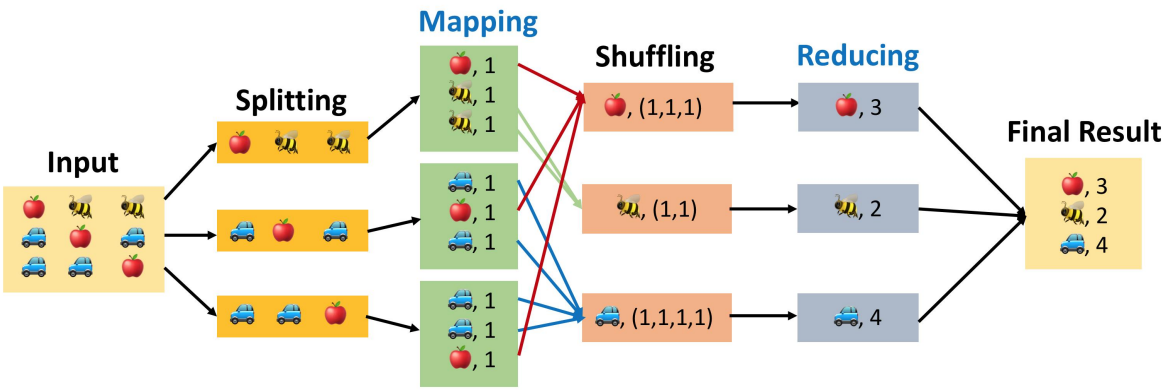


Task of Mapper

- The input is mapped into Key-Value (KV) pair.
- For example < , 1> is in the format of <key, value>.

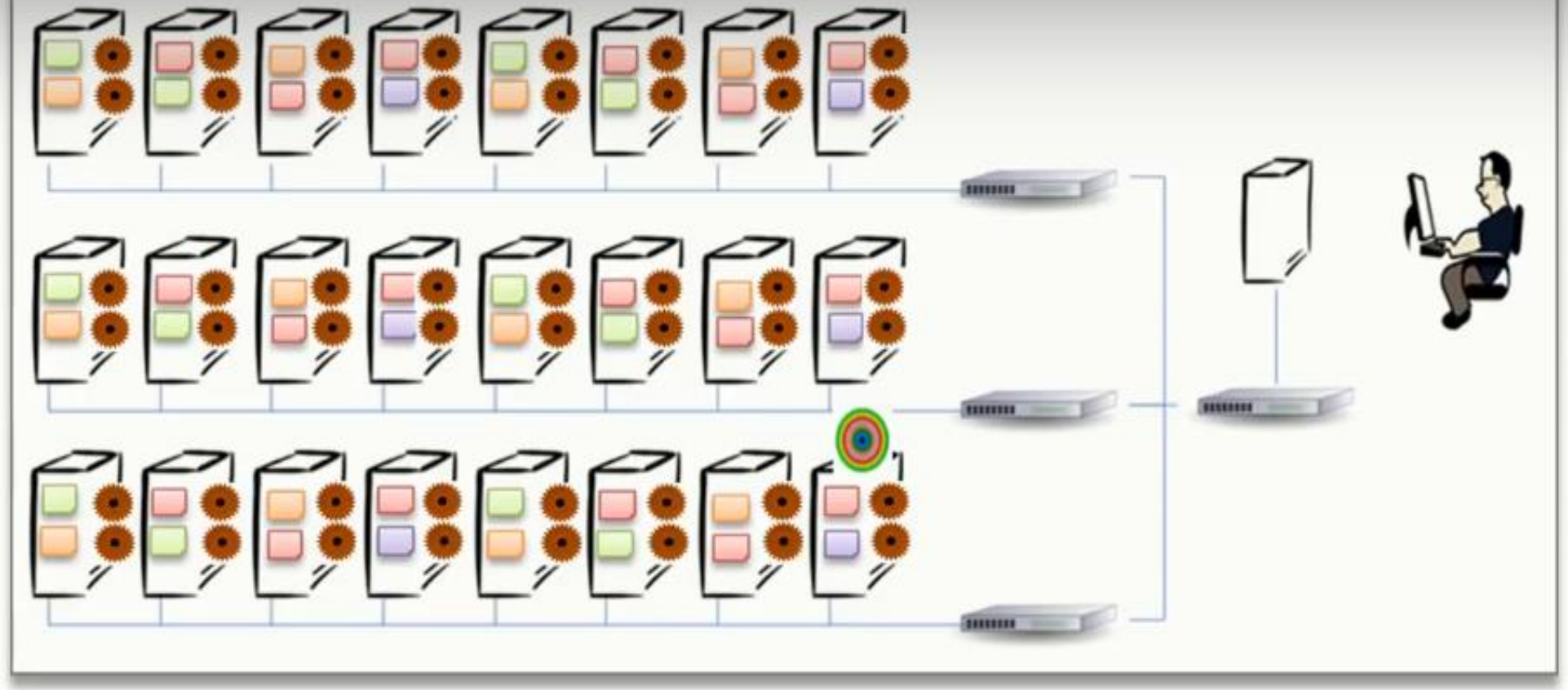
Intermediate Process

- Mapper output undergoes shuffle and sorting.
- The intermediate data would be stored in local file system without having replications in other nodes.



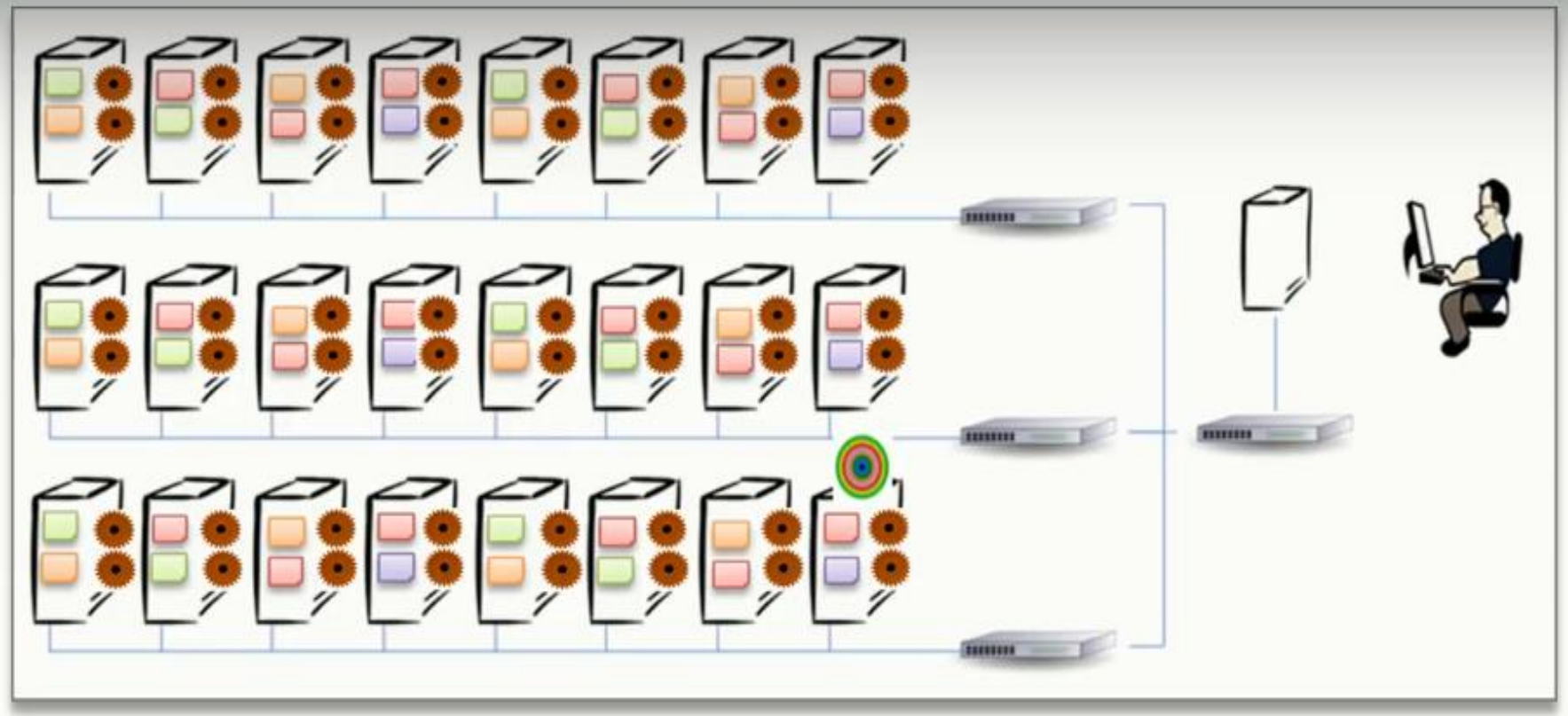
Task of Reducer

- Started only after all the mappers have completed their operations.
- Perform mathematical operations (such as aggregation/summation).
- User could define function to meet custom business logic.
- The output of Reducer is stored in HDFS.

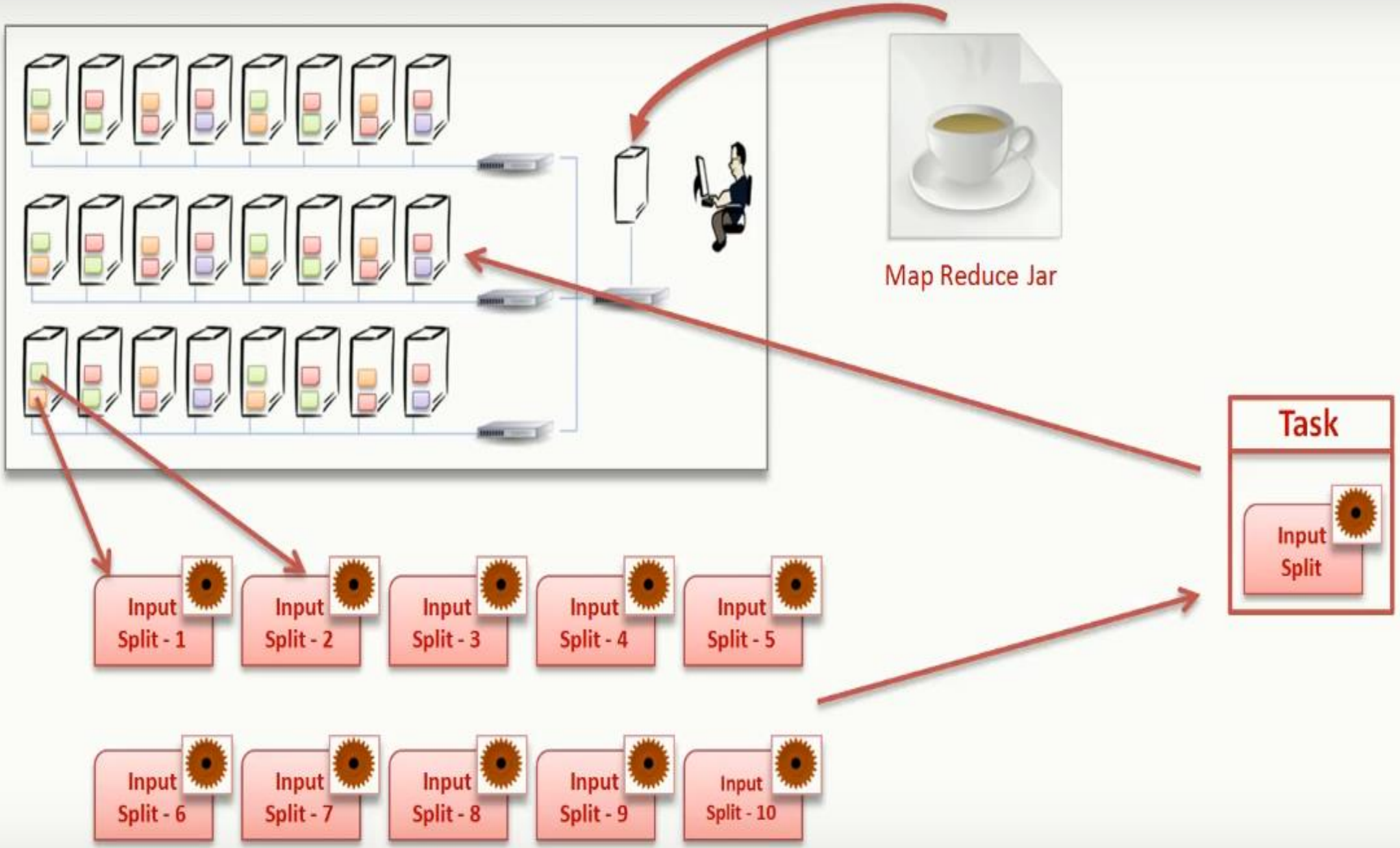


What is Map Reduce?

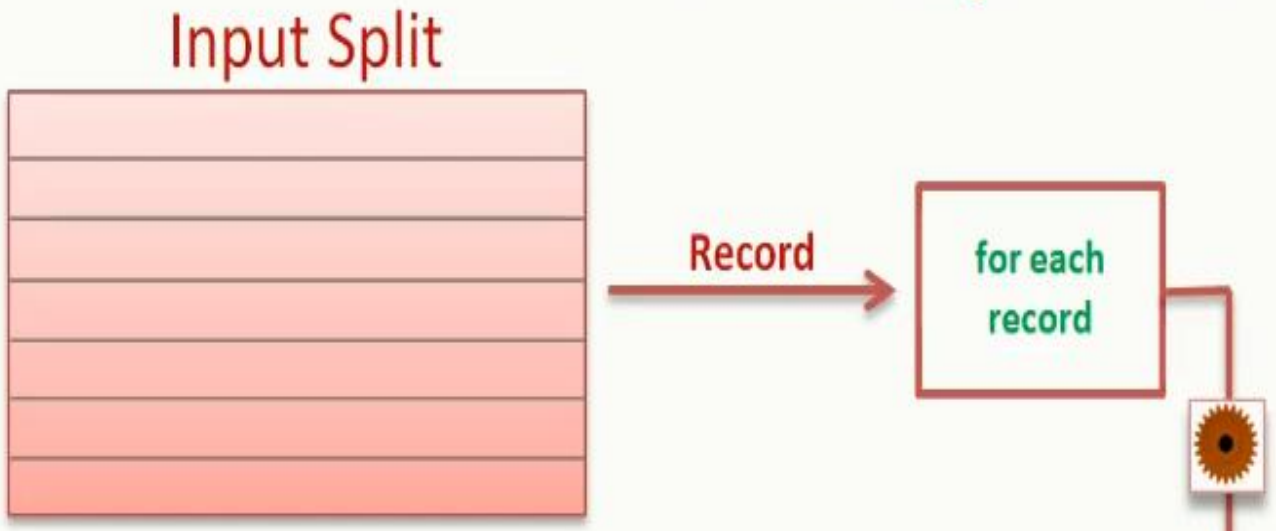
- Map Function
- Reduce Function



- How will a Map function read the lines from a block?
- Who will collect data from each Map computer and send it to the Reduce computer?


Map Reduce Jar
Task
Input Split
Input Split - 1
Input Split - 2
Input Split - 3
Input Split - 4
Input Split - 5
Input Split - 6
Input Split - 7
Input Split - 8
Input Split - 9
Input Split - 10

The Map Function

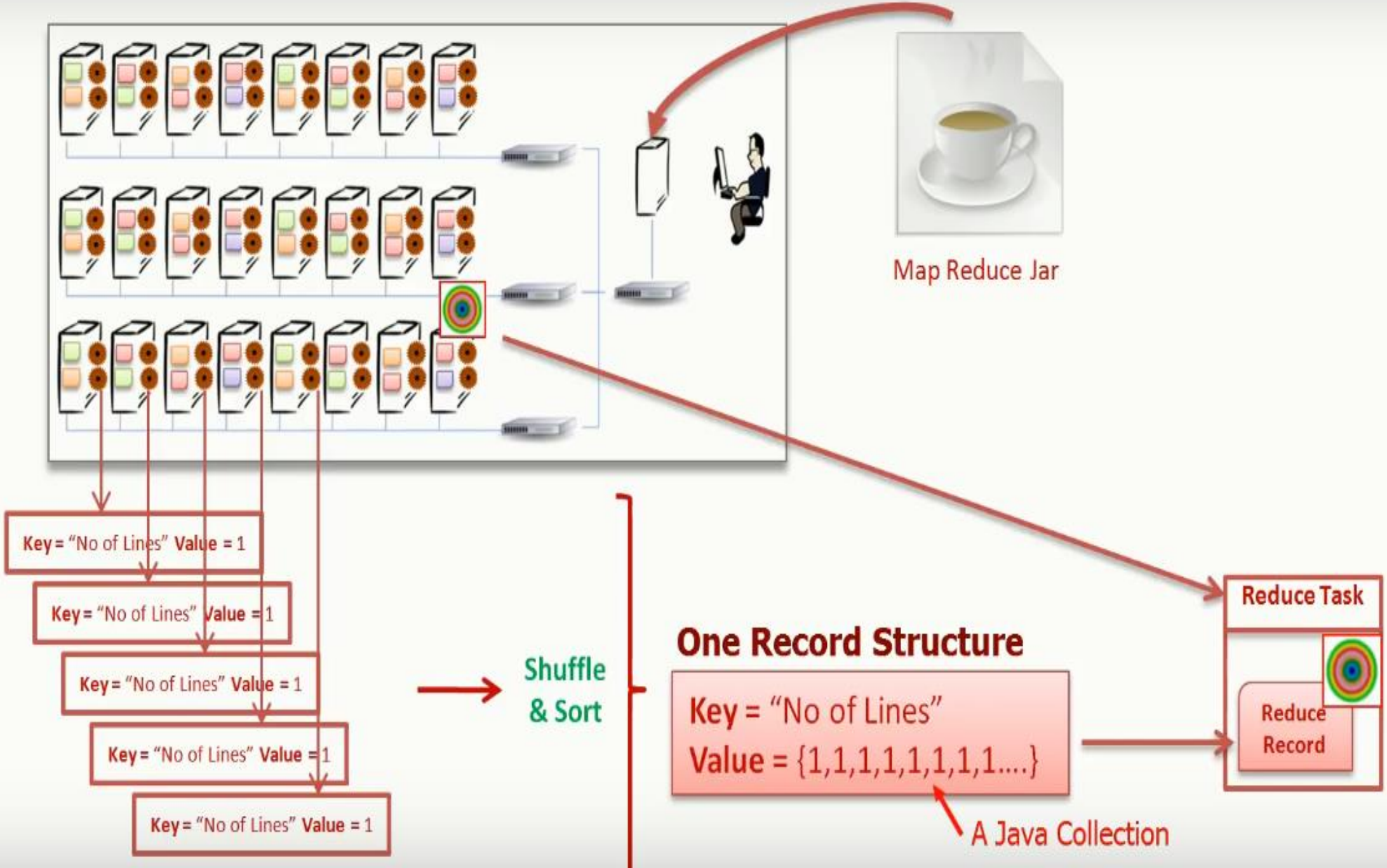



```
public void map(Object key, Text value, Context context
                ) throws IOException, InterruptedException {

  String line = value.toString();
  if (line != null && !line.isEmpty())
    context.write(new Text("No of Lines"), new IntWritable(1));
}
```

Key

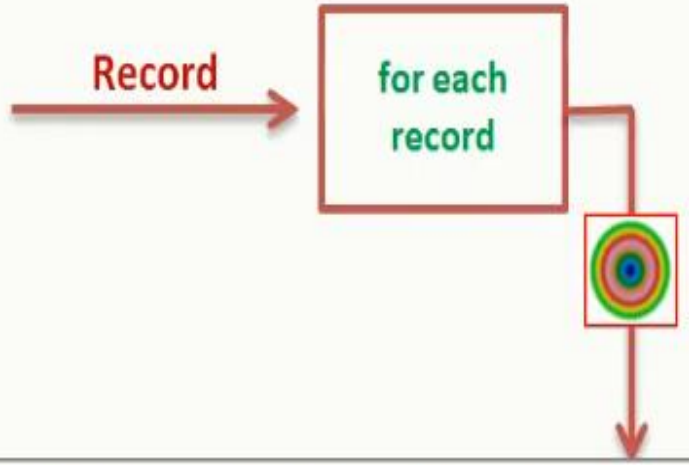
Value


Map Reduce Jar
Key = "No of Lines" Value = 1
Key = "No of Lines" Value = 1
Key = "No of Lines" Value = 1
Key = "No of Lines" Value = 1
Key = "No of Lines" Value = 1
Shuffle & Sort
One Record Structure
Key = "No of Lines"
Value = {1,1,1,1,1,1,1,1....}
A Java Collection
Reduce Task
Reduce Record

The Reduce Function

Reducer Records

Key = "No of Lines"
Value = {1,1,1,1,1,1,1,1....}




```
public void reduce(Text key, Iterable<IntWritable> values,
                   Context context
                   ) throws IOException, InterruptedException {
  int sum = 0;
  for (IntWritable val : values) {
    sum += val.get();
  }
  result.set(sum);
  context.write(key, result);
}
```

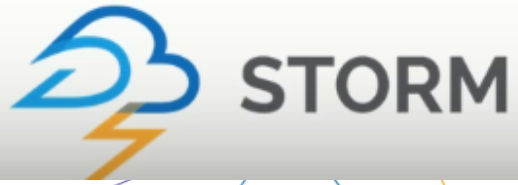
- The MR programming technique is all about modelling solutions in two phases. The Map function and a Reduce function.
- The framework will execute multiple instances of your Map function on different computers in your cluster.
- The framework will also attach one input split with each Map function instance.
- The framework will call your Map function in a loop and pass one record at a time from the split.
- The Map function is the place where you read data and do whatever you want to do with the data. You can simply count, compute some calculations, filter your rows or transform each row.
- The output of Map function is a Key/Value pair, and it goes back to MR framework.
- The framework will assemble all the Map outputs into one record per Key. So if you have written N different keys as output from you Map function, you get N records.
- The framework will pass each Record to the Reduce function one at a time in a loop.
- The reduce function is where you get all the data for a particular Key. So, you can aggregate the result at this stage.
- The output of the Reduce function is again a Key/Value pair, and the framework writes it to HDFS file.

The Center of Applied Data Science

Principles of Hadoop

MapReduce

Real-time Processing



What is YARN?

- It's a resource manager for Hadoop clusters.
- YARN is the Cluster Manager for Hadoop 2.x.
- YARN is a framework to provide computational resources for execution engines.

