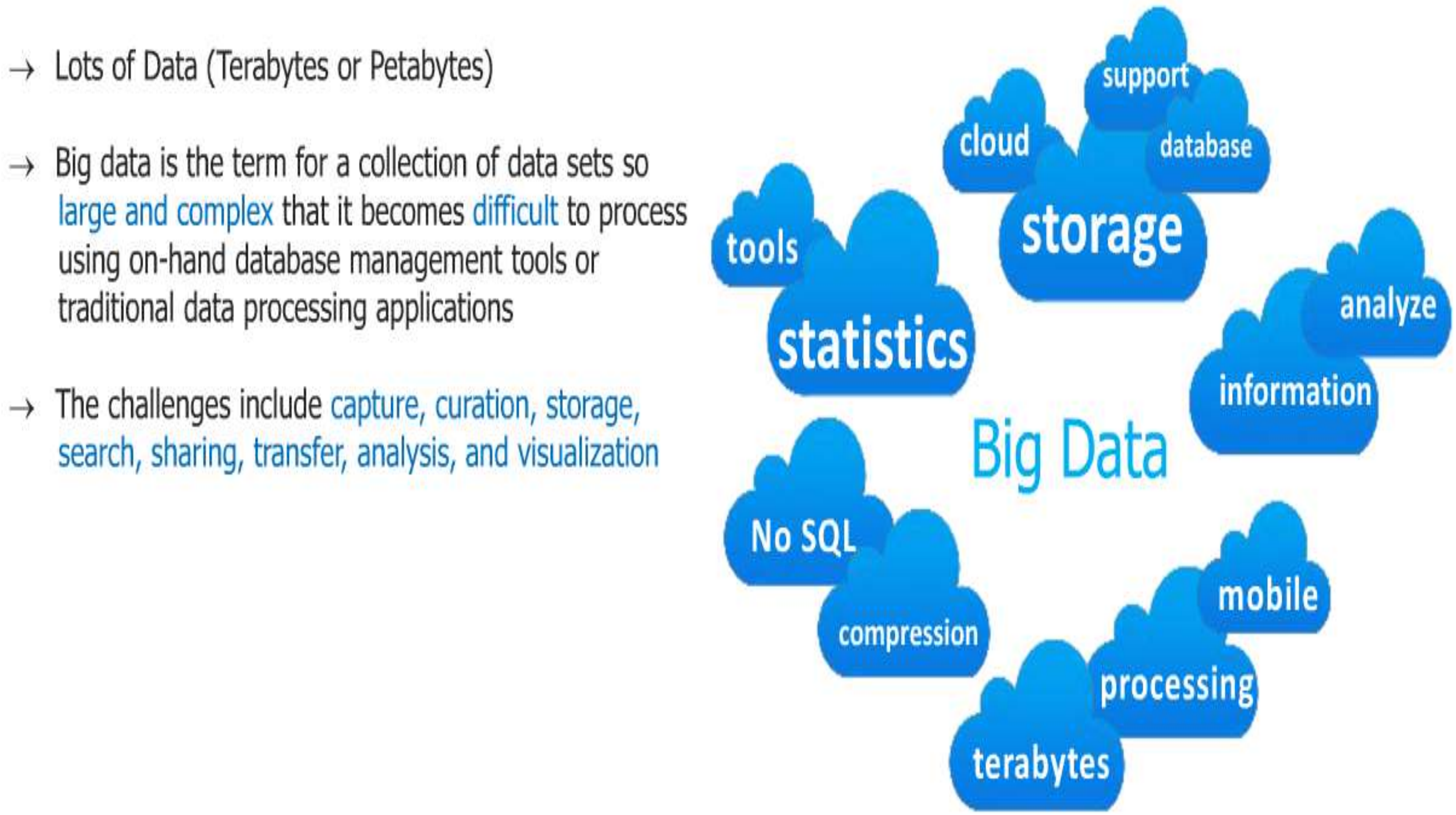
CADT IDRI

Big Data Analytics

Expand the capacity to implement data science research, data science training, and industry collaborations using cutting-edge computational tools.

Prepared by: TENG CHANTO

1. What is big data?



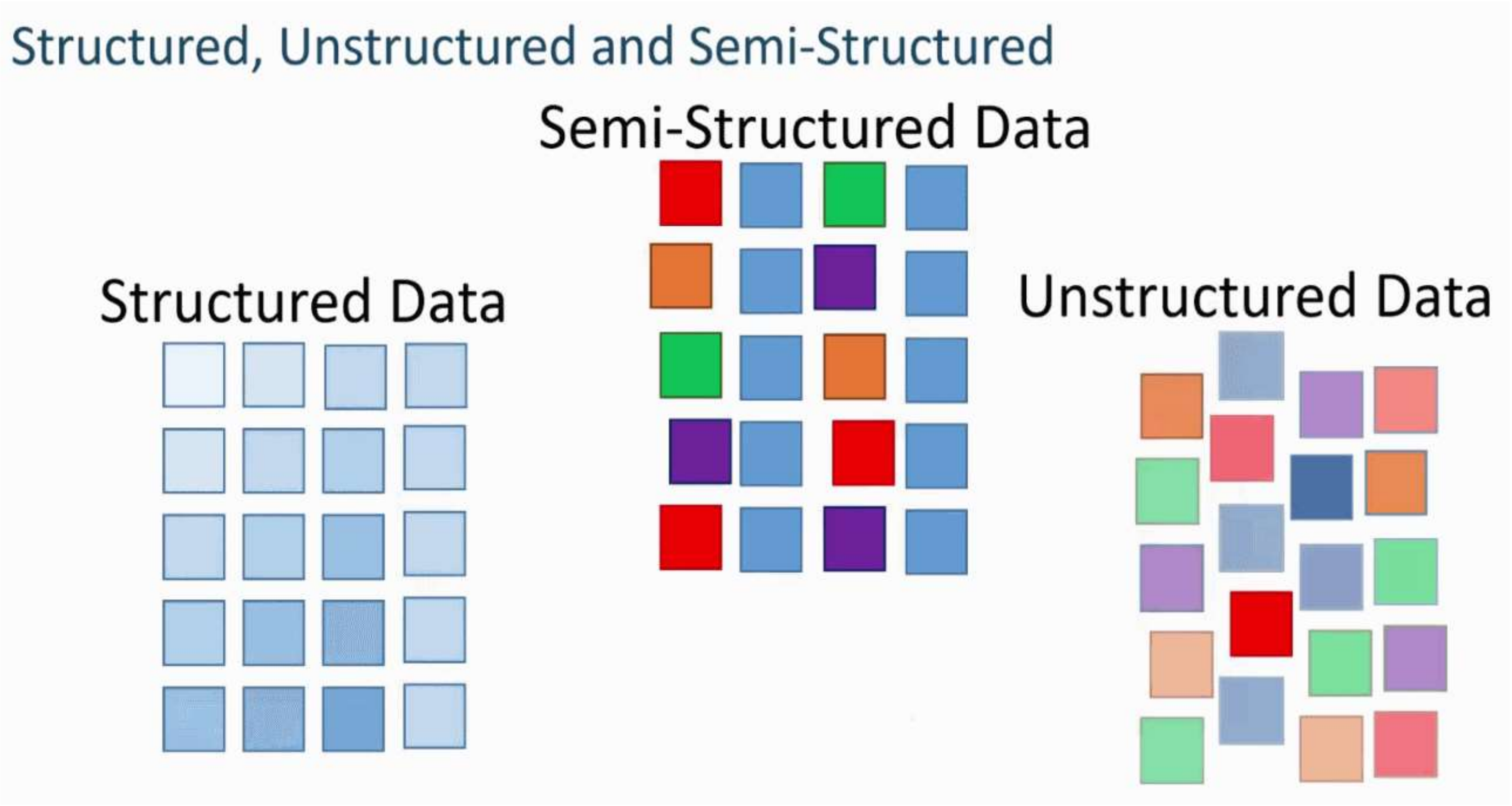
→ Lots of Data (Terabytes or Petabytes)

→ 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

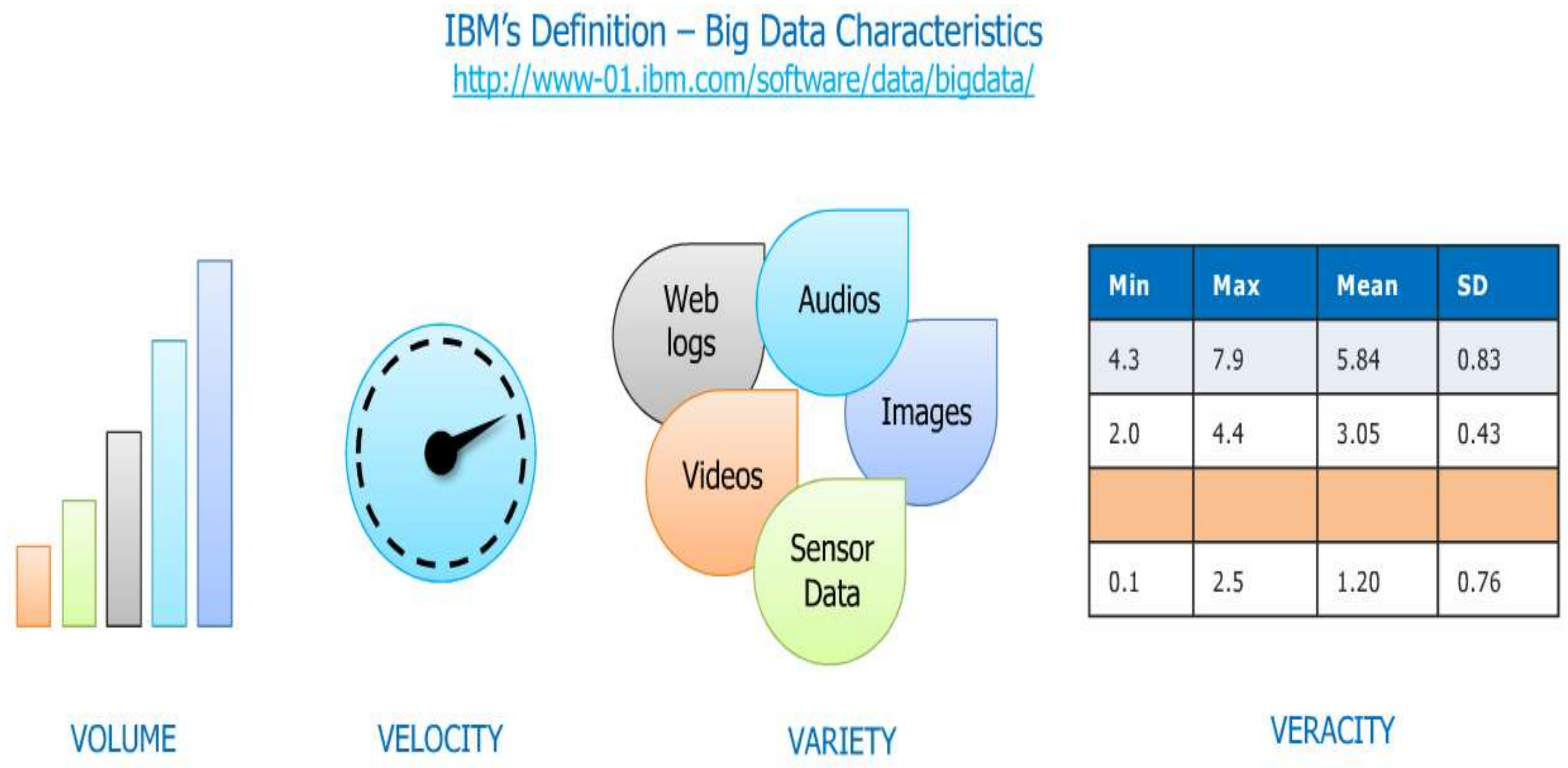
→ The challenges include capture, curation, storage, search, sharing, transfer, analysis, and visualization

support
cloud
database
storage
tools
statistics
analyze
information
Big Data
No SQL
compression
mobile
processing
terabytes

Types of Big Data



IBM Definition of big data



IBM's Definition – Big Data Characteristics

http://www-01.ibm.com/software/data/bigdata/

Min	Max	Mean	SD
4.3	7.9	5.84	0.83
2.0	4.4	3.05	0.43
0.1	2.5	1.20	0.76

VOLUME

VELOCITY

VARIETY

VERACITY

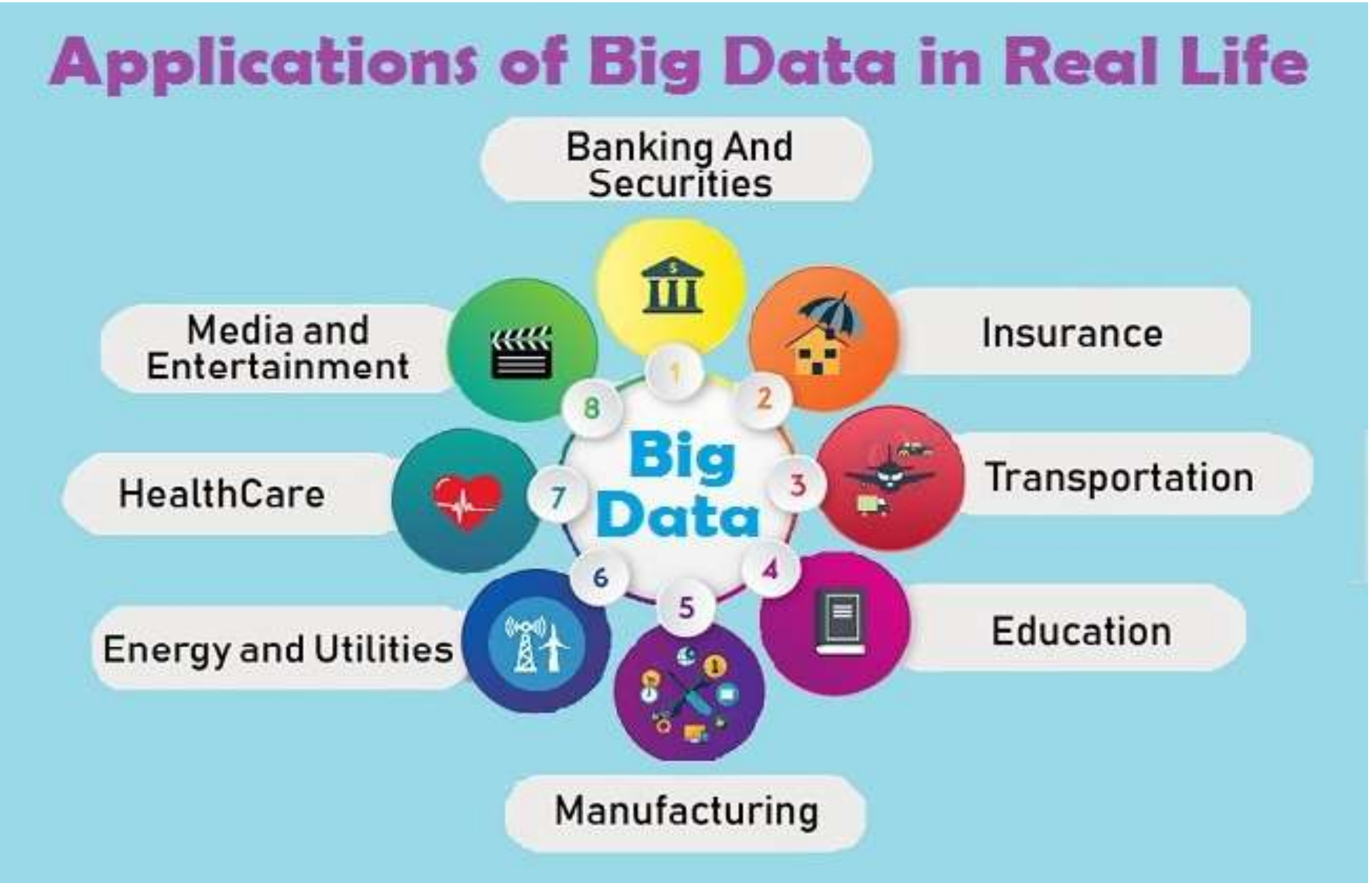
Examples of Big Data

- Walmart handles more than **1 million** customer transactions every hour.
- Facebook stores, accesses, and analyzes **30+ Petabytes** of user generated data.
- **230+ millions** of tweets are created every day.
- More than **5 billion** people are calling, texting, tweeting and browsing on mobile phones worldwide.
- YouTube users upload **48 hours** of new video every minute of the day.
- Amazon handles **15 million** customer click stream user data per day to recommend products.
- **294 billion** emails are sent every day. Services analyses this data to find the spams.
- Modern cars have close to **100 sensors** which monitors fuel level, tire pressure etc. , each vehicle generates a lot of sensor data.

Challenges with Big Data

- Data Quality
- Discovery
- Storage
- Analytics
- Security
- Lack of Talent

Big Data Applications



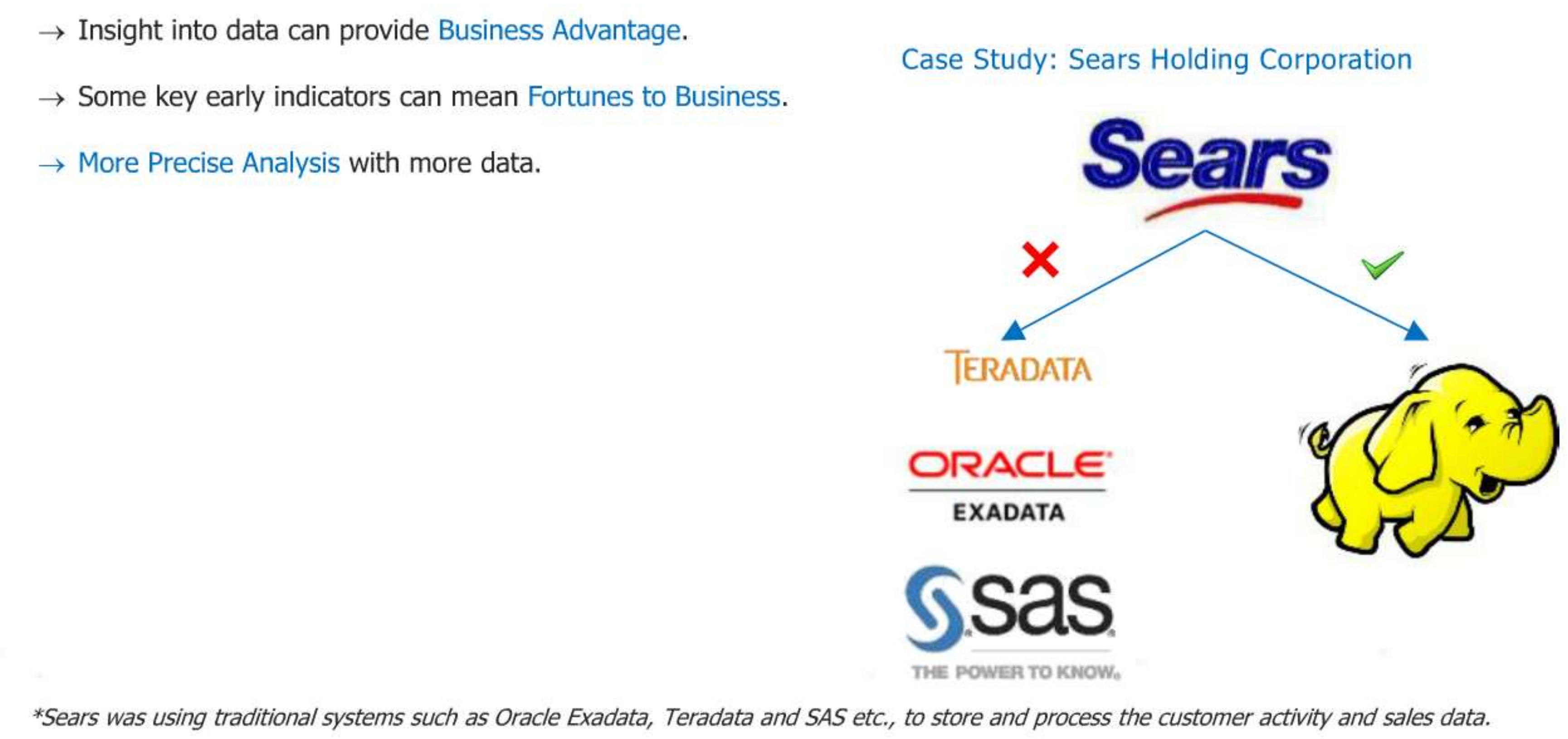
Big Data in Government

- Cyber security & Intelligence
- Crime Prediction and Prevention
- Pharmaceutical Drug Evaluation
- Scientific Research
- Weather Forecasting
- Tax Compliance
- Traffic Optimization

Real time scenarios

- Facebook: Ad targeting
- Air traffic
- Electricity grid
- Web &e tailing:recommmendations
- Twitter
- Marketing engine

Sears use Hadoop for analysis



→ Insight into data can provide Business Advantage.

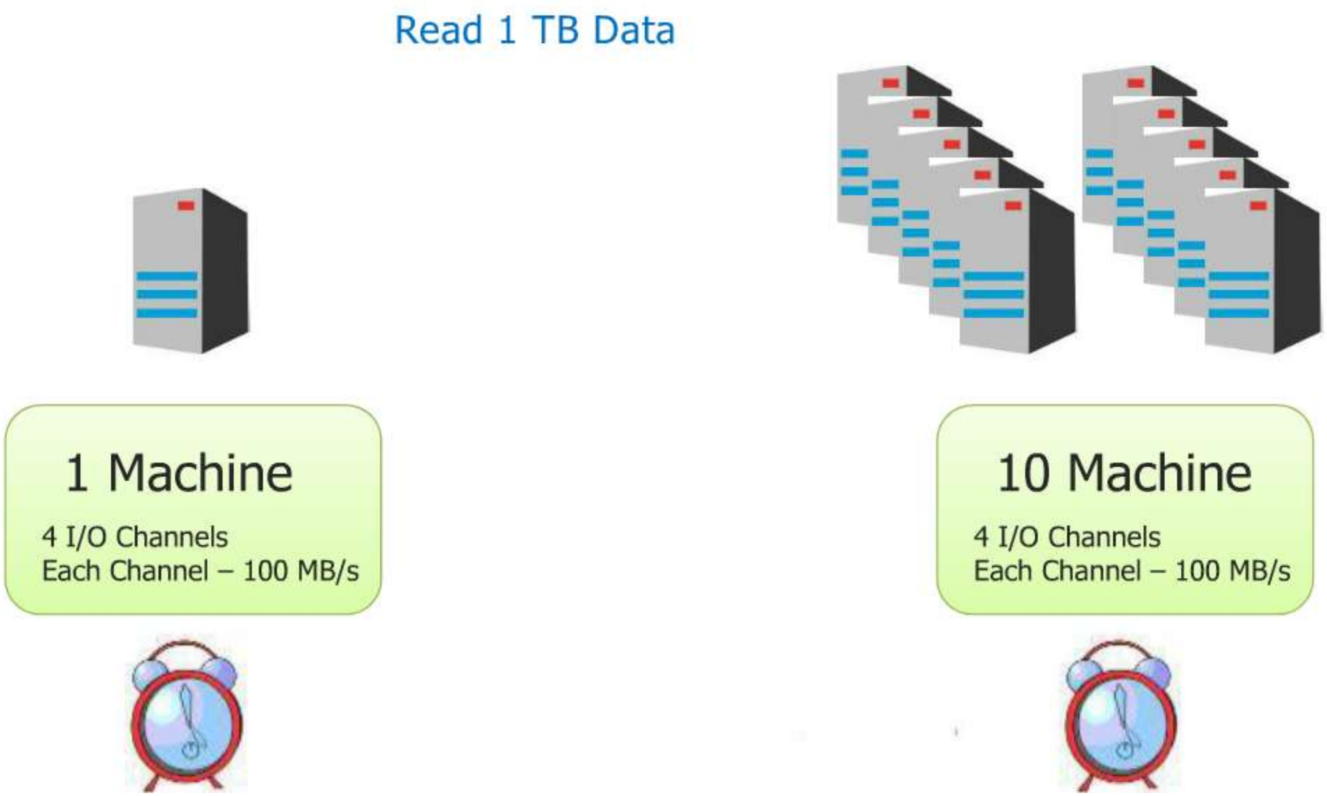
→ Some key early indicators can mean Fortunes to Business.

→ More Precise Analysis with more data.

Case Study: Sears Holding Corporation

*Sears was using traditional systems such as Oracle Exadata, Teradata and SAS etc., to store and process the customer activity and sales data.

