BIG DATA AND BUSINESS ANALYTICS

IT

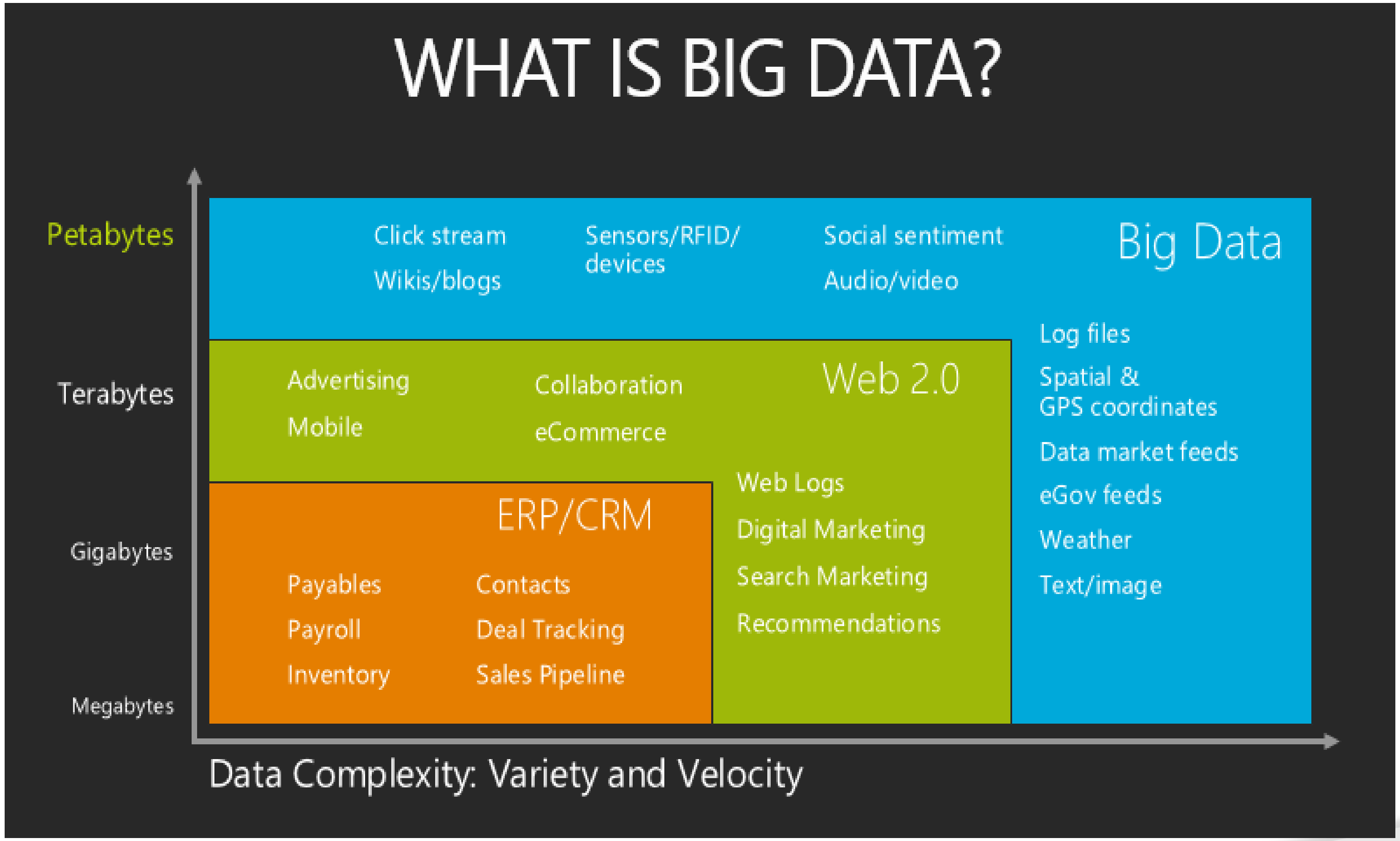
VII SEMESTER

PPT ON
BIG DATA AND BUSINESS ANALYTICS
VII SEM (IARE-R16)

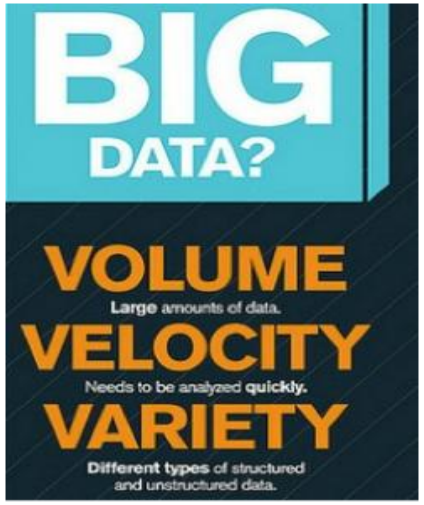
UNIT 1
INTRODUCTION TO BIG DATA

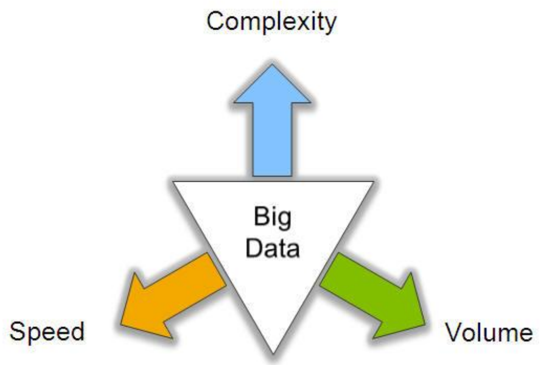
Big Data

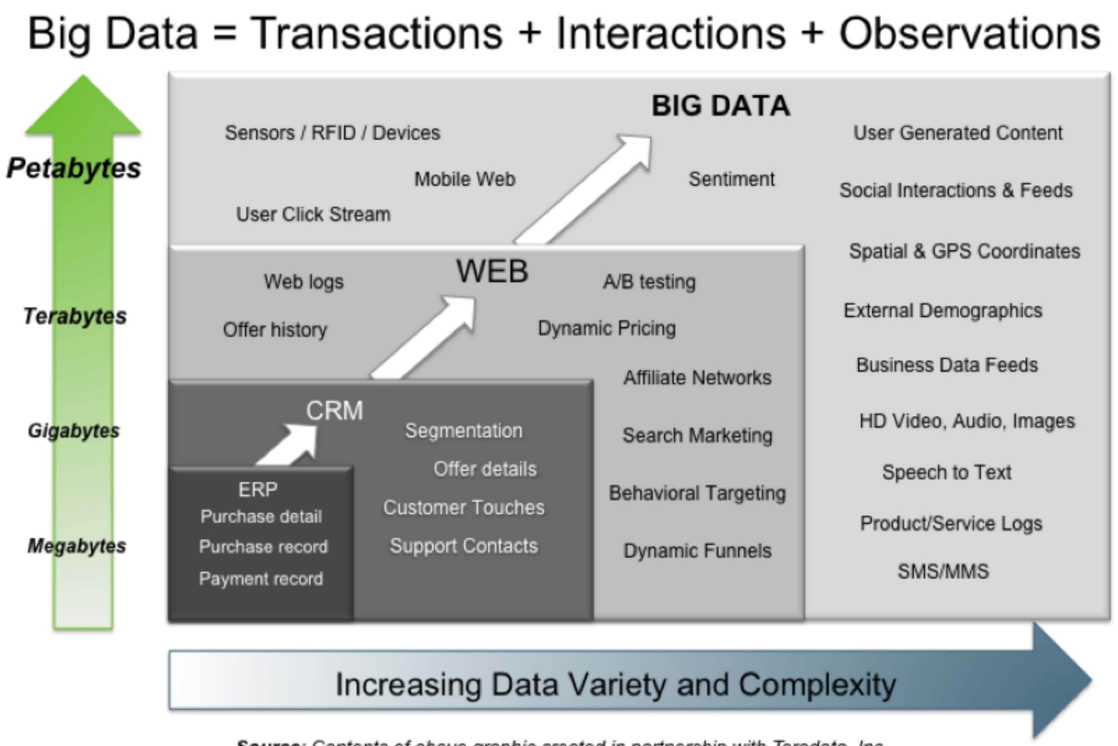
- Big data is the term for a collection of data sets so large and complex that it becomes difficult to process using on-hand database management tools or traditional data processing applications.
- The challenges include capture, curation, storage, search, sharing, transfer, analysis, and visualization.
- The trend to larger data sets is due to the additional information derivable from analysis of a single large set of related data, as compared to separate smaller sets with the same total amount of data, allowing correlations to be found to "spot business trends, determine quality of research, prevent diseases, link legal citations, combat crime, and determine real-time roadway traffic conditions.”


WHAT IS BIG DATA?
Petabytes
Terabytes
Gigabytes
Megabytes
Click stream
Wikis/blogs
Sensors/RFID/ devices
Social sentiment
Audio/video
Big Data
Log files
Spatial & GPS coordinates
Data market feeds
eGov feeds
Weather
Text/image
Advertising
Mobile
Collaboration
eCommerce
Web 2.0
Web Logs
Digital Marketing
Search Marketing
Recommendations
ERP/CRM
Payables
Payroll
Inventory
Contacts
Deal Tracking
Sales Pipeline
Data Complexity: Variety and Velocity

Big Data: 3V's


BIG
DATA?
VOLUME
Large amounts of data.
VELOCITY
Needs to be analyzed quickly.
VARIETY
Different types of structured and unstructured data.

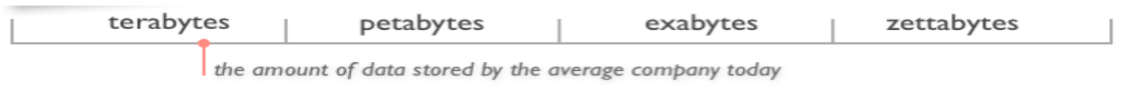


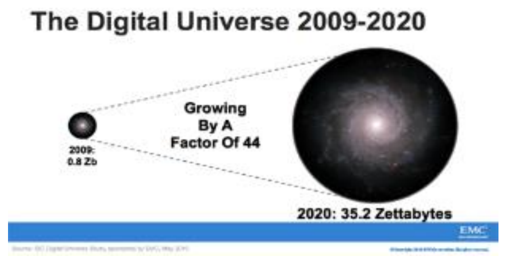


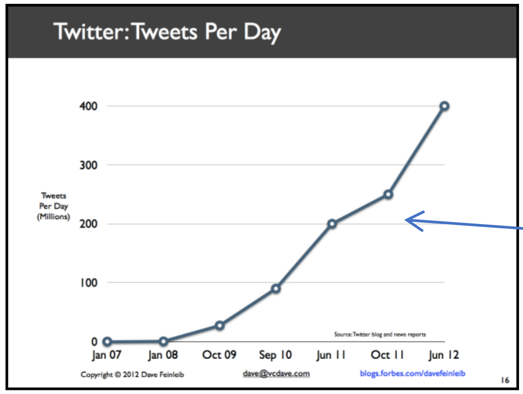
***Source**: Contents of above graphic created in partnership with Teradata, Inc.*

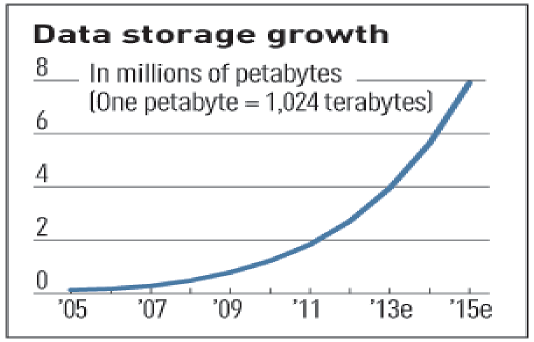
Volume (Scale)

- **Data Volume**
  - 44x increase from 2009 2020
  - From 0.8 zettabytes to 35zb
- Data volume is increasing









Exponential increase in collected/generated data

12+ TBs
of tweet data every day

? TBs of data every day

25+ TBs of log data every day

30 billion RFID tags today (1.3B in 2005)

4.6 billion camera phones world wide

100s of millions of GPS enabled devices sold annually

76 million smart meters in 2009… 200M by 2014

2+ billion people on the Web by end 2011

CERN’s Large Hydron Collider (LHC) generates 15 PB a year

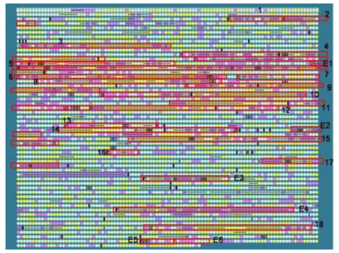
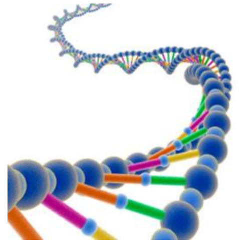
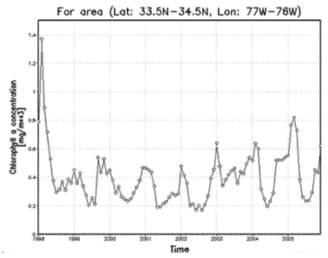
The Earthscope

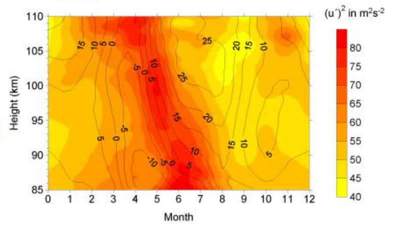
The Earthscope is the world's largest science project. Designed to track North America's geological evolution, this observatory records data over 3.8 million square miles, amassing 67 terabytes of data. It analyzes seismic slips in the San Andreas fault, sure, but also the plume of magma underneath Yellowstone and muchmore.(http://www.msnbc.msn.com/id/44363598/ns/technology_and_science-future_of_technology/#.TmetOdQ--ul)

1
Annual budget: $25,000,000
Construction cost: $197,000,000
Staff: 110
Physical size: 3.8 million square miles
Scientific utility: 10
WIIFY: 10
Wow factor: 10

Variety (Complexity)

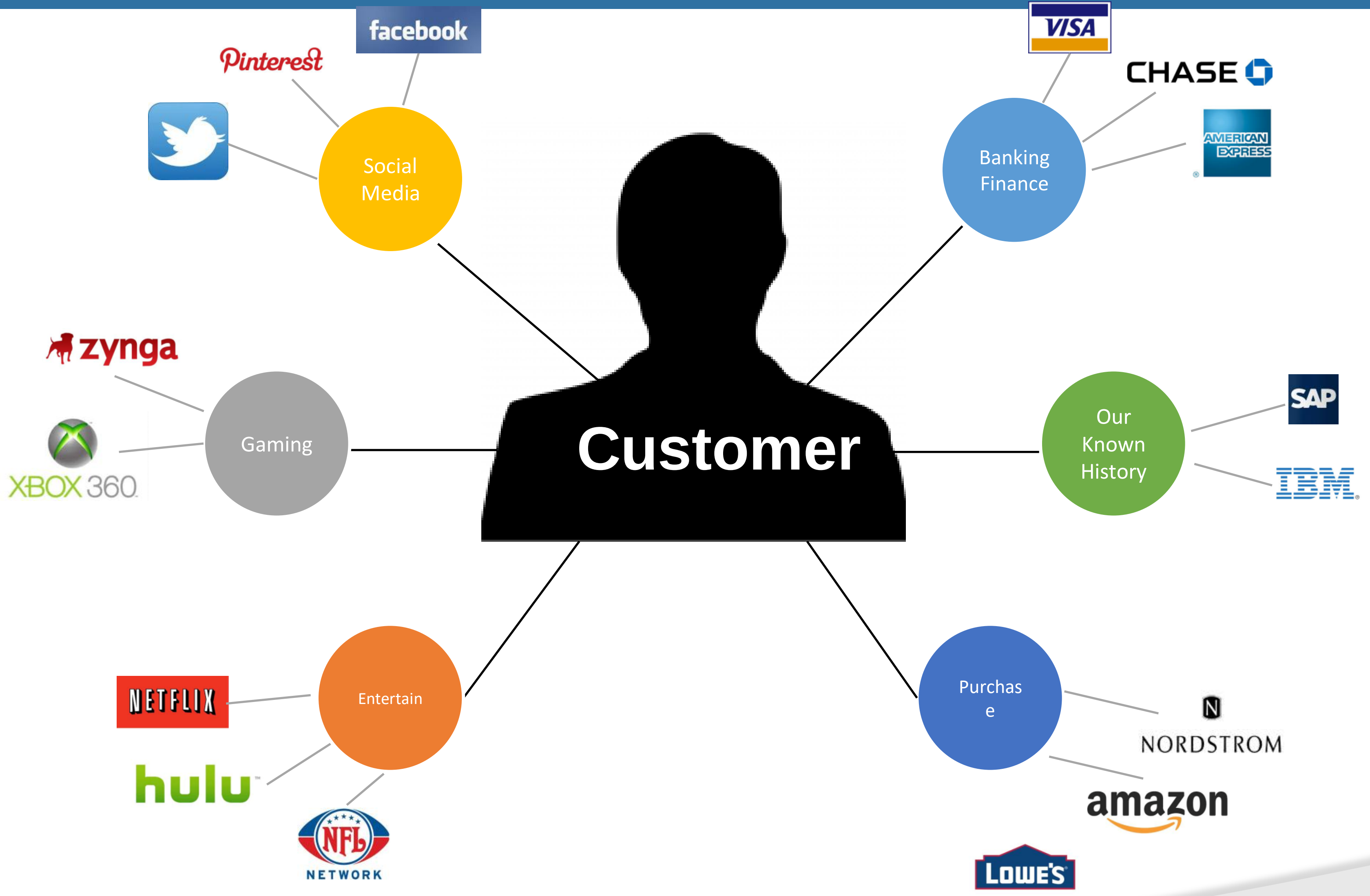
- Relational Data (Tables/Transaction/Legacy Data)
- Text Data (Web)
- Semi-structured Data (XML)
- Graph Data
  - Social Network, Semantic Web (RDF), …
- Streaming Data
  - You can only scan the data once
- A single application can be generating/collecting many types of data
- Big Public Data (online, weather, finance, etc)




For area (Lat: 33.5N–34.5N, Lon: 77W–76W)
Chlorophyll a concentration [mg/m**3]
Time



To extract knowledge➔ all these types of data need to linked together

A Single View to the Customer



Velocity (Speed)

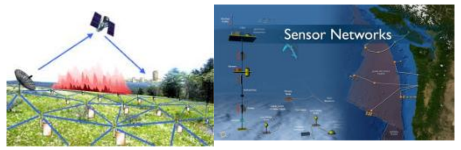
- Data is begin generated fast and need to be processed fast
- Online Data Analytics
- Late decisions ➔ missing opportunities
- **Examples**
  - **E-Promotions:** Based on your current location, your purchase history, what you like ➔ send promotions right now for store next to you
  - **Healthcare monitoring:** sensors monitoring your activities and body ➔ any abnormal measurements require immediate reaction

Real-time/Fast Data

Social media and networks
(all of us are generating data)

Scientific instruments
(collecting all sorts of data)

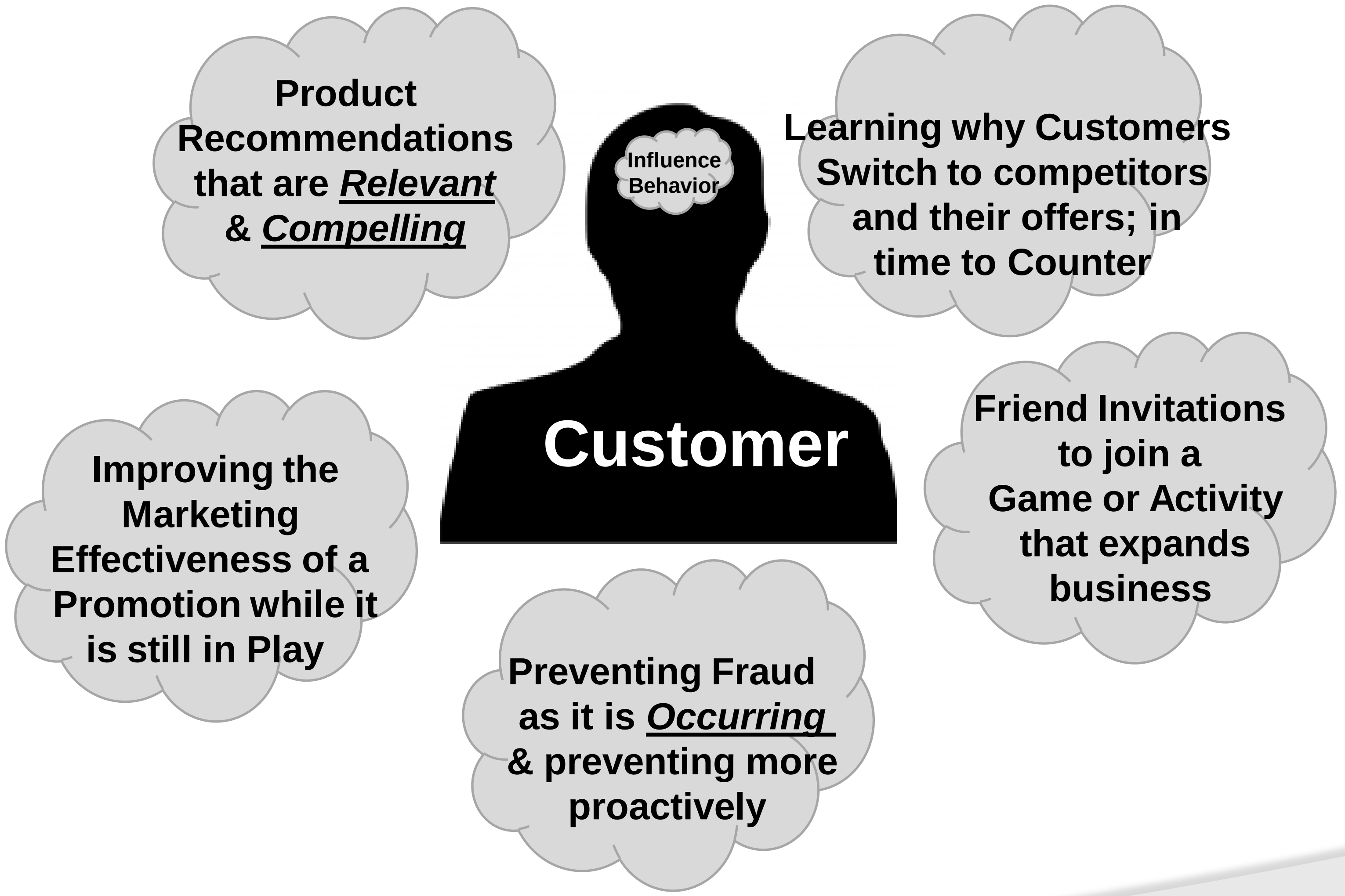
Mobile devices
(tracking all objects all the time)



Sensor technology and networks
(measuring all kinds of data)

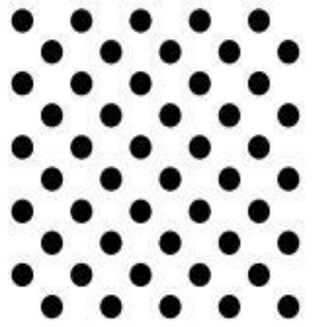
- The progress and innovation is no longer hindered by the ability to collect data
- But, by the ability to manage, analyze, summarize, visualize, and discover knowledge from the collected data in a timely manner and in a scalable fashion

Real-Time Analytics/Decision Requirement


Product Recommendations that are *Relevant* & *Compelling*
Influence Behavior
Learning why Customers Switch to competitors and their offers; in time to Counter
Customer
Improving the Marketing Effectiveness of a Promotion while it is still in Play
Friend Invitations to join a Game or Activity that expands business
Preventing Fraud as it is *Occurring* & preventing more proactively

Some Make it 4V's

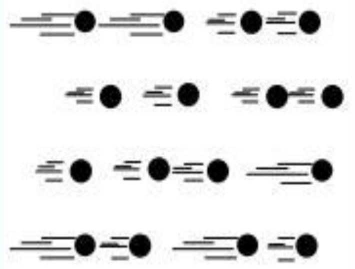
Volume



Data at Rest

Terabytes to exabytes of existing data to process

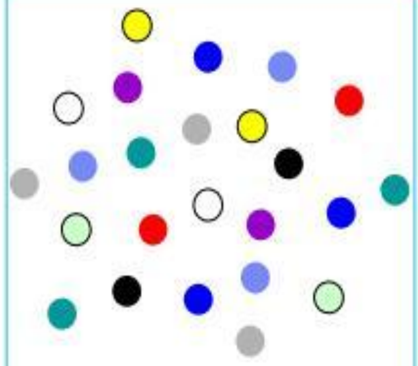
Velocity



Data in Motion

Streaming data, milliseconds to seconds to respond

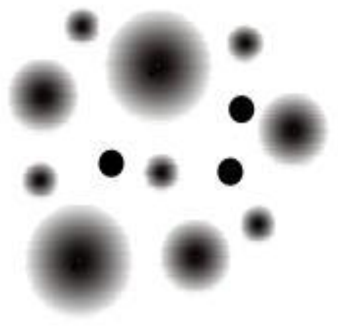
Variety



Data in Many Forms

Structured, unstructured, text, multimedia

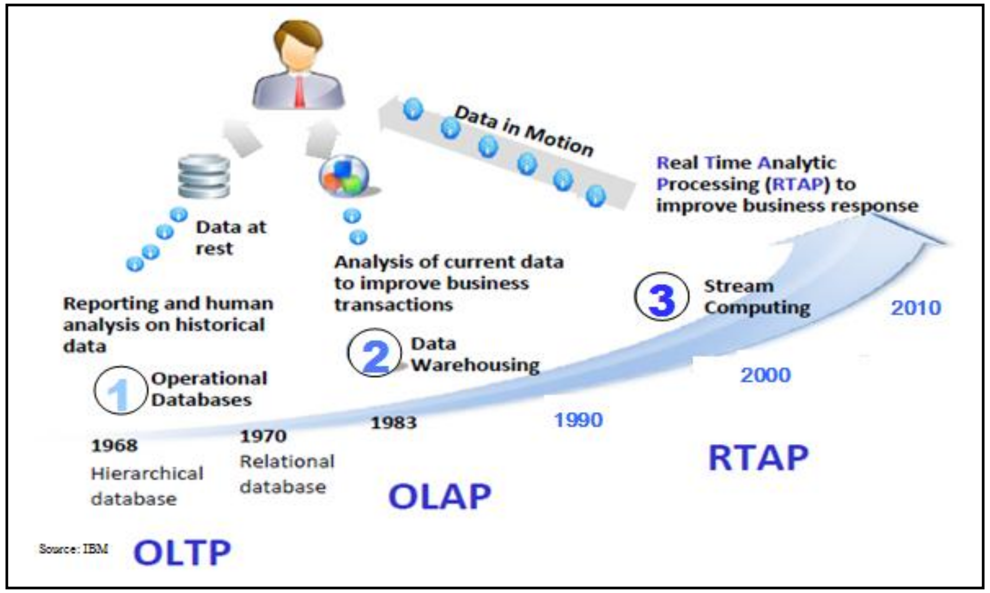
Veracity*



Data in Doubt

Uncertainty due to data inconsistency & incompleteness, ambiguities, latency, deception, model approximations

Harnessing Big Data



- **OLTP:** Online Transaction Processing (DBMSs)
- **OLAP:** Online Analytical Processing (Data Warehousing)
- **RTAP:** Real-Time Analytics Processing (Big Data Architecture & technology)

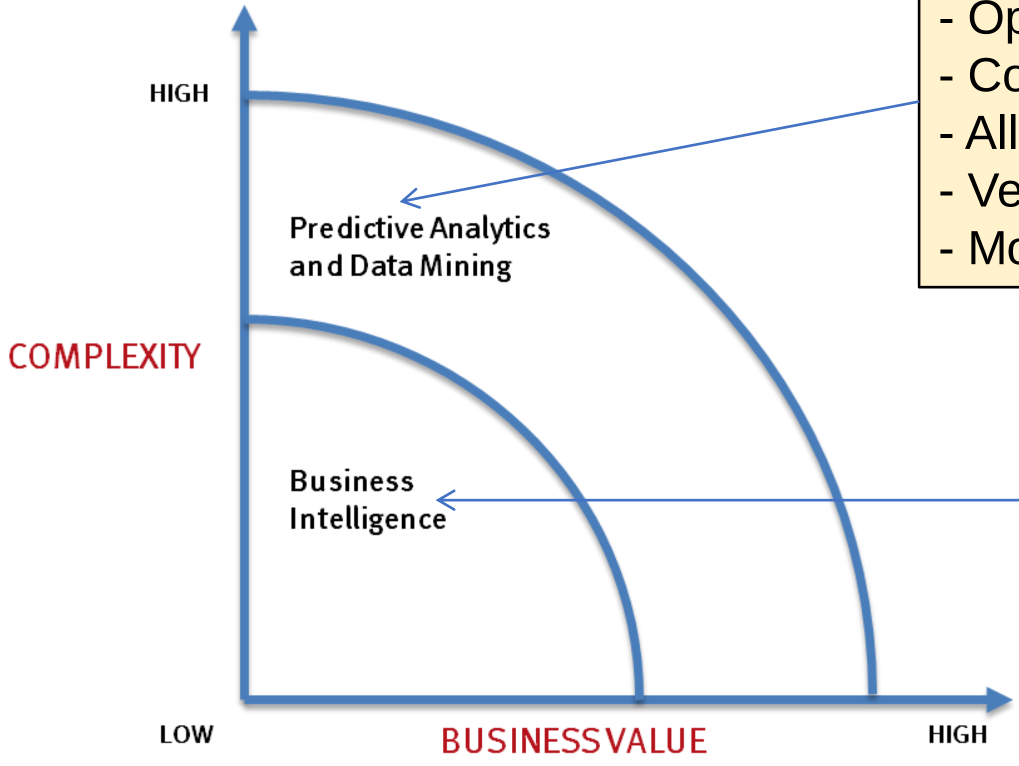
The Model Has Changed…

The Model of Generating/Consuming Data has Changed

Old Model: Few companies are generating data, all others are consuming data

New Model: all of us are generating data, and all of us are consuming data

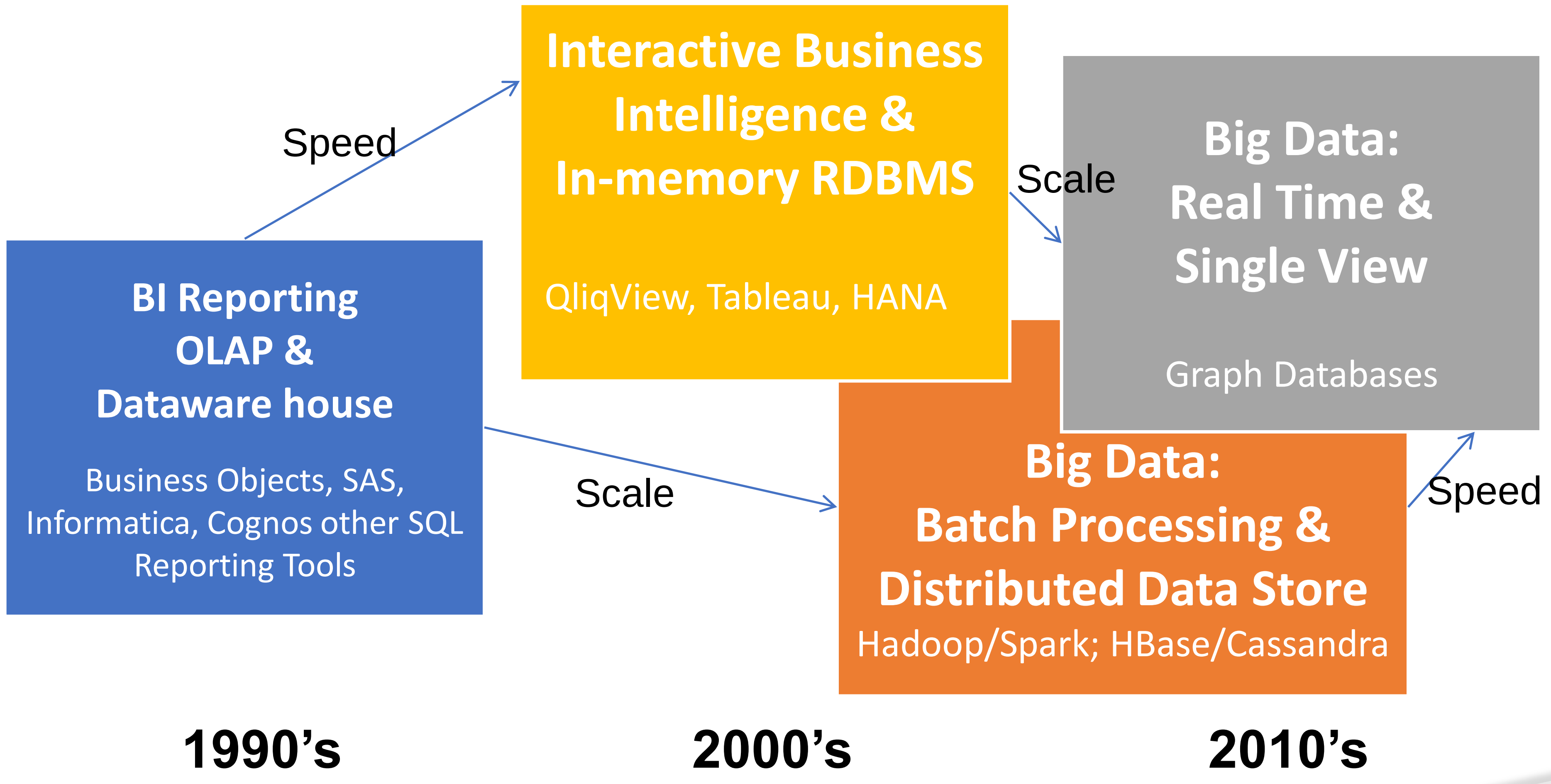
What's driving Big Data



- Optimizations and predictive analytics
- Complex statistical analysis
- All types of data, and many sources
- Very large datasets
- More of a real-time

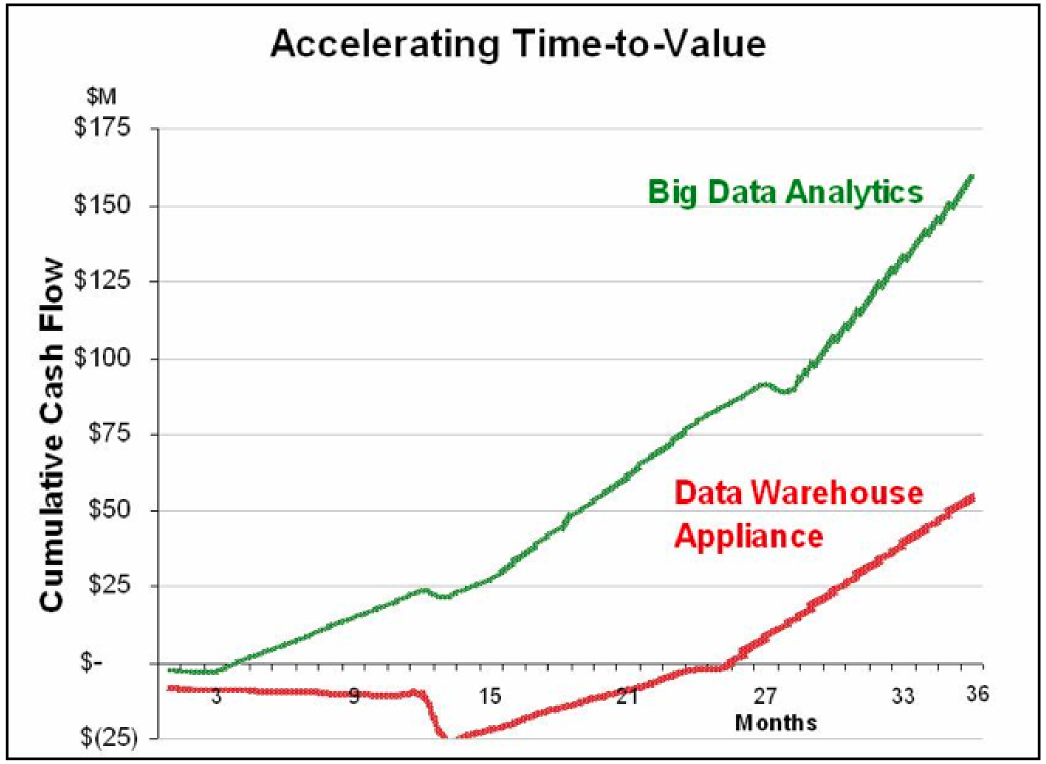
- Ad-hoc querying and reporting
- Data mining techniques
- Structured data, typical sources
- Small to mid-size datasets

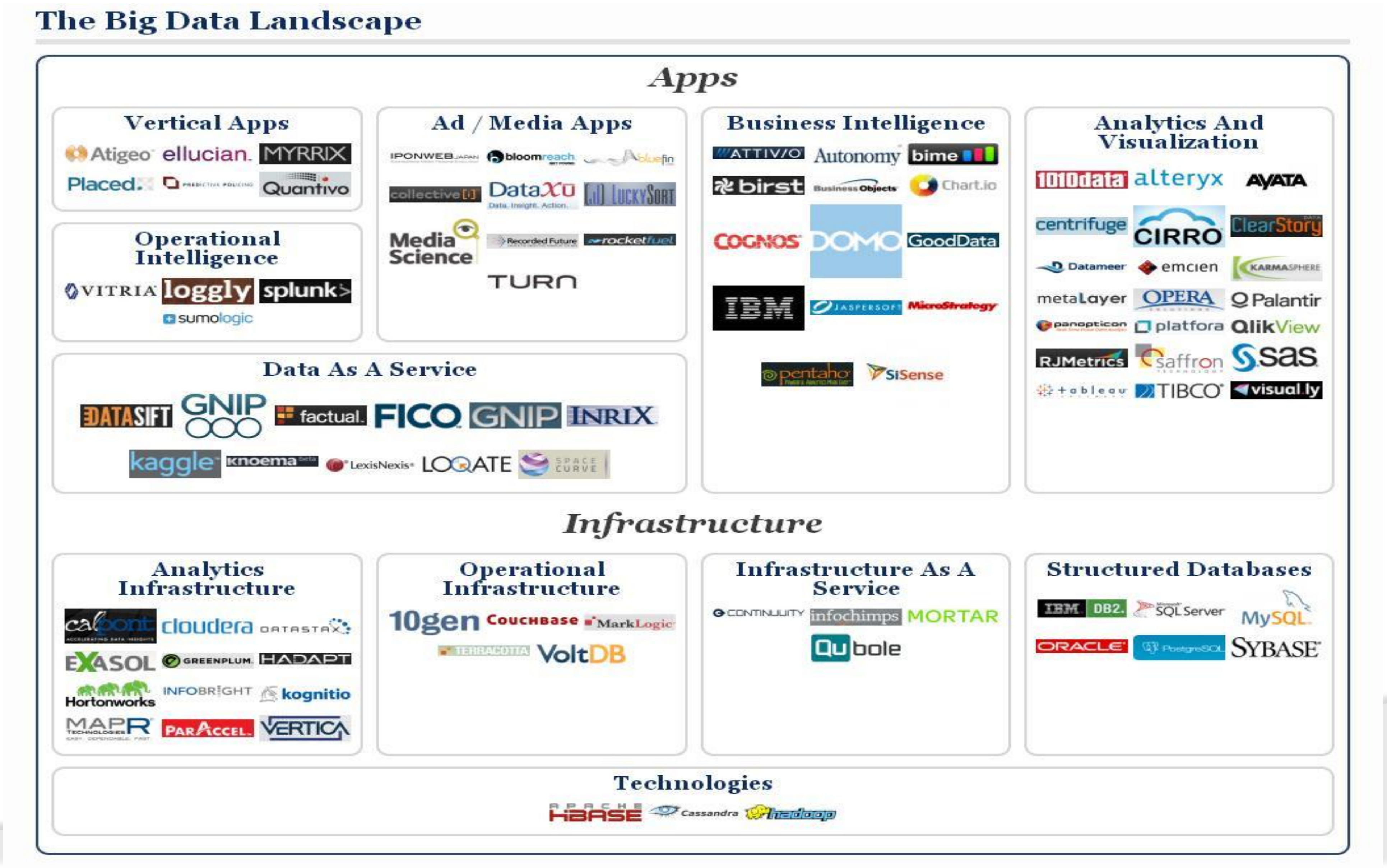
The Evolution of Business Intelligence


Speed
Interactive Business Intelligence & In-memory RDBMS
QliqView, Tableau, HANA
Scale
Big Data: Real Time & Single View
Graph Databases
BI Reporting OLAP & Dataware house
Business Objects, SAS, Informatica, Cognos other SQL Reporting Tools
Scale
Big Data: Batch Processing & Distributed Data Store
Hadoop/Spark; HBase/Cassandra
Speed
1990's
2000's
2010's

