BIG DATA - HADOOP

Outline

- Introduction to BigData / Hadoop
- HDFS Overview
- MapReduce Framework
- YARN Overview

Introduction to Big Data / Hadoop

- What is BigData?
- 3Vs – Volume, Velocity, Varity
- Big Data in Industry
- Hadoop Overview
- Hadoop Characteristics and Benefits
  - Accessible, Robust, Scalable, Simple
  - Economical, Reliable, Flexible
- Hadoop Architecture
- Hadoop Ecosystem
- Hadoop 1.x vs Hadoop 2.x vs Hadoop 3.x

What is Big Data?

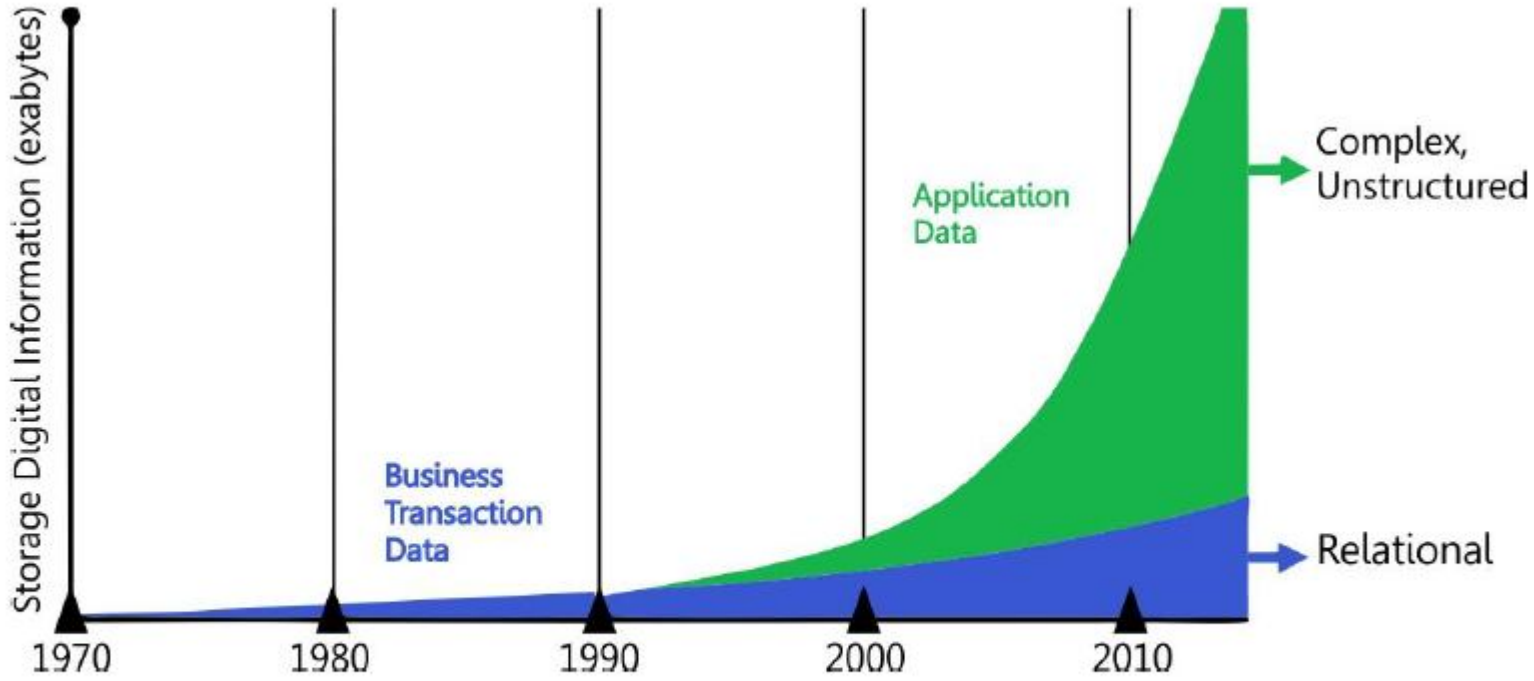
- Huge Amount of Data (Terabytes or Petabytes)
- Big data is the term for a collection of data sets so **large and complex** that it becomes **difficult** to process using on-hand database management tools or traditional data processing applications
- The challenges include capture, curation, storage, search, sharing, transfer, analysis, and visualization