WHY DFS? Distributed file system



Read 1 TB Data

1 Machine

4 I/O Channels
Each Channel – 100 MB/s

10 Machine

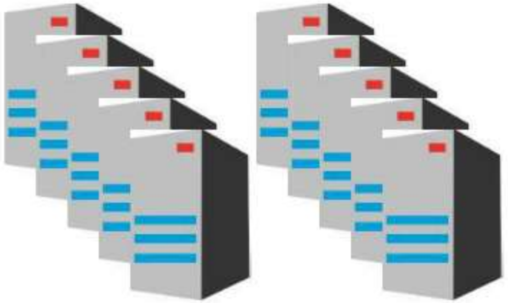
4 I/O Channels
Each Channel – 100 MB/s

Read 1 TB Data

1 Machine

4 I/O Channels
Each Channel – 100 MB/s

43 Minutes



10 Machine

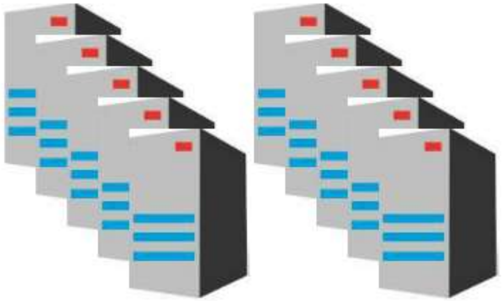
4 I/O Channels
Each Channel – 100 MB/s

Read 1 TB Data

1 Machine

4 I/O Channels
Each Channel – 100 MB/s

43 Minutes

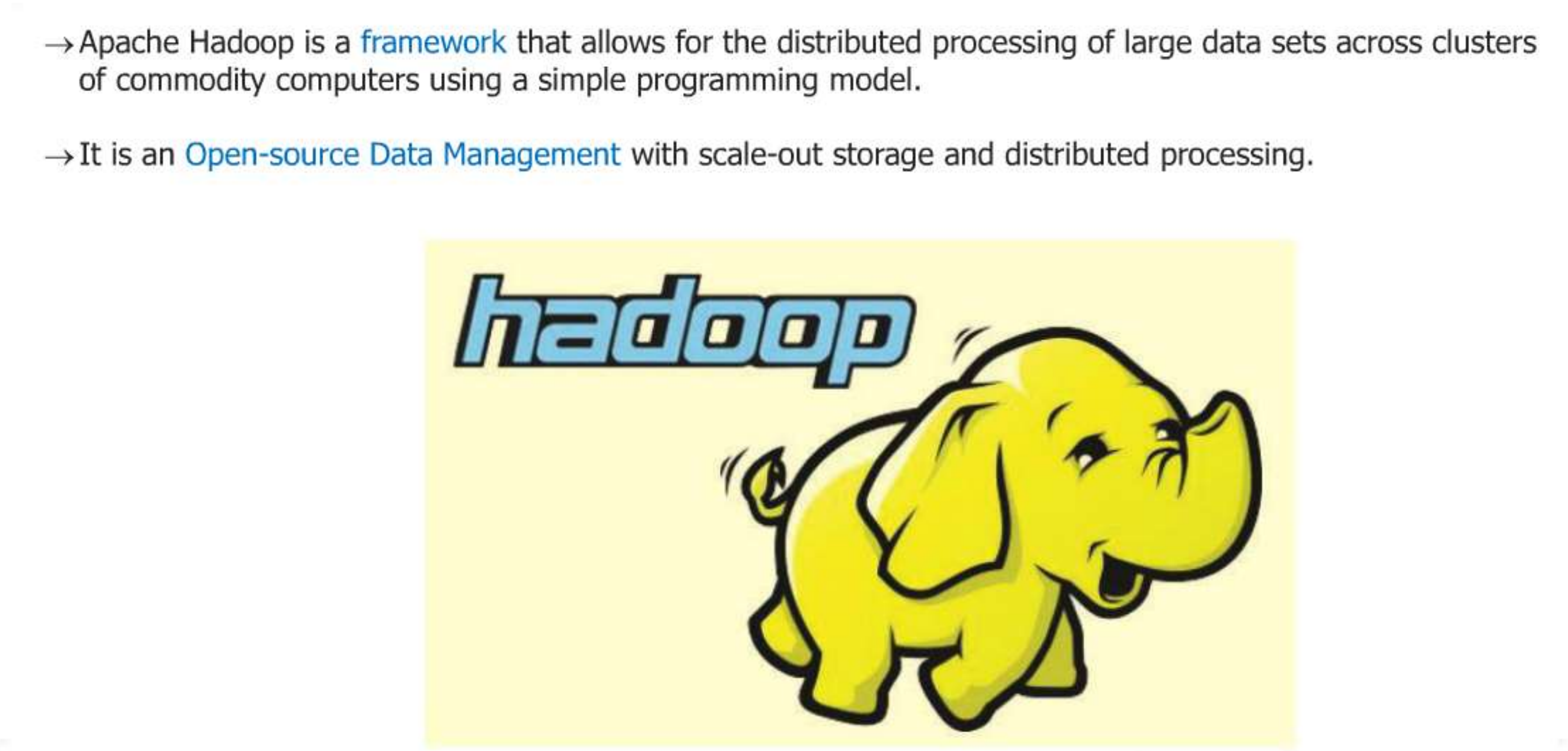


10 Machine

4 I/O Channels
Each Channel – 100 MB/s

4.3 Minutes

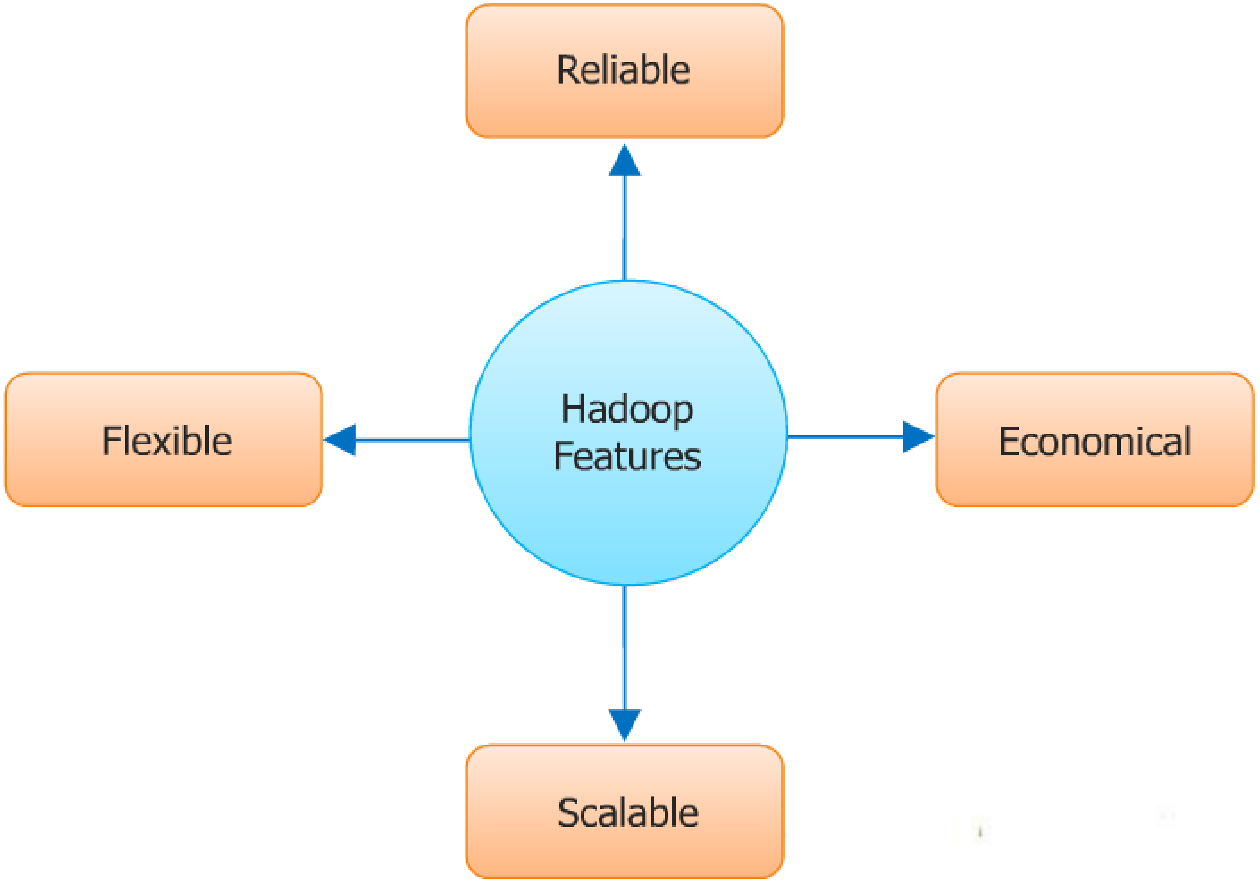
What is Hadoop?



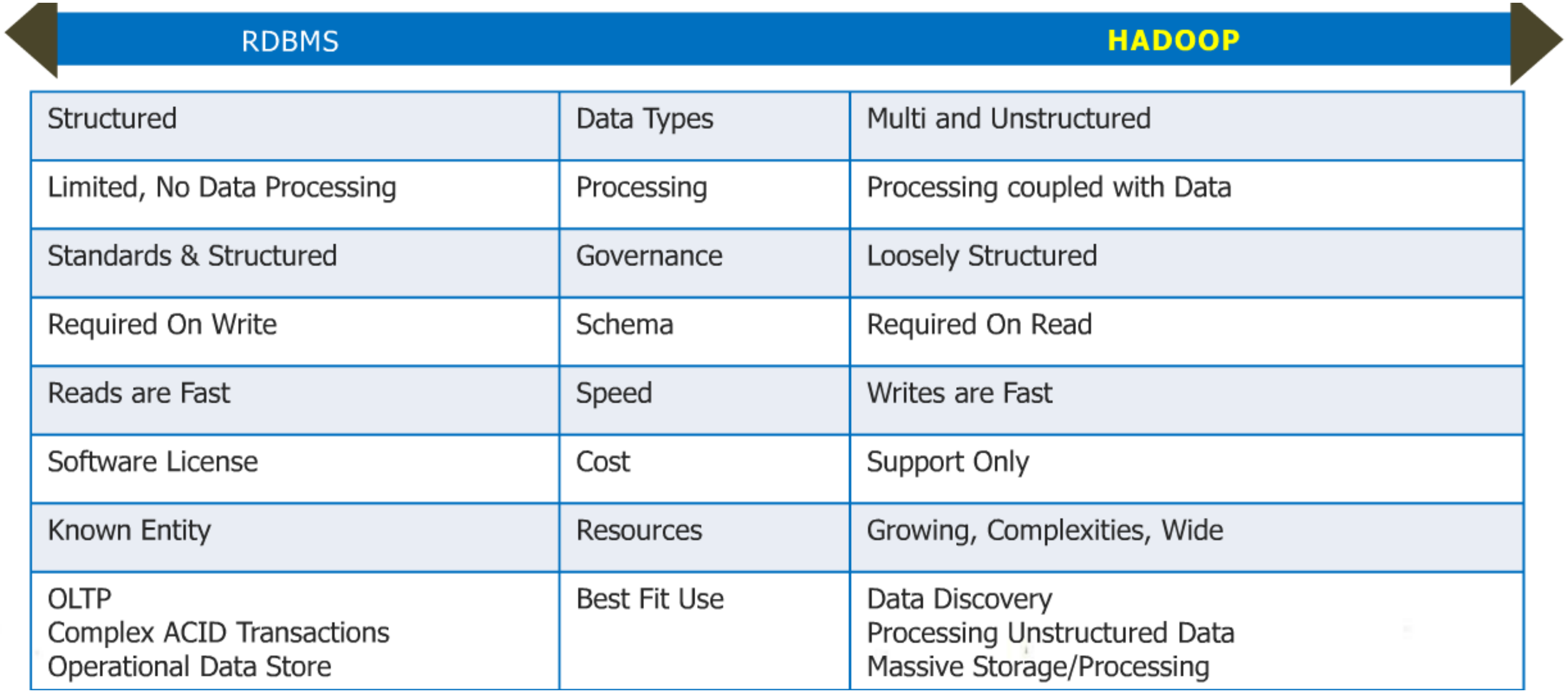
→Apache Hadoop is a framework that allows for the distributed processing of large data sets across clusters of commodity computers using a simple programming model.

→It is an Open-source Data Management with scale-out storage and distributed processing.

Characteristics

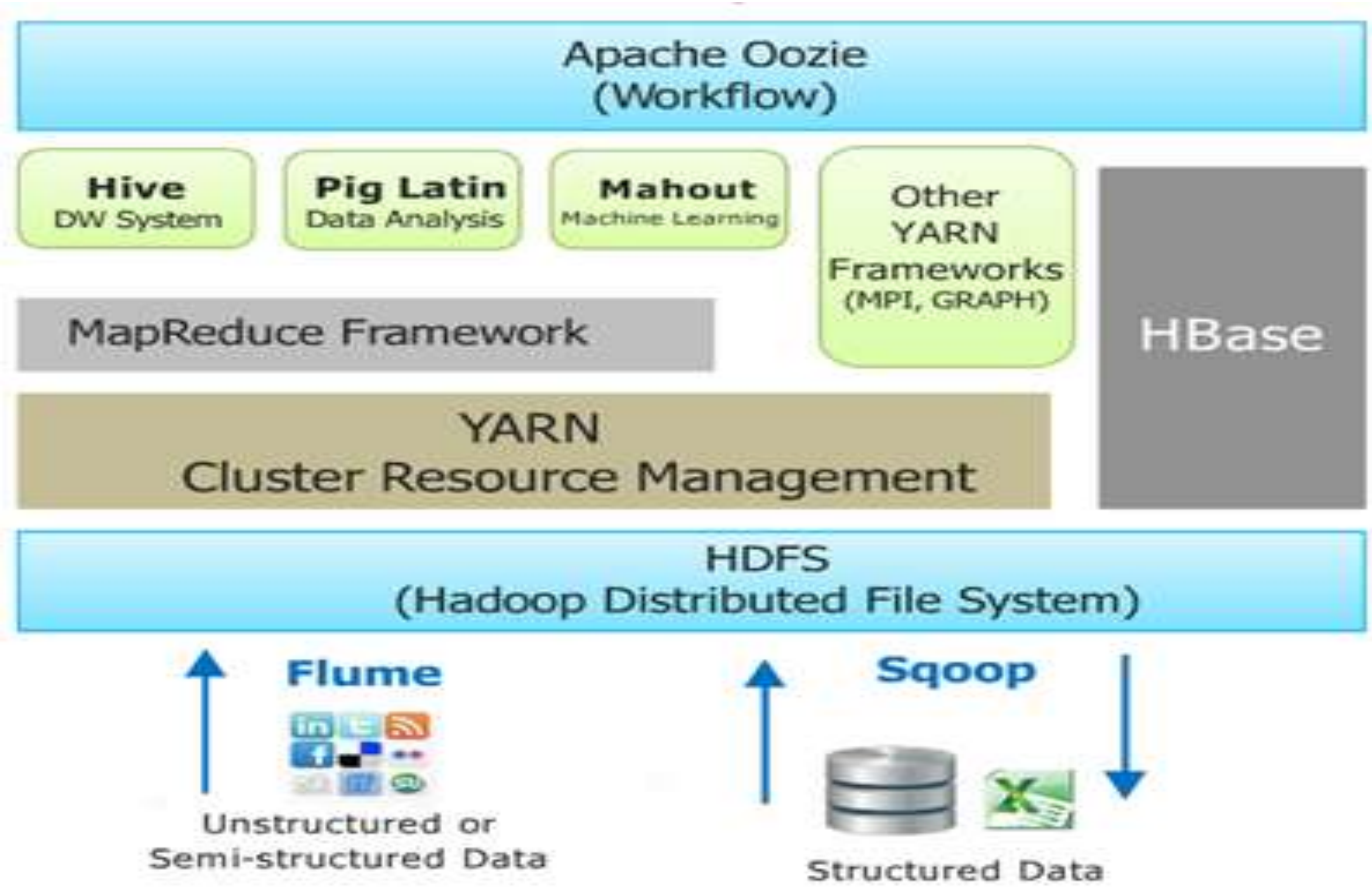


Hadoop-its about scale and structure

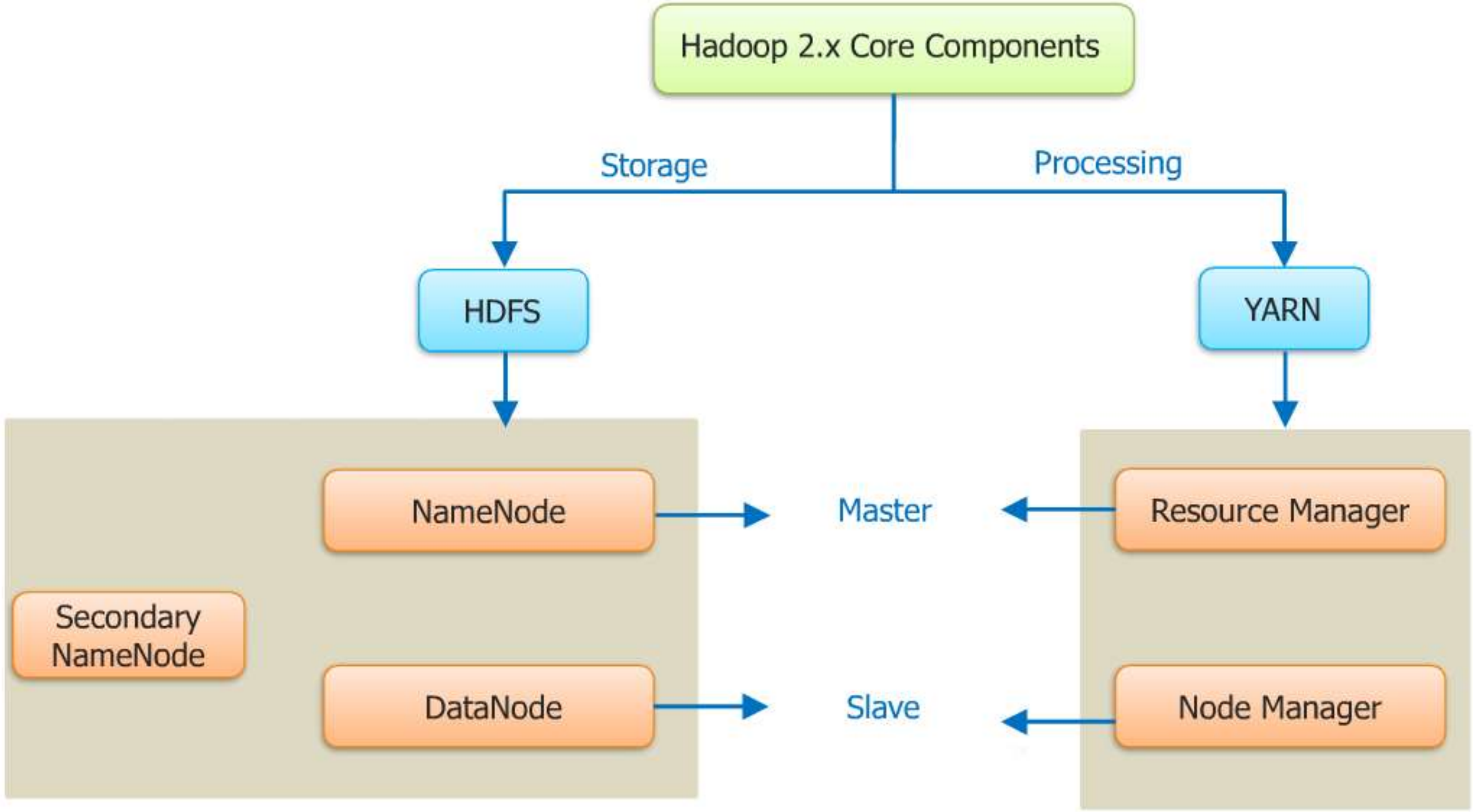


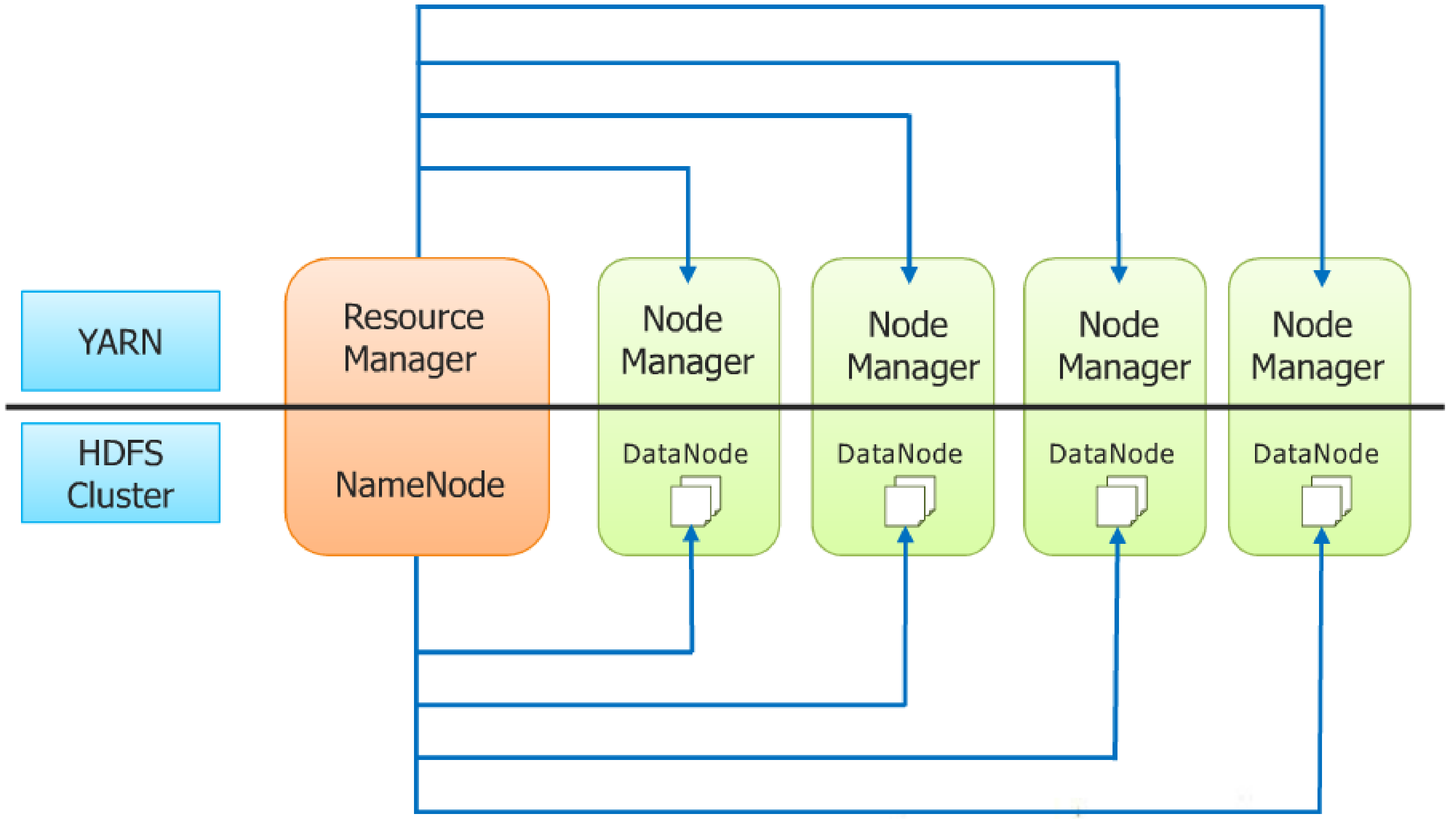
RDBMS		HADOOP
Structured	Data Types	Multi and Unstructured
Limited, No Data Processing	Processing	Processing coupled with Data
Standards & Structured	Governance	Loosely Structured
Required On Write	Schema	Required On Read
Reads are Fast	Speed	Writes are Fast
Software License	Cost	Support Only
Known Entity	Resources	Growing, Complexities, Wide
OLTP Complex ACID Transactions Operational Data Store	Best Fit Use	Data Discovery Processing Unstructured Data Massive Storage/Processing

Hadoop ecosystem



Hadoop 2.X core components




YARN
HDFS Cluster
Resource Manager
NameNode
Node Manager
DataNode
Node Manager
DataNode
Node Manager
DataNode
Node Manager
DataNode

Main components of HDFS

→ NameNode:

- » Master of the system
- » Maintains and manages the blocks which are present on the DataNodes

→ DataNodes:

- » Slaves which are deployed on each machine and provide the actual storage
- » Responsible for serving read and write requests for the clients

Name Node Metadata

→ Meta-data in Memory

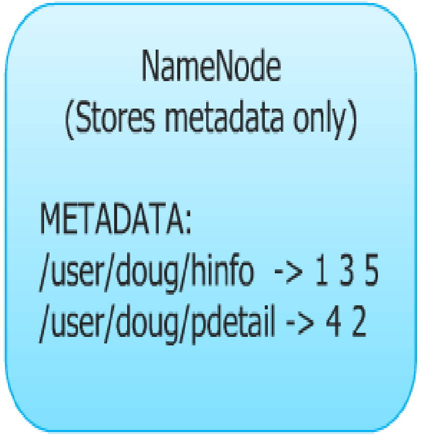
- » The entire metadata is in main memory
- » No demand paging of FS meta-data

→ Types of Metadata

- » List of files
- » List of Blocks for each file
- » List of DataNode for each block
- » File attributes, e.g. access time, replication factor

→ A Transaction Log

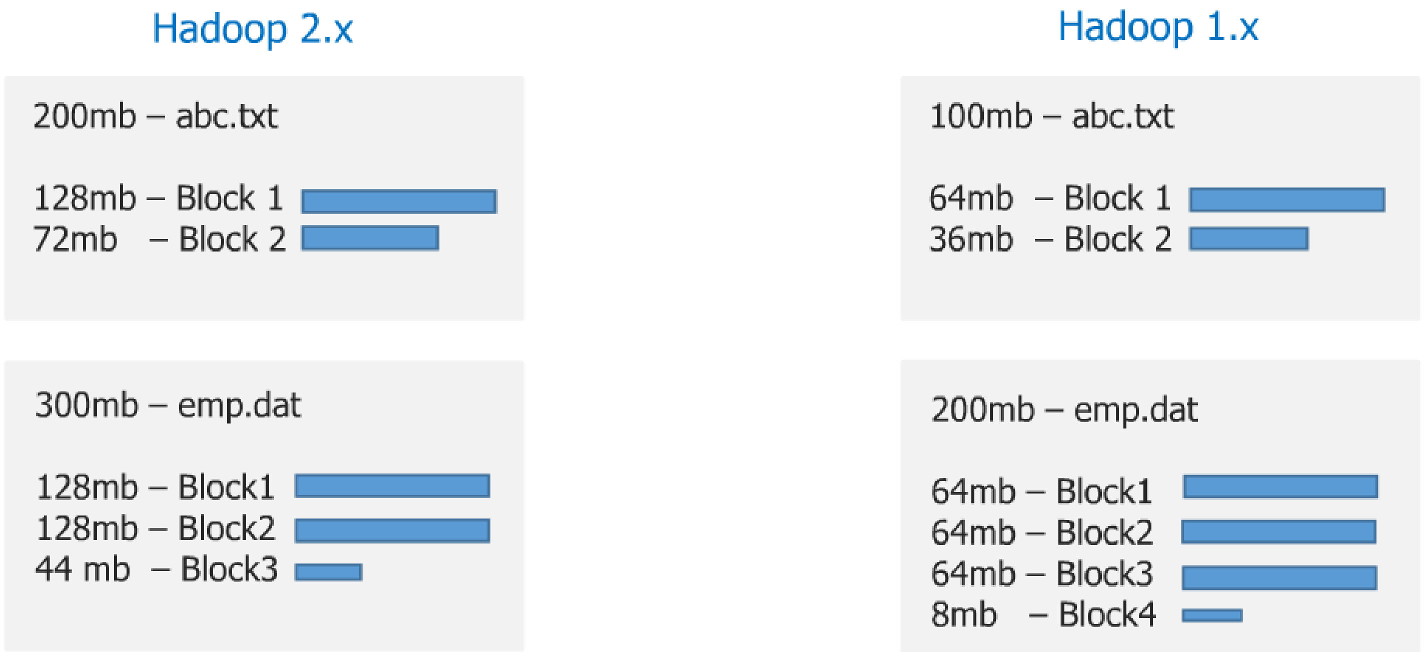
- » Records file creations, file deletions etc.