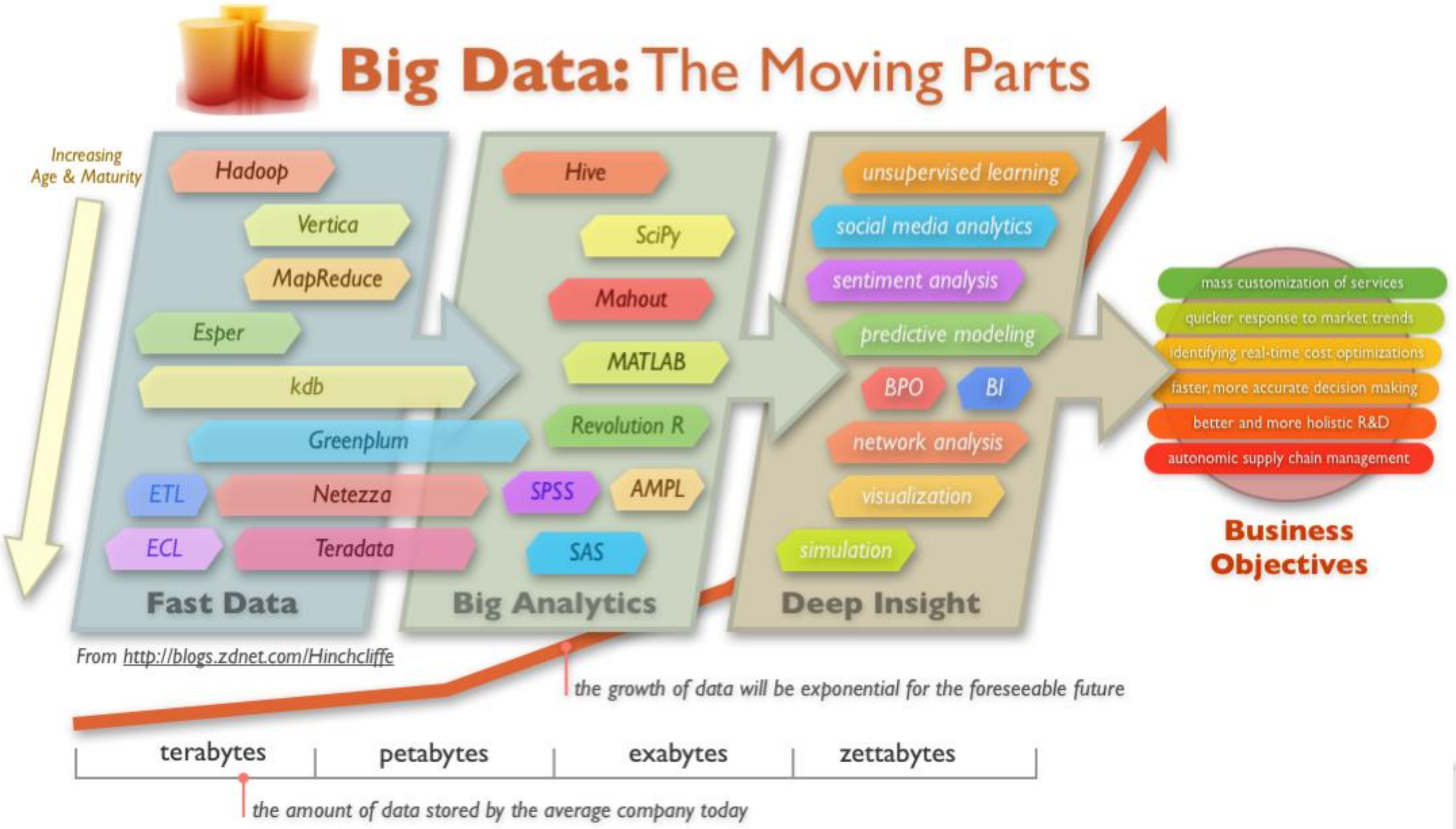
Big Data Analytics

- Big data is more real-time in nature than traditional DW applications
- Traditional DW architectures (e.g. Exadata, Teradata) are not well-suited for big data apps
- Shared nothing, massively parallel processing, scale out architectures are well-suited for big data apps



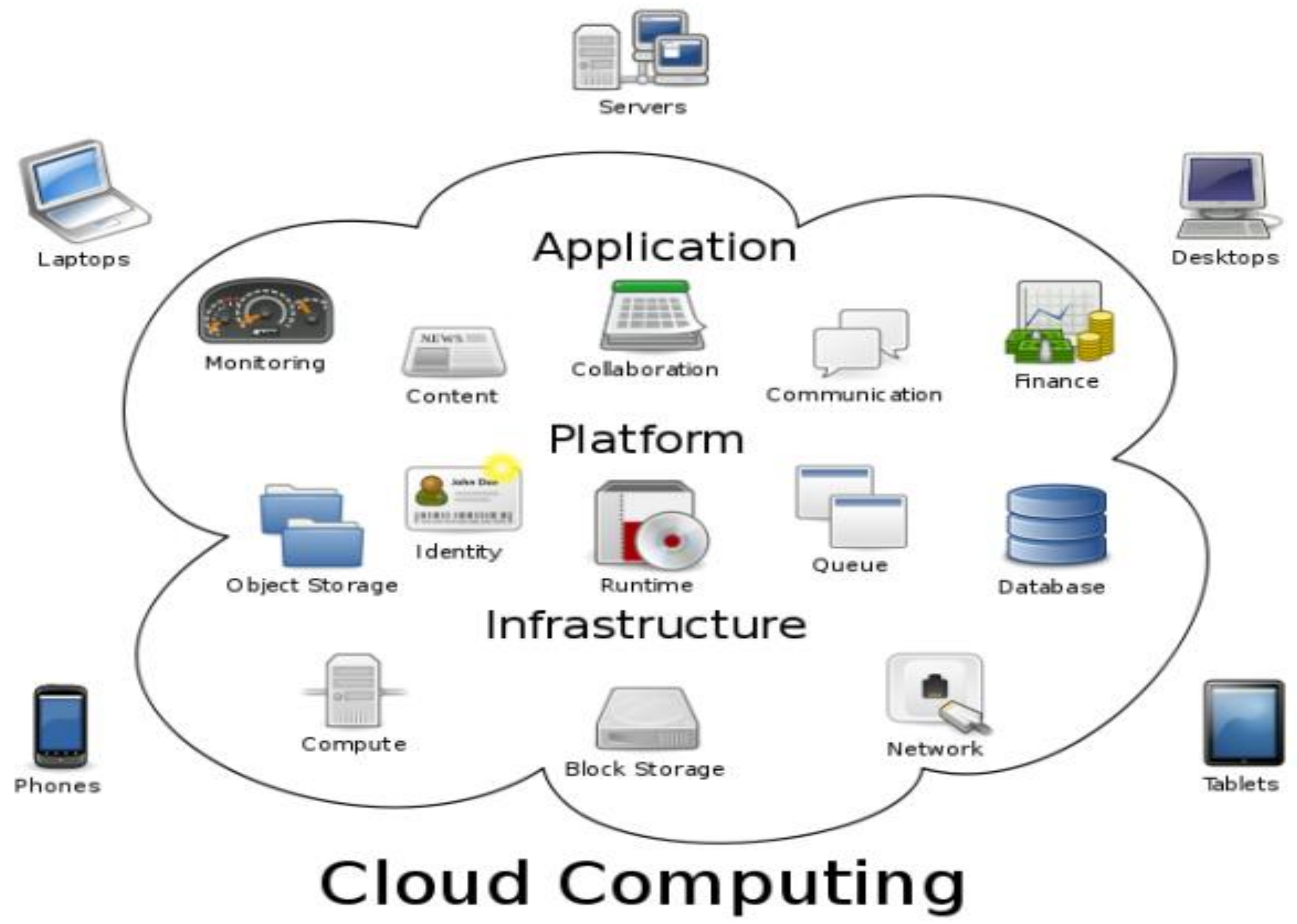

The Big Data Landscape
Apps
Vertical Apps
Atigeo
ellucian
MYRRIX
Placed
Predictive Policing
Quantivo
Operational Intelligence
VITRIA
loggly
splunk
sumologic
Ad / Media Apps
IPONWEB
bloomreach
bluefin
collective[i]
DataXu
LuckySort
Media Science
Recorded Future
rocketfuel
TURN
Data As A Service
DATASIFT
GNIP
factual.
FICO
GNIP
INRIX
kaggle
knoema
LexisNexis
LOQATE
SPACE CURVE
Business Intelligence
ATTIVIO
Autonomy
bime
birst
Business Objects
Chart.io
COGNOS
DOMO
GoodData
IBM
JASPERSOFT
MicroStrategy
pentaho
SiSense
Analytics And Visualization
1010data
alteryx
AYATA
centrifuge
CIRRO
ClearStory
Datameer
emcien
KARMASPHERE
metaLayer
OPERA
Palantir
panopticon
platfora
QlikView
RJMetrics
saffron
sas
tableau
TIBCO
visual.ly
Infrastructure
Analytics Infrastructure
calpont
cloudera
DATASTAX
EXASOL
GREENPLUM
HADAPT
Hortonworks
INFOBRIGHT
kognitio
MAPR
PARACCEL
VERTICA
Operational Infrastructure
10gen
COUCHBASE
MarkLogic
TERRACOTTA
VoltDB
Infrastructure As A Service
CONTINUUITY
infochimps
MORTAR
Qubole
Structured Databases
IBM DB2
SQL Server
MySQL
ORACLE
PostgreSQL
SYBASE
Technologies
APACHE HBASE
Cassandra
hadoop

Big Data Technology


Big Data: The Moving Parts
Increasing Age & Maturity
Hadoop
Vertica
MapReduce
Esper
kdb
Greenplum
ETL
Netezza
ECL
Teradata
Fast Data
Hive
SciPy
Mahout
MATLAB
Revolution R
SPSS
AMPL
SAS
Big Analytics
unsupervised learning
social media analytics
sentiment analysis
predictive modeling
BPO
BI
network analysis
visualization
simulation
Deep Insight
mass customization of services
quicker response to market trends
identifying real-time cost optimizations
faster, more accurate decision making
better and more holistic R&D
autonomic supply chain management
Business Objectives
From http://blogs.zdnet.com/Hinchcliffe
the growth of data will be exponential for the foreseeable future
terabytes
petabytes
exabytes
zettabytes
the amount of data stored by the average company today

Cloud Computing

- IT resources provided as a service
    - Compute, storage, databases, queues
- Clouds leverage economies of scale of commodity hardware
    - Cheap storage, high bandwidth networks & multicore processors
    - Geographically distributed data centers
- Offerings from Microsoft, Amazon, Google

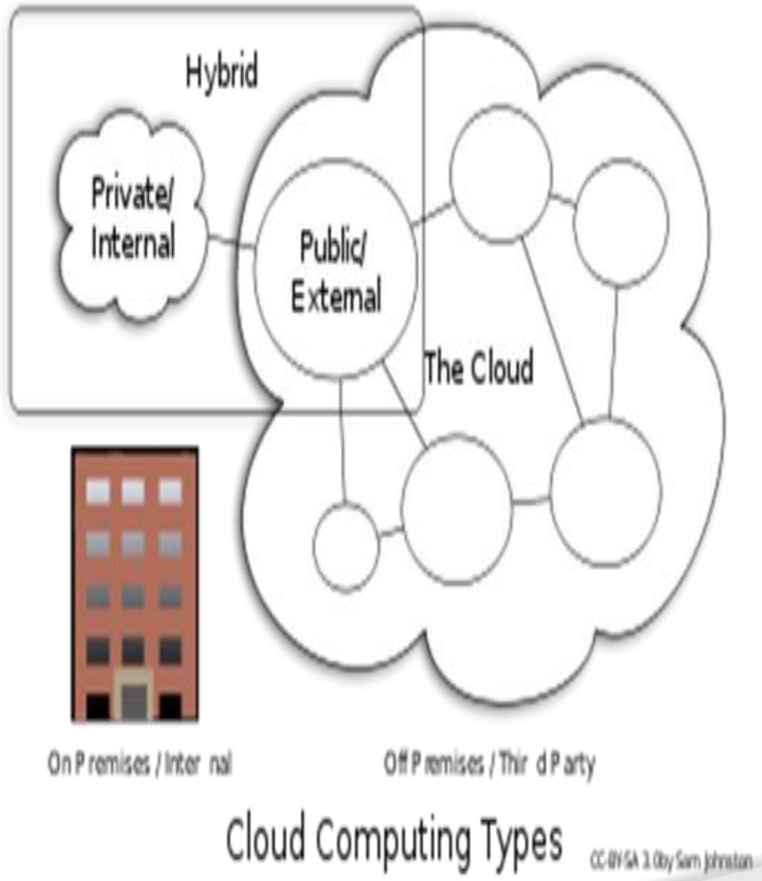

Servers
Laptops
Desktops
Application
Monitoring
Content
Collaboration
Communication
Finance
Platform
Identity
Object Storage
Runtime
Queue
Database
Infrastructure
Compute
Block Storage
Network
Phones
Tablets
Cloud Computing

Benefits

- Cost & management
  - Economies of scale, “out-sourced” resource management
- Reduced Time to deployment
  - Ease of assembly, works “out of the box”
- Scaling
  - On demand provisioning, co-locate data and compute
- Reliability
  - Massive, redundant, shared resources
- Sustainability
  - Hardware not owned

Types of Cloud Computing

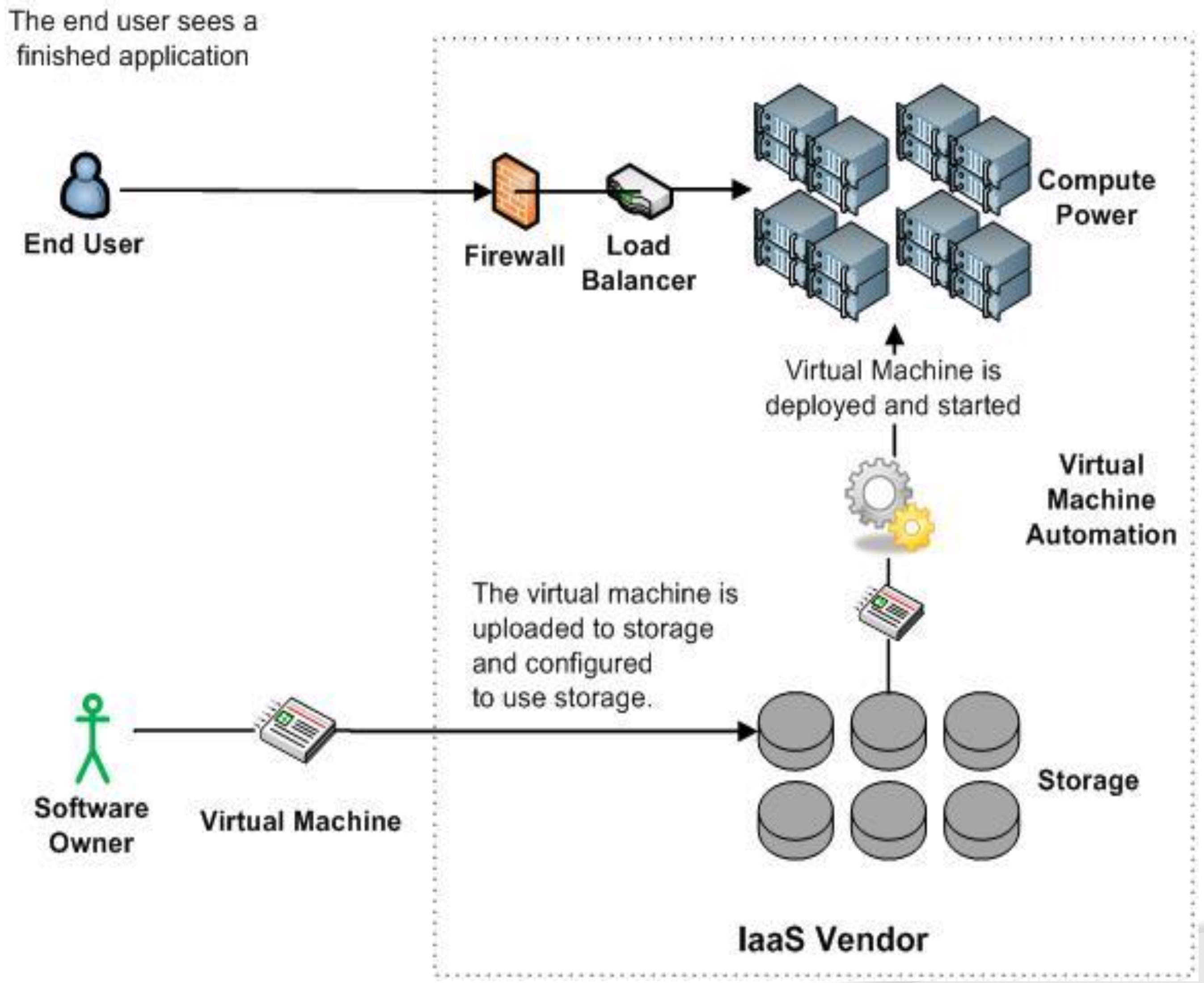
- **Public Cloud**: Computing infrastructure is hosted at the vendor’s premises.
- **Private Cloud**: Computing architecture is dedicated to the customer and is not shared with other organisations.
- **Hybrid Cloud**: Organisations host some critical, secure applications in private clouds. The not so critical applications are hosted in the public cloud
- **Cloud bursting**: the organisation uses its own infrastructure for normal usage, but cloud is used for peak loads.



Classification of Cloud Computing based on Service Provided

- Infrastructure as a service (IaaS)
  - Offering hardware related services using the principles of cloud computing. These could include storage services (database or disk storage) or virtual servers.
  - Amazon EC2, Amazon S3, Rackspace Cloud Servers and Flexiscale.
- Platform as a Service (PaaS)
  - Offering a development platform on the cloud.
  - Google's Application Engine, Microsofts Azure, Salesforce.com's force.com .
- Software as a service (SaaS)
  - Including a complete software offering on the cloud. Users can access a software application hosted by the cloud vendor on pay-per-use basis. This is a well-established sector.
  - Salesforce.coms' offering in the online Customer Relationship Management (CRM) space, Googles gmail and Microsofts hotmail, Google docs.

Infrastructure as a Service (IaaS)


The end user sees a finished application
End User
Firewall
Load Balancer
Compute Power
Virtual Machine is deployed and started
Virtual Machine Automation
The virtual machine is uploaded to storage and configured to use storage.
Software Owner
Virtual Machine
Storage
IaaS Vendor

More Refined Categorization

- Storage-as-a-service
- Database-as-a-service
- Information-as-a-service
- Process-as-a-service
- Application-as-a-service
- Platform-as-a-service
- Integration-as-a-service
- Security-as-a-service
- Management/

  Governance-as-a-service
- Testing-as-a-service
- Infrastructure-as-a-service

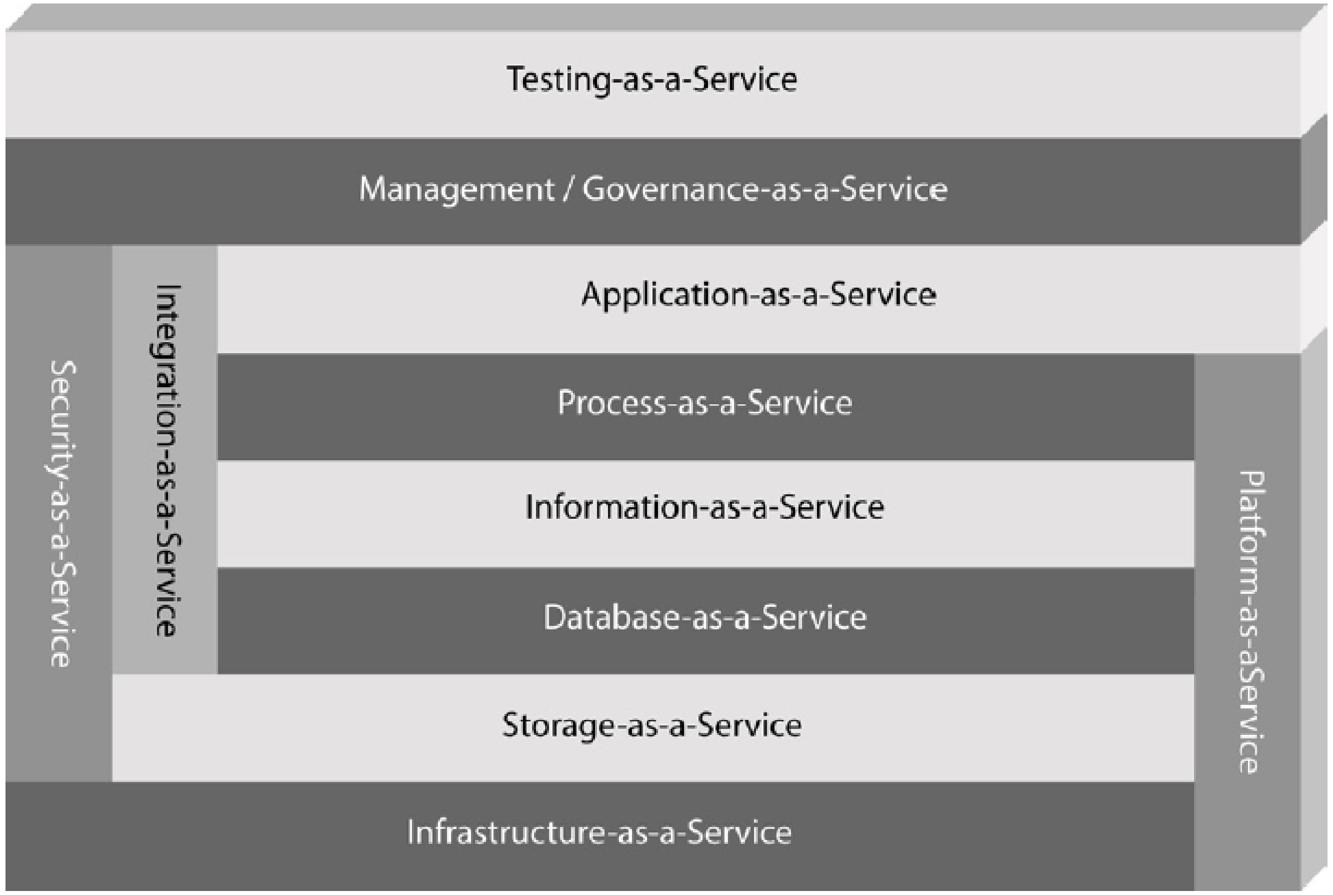


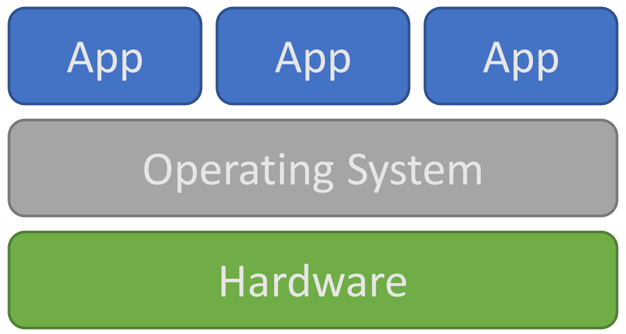
Figure 1: The patterns or categories of cloud computing providers allow you to use a discrete set of services within your architecture.

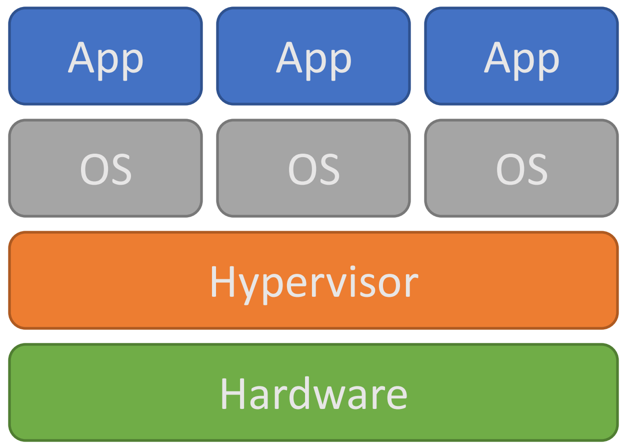
InfoWorld Cloud Computing Deep Dive

Key Ingredients in Cloud Computing

- Service-Oriented Architecture (SOA)
- Utility Computing (on demand)
- Virtualization (P2P Network)
- SAAS (Software As A Service)
- PAAS (Platform AS A Service)
- IAAS (Infrastructure AS A Servie)
- Web Services in Cloud

Enabling Technology: Virtualization





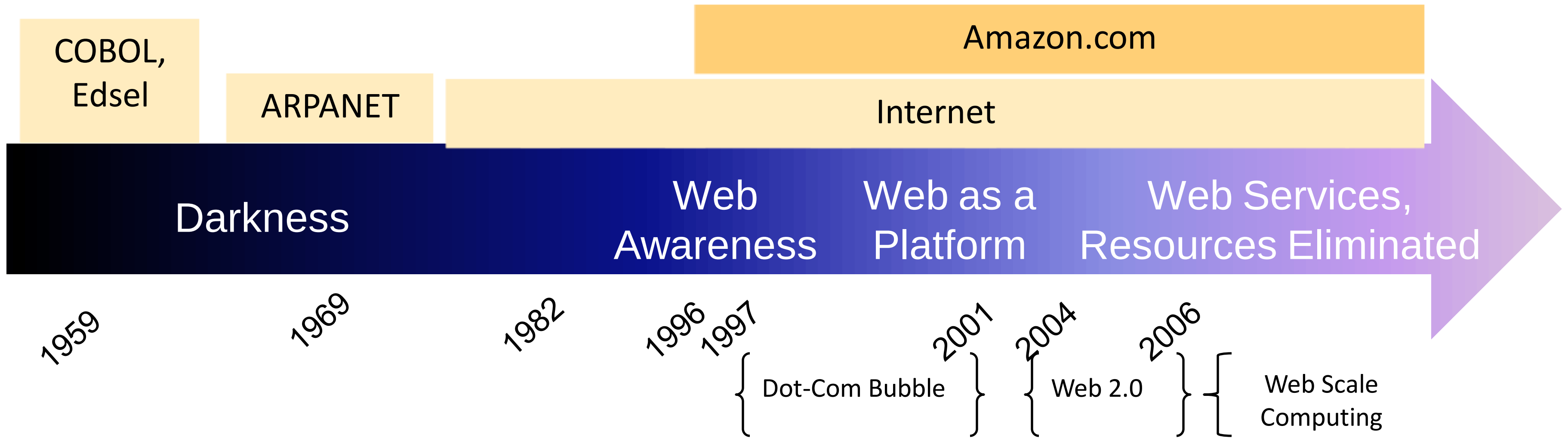
Everything as a Service

- Utility computing = Infrastructure as a Service (IaaS)
  - Why buy machines when you can rent cycles?
  - Examples: Amazon’s EC2, Rackspace
- Platform as a Service (PaaS)
  - Give me nice API and take care of the maintenance, upgrades, …
  - Example: Google App Engine
- Software as a Service (SaaS)
  - Just run it for me!
  - Example: Gmail, Salesforce

Cloud versus cloud

- Amazon Elastic Compute Cloud
- Google App Engine
- Microsoft Azure
- GoGrid
- AppNexus

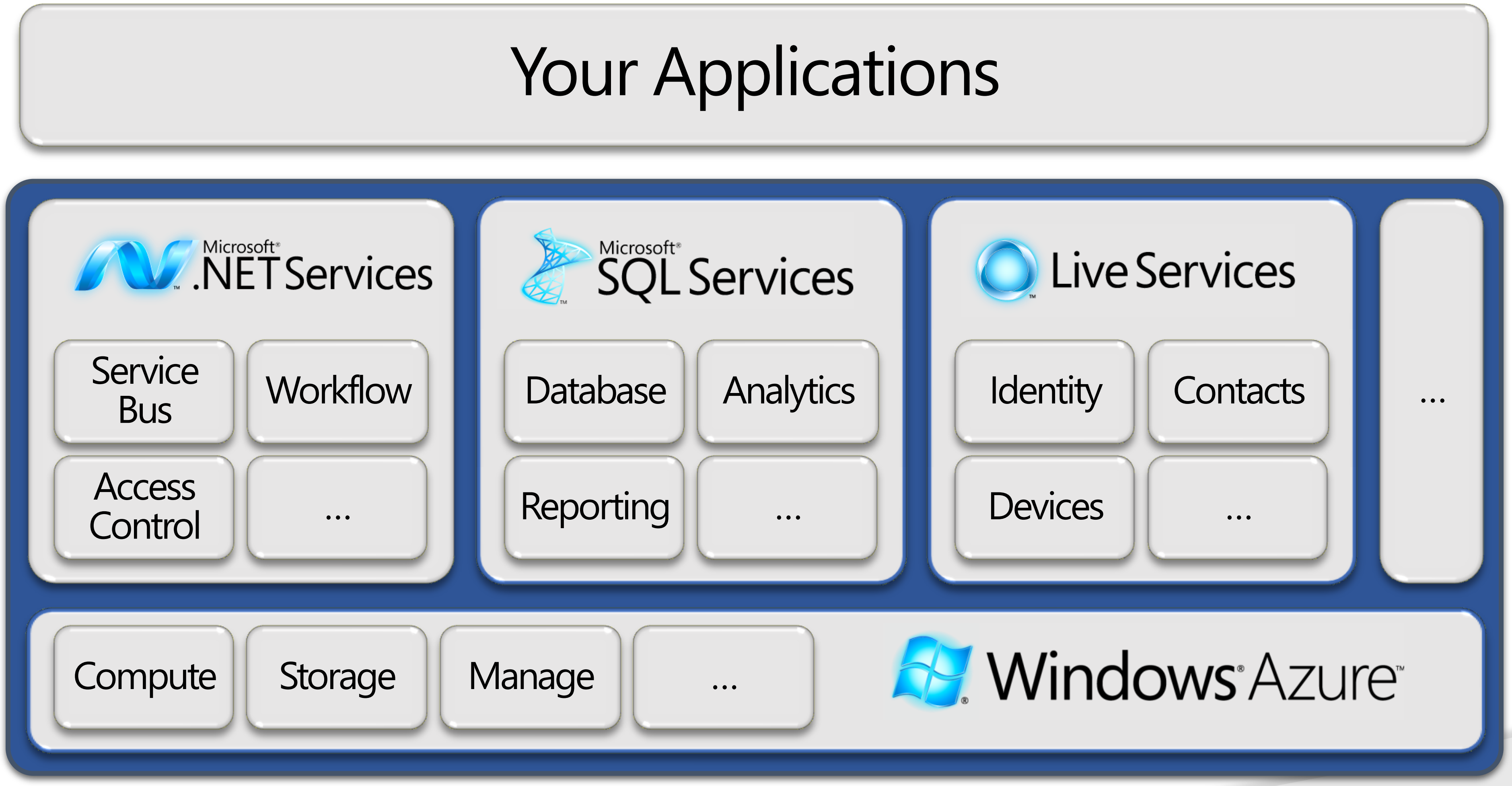
The Obligatory Timeline



AWS

- Elastic Compute Cloud – EC2 (IaaS)
- Simple Storage Service – S3 (IaaS)
- Elastic Block Storage – EBS (IaaS)
- SimpleDB (SDB) (PaaS)
- Simple Queue Service – SQS (PaaS)
- CloudFront (S3 based Content Delivery Network – PaaS)
- Consistent AWS Web Services API

Azure platform offer to developers



Topic 1: Data Analytics & Data Mining

- Exploratory Data Analysis
- Linear Classification (Perceptron & Logistic Regression)
- Linear Regression
- C4.5 Decision Tree
- Apriori
- K-means Clustering
- EM Algorithm
- PageRank & HITS
- Collaborative Filtering

Topic 2: Hadoop/MapReduce Programming & Data Processing

- Architecture of Hadoop, HDFS, and Yarn
- Programming on Hadoop
- Basic Data Processing: Sort and Join
- Information Retrieval using Hadoop
- Data Mining using Hadoop (Kmeans+Histograms)
- Machine Learning on Hadoop (EM)
- Hive/Pig
- HBase and Cassandra

Topic 3: Graph Database and Graph Analytics

- Graph Database (http://en.wikipedia.org/wiki/Graph_database)
  - Native Graph Database (Neo4j)
  - Pregel/Giraph (Distributed Graph Processing Engine)
- Neo4j/Titan/GraphLab/GraphSQL

Cloud Resources

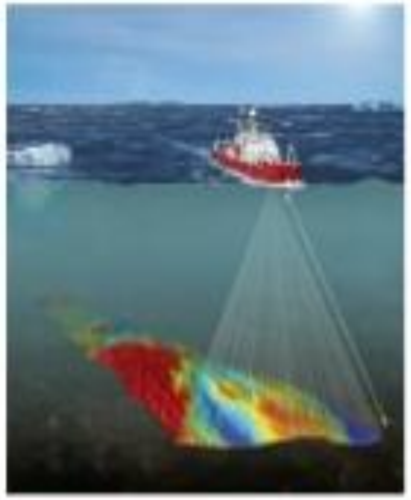
- Hadoop on your local machine
- Hadoop in a virtual machine on your local machine (Pseudo-Distributed on **Ubuntu**)
- Hadoop in the clouds with Amazon EC2

Where data comes from?

- Different data generators in the world
  - Sensors
  - CC cameras
  - Social networks- Facebook, Linkedin,Twitter,Youtube
  - Online shopping
  - Airlines
  - National Climatic Data Center[NCDC]
  - Hospitlity data
  - Newyork Stock Exchange[NYSE]

Social Media & Networks
(All of us are generating data)

Mobile Devices
(Tracking all the objects all the time)



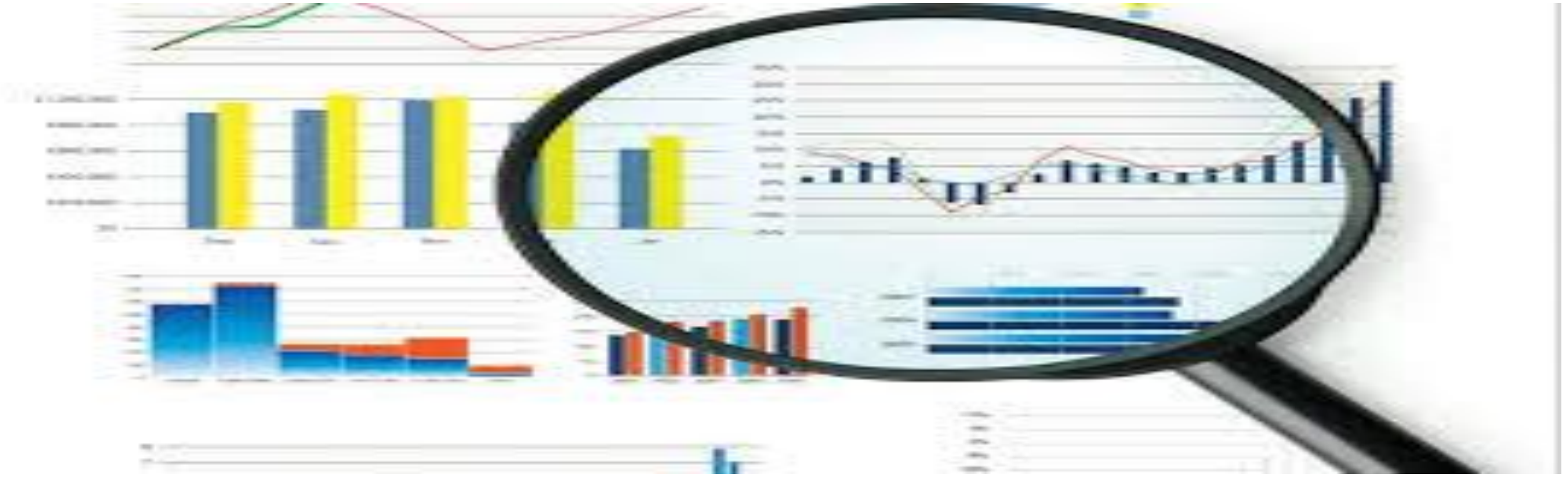
Sensor Technology & Networks
(Measuring all kinds of data)

Scientific Instruments
(Collecting all sorts of data)

Why to store data?

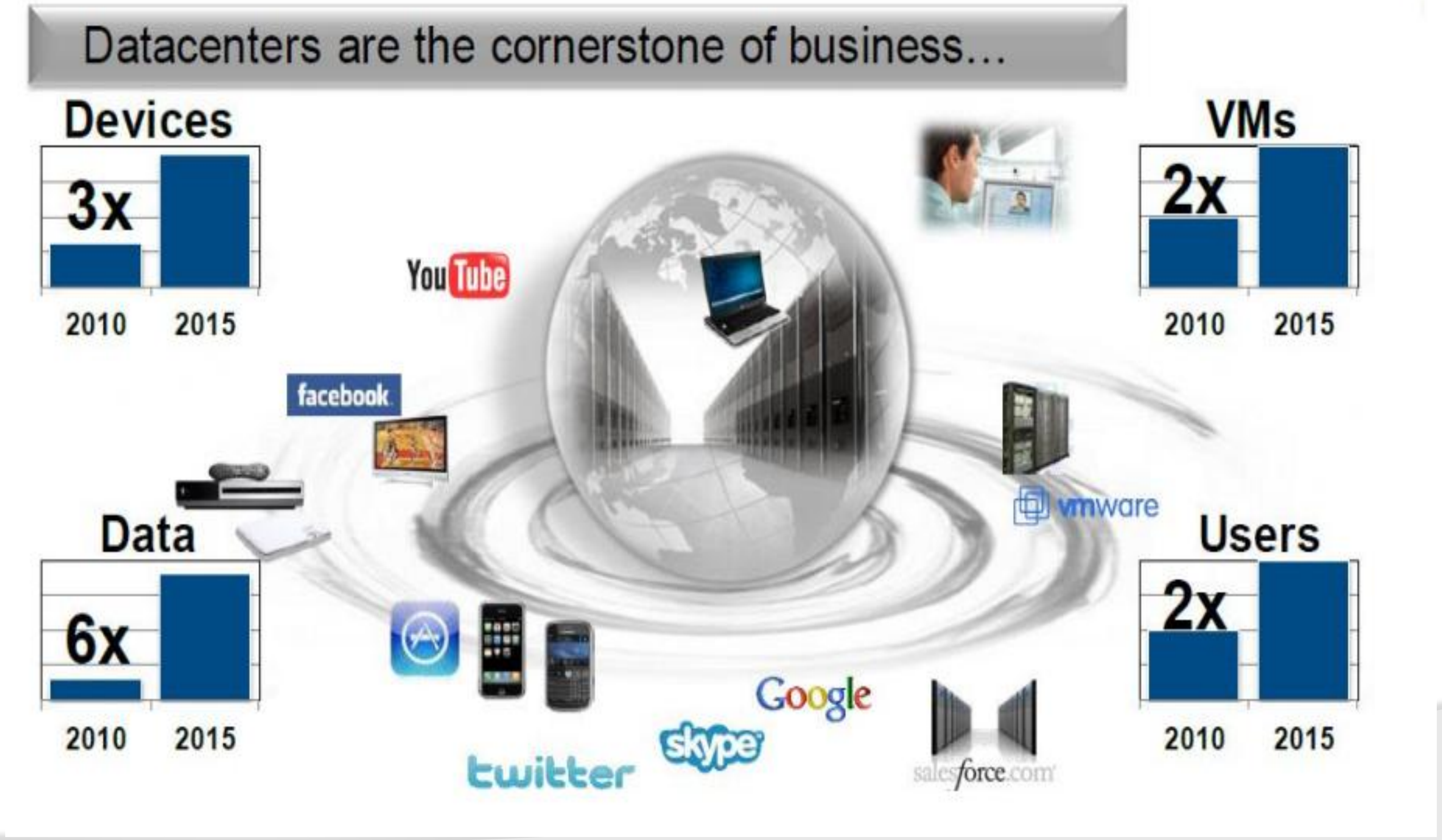
- Data dictates the fate of company, because any business/company do some meaningful analysis for our further estimates about the projects.
- Data analysis is important to business

Data Analysis:

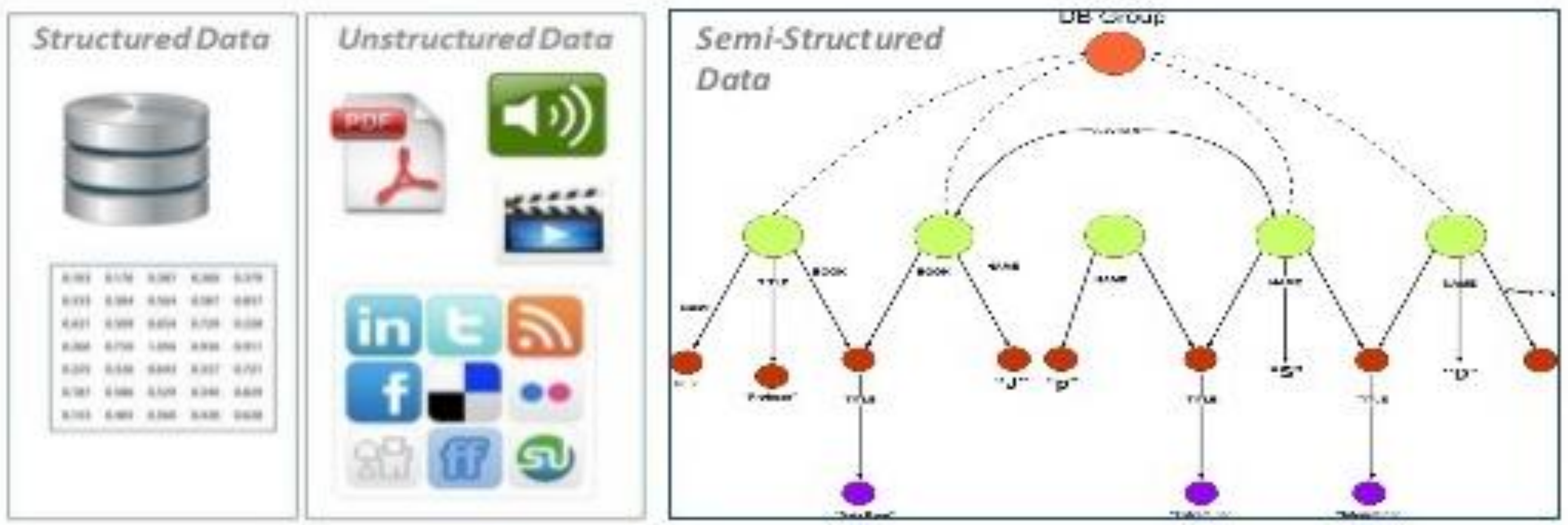


In fact, no business can survive without analyzing available data.