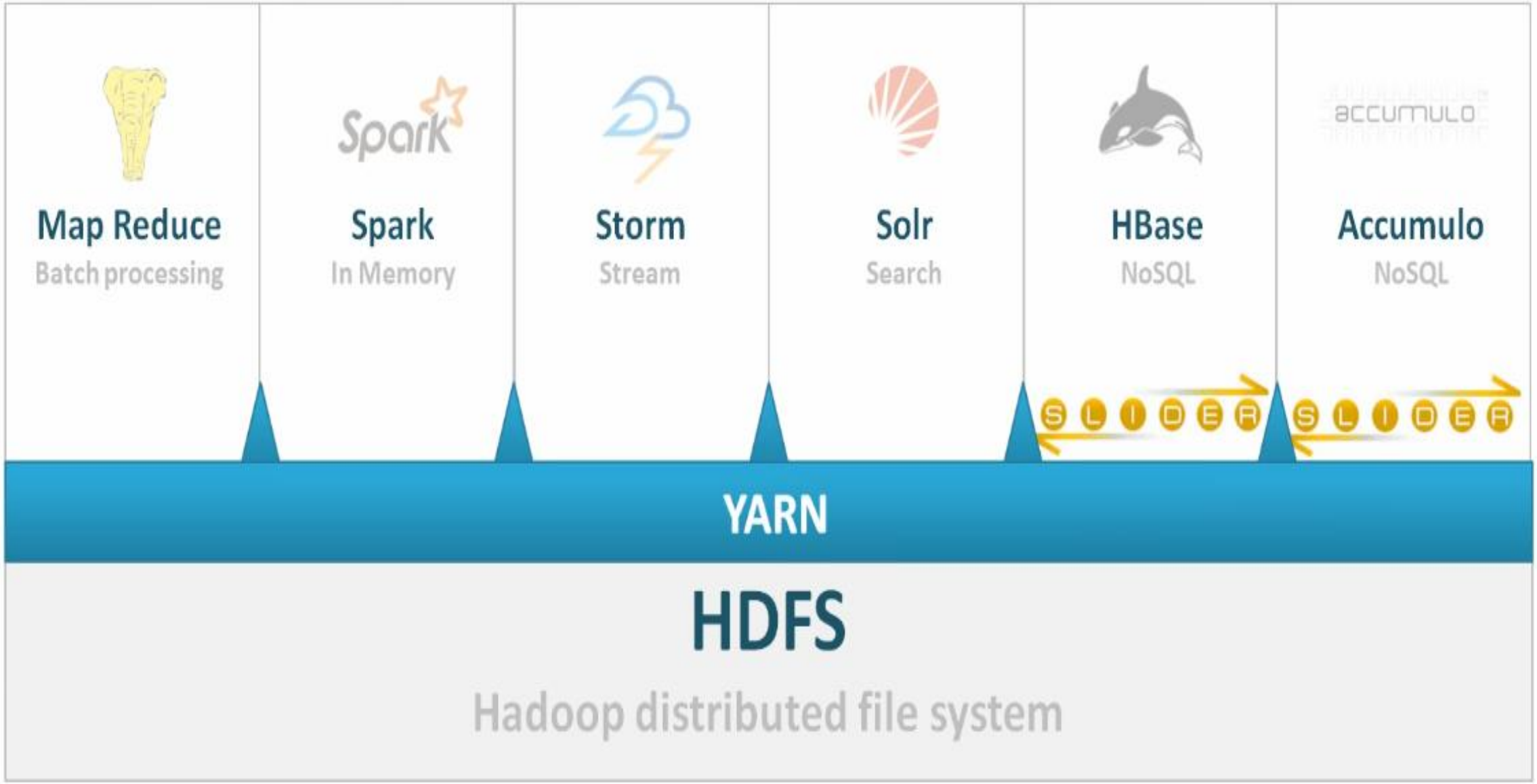
Map Reduce

Batch processing

YARN

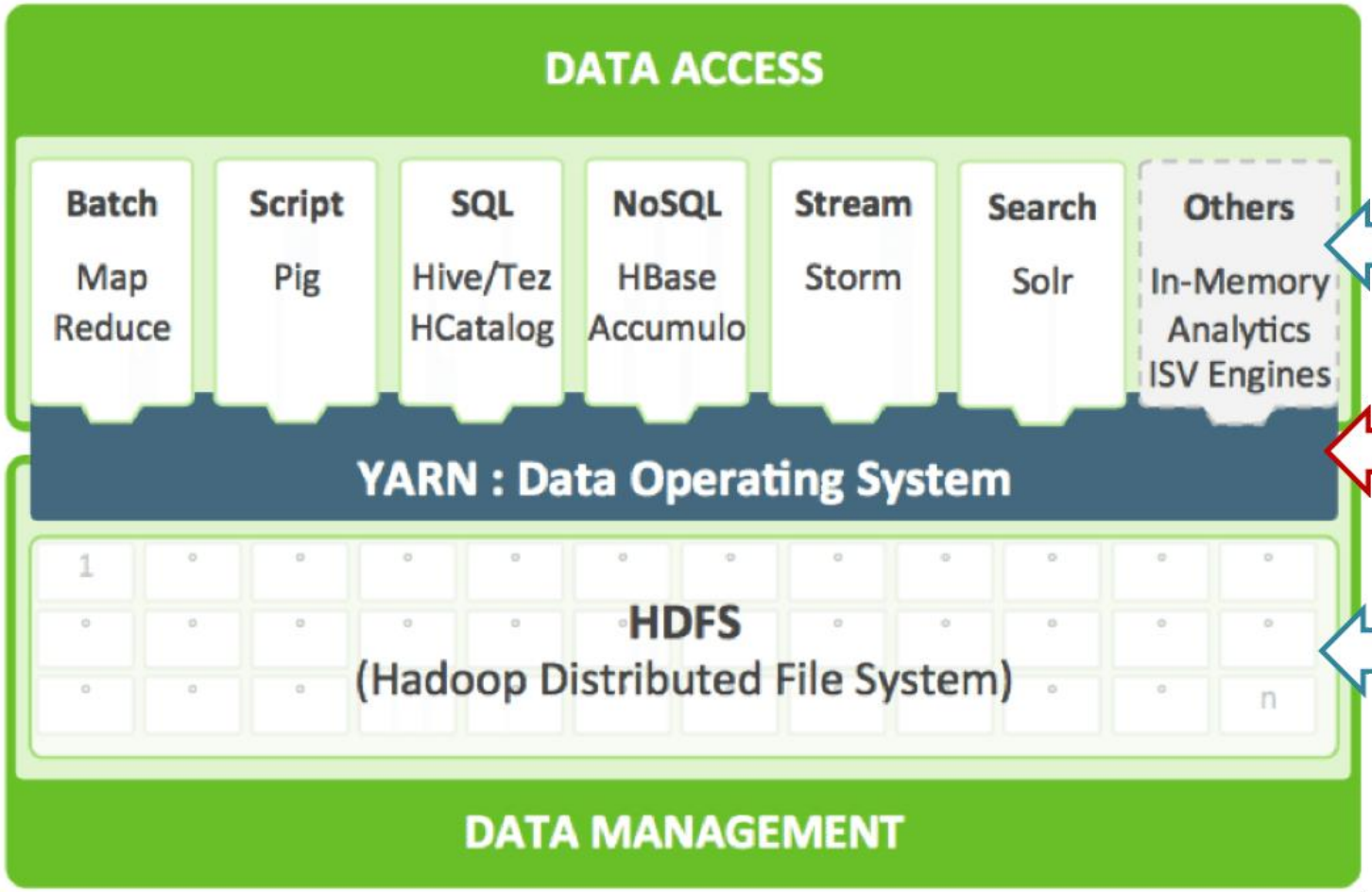
HDFS

Hadoop distributed file system


Map Reduce
Batch processing
Spark
In Memory
Storm
Stream
Solr
Search
HBase
NoSQL
Accumulo
NoSQL
SLIDER
SLIDER
YARN
HDFS
Hadoop distributed file system

Principles of Hadoop

YARN



More platforms than just MapReduce

YARN for better resource utilization

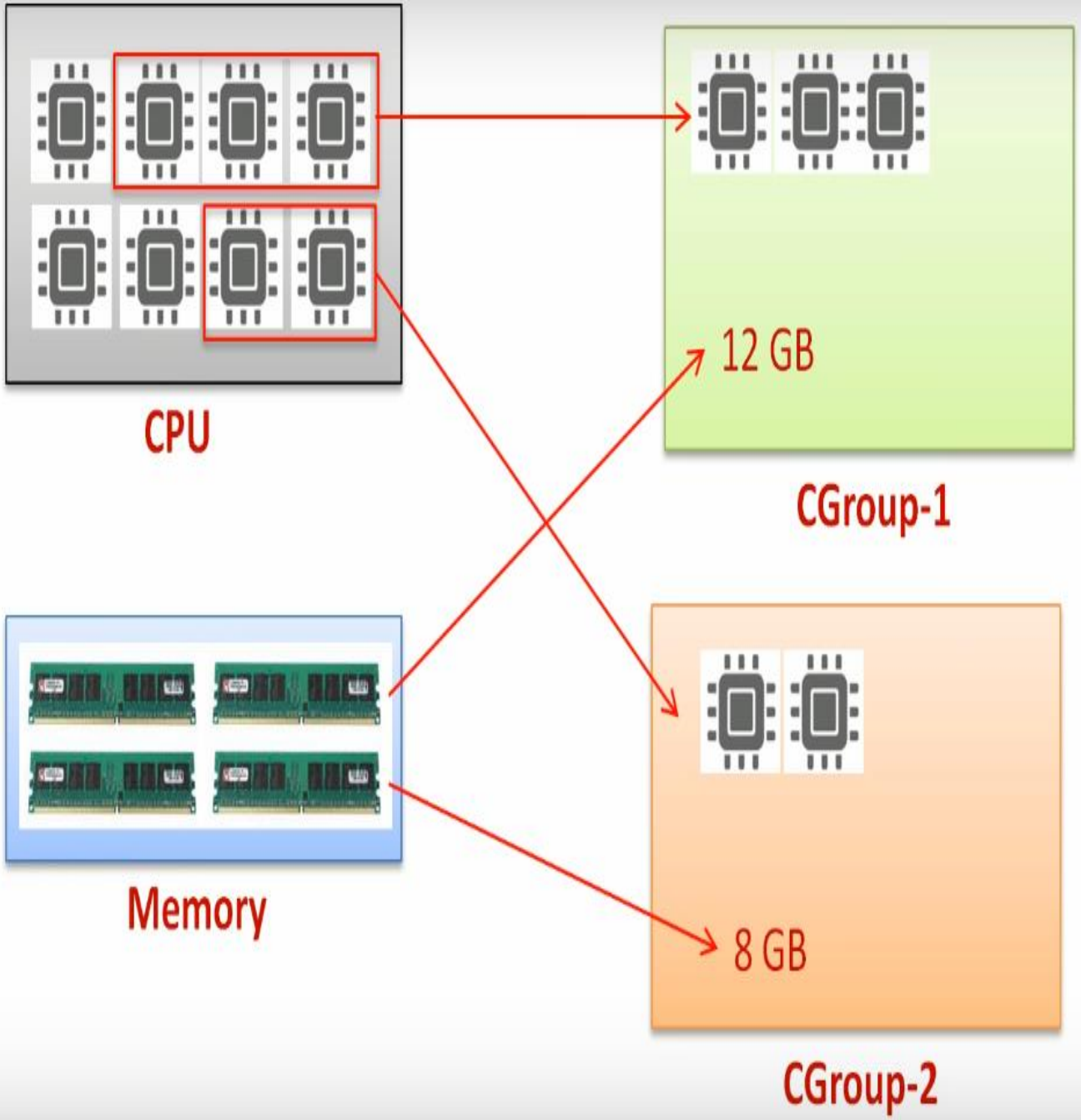
Common HDFS Foundation

Source: https://hortonworks.com/blog/apache-hadoop-yarn/

Components of YARN

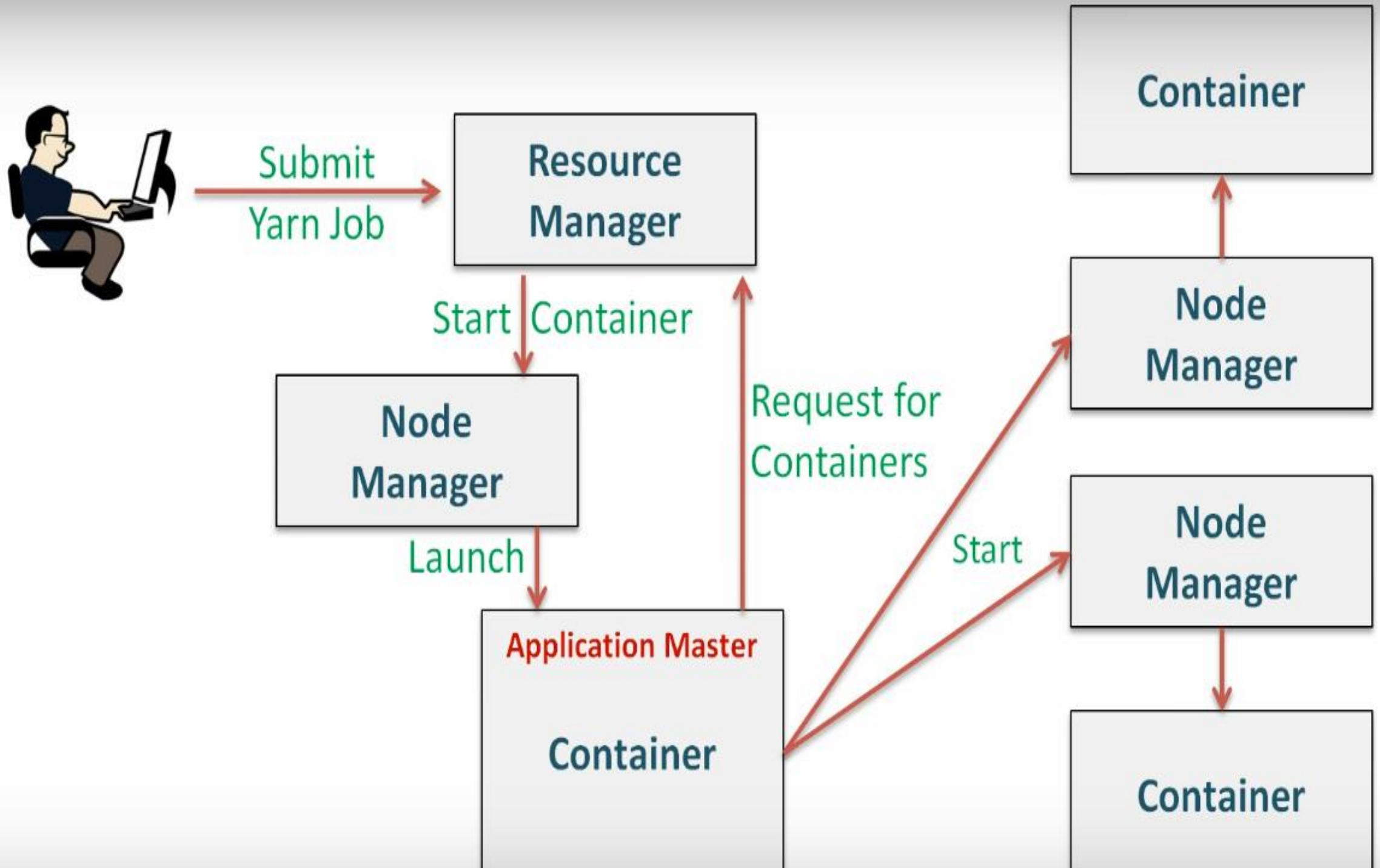
- Resource Manager (One per Cluster)
- Node Manager (One per data Node)

Hadoop Cluster
Name Node
Resource Manager
Stand By RM



CPU – 40%
Memory – 40%
Disk I/O – 20%
Network – 60%

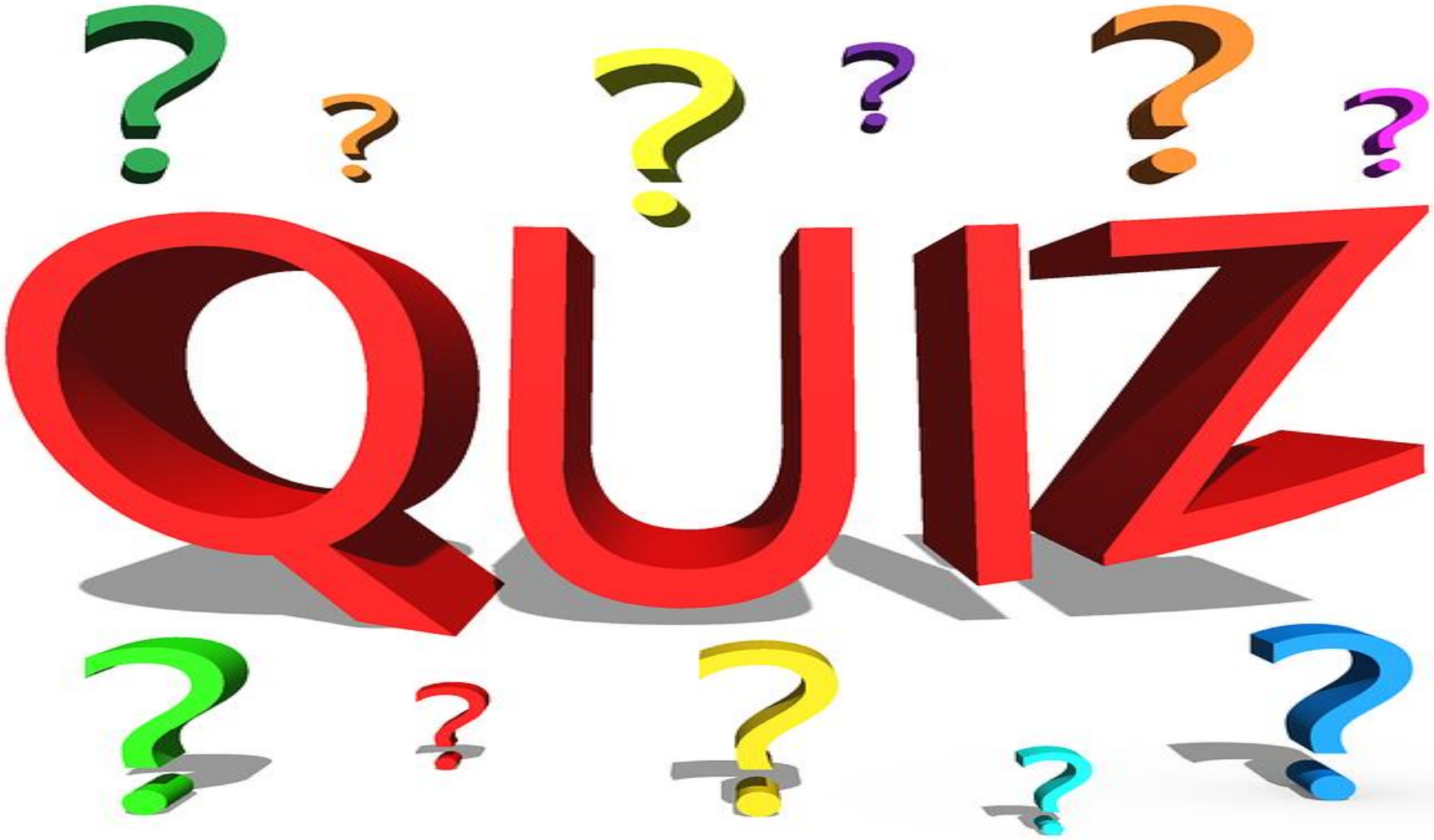
Container


Submit
Yarn Job
Resource
Manager
Start Container
Node
Manager
Launch
Application Master
Container
Request for
Containers
Start
Node
Manager
Node
Manager
Container
Container

The Center of Applied Data Science

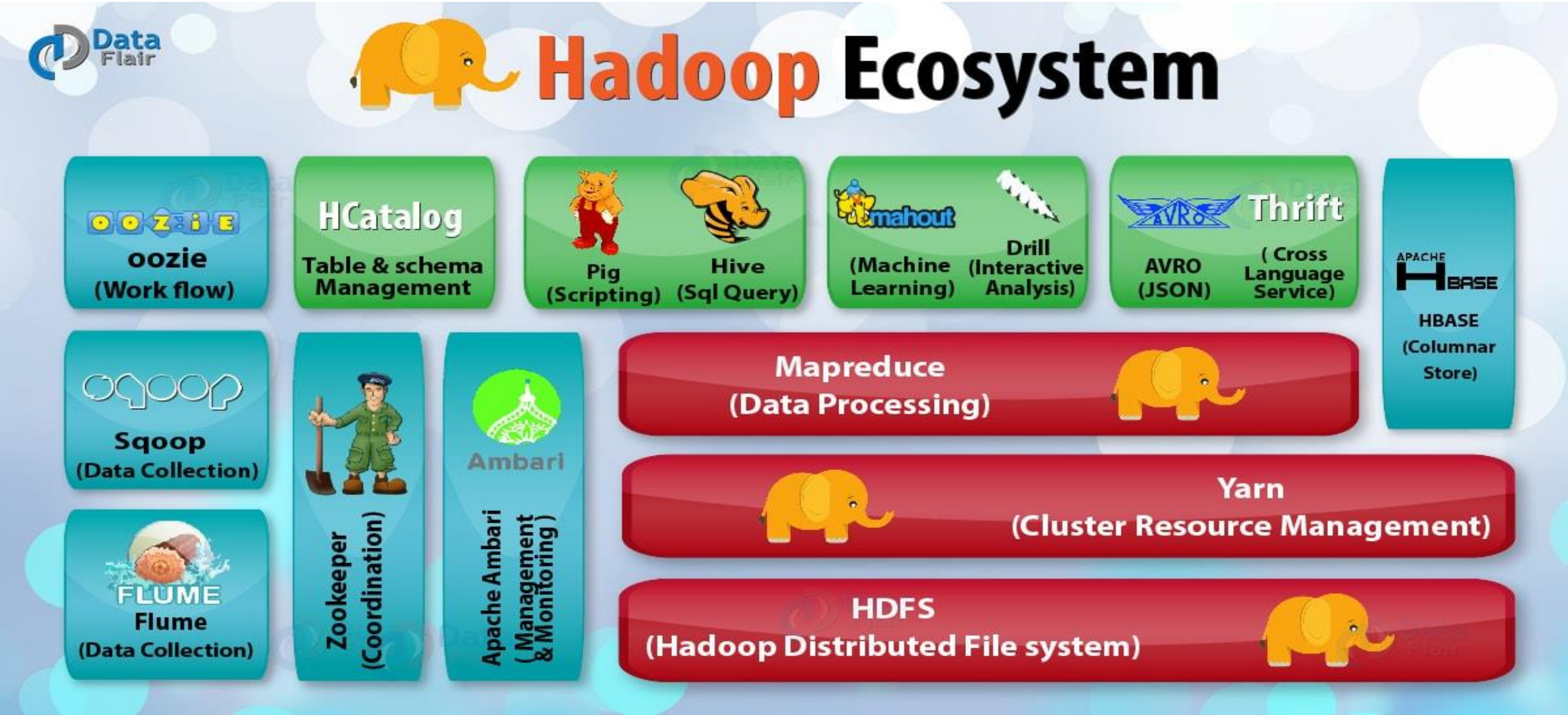
QUIZ 2

https://forms.gle/1hkYgwviiaHPVwoG8



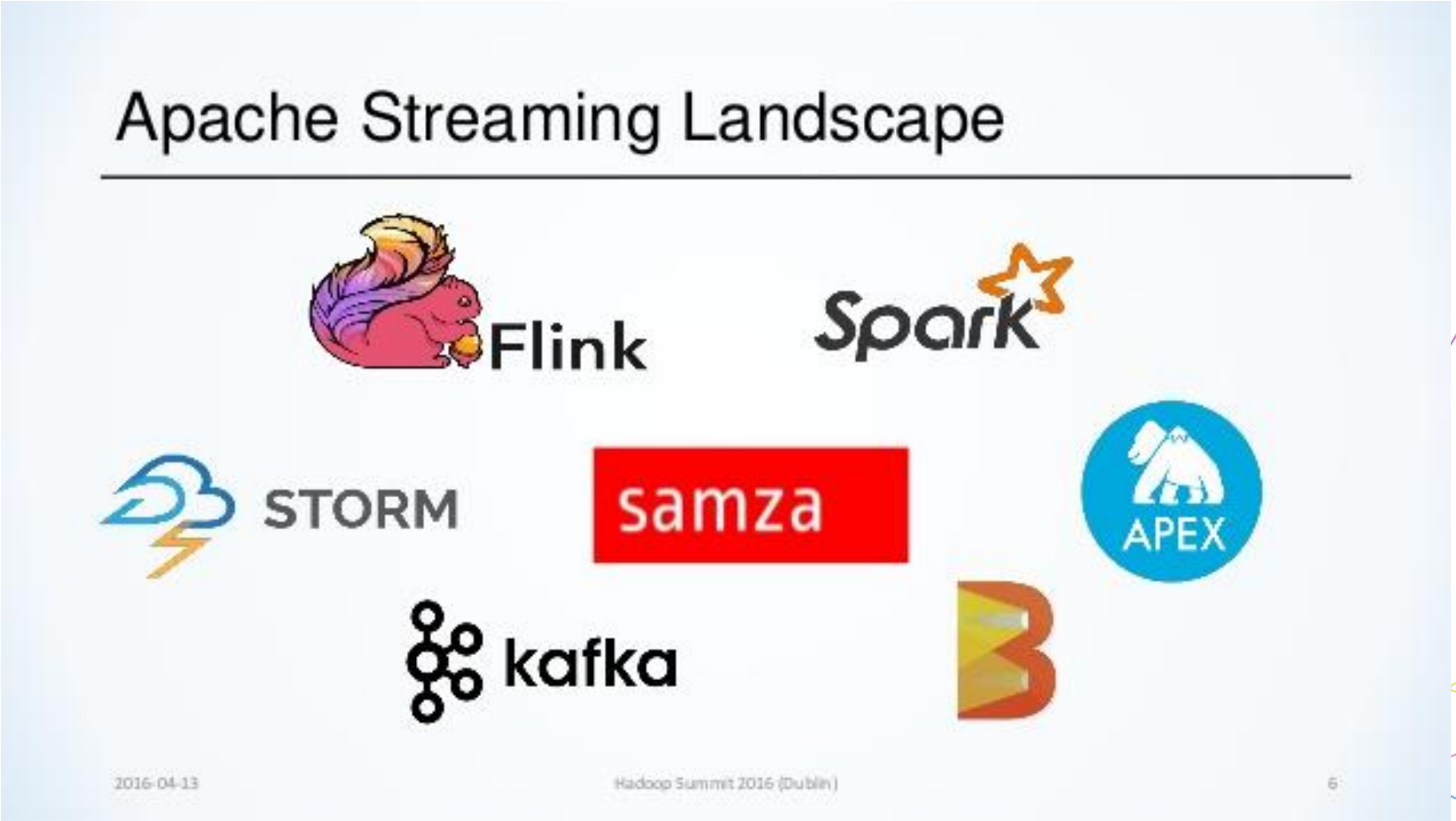
The Center of Applied Data Science

Hadoop Ecosystem



The Center of Applied Data Science

Stream Processing in Hadoop Ecosystem



Overview

- Open source cluster computing framework that is suitable for large-scale data processing.
- Provides machine learning projects, batch processing, near real-time processing, and graph analysis.

Application

- Suitable for large-scale Data Science use cases.
- Able to run on Hadoop, Amazon AWS cloud, and different databases such as Cassandra, Amazon Dynamo DB etc.

APACHE
Spark
Spark is 10 to 100 times faster than Hadoop's Map Reduce.

Secure | https://spark.apache.org

Lightning-fast cluster computing

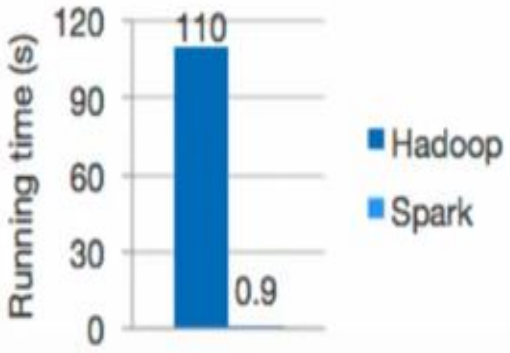
Download | Libraries | Documentation | Examples | Community | Developers | Apache Software Foundation

Apache Spark™ is a fast and general engine for large-scale data processing.

Speed

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 acyclic data flow and in-memory computing.



Logistic regression in Hadoop and Spark

Latest News

Spark 2.1.2 released (Oct 09, 2017)

Spark Summit Europe (October 24-26th, 2017, Dublin, Ireland) agenda posted (Aug 28, 2017)

Spark 2.2.0 released (Jul 11, 2017)

Spark 2.1.1 released (May 02, 2017)

Archive

Download Spark

What is Apache Spark™?

Apache Spark is a powerful open source processing engine built around speed, ease of use, and sophisticated analytics. It was originally developed at UC Berkeley in 2009.

The largest open source project in data processing.

Since its release, Apache Spark has seen rapid adoption by enterprises across a wide range of industries. Internet powerhouses such as Netflix, Yahoo, and eBay have deployed Spark at massive scale, collectively processing multiple petabytes of data on clusters of over 8,000 nodes. It has quickly become the largest open source community in big data, with over 1000 contributors from 250+ organizations.

The team that created Apache Spark founded Databricks in 2013.