NameNode:
Keeps track of overall file directory structure and the placement of Data Block

File Blocks

→By Default, block size is **128mb** in Hadoop 2.x and **64mb** in Hadoop 1.x



Hadoop 2.x

200mb – abc.txt

128mb – Block 1
72mb – Block 2

300mb – emp.dat

128mb – Block1
128mb – Block2
44 mb – Block3

Hadoop 1.x

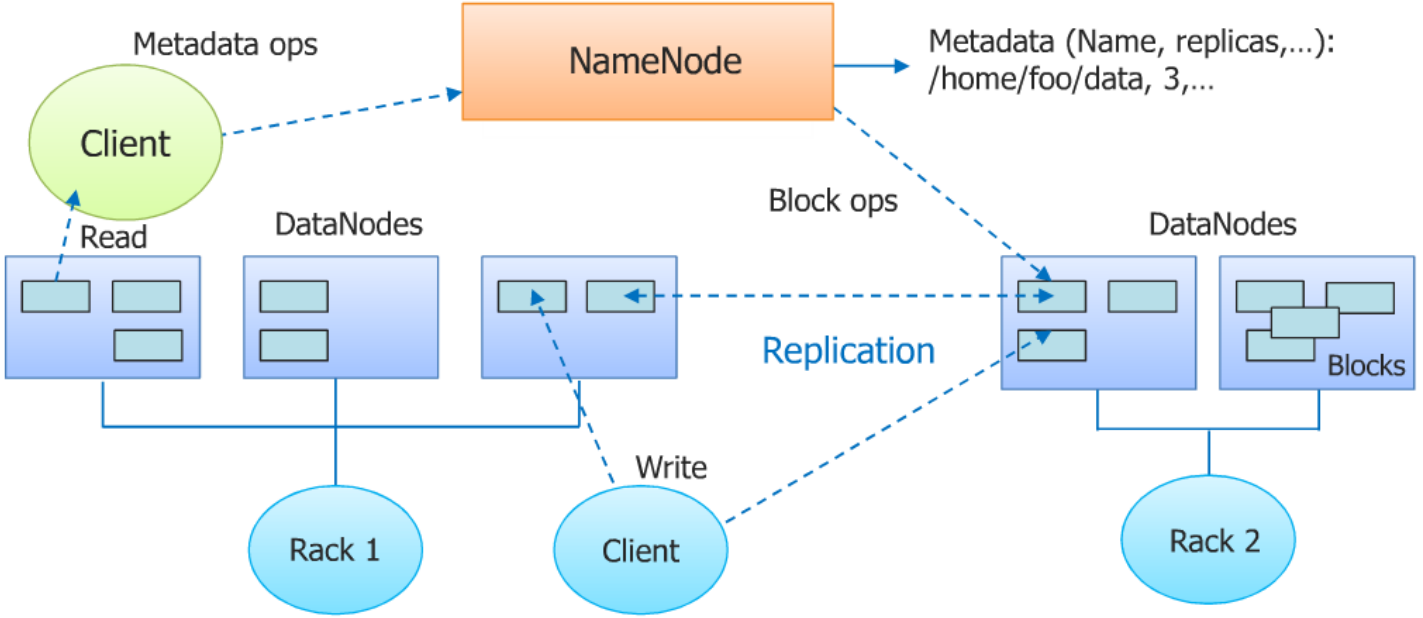
100mb – abc.txt

64mb – Block 1
36mb – Block 2

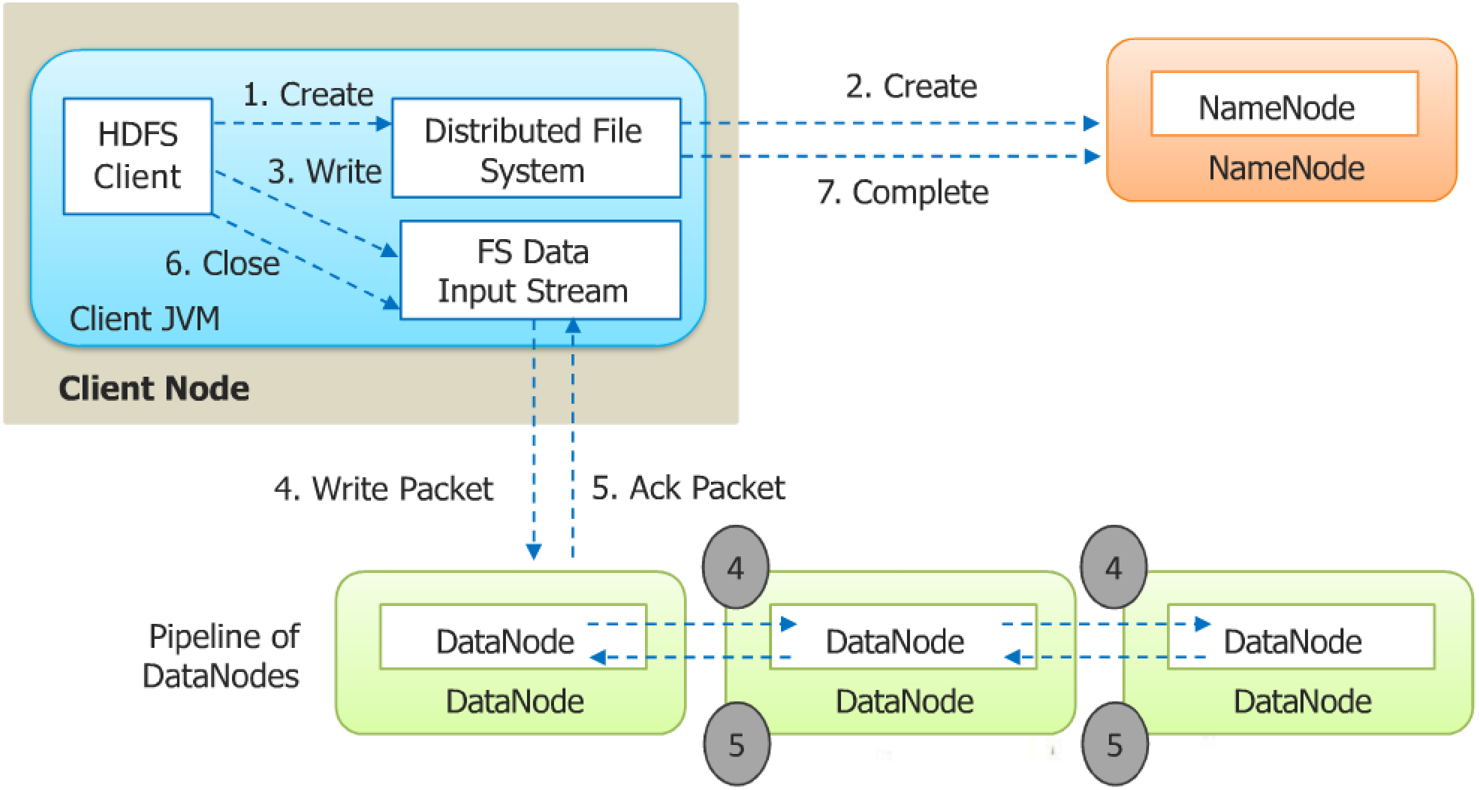
200mb – emp.dat

64mb – Block1
64mb – Block2
64mb – Block3
8mb – Block4

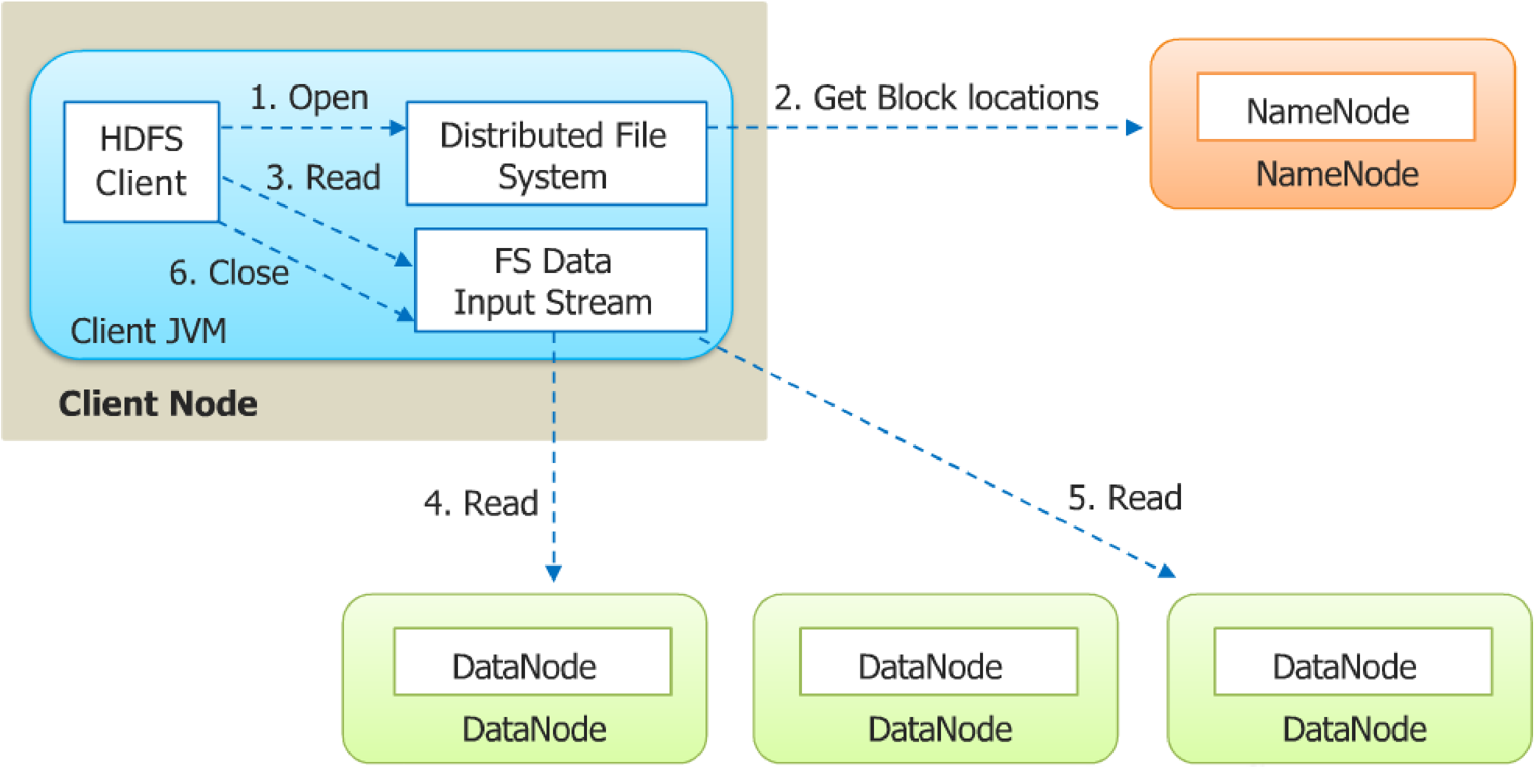
HDFS Architecture

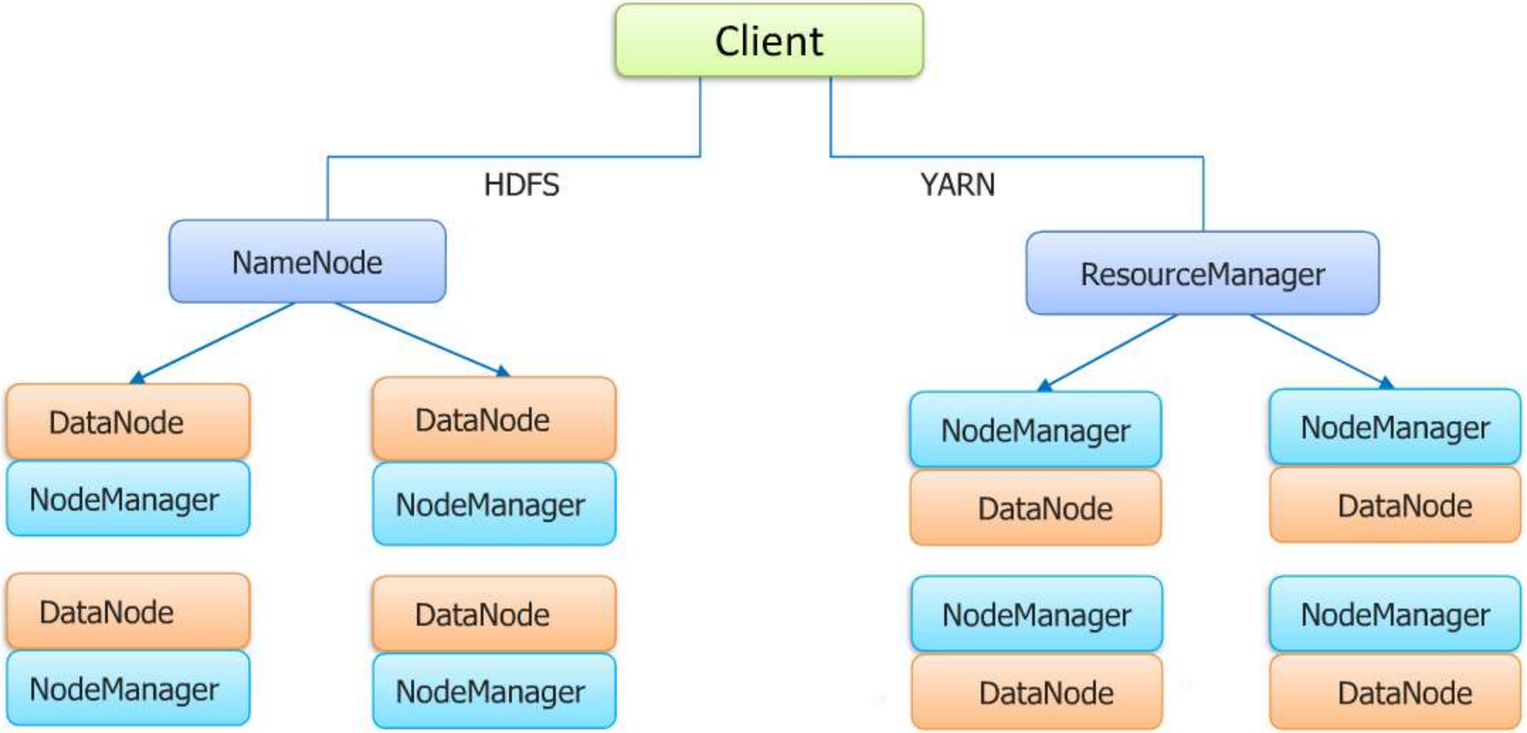


Anatomy of a file write

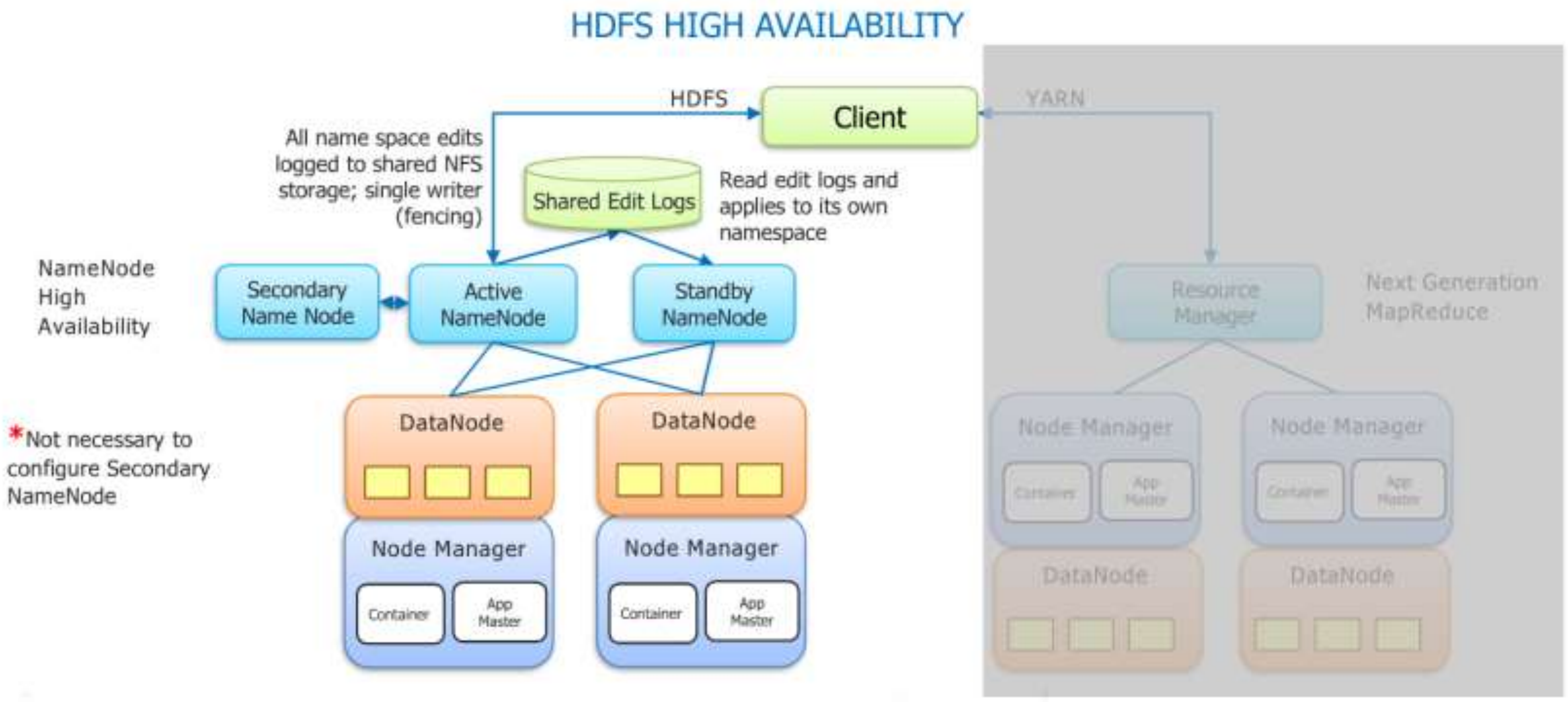


Anatomy of a file read



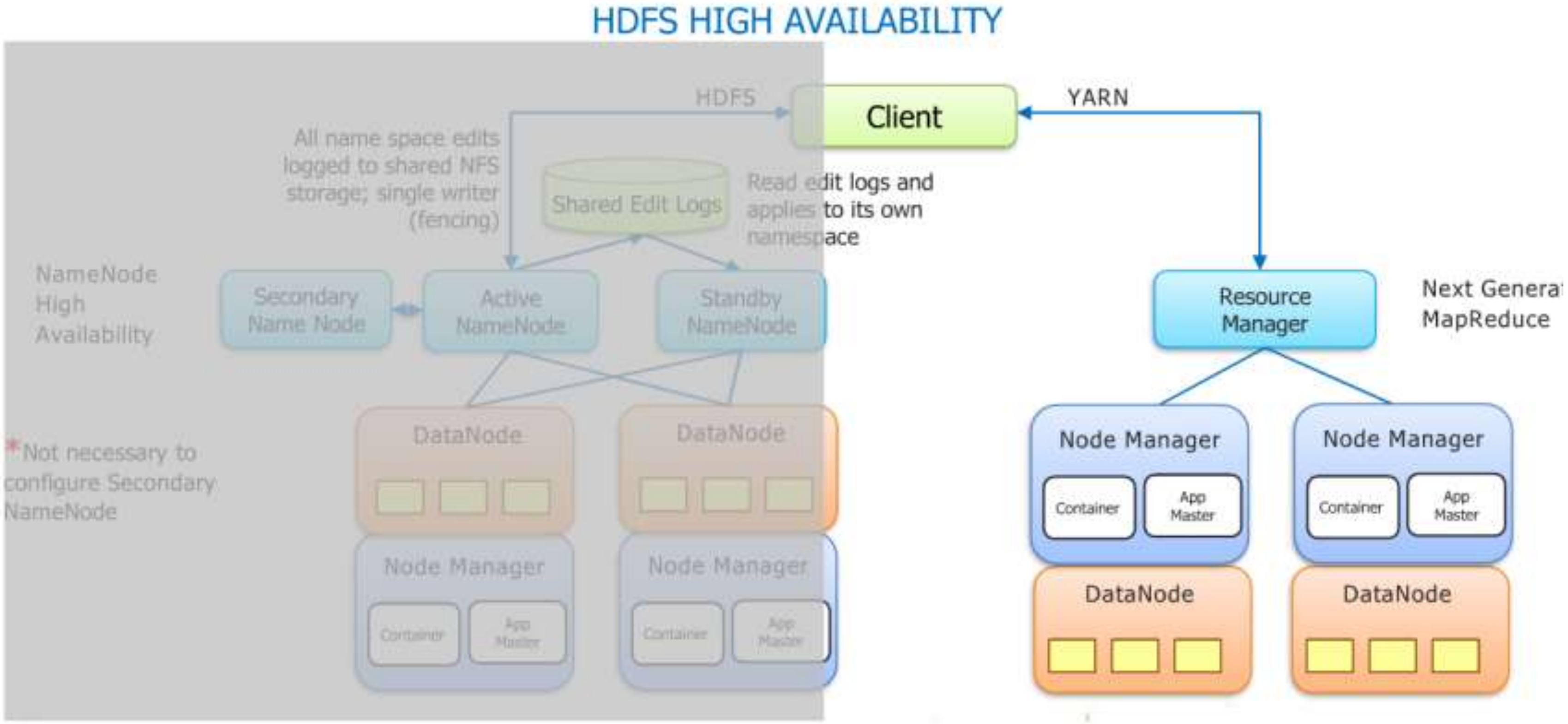

Client
HDFS
YARN
NameNode
ResourceManager
DataNode
NodeManager
DataNode
NodeManager
DataNode
NodeManager
DataNode
NodeManager
NodeManager
DataNode
NodeManager
DataNode
NodeManager
DataNode
NodeManager
DataNode

Hadoop 2.x-High availability

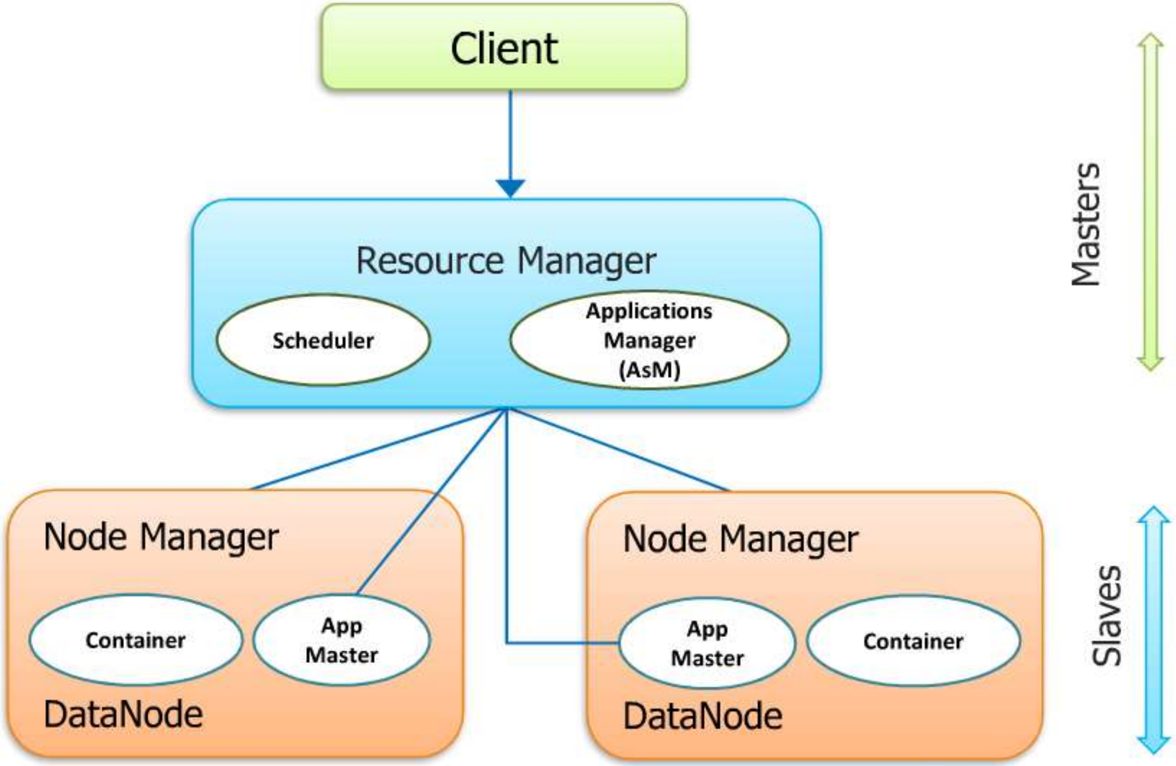


http://hadoop.apache.org/docs/stable2/hadoop-yarn/hadoop-yarn-site/HDFSHighAvailabilityWithNFS.html