Data Center Concept


Datacenters are the cornerstone of business…
Devices
3x
2010
2015
VMs
2x
2010
2015
Data
6x
2010
2015
Users
2x
2010
2015
YouTube
facebook
vmware
Google
skype
twitter
salesforce.com

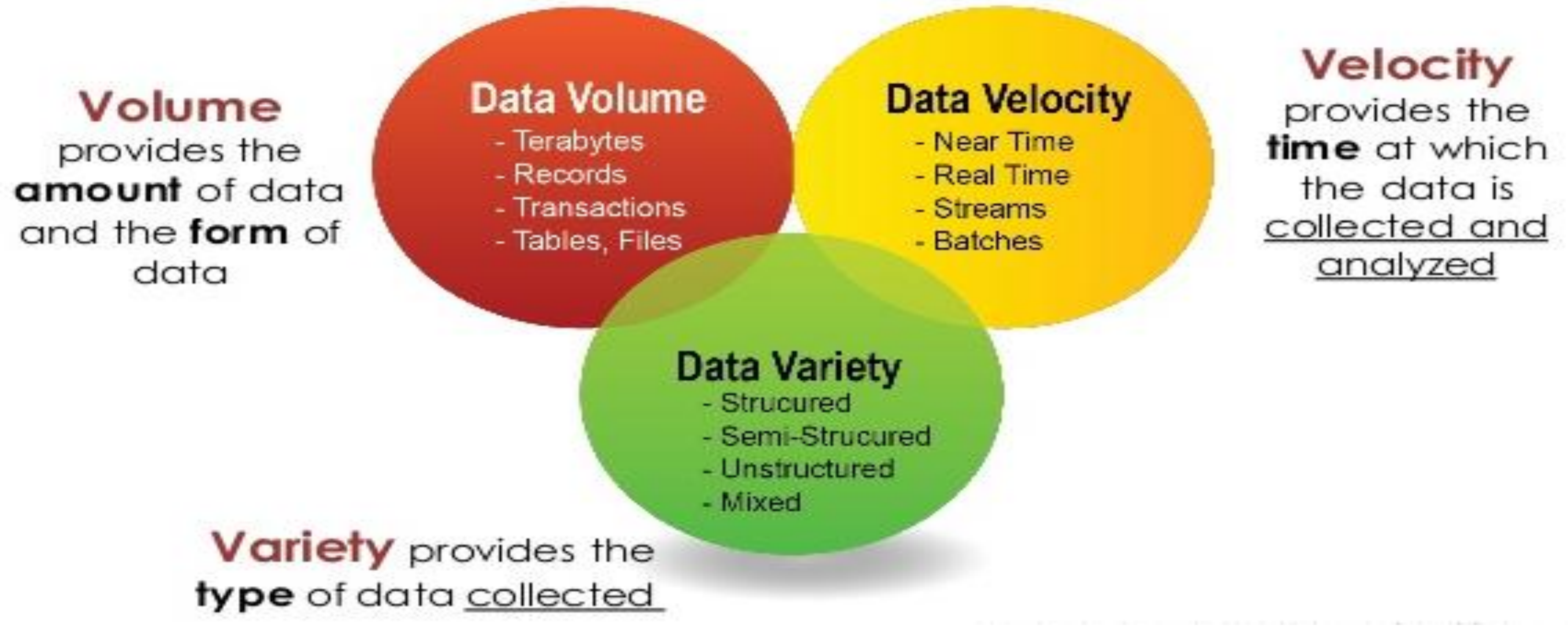
Classification of big data



Traditional RDBMS deals with only Structured data.

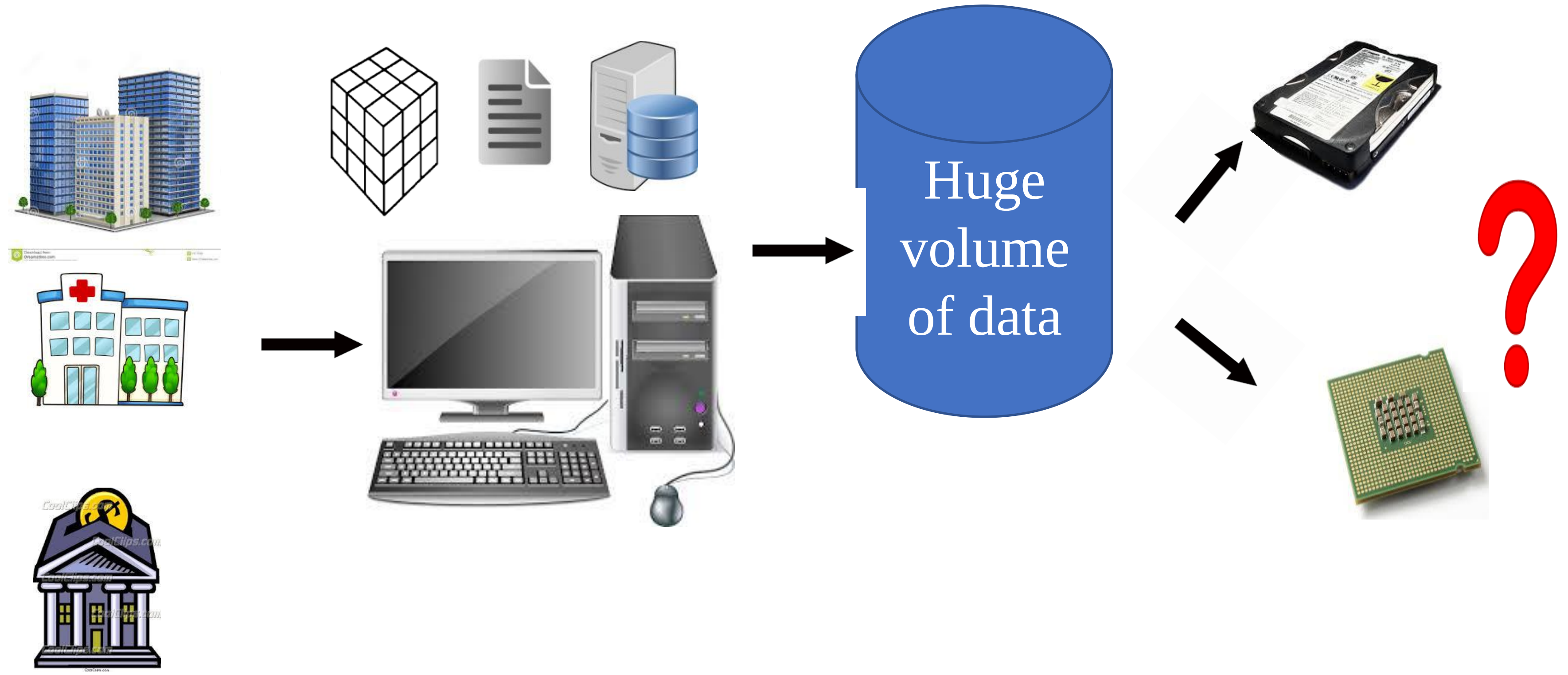
Need of a technology which deals with Semi-structured data, Unstructured data and Structured data as well

Characteristics :3 V’s of big data



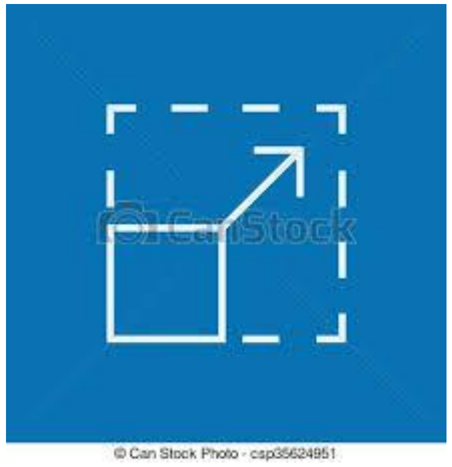
Volume provides the **amount** of data and the **form** of data
Data Volume
- Terabytes
- Records
- Transactions
- Tables, Files
Data Velocity
- Near Time
- Real Time
- Streams
- Batches
Velocity provides the **time** at which the data is collected and analyzed
Data Variety
- Strucured
- Semi-Strucured
- Unstructured
- Mixed
Variety provides the **type** of data collected

Draw backs of traditional system

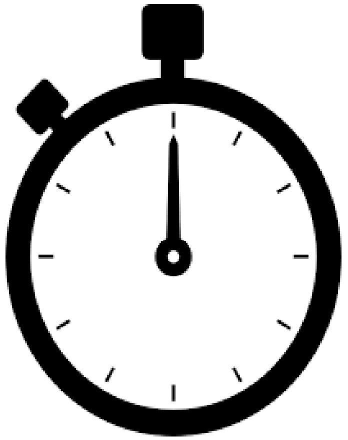


Draw backs:

Expensive

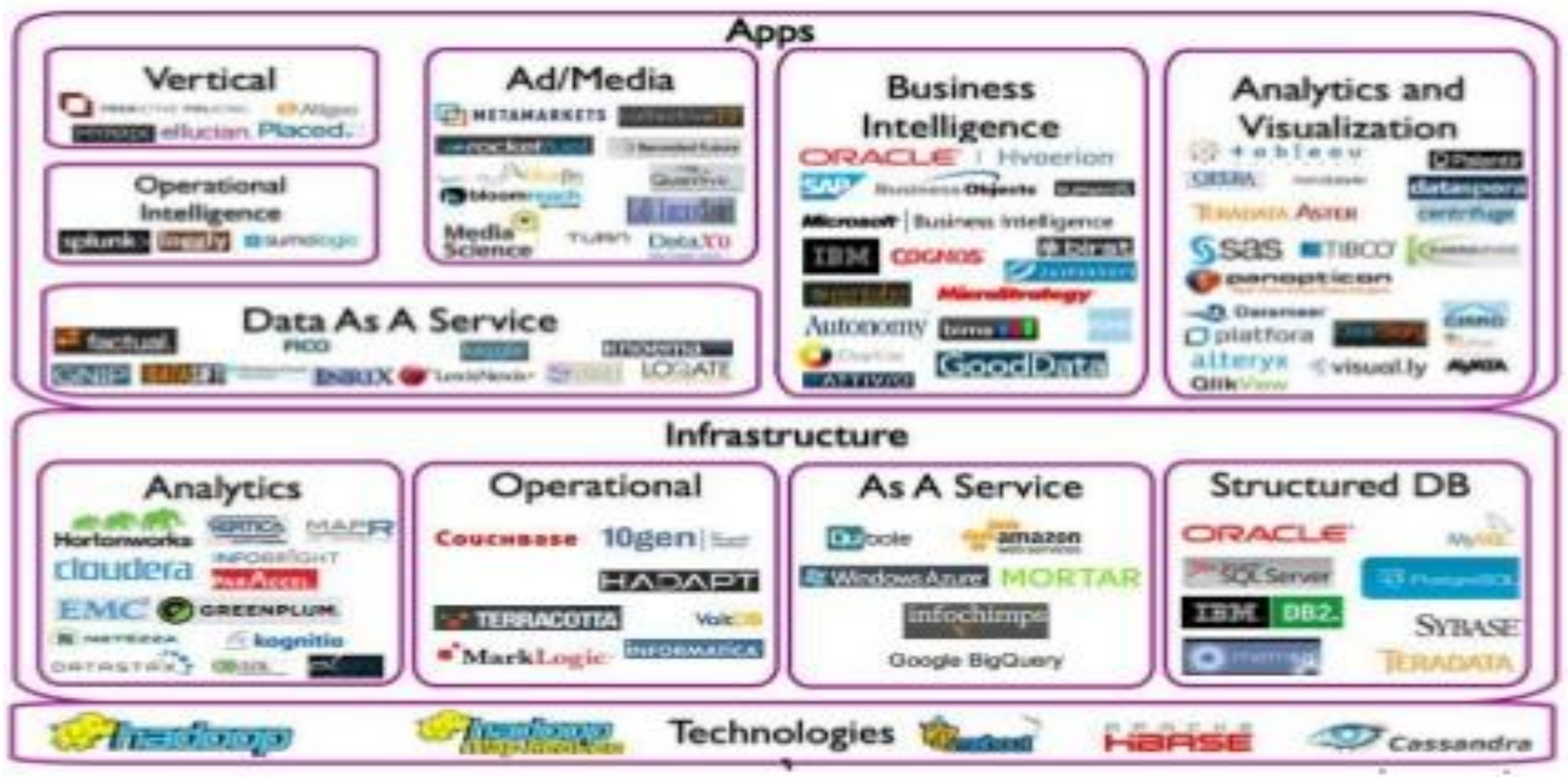


Scalability

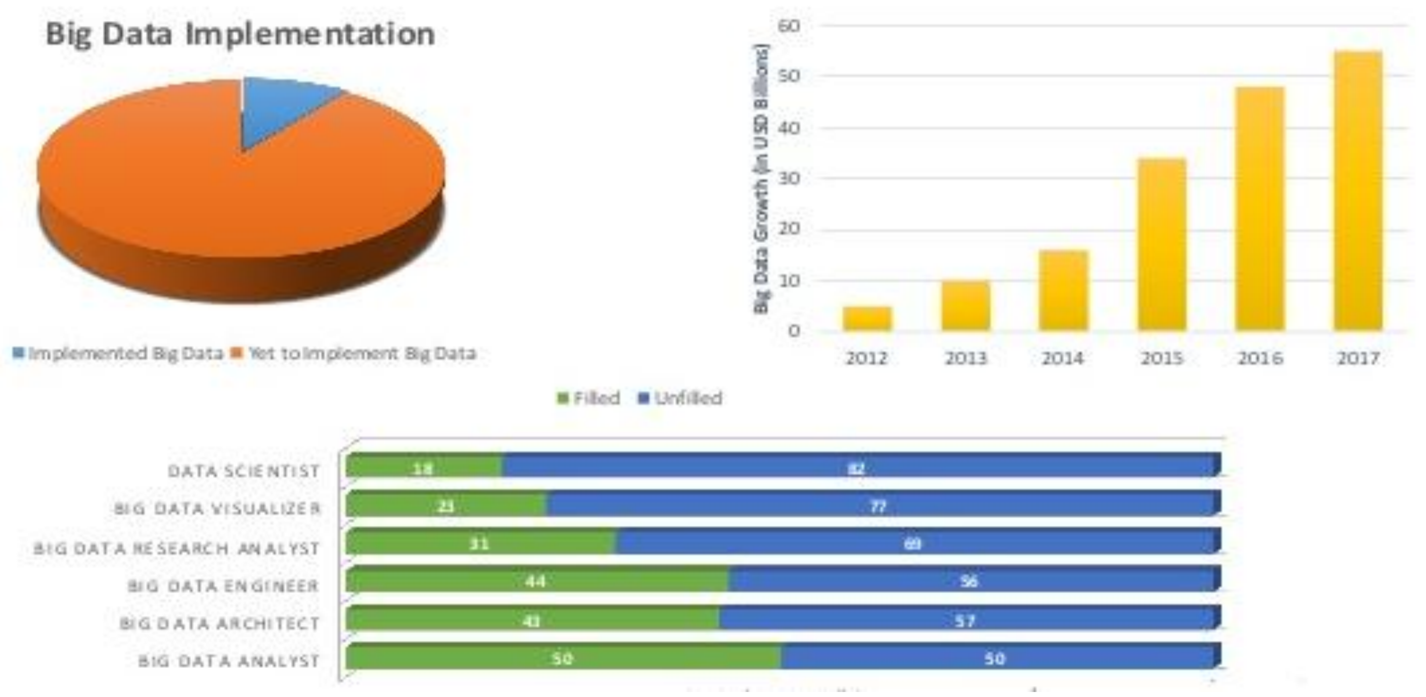


Time consuming

Various big data technologies

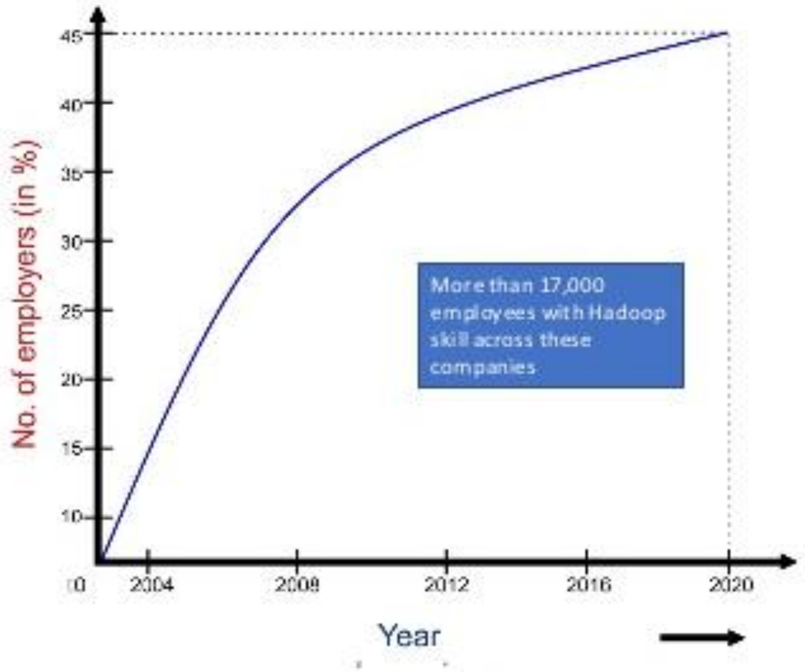


Big data Global Market

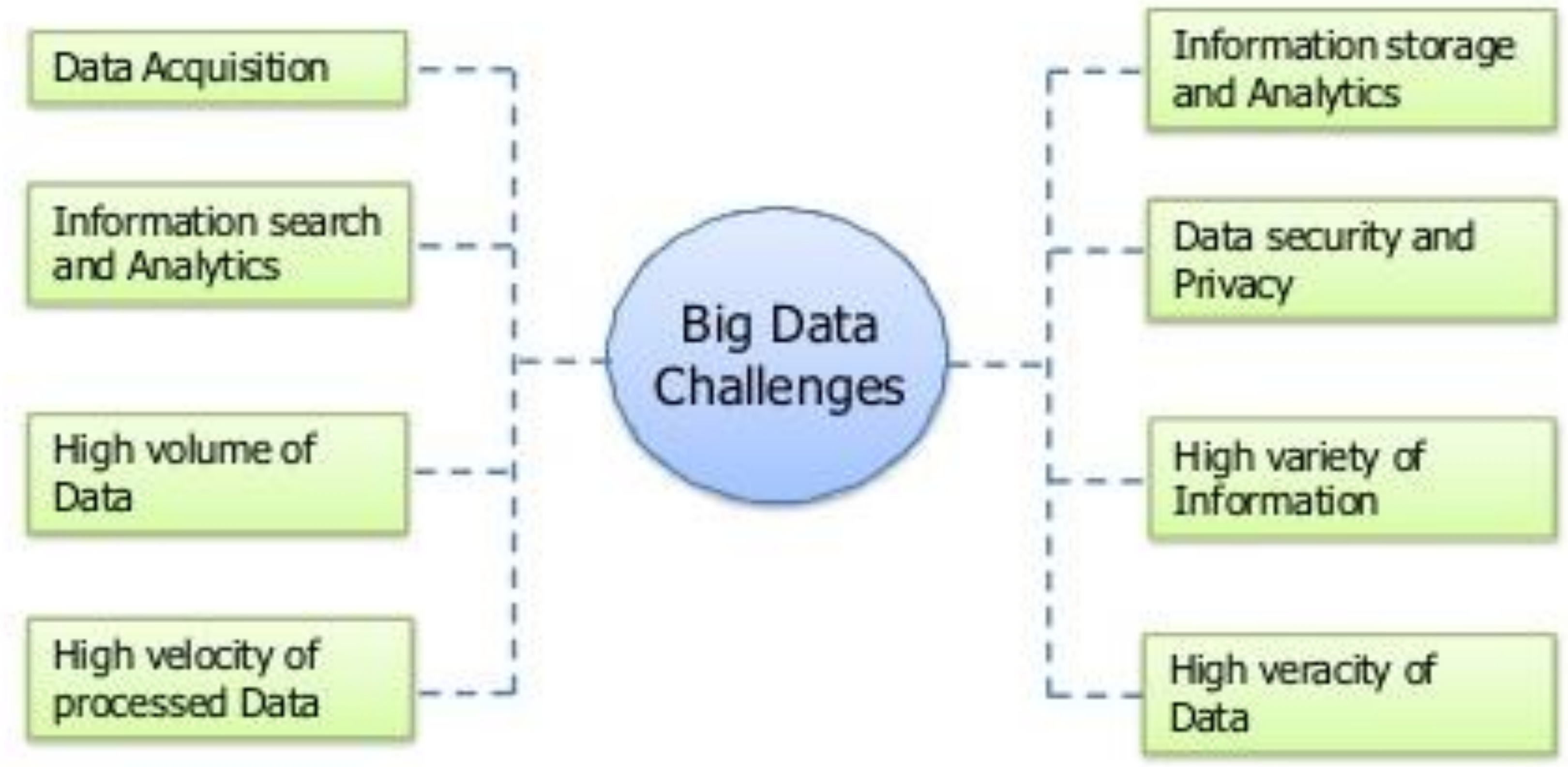


Hadoop Global Job Trends

Top Hadoop Technology Companies



Challenges of big data


Data Acquisition
Information search and Analytics
High volume of Data
High velocity of processed Data
Big Data Challenges
Information storage and Analytics
Data security and Privacy
High variety of Information
High veracity of Data

How to overcome the challenges?

Problem 1: Data is too big to store on one machine.

HDFS: Store the data on multiple machines!

Problem 2: Very high end machines are too expensive

HDFS: Run on commodity hardware!

Problem 3: Commodity hardware will fail!

HDFS: Software is intelligent enough to handle hardware failure!

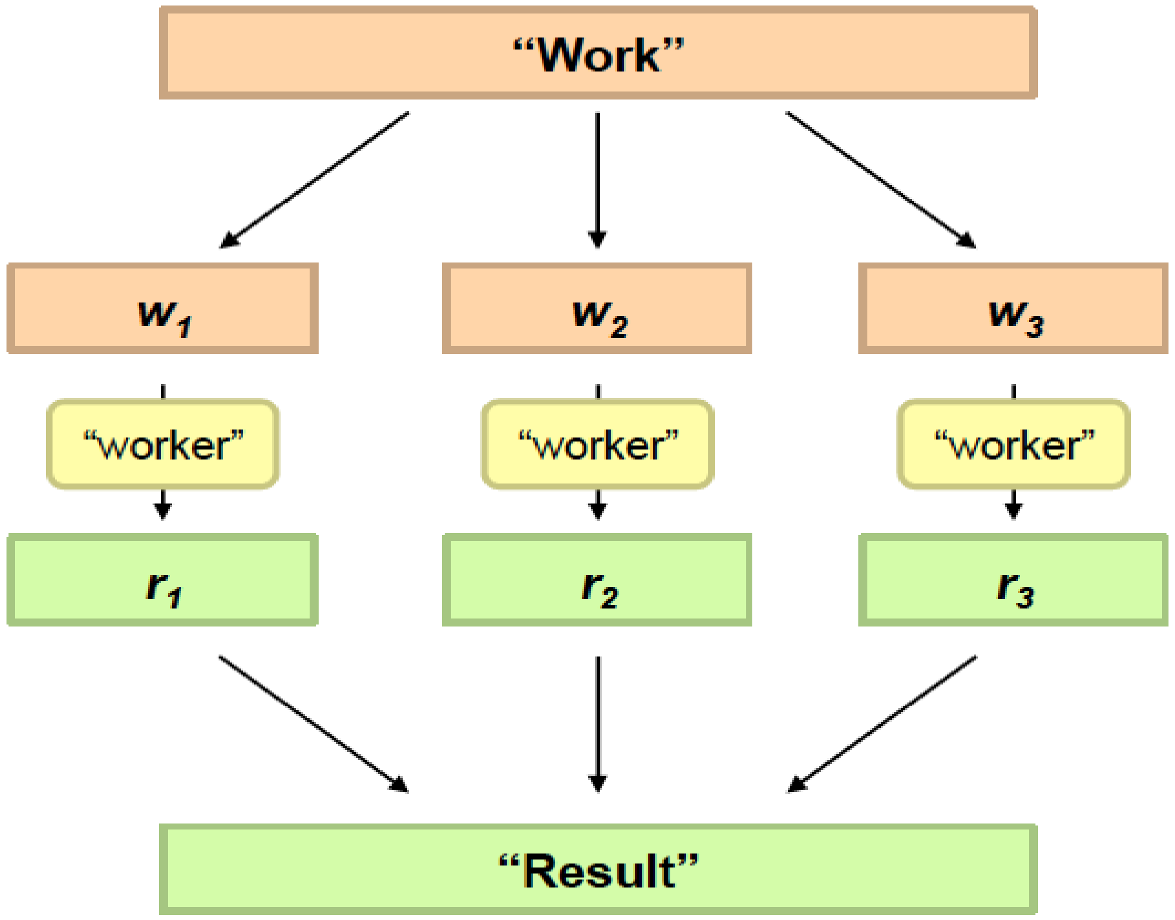
Problem 4: What happens to the data if the machine stores the data fails?

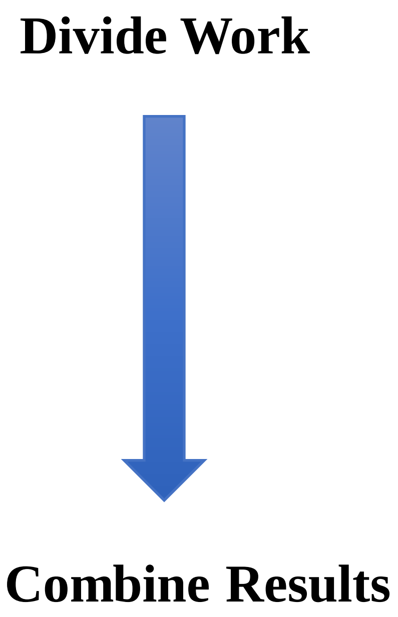
HDFS: Replicate the data!

Problem 5: How can distributed machines organize the data in a coordinated way?

HDFS: Master-Slave Architecture!

Divide and conquer philosophy:





Distributed processing is non-trivial

✓How to assign tasks to different workers in an efficient way?

✓What happens if tasks fail?

✓How do workers exchange results?

✓How to synchronize distributed tasks allocated to different workers?

Big data storage is challenging

✓Data Volumes are massive

✓Reliability of Storing PBs of data is challenging

✓All kinds of failures: Disk/Hardware/Network Failures

✓Probability of failures simply increase with the number of machines …

One popular solution: Hadoop

Hadoop Cluster at Yahoo! (Credit: Yahoo)

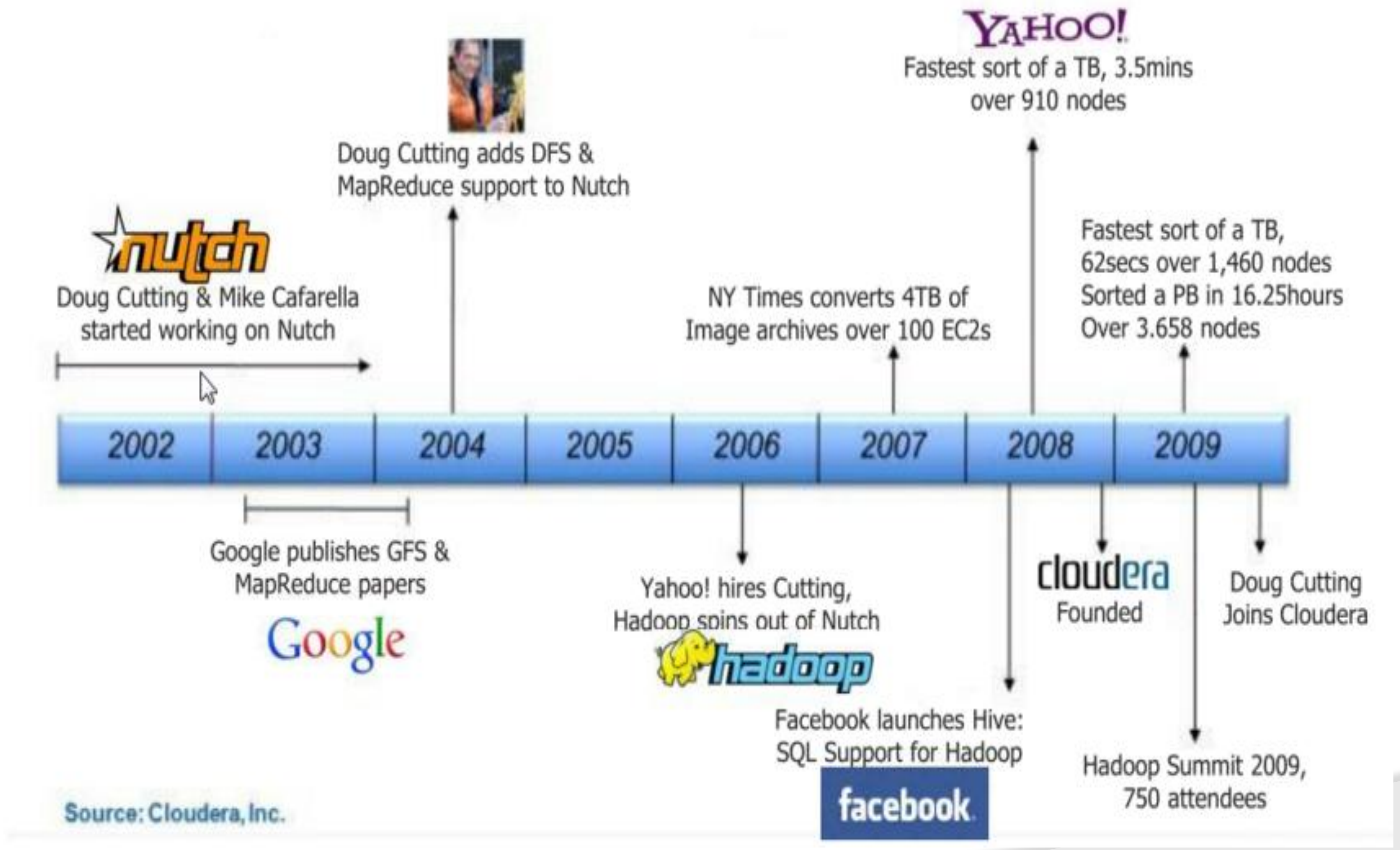
BIG DATA QUESTIONS

- What is data?
- What is information?
- List out different data generators in the world?
- The data generation was vast from -----year and % ---?
- How the data is upgraded and show the hierarchy?
- What is data ananlysis?
- What is data center?
- What is batch data?
- What is Streaming Data?
- What is commodity hardware?

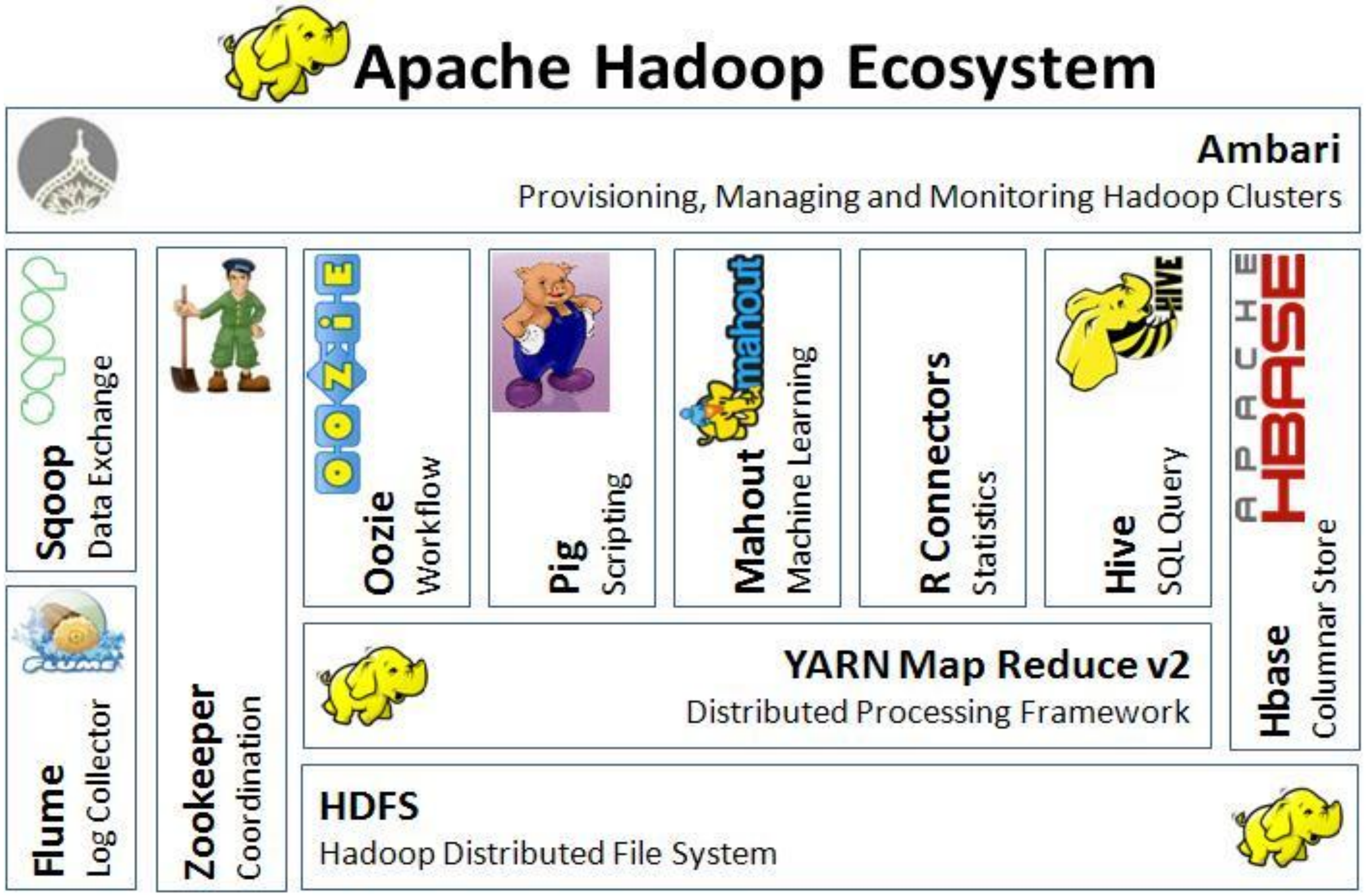
- Drastical change in the environment of data analysis?
- What is OLAP?
- What is OLTP?
- What is RTAP?
- What is BIG DATA?
- BIG DATA example?
- What are classifications of big data?
- What are characteristics of big data?
- What are challenges of big data?
- Drawbacks of traditional system?
- Difference between big data and Hadoop?

UNIT 2
INTRODUCTION TO HADOOP

History of hadoop

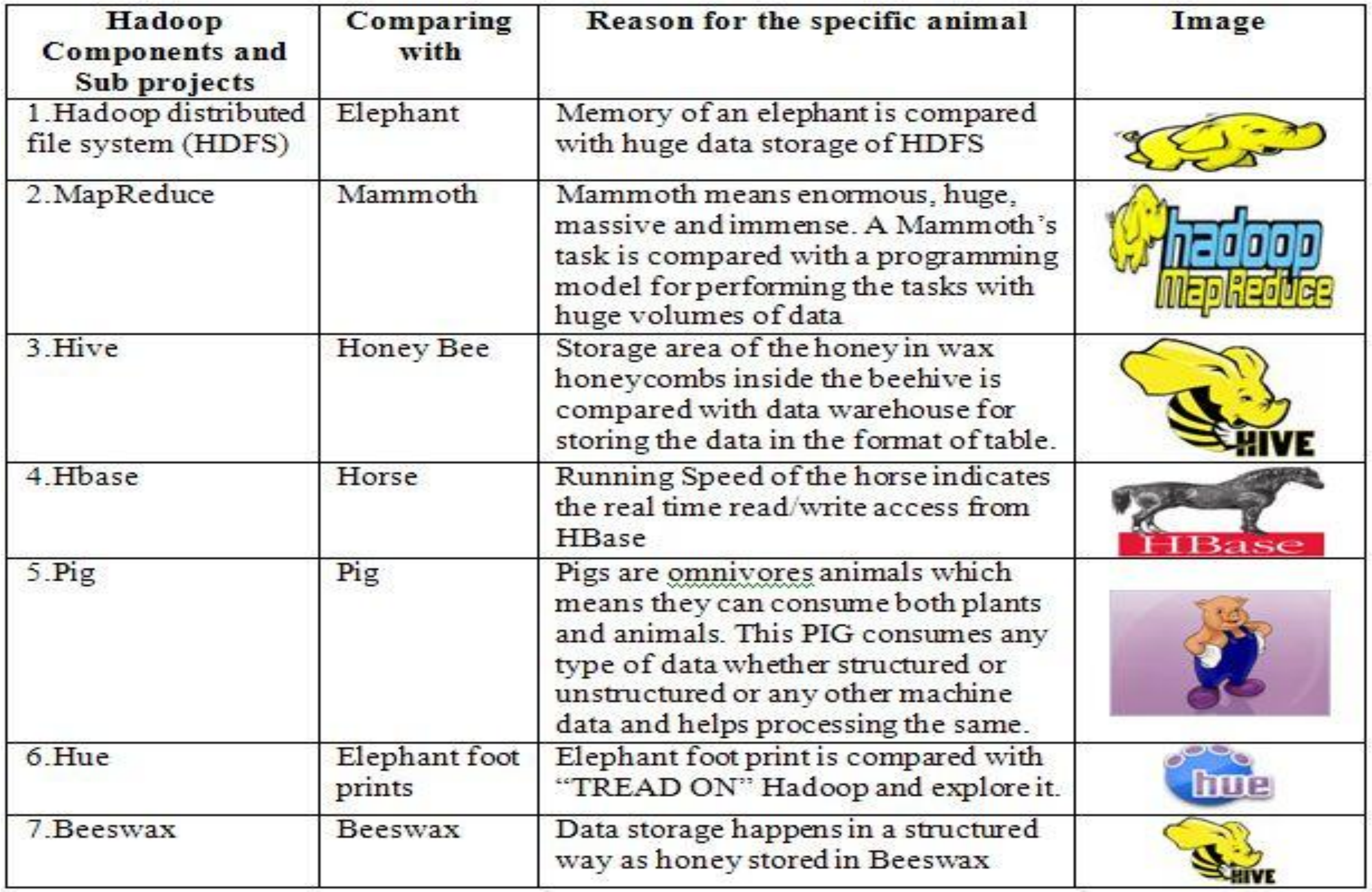


YAHOO!
Fastest sort of a TB, 3.5mins over 910 nodes
Doug Cutting adds DFS & MapReduce support to Nutch
nutch
Doug Cutting & Mike Cafarella started working on Nutch
Fastest sort of a TB, 62secs over 1,460 nodes Sorted a PB in 16.25hours Over 3.658 nodes
NY Times converts 4TB of Image archives over 100 EC2s
2002
2003
2004
2005
2006
2007
2008
2009
Google publishes GFS & MapReduce papers
Google
Yahoo! hires Cutting, Hadoop spins out of Nutch
hadoop
cloudera
Founded
Doug Cutting Joins Cloudera
Facebook launches Hive: SQL Support for Hadoop
facebook
Hadoop Summit 2009, 750 attendees
Source: Cloudera, Inc.



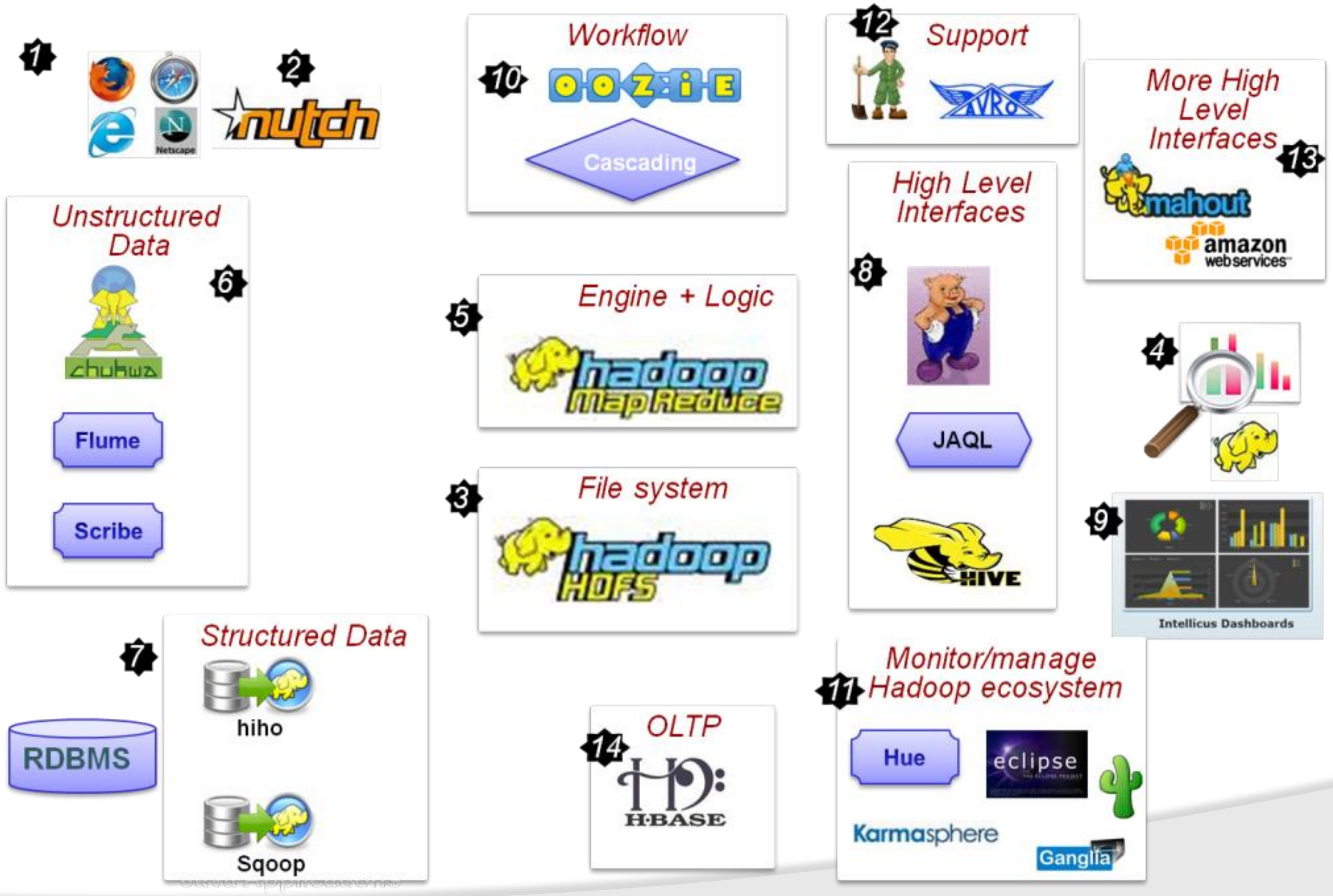
Apache Hadoop Ecosystem
Ambari
Provisioning, Managing and Monitoring Hadoop Clusters
Sqoop
Data Exchange
Flume
Log Collector
Zookeeper
Coordination
Oozie
Workflow
Pig
Scripting
Mahout
Machine Learning
R Connectors
Statistics
Hive
SQL Query
Hbase
Columnar Store
YARN Map Reduce v2
Distributed Processing Framework
HDFS
Hadoop Distributed File System

Animal planet for hadoop



Hadoop Components and Sub projects	Comparing with	Reason for the specific animal	Image
1.Hadoop distributed file system (HDFS)	Elephant	Memory of an elephant is compared with huge data storage of HDFS	
2.MapReduce	Mammoth	Mammoth means enormous, huge, massive and immense. A Mammoth's task is compared with a programming model for performing the tasks with huge volumes of data	
3.Hive	Honey Bee	Storage area of the honey in wax honeycombs inside the beehive is compared with data warehouse for storing the data in the format of table.	
4.Hbase	Horse	Running Speed of the horse indicates the real time read/write access from HBase	
5.Pig	Pig	Pigs are omnivores animals which means they can consume both plants and animals. This PIG consumes any type of data whether structured or unstructured or any other machine data and helps processing the same.	
6.Hue	Elephant foot prints	Elephant foot print is compared with "TREAD ON" Hadoop and explore it.	
7.Beeswax	Beeswax	Data storage happens in a structured way as honey stored in Beeswax	

Hadoop Ecosystem Map



How hadoop was born?