Sources of Big Data Generation

- Systems / Enterprises generate huge amount of data from Terabytes to Petabytes of information

NYSE generates about one terabyte of new trade data per day to perform stock trading analytics to determine trends for optimal trades

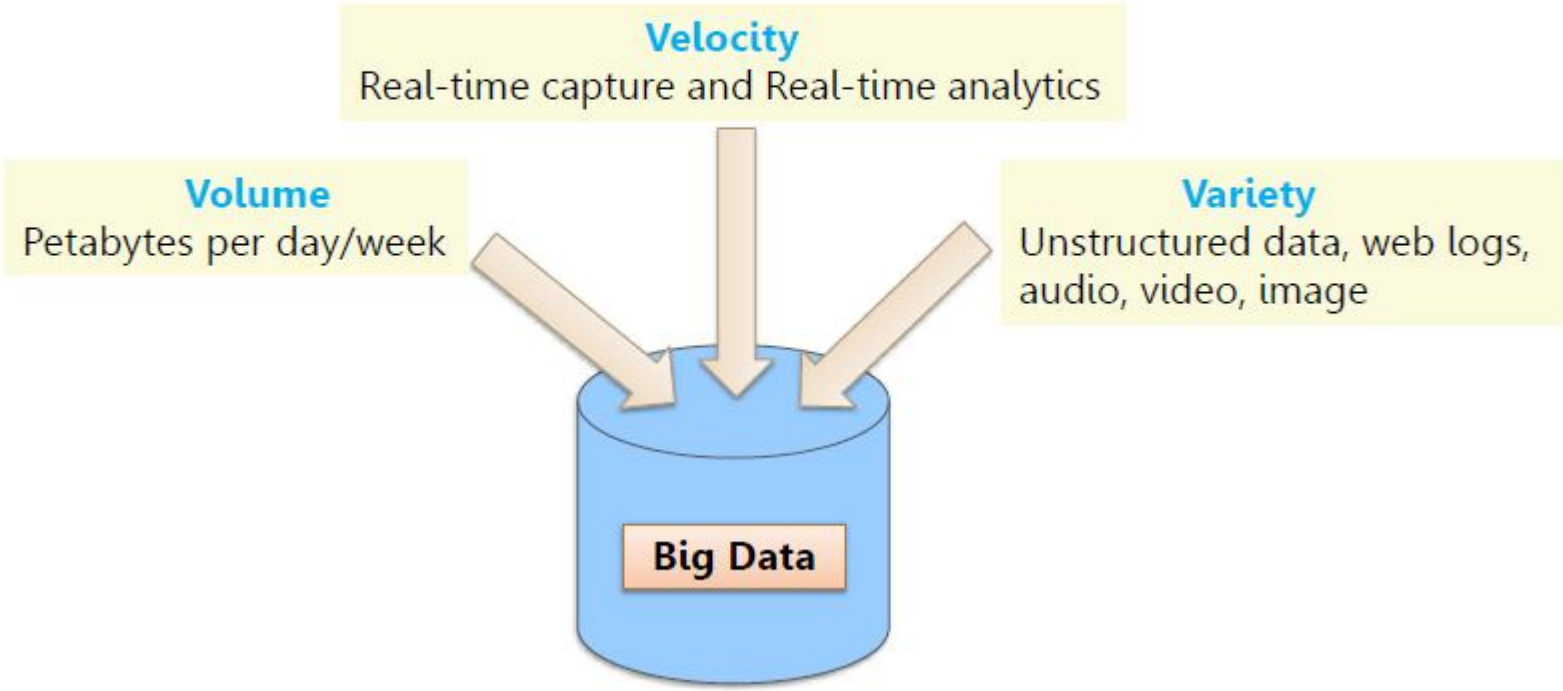
Challenges of Big Data World



- 2,500 exabytes of new information in 2012 with interest as primary driver
- Digital universe grew by 62% last year to 800K petabytes and will go to 1.2 "Zettabytes" this year