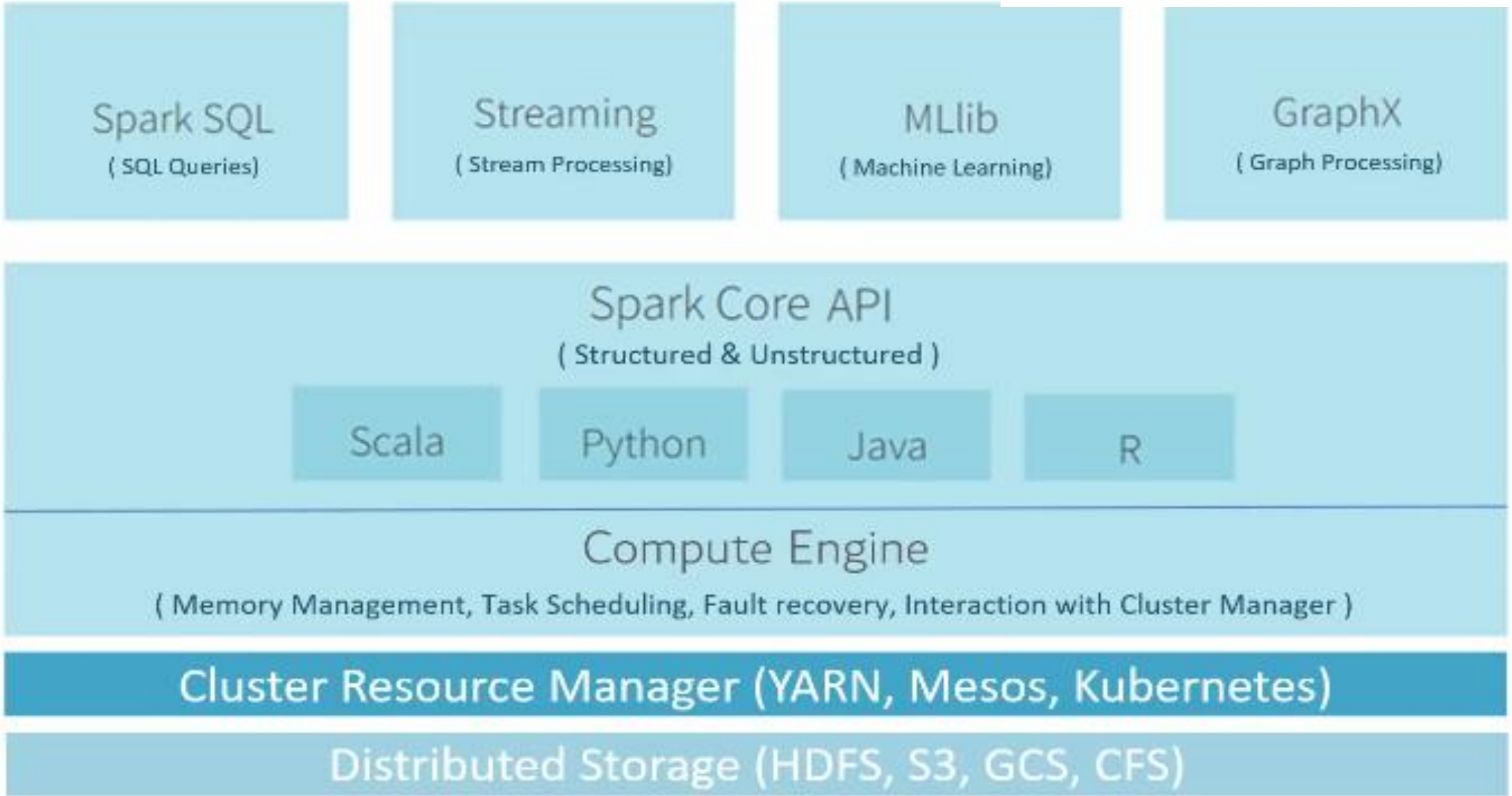
Apache Spark is 100% open source, hosted at the vendor-independent Apache Software Foundation. At Databricks, we are fully committed to maintaining this open development model. Together with the Spark community, Databricks continues to contribute heavily to the Apache Spark project, through both development and community evangelism.

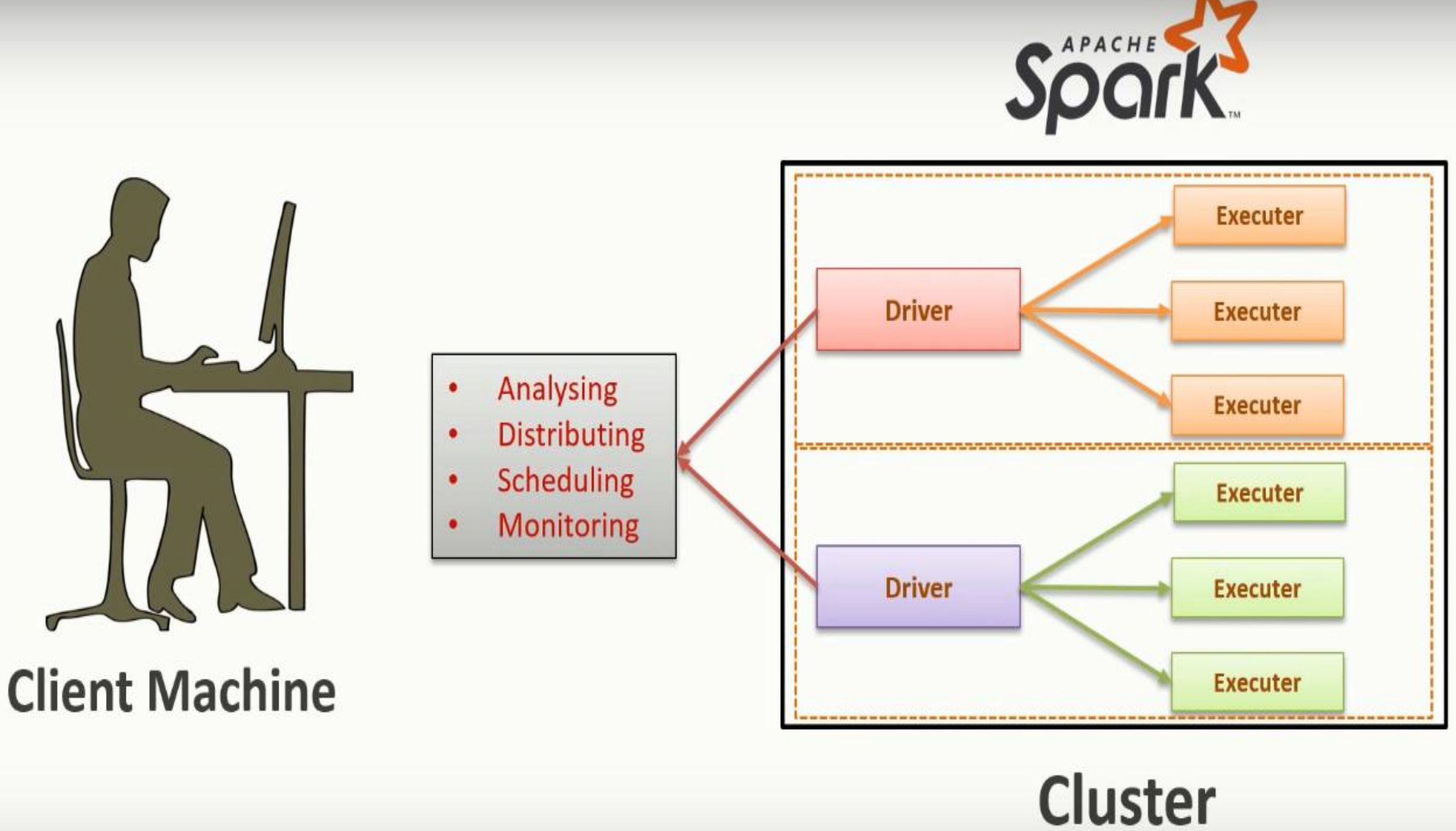
Apache Spark is a fast and general purpose **engine for large-scale data processing**. Under the hood, it works on a **cluster of computers**.

1. **A cluster computing engine.**
2. **A set of libraries, APIs, and DSLs.**

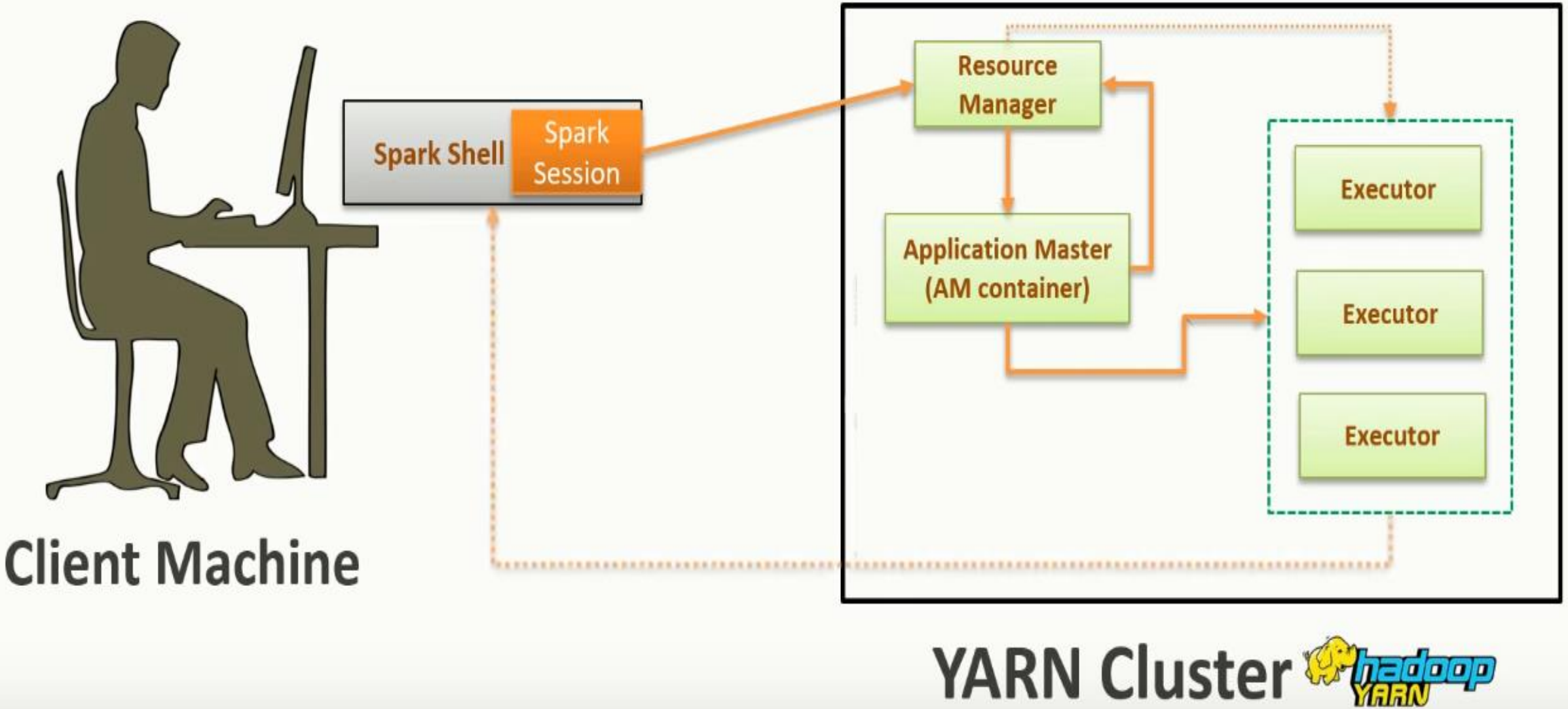
Why Spark?

1. Abstract the Parallel Programming
2. Unified Platform
3. Ease of Use

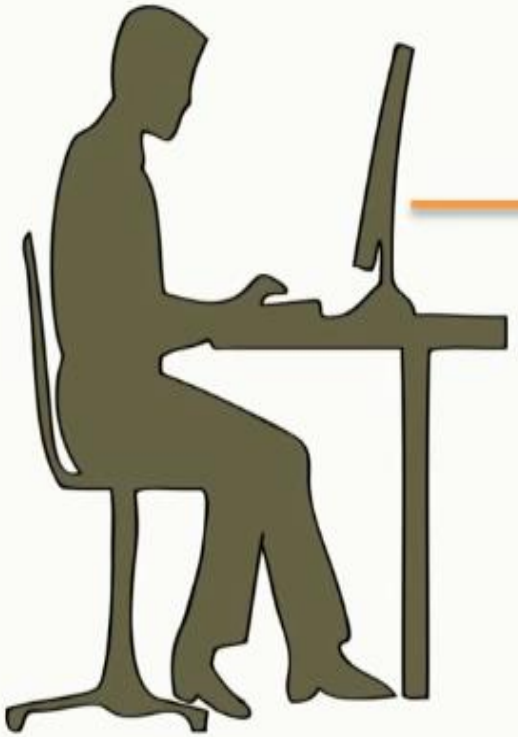



APACHE Spark
Client Machine
Analysing
Distributing
Scheduling
Monitoring
Driver
Executer
Executer
Executer
Driver
Executer
Executer
Executer
Cluster

Client Mode


Spark Shell
Spark Session
Resource Manager
Application Master (AM container)
Executor
Executor
Executor
Client Machine
YARN Cluster

Cluster Mode



Client Machine

Spark-submit

Resource Manager

Application Master (AM container)

Spark Driver

Executor

Executor

Executor

YARN Cluster

Overview

- A real-time computational system for accepting high volume data coming in high velocity, possibly from various sources.
- Easy to implement and can be integrated with any programming language.

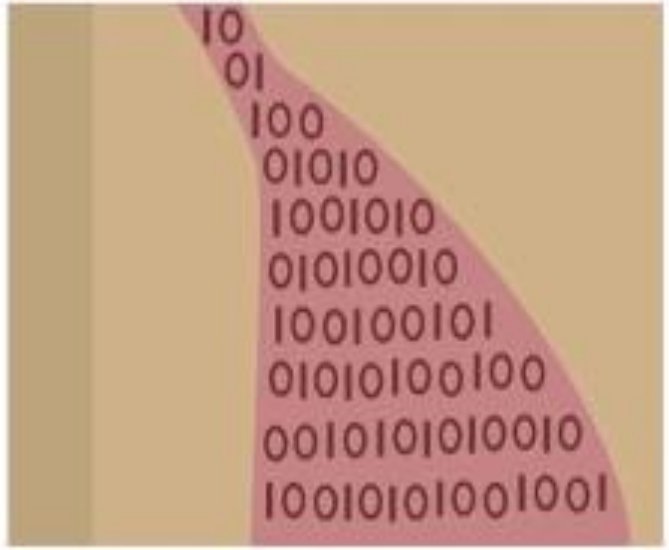
Application

- Suitable for applications that primarily focused on stream processing and CEP-style processing.
- Apache Storm has the advantage of broader language support over Apache Spark.

What is Storm?

Storm is a distributed, reliable, fault-tolerant system for processing streams of data.

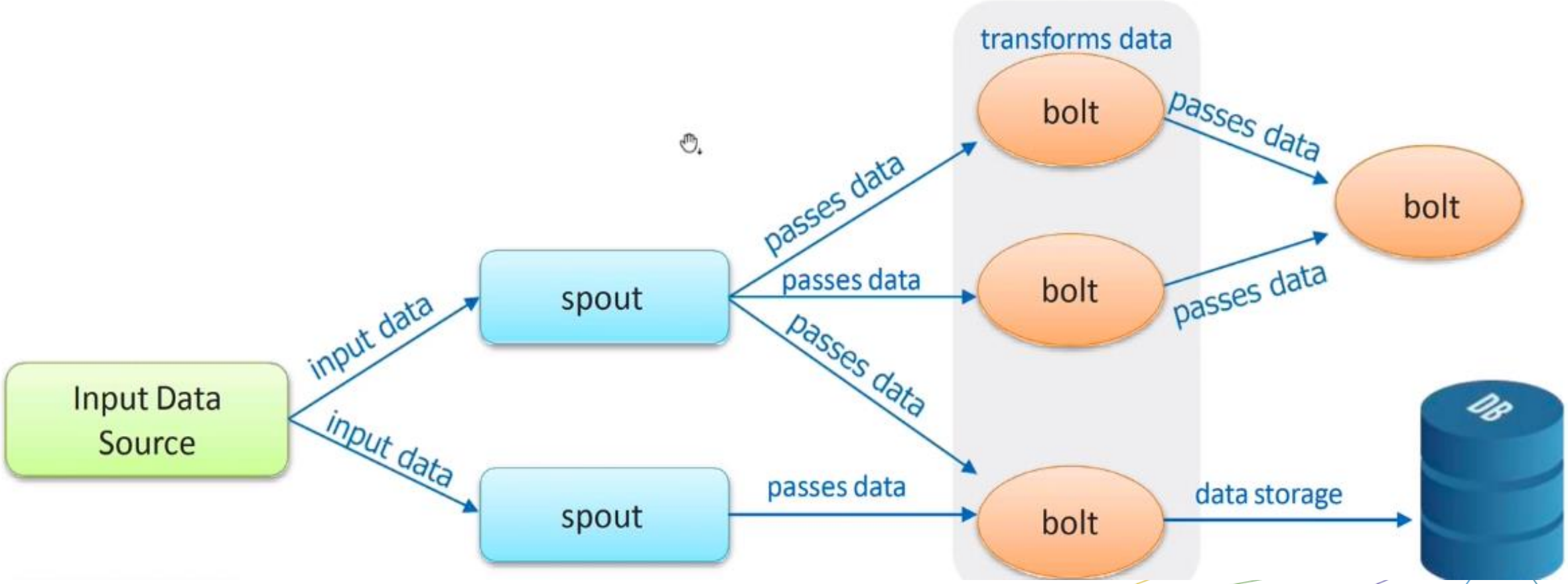
processing



Streams of Data

What is Storm?

- The work is delegated to different types of components that are each responsible for a simple specific processing task.
- The input stream of a Storm cluster is handled by a component called a spout.
- The spout passes the data to a component called a bolt, which transforms it in some way.
- A bolt either persists the data in some sort of storage, or passes it to some other bolt.



Storm Components

A Storm cluster has 3 sets of nodes

1. Nimbus node
2. Zookeeper nodes
3. Supervisor nodes

→ Nimbus node (master node, similar to the Hadoop JobTracker):

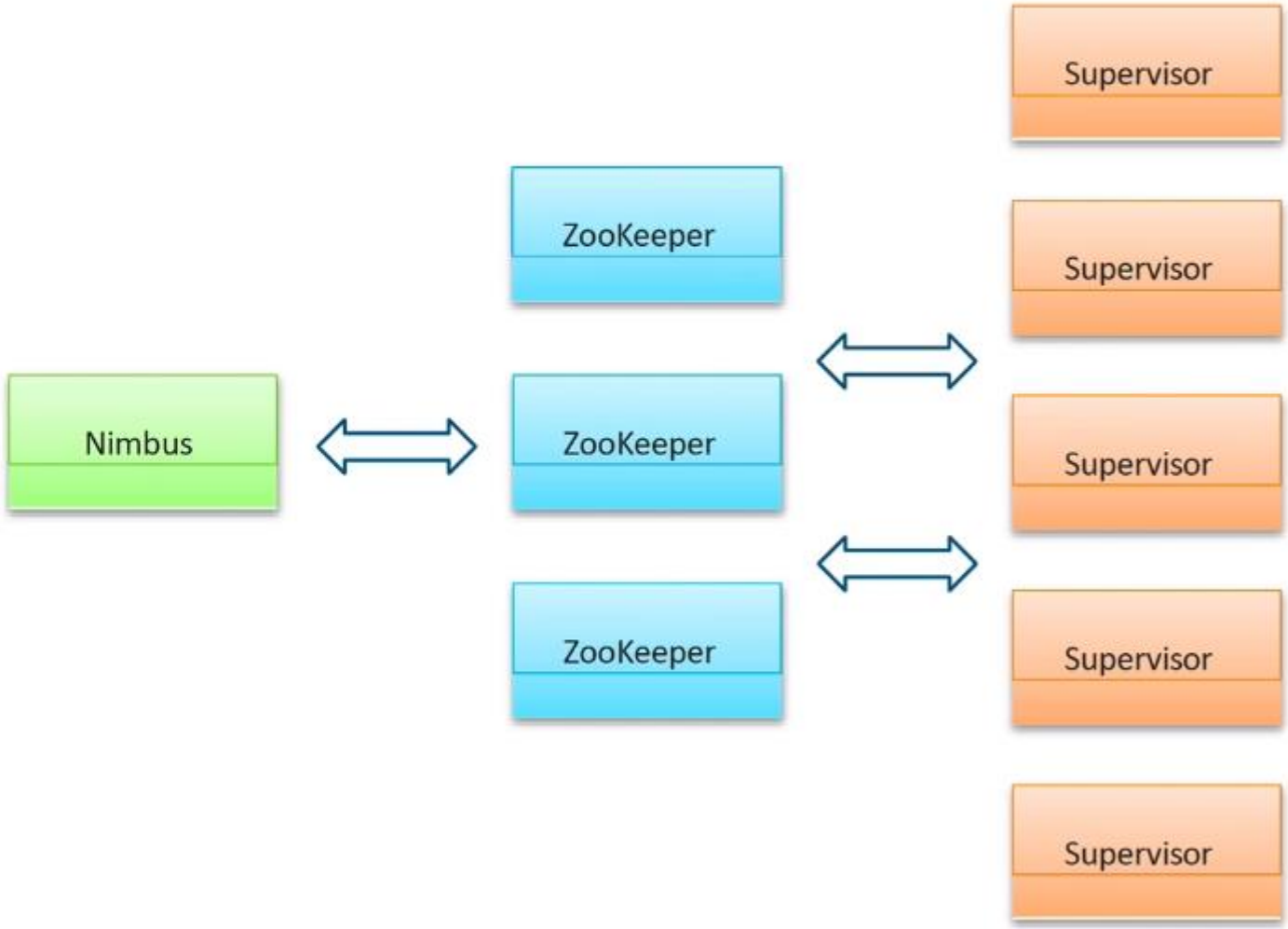
» Uploads computations for execution
» Distributes code across the cluster
» Launches workers across the cluster
» Monitors computation and reallocates workers as needed

→ ZooKeeper nodes:

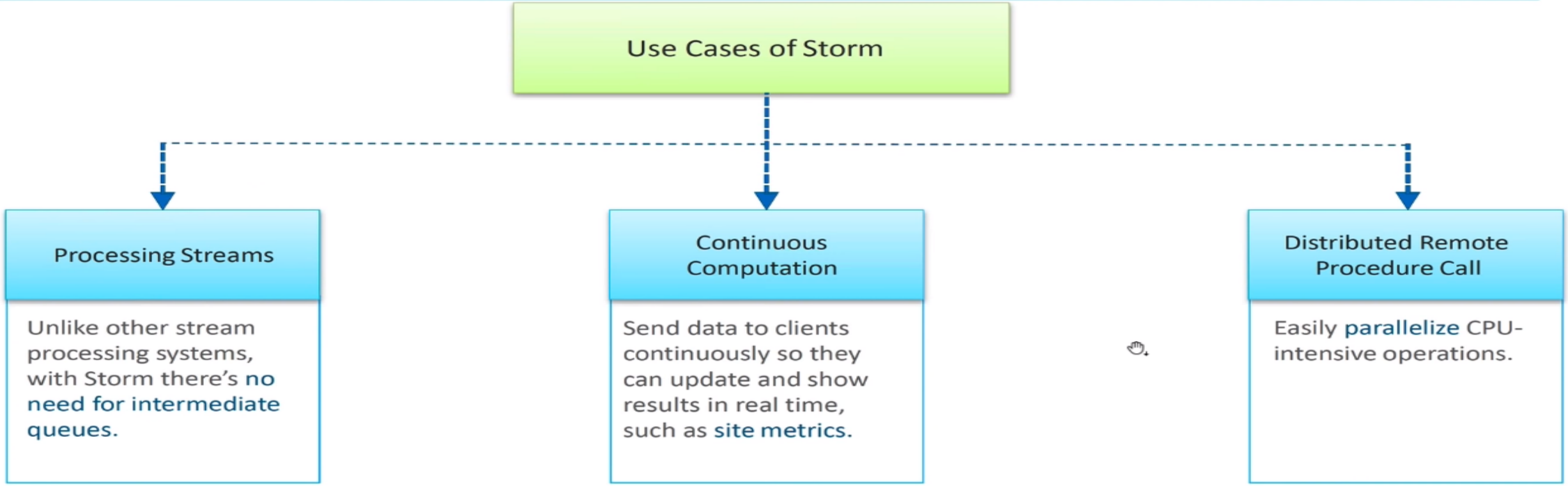
» Coordinates the Storm cluster

→ Supervisor nodes:

» Communicates with Nimbus through Zookeeper, starts and stops workers according to signals from Nimbus



Use Cases of Storm



- Financial Services
  - » Securities Fraud
  - » Compliance Violations
  - » Order Routing
  - » Pricing
- Telecom
  - » Security Breaches
  - » Network Outages
- Retail
  - » Shrinkage
  - » Stock outs
  - » Offers
  - » Pricing
- Web
  - » Application Failure
  - » Operational Issues

Overview

- A tool for data streaming and processing applications and it is excels for stateful streaming applications at any scale.

- Provides real-time processing, machine learning projects, batch processing, and graph analysis.

Application

- Apache Flink is able to run on third-party data sources such as Amazon Kinesis Streams, Elasticsearch, Cassandra, and Twitter Streaming API.

- The introduction of ACID into data Artisans platform reinforces position of Apache Flink as the integration hub for the real-time large financial and eCommerce organizations.