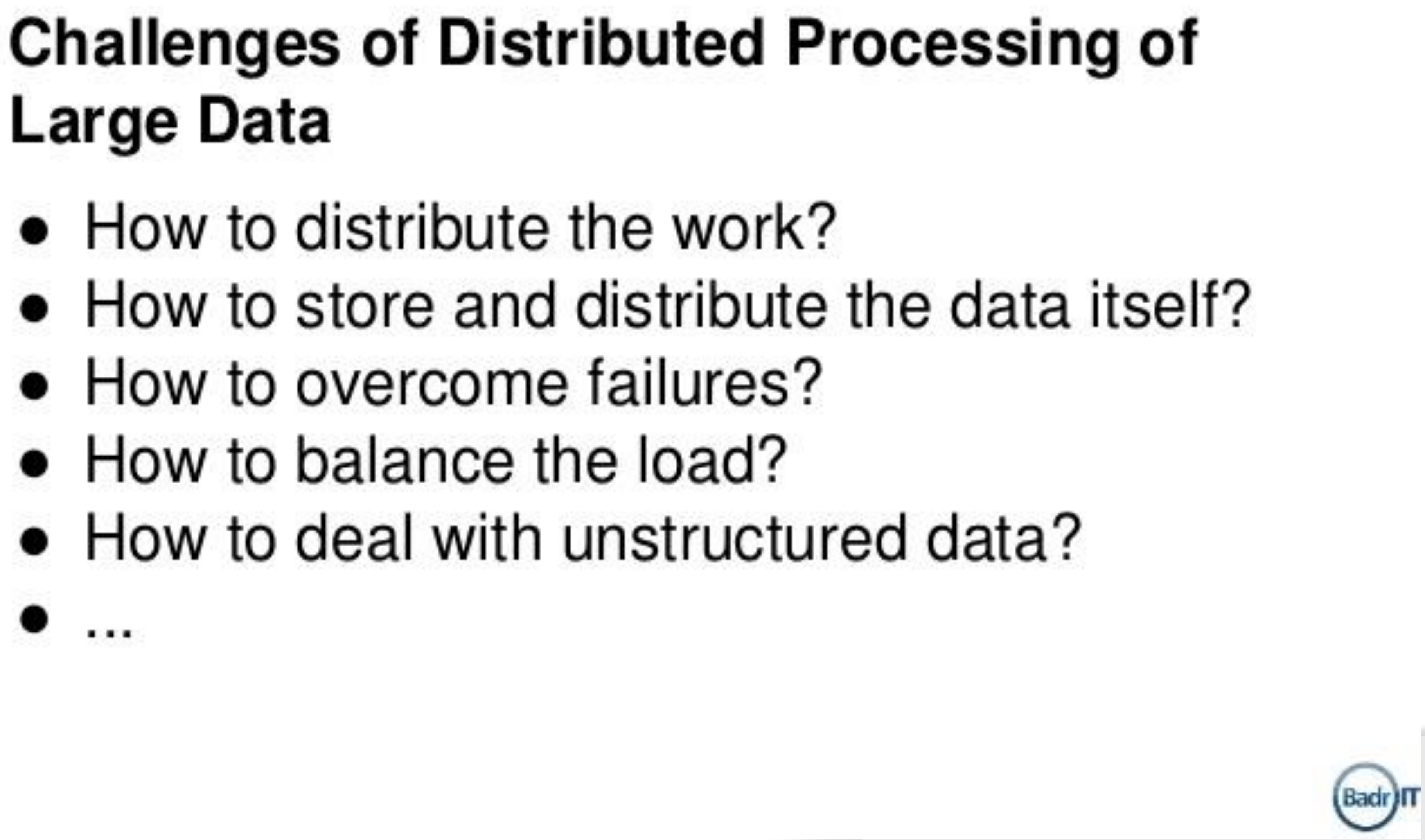
Lucene

nutch

Google

hadoop

Doug Cutting



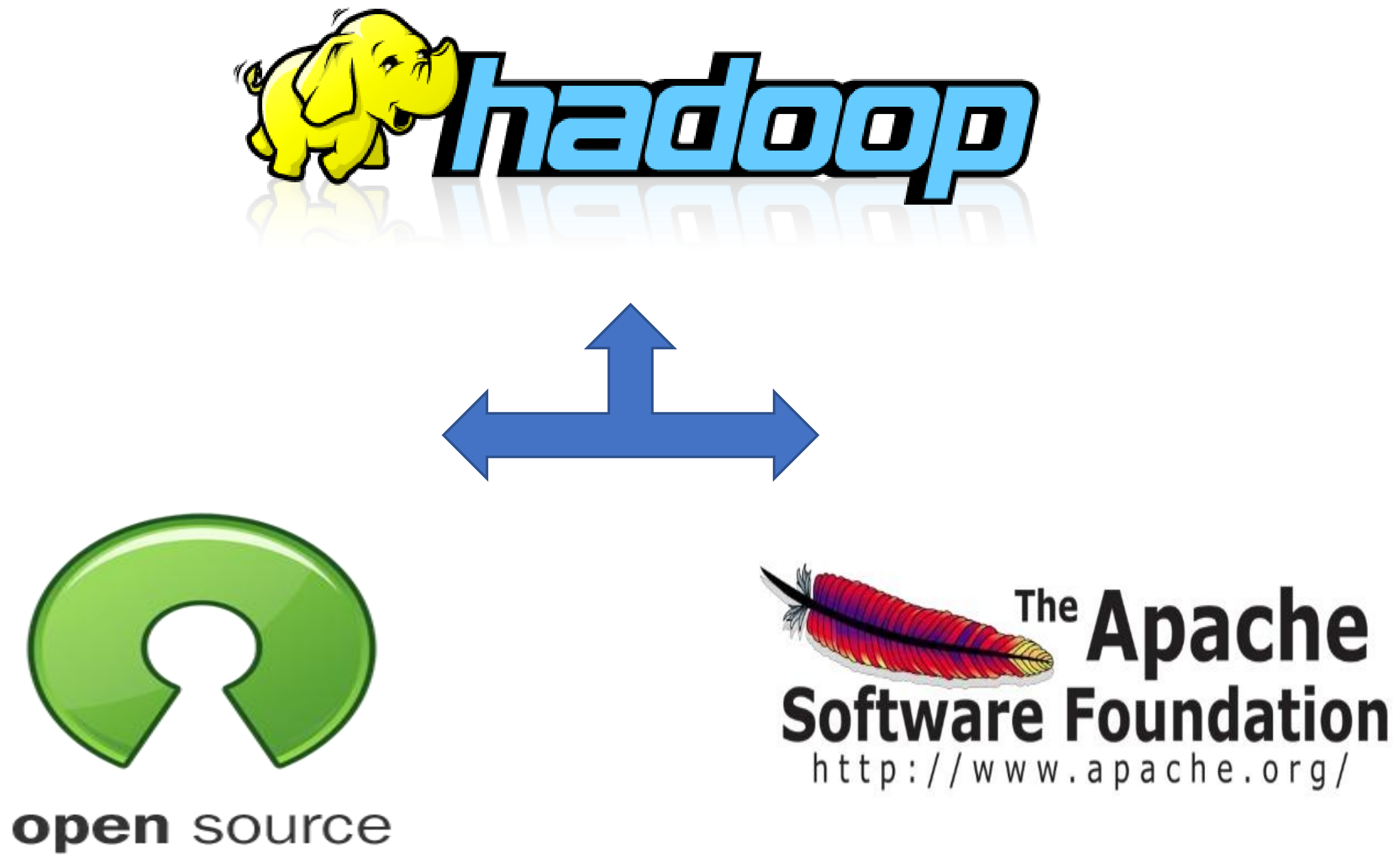
Challenges of Distributed Processing of Large Data

- How to distribute the work?
- How to store and distribute the data itself?
- How to overcome failures?
- How to balance the load?
- How to deal with unstructured data?
- ...

Hadoop tackles these challenges!

So, what's Hadoop?

What is hadoop?

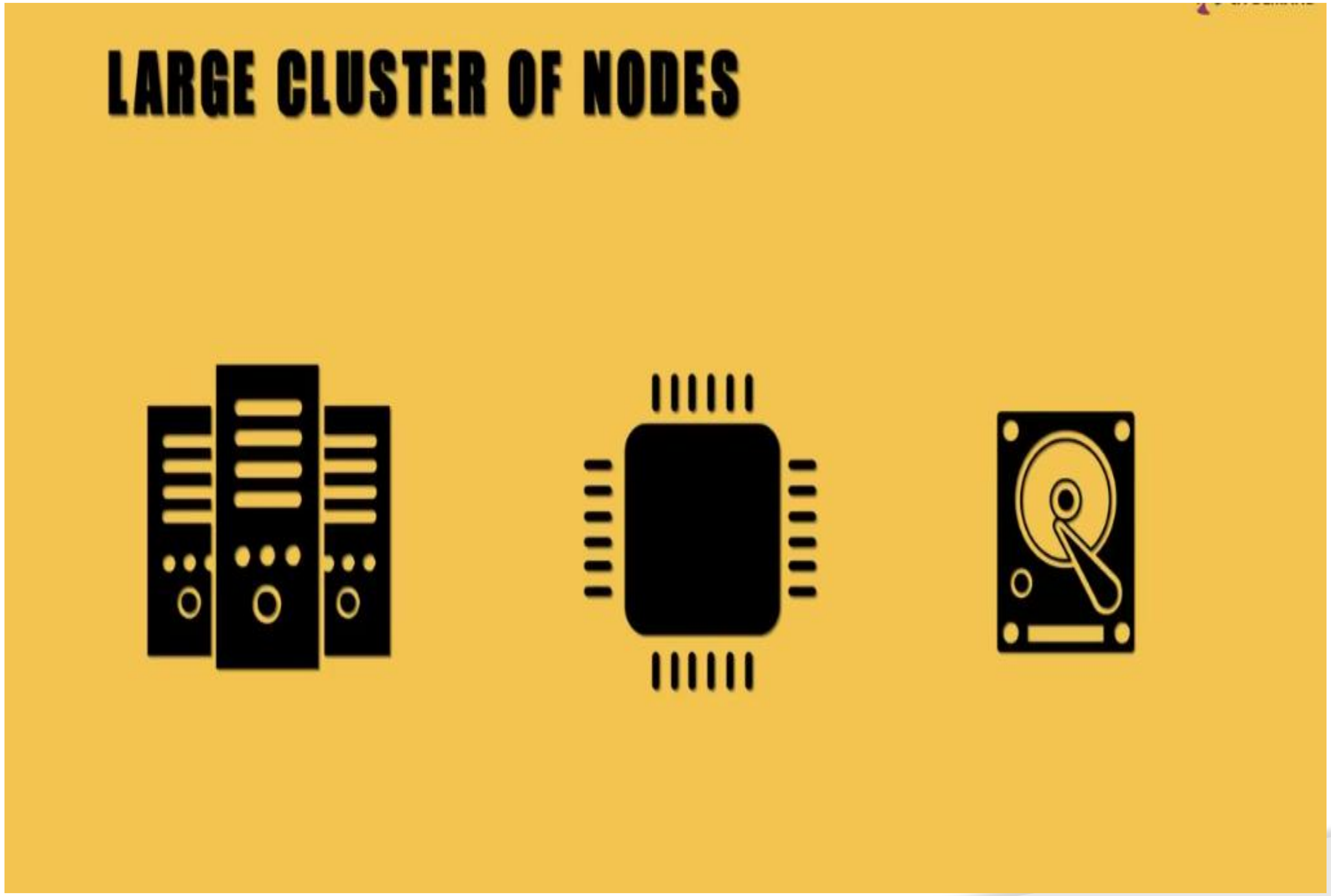


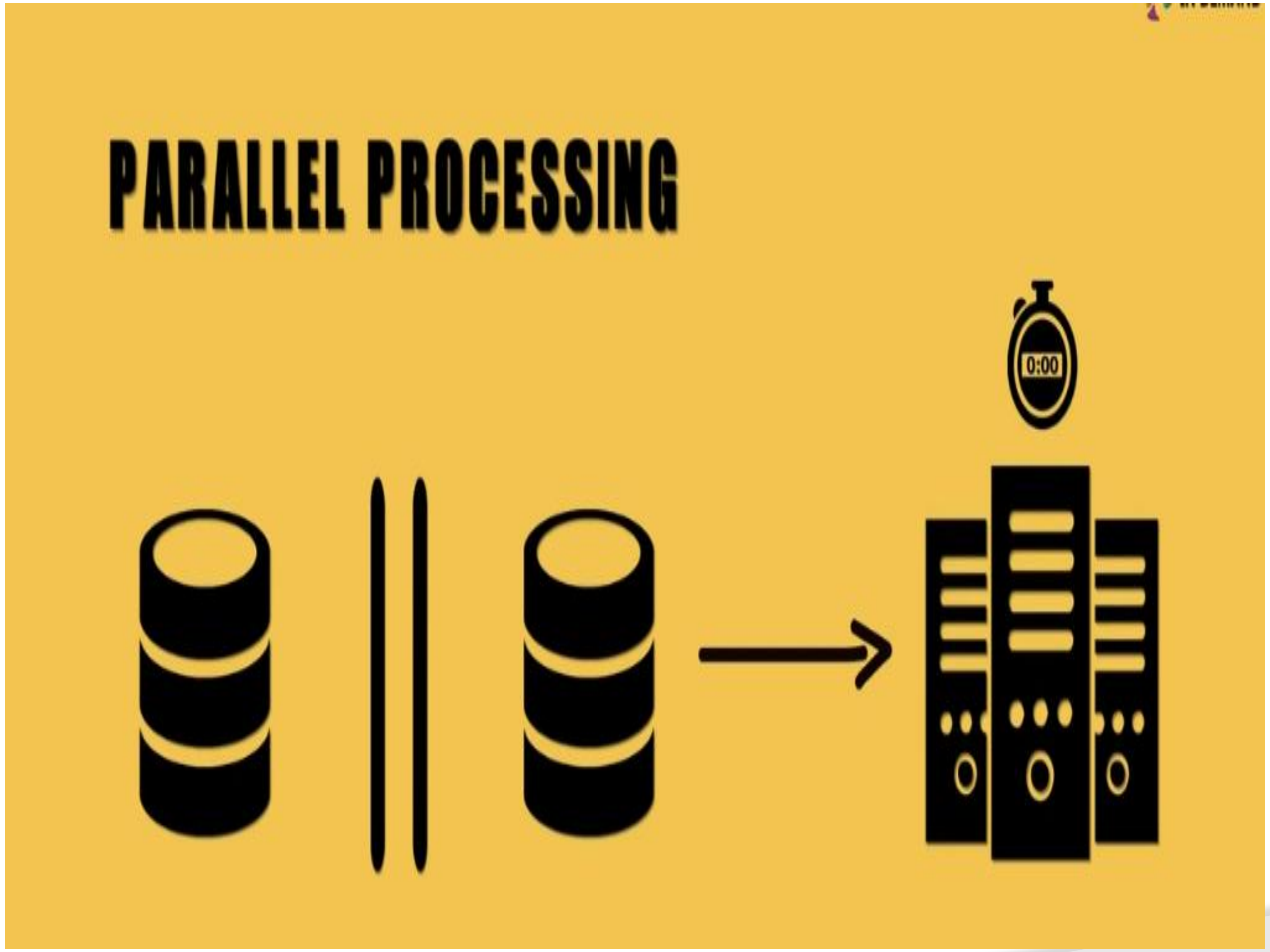
The **name Hadoop** is not an acronym; it's a made-up **name**. The project's creator, Doug Cutting, explains how the **name** came about: The **name** my kid gave a stuffed yellow elephant. usually called the toy as **hadoop**

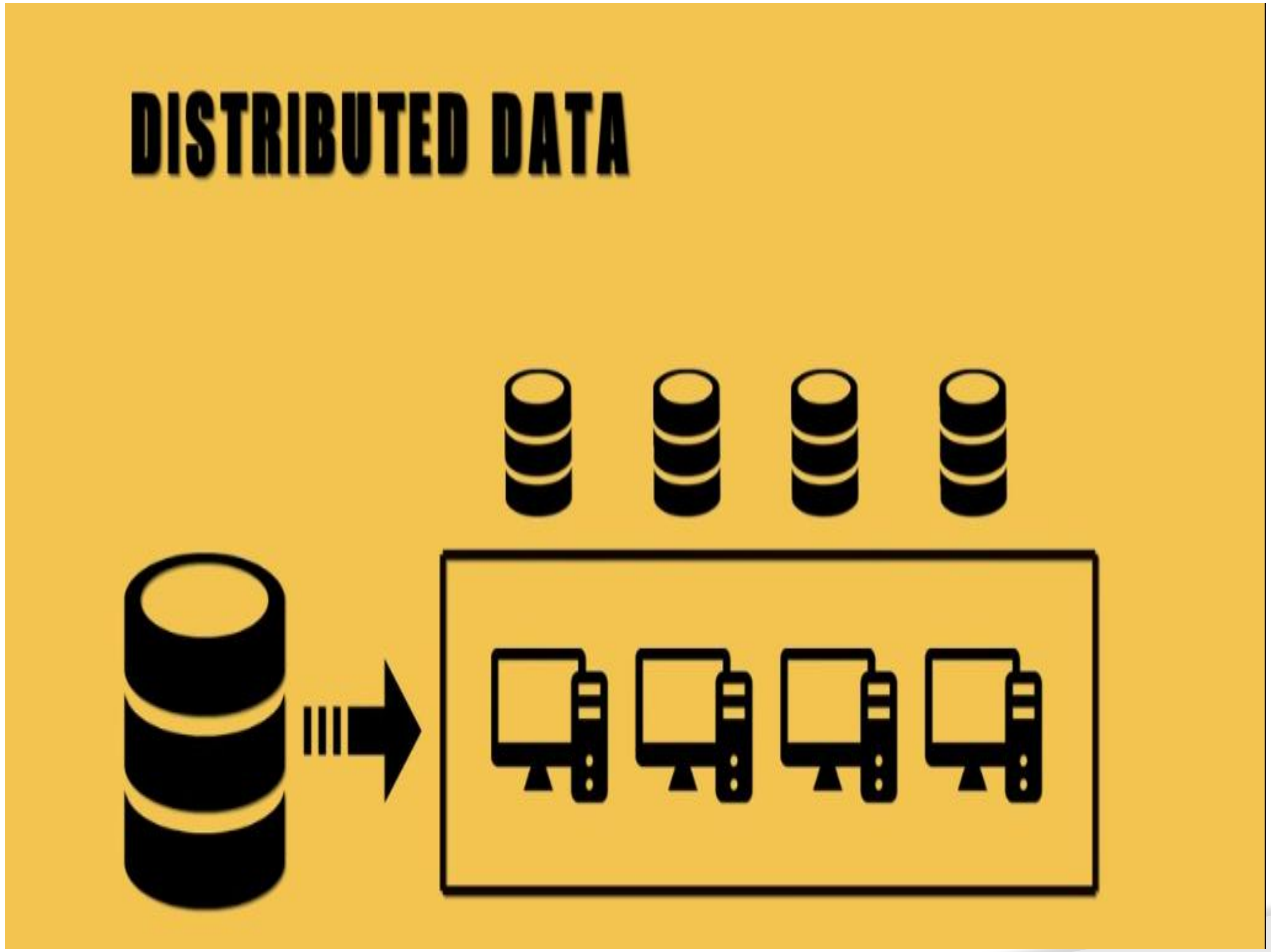
Features of hadoop:

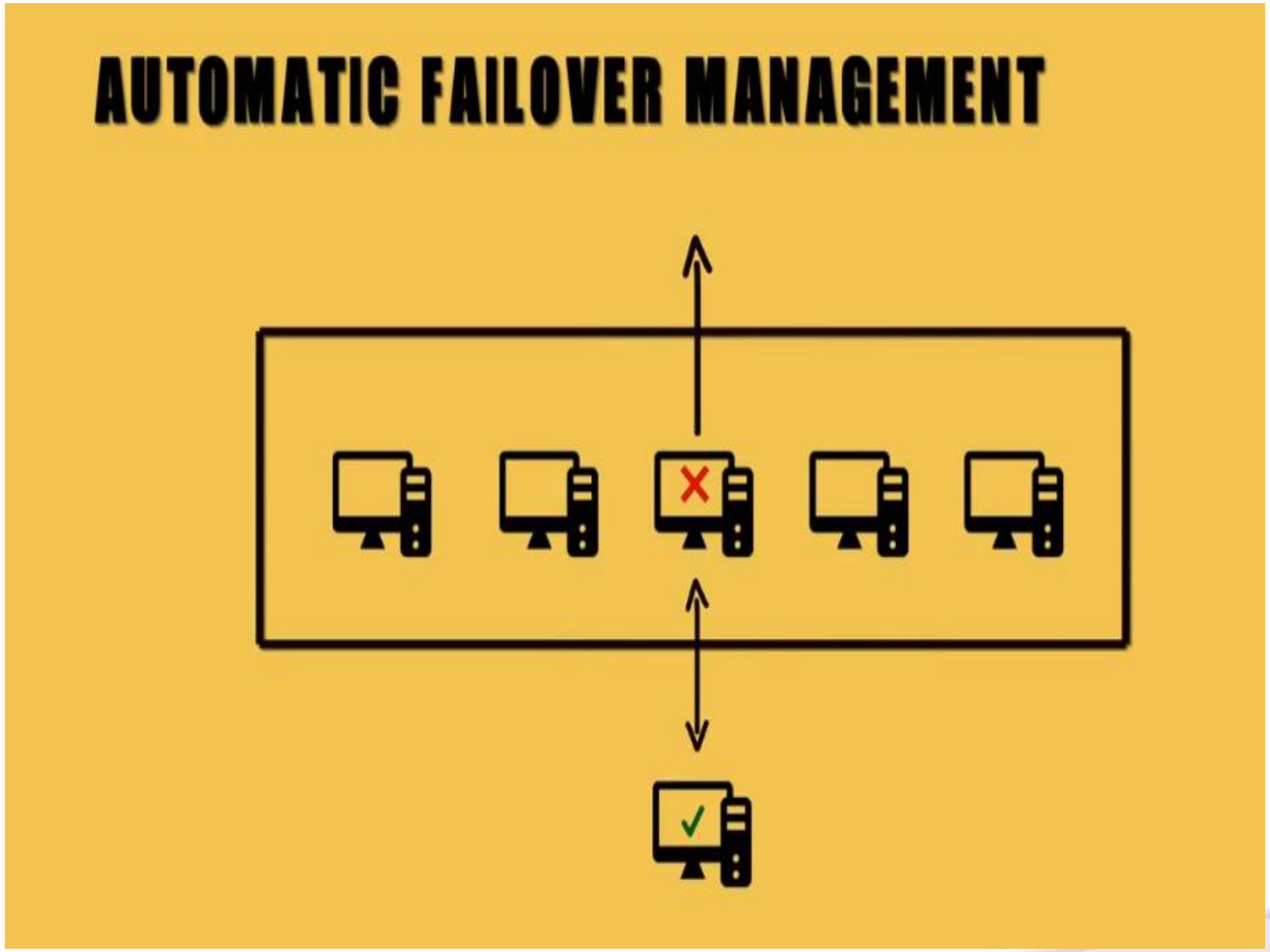
- ✓Cost Effective System
- ✓A Large Cluster of Nodes
- ✓Parallel Processing Data
- ✓Distributed Data
- ✓Automatic Failover Management
- ✓Data Locality Optimization
- ✓Heterogenous Cluster
- ✓Scalability

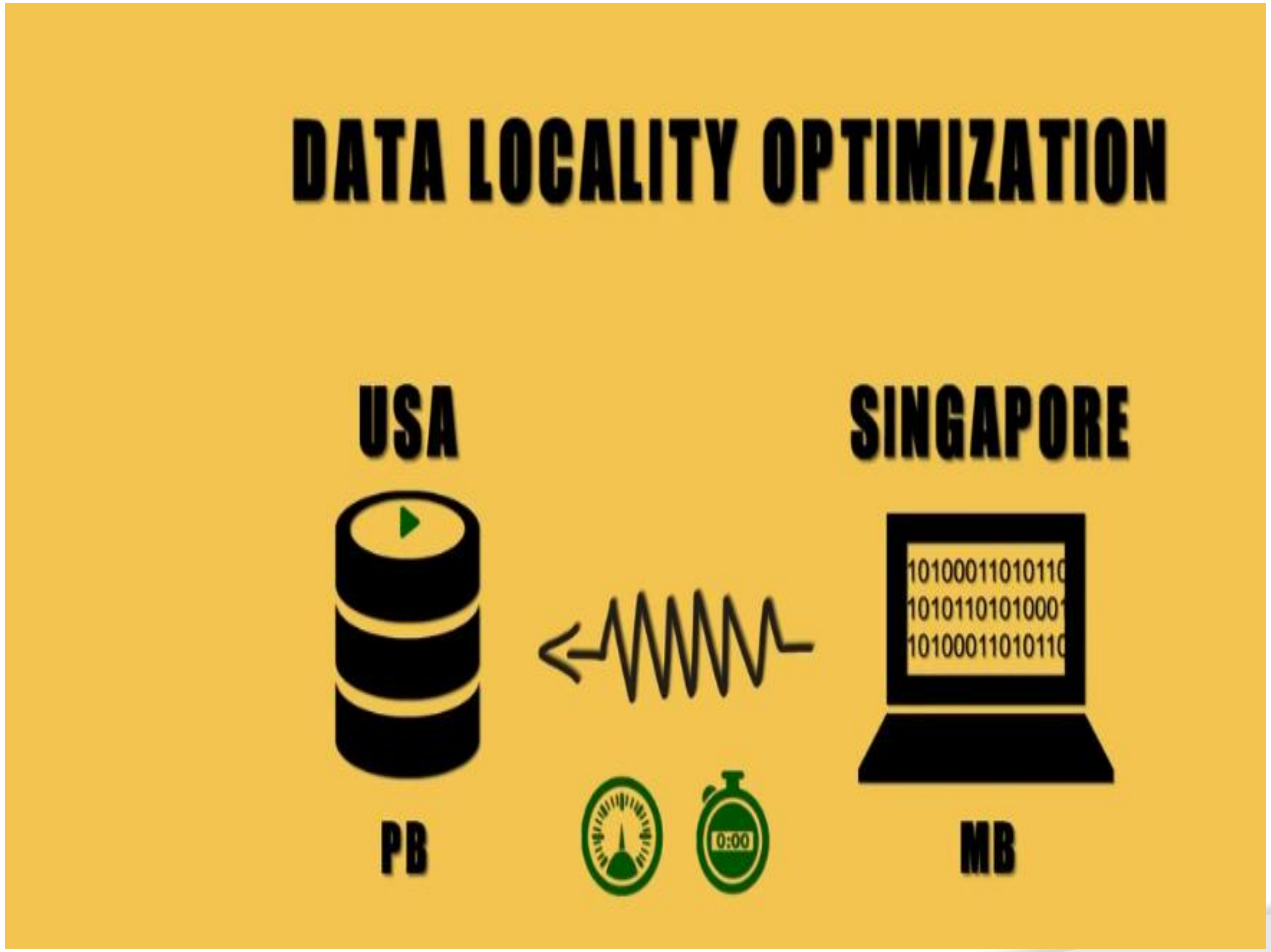
COST EFFECTIVE SYSTEM

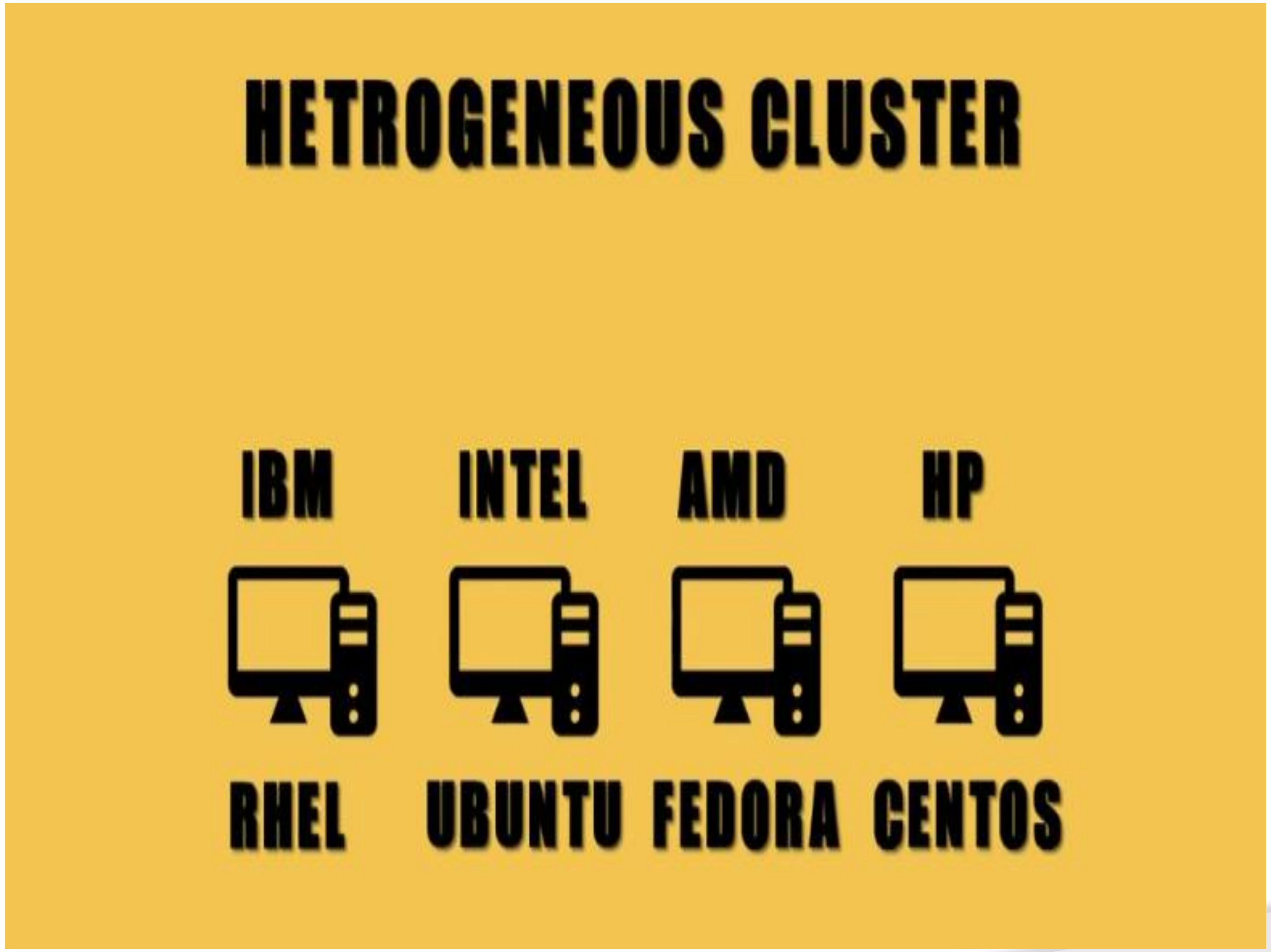

LARGE CLUSTER OF NODES

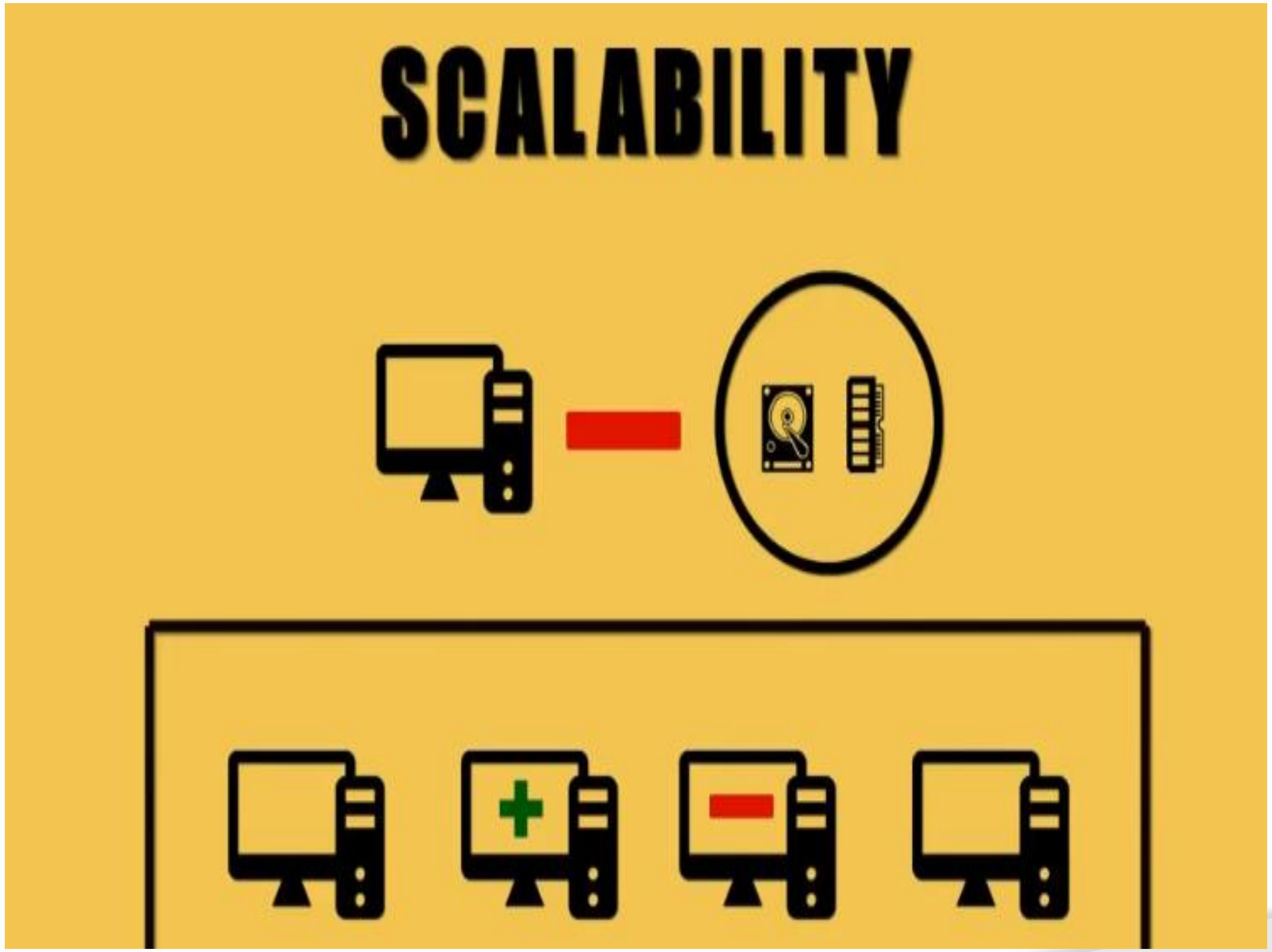

PARALLEL PROCESSING
0:00


DISTRIBUTED DATA

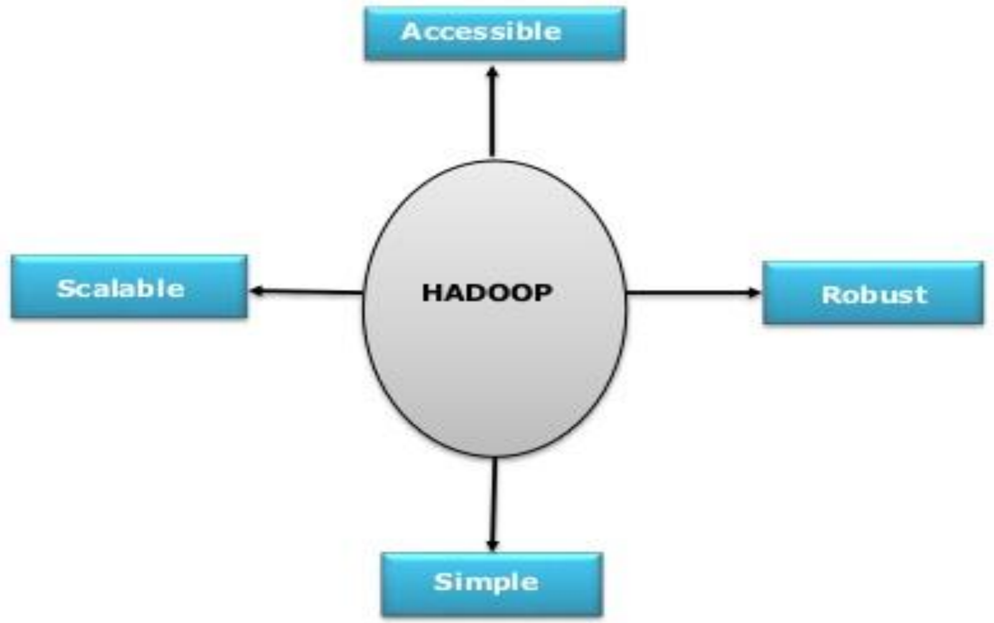

AUTOMATIC FAILOVER MANAGEMENT

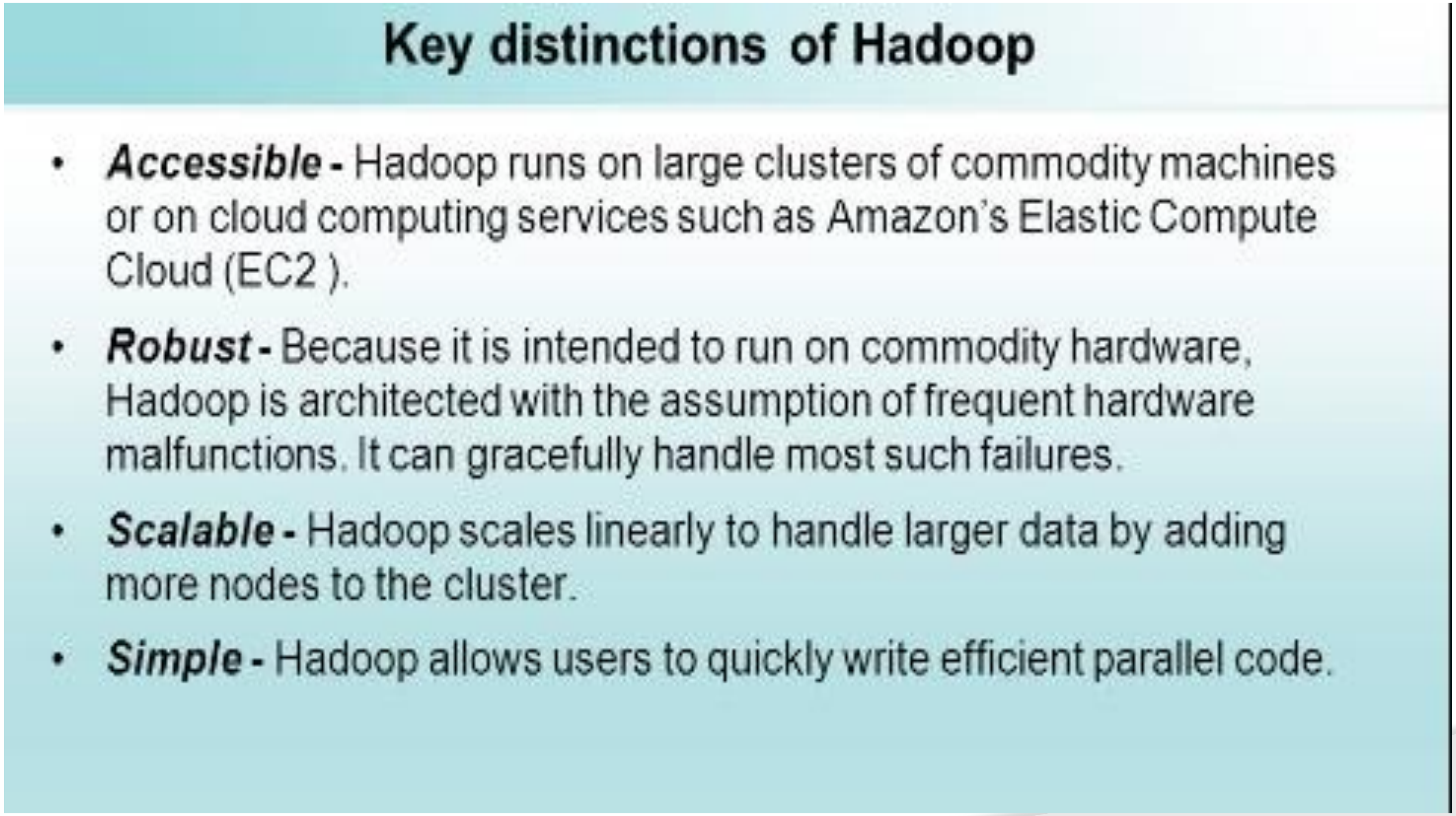

DATA LOCALITY OPTIMIZATION
USA
SINGAPORE
1010001101011
1010110101000
1010001101011
PB
0:00
MB


HETROGENEOUS CLUSTER
IBM
INTEL
AMD
HP
RHEL
UBUNTU
FEDORA
CENTOS


SCALABILITY

Key Distinctions of Hadoop



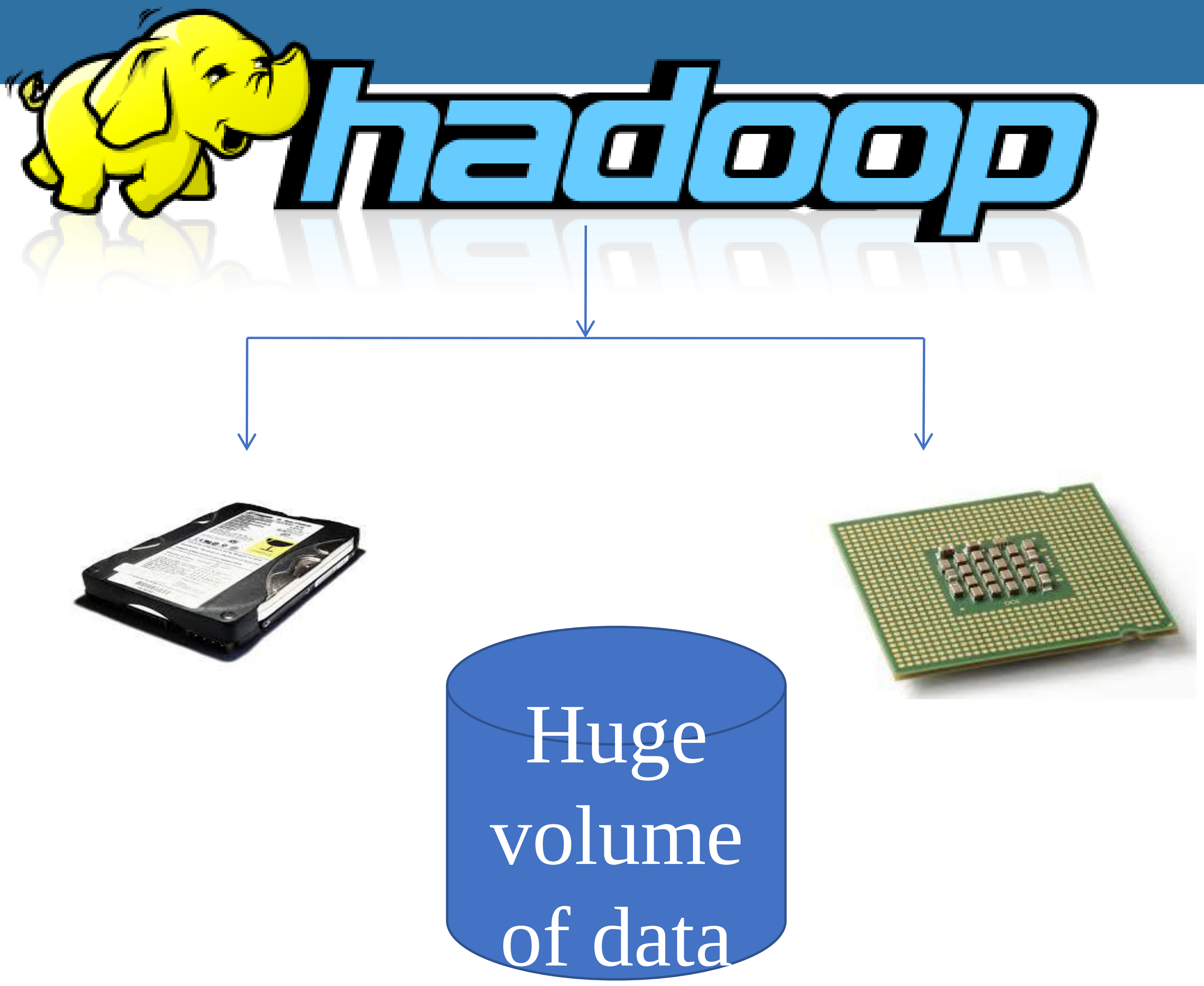


Key distinctions of Hadoop

- ***Accessible*** **-** Hadoop runs on large clusters of commodity machines or on cloud computing services such as Amazon's Elastic Compute Cloud (EC2).
- ***Robust*** **-** Because it is intended to run on commodity hardware, Hadoop is architected with the assumption of frequent hardware malfunctions. It can gracefully handle most such failures.
- ***Scalable*** **-** Hadoop scales linearly to handle larger data by adding more nodes to the cluster.
- ***Simple*** **-** Hadoop allows users to quickly write efficient parallel code.