Hadoop 2.x-Resource management

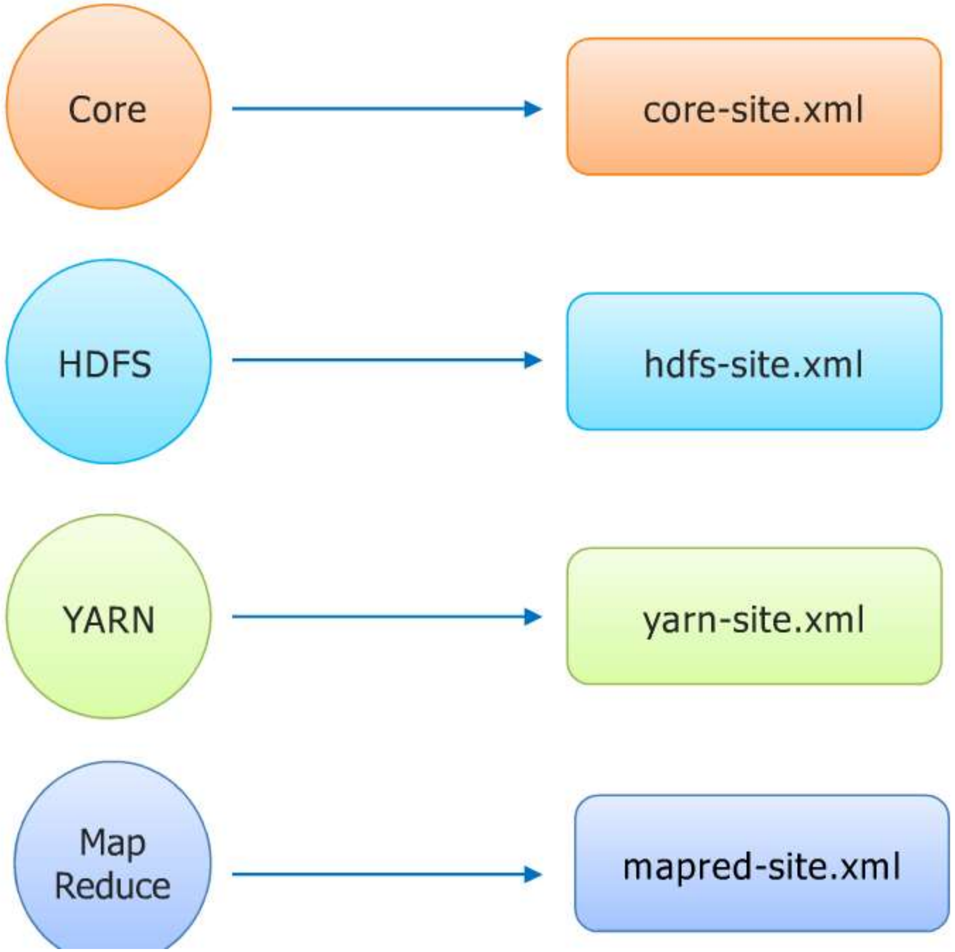


http://hadoop.apache.org/docs/stable2/hadoop-yarn/hadoop-yarn-site/HDFSHighAvailabilityWithNFS.html

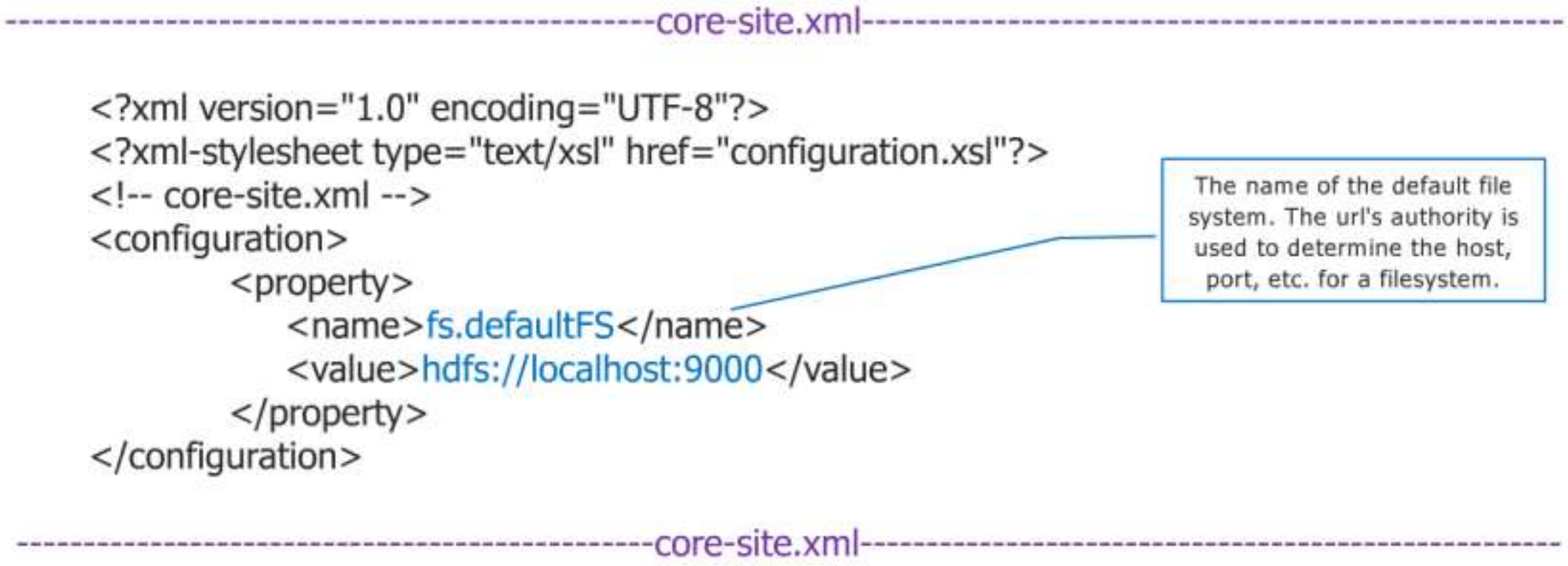


YARN – Yet Another Resource Negotiator

Configuration files



Core-site.xml



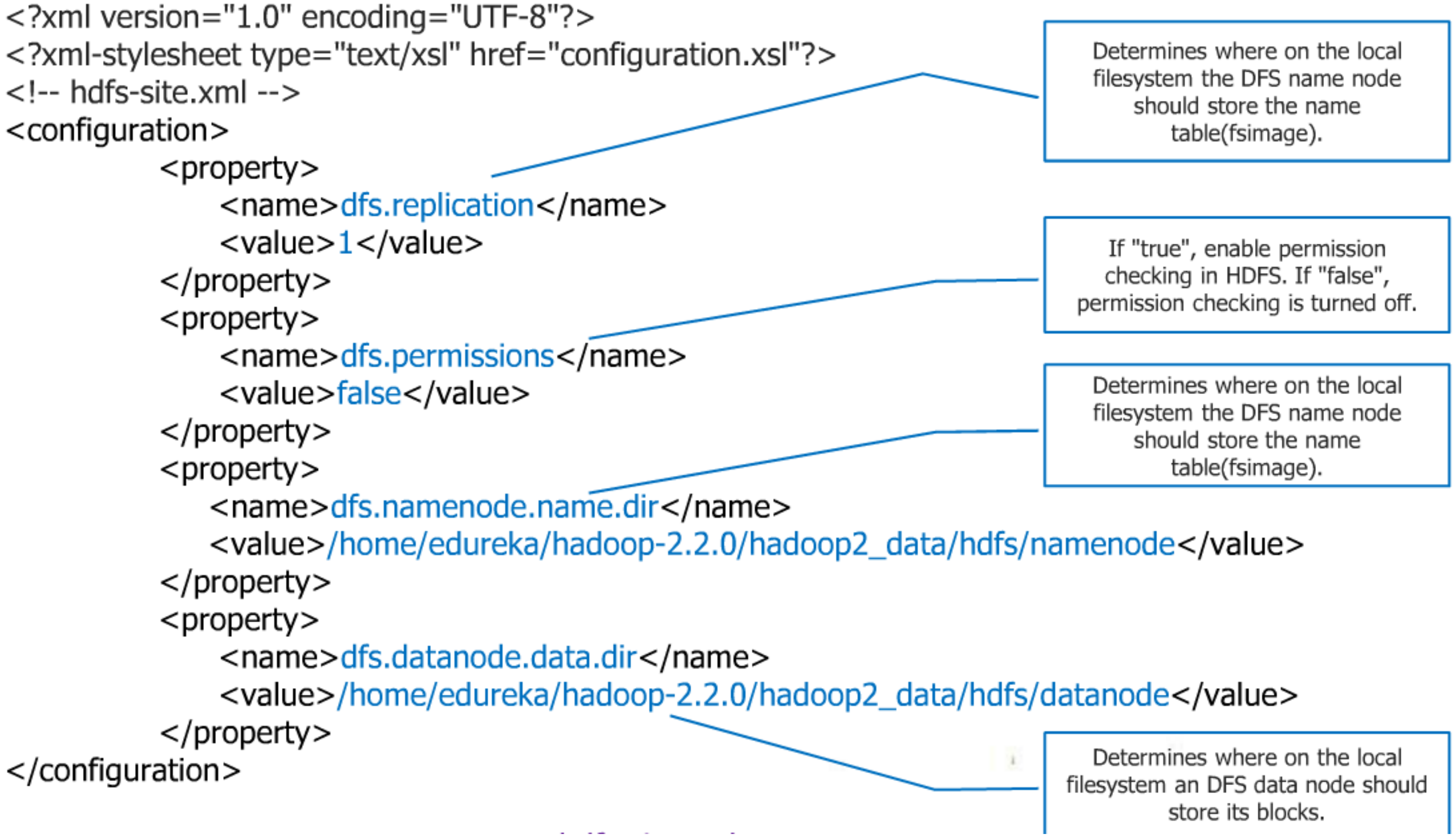
---core-site.xml---

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!-- core-site.xml -->
<configuration>
          <property>
              <name>fs.defaultFS</name>
              <value>hdfs://localhost:9000</value>
          </property>
</configuration>
```

The name of the default file system. The url's authority is used to determine the host, port, etc. for a filesystem.

---core-site.xml---

hdfs-site.xml



```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!-- hdfs-site.xml -->
<configuration>
        <property>
           <name>dfs.replication</name>
           <value>1</value>
        </property>
        <property>
           <name>dfs.permissions</name>
           <value>false</value>
        </property>
        <property>
          <name>dfs.namenode.name.dir</name>
          <value>/home/edureka/hadoop-2.2.0/hadoop2_data/hdfs/namenode</value>
        </property>
        <property>
           <name>dfs.datanode.data.dir</name>
           <value>/home/edureka/hadoop-2.2.0/hadoop2_data/hdfs/datanode</value>
        </property>
</configuration>
```

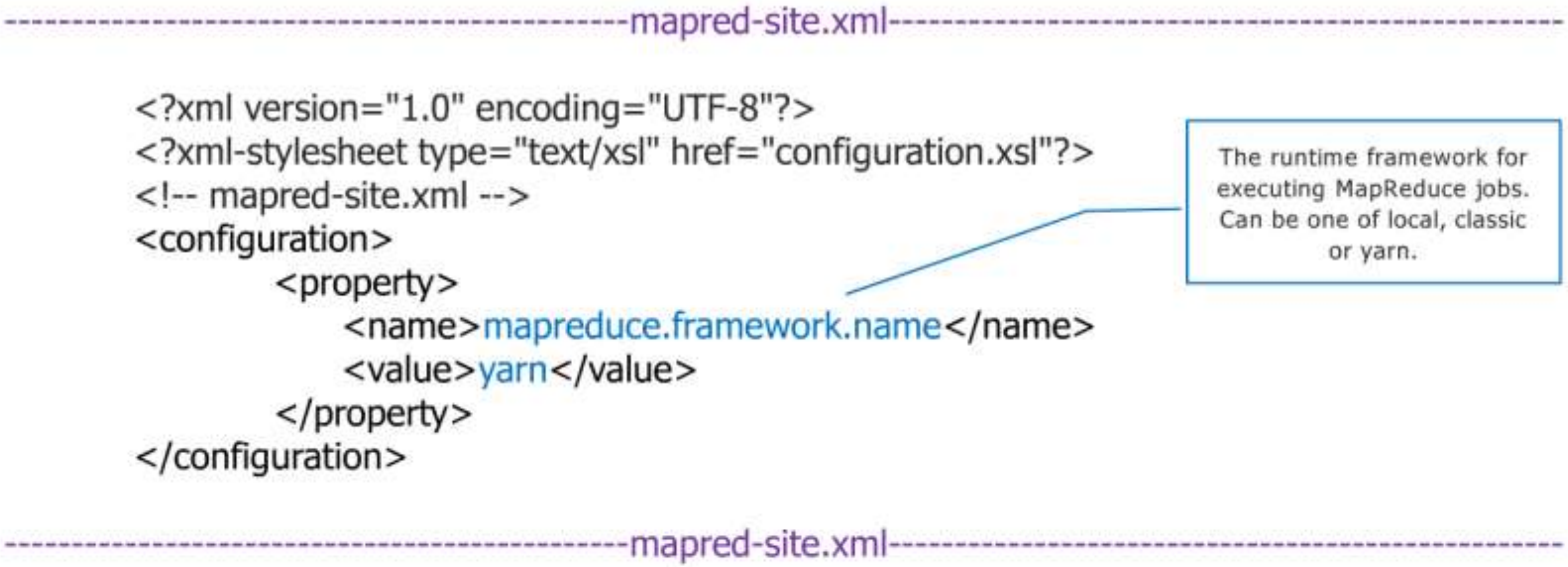
Determines where on the local filesystem the DFS name node should store the name table(fsimage).

If "true", enable permission checking in HDFS. If "false", permission checking is turned off.

Determines where on the local filesystem the DFS name node should store the name table(fsimage).

Determines where on the local filesystem an DFS data node should store its blocks.

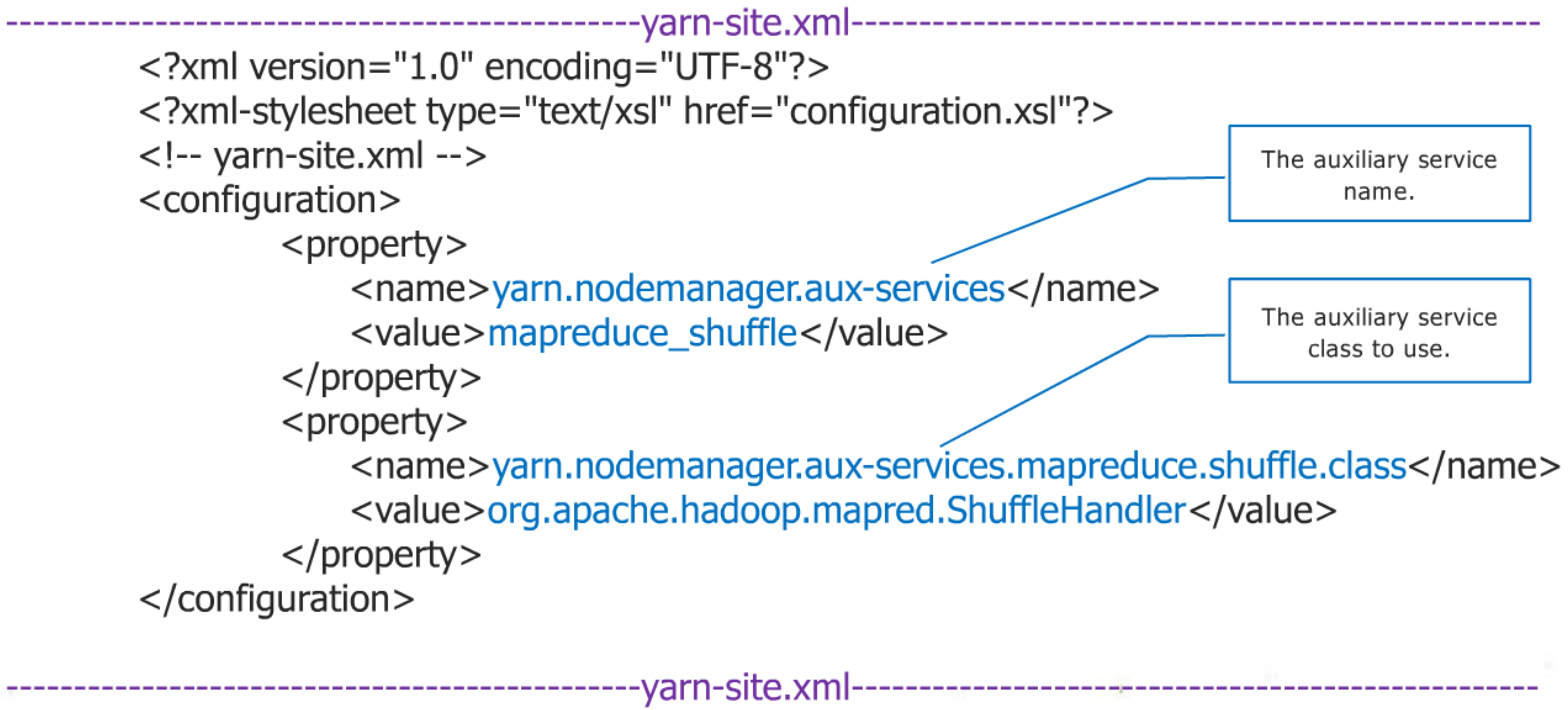
mapred-site.xml



```
---------------------------------------------mapred-site.xml---------------------------------------------

<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!-- mapred-site.xml -->
<configuration>
          <property>
               <name>mapreduce.framework.name</name>
               <value>yarn</value>
          </property>
</configuration>

---------------------------------------------mapred-site.xml---------------------------------------------
```

The runtime framework for executing MapReduce jobs. Can be one of local, classic or yarn.

yarn-site.xml



```
-------------------------------------------yarn-site.xml-------------------------------------------
        <?xml version="1.0" encoding="UTF-8"?>
        <?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
        <!-- yarn-site.xml -->
        <configuration>
                <property>
                    <name>yarn.nodemanager.aux-services</name>
                    <value>mapreduce_shuffle</value>
                </property>
                <property>
                    <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
                    <value>org.apache.hadoop.mapred.ShuffleHandler</value>
                </property>
        </configuration>

-------------------------------------------yarn-site.xml-------------------------------------------
```

The auxiliary service name.

The auxiliary service class to use.

Slaves and masters

Two files are used by the startup and shutdown commands:

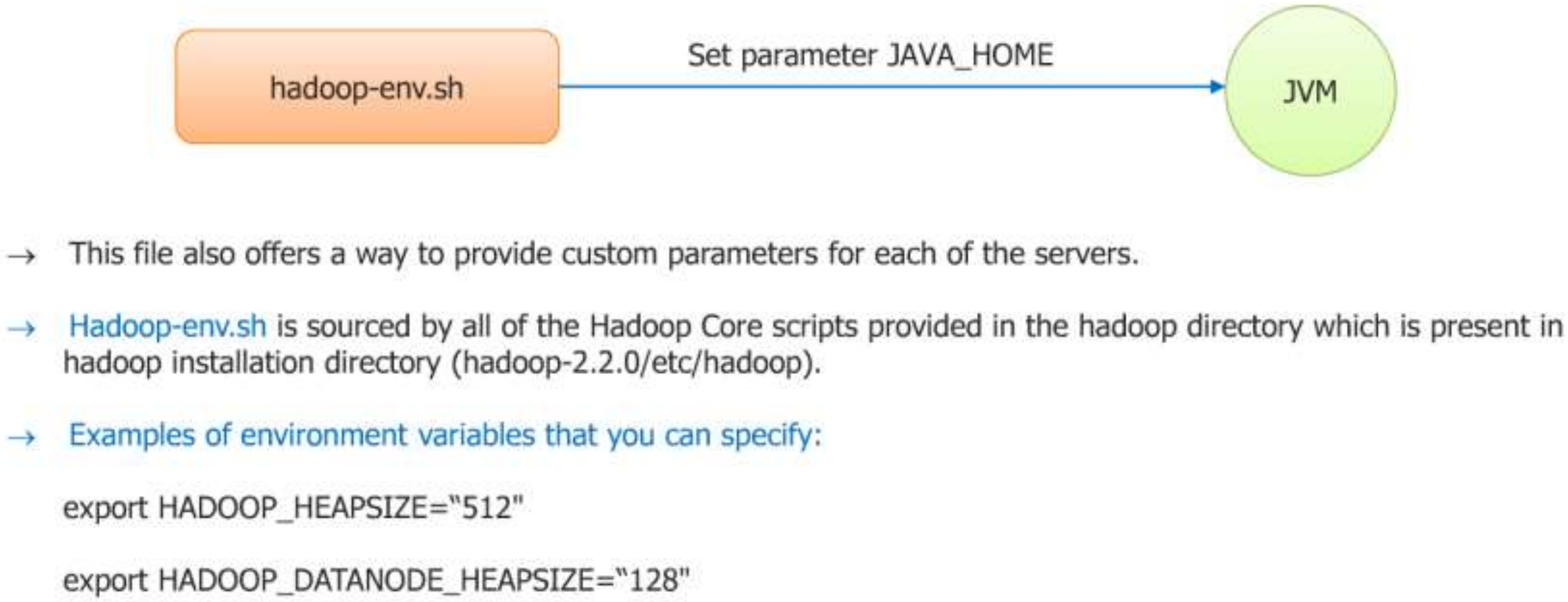
Slaves

→ Contains a list of hosts, one per line, that are to host DataNode and NodeManager servers.

Masters

→ Contains a list of hosts, one per line, that are to host Secondary NameNode servers.

Pre process run time environment


hadoop-env.sh
Set parameter JAVA_HOME
JVM

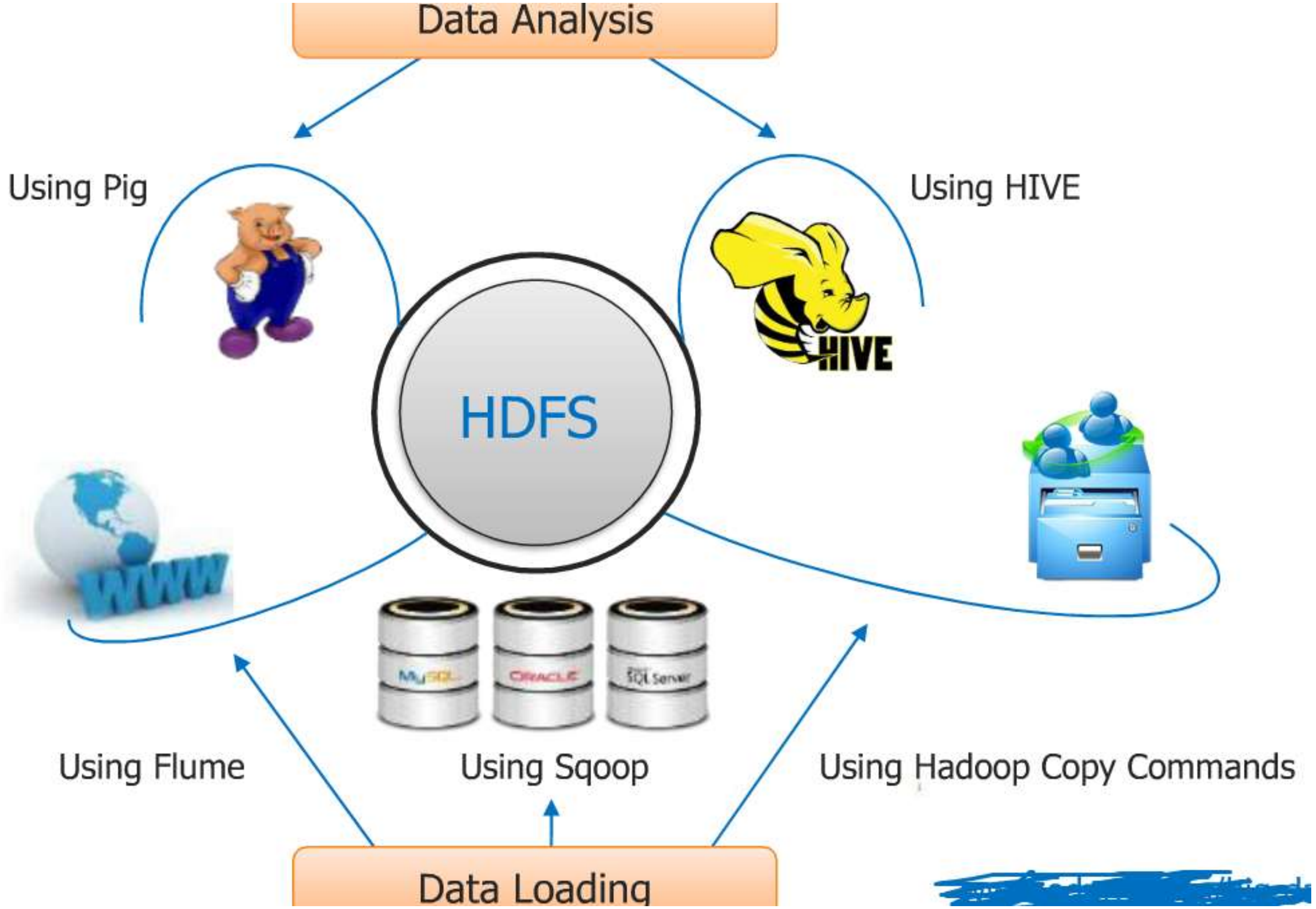
→ This file also offers a way to provide custom parameters for each of the servers.

→ Hadoop-env.sh is sourced by all of the Hadoop Core scripts provided in the hadoop directory which is present in hadoop installation directory (hadoop-2.2.0/etc/hadoop).

→ Examples of environment variables that you can specify:

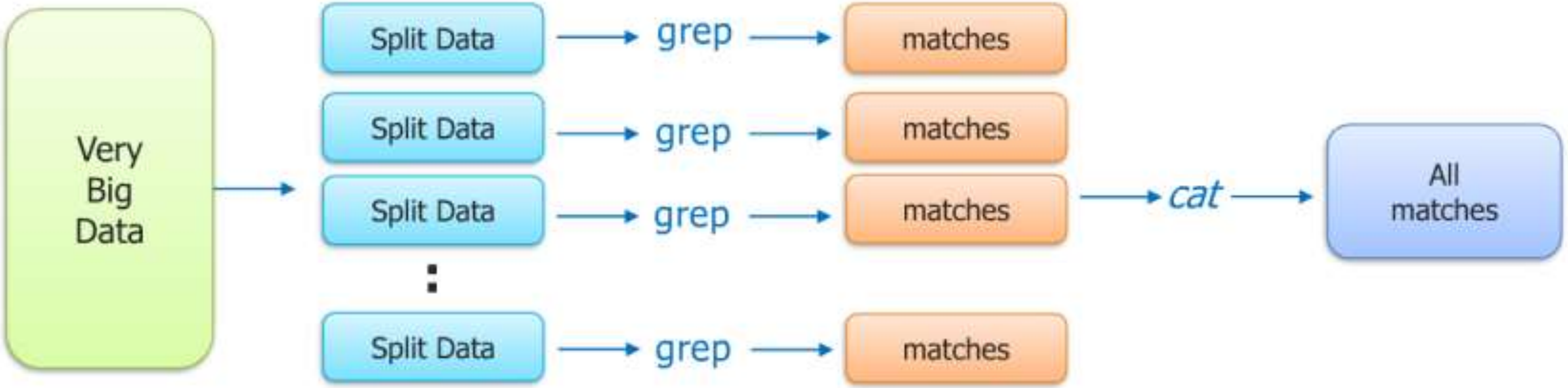
```
export HADOOP_HEAPSIZE="512"