About Flink

Open Source

- Started in 2009 by Berlin based database research group
- In December 2014, Flink was accepted as an Apache top-level project

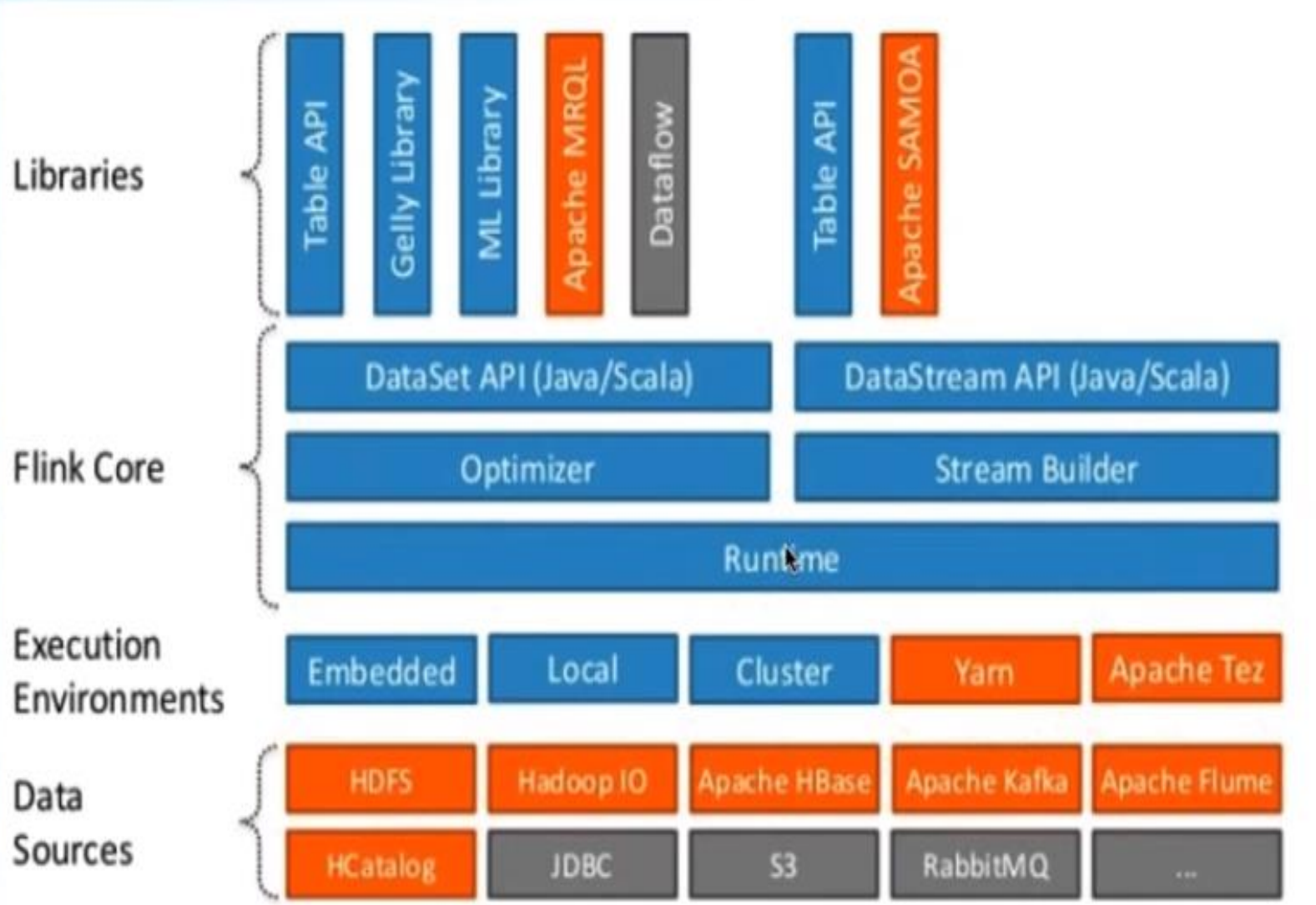
Fast + Reliable

- Fast, general purpose distributed data processing system
- Upto 100x faster than Hadoop
- Low latency and high throughput

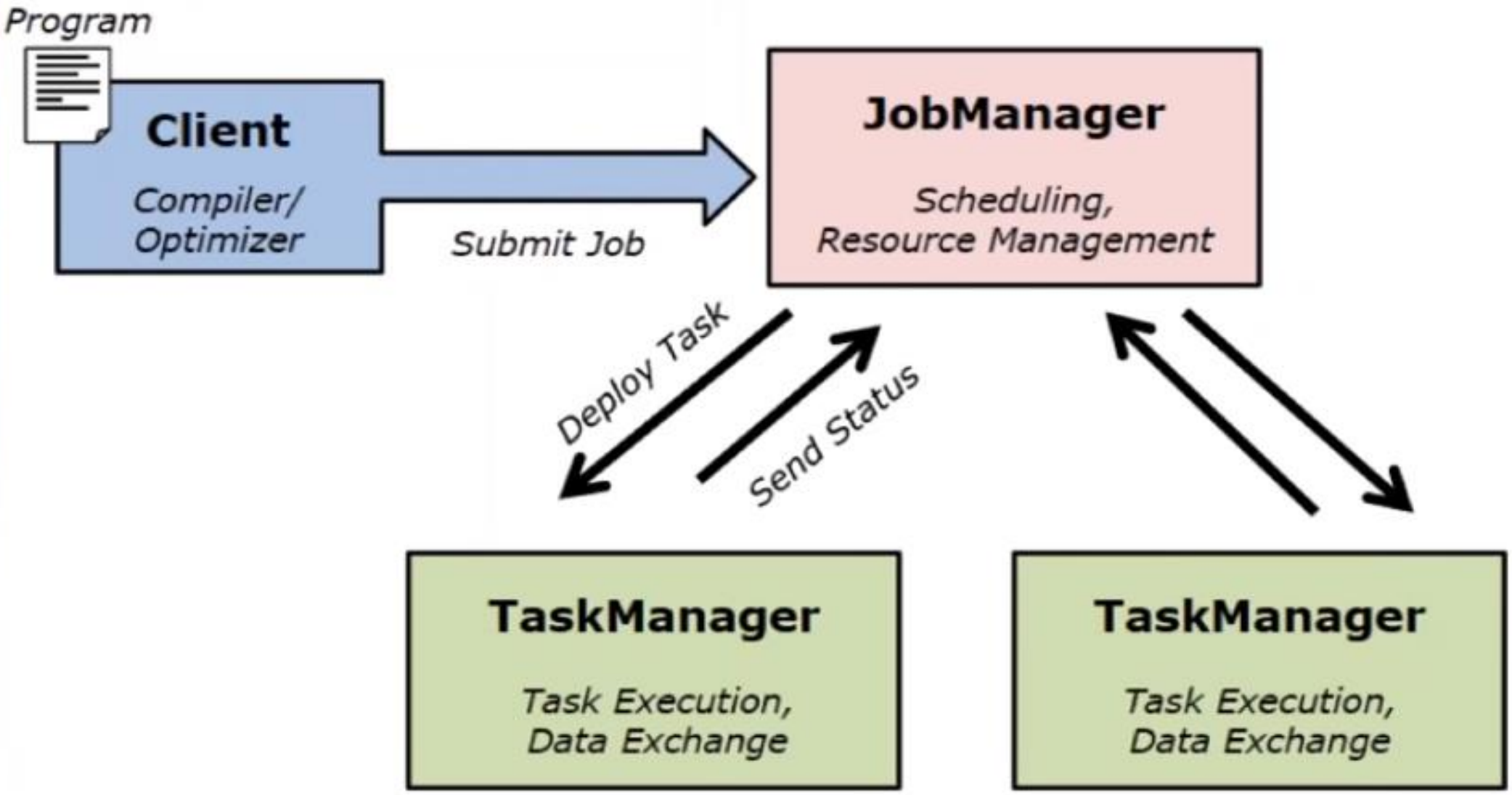
Ready to Use

- Programming APIs for Java and Scala
- Tested on large clusters

Apache Flink Ecosystem



Flink Architecture



Spark	Apache Flink
Spark is not a true real time processing. It is Near to real time processing framework.	Flink is True processing framework.
At heart Spark is a Batch processing framework.	At heart Flink is a Stream processing framework.
Spark's streaming computation model is based on Microbatching.	Flink's Streaming model is based on Windowing and Checkpointing.
Spark is implemented in Scala.	Flink is implemented in Java.
Spark does not have an efficient memory manager.	Flink has it's own efficient automatic memory manager.
Frequently gets Out of Memory.	Rarely gets Out of Memory.
Uses DAG as its execution engine.	Flink uses controlled cyclic Dependency graph as its execution engine.

Flink is an excellent alternative to Spark:

- High Throughput/Low Latency
- Robust memory management
- Built-in cyclic processing for machine learning
- Batch Processing
- **Stream Processing!!!**
- SQL Processing
- Stateful Streaming and Fault Tolerance

Overview

- Designed to build a central data backbone for a large organization with a single cluster for processing ingests data in real-time.

- A single Kafka broker can handle hundreds of megabytes of reads and writes per second from thousands of clients.

Application

- Suitable for publish-subscribe messaging. Users can publish and subscribe to information as and when they occur.

- Suitable to manage the variety of use cases commonly required for a Data Lake.

- Able to render streaming data through a combination of Apache Hbase, Apache Storm, and Apache Spark systems.

Definition:- Kafka is a distributed streaming platform.

What it can do?

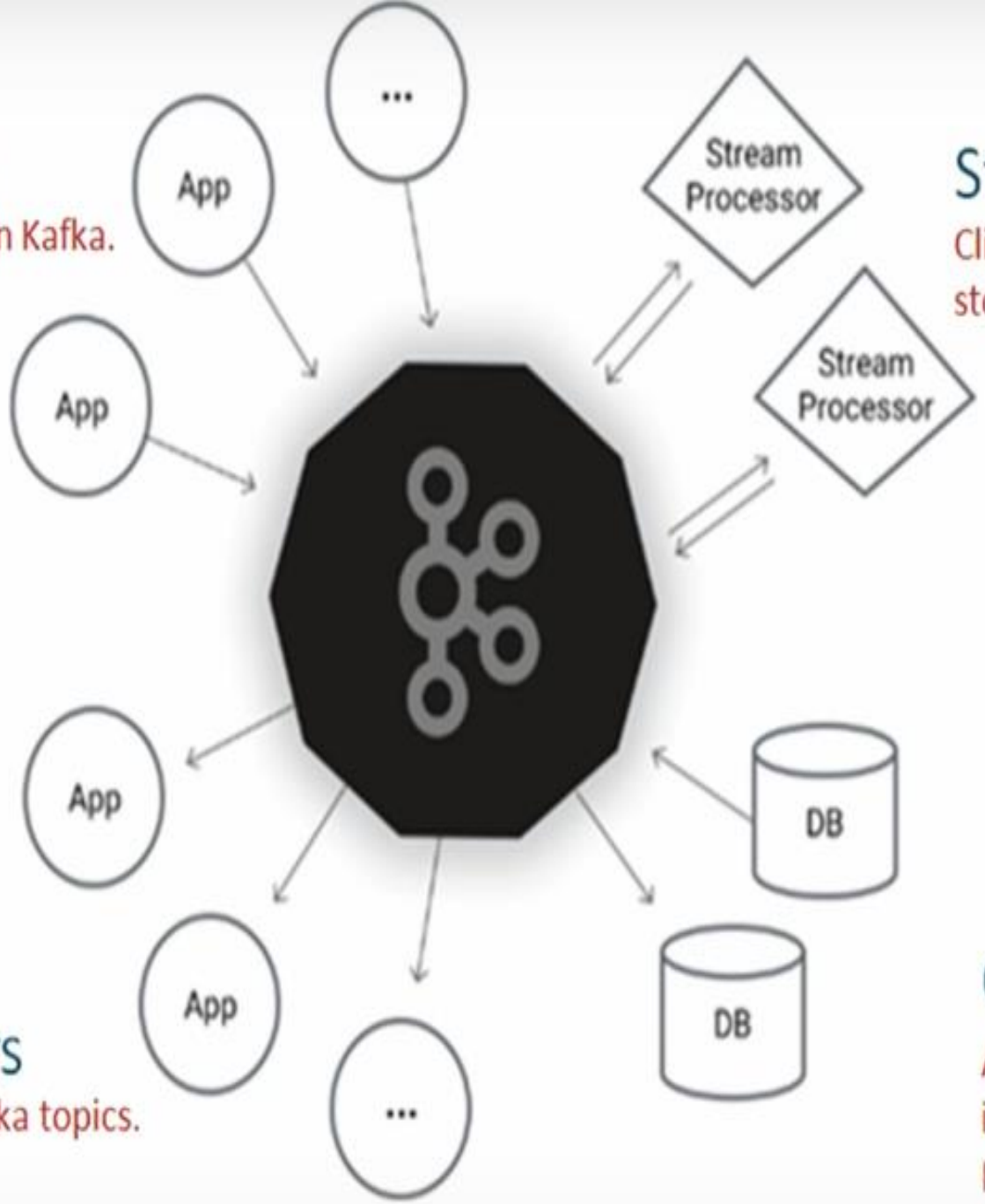
- It can be used as an enterprise messaging system.
- It can be used for stream processing.
- It also provides connectors to import and export bulk data from databases and other systems.

Producers

Publishes records to the topics in Kafka.

Streams

Client library for processing data stored in Kafka.



Consumers

Pulls records stored in Kafka topics.

Connectors

A framework to build connectors for in/out movement of data between Kafka and other systems.

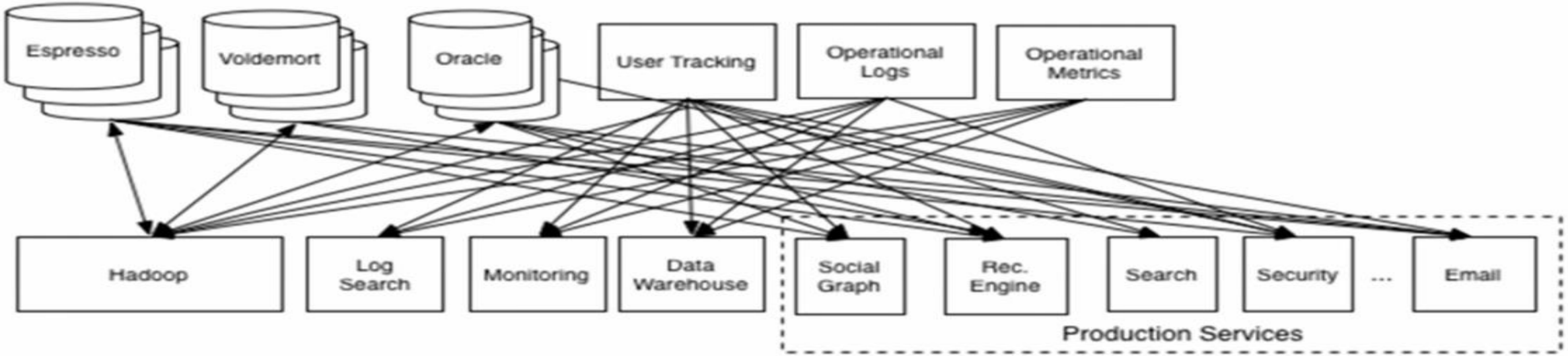
APACHE kafka™
A distributed streaming platform

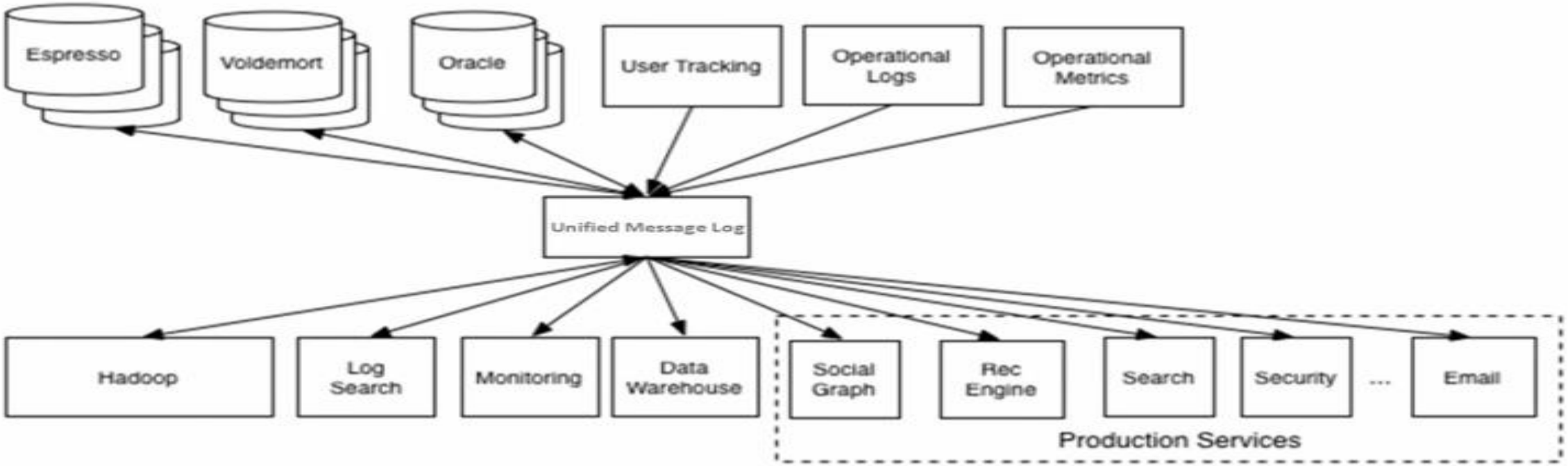
Introduction

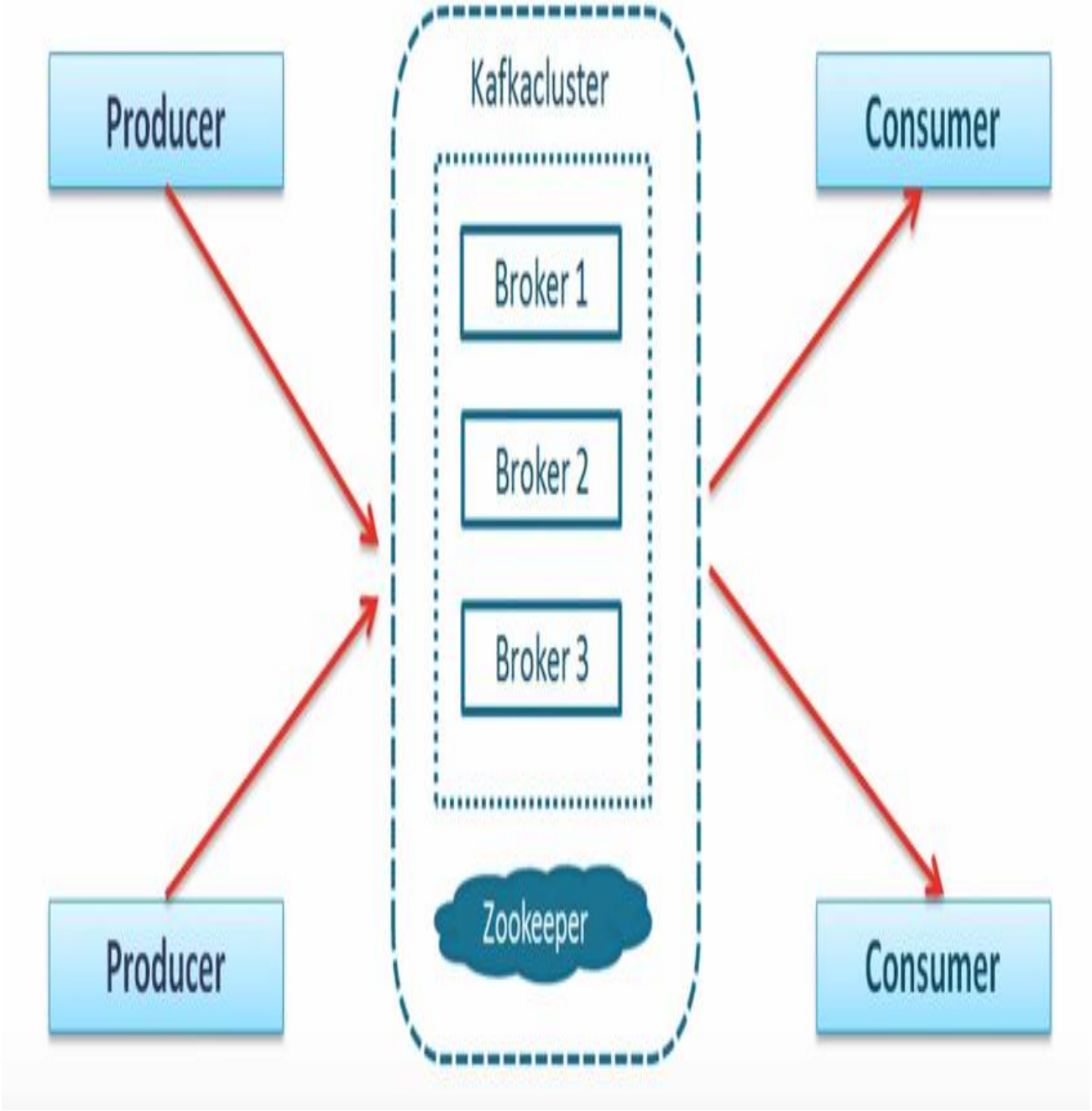
Kafka™ is *a distributed streaming platform*. What exactly does that mean?

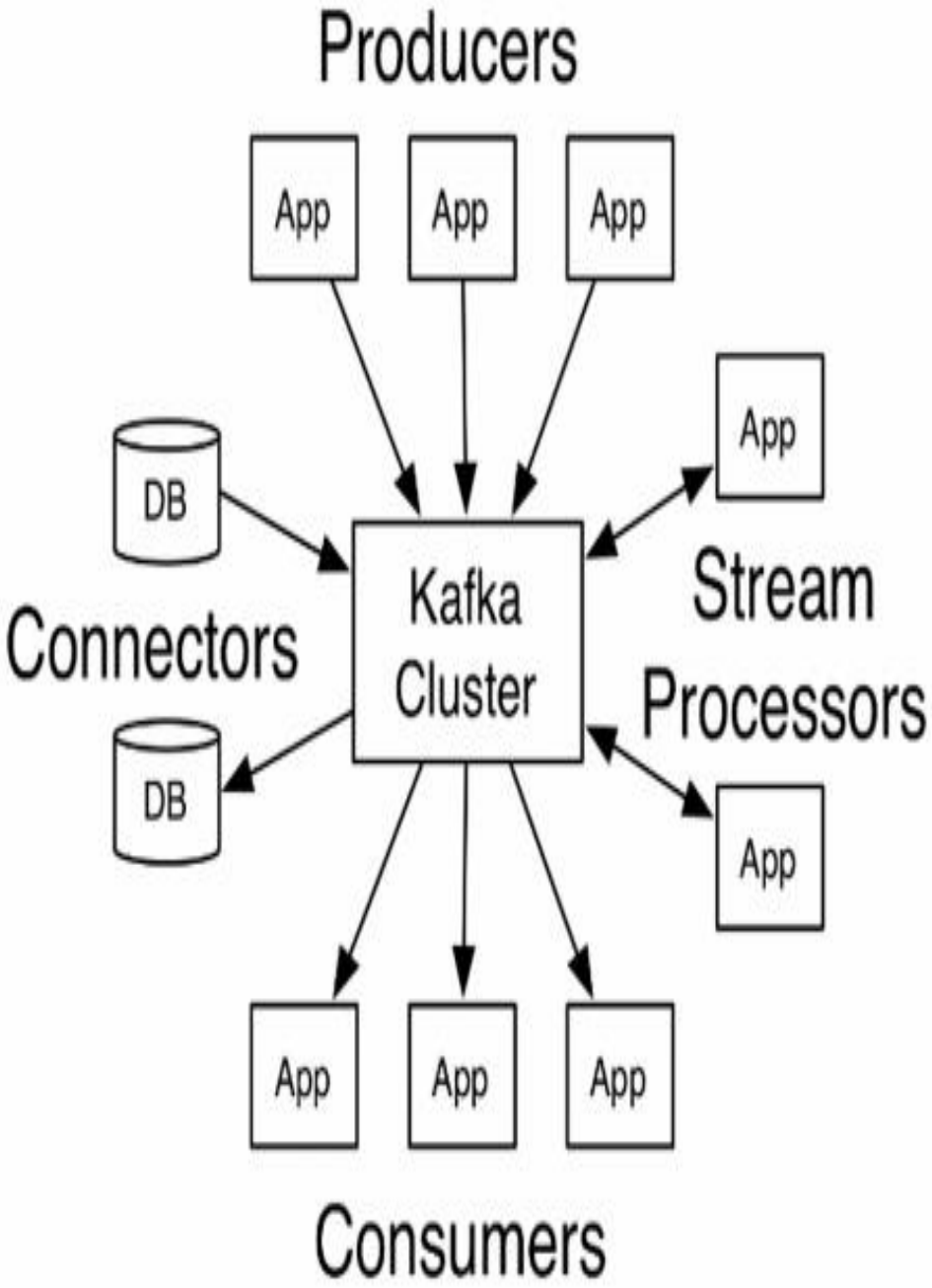
We think of a streaming platform as having three key capabilities:

1. It lets you publish and subscribe to streams of records. In this respect it is similar to a message queue or enterprise messaging system.
2. It lets you store streams of records in a fault-tolerant way.
3. It lets you process streams of records as they occur.