Hadoop Components

- **HDFS** – Distributed Filesystem
- **MapReduce** – Distributed Data Processing Model


hadoop
Huge volume of data

- Hadoop :It is basically for storing and processing for only huge volume of data(rather than smaller) with in the cluster of commodity hardware.

  Example:10GB(for smaller)

- Cluster
- Commodity hardware: mobiles, laptops, pc’s

  Example:500 TB

UNIT-3

THE HADOOP DISTRIBUTED FILESYSTEM

HDFS

What is HDFS

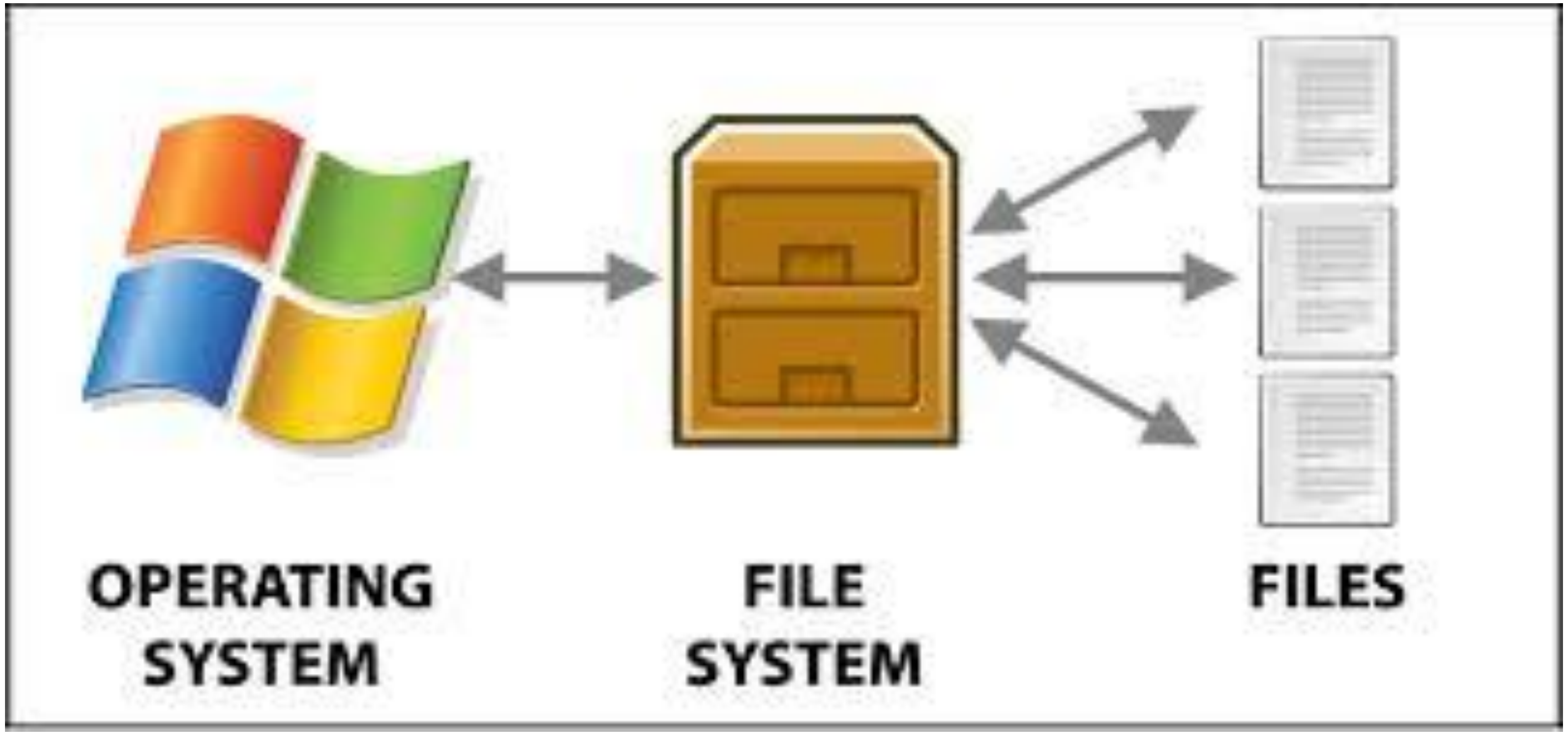
✓HDFS- Hadoop Distributed File System.

✓HDFS-It is specially designed Filesystem for storing huge dataset with a cluster of commodity hardware’s with streaming access pattern.

What is Streaming access pattern?

- ✓JAVA Slogan : Write once and run anywhere.
- ✓HDFS : Write once and read any number of times.

What is a FileSystem



Comparing FS and HDFS

- Block size is 4KB.
- Wastage of free space in the blocks
- No reuse
- Memory wastage

- Block size is 64MB.
- No Wastage of free space in the blocks
- Reuse
- No memory wastage

Why the block size is more?

Example:

HDFS: 1GB-64MB=16 blocks.

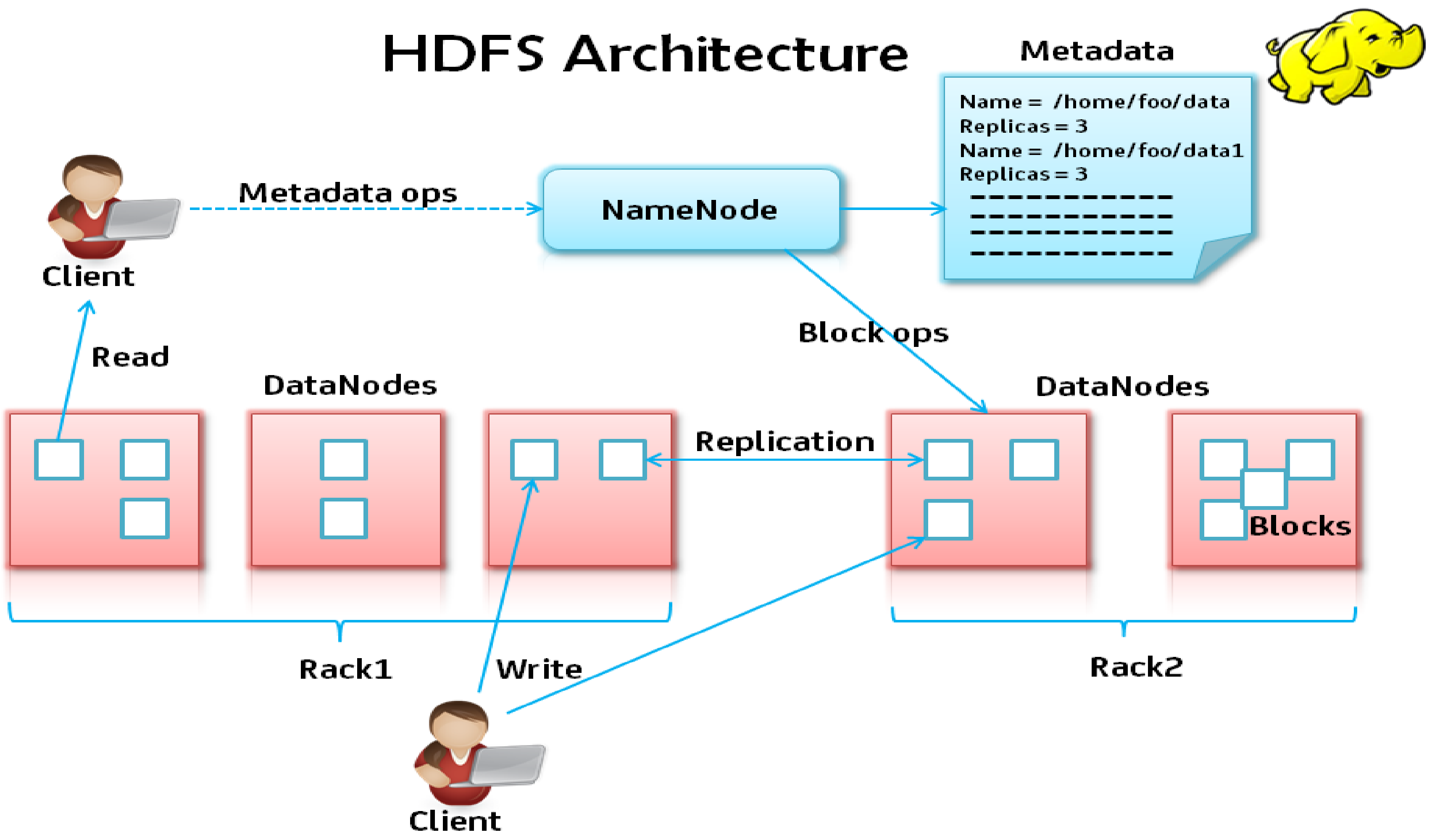
500GB-64MB=7,500blocks.

FS: 1GB-4KB=10,48,576blocks

500GB-4KB=52,42,88,000blocks

What is Hadoop cluster?

- ✓Master – Slave Architecture.
- ✓Data Node
- ✓Secondary Data Node
- ✓Job tracker
- ✓Name Node
- ✓Task Tracker


HDFS Architecture
Metadata
Name = /home/foo/data
Replicas = 3
Name = /home/foo/data1
Replicas = 3
Client
Metadata ops
NameNode
Block ops
Read
DataNodes
DataNodes
Replication
Blocks
Rack1
Write
Rack2
Client

Hadoop vs SQL

- Structured and Unstructured data.
- Datastore and Data Analysis.
- Scale-out and Scale-up.
- Offline batch processing and Online transactions.

UNIT-4

UNDERSTANDING MAP REDUCE FUNDAMENTALS

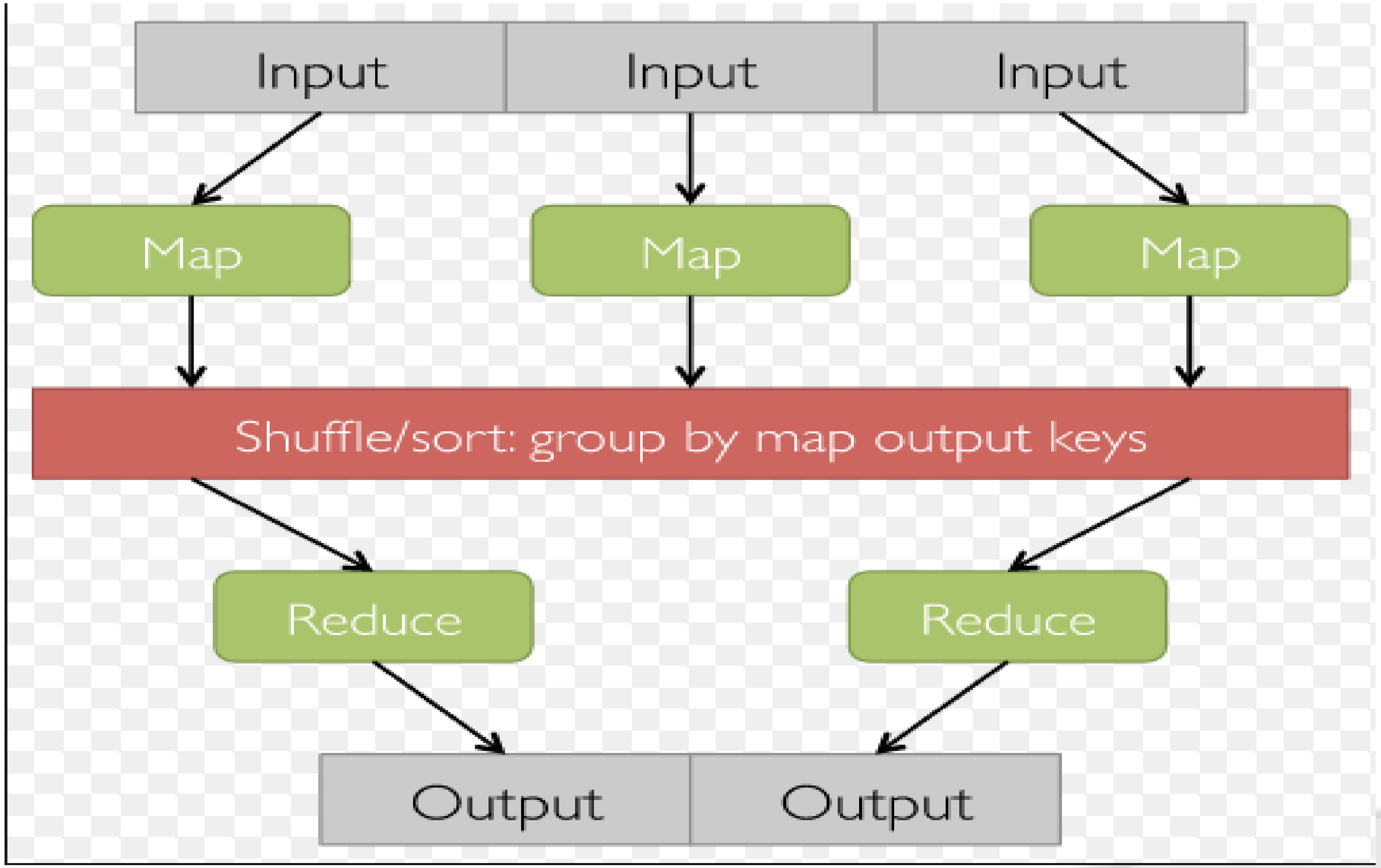
- ✓ What is Map Reduce
- ✓ Few interesting facts about Map Reduce
- ✓ Map Reduce component and architecture
- ✓ How Map Reduce works in Hadoop

What is Map Reduce

Map Reduce is a programming model which is used to process large data sets in a batch processing manner.

A Map Reduce program is composed of

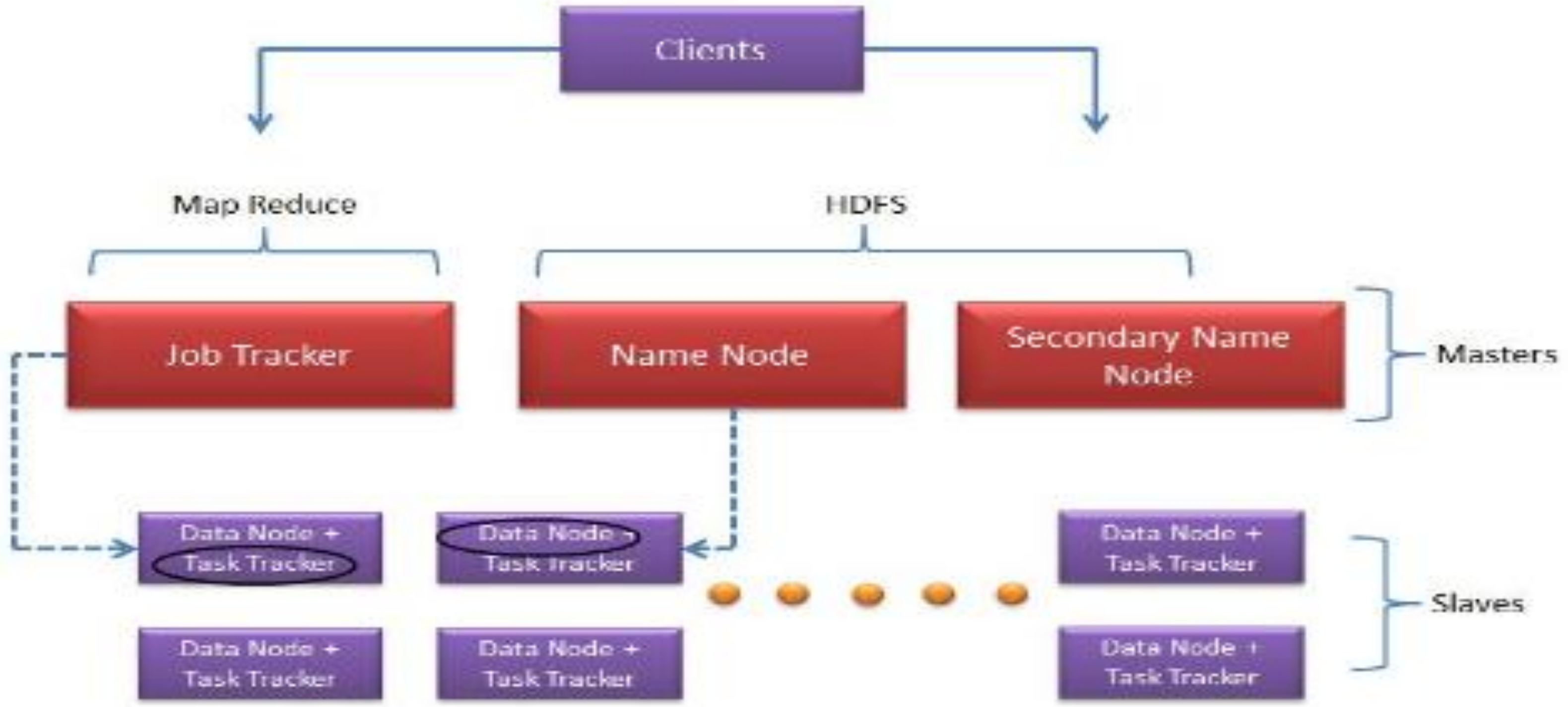
- ✓ a Map() procedure that performs filtering and sorting (such as sorting students by first name into queues, one queue for each name)
- ✓ a Reduce() procedure that performs a summary operation (such as counting the number of students in each queue, yielding name frequencies).

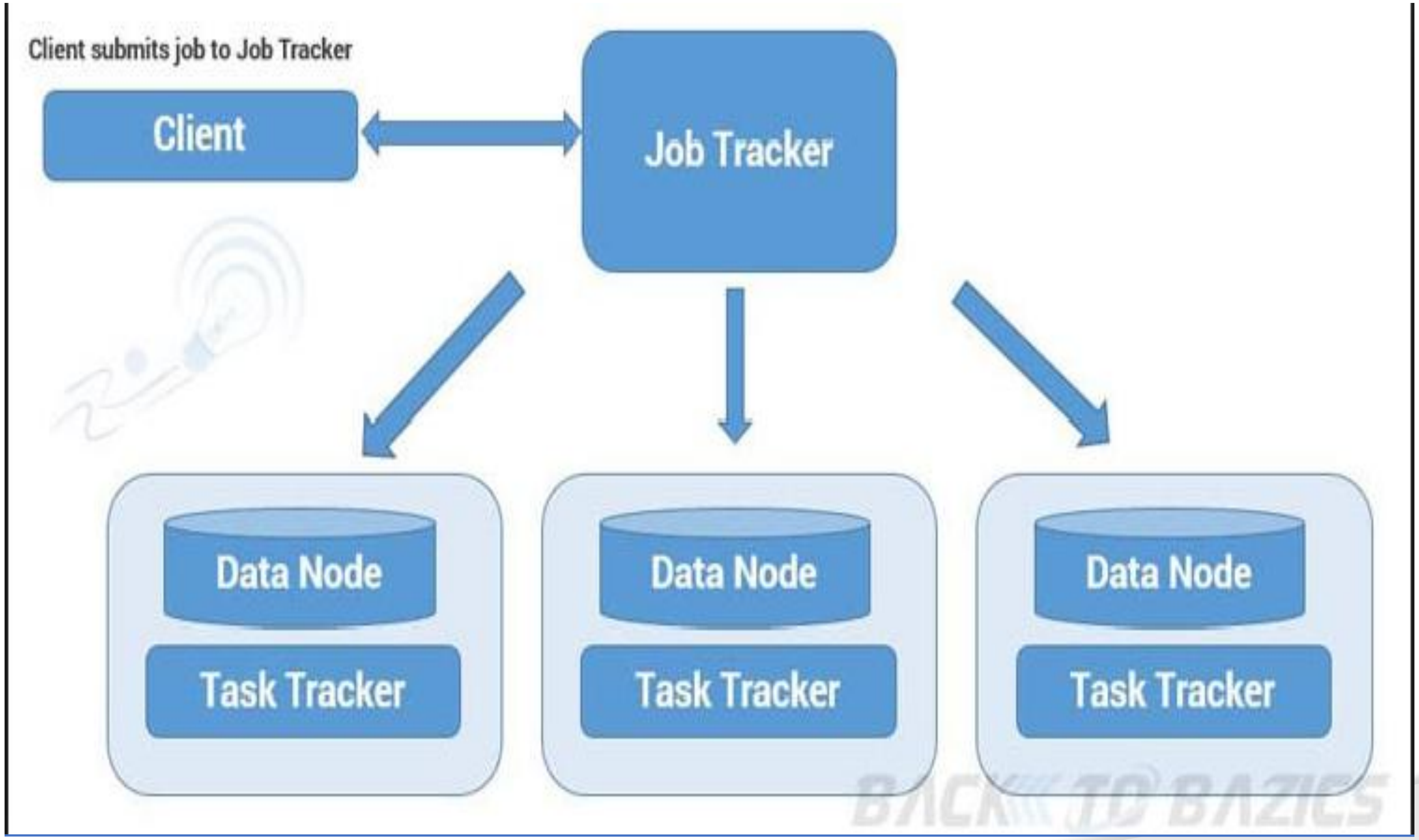

Input
Input
Input
Map
Map
Map
Shuffle/sort: group by map output keys
Reduce
Reduce
Output
Output

Few interesting facts about Map Reduce

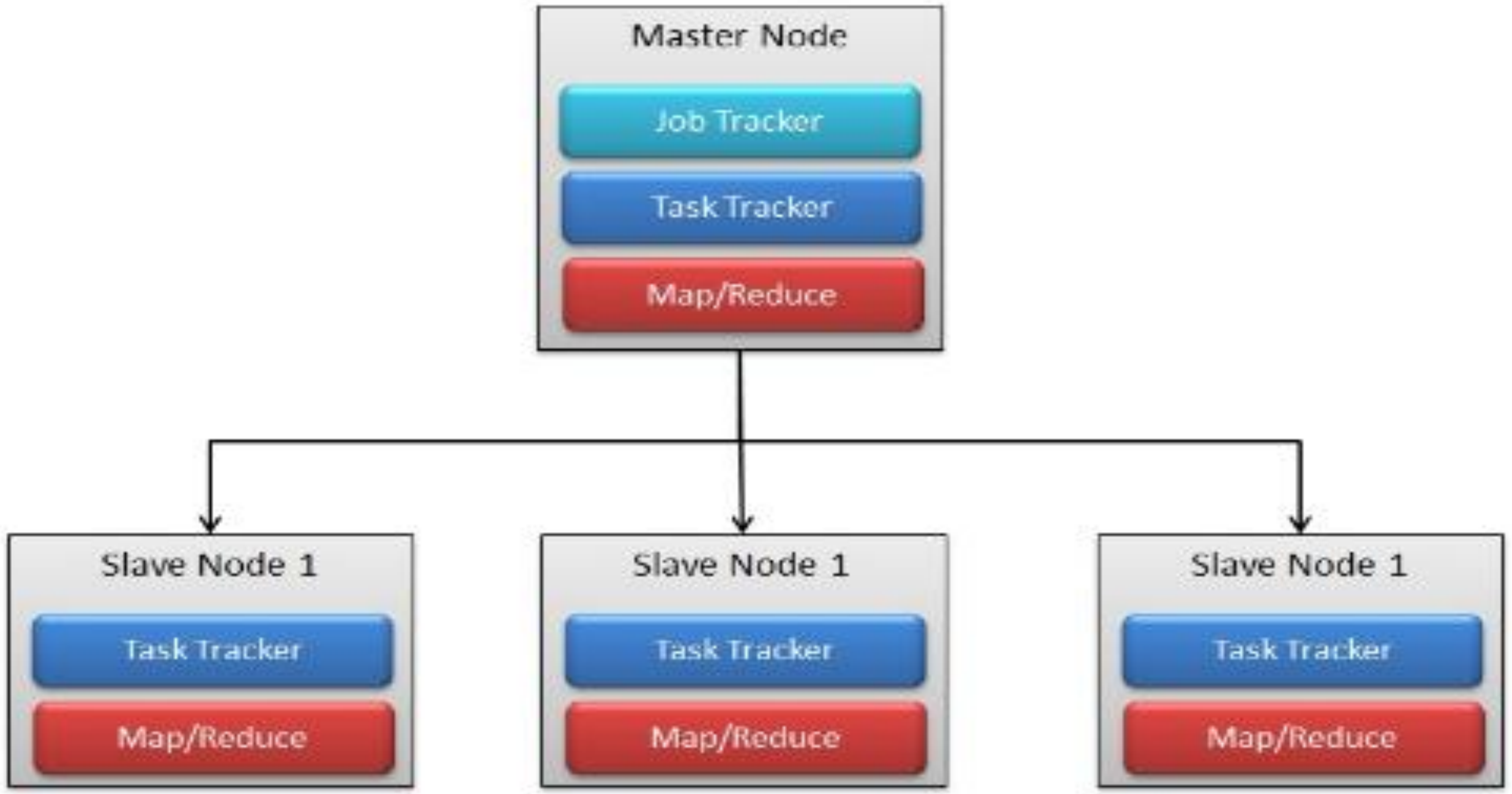
- ✓ Apache Hadoop Map-Reduce is an open source implementation of Google's Map Reduce Framework.
- ✓ Although there are so many map-reduce implementation like Dryad from Microsoft, Dicso from Nokia which have been developed for distributed systems but Hadoop being the most popular among them offering open source implementation of Map-reduce framework.
- ✓ Hadoop Map-Reduce framework works on Master/Slave architecture.

Hadoop Core Components

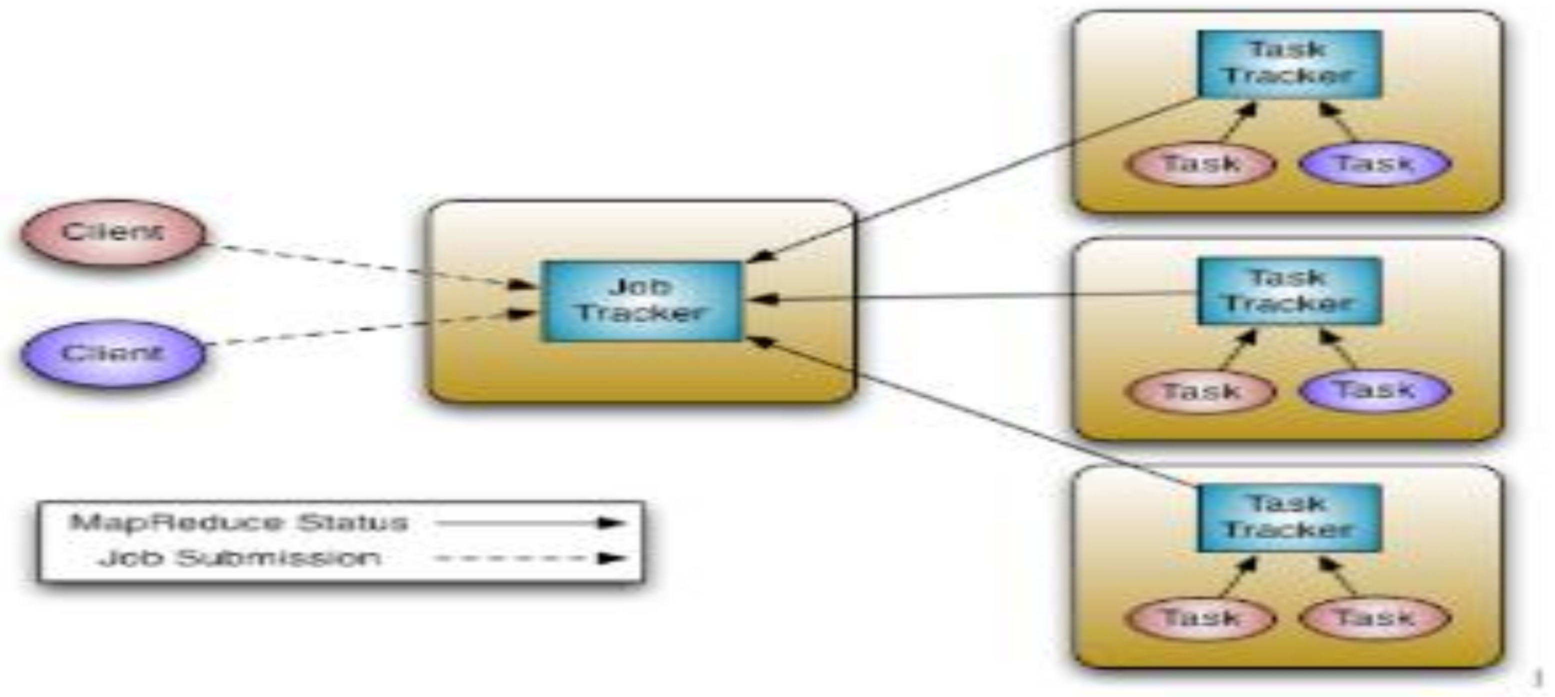



Client submits job to Job Tracker
Client
Job Tracker
Data Node
Task Tracker
Data Node
Task Tracker
Data Node
Task Tracker
BACK TO BAZICS

Map Reduce Architecture



Job Tracker:


Client
Client
Job Tracker
Task Tracker
Task
Task
Task Tracker
Task
Task
Task Tracker
Task
Task
MapReduce Status
Job Submission

- ✓ Job Tracker is the one to which client application submit map reduce programs(jobs).
- ✓ Job Tracker schedule clients jobs and allocates task to the slave task trackers that are running on individual worker machines(date nodes).
- ✓Job tracker manage overall execution of Map-Reduce job.
- ✓Job tracker manages the resources of the cluster like:
  - ▪Manage the data nodes i.e. task tracker.
  - ▪To keep track of the consumed and available resource.
  - ▪To keep track of already running task, to provide fault-tolerance for task etc.

Task Tracker

- ✓ Each Task Tracker is responsible to execute and manage the individual tasks assigned by Job Tracker.
- ✓ Task Tracker also handles the data motion between the map and reduce phases.
- ✓ One Prime responsibility of Task Tracker is to constantly communicate with the Job Tracker the status of the Task.
- ✓ If the Job Tracker fails to receive a heartbeat from a Task Tracker within a specified amount of time, it will assume the Task Tracker has crashed and will resubmit the corresponding tasks to other nodes in the cluster.

How Map Reduce Engine Works:

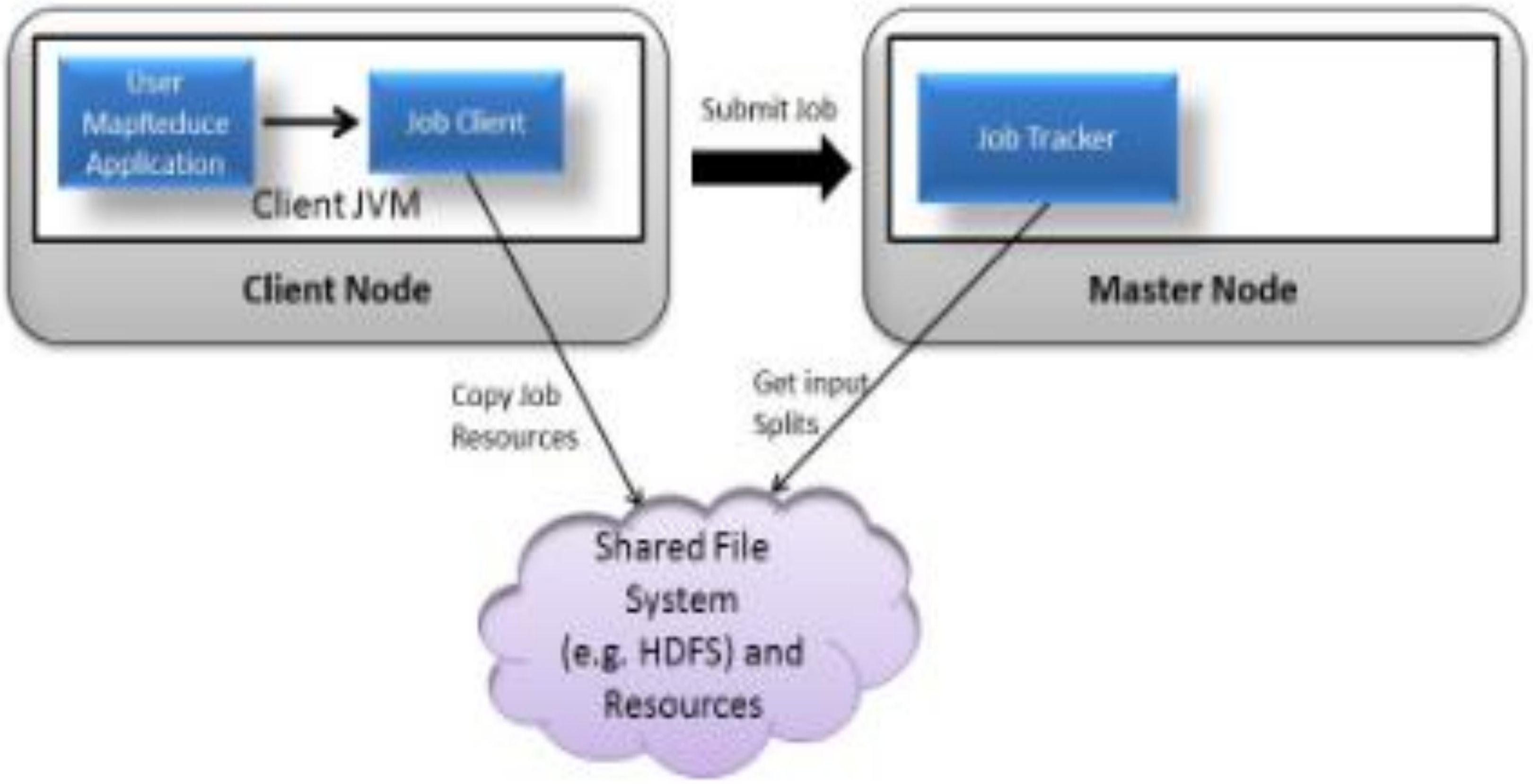
The entire process can be listed as follows:

- ✓Client applications submit jobs to the Job Tracker.
- ✓The Job Tracker talks to the Name Node to determine the location of the data
- ✓The Job Tracker locates TaskTracker nodes with available slots at or near the data
- ✓The Job Tracker submits the work to the chosen TaskTracker nodes.
- ✓The TaskTracker nodes are monitored. If they do not submit heartbeat signals often enough, they are deemed to have failed and the work is scheduled on a different TaskTracker.
- ✓A TaskTracker will notify the Job Tracker when a task fails. The Job Tracker decides what to do then: it may resubmit the job elsewhere, it may mark that specific record as something to avoid, and it may may even blacklist the TaskTracker as unreliable.
- ✓When the work is completed, the Job Tracker updates its status.
- ✓Client applications can poll the Job Tracker for information.

Step by Step Process:

1.Client submits MapReduce job to Job Tracker:

Whenever client/user submit map-reduce jobs, it goes straightaway to Job tracker. Client program contains all information like the map, combine and reduce function, input and output path of the data.