export HADOOP_DATANODE_HEAPSIZE="128"
```

Data loading techniques and analysis

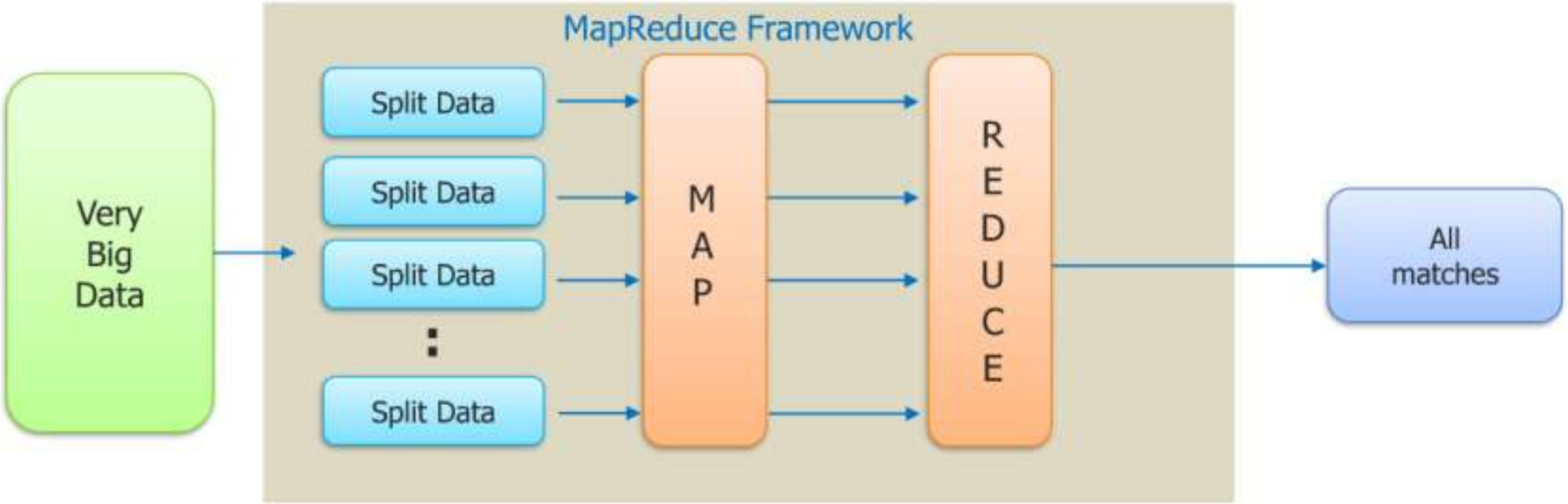


[redacted]

The Traditional way



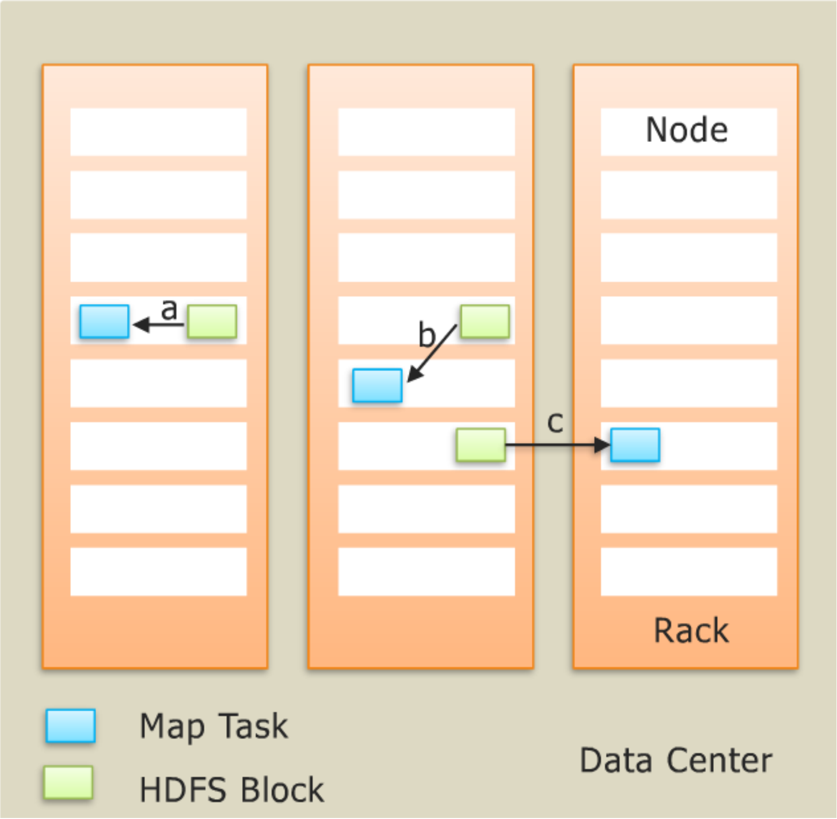
Map Reduce way



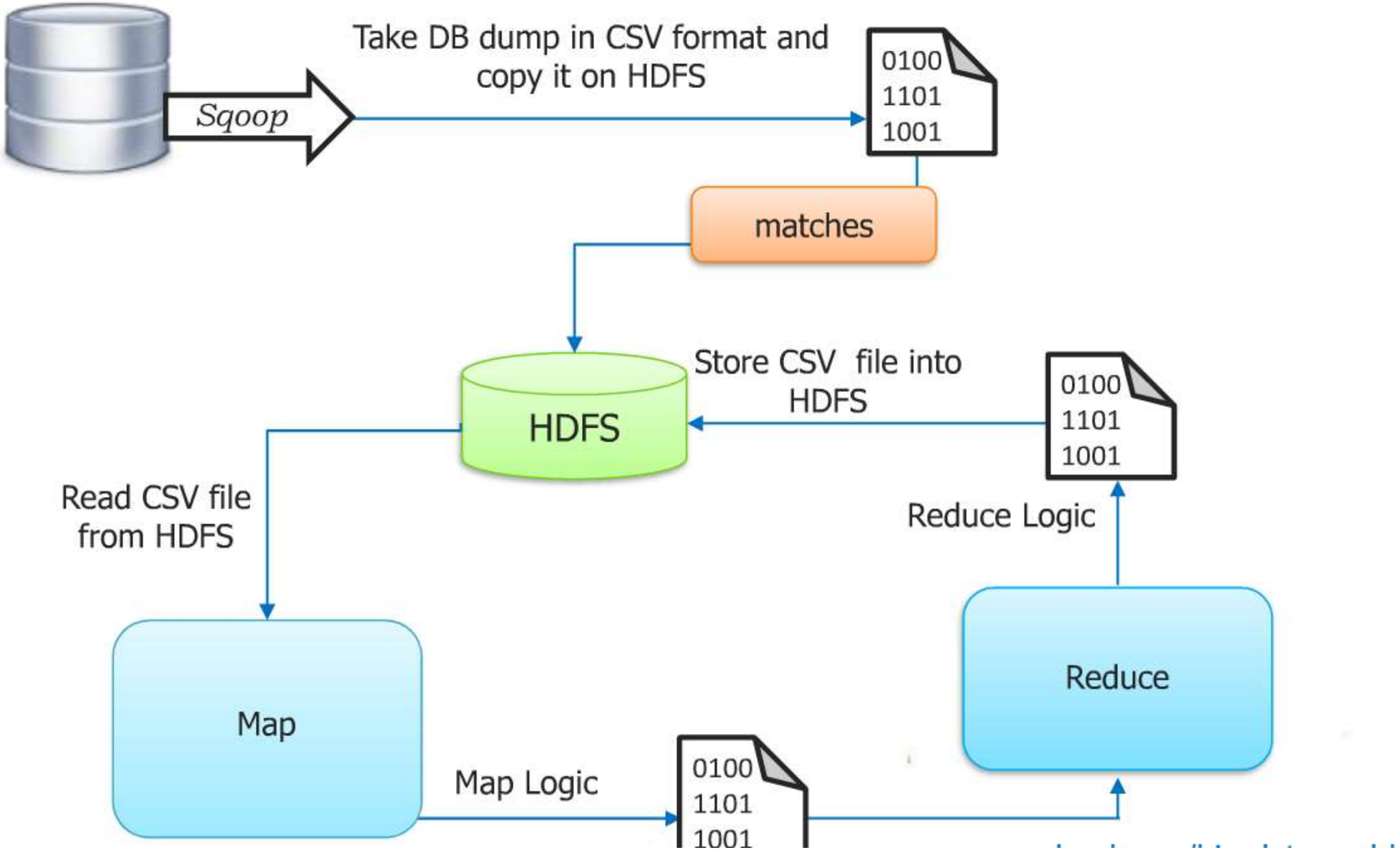
Why MAP REDUCE

→ Two biggest Advantages:

- » Taking processing to the data
- » Processing data in parallel



Solving the problem with map reduce



Sqoop

Take DB dump in CSV format and copy it on HDFS

0100
1101
1001

matches

HDFS

Store CSV file into HDFS

0100
1101
1001

Read CSV file from HDFS

Reduce Logic

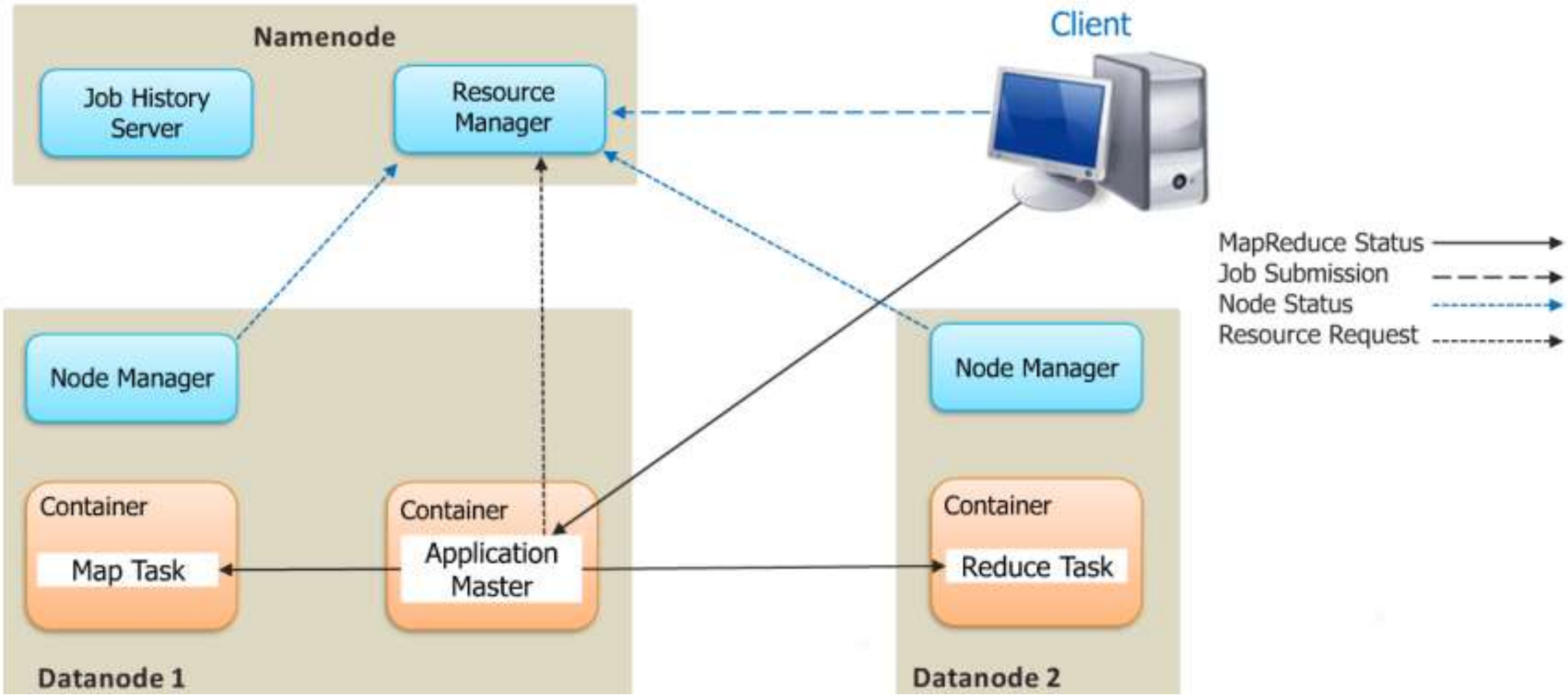
Reduce

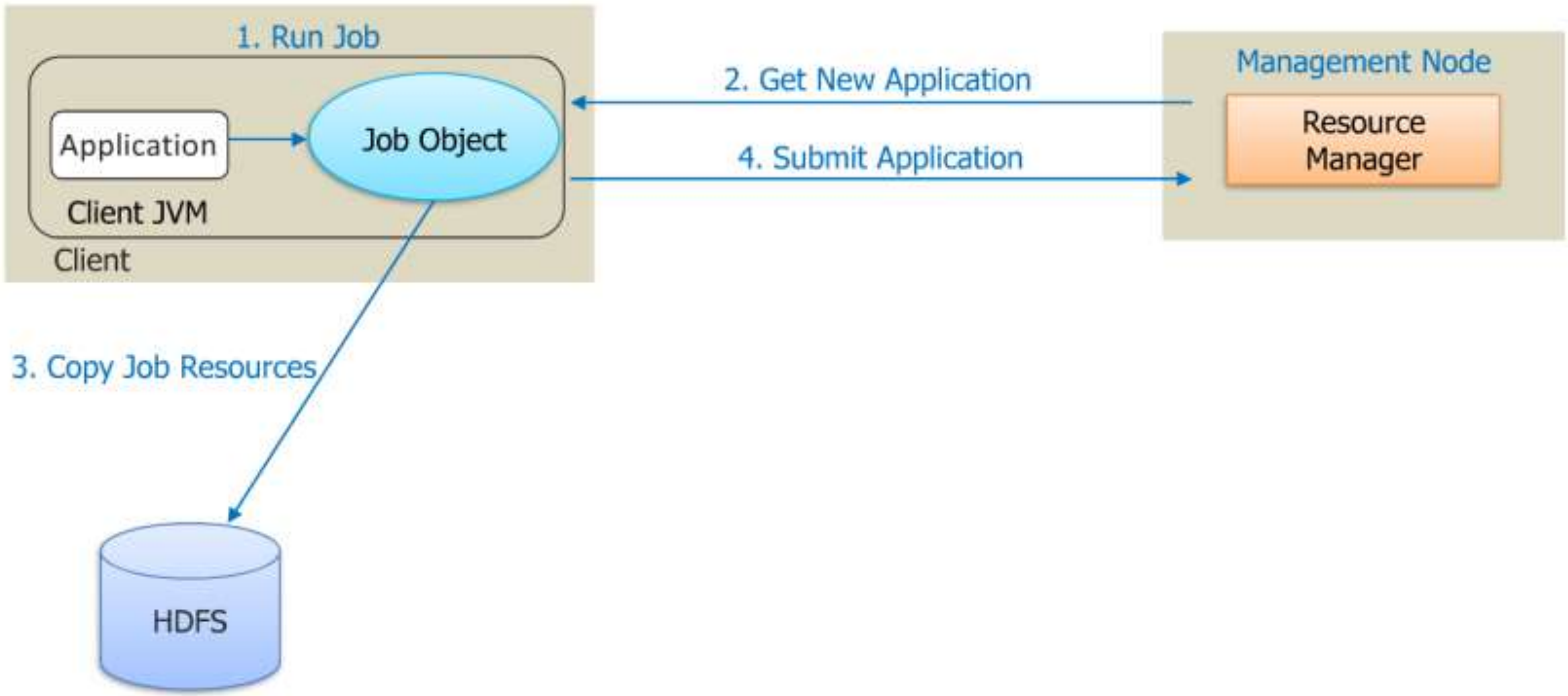
Map

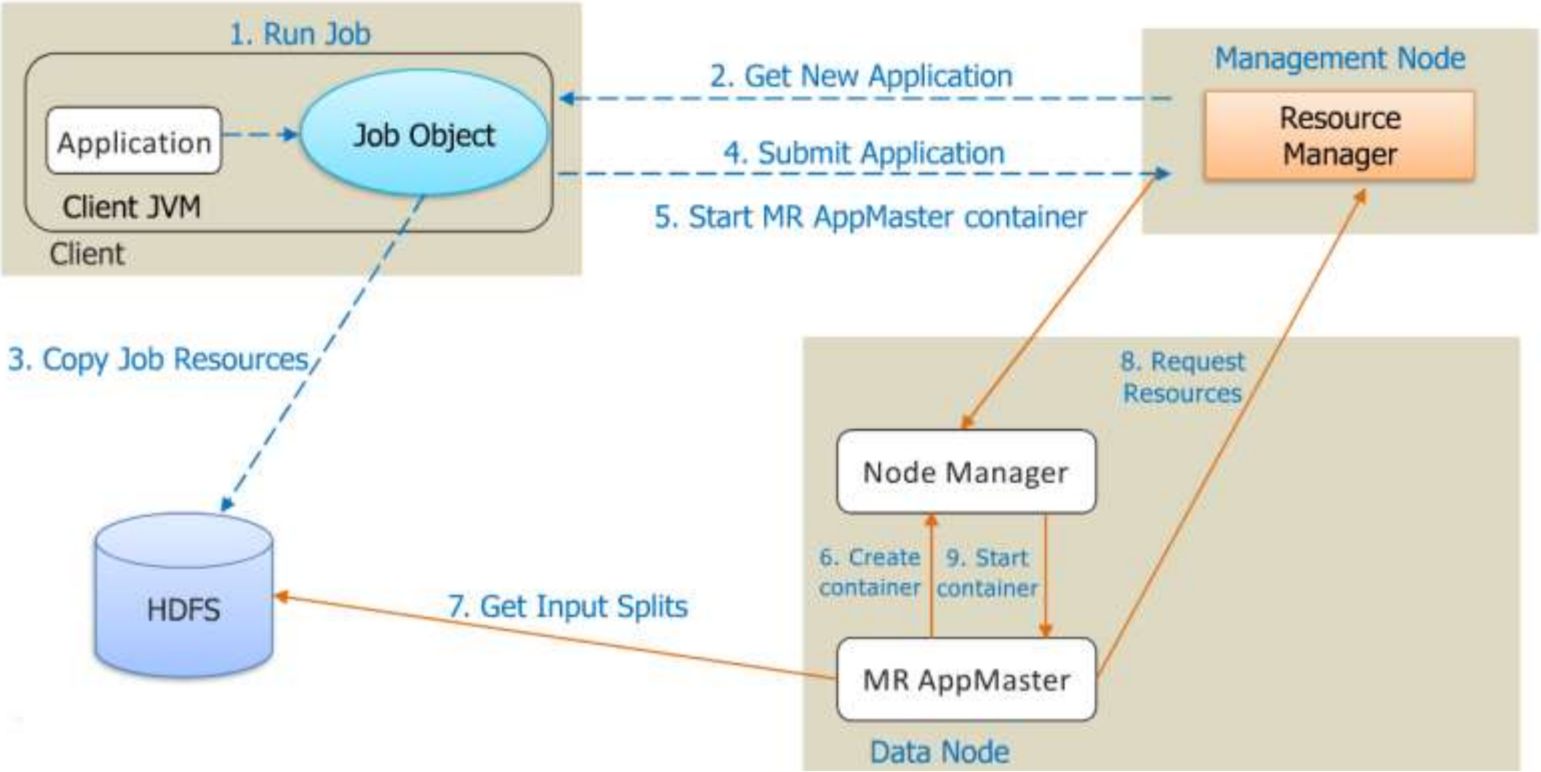
Map Logic

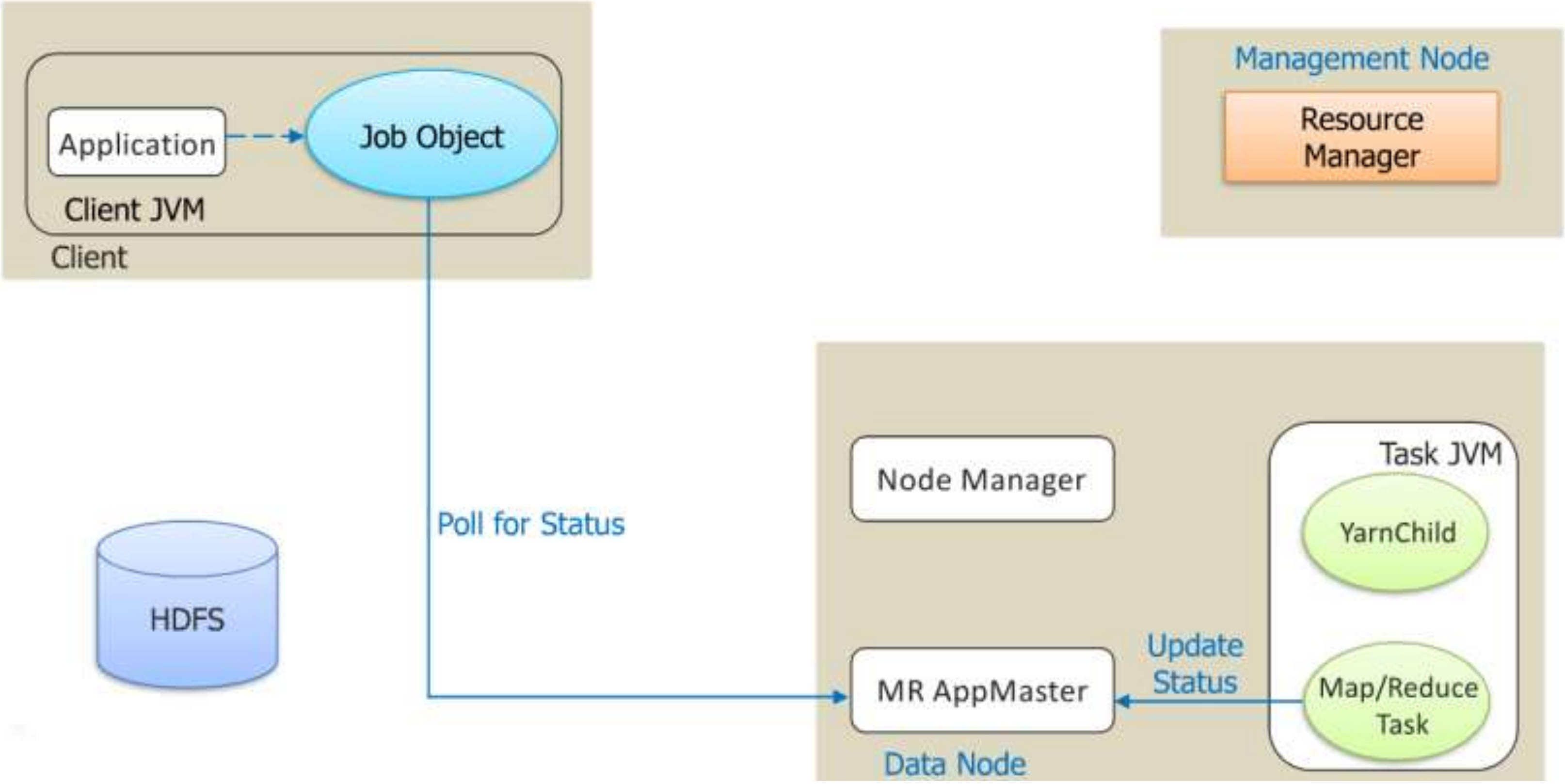
0100
1101
1001

Hadoop 2.x map reduce architecture

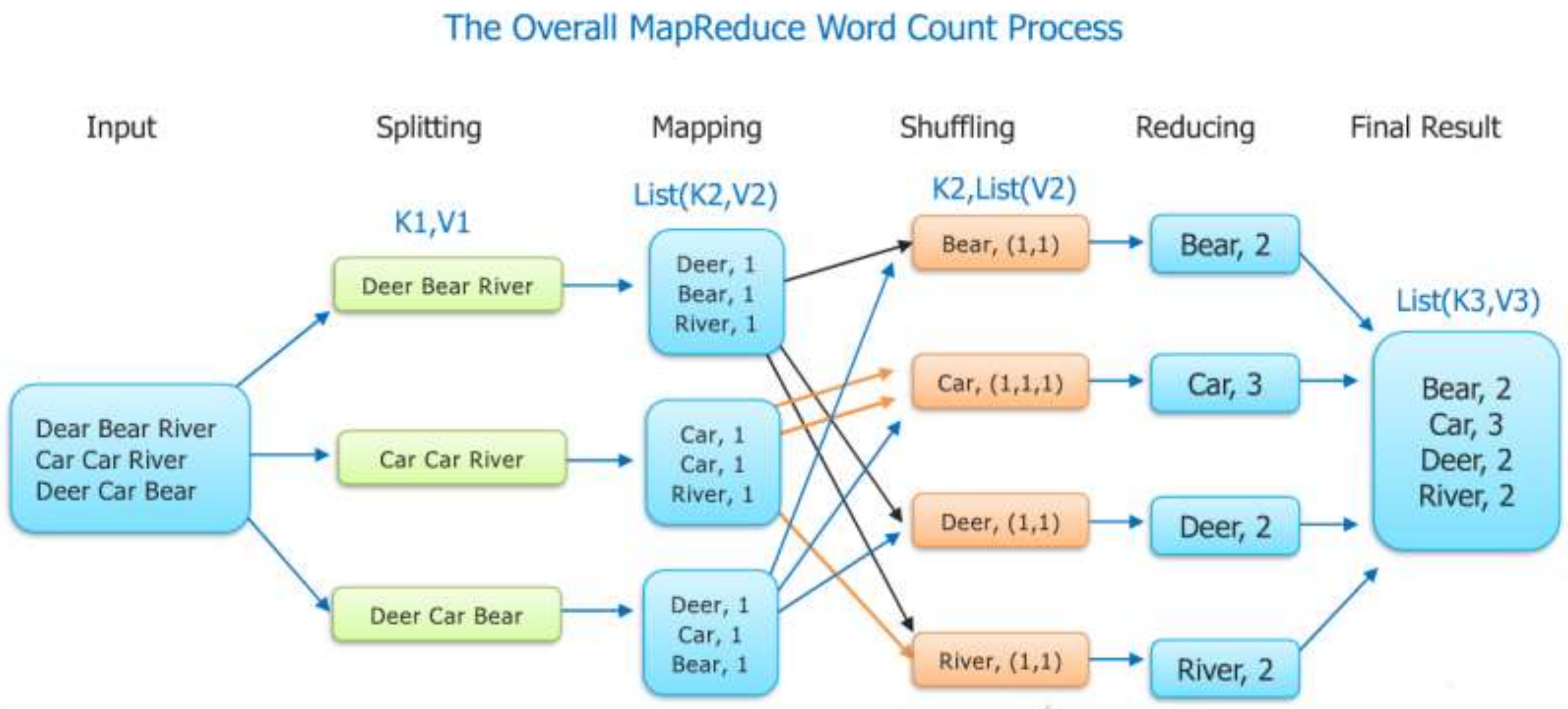



1. Run Job
Application
Job Object
Client JVM
Client
2. Get New Application
4. Submit Application
Management Node
Resource Manager
3. Copy Job Resources
HDFS

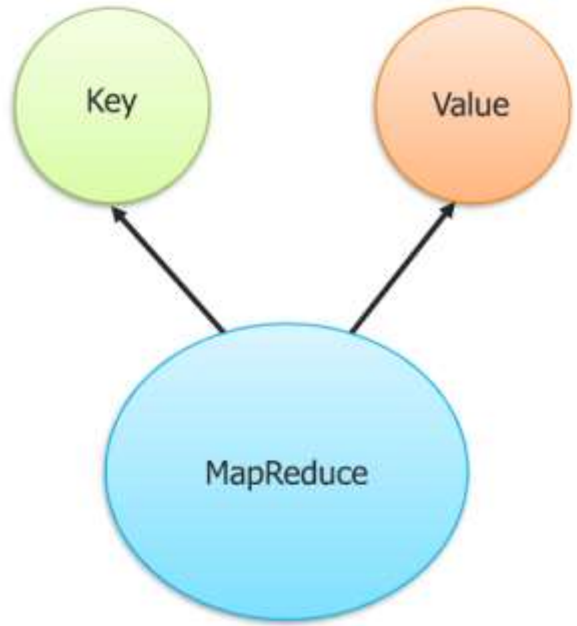

1. Run Job
Application
Job Object
Client JVM
Client
2. Get New Application
4. Submit Application
5. Start MR AppMaster container
Management Node
Resource Manager
3. Copy Job Resources
8. Request Resources
Node Manager
6. Create container
9. Start container
HDFS
7. Get Input Splits
MR AppMaster
Data Node

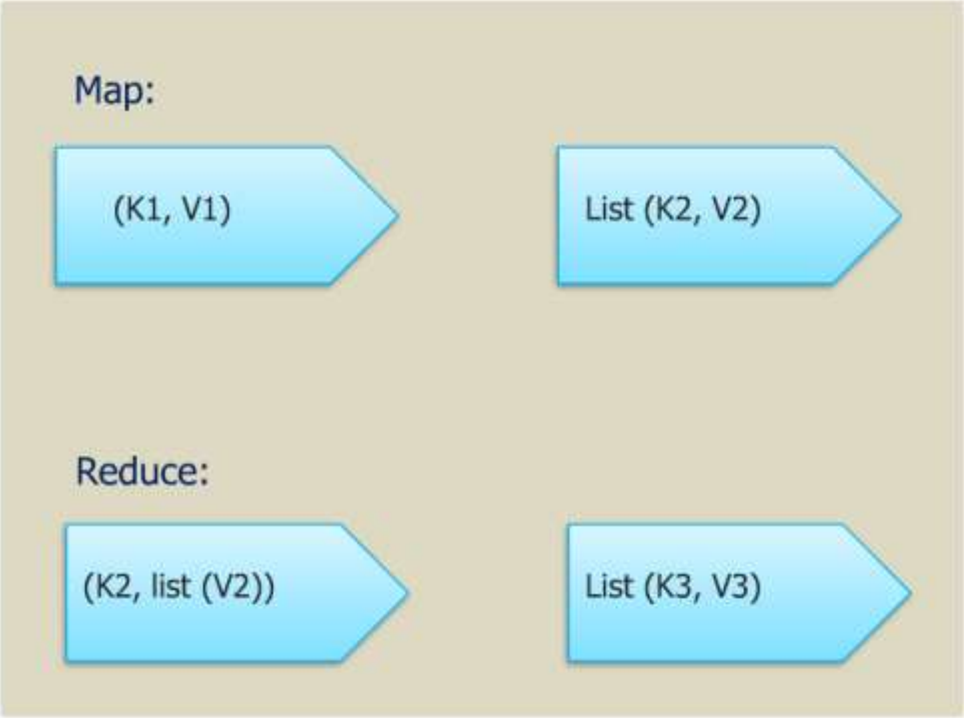

Application
Job Object
Client JVM
Client
Management Node
Resource Manager
Poll for Status
HDFS
Node Manager
Task JVM
YarnChild
Update Status
Map/Reduce Task
MR AppMaster
Data Node

Map reduce paradigm



Anatomy of map reduce program




Map:

(K1, V1)

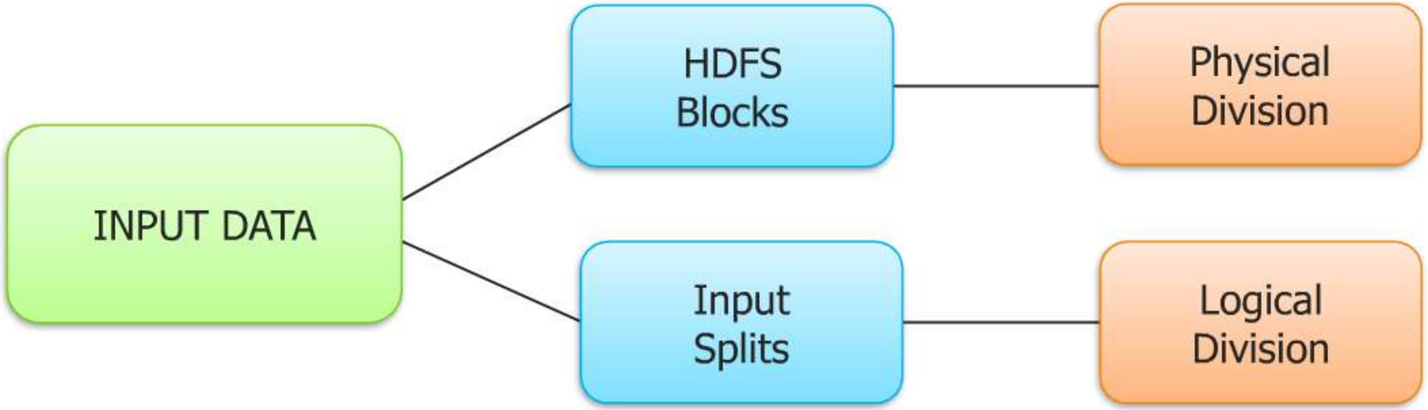
List (K2, V2)

Reduce:

(K2, list (V2))

List (K3, V3)

Input splits



Map reduce job submission flow

Input data is distributed to nodes

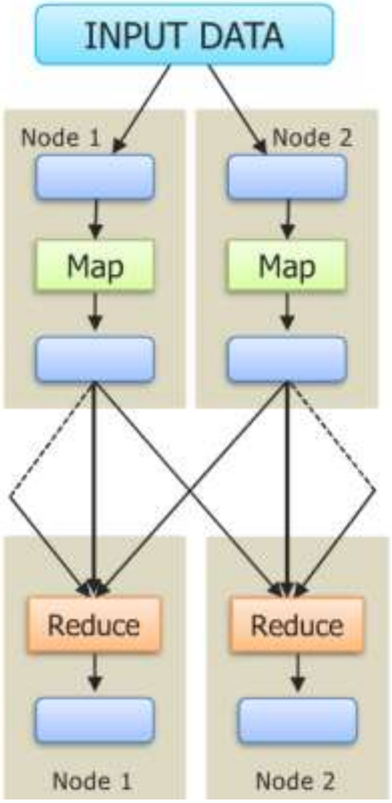
Each map task works on a "split" of data

Mapper outputs intermediate data

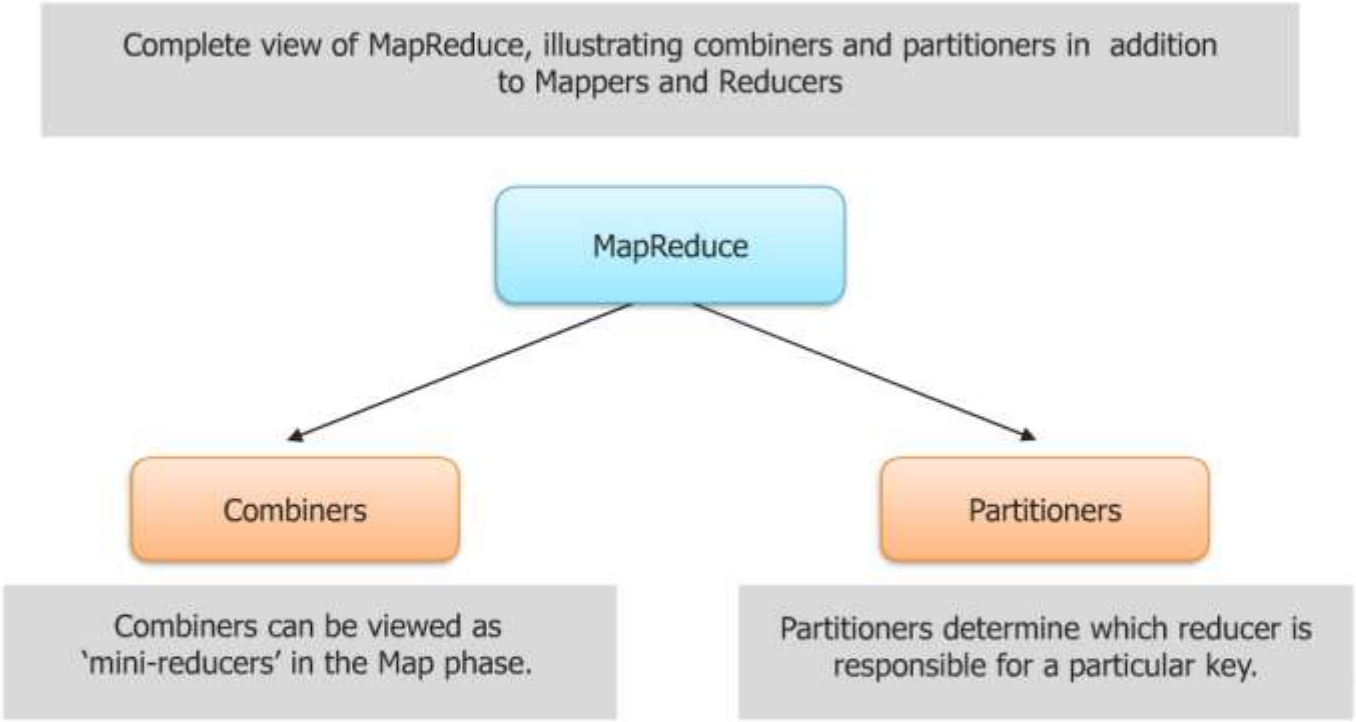
Data exchange between nodes in a "shuffle" process

Intermediate data of the same key goes to the same reducer

Reducer output is stored



Overview of map reduce



Complete view of MapReduce, illustrating combiners and partitioners in addition to Mappers and Reducers

MapReduce

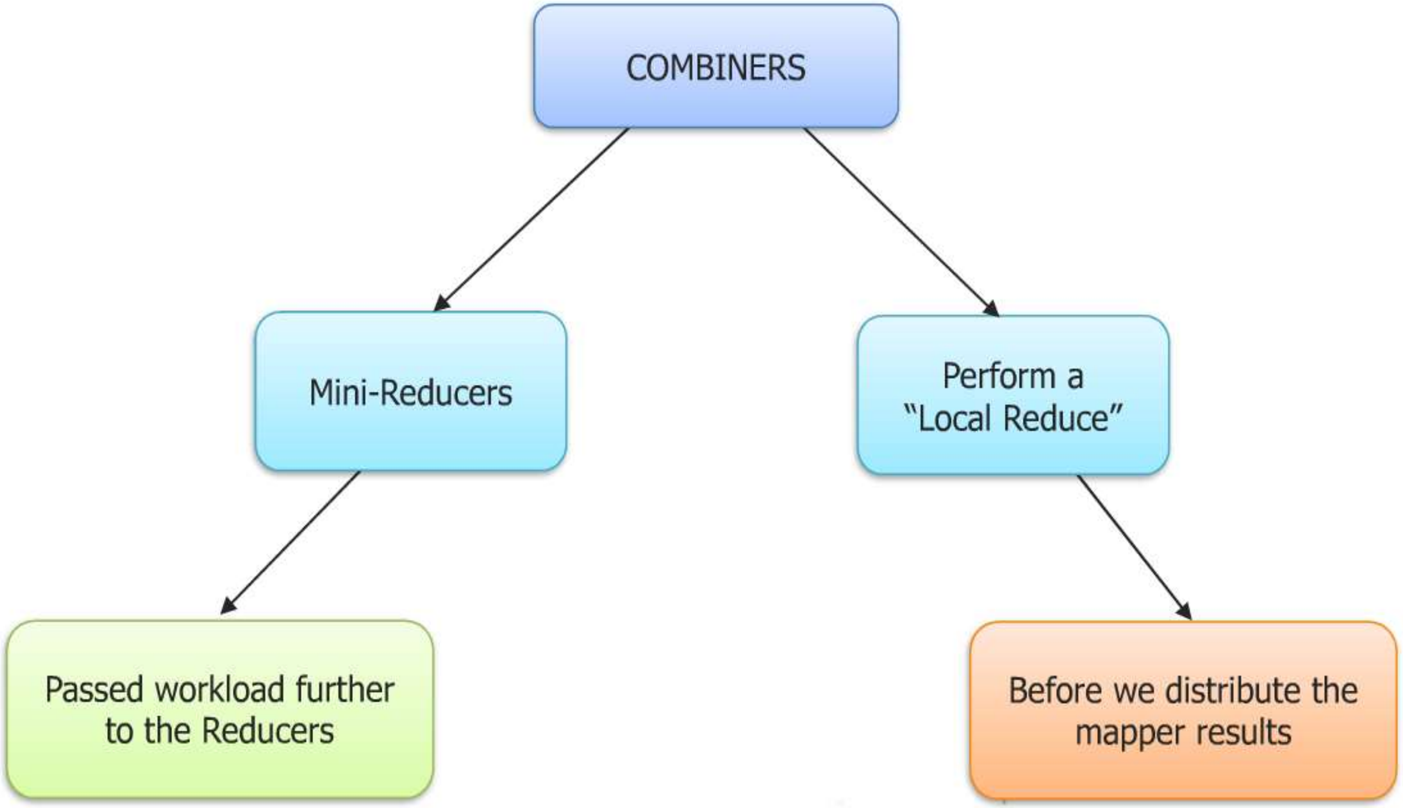
Combiners

Partitioners

Combiners can be viewed as 'mini-reducers' in the Map phase.