Espresso
Voldemort
Oracle
User Tracking
Operational Logs
Operational Metrics
Hadoop
Log Search
Monitoring
Data Warehouse
Social Graph
Rec. Engine
Search
Security
...
Email
Production Services

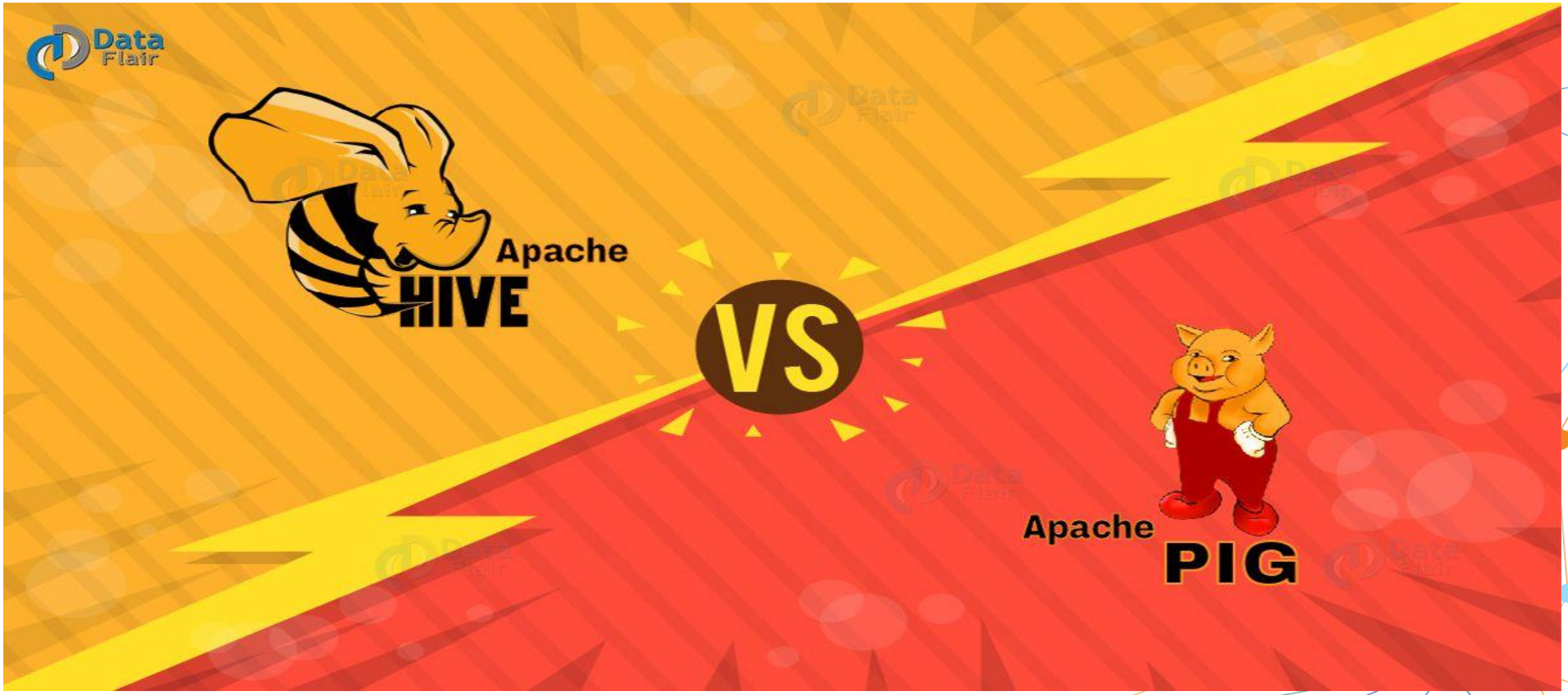

Espresso
Voldemort
Oracle
User Tracking
Operational Logs
Operational Metrics
Unified Message Log
Hadoop
Log Search
Monitoring
Data Warehouse
Social Graph
Rec Engine
Search
Security
...
Email
Production Services


Producer
Producer
Kafkacluster
Broker 1
Broker 2
Broker 3
Zookeeper
Consumer
Consumer


Producers
App
App
App
App
DB
DB
Kafka
Cluster
Connectors
Stream
Processors
App
App
App
App
App
Consumers

The Center of Applied Data Science

Languages in Hadoop Ecosystem



Overview

- Platform for analyzing large data sets.
- Apache Pig employs Pig Latin for queries and data manipulation.
- Apache Pig has competitive advantages to perform more complex data manipulation queries by providing:
  - Nested data types like Maps, Tuples, and Bags.
  - Support to major data operations like Ordering, Filters, and Joins.

Application

- Suitable for constructing scheduled job. Hence, it is appropriate for automated batch jobs that move data between HDFS and other systems.

- Suitable to read data from the databases reside in Hadoop that are not structured with Apache Hive metadata schemas.

Apache Pig vs MapReduce

- High-level data flow tool
- No need to write complex programs
- Built-in support for data operations like joins, filters, ordering, sorting etc.
- Provides nested data types like tuples, bags, and maps

- Low-level data processing paradigm
- You need write programs in Java/Python etc.
- Performing data operations in MapReduce is a humongous task
- Nested data types are not there in MapReduce

Why Apache Pig?

→ Can take any data
- Structured data
- Semi-Structured data
- Unstructured data

→ Easy to learn, Easy to write and Easy to read
- Data Flow Language
- Reads like a series of steps

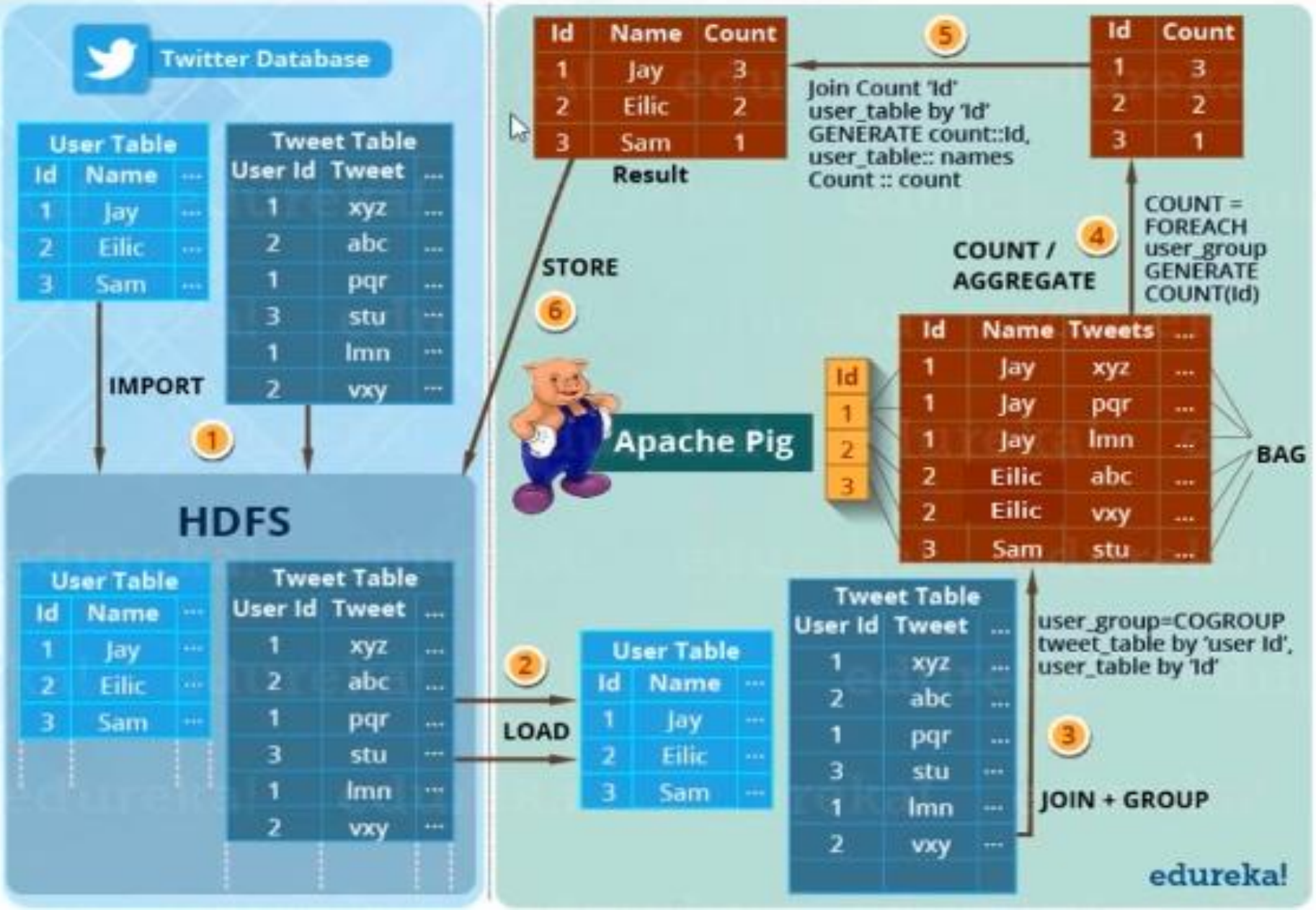
→ Extensible by UDF (User Defined Functions)
- Java
- Python
- JavaScript
- Ruby

→ Provides common data operations filters, joins, ordering, etc. and nested data types tuples, bags, and maps missing from MapReduce.

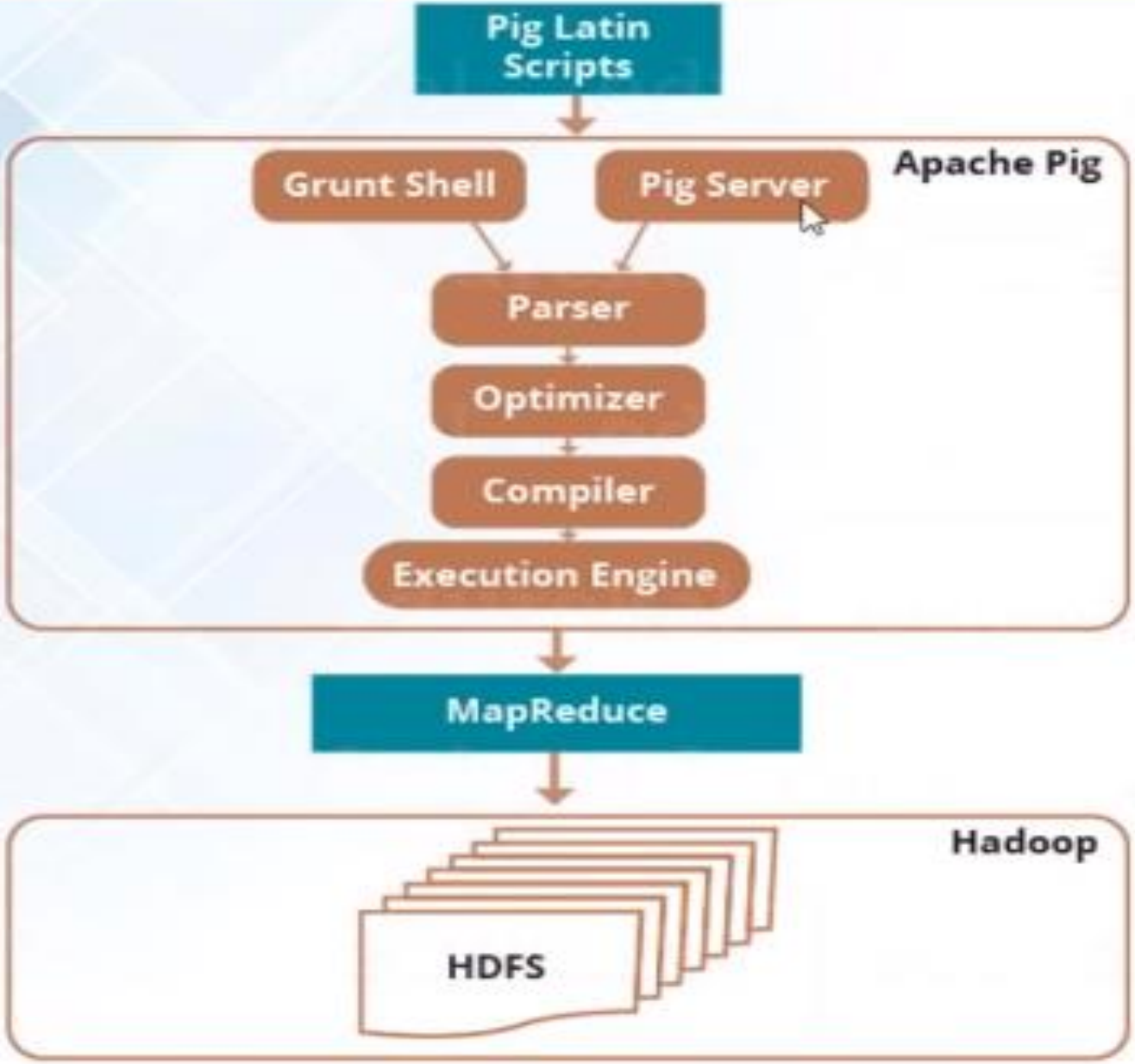
→ An ad-hoc way of creating and executing map-reduce jobs on very large data sets

→ Open source and actively supported by a community of developers.

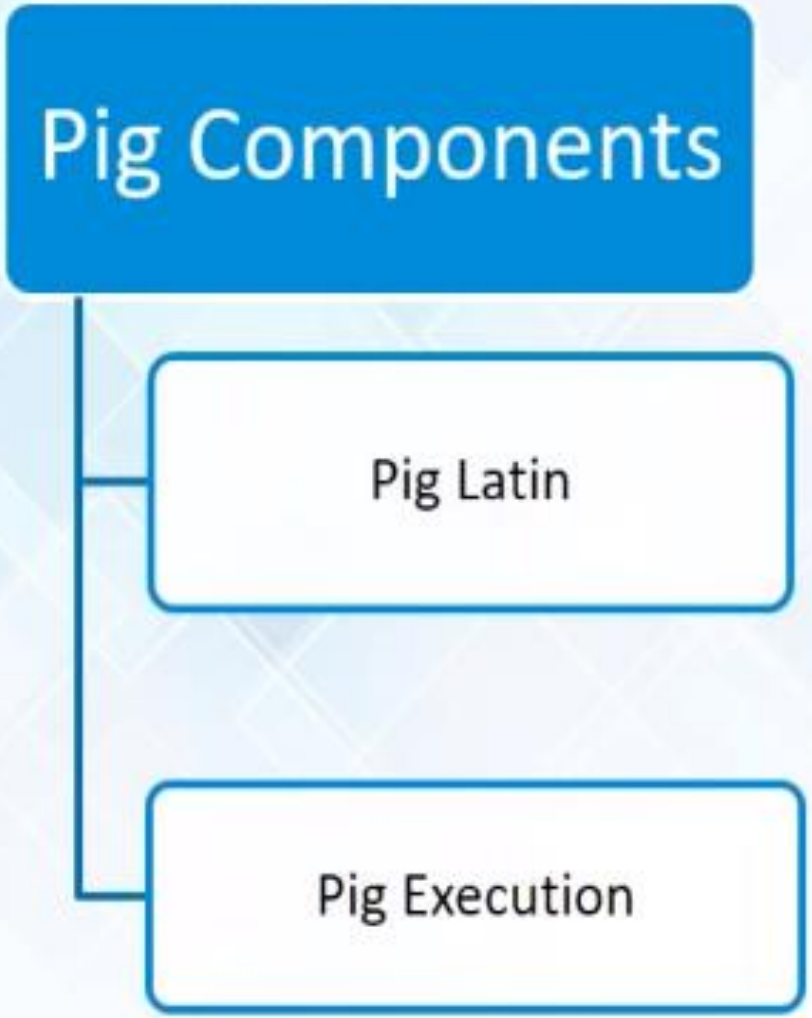
Detailed Implementation Flow



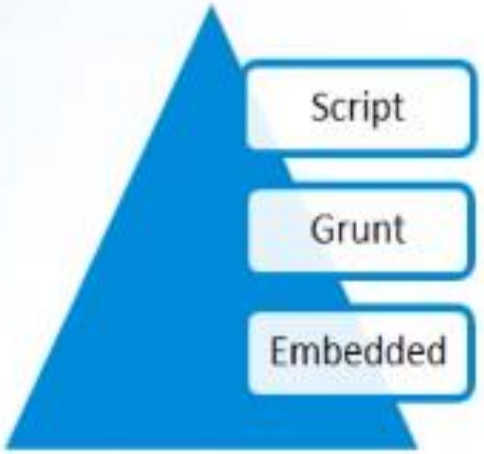
Apache Pig Architecture



Apache Pig Components



It is made up of a series of operations or transformations that are applied to the input data to produce output.



Contains Pig commands in a file (.pig)

Interactive shell for running Pig commands

Provisioning pig script in Java

Overview

- Virtual data warehouse software to perform MapReduce based SQL engine that runs on top of Hadoop.

- Apache Hive employs HiveQL (SQL-like query language) to access the files stored in Apache HDFS or other data storage system such as Apache Hbase.

Application

- Suitable to build data warehouse without requiring programmers to write complex MapReduce code.

- A real world application is the friend recommendation system on Facebook. Recommendation system has two characteristics:
  - Require high volume of input data.
  - Outputs/Recommendations do not change frequently.

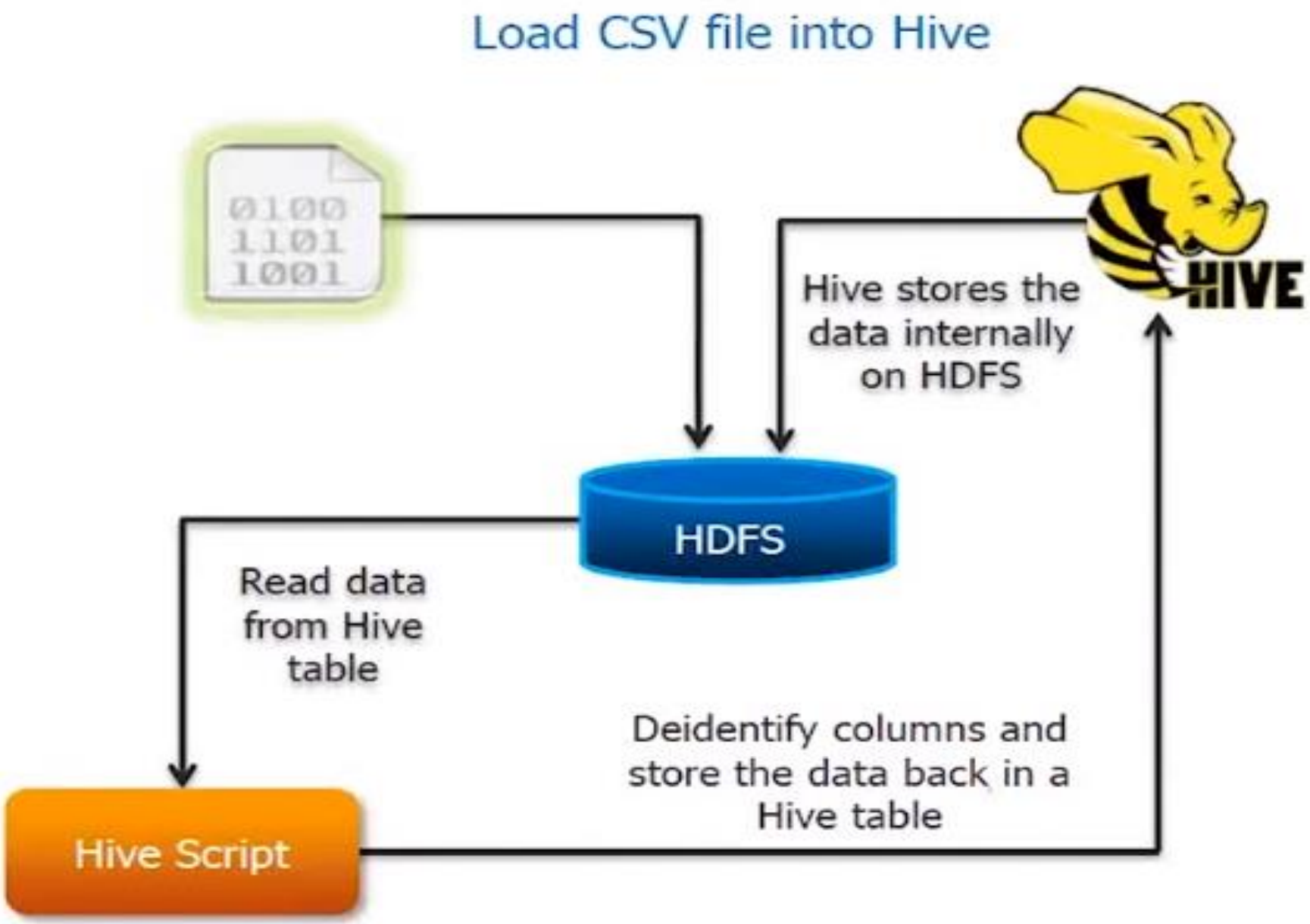
Traditional RDBMS... X

Solution...

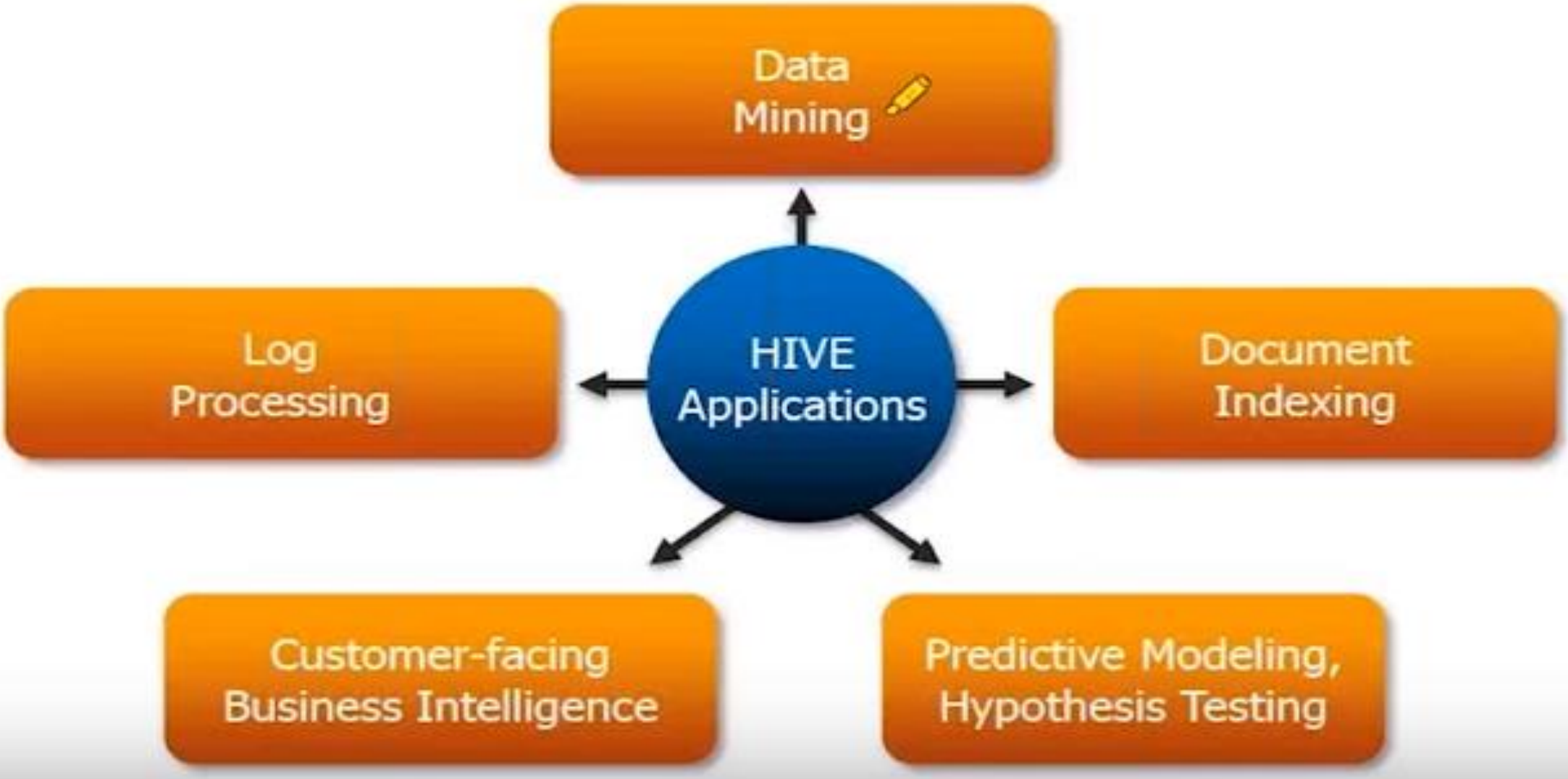
What is Hive?

- ✓ Data Warehousing package built on top of Hadoop
- ✓ Used for data analysis
- ✓ Targeted towards users comfortable with SQL
- ✓ It is similar to SQL and called HiveQL
- ✓ For managing and querying structured data
- ✓ Abstracts complexity of Hadoop
- ✓ No need learn java and Hadoop APIs
- ✓ Developed by Facebook and contributed to community
- ✓ Facebook analyzed several Terabytes of data everyday using Hive

Use Case in Healthcare



Where to use Hive?



Why go for Hive When Pig is there?