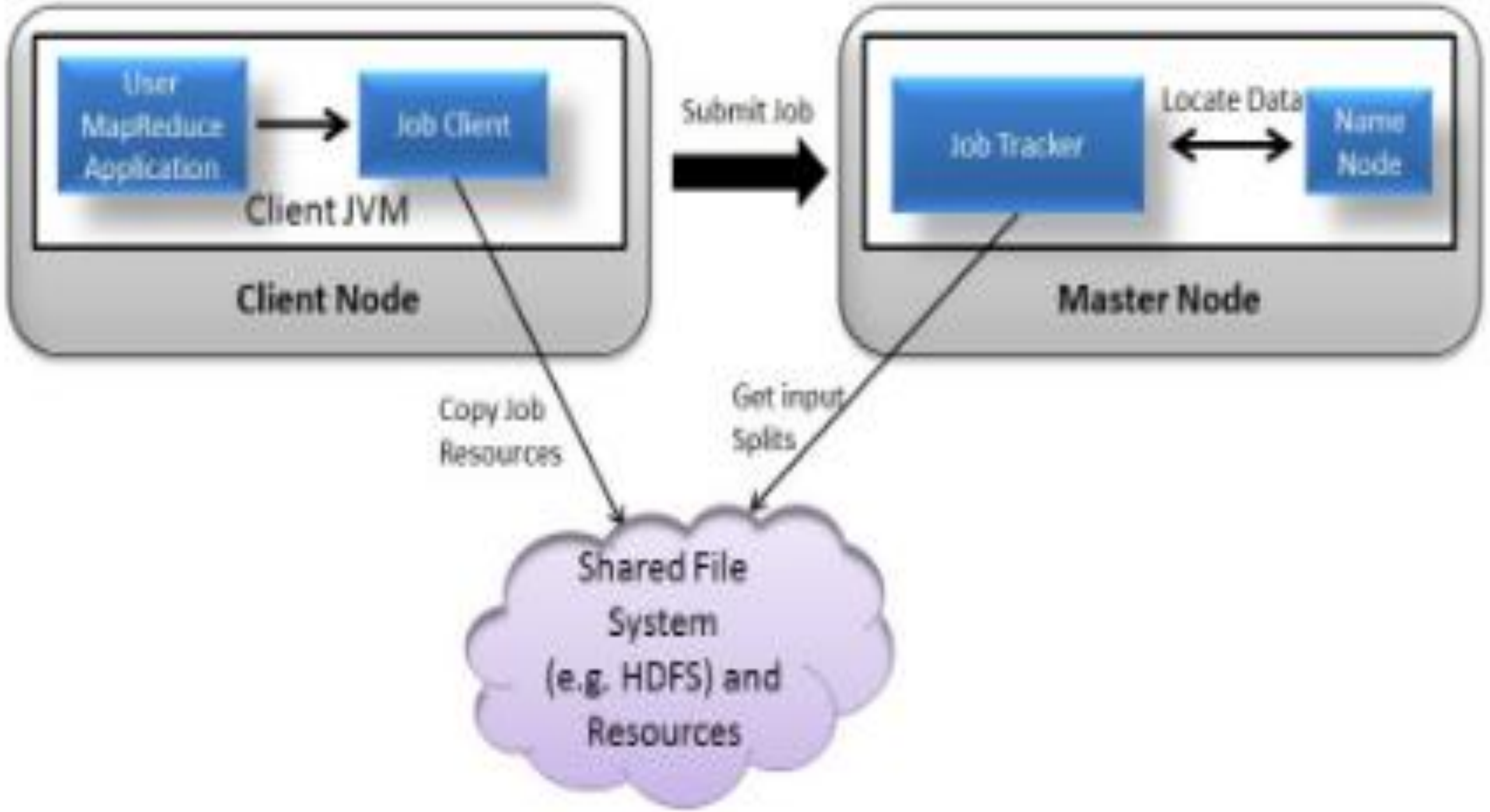

User MapReduce Application
Job Client
Client JVM
Client Node
Submit Job
Job Tracker
Master Node
Copy Job Resources
Get input Splits
Shared File System (e.g. HDFS) and Resources

2.Job Tracker Manage and Control Job:

✓The Job Tracker puts the job in a queue of pending jobs and then executes them on a FCFS(first come first serve) basis.

✓The Job Tracker first determine the number of split from the input path and assign different map and reduce tasks to each Task Tracker in the cluster. There will be one map task for each split.

✓Job tracker talks to the Name Node to determine the location of the data i.e. to determine the data node which contains the data.


User MapReduce Application
Job Client
Client JVM
Client Node
Submit Job
Job Tracker
Locate Data
Name Node
Master Node
Copy Job Resources
Get input Splits
Shared File System (e.g. HDFS) and Resources

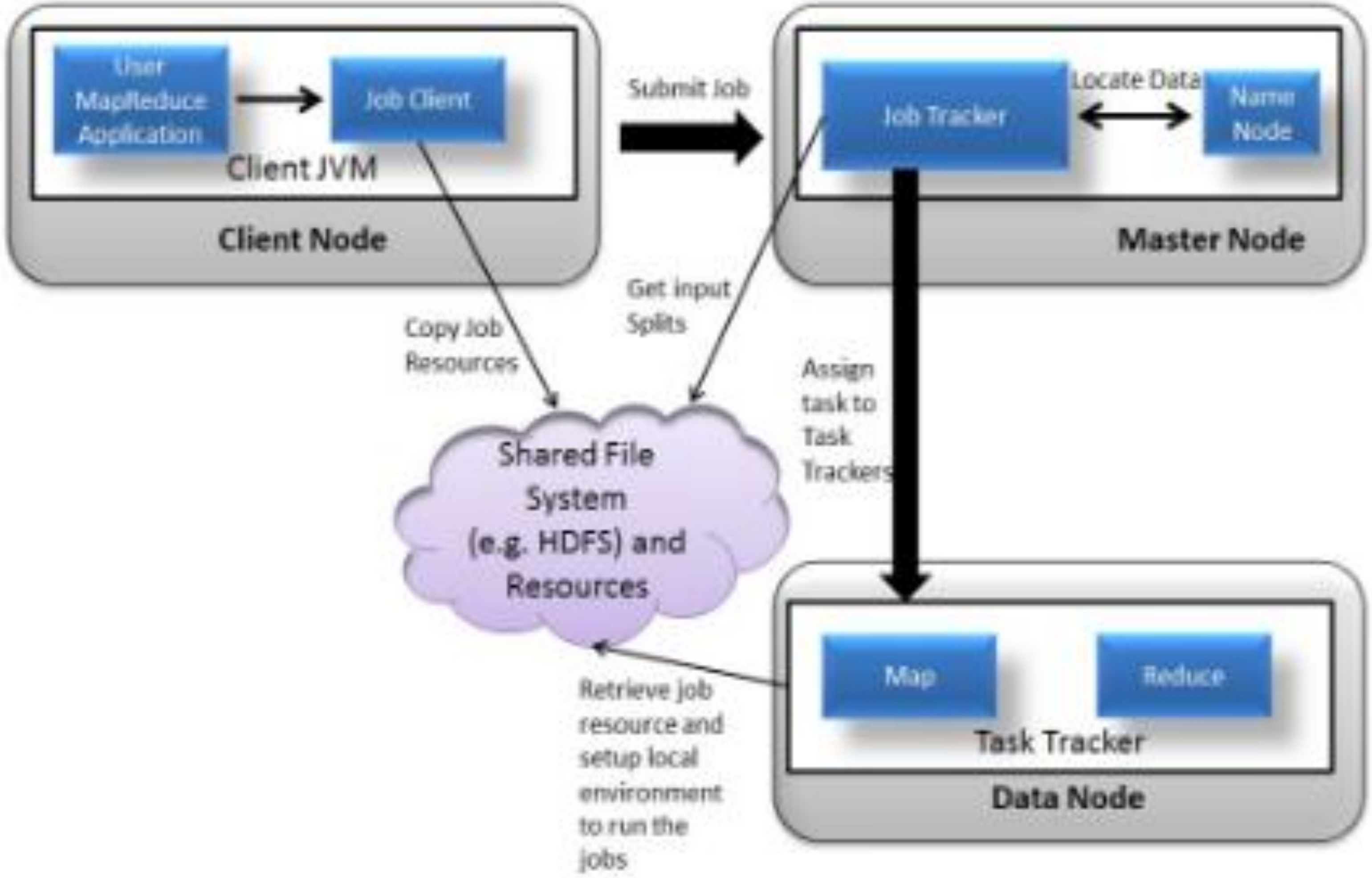
3.Task Assignment to Task Tracker by Job Tracker:

✓The task tracker is pre-configured with a number of slots which indicates that how many task(in number) Task Tracker can accept.

✓For example, a TaskTracker may be able to run two map tasks and two reduce tasks simultaneously.

✓When the job tracker tries to schedule a task, it looks for an empty slot in the TaskTracker running on the same server which hosts the datanode where the data for that task resides.

✓If not found, it looks for the machine in the same rack. There is no consideration of system load during this allocation.


User MapReduce Application
Job Client
Client JVM
Client Node
Submit Job
Job Tracker
Locate Data
Name Node
Master Node
Copy Job Resources
Get input Splits
Assign task to Task Trackers
Shared File System (e.g. HDFS) and Resources
Retrieve job resource and setup local environment to run the jobs
Map
Reduce
Task Tracker
Data Node

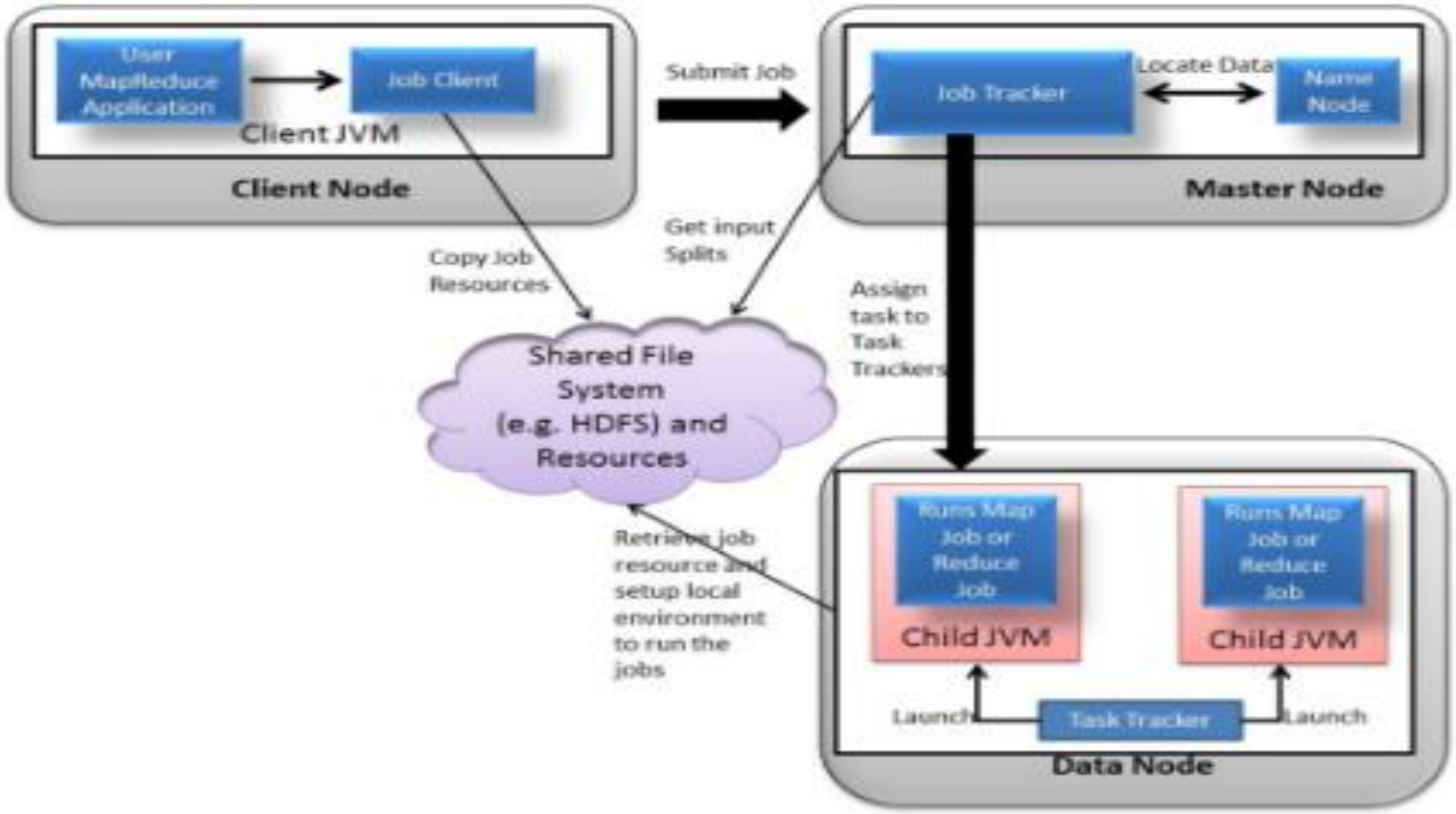
4.Task Execution by Task Tracker:

✓Now when the Task is assigned to Task Tracker, Task tracker creates local environment to run the Task.

✓Task Tracker need the resources to run the job. Hence it copies any files needed from the distributed cache by the application to the local disk, localize all the job Jars by copying it from shared File system to Task Tracker's file system.

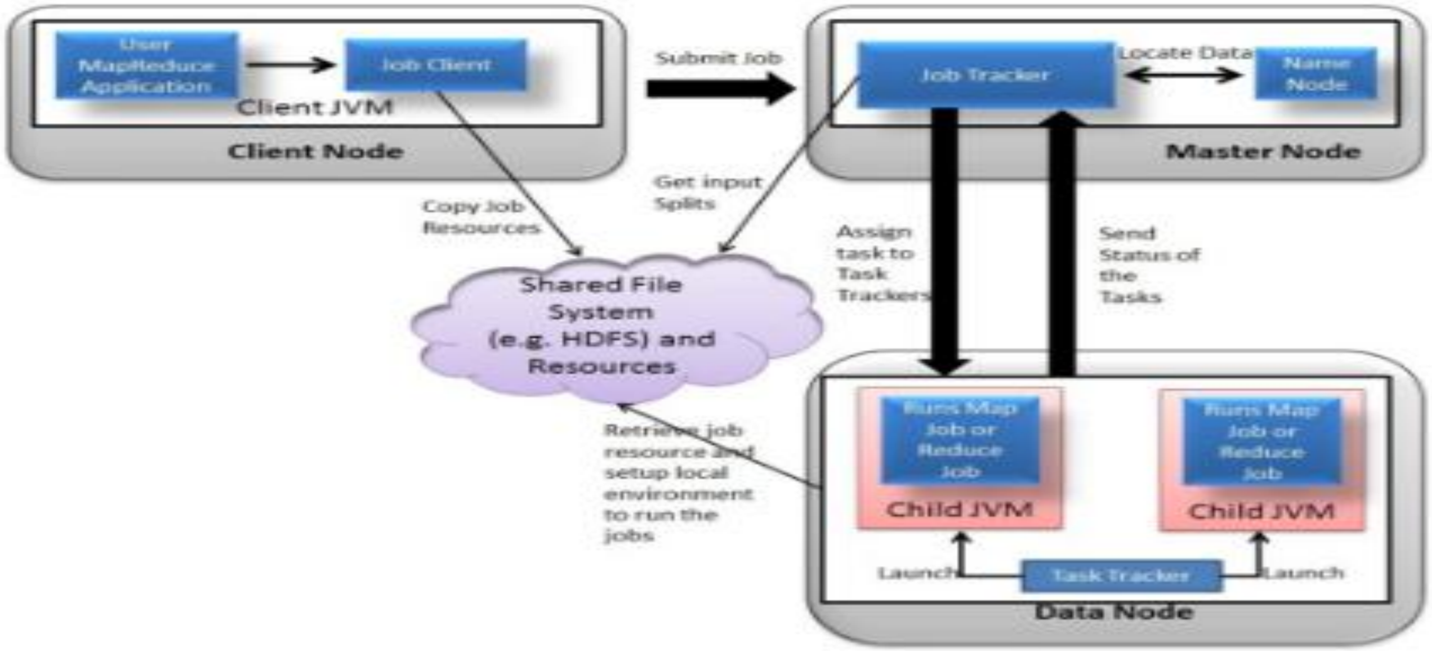
✓Task Tracker can also spawn multiple JVMs to handle many map or reduce tasks in parallel.

✓TaskTracker actually initiates the Map or Reduce tasks and reports progress back to the Job Tracker.


User MapReduce Application
Job Client
Client JVM
Client Node
Submit Job
Job Tracker
Locate Data
Name Node
Master Node
Copy Job Resources
Get input Splits
Assign task to Task Trackers
Shared File System (e.g. HDFS) and Resources
Retrieve job resource and setup local environment to run the jobs
Runs Map Job or Reduce Job
Child JVM
Runs Map Job or Reduce Job
Child JVM
Launch
Task Tracker
Launch
Data Node

5.Send notification to Job Tracker:

When all the map tasks are done by different task tracker they will notify the Job Tracker. Job Tracker then ask the selected Task Trackers to do the Reduce Phase


User MapReduce Application
Job Client
Client JVM
Client Node
Submit Job
Job Tracker
Locate Data
Name Node
Master Node
Copy Job Resources
Get input Splits
Assign task to Task Trackers
Send Status of the Tasks
Shared File System (e.g. HDFS) and Resources
Retrieve job resource and setup local environment to run the jobs
Runs Map Job or Reduce Job
Child JVM
Runs Map Job or Reduce Job
Child JVM
Launch
Task Tracker
Launch
Data Node

6.Task recovery in failover situation:

Although there is single TaskTracker on each node, Task Tracker spawns off a separate Java Virtual Machine process to prevent the TaskTracker itself from failing if the running job(process) crashes the JVM due to some bugs defined in user written map reduce function

7.Monitor Task Tracker :

✓The Task Tracker nodes are monitored. A heartbeat is sent from the Task Tracker to the Job Tracker every few minutes to check its status.

✓If Task Tracker do not submit heartbeat signals often enough, they are deemed to have failed and the work is scheduled on a different Task Tracker.

✓A Task Tracker will notify the Job Tracker when a task fails. The Job Tracker decides what to do then: it may resubmit the job elsewhere, it may mark that specific record as something to avoid, and it may even blacklist the Task Tracker as unreliable.

8. Job Completion:

✓When the work is completed, the Job Tracker updates its status.

✓Client applications can poll the Job Tracker for information.

What is Parallel Programming?

✓In a parallel program, the processing is broken up into several parts and each of the part gets executed concurrently.

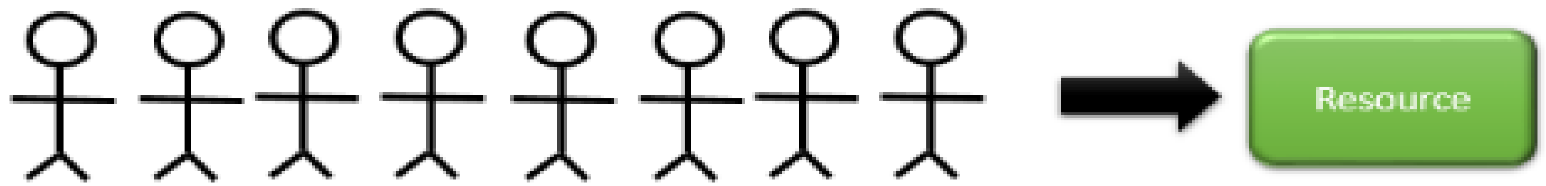
✓Important point to note here is that “**Each part should have no dependency on other parts, No communication should be needed to other parts when executing all parts in parallel manner.**”

✓Suppose you are writing a program and your program needs to access some common data.

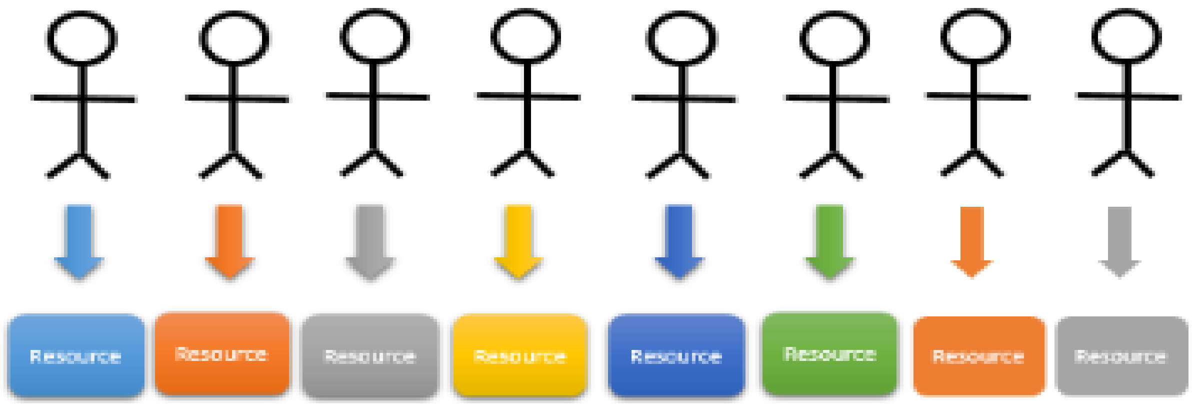
✓Now when you are executing several instances of that program at the same time, there could be conflicts like one instance of program is changing some data while other instance is reading it.

✓ So, you have to handle these cases in your program code. Here you're doing **Concurrency**.

✓But if your program instance is working on some data which no other instance needs, then you're doing here **Parallelism**.



A concurrent solution to n event handling problem:
using a queue to resolve conflicts



A parallel solution to a computational problem:
avoiding unnecessary conflicts

✓In other words - Those programs where data is not dependent on each other and is isolated can be executed in parallel programming manner.

Parallel programs are not only faster but they can also be used to solve problems on large datasets using non-local resources.

MapReduce Works on Parallel Programming Concept

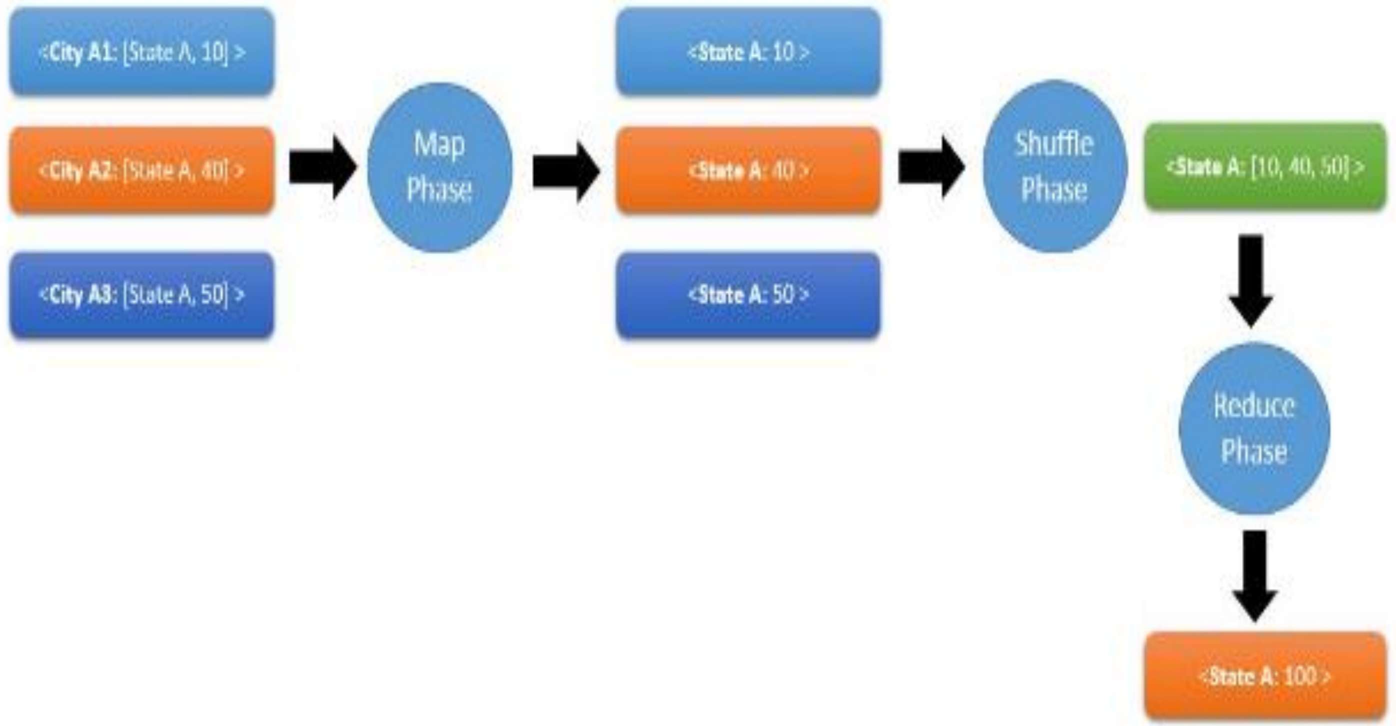
3 Stages of MapReduce – Map, Shuffle and Reduce:

MapReduce programming model comes with 3 simple stages.

The Shuffle part is done automatically by Hadoop, we just need to implement the Map and Reduce parts.
Input for **Map** stage is always a <Key, Value> pair and it produces a set of intermediate key/value pairs as an output.

✓All intermediate values associated with the same intermediate key are grouped together and passed through to the reduce stage.

✓The **Reduce stage** concepts an intermediate key and a set of values for that key. It merges together these values to form a possibly smaller set of values.



<City A1: [State A, 10] >
<City A2: [State A, 40] >
<City A3: [State A, 50] >
Map Phase
<State A: 10 >
<State A: 40 >
<State A: 50 >
Shuffle Phase
<State A: [10, 40, 50] >
Reduce Phase
<State A: 100 >

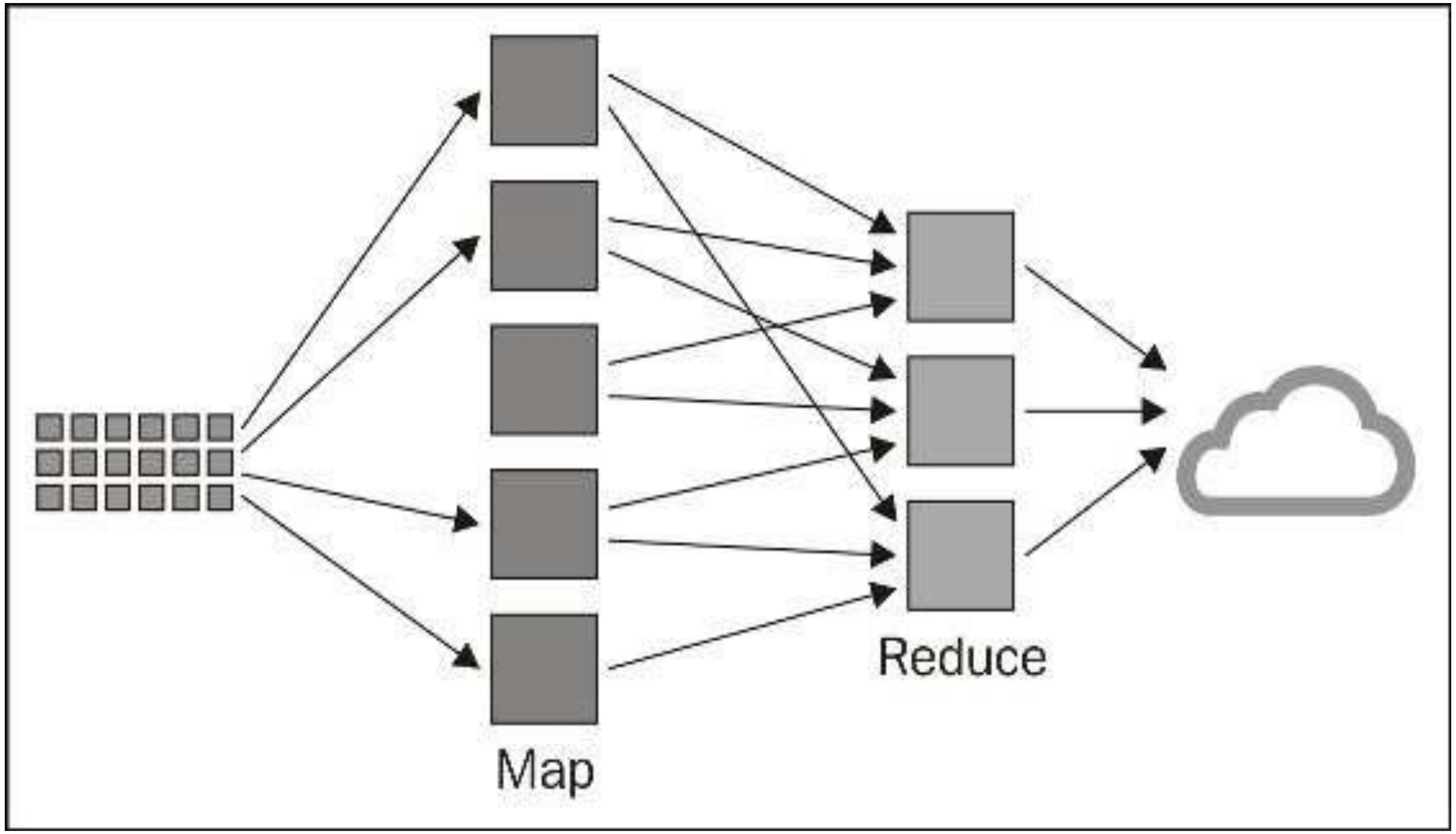
Map Reduce:

✓It is a programming model for processing 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.**

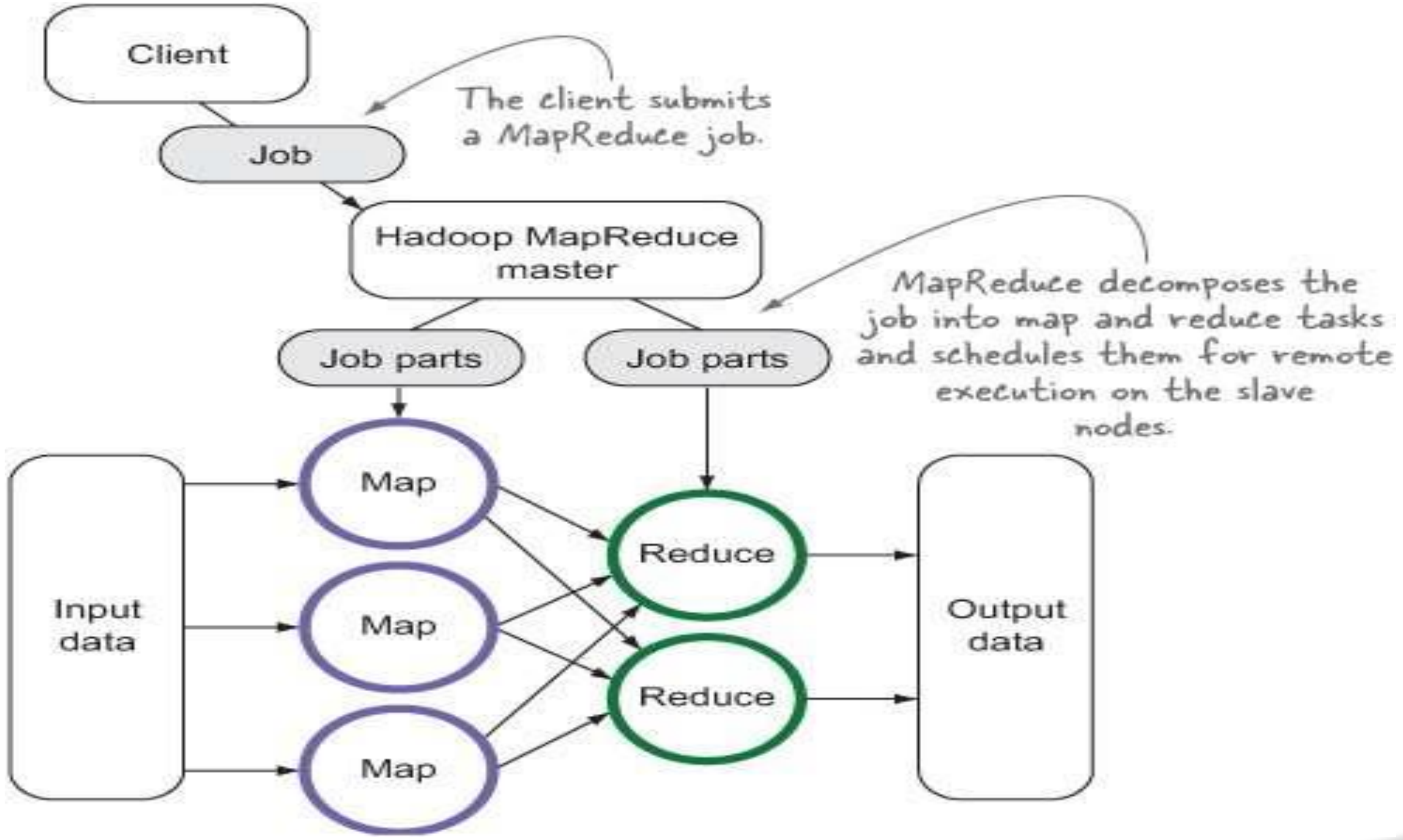
✓In Map reduce(MR), computation is near the data i.e. the computational tasks are directly performed on the data wherever it happens to reside, rather than the previous practices of first copying and aggregating raw data into a single repository before processing it. These older practices simply won’t work when the amount of data is beyond terabytes.

✓In MR, instead of moving huge volumes of raw data across a network, only code is sent over the network.

Map Reduce Phases:



Map Reduce Job:


Client
The client submits a MapReduce job.
Job
Hadoop MapReduce master
MapReduce decomposes the job into map and reduce tasks and schedules them for remote execution on the slave nodes.
Job parts
Job parts
Input data
Map
Map
Map
Reduce
Reduce
Output data

Advantage of MR

Same program can run on a small dataset as well as a huge dataset (easy scaling) with just a simple configuration change.

Input & Output Forms of MR:

✓In order for mapping, reducing (and other phases like – combining, partitioning, and shuffling) to seamlessly work together, we need to agree on a common structure for the data being processed. Hence, we use key/value pairs as input &

	INPUT	OUTPUT
Map()	<K1,V1>	list(<K2,V2>)
Reduce()	<K2,list(V2)>	list(<K3,V3>)

Map Phase:

✓The map function takes a key/value pair (<k1,v1>) and it produces zero or more key/value pairs(list(<k2,v2>)) for one input pair.

map(k1, v1) -->list(k2, v2)

✓A given input file will be processed in parallel by several Mappers (as seen in figure-2). There will be one mapper task for each block of data of the input file. Each mapper will process all the key/value pairs in the block that it is running on.

✓So, number of mappers that run in MR job is equal to the total number of blocks that the input file has.

Reduce Phase:

✓After all the mappers complete execution, then the Reduce phase will start. The reduce function is called once per unique (each) map output key. i.e. it will receive **a key and a list of corresponding values of that key (emitted by all the mappers).**

✓Like the map function, the reducer also emits zero or more key/value pairs. Reducer output is the final output of a MR Job and it can be written to flat files in HDFS

reduce(k2, list(v2)) -->list(k3, v3)

✓Number of reducers is configurable (can be chosen by the user).

✓Reducer phase is optional (You would not have a reducing phase when you don’t need any kind of aggregation)

Shuffle & Sort Phase:

The shuffle and sort phase occurs between Map & Reduce phases. By default, hadoop framework handles this phase. It is responsible for two primary activities:

1. Determining the reducer that should receive the map output key/value pair(called
partitioning).

2. Sorting all the input keys for a given reducer.

Java API to MapReduce:

✓The Java API to MapReduce is exposed by the

org.apache.hadoop.mapreduce package.

✓**Writing** a MapReduce program, at its core, is a matter of subclassing Hadoop-provided Mapper and Reducer base classes, and overriding the map() and reduce() methods with our own implementation.

The Mapper Class:

✓To implement a Mapper, we will subclass the Mapper base class (which is present in org.apache.hadoop.mapreduce package) and override the map()method, as follows:

```
class Mapper<K1, V1, K2, V2>

        {
                void map(K1 key, V1 value, Mapper.Context
context)                throws
IOException,InterruptedException

                ...
        }
```

✓The **Mapper class is defined in terms of the key/value input and output types (K1, V1, K2, V2),** and then the **map() method takes an input key/value pair(K1 key, V1 value) as its parameters.**

✓The other parameter is an instance of the **Context class(Mapper.Context context) that provides** various mechanisms to communicate with the Hadoop framework, one of which is to output the results of a map method.

- ✓Notice that the map method only refers to a **single instance of K1 and V1 key/value pairs.**
- ✓ **This** is a critical aspect of the MapReduce model in which you write classes that process a single record, and the framework is responsible for all the work required to turn an enormous dataset into a stream of key/value pairs. You will never have to write map or reduce classes that try to deal with the full dataset.
- ✓Some other methods present in Mapper class are – setup(), cleanup() & run().All these 3 methods take Context class reference as their input parameter

```
public void map(Object key, Text value, Context context ) throws
                                        IOException, InterruptedException
             {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens())
                    {
                          word.set(itr.nextToken());
                          context.write(word, one);
                    }
             }
```

sample input the first map emits:

< Hello, 1> < World, 1> < Bye, 1> < World, 1>

The second map emits:

< Hello, 1> < Hadoop, 1> < Goodbye, 1> < Hadoop, 1>

job.setCombinerClass(IntSumReducer.class);

The output of the first map:

< Bye, 1> < Hello, 1> < World, 2>`

The output of the second map:

The Reducer Class:

✓The Reducer base class works similar to the Mapper class and usually requires its subclasses to override a single reduce() method. Here is the class definition in brief:

```
public class Reducer<K2, V2, K3, V3>
        {
                void reduce(K2 key, Iterable<V2>
values,
                Reducer.Context context)throws
IOException,
        InterruptedException
                ...
        }
```

- ✓The Reducer class is defined in terms of the key/value input and output types (K2, V2, K3, V3), and then the reduce() method takes a single key and its associated list of values (K2 key,Iterable<V2> values) as its parameters.

- ✓The other parameter is an instance of the Context class(Reducer.Context context) that provides various mechanisms to communicate with the Hadoop framework, one of which is to output the results of a reduce method.

- ✓Some other methods present in Reducer class are – setup(), cleanup() & run(). All these 3 methods take Context class reference as their input parameter.

- Thus the output of the job is:

```
< Bye, 1>
< Goodbye, 1>
< Hadoop, 2>
 < Hello, 2>
 < World, 2>
```

The Driver Class:

✓The **Driver class communicates with Hadoop framework and specifies the configuration** elements needed to run a MR job like – which Mapper & Reducer class to use, where to find input data and its format, where to find output data and its format.

✓The driver logic usually exists in the main method of the class written to encapsulate a MR job.

✓There is no default parent Driver class to subclass.

```
public static void main(String[] args) throws Exception
        {
            // Create a configuration Object that is used to set
            other
            options
            Configuration conf = new Configuration();
            GenericOptionsParser args = new
            GenericOptionsParser(conf,
            args).getRemainingArgs();
            // Create the Object representing the Job
            Job job = Job.getInstance(conf, "word count");
            // Set the name of the main class in the job jarfile
            job.setJarByClass(ExampleDriver.class);
// Set the Mapper Class
job.setMapperClass(ExampleMapper.class);
// Set the Combiner Class
job.setCombinerClass(ExampleReducer.class);
// Set the Reducer Class
job.setReducerClass(IntSumReducer.class);
// Set the types for the final output key and value
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
//set input and output file paths
FileInputFormat.addInputPath(job, new Path(args(0)));
FileOutputFormat.setOutputPath(job, new
Path(args(1)));
//Execute the job and wait for it to complete
System.exit(job.waitForCompletion(true) ? 0 : 1);
    }
```

The Combiner Class:

In a MR job, data gets transferred between Map and Reduce phases. It consumes lot of network bandwidth. We can minimize this bandwidth by using a combiner function.

A Combiner runs on MapReduce map() output and sends its output to reduce(). Combiner does local aggregation. Hence it is called as a mini-reducer.

Combiner function is an optimization and hence there is no guarantee on how many times it will be called.

Combiner’s Contract:

✓A Combiner can be used only if the algorithm satisfies commutative and associative properties.