Partitioners determine which reducer is responsible for a particular key.

Combiner:Local reduce



COMBINERS

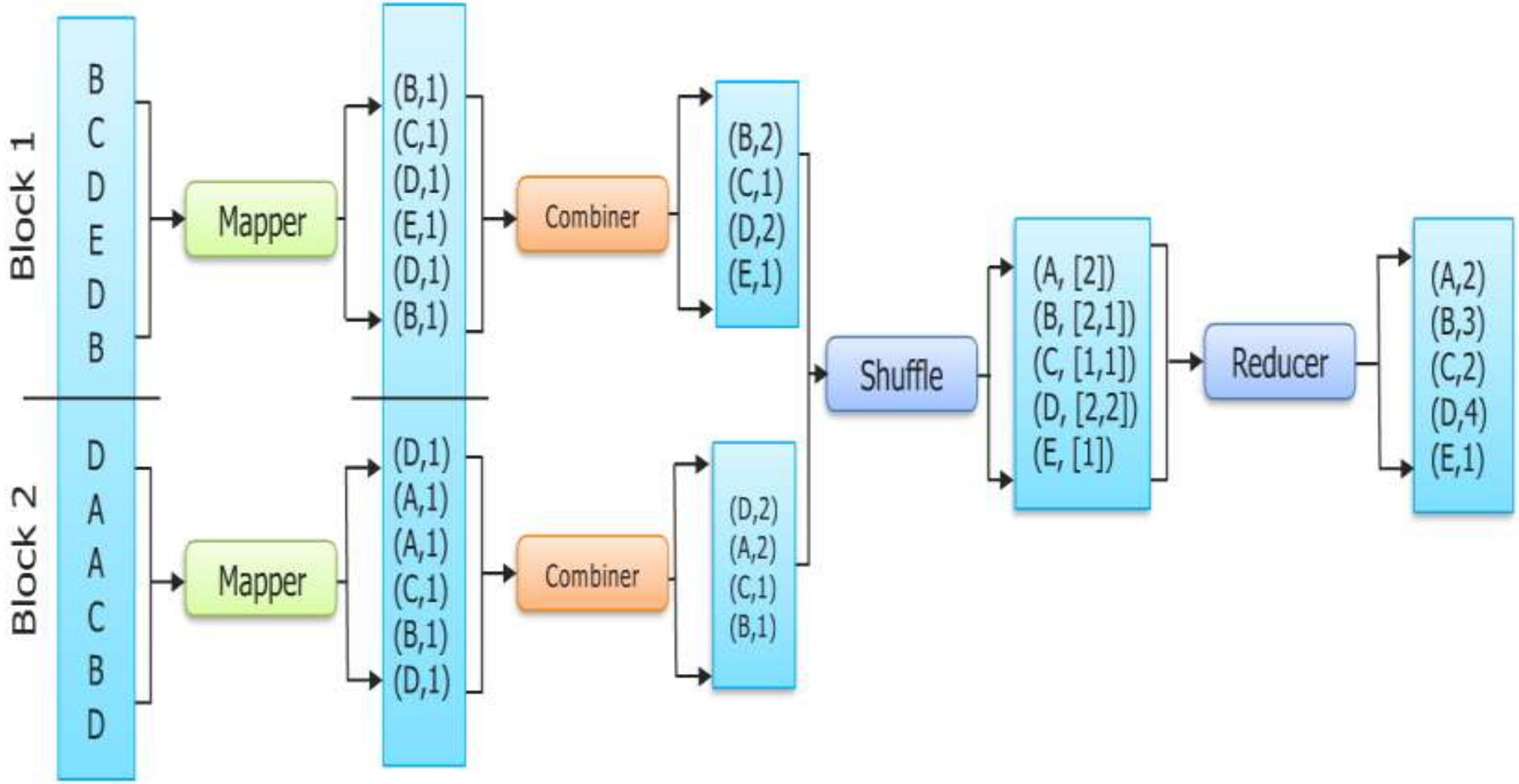
Mini-Reducers

Perform a "Local Reduce"

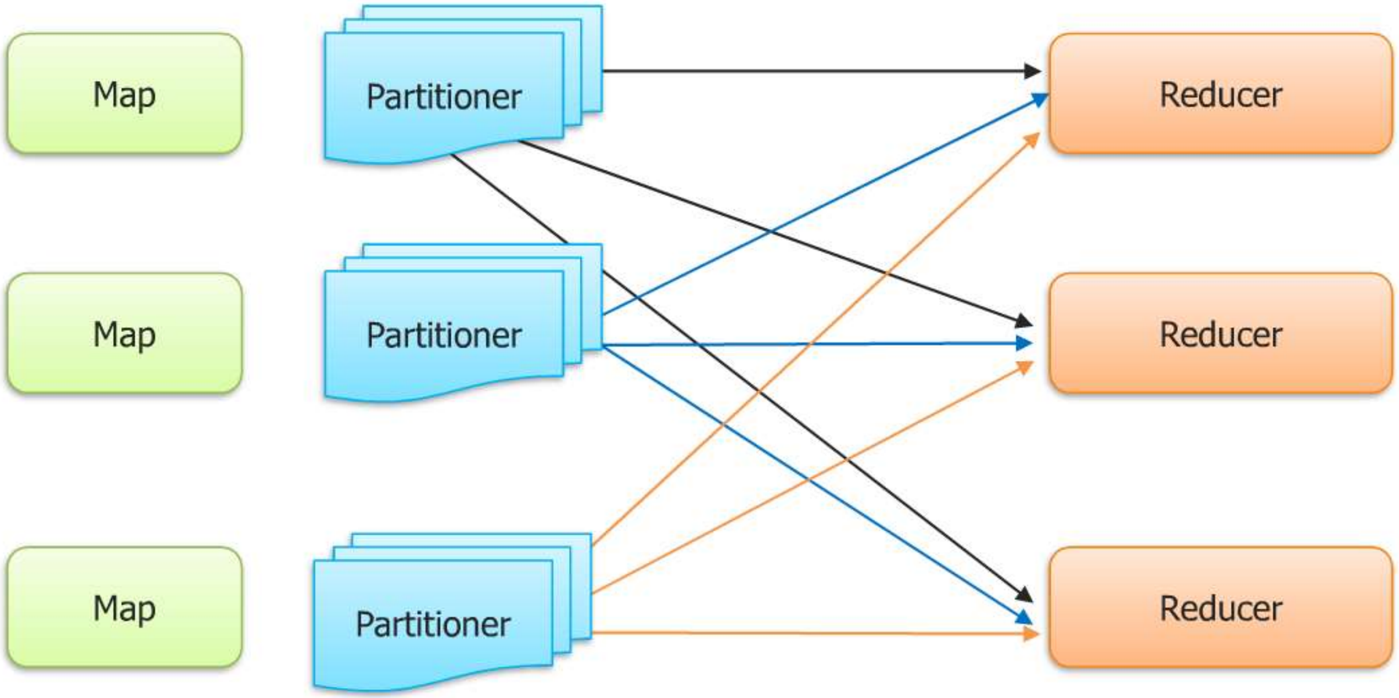
Passed workload further to the Reducers

Before we distribute the mapper results

Combiner


Block 1
B
C
D
E
D
B
Block 2
D
A
A
C
B
D
Mapper
Mapper
(B,1)
(C,1)
(D,1)
(E,1)
(D,1)
(B,1)
(D,1)
(A,1)
(A,1)
(C,1)
(B,1)
(D,1)
Combiner
Combiner
(B,2)
(C,1)
(D,2)
(E,1)
(D,2)
(A,2)
(C,1)
(B,1)
Shuffle
(A, [2])
(B, [2,1])
(C, [1,1])
(D, [2,2])
(E, [1])
Reducer
(A,2)
(B,3)
(C,2)
(D,4)
(E,1)

Parttioning:redirecting output from Mapper



2. Why should I go for Pig when there is MR?

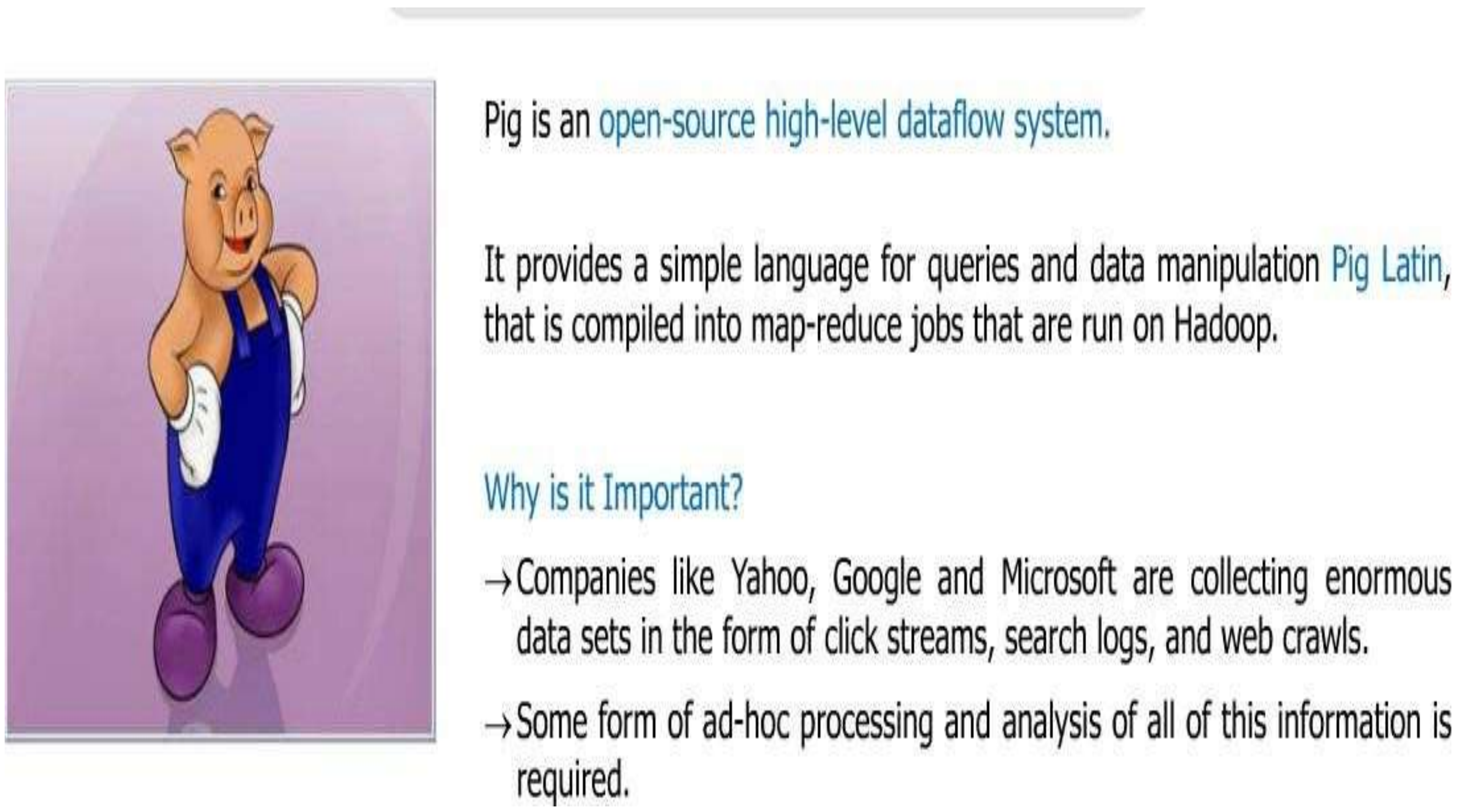
→ Map-Reduce

- » Powerful model for parallelism.
- » Based on a rigid procedural structure.
- » Provides a good opportunity to parallelize algorithm.

→PIG

- » It is desirable to have a higher level declarative language.
- » Similar to SQL query where the user specifies the "what" and leaves the "how" to the underlying processing engine.

What is Pig?



Pig is an open-source high-level dataflow system.

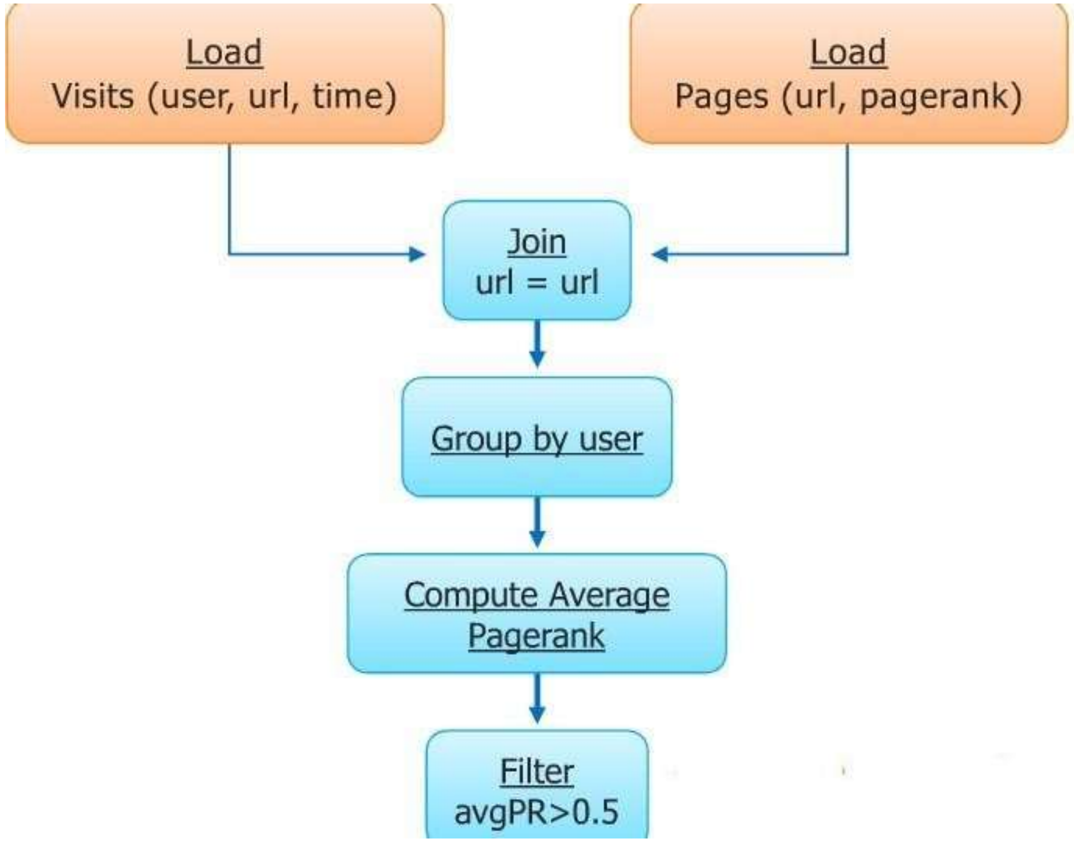
It provides a simple language for queries and data manipulation Pig Latin, that is compiled into map-reduce jobs that are run on Hadoop.

Why is it Important?

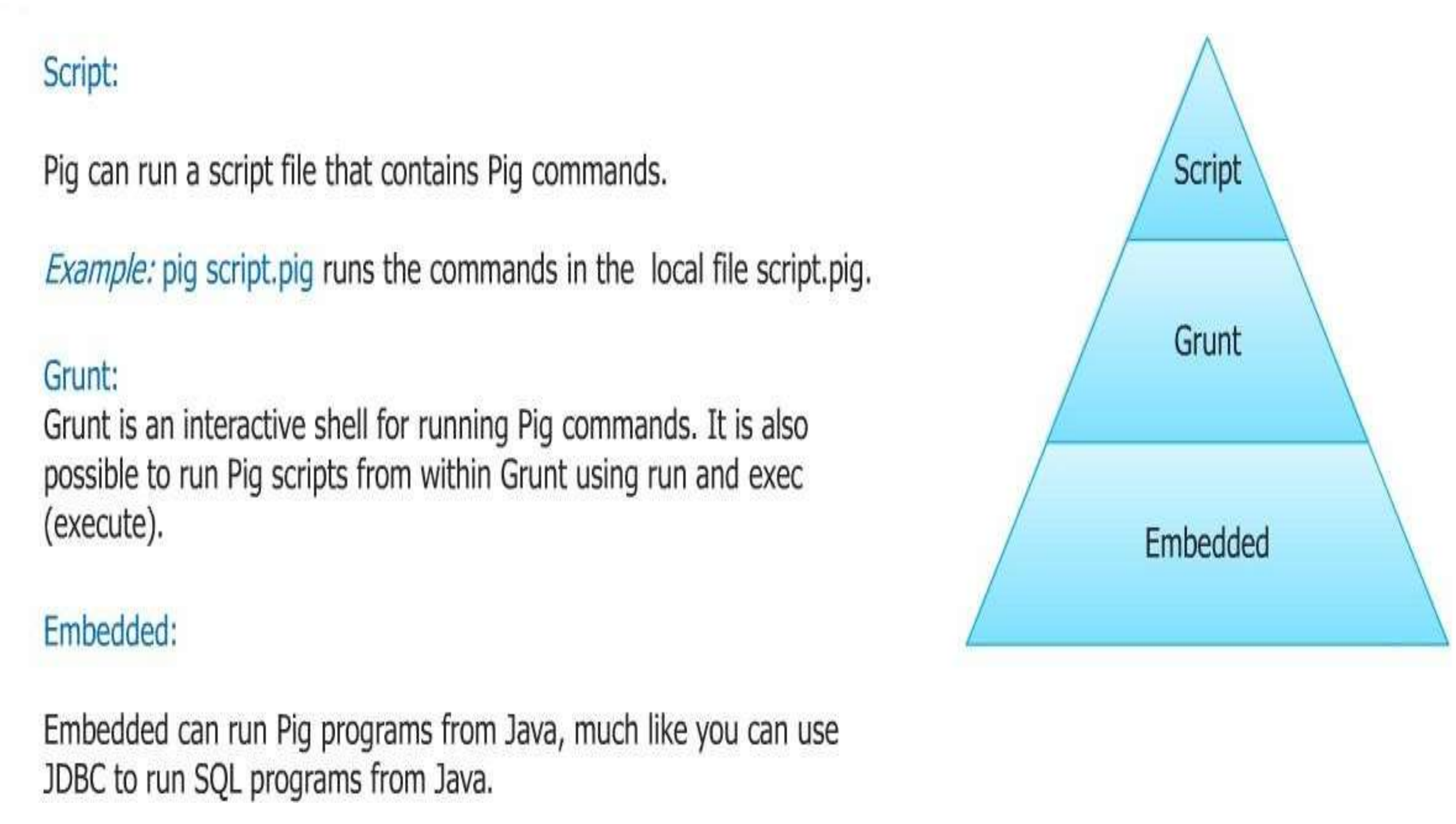
→Companies like Yahoo, Google and Microsoft are collecting enormous data sets in the form of click streams, search logs, and web crawls.

→Some form of ad-hoc processing and analysis of all of this information is required.

Conceptual data flow



Pig basic program structure



Script:

Pig can run a script file that contains Pig commands.

Example: pig script.pig runs the commands in the local file script.pig.

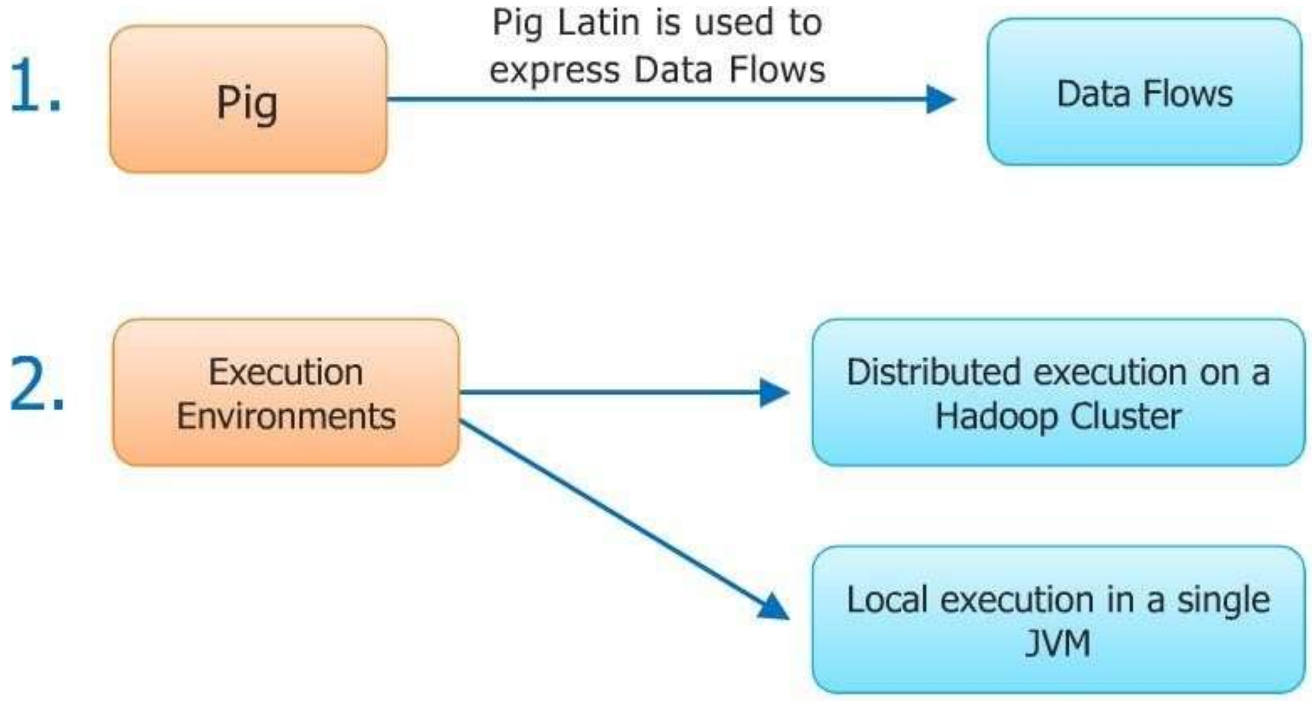
Grunt:

Grunt is an interactive shell for running Pig commands. It is also possible to run Pig scripts from within Grunt using run and exec (execute).

Embedded:

Embedded can run Pig programs from Java, much like you can use JDBC to run SQL programs from Java.

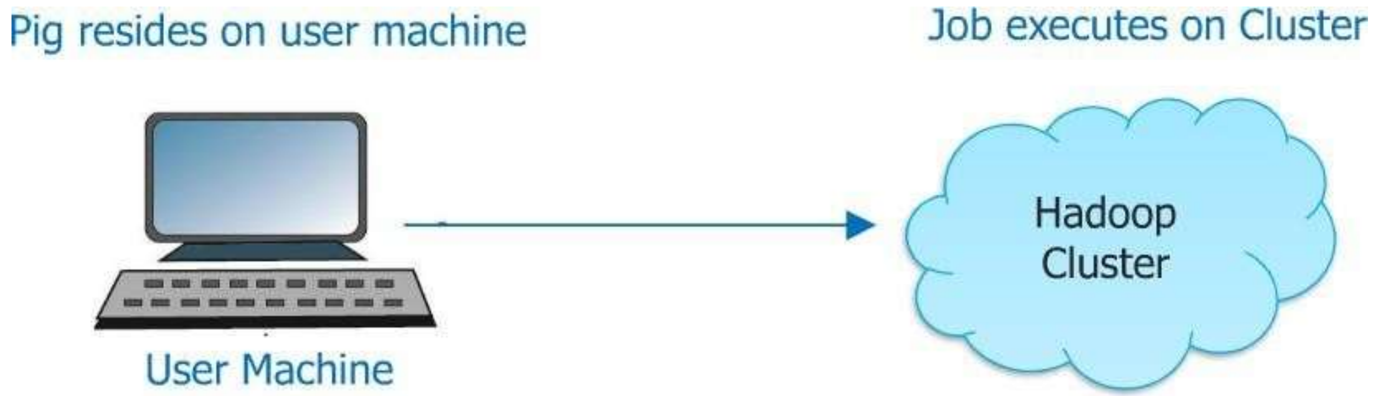
A pig is made of two components



1. Pig → Pig Latin is used to express Data Flows → Data Flows

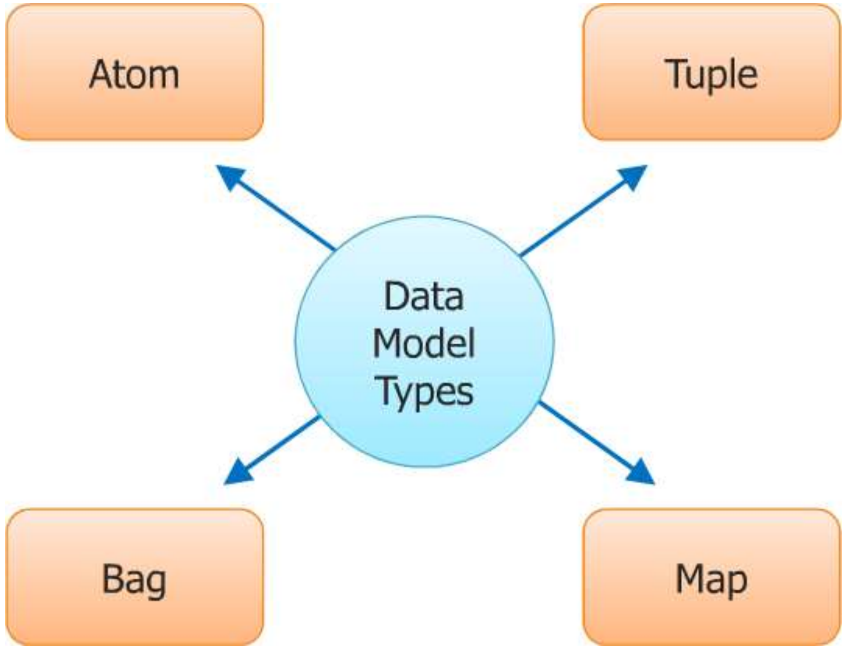
2. Execution Environments → Distributed execution on a Hadoop Cluster; Local execution in a single JVM