IBM’s Definition of Big Data

- IBM’s Definition – Big Data Characteristics
- http://www-01.ibm.com/software/data/bigdata/



Big Data in Industry

Web and e-tailing

- Recommendation Engines
- Ad Targeting
- Search Quality
- Abuse and Click Fraud Detection

Telecommunications

- Customer Churn Prevention
- Network Performance Optimization
- Calling Data Record (CDR) Analysis
- Analyzing Network to Predict Failure

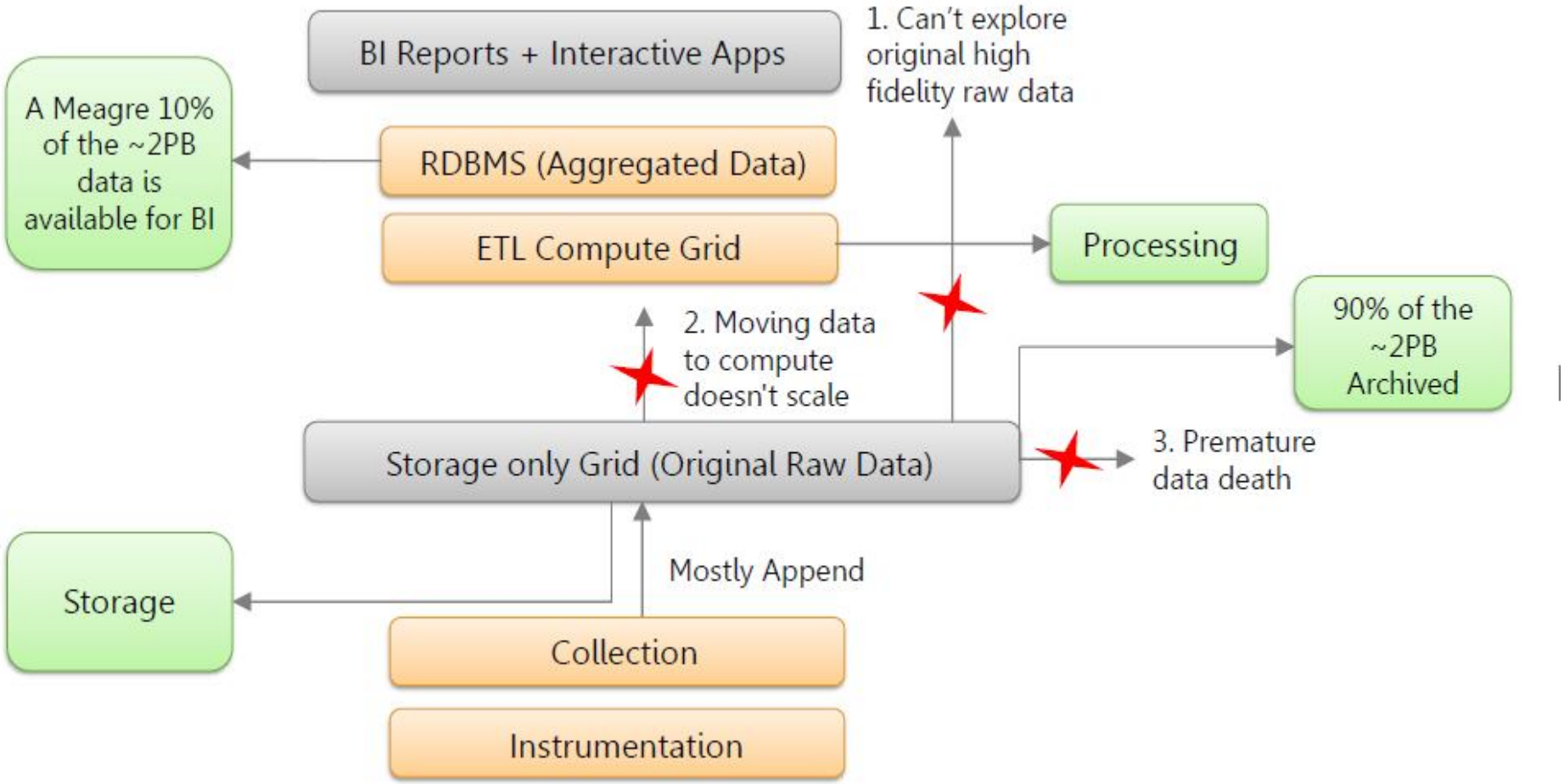
Government

- Fraud Detection and Cyber Security
- Welfare Schemes
- Justice

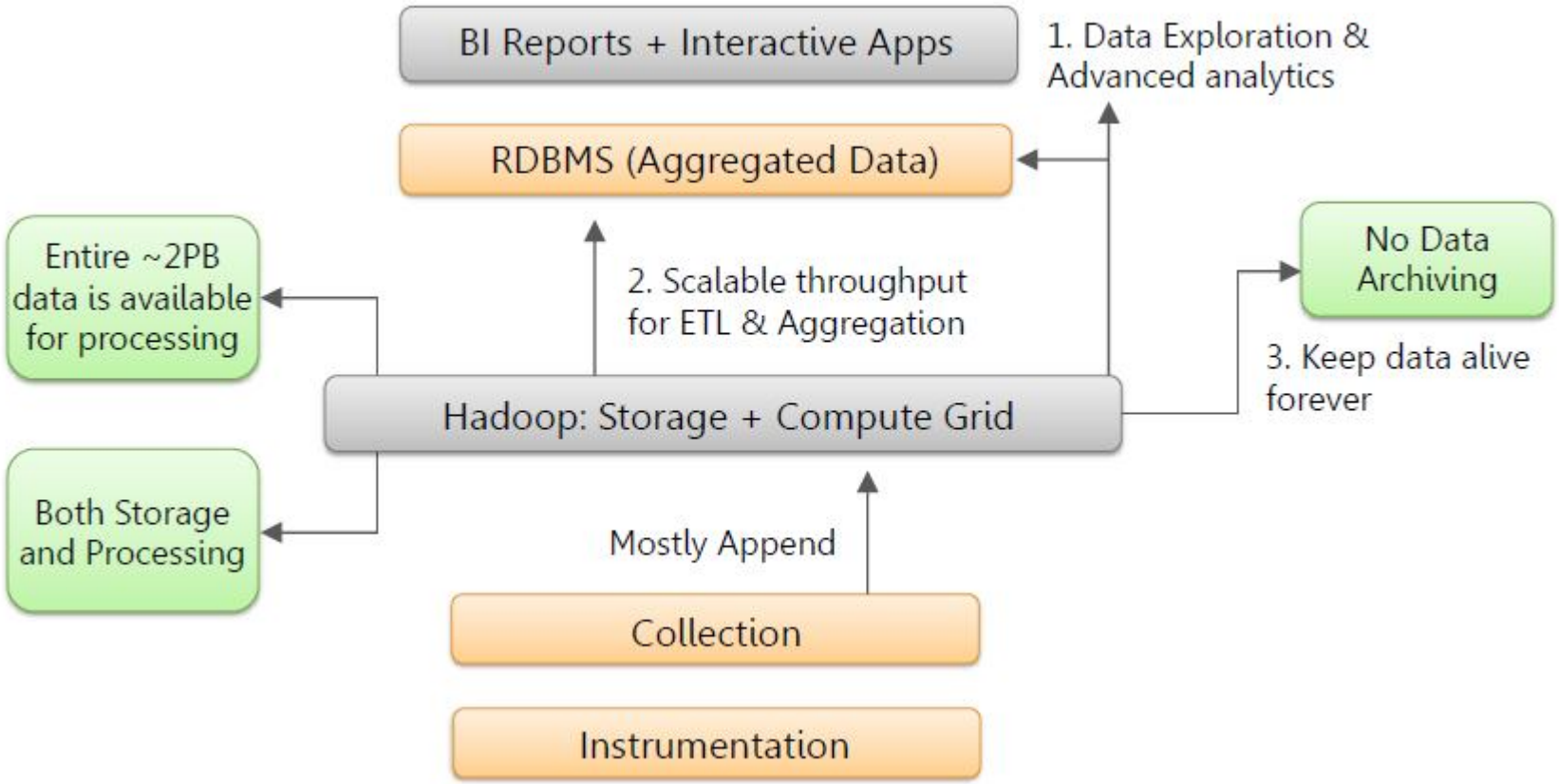
Healthcare and Life Sciences

- Health Information Exchange
- Gene Sequencing
- Serialization
- Healthcare Service Quality Improvements
- Drug Safety