✓eg: You can use combiner when you are trying to find the maximum value among a list of values but not when you want to find the average value of a list of value

max(10, 20, 25, 30) = 30
max(max(10,20), max(25,30)) = max(20,30)=30
whereas, avg(10, 20,25,30,35) = 85/4=21.25
& avg(avg(10,20),avg(25,30,35))=avg(15,30)=22.5

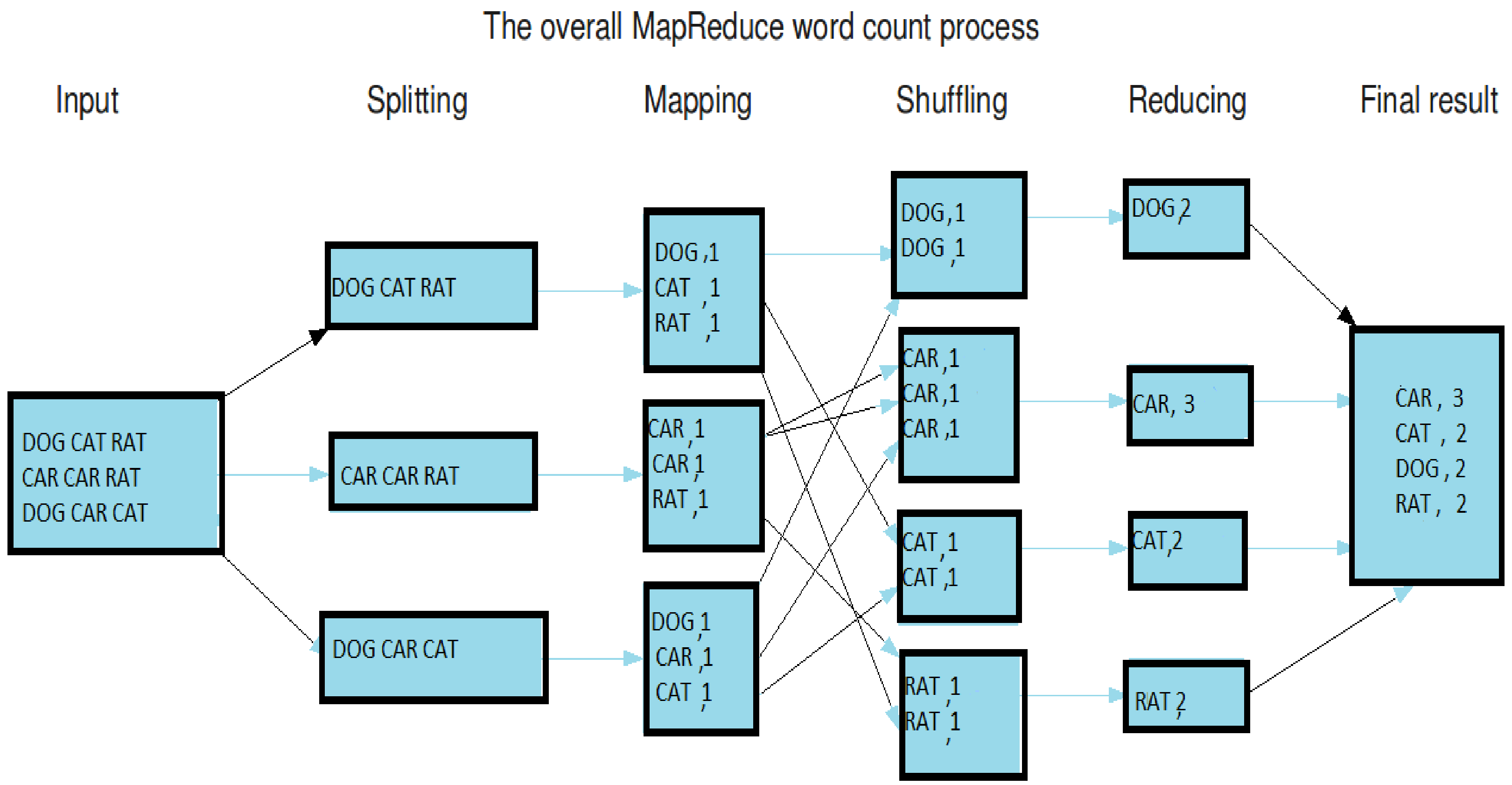
Partitioning:

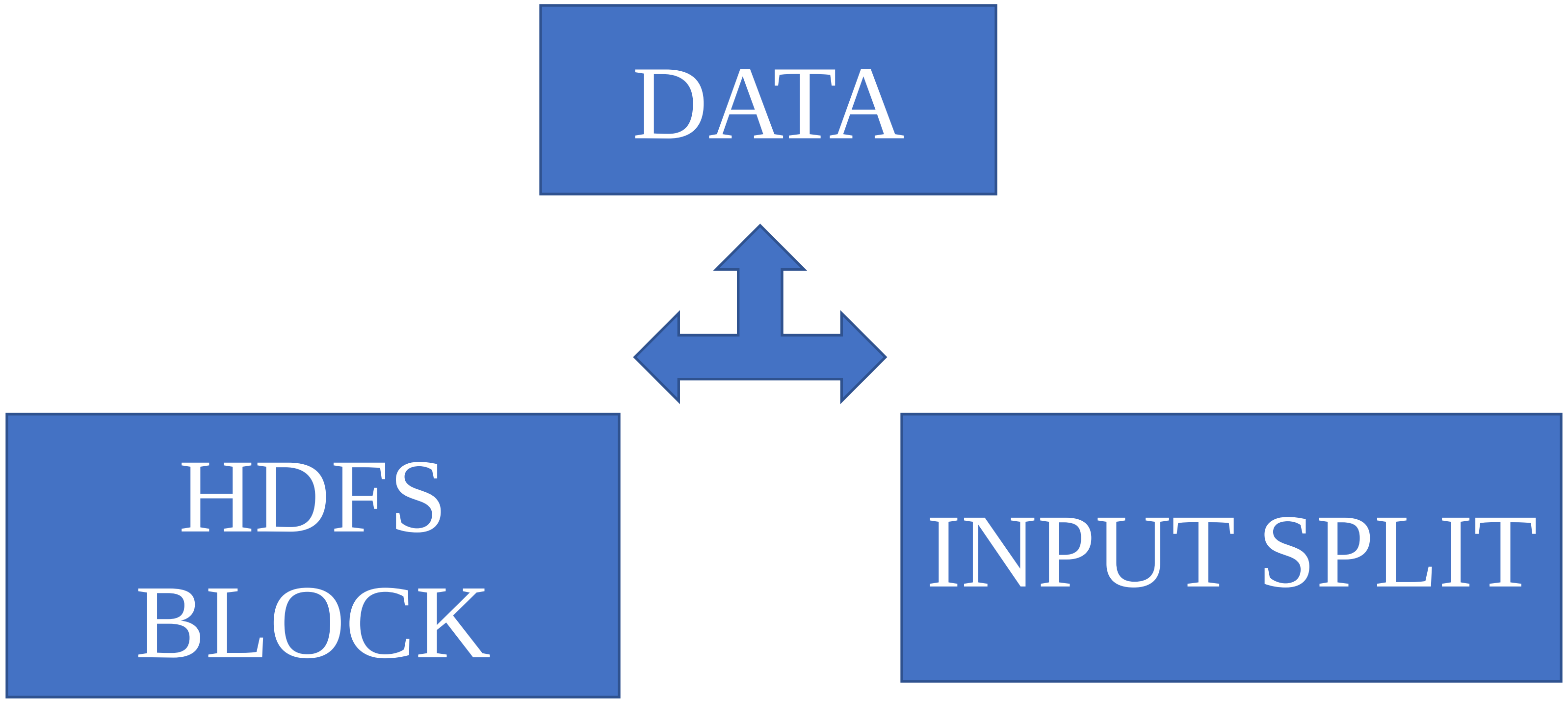
When there are multiple reducers, the map tasks partition their output, each creating one partition for each reduce task. There can be many keys (and their associated values) in each partition, but the records for any given key are all in a single partition.

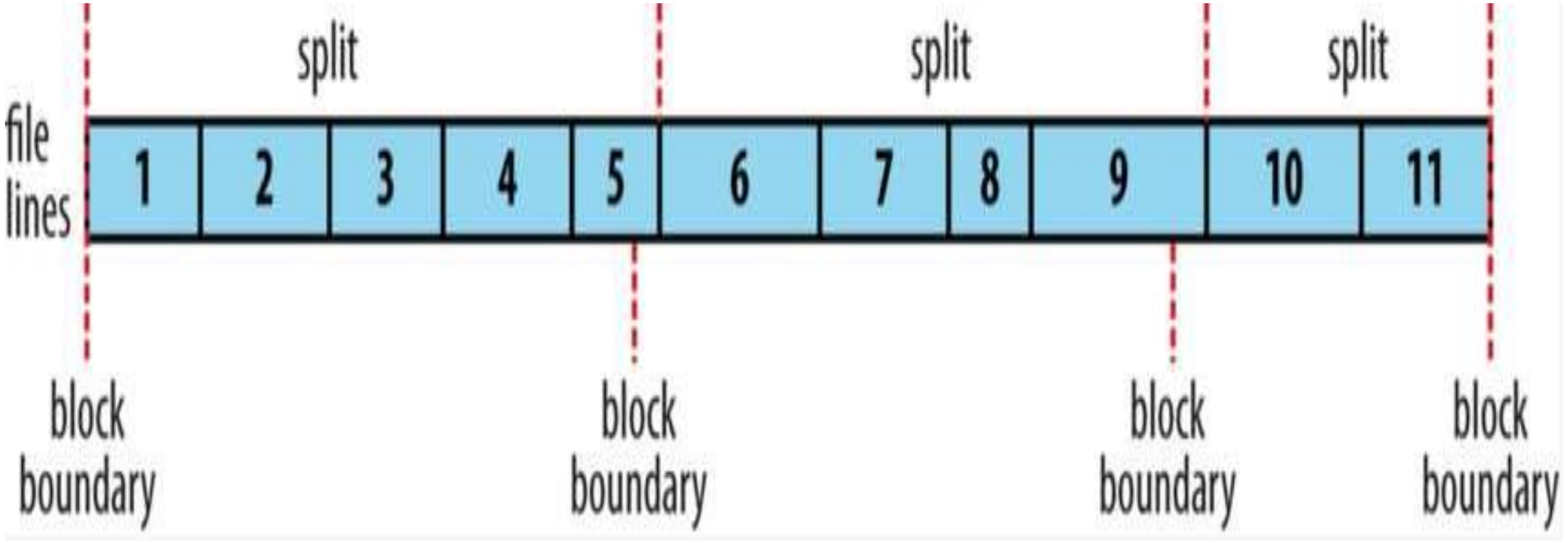
The partitioning can be controlled by a user-defined partitioning function, but normally the default partitioner—which buckets keys using a hash function—works very well. This functionality is provided by the **HashPartitioner class within the org.apache.hadoop.mapreduce.lib.partition package, but it's** necessary in some cases to provide a custom subclass of **Partitioner with application-specific** partitioning logic.

- Notice that the **getPartition function takes the key, value, and number of**
- partitions as parameters, any of which can be used by the custom partitioning logic.
- A custom partitioning strategy would be particularly necessary if, for example, the data
- provided a very uneven distribution when the standard hash function was applied. Uneven partitioning can result in some tasks having to perform significantly more work than others, leading to much longer overall job execution time

Map Reduce word count walkthrough




DATA
HDFS BLOCK
INPUT SPLIT


split
split
split
file lines
1
2
3
4
5
6
7
8
9
10
11
block boundary
block boundary
block boundary
block boundary

✓The need for Input Split is – In an HDFS block, we divide the data based on size (and not on the number of lines), so, the records/lines would not be cut neatly (the last record in a block might be split in half).

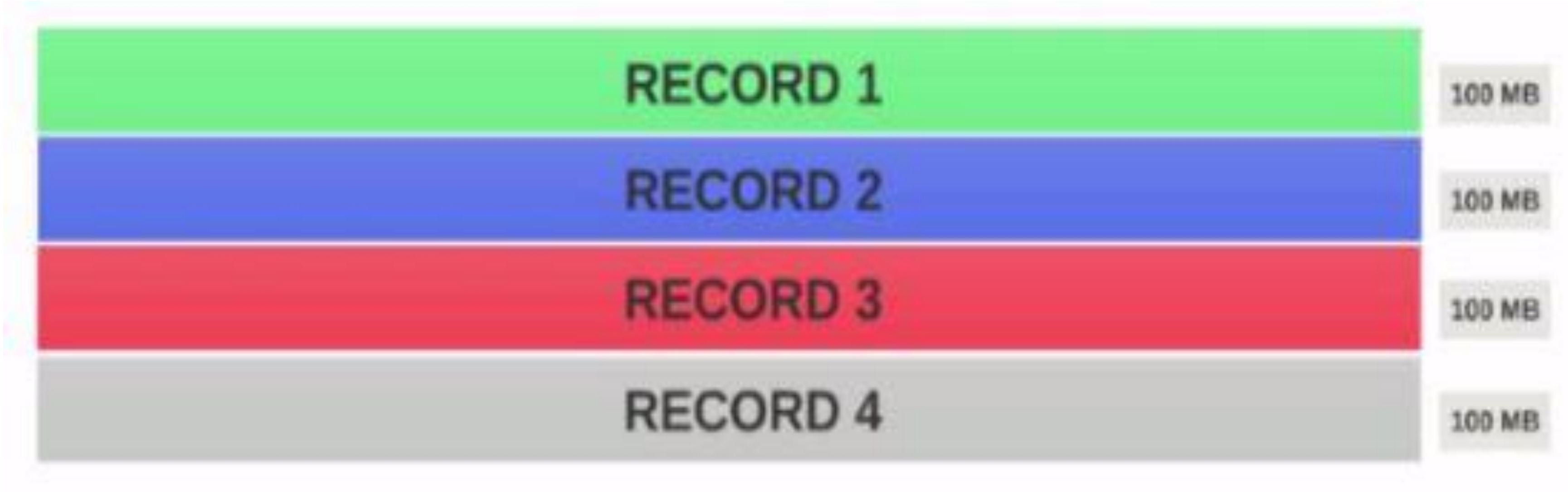
✓If we send this block of data for a mapper to process, the last record that it will receive would not be a complete record.

✓Hence, to overcome such issues, an Input Split would be formed which would have a record till the end even if that record crosses the block size

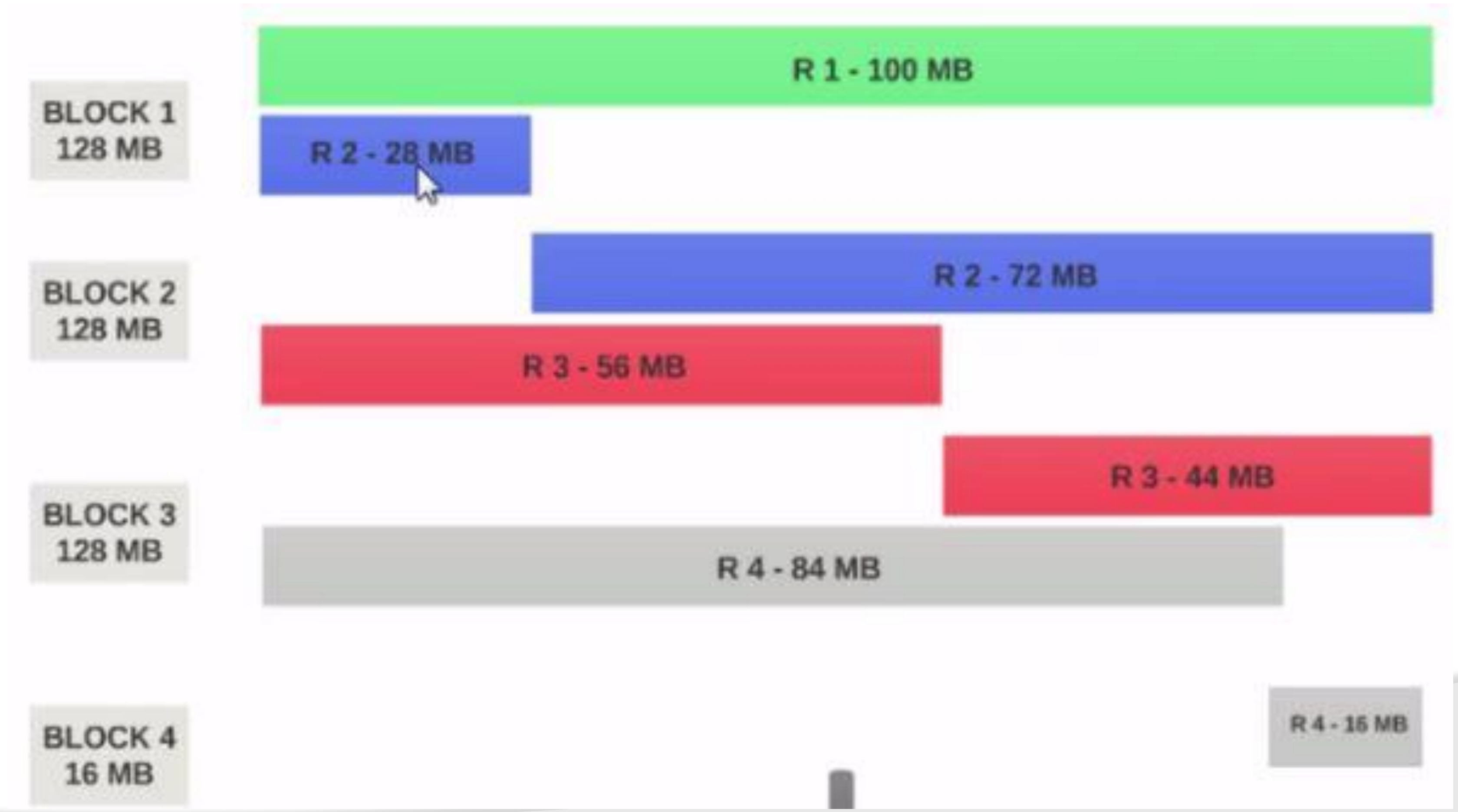
EXAMPLE:

- Assume we have a file of **400MB** with consists of **4 records**

(**e.g** : csv file of 400MB and it has 4 rows, 100MB each)



•If the HDFS **Block Size** is configured as **128MB**, then the 4 records will not be distributed among the blocks evenly. It will look like this



✓**Block 1** contains the entire first record and a 28MB chunk of the second record.

✓If a mapper is to be run on **Block 1**, the mapper cannot process since it won't have the entire second record.

✓This is the exact problem that **input splits** solve. **Input splits** respects logical record boundaries

- ✓Therefore the **input split 1** should have both the record 1 and record 2. And input split 2 will not start with the record 2 since record 2 has been assigned to input split 1. Input split 2 will start with record 3.

- ✓This is why an input split is only a **logical chunk** of data. It points to start and end locations with in blocks.

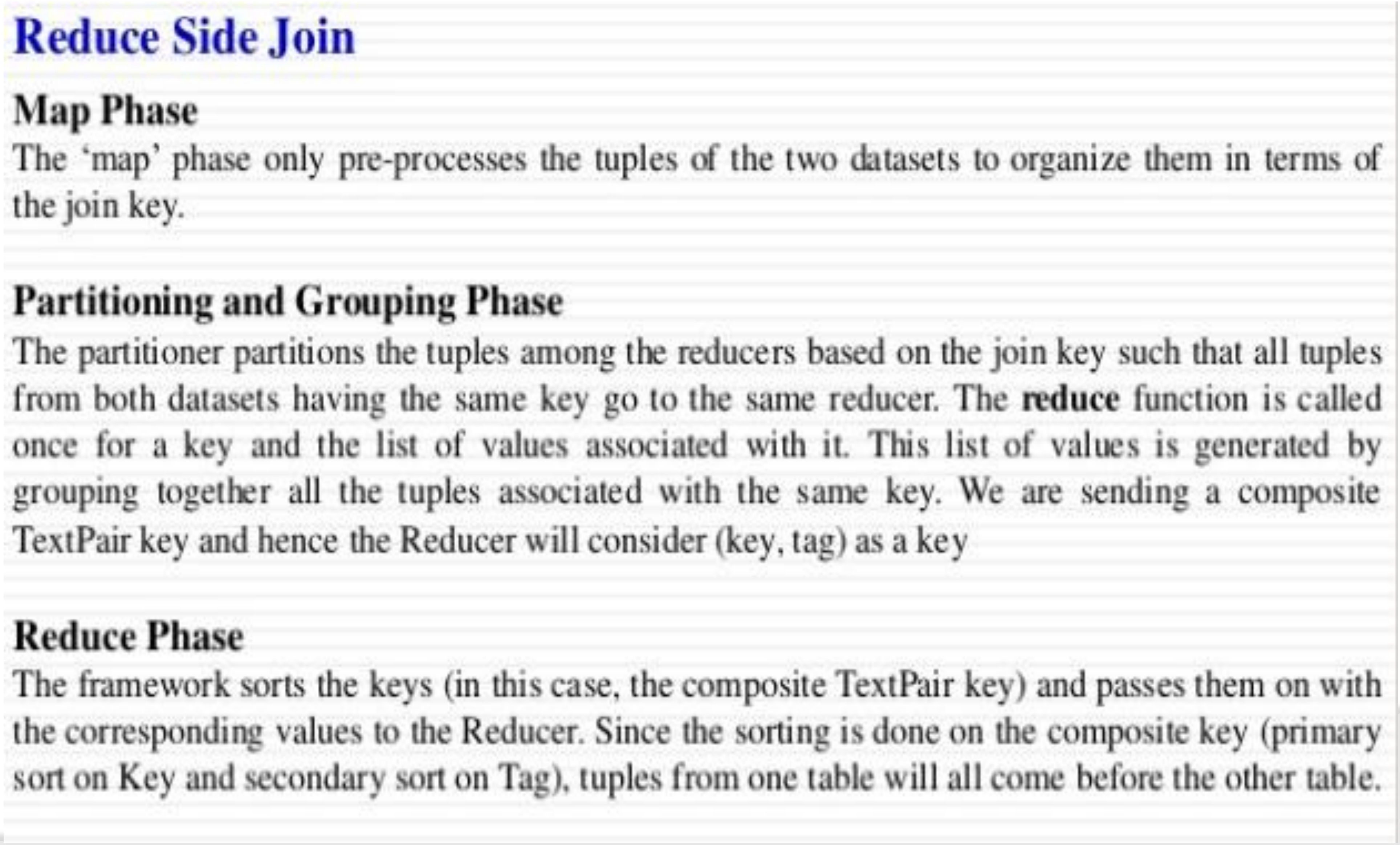
Data Locality Optimization:

✓A MR job is split into several map & reduce tasks and Map tasks run on the Input splits.

✓Ideally, the task JVM to run a map task would be initiated in the node where the split/block of data exists.

✓While in some scenarios, JVMs might not be free to accept another task.

✓In Such case, Task Tracker JVM will be initiated at a different location (can be on another Node in the same Rack or a different Rack)



Reduce Side Join

Map Phase

The ‘map’ phase only pre-processes the tuples of the two datasets to organize them in terms of the join key.

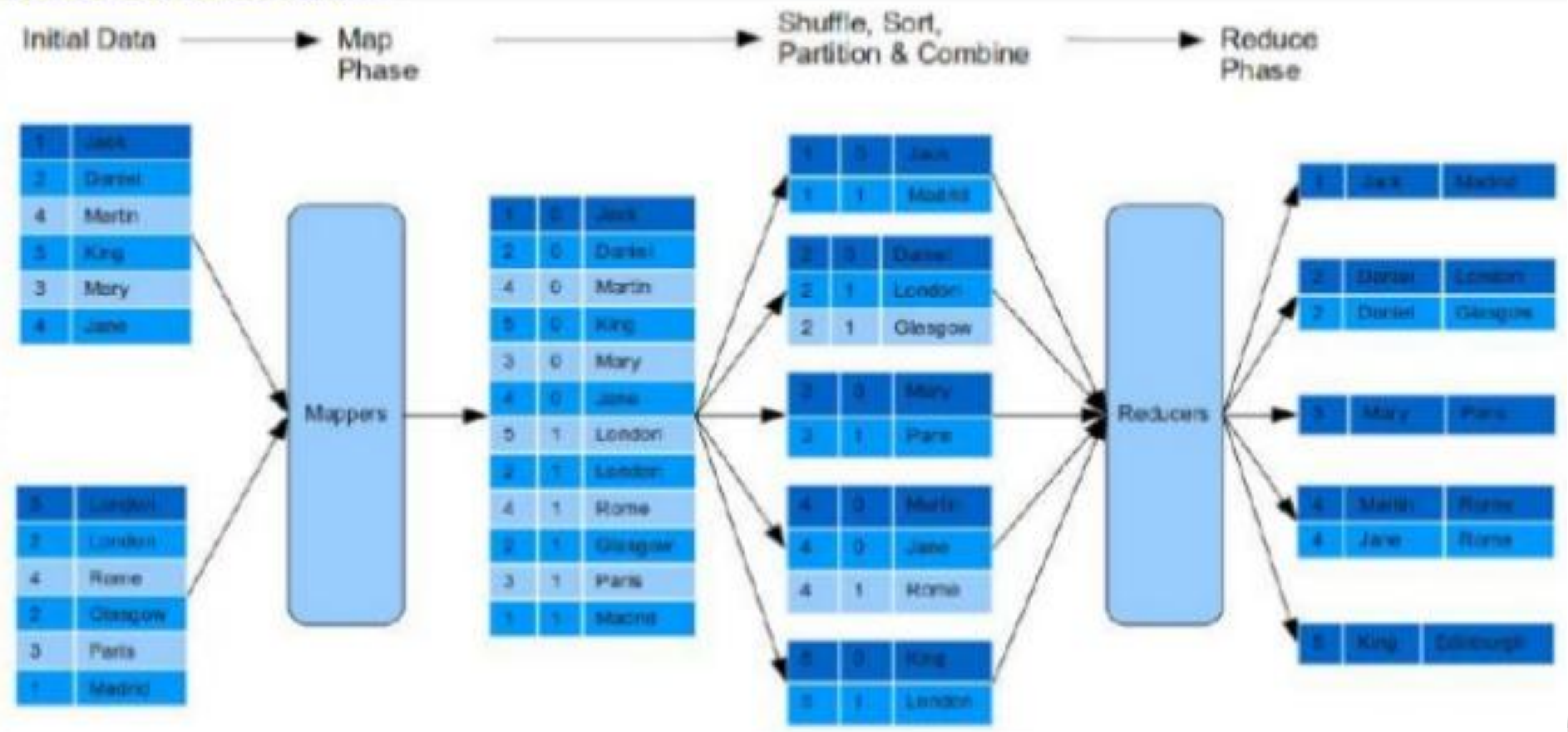
Partitioning and Grouping Phase

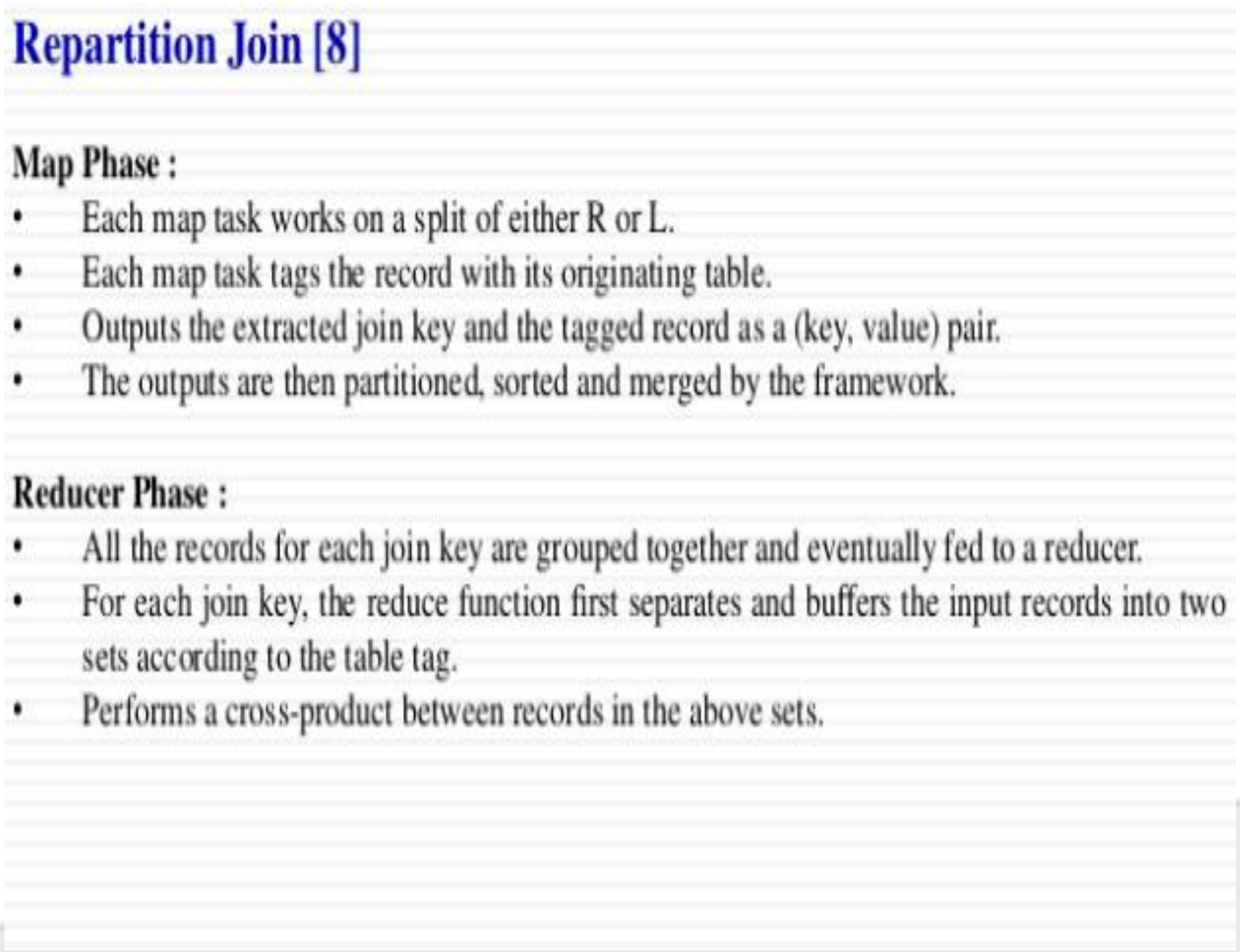
The partitioner partitions the tuples among the reducers based on the join key such that all tuples from both datasets having the same key go to the same reducer. The **reduce** function is called once for a key and the list of values associated with it. This list of values is generated by grouping together all the tuples associated with the same key. We are sending a composite TextPair key and hence the Reducer will consider (key, tag) as a key

Reduce Phase

The framework sorts the keys (in this case, the composite TextPair key) and passes them on with the corresponding values to the Reducer. Since the sorting is done on the composite key (primary sort on Key and secondary sort on Tag), tuples from one table will all come before the other table.

Reduce Side Join





Repartition Join [8]

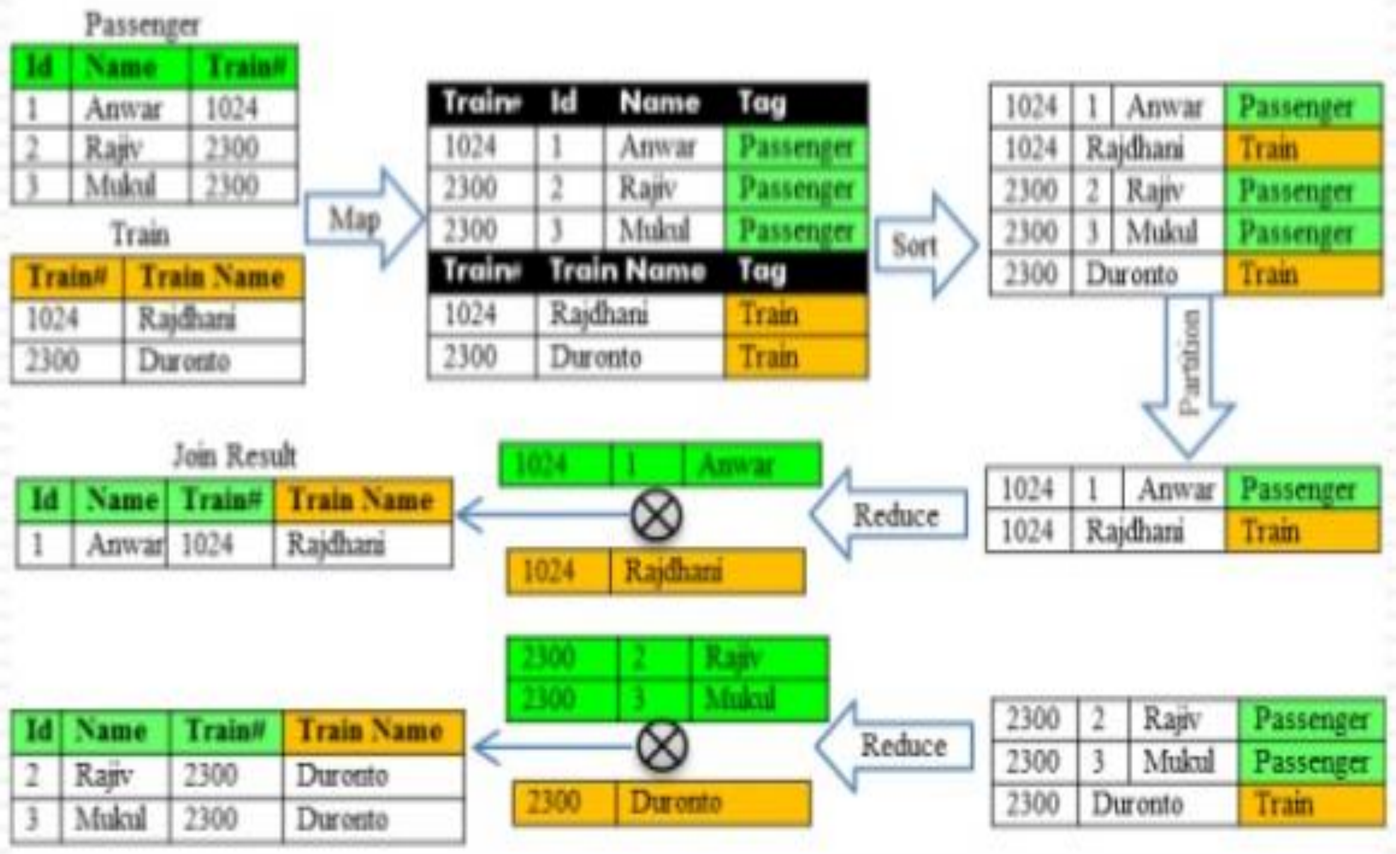
Map Phase :

- Each map task works on a split of either R or L.
- Each map task tags the record with its originating table.
- Outputs the extracted join key and the tagged record as a (key, value) pair.
- The outputs are then partitioned, sorted and merged by the framework.

Reducer Phase :

- All the records for each join key are grouped together and eventually fed to a reducer.
- For each join key, the reduce function first separates and buffers the input records into two sets according to the table tag.
- Performs a cross-product between records in the above sets.

Repartition Join [8]



Passenger

Id	Name	Train#
1	Anwar	1024
2	Rajiv	2300
3	Mukul	2300

Train

Train#	Train Name
1024	Rajdhani
2300	Duronto

Train#	Id	Name	Tag
1024	1	Anwar	Passenger
2300	2	Rajiv	Passenger
2300	3	Mukul	Passenger

Train#	Train Name	Tag
1024	Rajdhani	Train
2300	Duronto	Train

1024	1	Anwar	Passenger
1024	Rajdhani		Train
2300	2	Rajiv	Passenger
2300	3	Mukul	Passenger
2300	Duronto		Train

1024	1	Anwar	Passenger
1024	Rajdhani		Train

1024	1	Anwar

$\otimes$

1024	Rajdhani

Join Result

Id	Name	Train#	Train Name
1	Anwar	1024	Rajdhani

2300	2	Rajiv	Passenger
2300	3	Mukul	Passenger
2300	Duronto		Train

2300	2	Rajiv
2300	3	Mukul

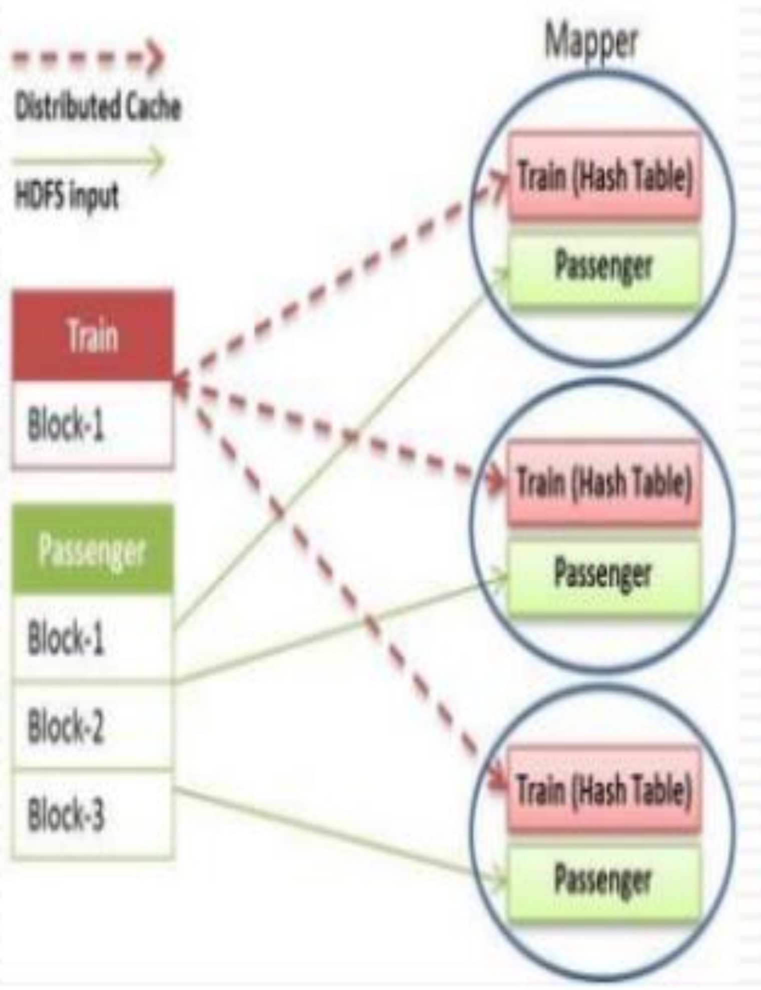
$\otimes$

2300	Duronto

Id	Name	Train#	Train Name
2	Rajiv	2300	Duronto
3	Mukul	2300	Duronto

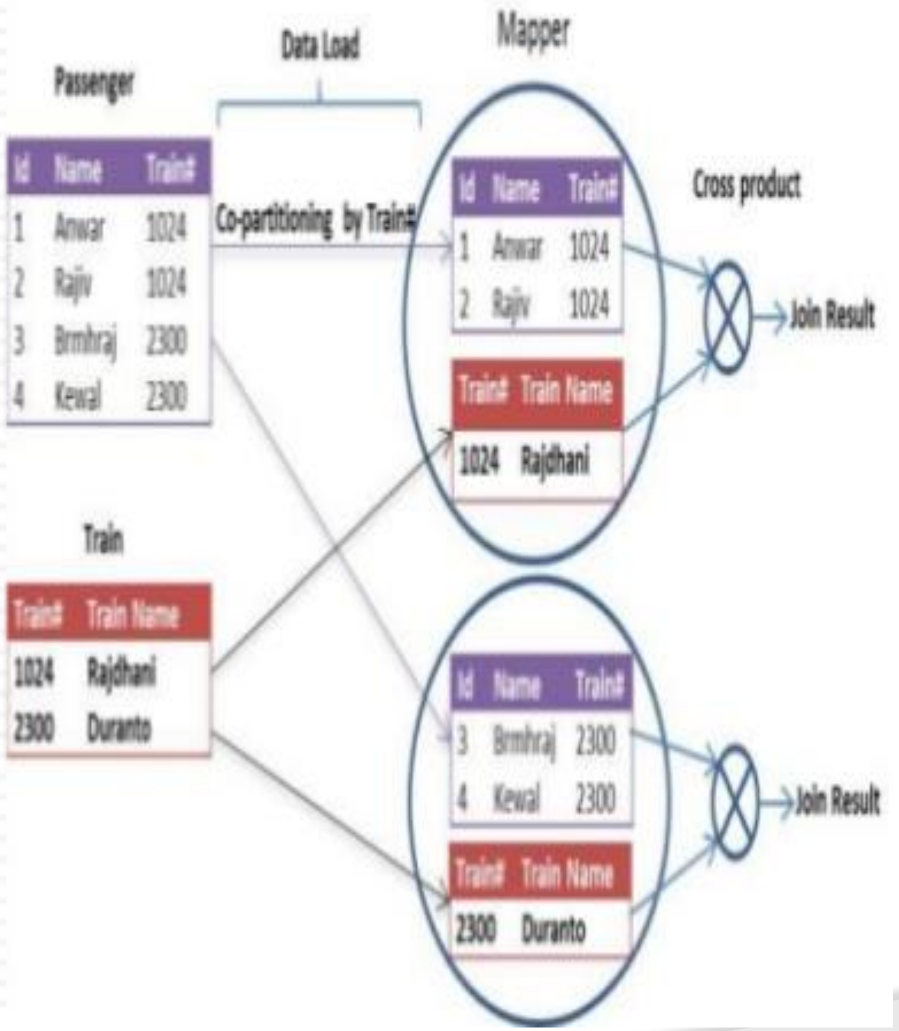
Broadcast Join [8]

- Broadcast join is used there need to join the smaller data set with large data set.
- With the assumption that smaller data set will fit into memory easily.
- The smaller data set is pushed into distributed cache and replicated among the cluster nodes.
- In init method of mapper a hash map(or other associative array) is built from smaller data set.
- And join operation is performed in map method and output is emitted.