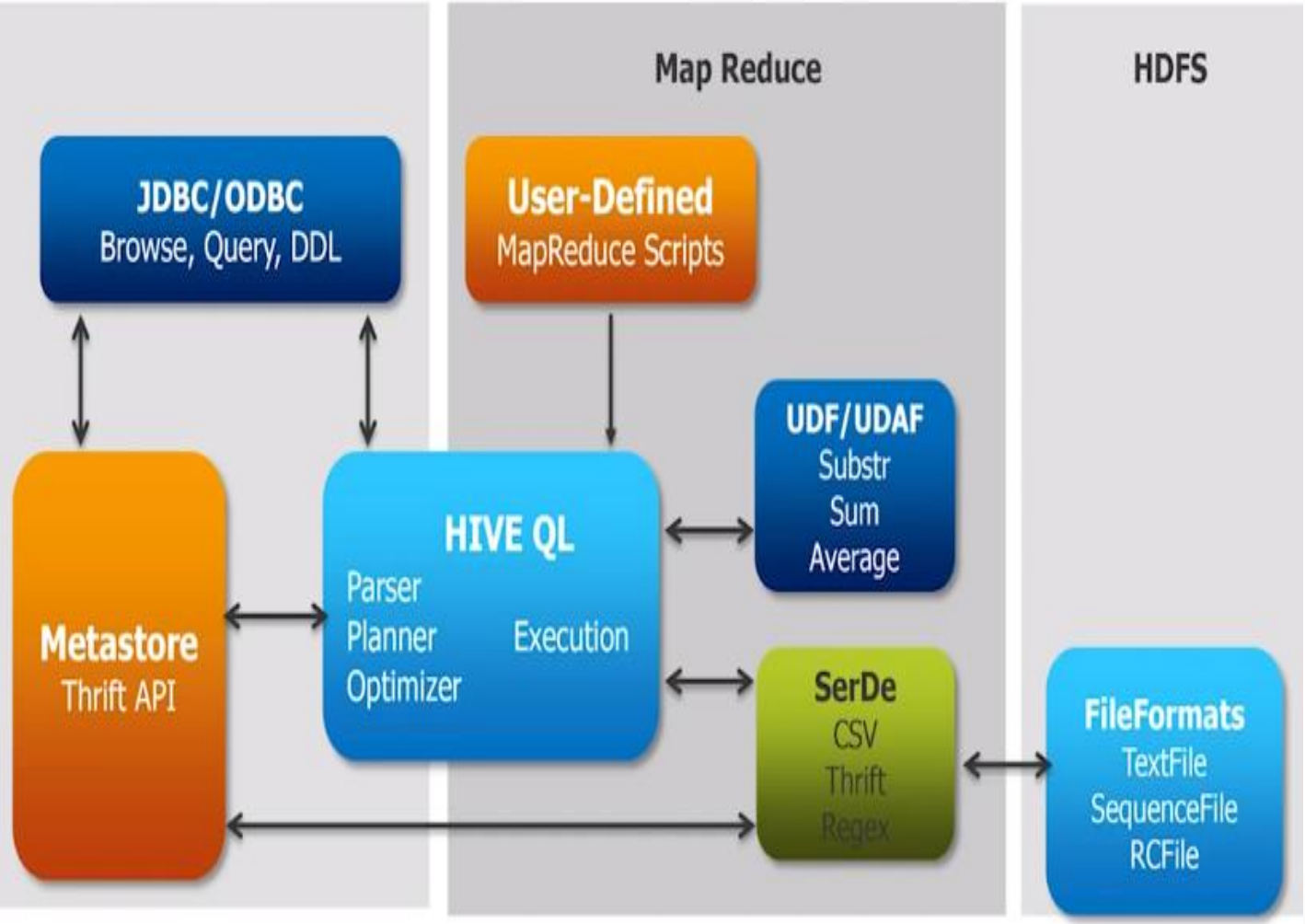
PigLatin:

- ✓ Procedural data-flow language
  A = load 'mydata';
  Dump A;
- ✓ Pig is used by Programmers and Researchers

HiveQL:

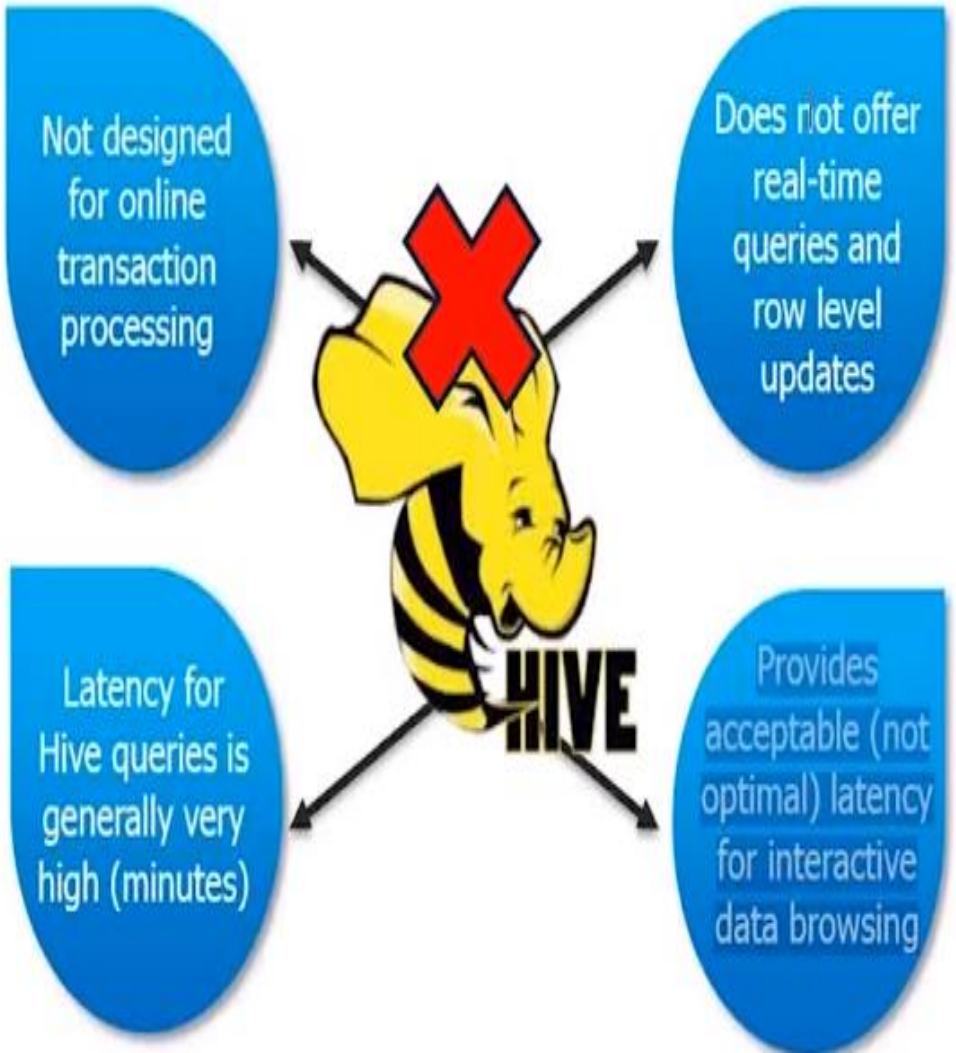
- ✓ Declarative SQLish language
  Select * from 'mytable';
- ✓ Hive is used by Analysts generating daily reports

Hive Architecture



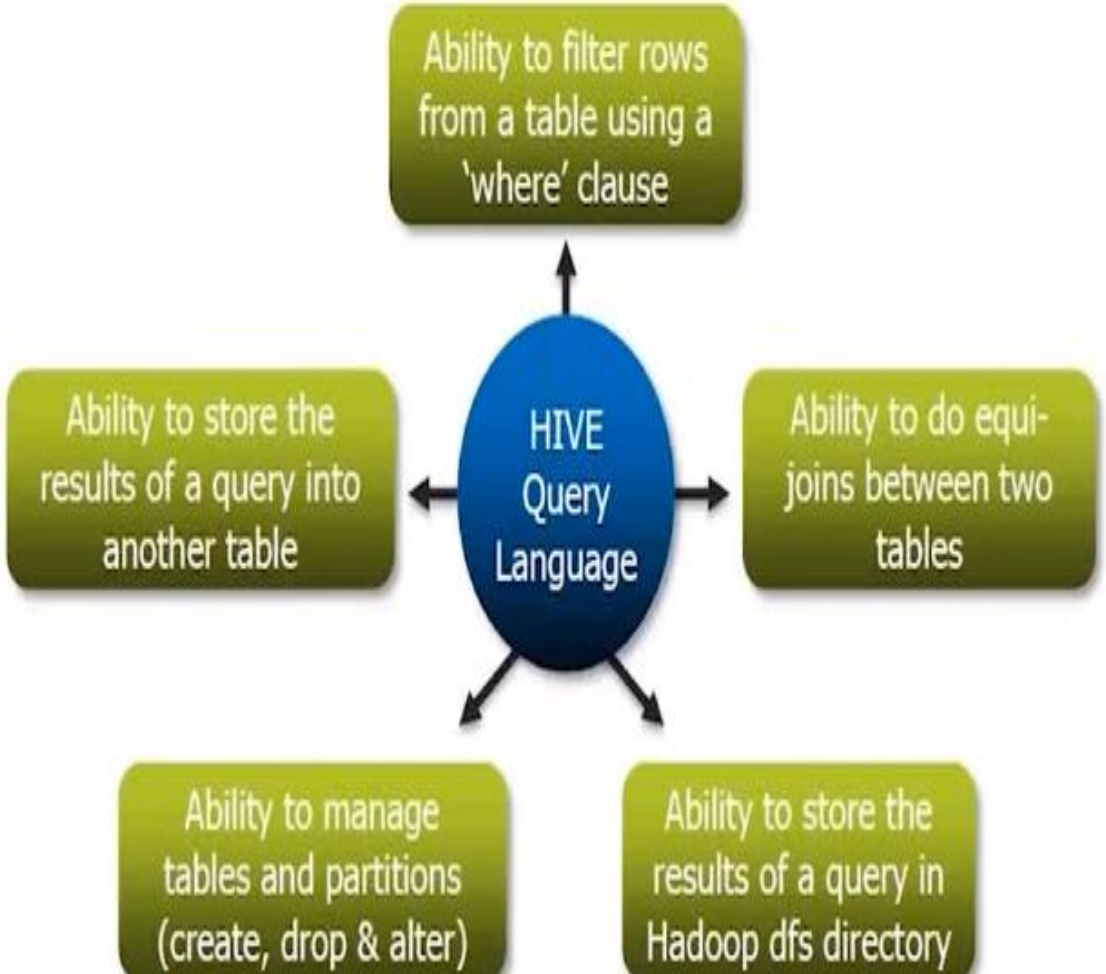
Hive Components

Limitations of HIVE?



Abilities of HIVE Query Language

Hive Query Language provides the basic SQL-like operations



Hive Components

Overview

- Distributed, scalable, and multi-level big data store on top of Hadoop and HDFS.

- NoSQL database used for real-time data streaming due to the two advantages:
  - Able to provide fast and random read-writes.
  - Able to work well with sparse data due to the column-oriented property

Application

- Suitable for random, real-time read/write access to big data.

- Real world applications of Apache Hbase:
  - Helps Facebook to perform real-time analytics, such as counting Facebook likes and for messaging.
  - Helps Financial Industry Regulatory Authority (FINRA) and Pinterest to store graphs.
  - Helps Flipboard to personalize the content feef for the users.

NoSQL Landscape

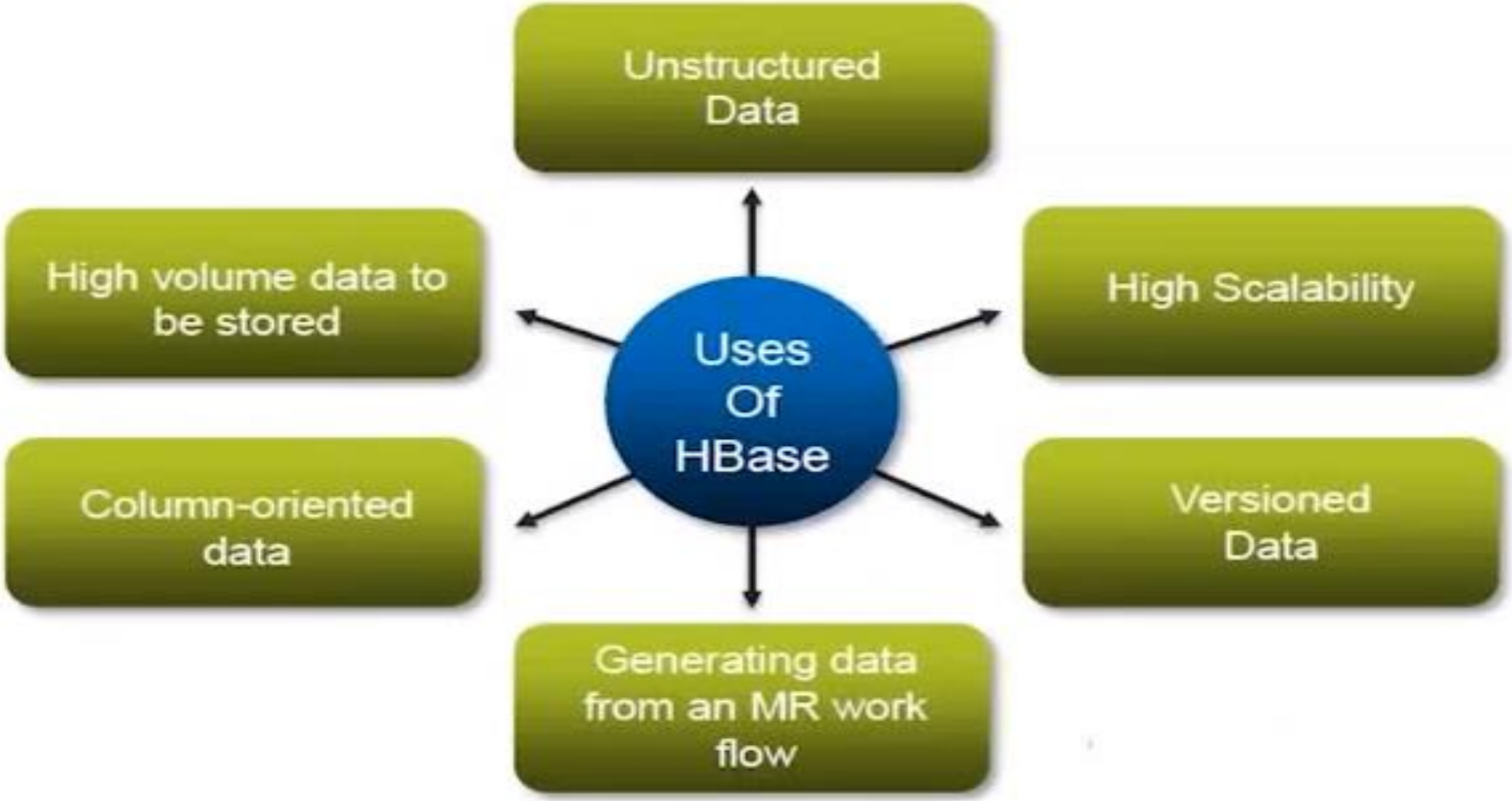
Hbase Definition

- ✓ HBase is a key/value store. Specifically it is:

HBase Vs. RDBMS

HBase	RDBMS
Column-oriented	Row oriented (mostly)
Flexible schema, add columns on the fly	Fixed schema.
Good with sparse tables,	Not optimized for sparse tables.
Joins using MR –not optimized	Optimized for joins.
Tight integration with MR	Not really...
Horizontal scalability –just add hardware	Hard to shard and scale
Good for semi-structured data as well as structured data	Good for structured data

When to use HBase?



More HBase Implementation

A number of applications including people search rely on HBase internally for data generation.

Uses HBase to power their Messages infrastructure http://sites.computer.org\debull\A12june\facebook.pdf

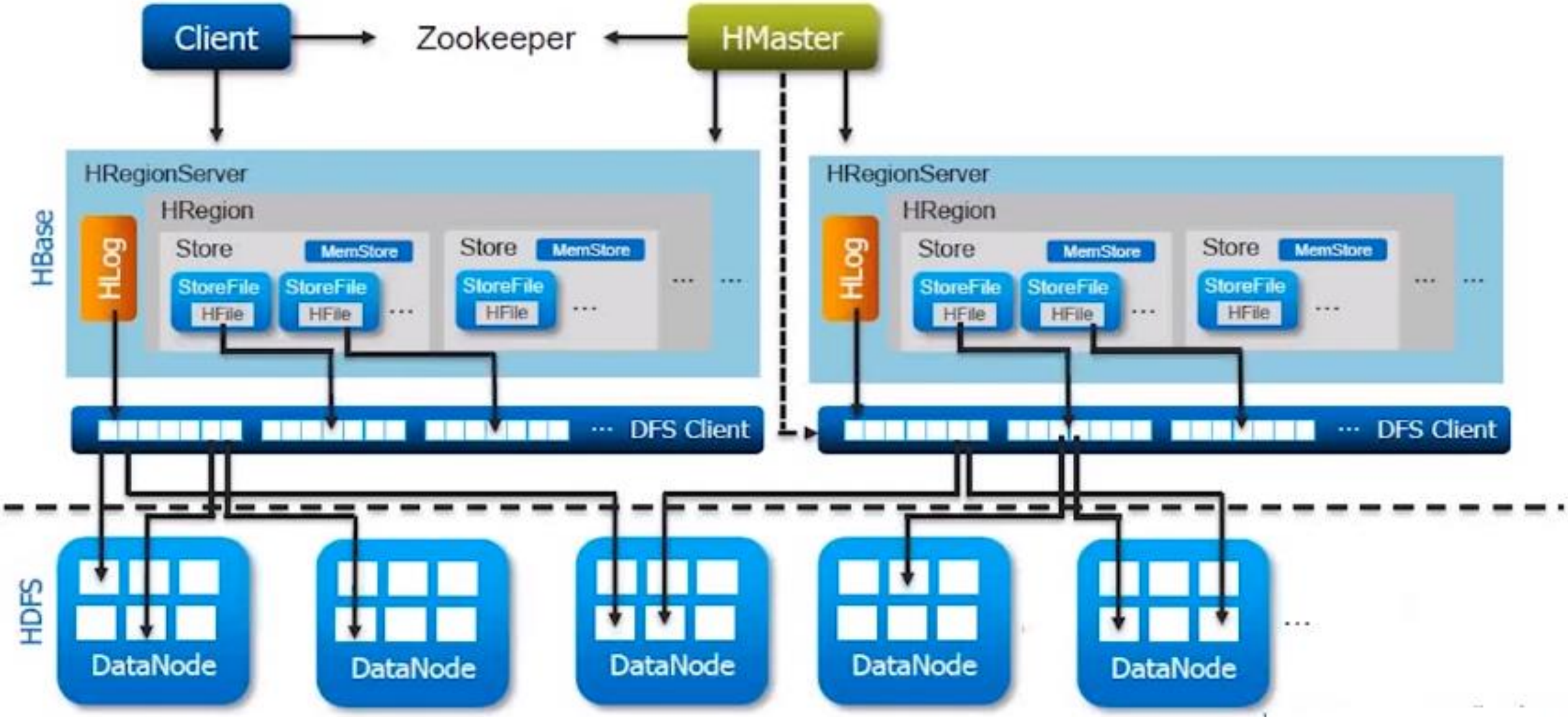
We use HBase as a real time data storage and analytics platform.

Uses HBase to store document fingerprint for detecting near-duplications. We have a cluster of few nodes that runs HDFS, mapreduce, and HBase.

Uses an HBase cluster containing over a billion anonymized clinical records.

Uses HBase as a foundation for cloud scale storage for a variety of applications.

HBase Storage Architecture



Overview

Apache ZooKeeper is a software project of the Apache Software Foundation, providing an open source distributed configuration service, synchronization service, and naming registry for large distributed systems

Why Did we need ZooKeeper - PROBLEMS

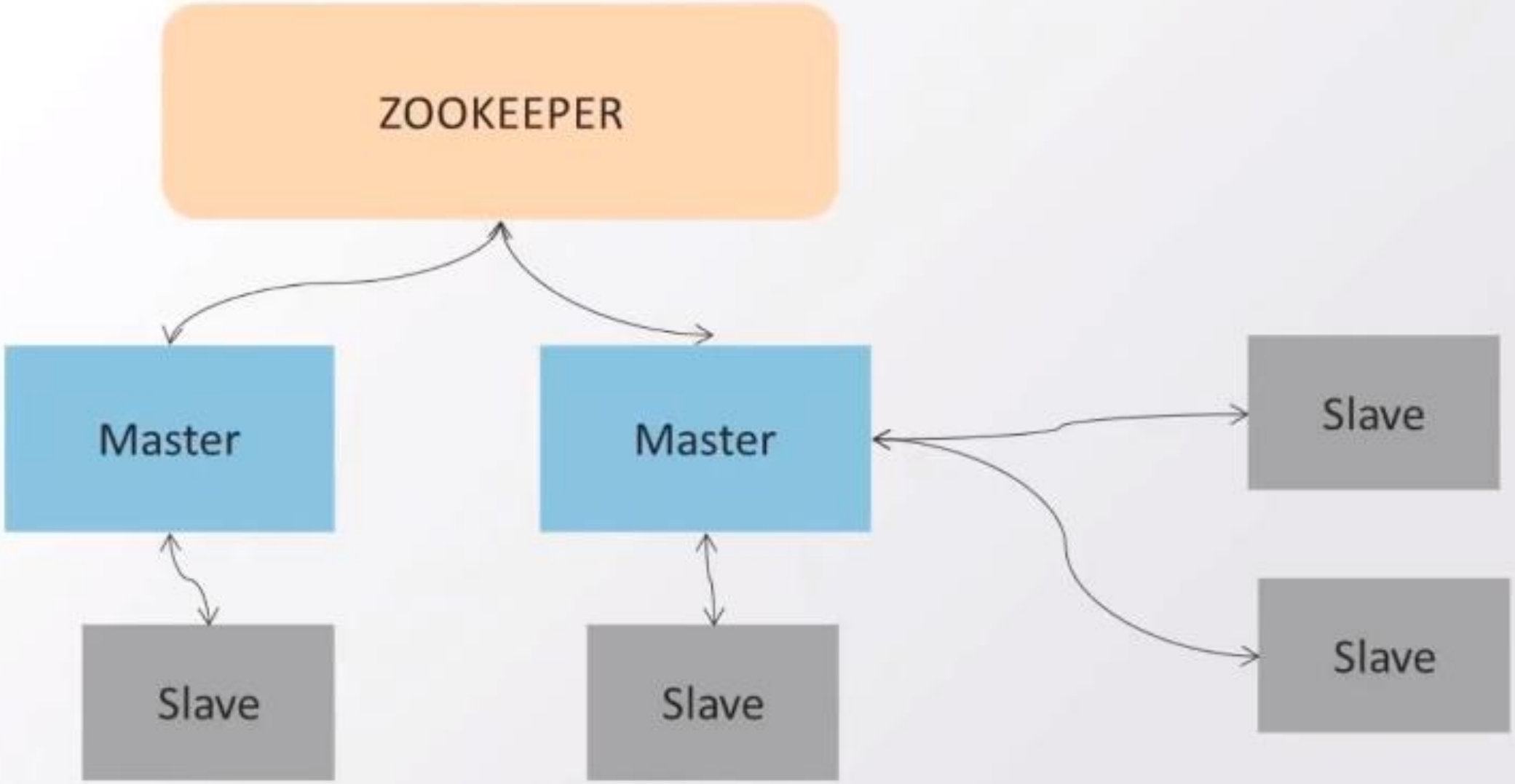
Controller is a critical part of the system, but not much related to application

- Single point of failure
- Not reliable(gets the least amount of resources)
- Limited scalability

Redundant, HA services difficult to implement and test

Admin need to know the setup, monitor and recover from failures

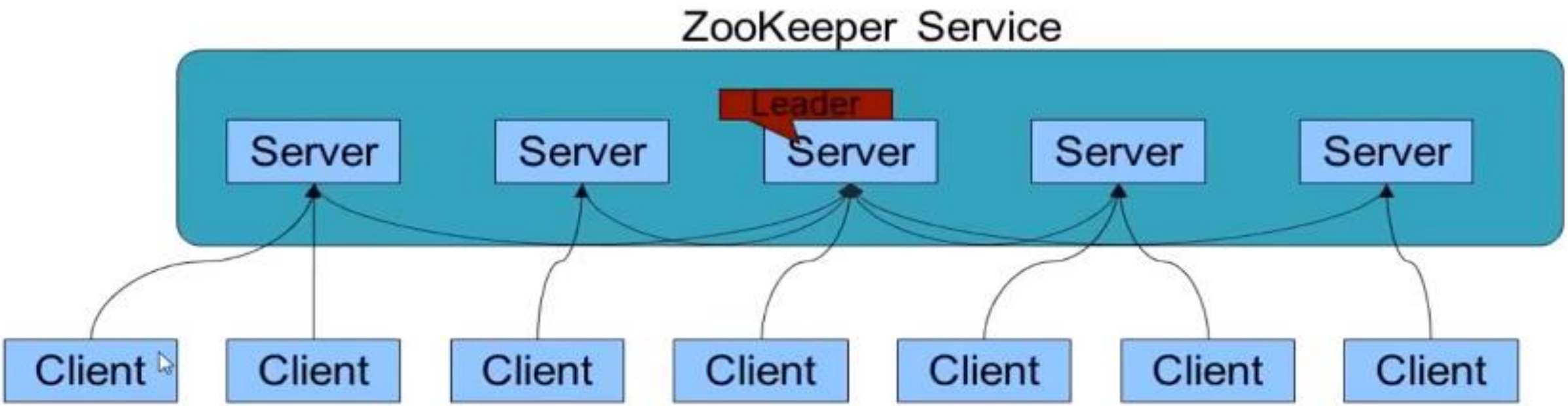
ZOOKEEPER invented to simplify this



How Information is stored in ZK

- ✓ Zookeeper stores information in tree like structure
- ✓ It is a traditional file system like structure. A file is called Znode
- ✓ A file can store data and reference to other Znodes(i.e. Its children).
- ✓ Each Znode can store data upto 1 MB.
- ✓ Data is kept in memory and is backed up to a log for reliability. By using memory, ZooKeeper is very fast and can handle the high loads.