Pig execution

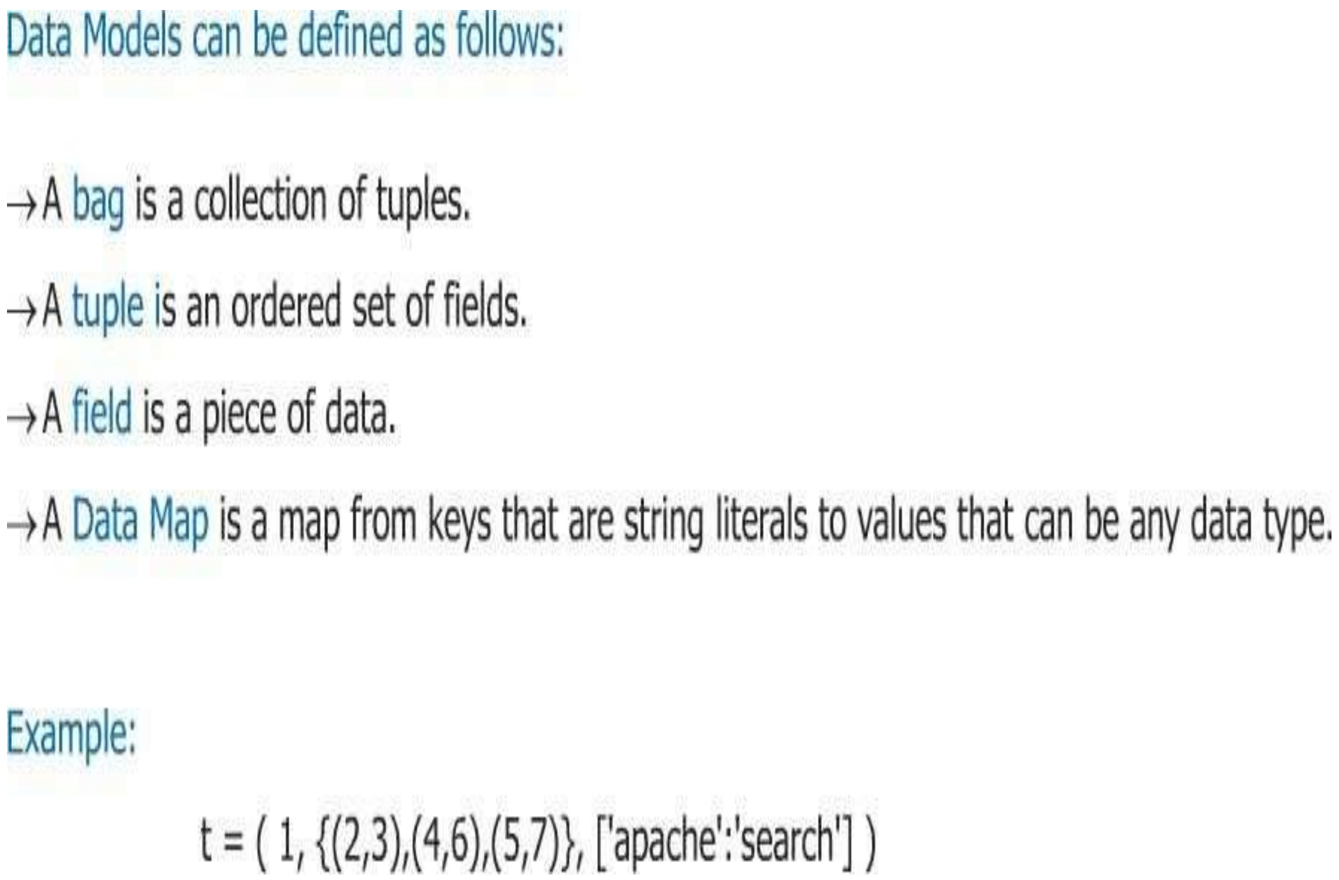


No need to install anything extra on your Hadoop Cluster!

Four basic types of data models



Data models



Data Models can be defined as follows:

→A bag is a collection of tuples.

→A tuple is an ordered set of fields.

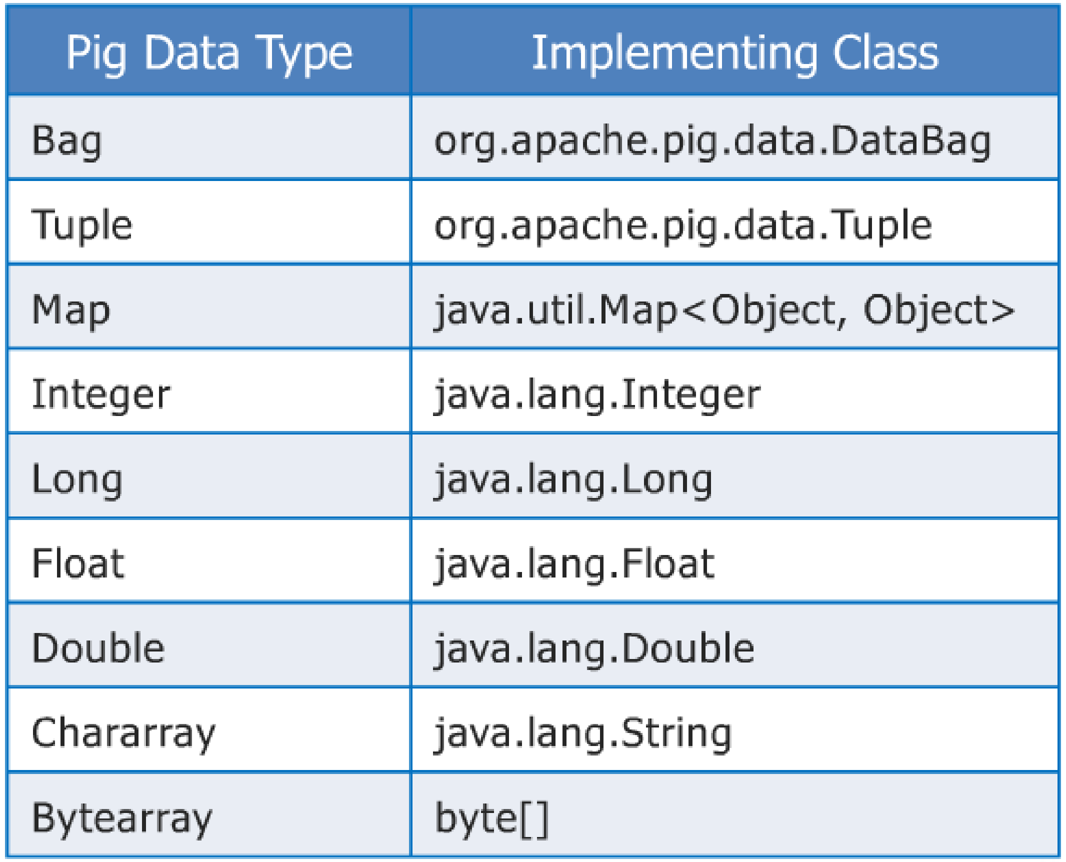
→A field is a piece of data.

→A Data Map is a map from keys that are string literals to values that can be any data type.

Example:

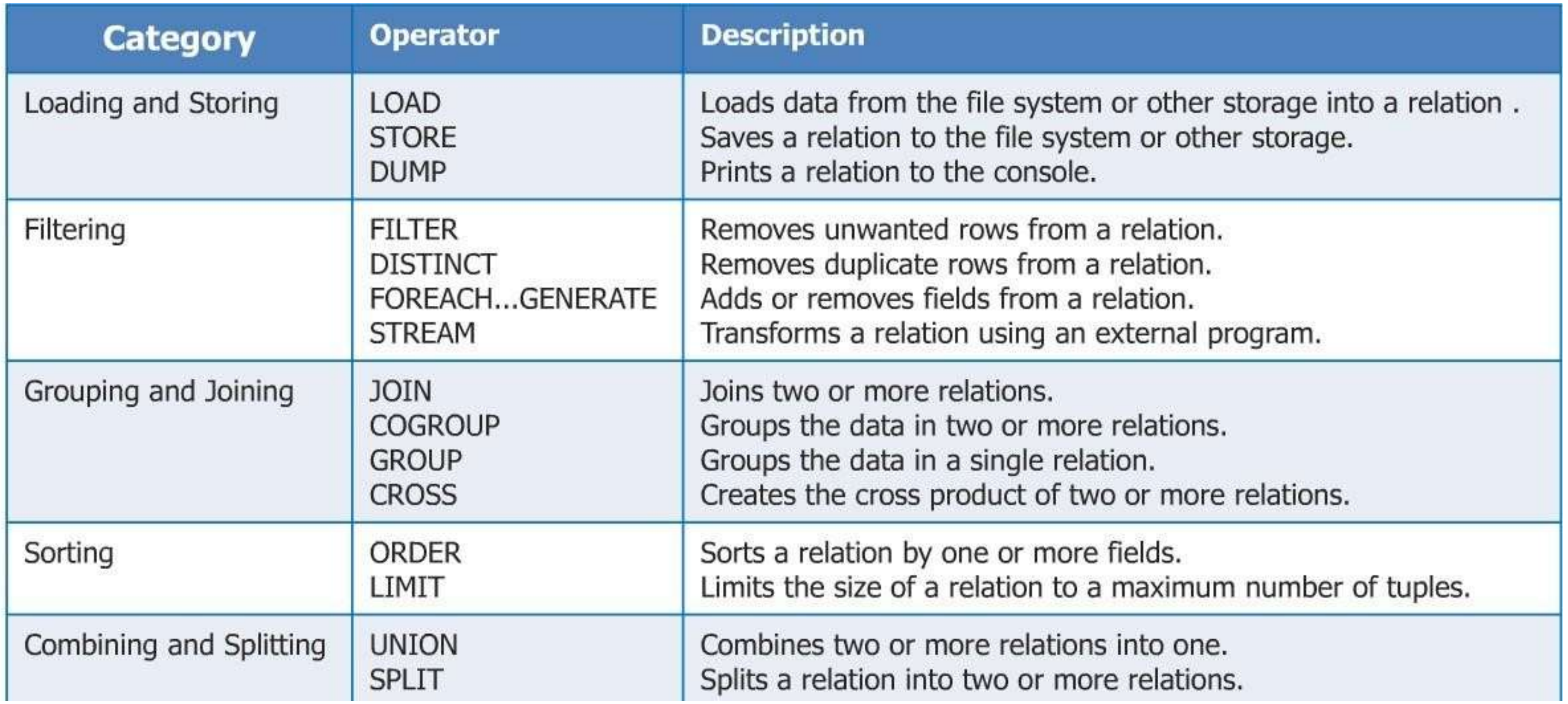
```
t = ( 1, {(2,3),(4,6),(5,7)}, ['apache':'search'] )
```

Pig data types



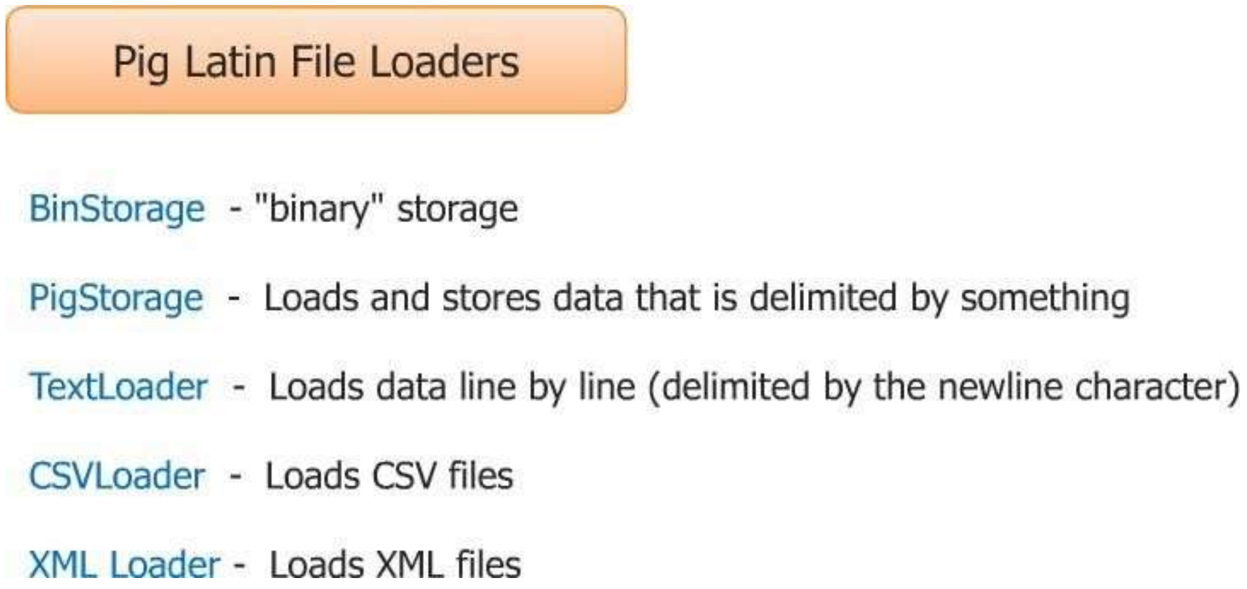
Pig Data Type	Implementing Class
Bag	org.apache.pig.data.DataBag
Tuple	org.apache.pig.data.Tuple
Map	java.util.Map<Object, Object>
Integer	java.lang.Integer
Long	java.lang.Long
Float	java.lang.Float
Double	java.lang.Double
Chararray	java.lang.String
Bytearray	byte[]

Pig Latin-Relational operators



Category	Operator	Description
Loading and Storing	LOAD	Loads data from the file system or other storage into a relation .
	STORE	Saves a relation to the file system or other storage.
	DUMP	Prints a relation to the console.
Filtering	FILTER	Removes unwanted rows from a relation.
	DISTINCT	Removes duplicate rows from a relation.
	FOREACH...GENERATE	Adds or removes fields from a relation.
	STREAM	Transforms a relation using an external program.
Grouping and Joining	JOIN	Joins two or more relations.
	COGROUP	Groups the data in two or more relations.
	GROUP	Groups the data in a single relation.
	CROSS	Creates the cross product of two or more relations.
Sorting	ORDER	Sorts a relation by one or more fields.
	LIMIT	Limits the size of a relation to a maximum number of tuples.
Combining and Splitting	UNION	Combines two or more relations into one.
	SPLIT	Splits a relation into two or more relations.

Pig Latin File Loaders



Pig Latin File Loaders

BinStorage - "binary" storage

PigStorage - Loads and stores data that is delimited by something

TextLoader - Loads data line by line (delimited by the newline character)

CSVLoader - Loads CSV files

XML Loader - Loads XML files

Data

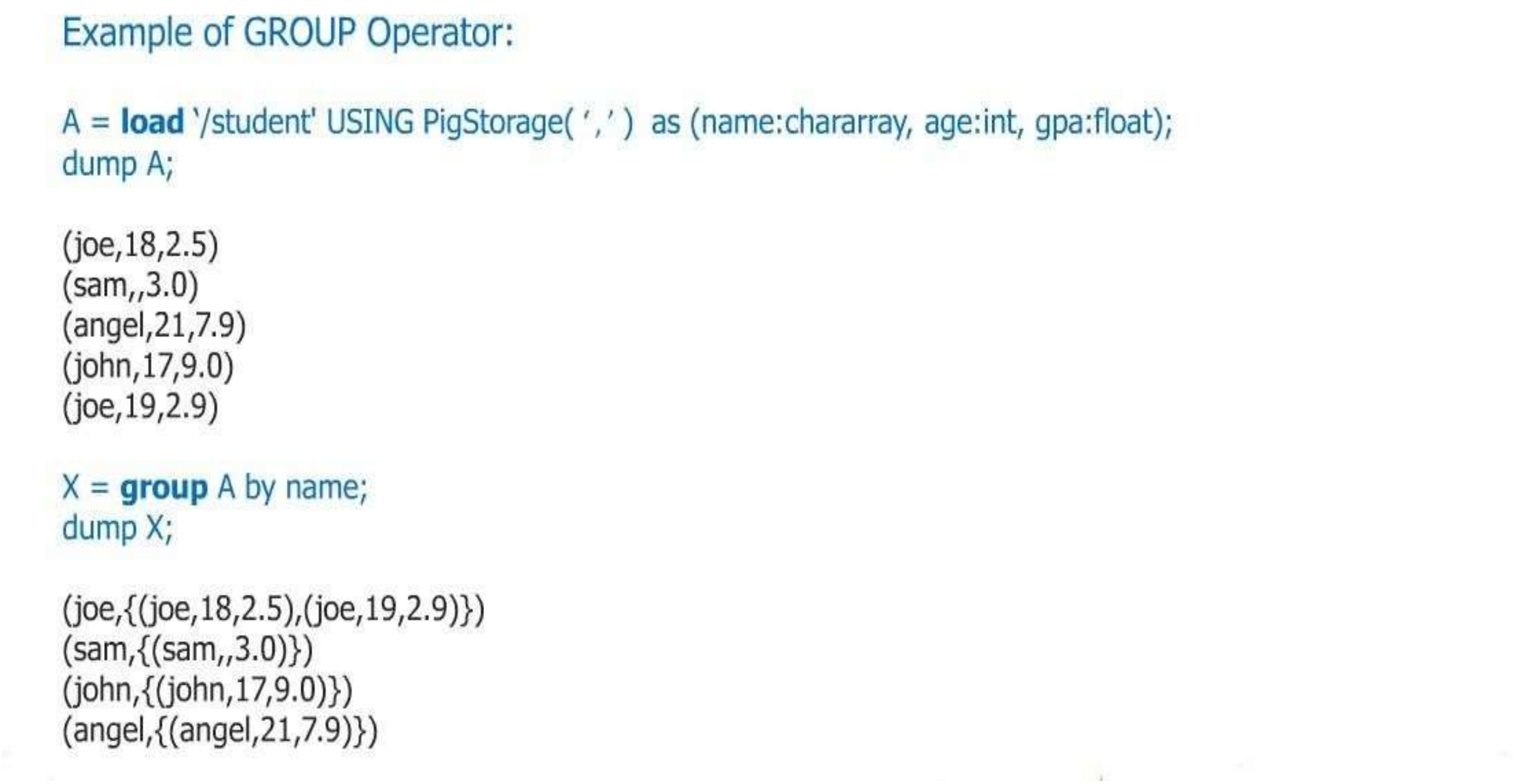
File – Student

Name	Age	GPA
Joe	18	2.5
Sam		3.0
Angel	21	7.9
John	17	9.0
Joe	19	2.9

File – Student Roll

Name	Roll No.
Joe	45
Sam	24
Angel	1
John	12
Joe	19

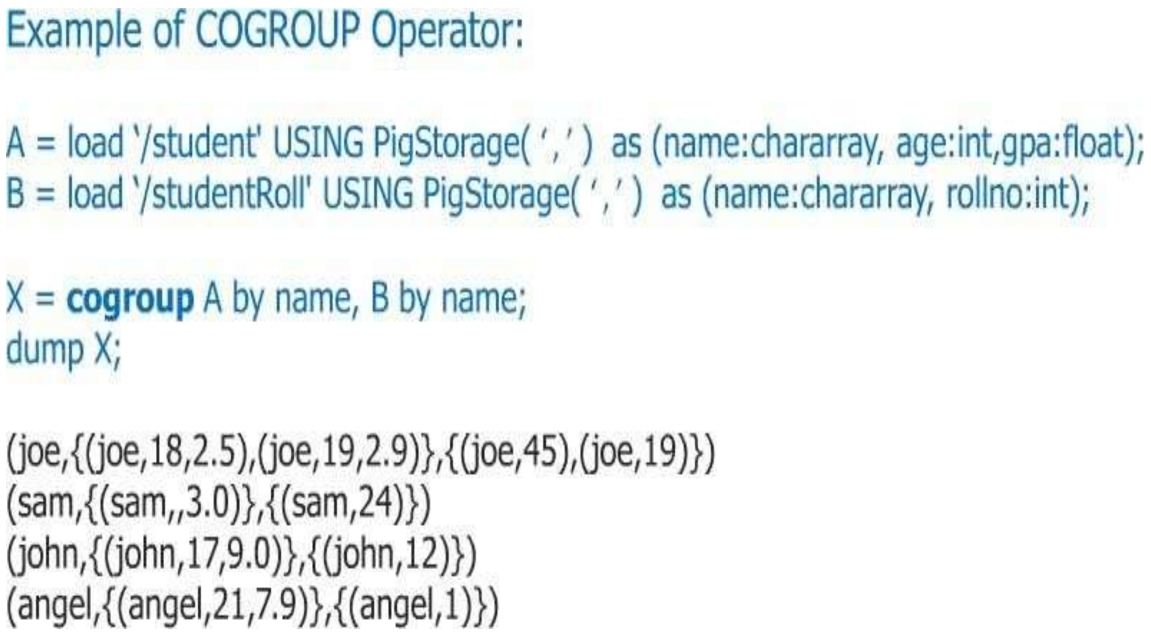
Group operator



Example of GROUP Operator:

```
A = load ‘/student' USING PigStorage( ‘,’ )  as (name:chararray, age:int, gpa:float);
dump A;

(joe,18,2.5)
(sam,,3.0)
(angel,21,7.9)
(john,17,9.0)
(joe,19,2.9)

X = group A by name;
dump X;

(joe,{(joe,18,2.5),(joe,19,2.9)})
(sam,{(sam,,3.0)})
(john,{(john,17,9.0)})
(angel,{(angel,21,7.9)})
```

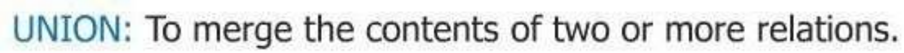
Cogroup operator



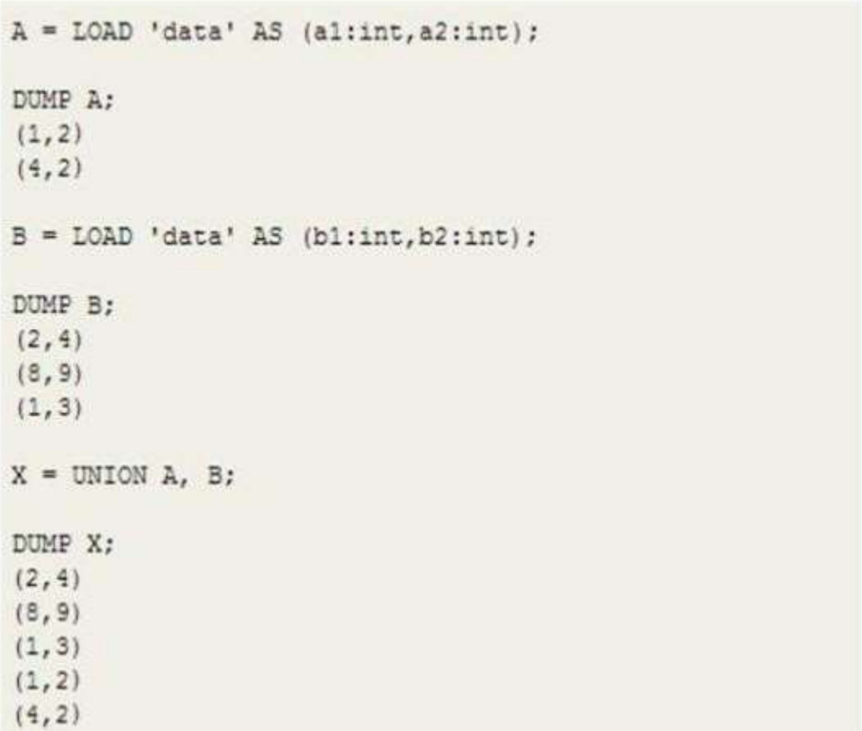
Example of COGROUP Operator:

```
A = load '/student' USING PigStorage( ',' )  as (name:chararray, age:int,gpa:float);
B = load '/studentRoll' USING PigStorage( ',' )  as (name:chararray, rollno:int);

X = cogroup A by name, B by name;
dump X;

(joe,{(joe,18,2.5),(joe,19,2.9)},{(joe,45),(joe,19)})
(sam,{(sam,,3.0)},{(sam,24)})
(john,{(john,17,9.0)},{(john,12)})
(angel,{(angel,21,7.9)},{(angel,1)})
```

Union



UNION: To merge the contents of two or more relations.



```
A = LOAD 'data' AS (a1:int,a2:int);

DUMP A;
(1,2)
(4,2)

B = LOAD 'data' AS (b1:int,b2:int);

DUMP B;
(2,4)
(8,9)
(1,3)

X = UNION A, B;

DUMP X;
(2,4)
(8,9)
(1,3)
(1,2)
(4,2)
```

3. Real Time Analytics

BZ-CBSSBI

Your AC XXXXX12345 Debited INR 12,345.67 on 28/06/13 - TFR. Avl Bal INR 9,876.54

Banking

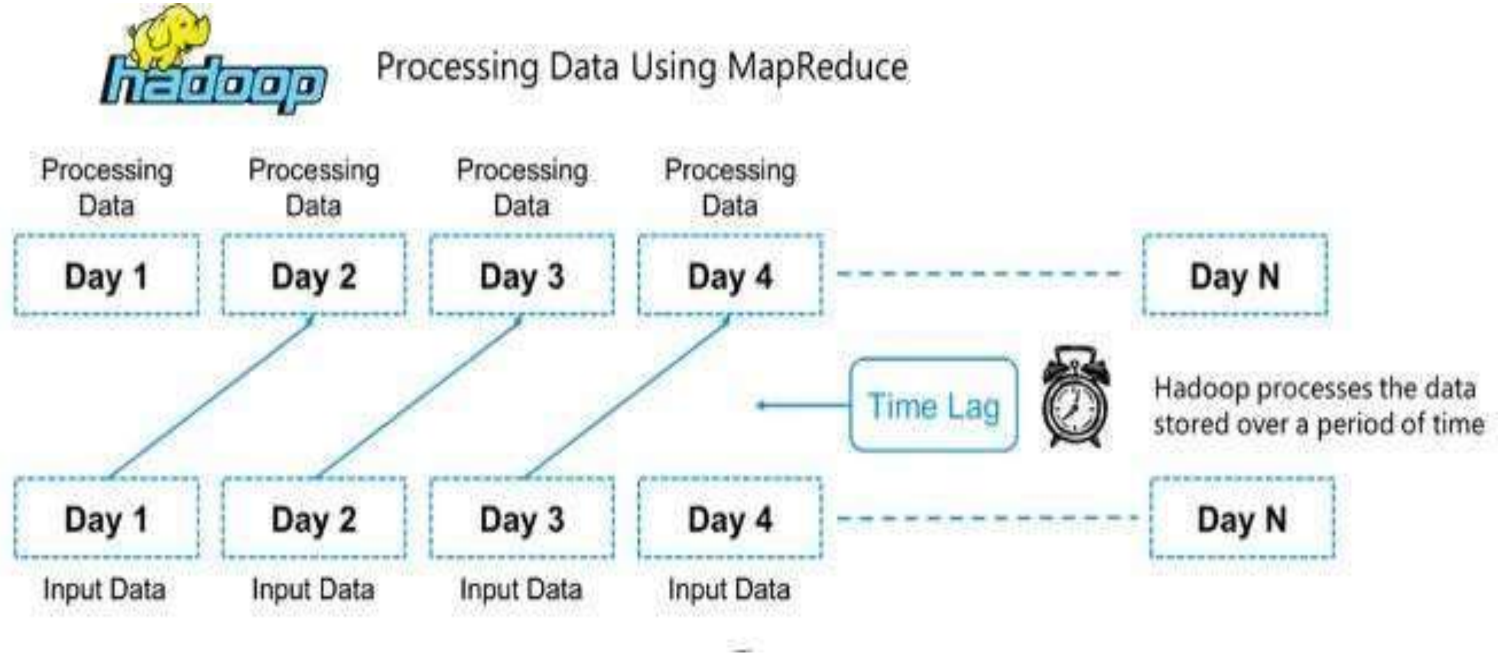
Government

Healthcare

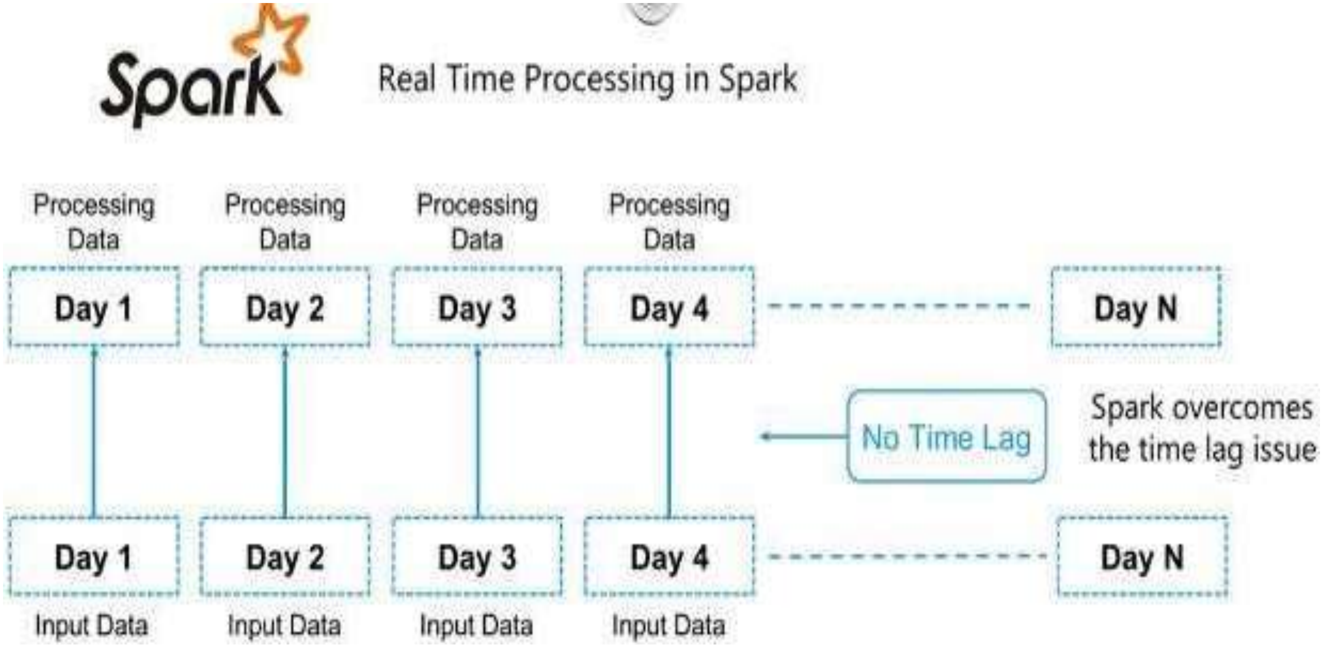
Telecommunications

Stock Market

Why Spark when Hadoop is already there?



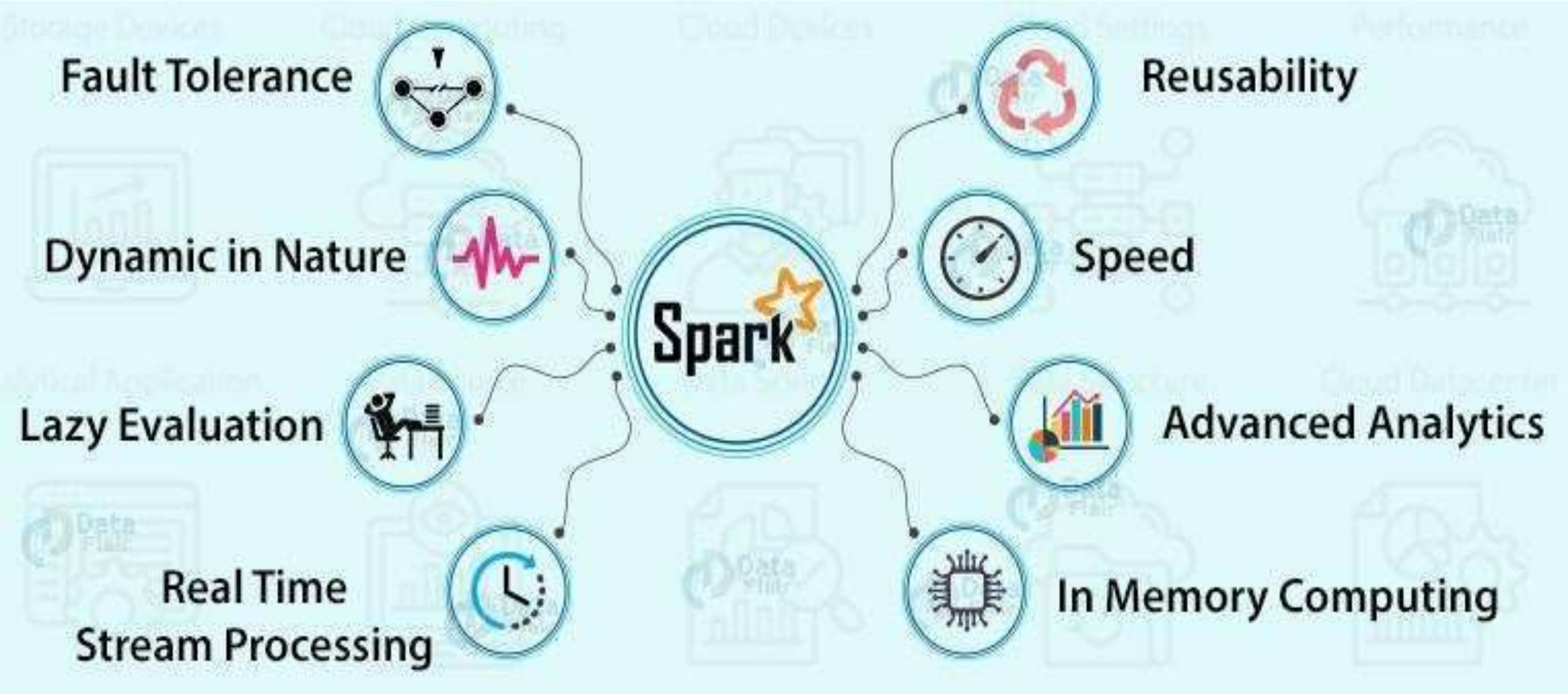
Why Spark when Hadoop is already there?



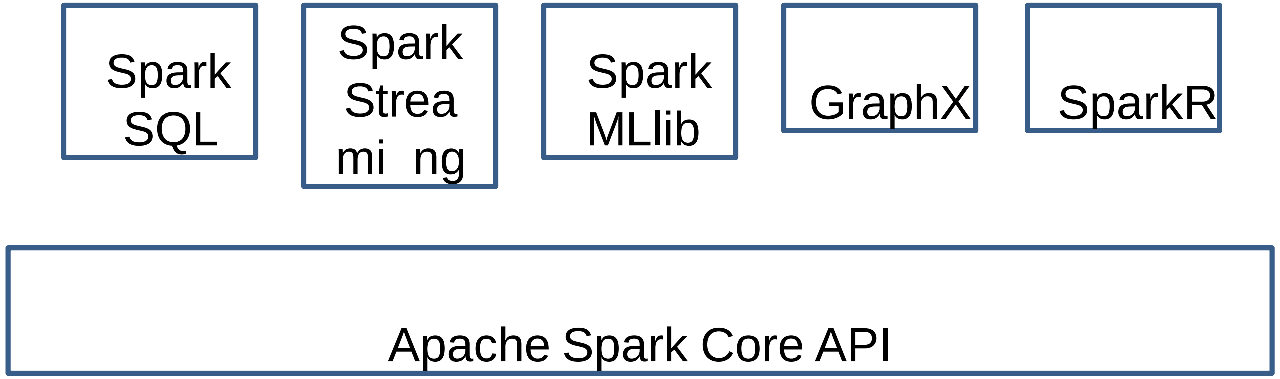
What is Spark?

- Apache Spark is an open source cluster computing framework for real-time data processing.
- Main feature of Apache Spark : in-memory cluster computing that increases the
  processing speed of an application.
- Spark provides an interface for programming entire clusters with implicit data
  parallelism and fault tolerance.

Features of Apache Spark



Spark Components



Spark SQL

Spark Streaming

Spark MLlib

GraphX

SparkR

Apache Spark Core API

Spark Deployment Modes

- Standalone (used for learning & development)
- Local mode (used for learning & development)
- Cluster mode (can work with MESOS or YARN)

Resilient Distributed Dataset(RDD)

- RDDs are the building blocks of any Spark application. RDDs Stands for:

- **Resilient:** Fault tolerant and is capable of rebuilding data on failure
- **Distributed**: Distributed data among the multiple nodes in a cluster
- **Dataset:** Collection of partitioned data with values

Thank you