Trojan Join [8]

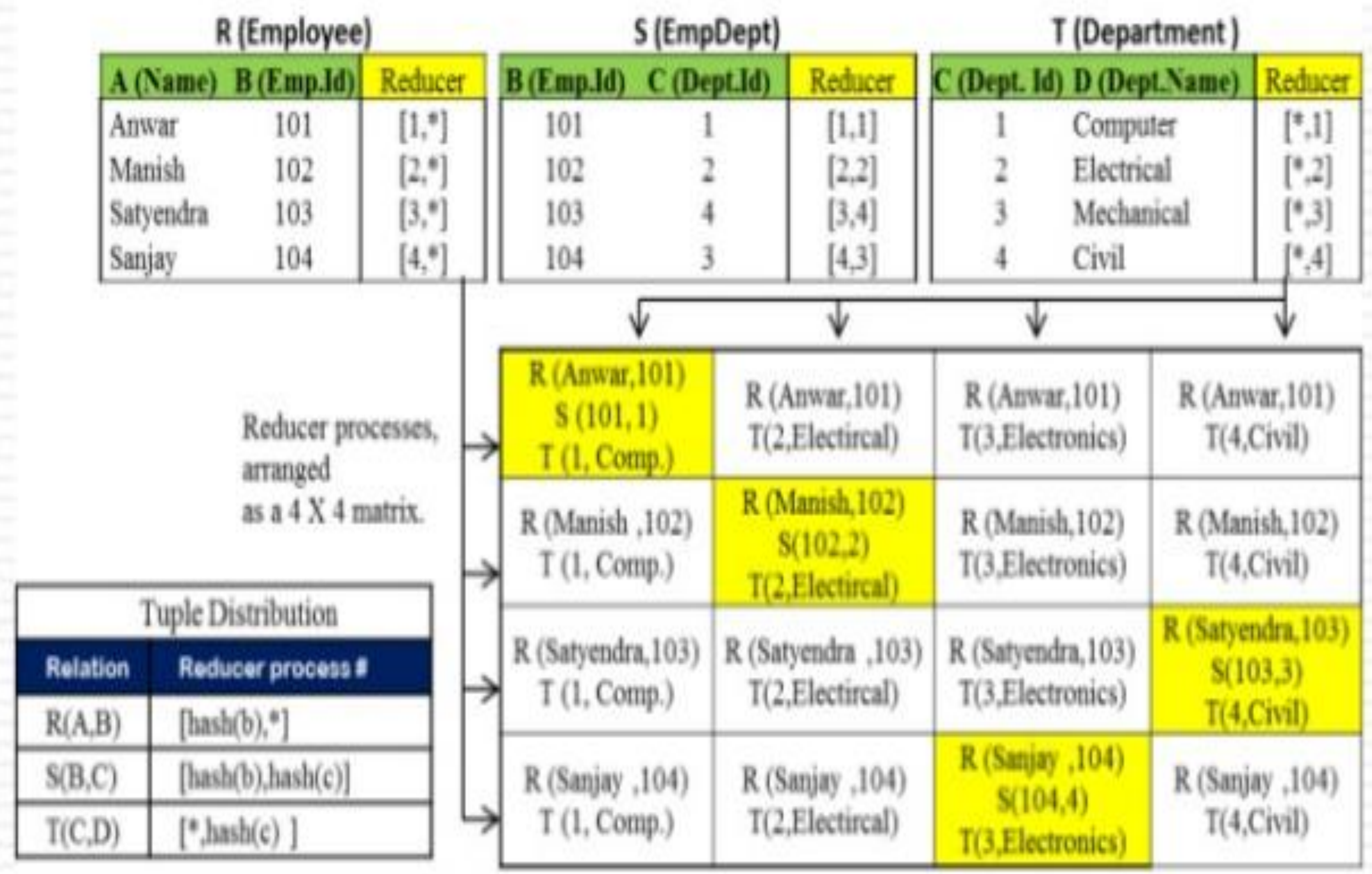
- Trojan Join supports more effective join by assuming we know the schema and workload
- Idea is to co-partition the data at load time.
- Joins are locally processed within each node at query time, but we are free to group the data on any attribute other than the join attribute in same mapreduce job



Id	Name	Train#
1	Anwar	1024
2	Rajiv	1024
3	Brnhraj	2300
4	Kewal	2300

Train#	Train Name
1024	Rajdhani
2300	Duranto

Replicated Join [8]



R (Employee)

A (Name)	B (Emp.Id)	Reducer
Anwar	101	[1,*]
Manish	102	[2,*]
Satyendra	103	[3,*]
Sanjay	104	[4,*]

S (EmpDept)

B (Emp.Id)	C (Dept.Id)	Reducer
101	1	[1,1]
102	2	[2,2]
103	4	[3,4]
104	3	[4,3]

T (Department)

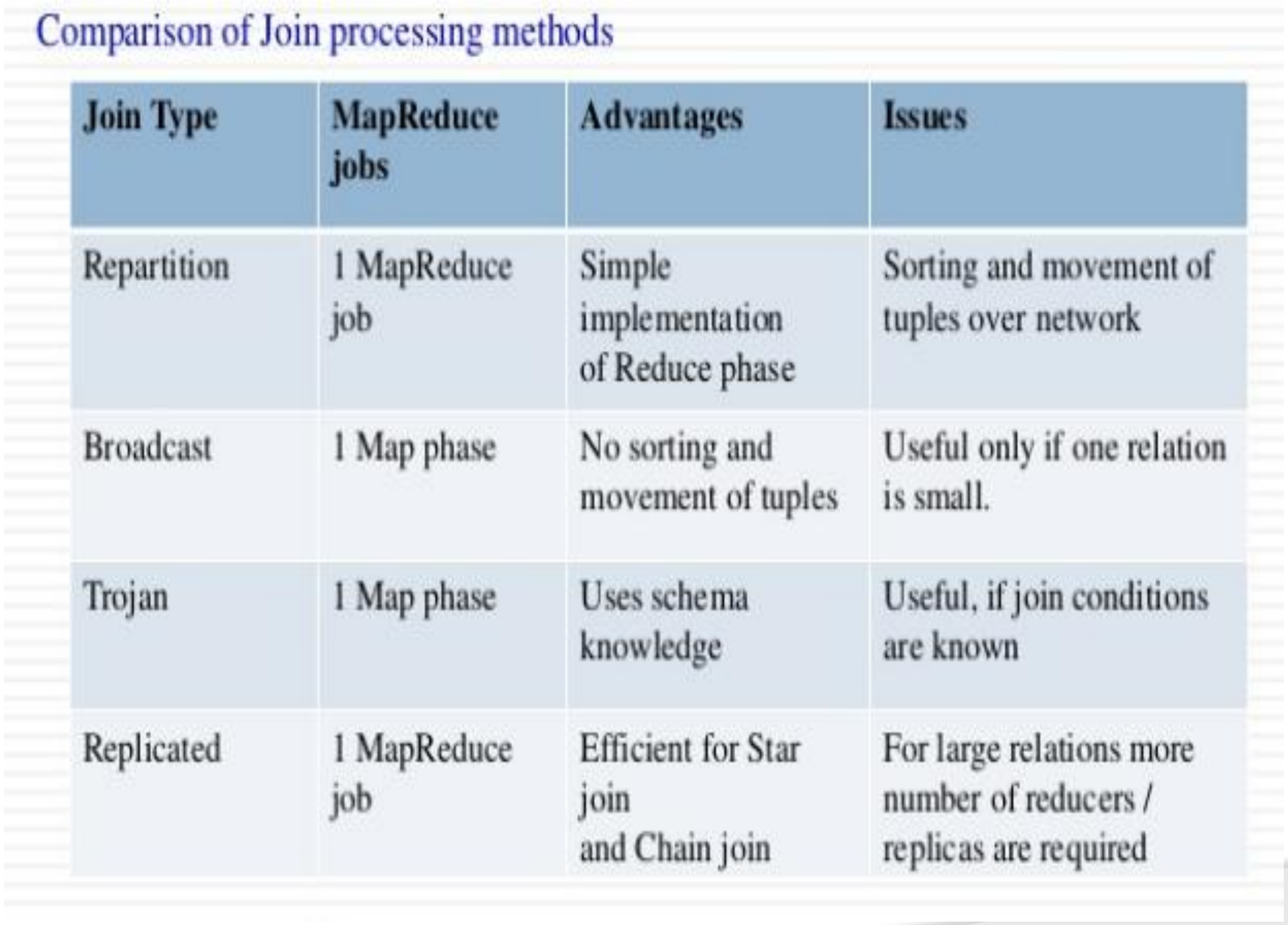
C (Dept. Id)	D (Dept.Name)	Reducer
1	Computer	[*,1]
2	Electrical	[*,2]
3	Mechanical	[*,3]
4	Civil	[*,4]

Reducer processes, arranged as a 4 X 4 matrix.

R (Anwar,101) S (101, 1) T (1, Comp.)	R (Anwar,101) T(2,Electrical)	R (Anwar,101) T(3,Electronics)	R (Anwar,101) T(4,Civil)
R (Manish ,102) T (1, Comp.)	R (Manish,102) S(102,2) T(2,Electrical)	R (Manish,102) T(3,Electronics)	R (Manish,102) T(4,Civil)
R (Satyendra,103) T (1, Comp.)	R (Satyendra ,103) T(2,Electrical)	R (Satyendra,103) T(3,Electronics)	R (Satyendra,103) S(103,3) T(4,Civil)
R (Sanjay ,104) T (1, Comp.)	R (Sanjay ,104) T(2,Electrical)	R (Sanjay ,104) S(104,4) T(3,Electronics)	R (Sanjay ,104) T(4,Civil)

Tuple Distribution

Relation	Reducer process #
R(A,B)	[hash(b),*]
S(B,C)	[hash(b),hash(c)]
T(C,D)	[*,hash(c)]



Comparison of Join processing methods

Join Type	MapReduce jobs	Advantages	Issues
Repartition	1 MapReduce job	Simple implementation of Reduce phase	Sorting and movement of tuples over network
Broadcast	1 Map phase	No sorting and movement of tuples	Useful only if one relation is small.
Trojan	1 Map phase	Uses schema knowledge	Useful, if join conditions are known
Replicated	1 MapReduce job	Efficient for Star join and Chain join	For large relations more number of reducers / replicas are required

UNIT-5

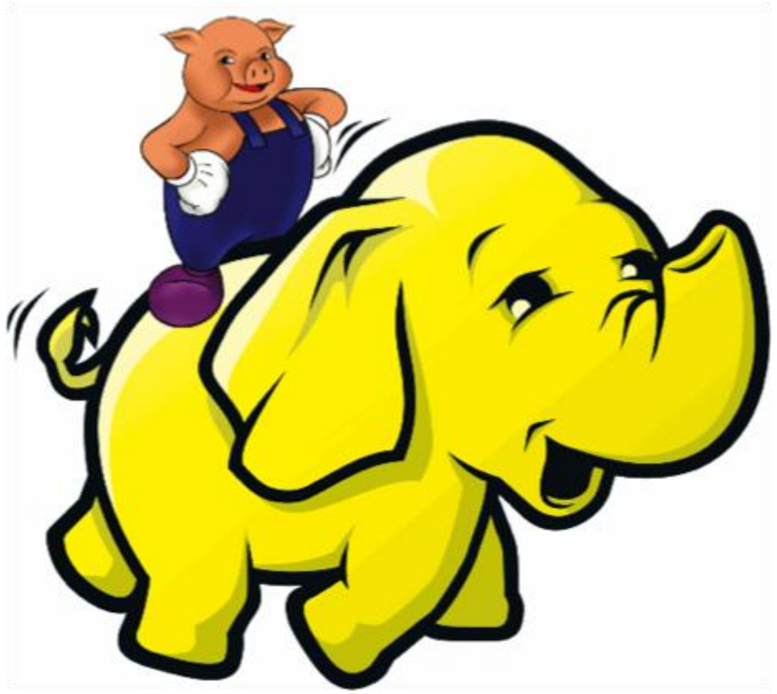
INTRODUCTION TO PIG and HIVE

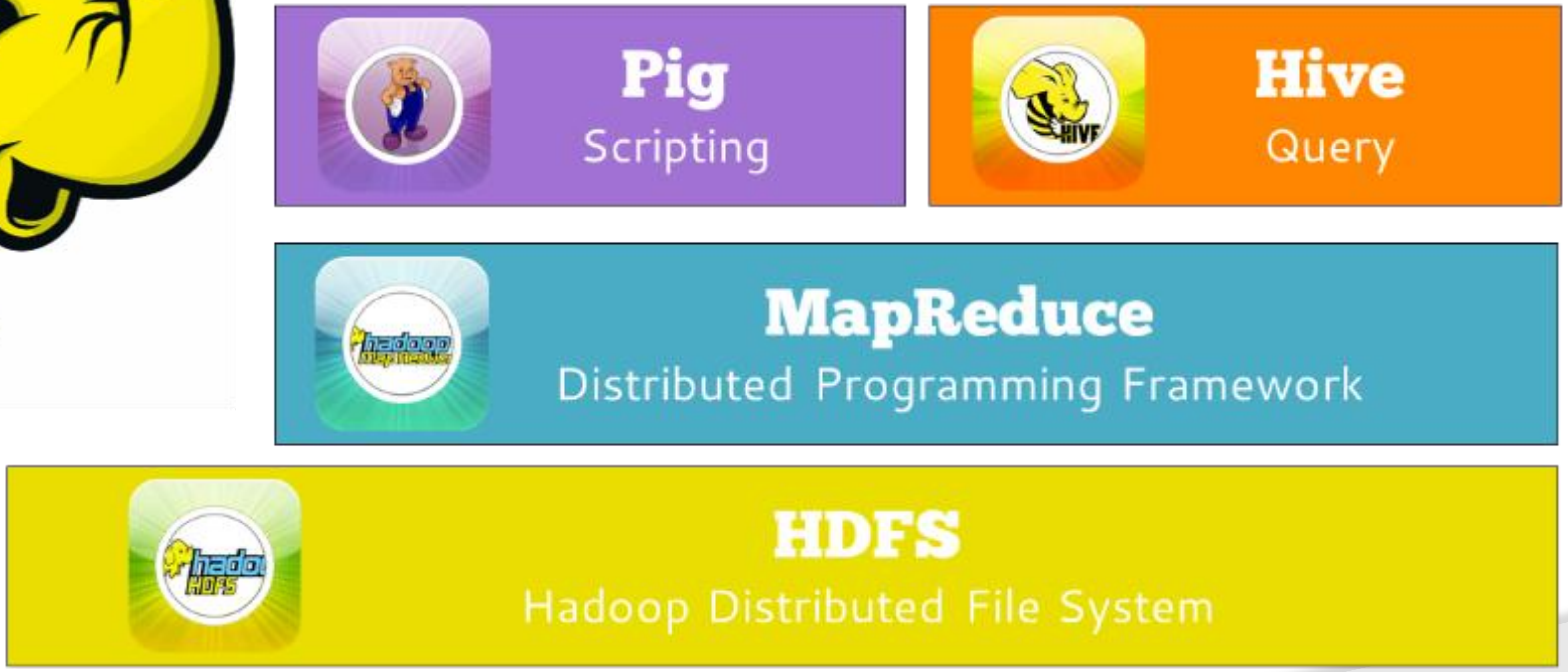
PIG

✓PIG’s are omnivores animals which means they can consume both plants and animals.

✓The PIG consumes any type of data whether Structured or unStructured or any other machine data & helps processing the same.

PIG is on the top of hadoop.



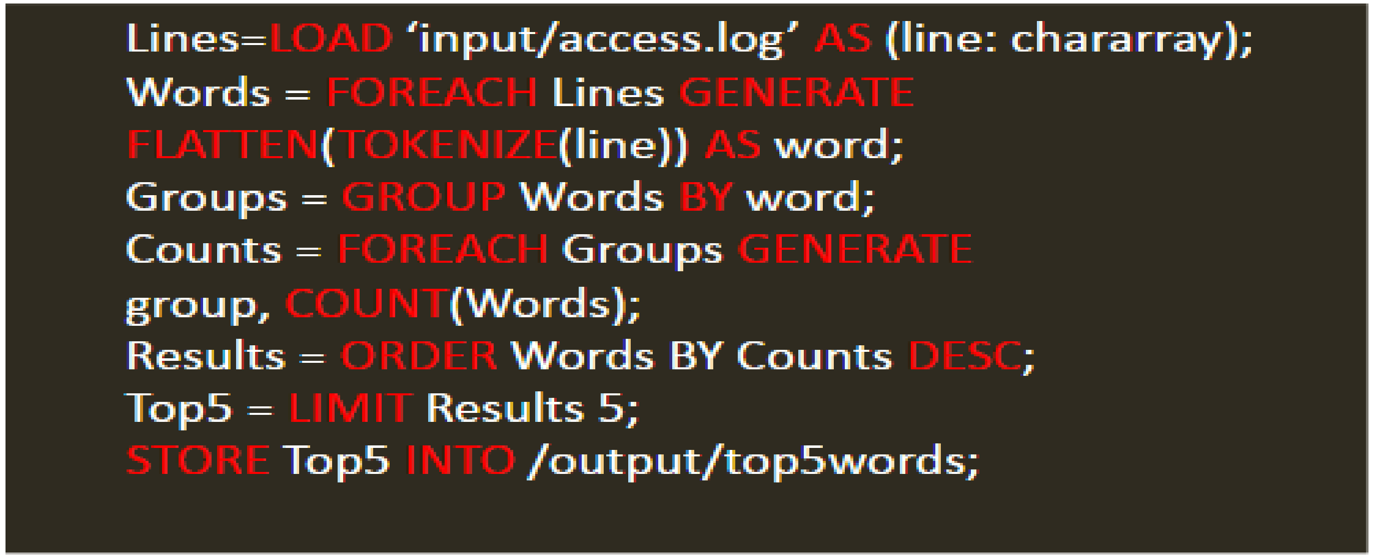

Pig
Scripting
Hive
Query
MapReduce
Distributed Programming Framework
HDFS
Hadoop Distributed File System

- Map Reduce is very powerful, but:

– It requires a Java programmer.

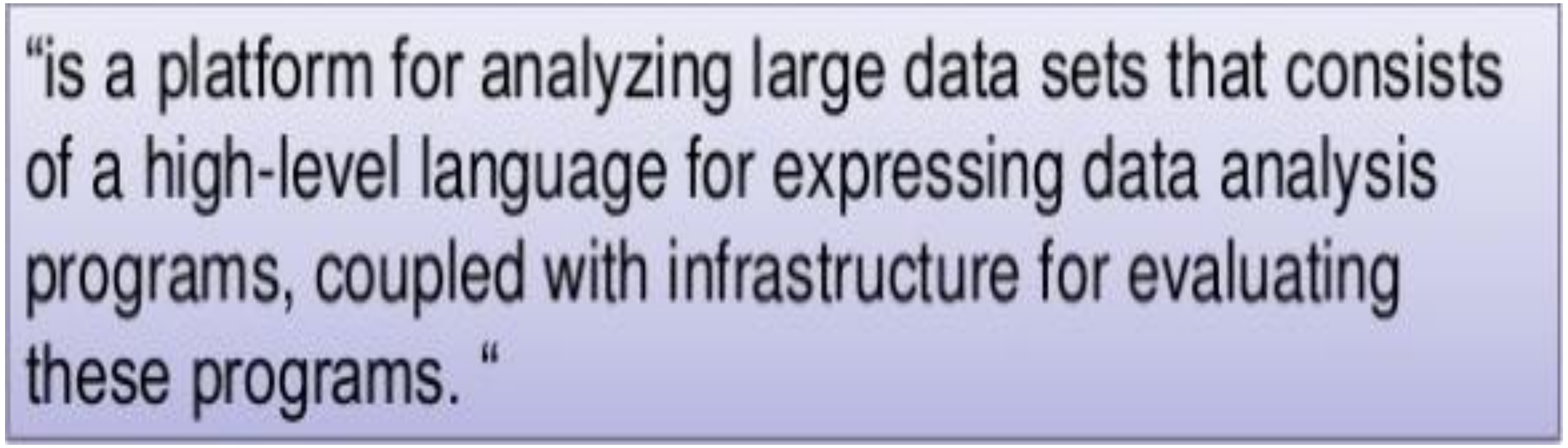
– User has to re-invent common functionality (join, filter, etc.).

Word Count using Pig



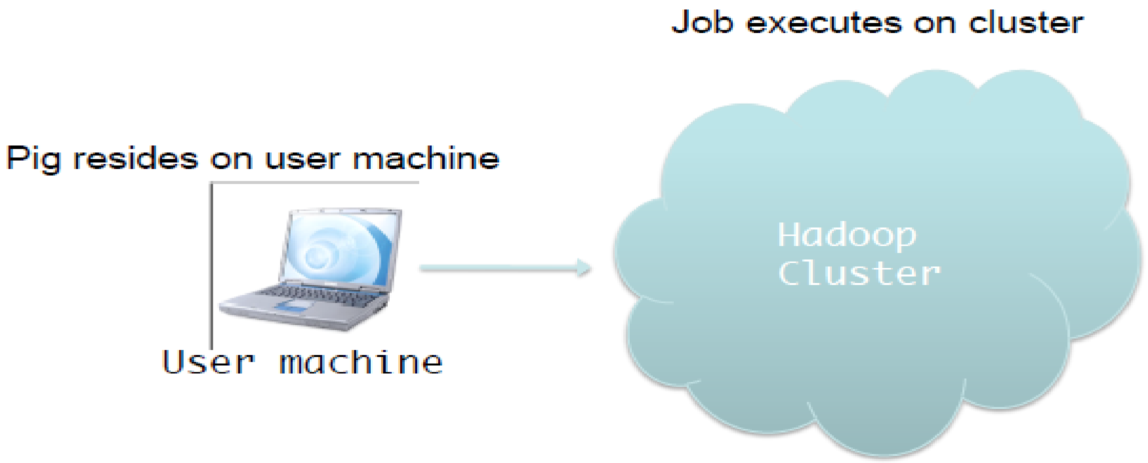
```
Lines=LOAD ‘input/access.log’ AS (line: chararray);
Words = FOREACH Lines GENERATE
FLATTEN(TOKENIZE(line)) AS word;
Groups = GROUP Words BY word;
Counts = FOREACH Groups GENERATE
group, COUNT(Words);
Results = ORDER Words BY Counts DESC;
Top5 = LIMIT Results 5;
STORE Top5 INTO /output/top5words;
```

What is PIG?



“is a platform for analyzing large data sets that consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs. “

Components



No need to install anything extra on your Hadoop cluster.

- Sub-project of Apache Hadoop
- Platform for analyzing large data sets
- Includes a data-flow language Pig Latin
- Built for Hadoop
  - Translates script to MapReduce program under the hood
- Originally developed at Yahoo!
  - Huge contributions from Hortonworks, Twitter

- Framework for analyzing large un-structured and semi-structured data on top of Hadoop.
- Pig Engine Parses, compiles Pig Latin scripts into MapReduce jobs run on top of Hadoop.
- Pig Latin is simple but powerful data flow language similar to scripting languages.
  - SQL – like language
  - Provide common data operations (e.g. filters, joins, ordering)

How It Works

Pig Latin

```
A = LOAD 'myfile'
    AS (x, y, z);
B = FILTER A by x > 0;
C = GROUP B BY x;
D = FOREACH A GENERATE
    x, COUNT(B);
STORE D INTO 'output';
```

$\longrightarrow$

pig.jar:

- •parses
- •checks
- •optimizes
- •plans execution
- •submits jar to Hadoop
- •monitors job progress

Execution Plan
Map:
    Filter

Reduce:
    Count

$\downarrow$

Hadoop

Why Pig?

- Makes writing hadoop jobs a lot simpler
  - 5% of the code, 5% of time
  - You don't have to be a programmer to write Pig scripts
- Provides major functionality required for DW and Analytics
  - Load, Filter, Join, Group By, Order, Transform, UDFs, Store
- User can write custom UDFs (User Defined Function)

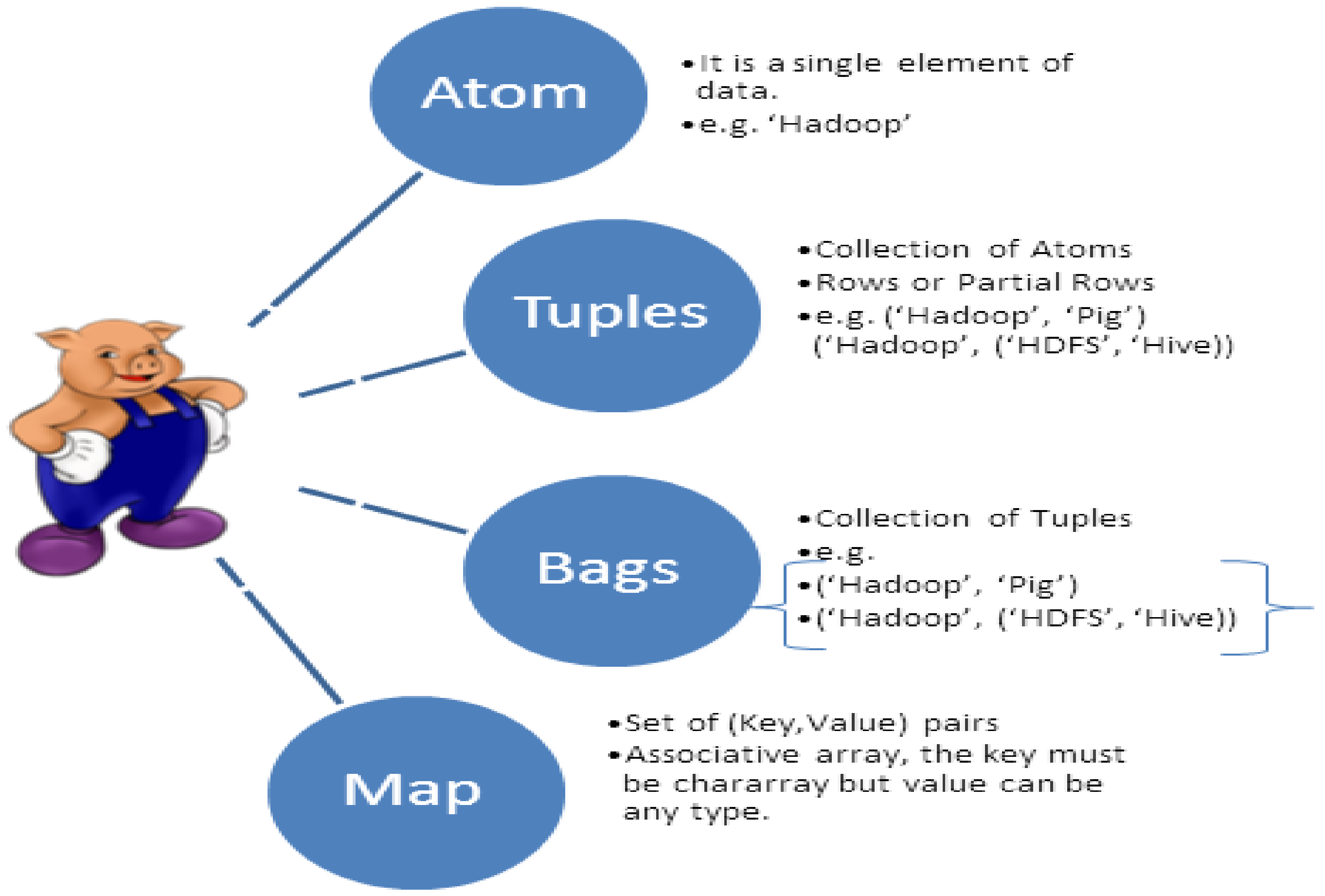
Pig Tutorial

- Basic Pig knowledge: (Word Count)
  - Pig Data Types
  - Pig Operations
  - How to run Pig Scripts

PigLatin – the dataflow language

- PigLatin statements work with relations
  - A relation (analogous to database table) is a bag
  - A bag is a collection of tuples
  - A tuple (analogous to database row) is an ordered set of fields
  - A field is a piece of data
- Example, A = LOAD ‘input.dat’;
  - Here ‘A’ is a relation
  - All records in ‘A’ (from the file ‘input.dat’) collectively form a bag
  - Each record in ‘A’ is a tuple
  - A field is a single cell in each tuple

To remember : A Pig relation is a bag of tuples



- It is a single element of data.
- e.g. ‘Hadoop’

- Collection of Atoms
- Rows or Partial Rows
- e.g. (‘Hadoop’, ‘Pig’) (‘Hadoop’, (‘HDFS’, ‘Hive))

- Collection of Tuples
- e.g.
- (‘Hadoop’, ‘Pig’)
- (‘Hadoop’, (‘HDFS’, ‘Hive))

- Set of (Key,Value) pairs
- Associative array, the key must be chararray but value can be any type.

Apache Pig Execution Modes

You can run Apache Pig in two modes, namely, **Local Mode** and **HDFS mode**.

Local Mode

In this mode, all the files are installed and run from your local host and local file system. There is no need of Hadoop or HDFS. This mode is generally used for testing purpose.

MapReduce Mode

MapReduce mode is where we load or process the data that exists in the Hadoop File System (HDFS) using Apache Pig. In this mode, whenever we execute the Pig Latin statements to process the data, a MapReduce job is invoked in the back-end to perform a particular operation on the data that exists in the HDFS.

Apache Pig Execution Mechanisms

Apache Pig scripts can be executed in three ways, namely, interactive mode, batch mode, and embedded mode.

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

Local mode

- All files are installed and run using your local host and file system
  - Does not involve a real hadoop cluster
- Great for starting off, debugging
- Specify local mode using the -x flag
  - $ pig -x local
  - $ grunt> a = load 'foo'; -- here the file 'foo' resides on local filesystem

Mapreduce mode

- Default mode
- Access to a Hadoop cluster and HDFS installation
- Point Pig to remote cluster by placing HADOOP_CONF_DIR on PIG_CLASSPATH
  - HADOOP_CONF_DIR is the directory containing your hadoop-site.xml, hdfs-site.xml, mapred-site.xml files
  - Example: $ export PIG_CLASSPATH=<path_to_hadoop_conf_dir>
  - $ pig
  - grunt> a = load 'foo'; -- here 'foo' refers to a file on HDFS

Data types

- int, long
- float, double
- chararray - Java String
- bytearray
  - default type of all fields if schema not specified
- Complex data types
  - tuple, eg (abc,def)
  - bag, eg {(19,2), (18,1)}
  - map, eg [sfdc#logs]

Pig Operations

- Loading data
  - LOAD loads input data
  - Lines=LOAD ‘input/access.log’ AS (line: chararray);
- Projection
  - FOREACH … GENERTE (similar to SELECT)
  - takes a set of expressions and applies them to every record.
- De-duplication
  - DISTINCT removes duplicate records
- Grouping
  - GROUPS collects together records with the same key
- Aggregation
  - AVG, COUNT, COUNT_STAR, MAX, MIN, SUM

Pig Commands

Pig Command	What it does
load	Read data from file system.
store	Write data to file system.
foreach	Apply expression to each record and output one or more records.
filter	Apply predicate and remove records that do not return true.
group/cogroup	Collect records with the same key from one or more inputs.
join	Join two or more inputs based on a key.
order	Sort records based on a key.
distinct	Remove duplicate records.
union	Merge two data sets.
split	Split data into 2 or more sets, based on filter conditions.
stream	Send all records through a user provided binary.
dump	Write output to stdout.
limit	Limit the number of records.

Loading data

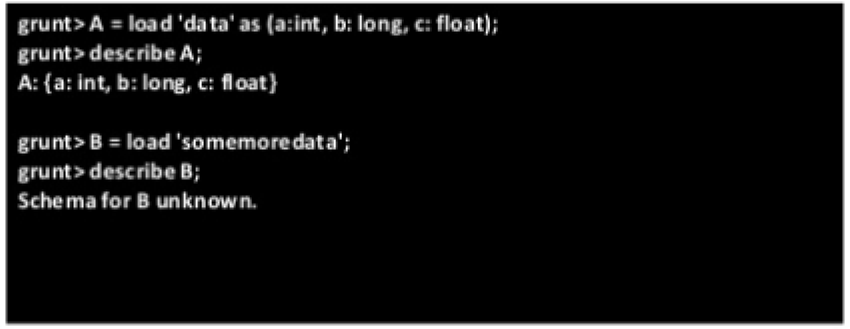
- LOAD
  - Reads data from the file system
- Syntax
  - LOAD ‘input’ [USING function] [AS schema];
  - **Eg,** A = LOAD ‘input’ USING PigStorage(‘\t’) AS (name:chararray, age:int, gpa:float);

Schema

- Use schemas to assign types to fields
- A = LOAD 'data' AS (name, age, gpa);
  - name, age, gpa default to bytearrays
- A = LOAD 'data' AS (name:chararray, age:int, gpa:float);
  - name is now a String (chararray), age is integer and gpa is float

Describing Schema

- Describe
  - Provides the schema of a relation
- Syntax
  - DESCRIBE [alias];
  - If schema is not provided, describe will say "Schema for alias unknown"



```
grunt> A = load 'data' as (a:int, b: long, c: float);
grunt> describe A;
A: {a: int, b: long, c: float}

grunt> B = load 'somemoredata';
grunt> describe B;
Schema for B unknown.
```

Dump and Store

- Dump writes the output to console
  - grunt> A = load 'data';
  - grunt> DUMP A; //This will print contents of A on Console
- Store writes output to a HDFS location
  - grunt> A = load 'data';
  - grunt> STORE A INTO '/user/username/output'; //This will write contents of A to HDFS
- Pig starts a job only when a DUMP or STORE is encountered

Referencing Fields

- Fields are referred to by positional notation OR by name (alias)
  - Positional notation is generated by the system
  - Starts with $0
  - Names are assigned by you using schemas. Eg, A = load 'data' as (*name*:chararray, *age*:int);
- With positional notation, fields can be accessed as
  - A = load 'data';
  - B = foreach A generate $0, $1; //1st& 2nd column

Limit

- Limits the number of output tuples
- Syntax
  - alias = LIMIT alias n;

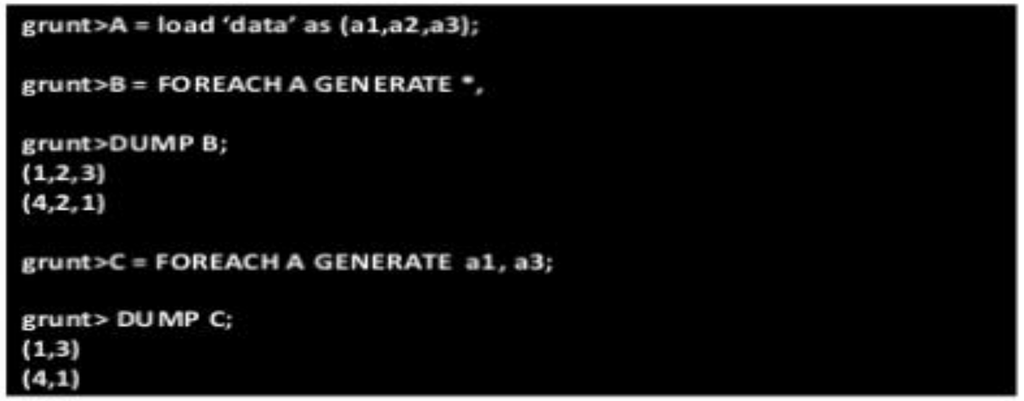
```
grunt> A = load 'data';

grunt> B = LIMIT A 10;

grunt> DUMP B; --Prints only 10 rows
```

Foreach.. Generate

- Used for data transformations and projections
- Syntax
  - alias = FOREACH { block | nested_block };
  - *nested_block usage later in the deck*



```
grunt>A = load 'data' as (a1,a2,a3);

grunt>B = FOREACH A GENERATE *,

grunt>DUMP B;
(1,2,3)
(4,2,1)

grunt>C = FOREACH A GENERATE a1, a3;

grunt> DUMP C;
(1,3)
(4,1)
```

Filter

- Selects tuples from a relation based on some condition
- Syntax
  - alias = FILTER alias BY expression;
  - Example, to filter for 'marcbenioff'
    - A = LOAD 'sfdcemployees' USING PigStorage(',') as (name:chararray,employeesince:int,age:int);
    - B = FILTER A BY name == 'marcbenioff';
  - You can use boolean operators (AND, OR, NOT)
    - B = FILTER A BY (employeesince< 2005) AND

(NOT(name == 'marcbenioff'));

Group By

- Groups data in one or more relations (similar to SQL GROUP BY)
- Syntax:
  - alias = GROUP alias { ALL | BY expression} [, alias ALL | BY expression …] [PARALLEL n];
  - Eg, to group by (employee start year at Salesforce)
    - A = LOAD 'sfdcemployees' USING PigStorage(',') as (name:chararray, employeesince:int, age:int);
    - B = GROUP A BY (employeesince);
  - You can also group by all fields together
    - B = GROUP B BY ALL;
  - Or Group by multiple fields
    - B = GROUP A BY (age, employeesince);

Using Grouped Results

- FOREACH works for grouped data
- Let's see an example to count the number of rows grouped by employee start year

```
grunt> A = load 'data' as (name, employeesince, age);
grunt> B = GROUP A by employeesince;
grunt> C = FOREACH B GENERATE group, COUNT(A);
```

- 'group' is an implicit field name given to group key
- Use the alias grouped, within an aggregation function - COUNT(A)

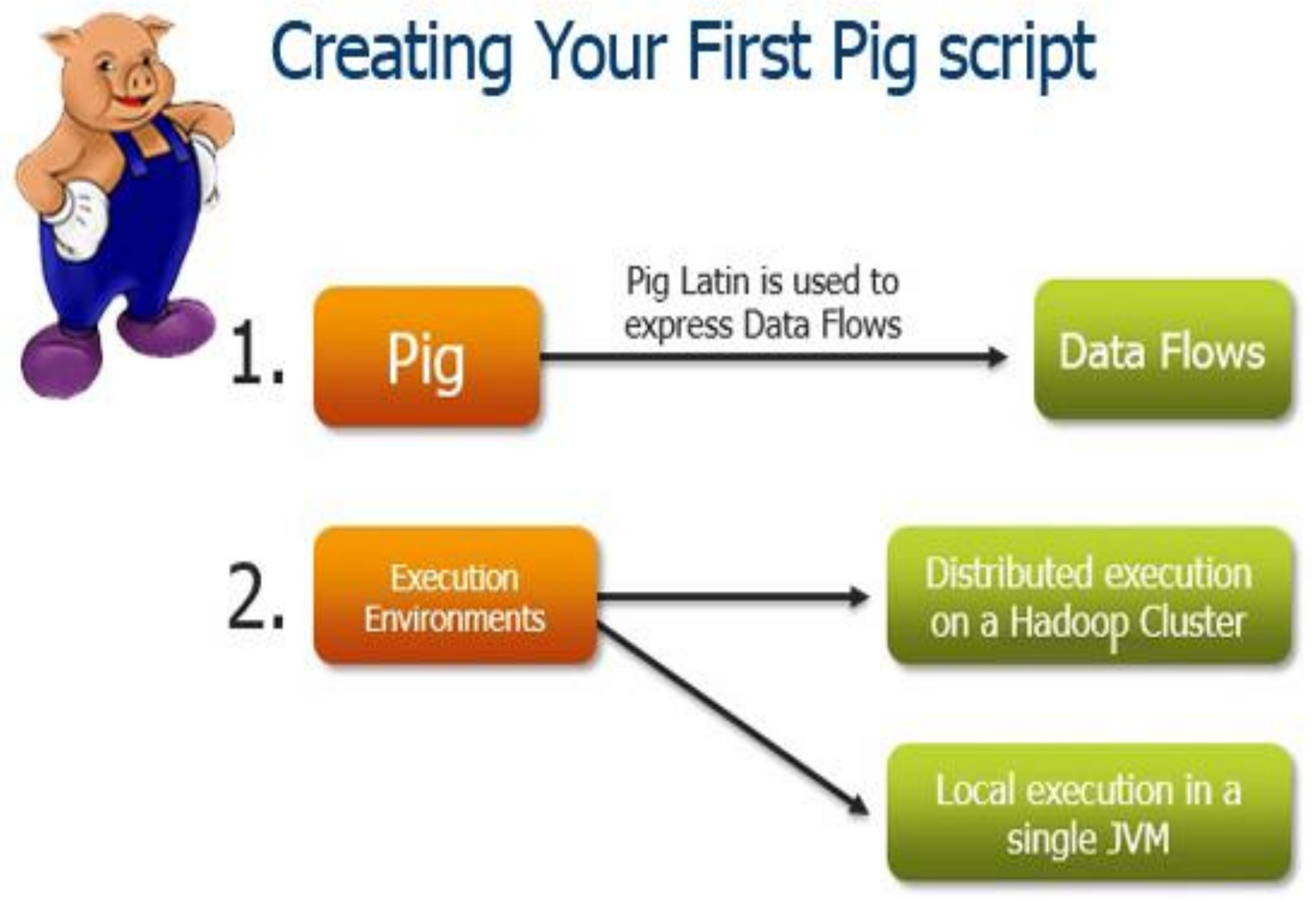
Aggregation

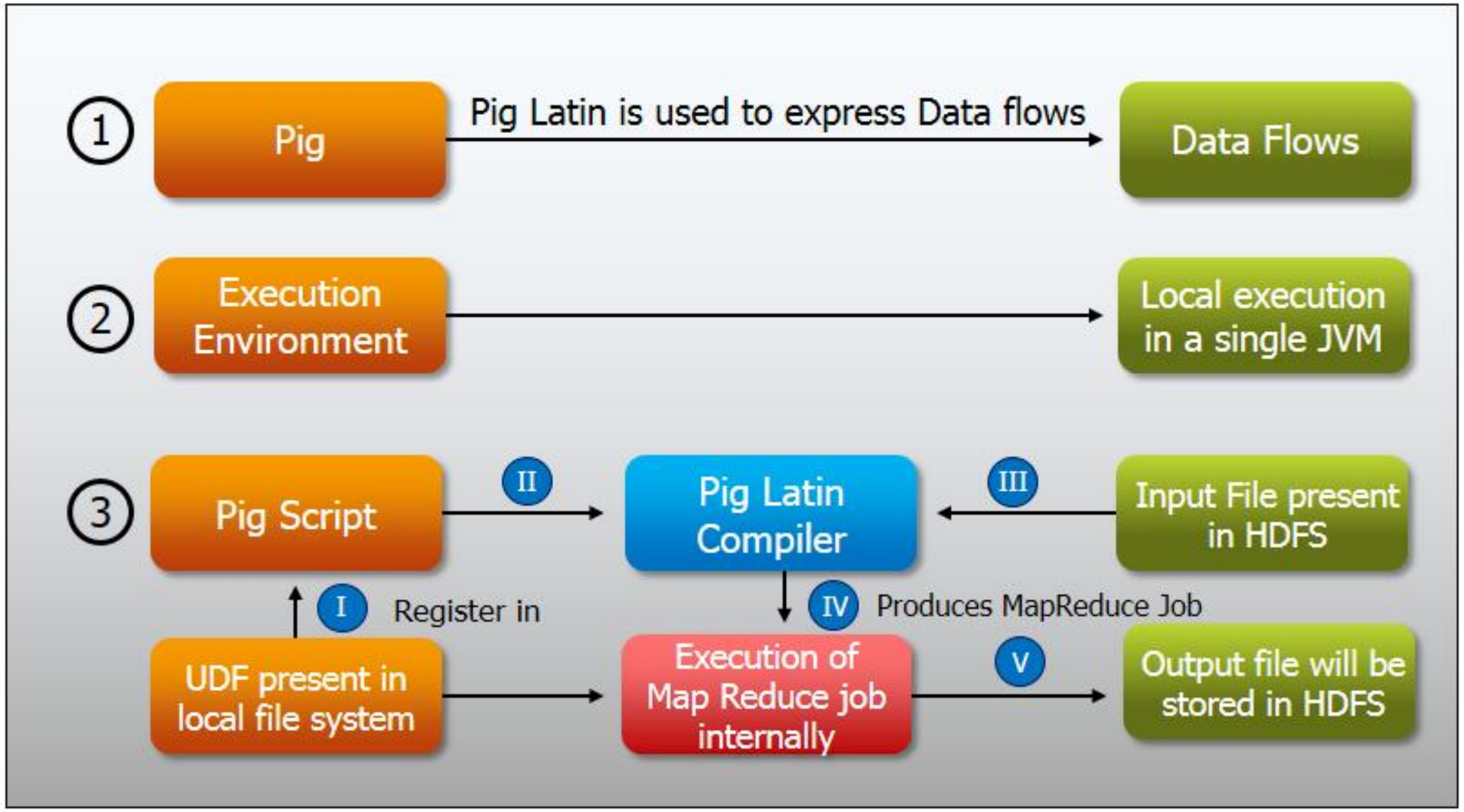
- Pig provides a bunch of aggregation functions
  - AVG
  - COUNT
  - COUNT_STAR
  - SUM
  - MAX
  - MIN

Case Sensitivity

- names (aliases) of relations and fields are case sensitive
  - A = load ‘input’; B = foreacha generate $0; *--Won’t work*
- UDF names are case sensitive
  - ‘LENGTH’ is not the same as ‘length’
- PigLatin keywords are case insensitive
  - Load, dump, Group by, foreach..generate, join

Creating Your First Pig script


1.
Pig
Pig Latin is used to express Data Flows
Data Flows
2.
Execution Environments
Distributed execution on a Hadoop Cluster
Local execution in a single JVM


1
Pig
Pig Latin is used to express Data flows
Data Flows
2
Execution Environment
Local execution in a single JVM
3
Pig Script
II
Pig Latin Compiler
III
Input File present in HDFS
I
Register in
IV
Produces MapReduce Job
UDF present in local file system
Execution of Map Reduce job internally
V
Output file will be stored in HDFS