ZooKeeper Quorum



- All servers store a copy of the data (in memory)
- A leader is elected at startup
- Followers service clients, all updates go through leader
- Update responses are sent when a majority of servers have persisted the change

Role in Hadoop Infrastructure

- HBase
  - Use ZooKeeper for master election, server lease management, bootstrapping, and coordination between servers.
- Hadoop and MapReduce
  - Using ZooKeeper to aid in high availability of ResourceManager
  - Adaptive MapReduce (IBM Big Insights)
- Flume
  - Using ZooKeeper for configuration purposes in recent releases

Real world use cases

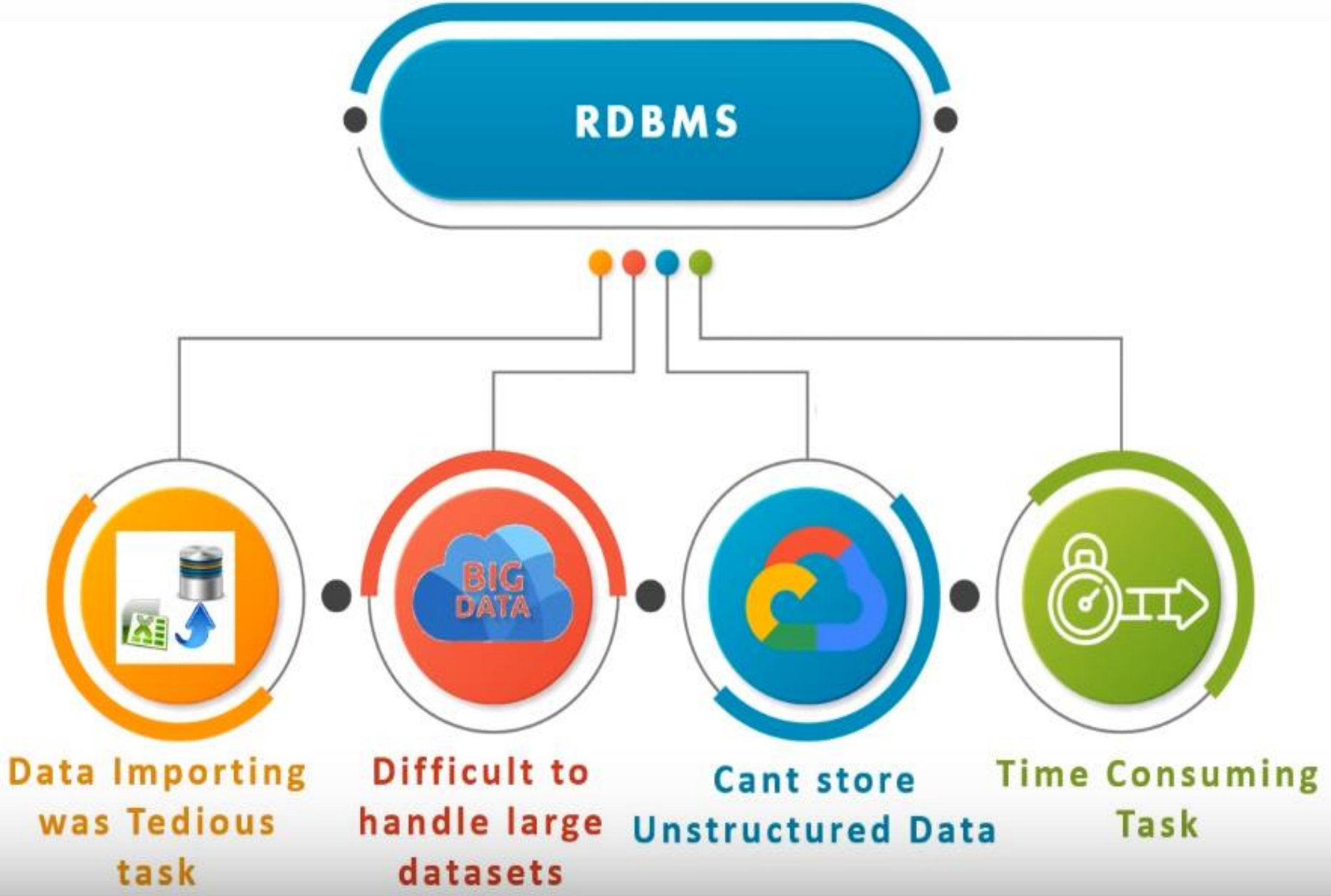
- **Rackspace** – "*The Email & Apps team uses ZooKeeper to coordinate sharding and responsibility changes in a distributed e-mail client that pulls and indexes data for search ZooKeeper also provides distributed locking for connections to prevent a cluster from overwhelming servers.*"
- **Twitter** – Service Discovery
- **Vast.com** – "*Used internally as a part of sharding services, distributed synchronization o data/index updates, configuration management and failover support*"
- **Yahoo!** – "*ZooKeeper is used for a myriad of services inside Yahoo! for doing leader election, configuration management, sharding, locking, group membership etc.*"
- **Zynga** – "*…used for a variety of services including configuration management, leader election, sharding and more…*"

(sources: Apache ZooKeeper wiki and blog.twitter.com)

Overview

- A tool for automating the transfer process of bulk data between Apache Hadoop and structured datastores efficiently.
- Able to execute the data transfer in parallel.

Problems with Relational Database



Need for Sqoop

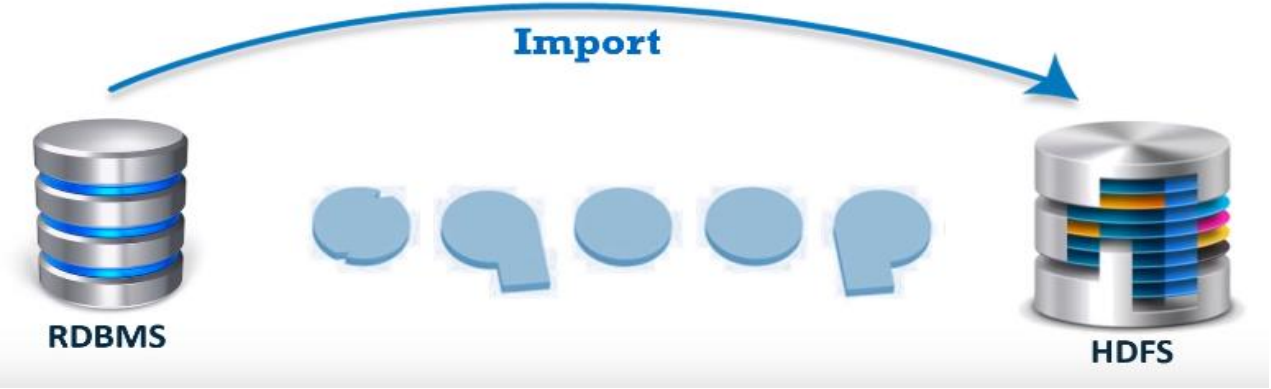
Data Analysis

Command Line Interface

High Performance

What is Sqoop?

Tool used to transfer bulk data between HDFS & Relational Database Servers



Features of Sqoop

Full Load

Data loading directly to HIVE

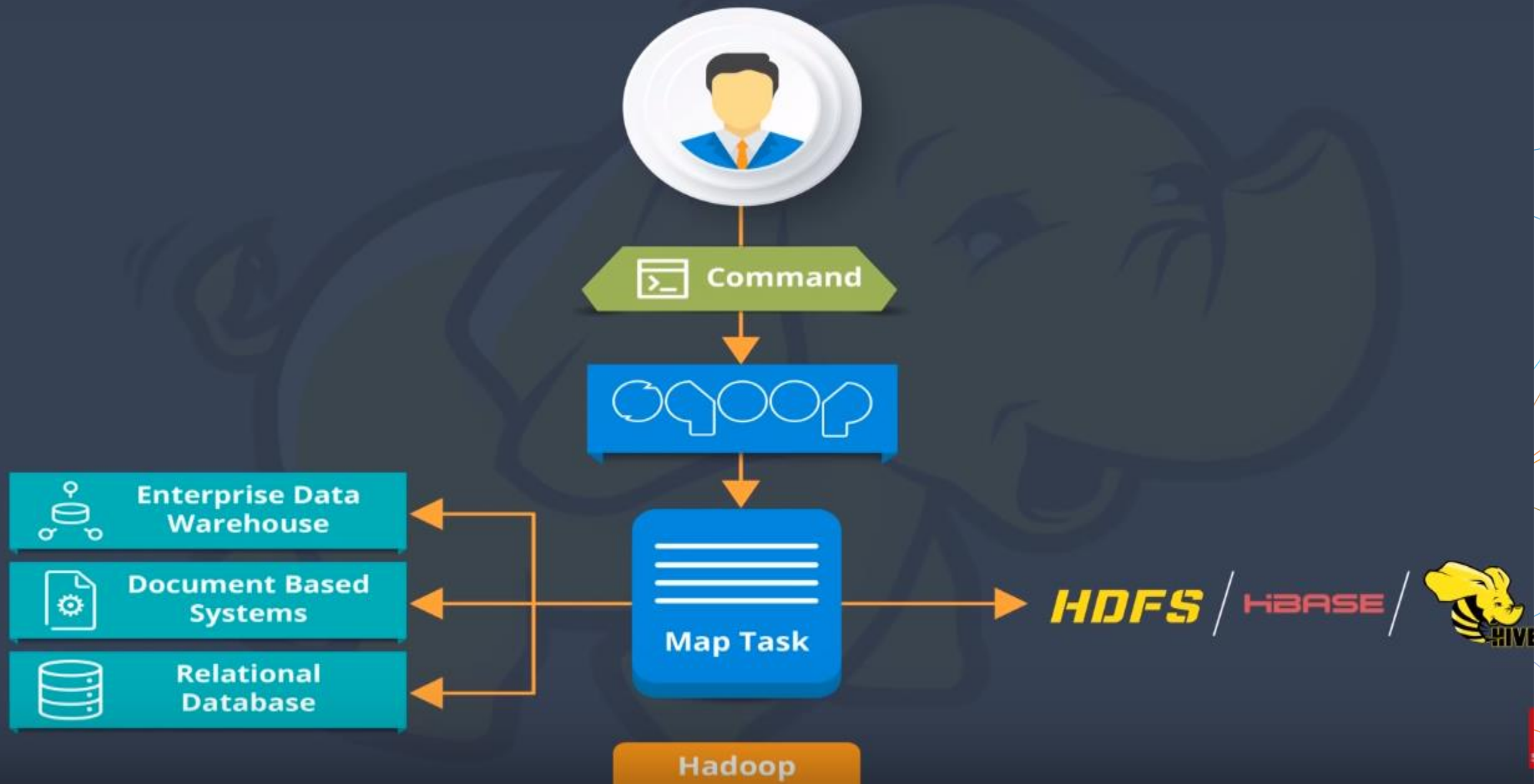
Incremental Load

Kerberos Security Integration

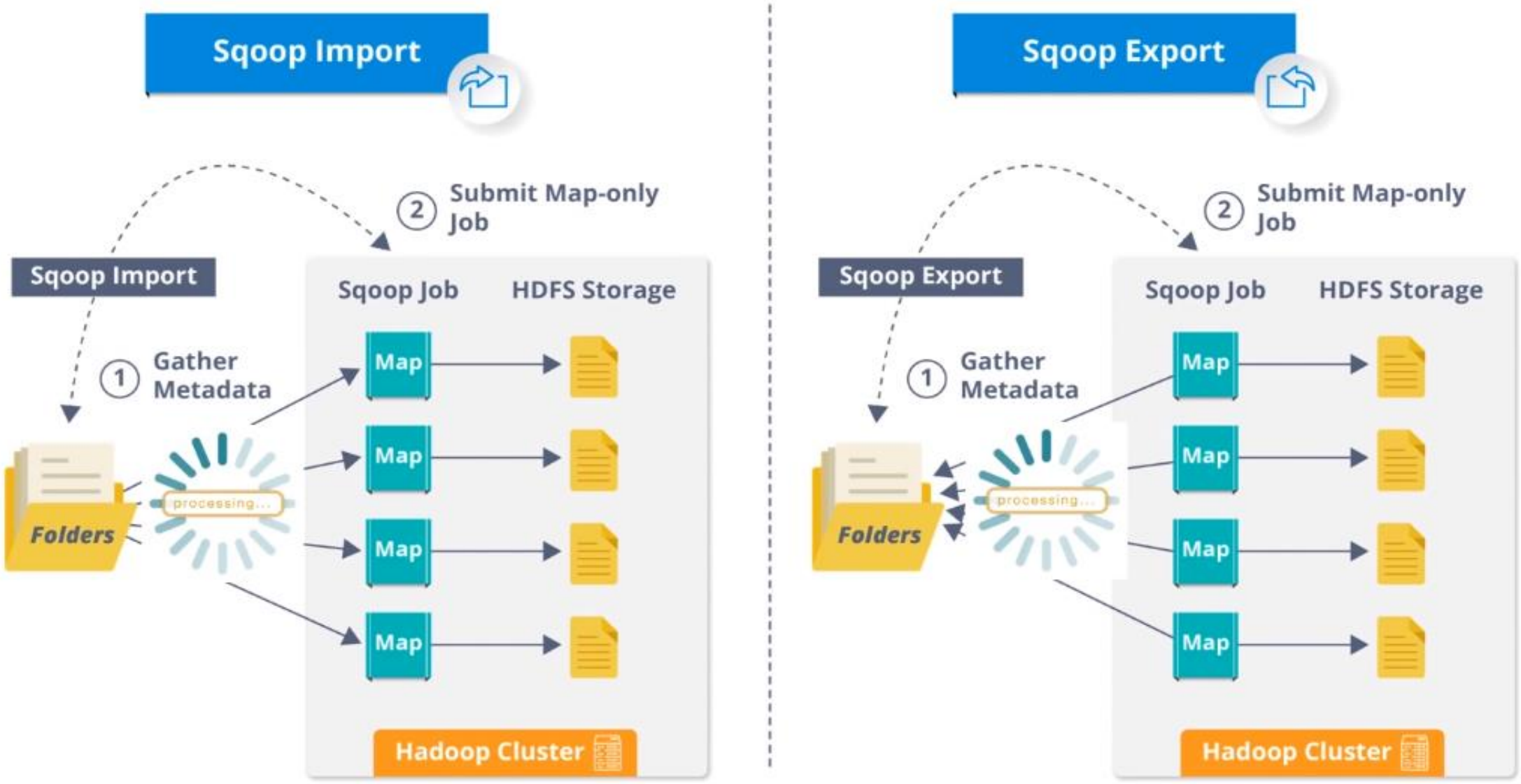
Parallel Import/Export

Compression

Sqoop Architecture



How Sqoop Import & Export Works?



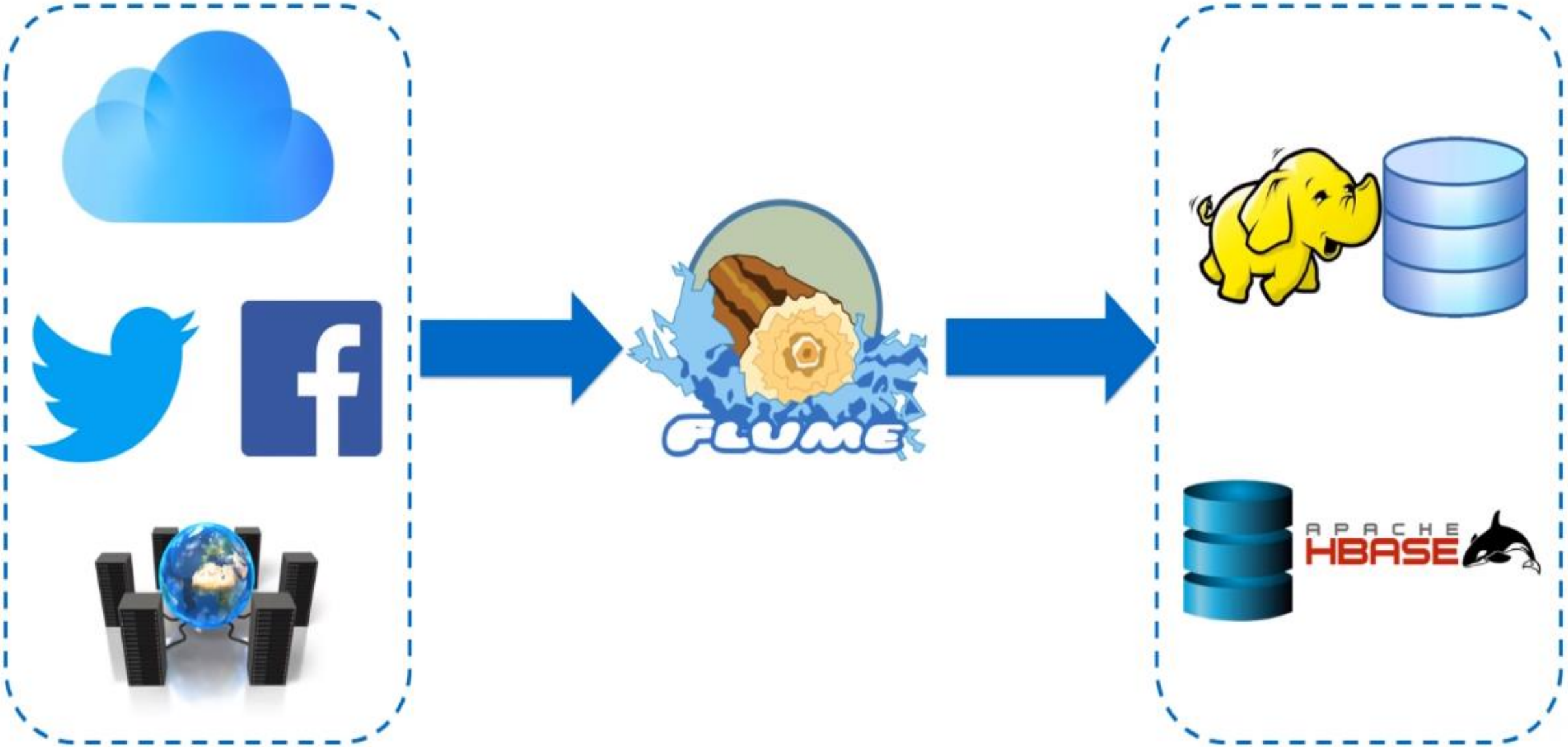
Overview

- A tool to collect, aggregate, and transport large amounts of streaming data from variety of sources in both real-time and batch mode to a centralized data store (e.g. HDFS).

Application

- Suitable to import huge volumes of event data generated by websites such as Facebook, Twitter, Amazon, and Flipkart in real-time.

What is Apache Flume?



Advantages of Apache Flume

Scalable

Reliable

Fault Tolerant

Centralized Data Storage

Data Ingestion

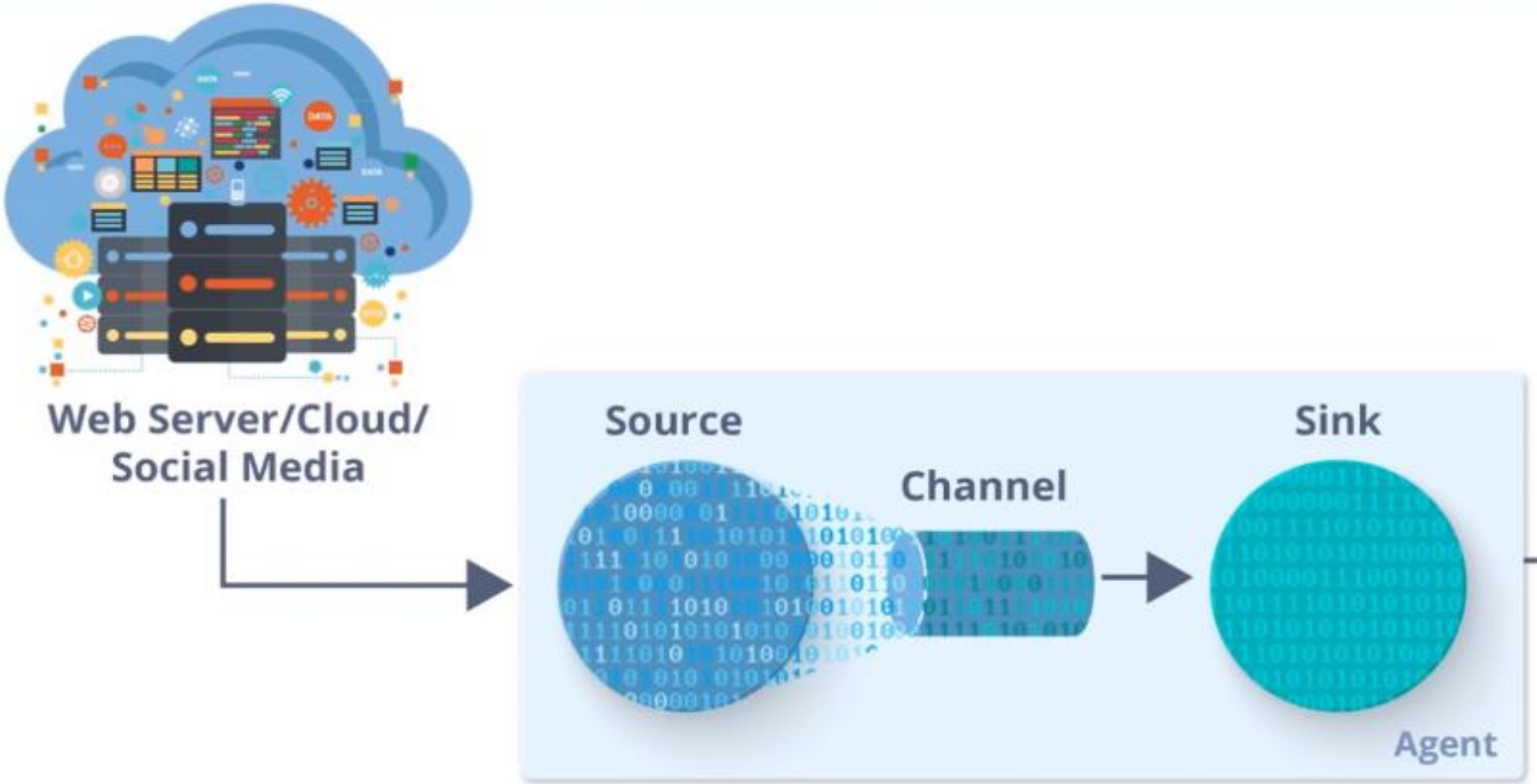
Large Data Sets

Multiple hop flows

Real-time data streaming

Source & Destination

Apache Flume Architecture



Twitter Data Streaming

Overview

- A low latency high performance SQL like queries engine to query data that stored on Hadoop clusters (e.g. HDFS, Apache Hbase) in real-time.

- Impala shares the same SQL syntax (Hive SQL), ODBC driver, metadata, and user interface (Hue Beeswax) as Apache Hive. Hive users can then use Impala with little setup overhead.

Application

- Suitable for the interactive applications that require complicated queries to react relatively fast. It allows users to obtain the outputs to the unexpected questions (complicated queries) in seconds or at most a few minutes.

Cloudera Impala is a massively parallel processing (MPP) SQL query execution engine that runs on the Hadoop platform.

Impala can be used to:

- run a query, evaluate the results immediately, and fine-tune the query.
- analyze Hadoop data via SQL or other business intelligence tools.
- perform interactive, ad-hoc, and batch queries together in the Hadoop system.

Benefits of Impala

The following are some benefits of Impala:

Flexibility

- Integrates well with the existing Hadoop components
- Adds capabilities that make SQL querying easier than before

Enhanced SQL Query Speed

- Enables new categories of solutions
- Enables interactive queries

Presence of Apache License

- Enables users to query low-latency SQL data from HDFS and Apache Hbase without causing any data movement or transformation

xploratory Business Intelligence

Advantages of Impala in exploratory business intelligence:

Prior to Using Impala

- Business intelligence data was typically condensed to a manageable chunk of high-value information.
- Information passed through a complicated ETL cycle before being uploaded to a database.

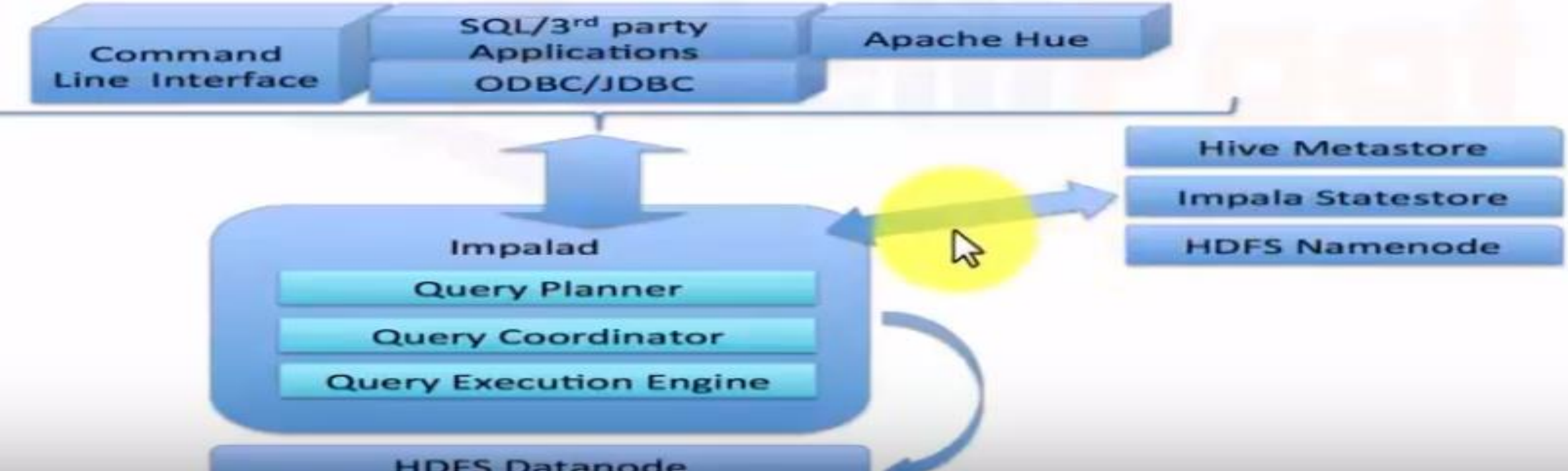
Using Impala

- The data arrives in Hadoop after fewer steps, and Impala queries it immediately.
- The high-capacity and high-speed storage system of a Hadoop cluster lets you bring in all the data.

Impala core components

Following three are the major components of Impala

- Impala daemon
- Impala State Store
- Impala Metadata and Metastore



Impala vs Hive

- Impala is SQL on HDFS while Hive is SQL on Hadoop
- Impala uses MPP engine to distribute the query processing while Hive uses Map Reduce
- No MR overhead in case of Impala while Hadoop, since use MR, has the overhead attached.
- Impala is not fault tolerant while Hive has the usual fault tolerance of Map Reduce
- MapReduce materializes all intermediate results, which enables better scalability and fault tolerance (while slowing down data processing). Impala streams intermediate results between executors (trading off scalability).
- Impala is recommended for real time SQL queries while Hive is recommended for large batch jobs
- Hive generates query expressions at compile time whereas Impala does runtime code generation for "big loops".

Overview

- A schema-free SQL query engine for Hadoop, NoSQL, and cloud storage.
- Does not depend on Hadoop as Drill does not use MapReduce job internally. Drill has its own distributed processing service called DrillBit.

Application

- Can be used to join data from multiple datastores with just a single query.
- Can be used to connect between standard BI/analytics tools and non-relational datastores by leveraging Apache Drill's JDBC and ODBC drivers.

Apache Drill

- Inspired by Google's Dremel System (developed at Google for interactively querying large datasets and powers Google's BigQuery service)
- Apache Drill is industry's first schema-free SQL Engine
- Drill is designed to scale to several thousands of nodes and query petabytes of data at interactive speeds that BI/Analytics environments require

APACHE DRILL

Why Drill?

- Drill is the world's first and only distributed SQL engine that doesn't require schemas. No need to define and maintain schemas or transform data (ETL). Drill automatically understands the structure of the data

- It can integrate with several data sources like hive, hbase, mongodb , file system, your RDBMS etc. Avro, CSV, TSV, PSV, Parquet , Hadoop Sequence file , all these input formats can be used in Drill easily

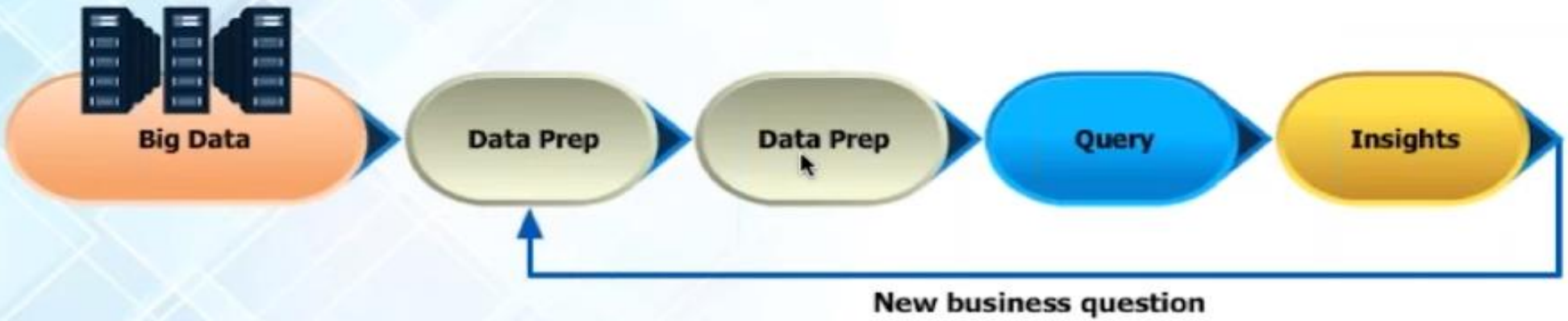
Why Drill?

- You can work with existing BI tools like Tableau, MicroStrategy, QlikView and Excel

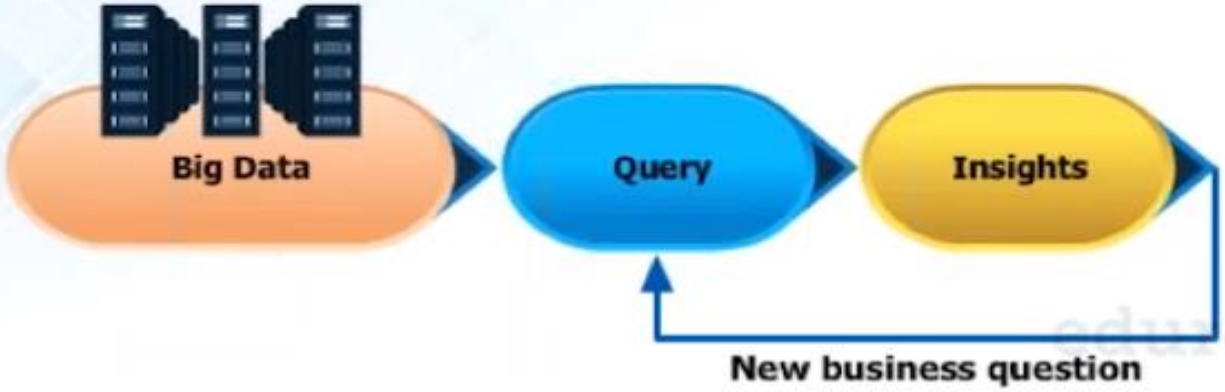
- It can combine data from multiple data sources on the fly in a single query
- You can deploy Drill on your Hadoop cluster (up to 1000 commodity servers), hence very much scalable
- Standard SQL 2003 support, no need to learn new sql type language

Without Drill Vs With Drill

Without Drill:



With Drill:



edureka!

Apache Arrow

Overview

- Built by the lead developers of many Apache projects.
- A component used to exchange data with low overhead and hence accelerating the data analytics.
- Apache Arrow is extremely important for Python and R communities as it provide data interoperability between the two communities with big data systems (which largely run on the JVM).

Apache Arrow

Application

- Used to reduce the time spent gathering and processing data. For example:
  - PySPark: IBM measured a 53x speedup in data processing by Python and Spark with the support of Apache Arrow in PySpark.

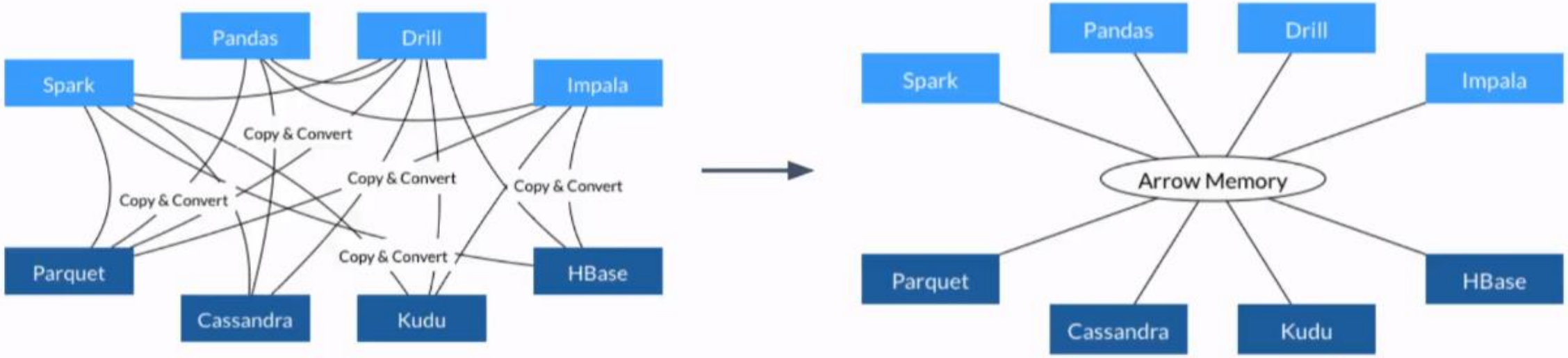
Apache Arrow

- https://github.com/apache/arrow
- Open source community initiative started in 2016
  - Backed by ~13 major OSS projects at start, significantly more now
- Shared standards and systems for memory interoperability and computation
- Cross-language libraries

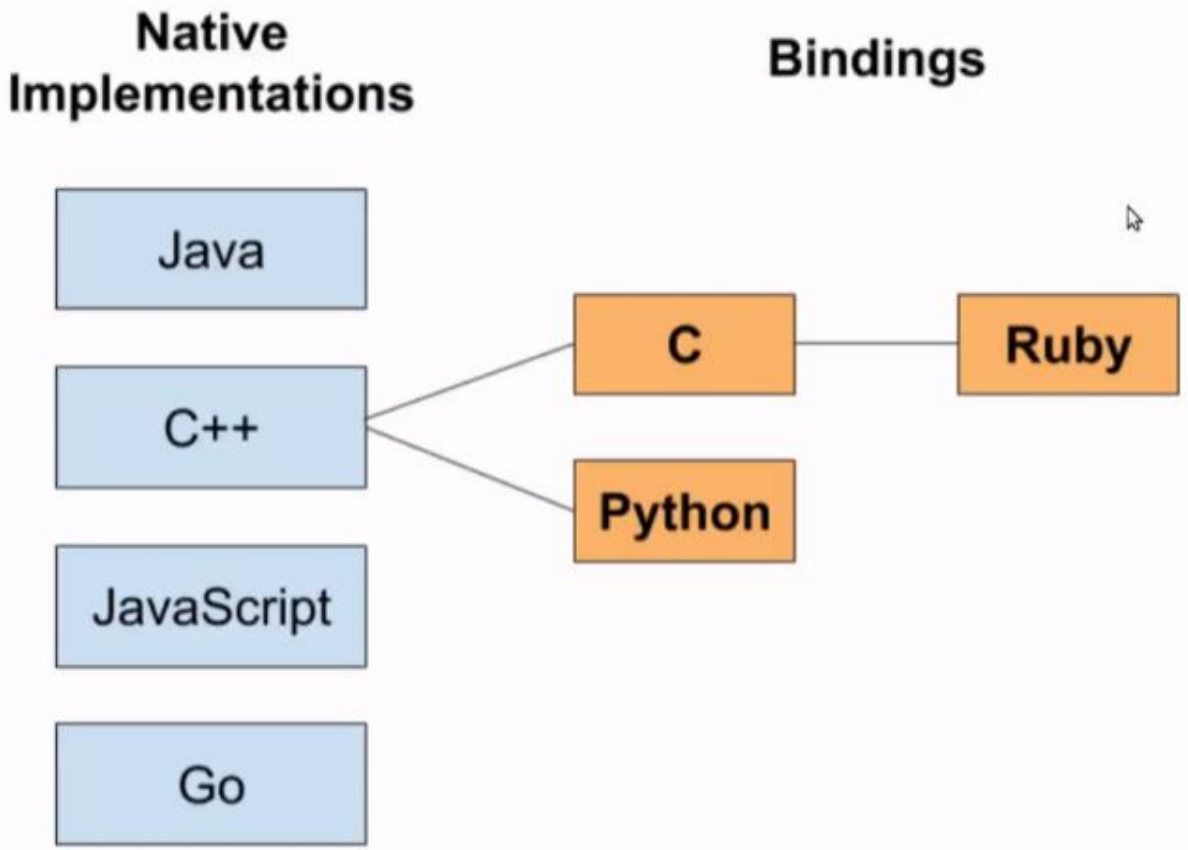
Some Arrow Use Cases

- Runtime in-memory format for analytical query engines
- Zero-copy (no deserialization) interchange via shared memory
- Low-overhead streaming messaging / RPC
- Serialization format implementation
  - Zero-copy random access to on-disk data
  - Example: Feather files
- Data ingest / data access

Defragmenting Data Interchange



Arrow Implementations and Bindings



Upcoming: Rust (native), R (binding), Julia (native)

Some Industry Contributors in Apache Arrow

ClearCode

BlueYonder

The Center of Applied Data Science

Other Tools

- Hcatalopg
  - HCatalog is a table and storage management layer for Hadoop that enables users with different data processing tools — Pig, MapReduce — to more easily read and write data on the grid.
- OOZIE
  - Oozie is a workflow scheduler system to manage Apache Hadoop jobs..Oozie Workflow jobs are Directed Acyclical Graphs (DAGs) of actions..Oozie Coordinator jobs are recurrent Oozie Workflow jobs triggered by time (frequency) and data availability..Oozie is integrated with the rest of the Hadoop stack supporting several types of Hadoop jobs out of the box (such as Java map-reduce, Streaming map-reduce, Pig, Hive, Sqoop and Distcp) as well as system specific jobs (such as Java programs and shell scripts). Oozie is a scalable, reliable and extensible system
- TES
  - A Framework for YARN-based, Data Processing Applications In Hadoop. Apache™ Tez is an extensible framework for building high performance batch and interactive data processing applications, coordinated by YARN in Apache Hadoop. Tez improves the MapReduce paradigm by dramatically improving its speed, while maintaining MapReduce's ability to scale to petabytes of data. Important Hadoop ecosystem projects like Apache Hive and Apache Pig use Apache Tez, as do a growing number of third party data access applications developed for the broader Hadoop ecosystem.
- MAHOUT
  - Apache Mahout(TM) is a **distributed linear algebra framework** and **mathematically expressive Scala DSL** designed to let mathematicians, statisticians, and data scientists quickly *implement their own algorithms*. Apache Spark is the recommended out-of-the-box distributed back-end, *or can be extended to other distributed backends.* Mathematically Expressive Scala DSL. Support for Multiple Distributed Backends (including Apache Spark). Modular Native Solvers for CPU/GPU/CUDA Acceleratio
- CASANDRA
  - The Apache Cassandra database is the right choice when you need scalability and high availability without compromising performance. Linear scalability and proven fault-tolerance on commodity hardware or cloud infrastructure make it the perfect platform for mission-critical data. Cassandra's support for replicating across multiple datacenters is best-in-class, providing lower latency for your users and the peace of mind of knowing that you can survive regional outages.
- Presto
  - Presto is a high performance, distributed SQL query engine for big data

The Center of Applied Data Science

Other Tools

- Chukwa
  - Apache Chukwa is an open source data collection system for monitoring large distributed systems. Apache Chukwa is built on top of the Hadoop Distributed File System (HDFS) and Map/Reduce framework and inherits Hadoop's scalability and robustness. Apache Chukwa also includes a flexible and powerful toolkit for displaying, monitoring and analyzing results to make the best use of the collected data.
- Hue
  - Hue is an open source SQL Cloud Editor for browsing, querying and visualizing data. Its goal is to make self service data querying more widespread in organizations.
- Ambari
  - The Apache Ambari project is aimed at making Hadoop management simpler by developing software for provisioning, managing, and monitoring Apache Hadoop clusters. Ambari provides an intuitive, easy-to-use Hadoop management web UI backed by its RESTful APIs.
- Solr
  - Solr is highly reliable, scalable and fault tolerant, providing distributed indexing, replication and load-balanced querying, automated failover and recovery, centralized configuration and more. Solr powers the search and navigation features of many of the world's largest internet sites.
- AVRO
  - data serialization system
- Thrift
  - The Apache Thrift software framework, for scalable cross-language services development, combines a software stack with a code generation engine to build services that work efficiently and seamlessly between C++, Java, Python, PHP, Ruby, Erlang, Perl, Haskell, C#, Cocoa, JavaScript, Node.js, Smalltalk, OCaml and Delphi and other languages.

The Center of Applied Data Science

QUIZ 3

https://forms.gle/YpH7McX2y6fZcmii8

The Center of Applied Data Science

Big Data Pipeline

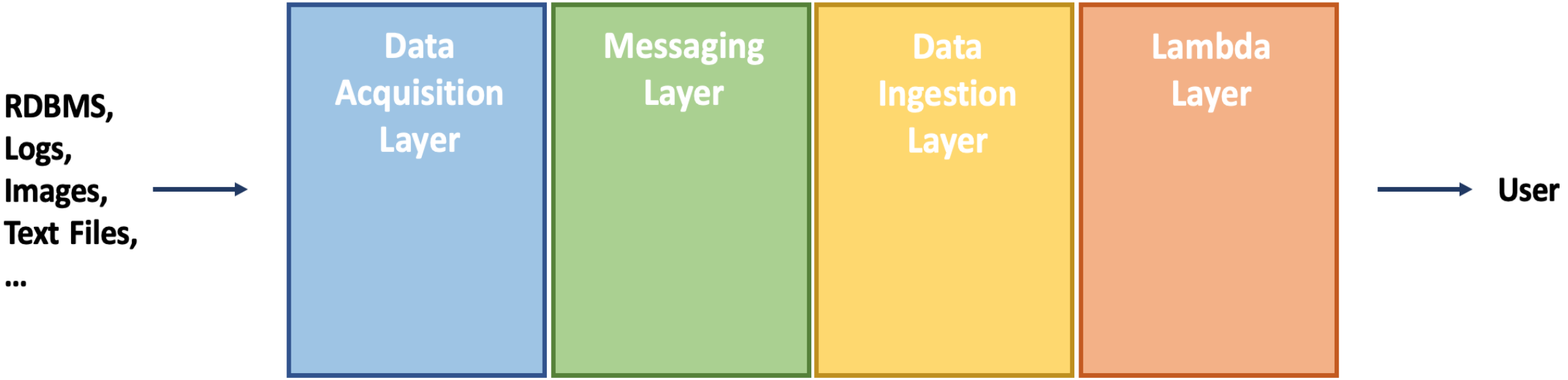
Why Need This?

- Extract information from extensive networking or web logs.
- Store large amounts of unstructured data.
- Process large datasets that are over 100GB in size.
- Analyze multiple large data sources that include both structured and unstructured.

The Center of Applied Data Science

Hadoop Ecosystem

Big Data Pipeline



The Center of Applied Data Science

Hadoop Ecosystem

Big Data Pipeline

Data Acquisition Layer

The layer is where the source data is first landed within the Data Lake.

Messaging Layer

This layer supports massive message streams from various sources.

Data Ingestion Layer

This layer will perform required transformation for ingesting the data into Lambda Layer.

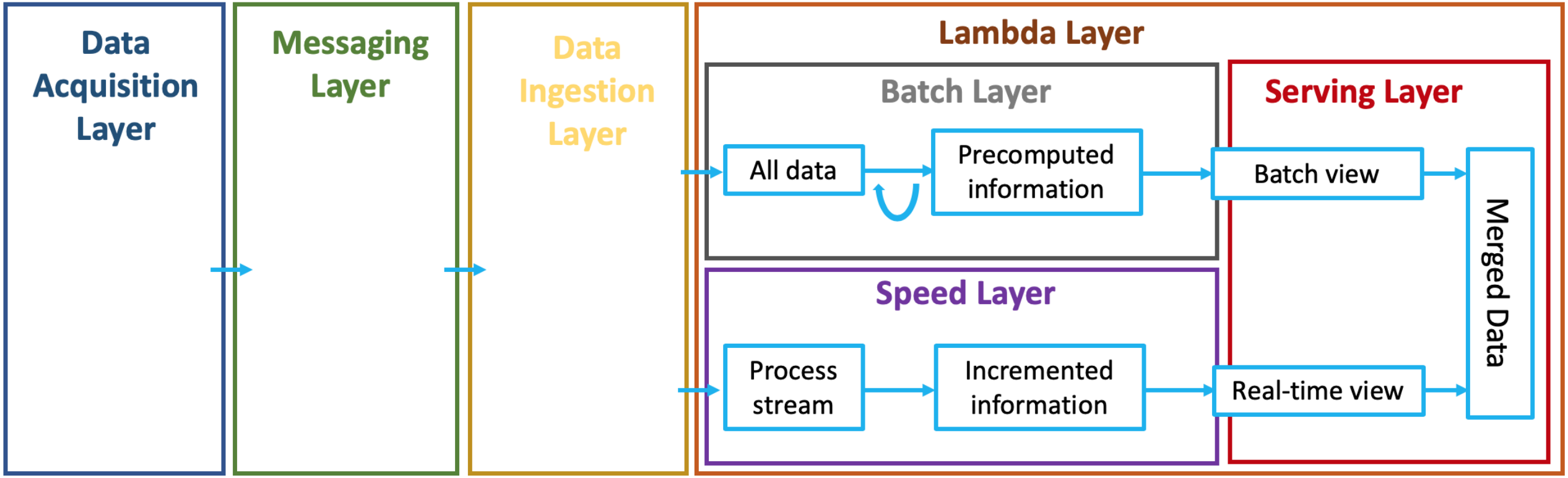
Lambda Layer

This layer handles massive amounts of data by taking into account of both batch and stream-processing methods.

The Center of Applied Data Science

Hadoop Ecosystem

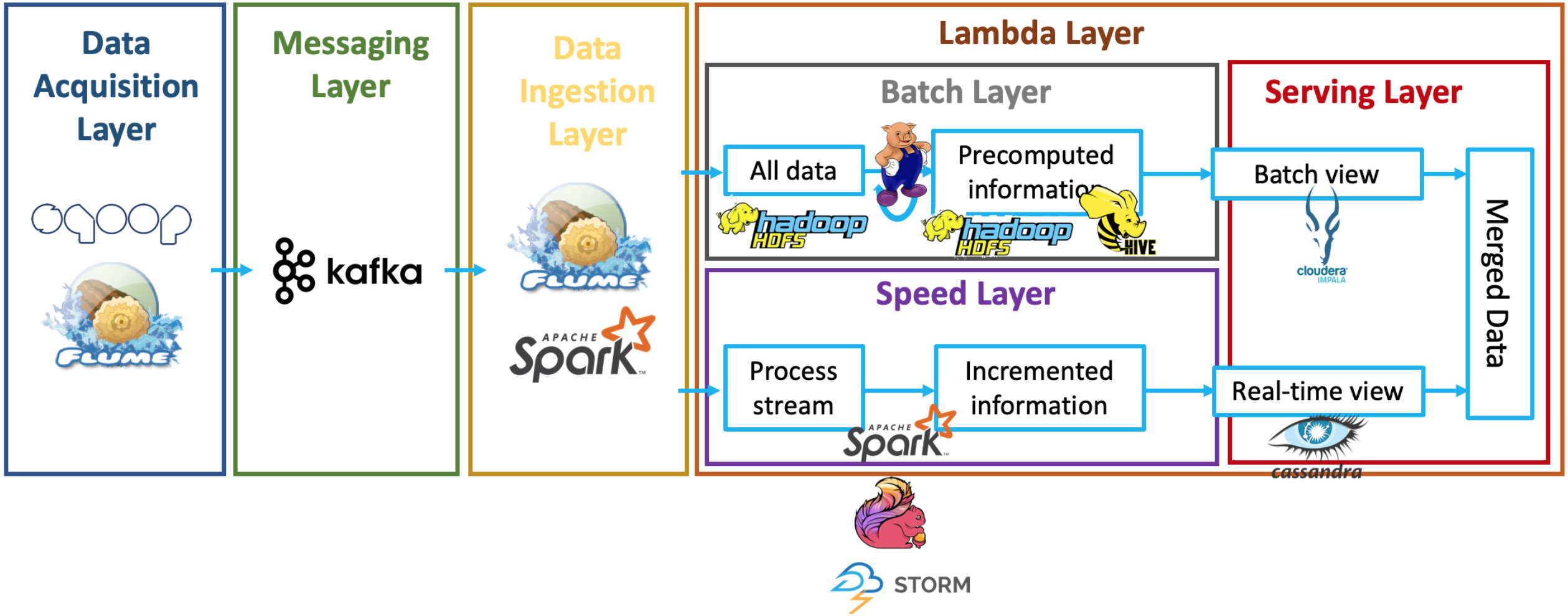
Big Data Pipeline



The Center of Applied Data Science

Hadoop Ecosystem

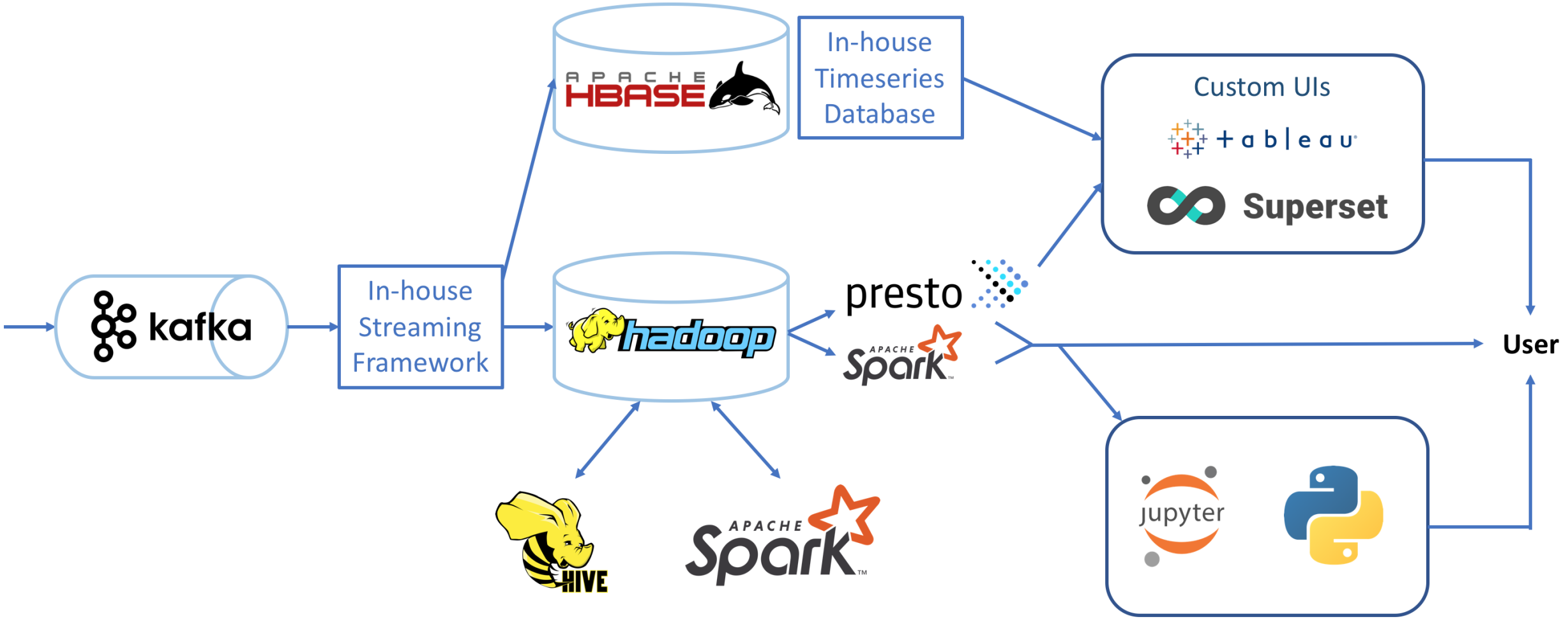
Big Data Pipeline



The Center of Applied Data Science

Hadoop Ecosystem

Big Data Technology Stack from Tesla



Thank you.

thecads.org

info@thecads.org

The Center of Applied Data Science

thecads.org

thecadsmalaysia

Instructor Evaluation

https://www.research.net/r/BigData_May19