Limitations of Traditional BI Architecture



Solution = Storage + Processing Layer

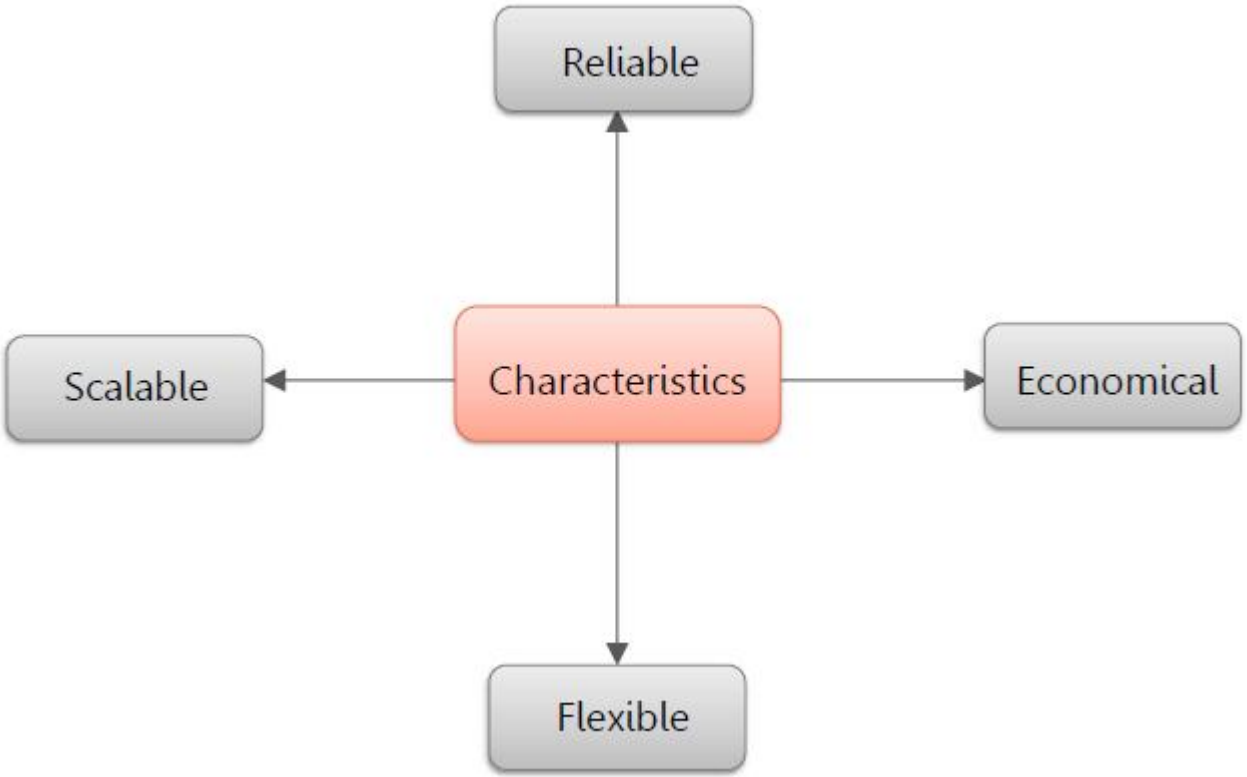


What is Hadoop?

Apache Hadoop is a framework that allows 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

Hadoop Key Characteristics



Hadoop Core Components

Hadoop is a system for large scale data processing. It has two main components:

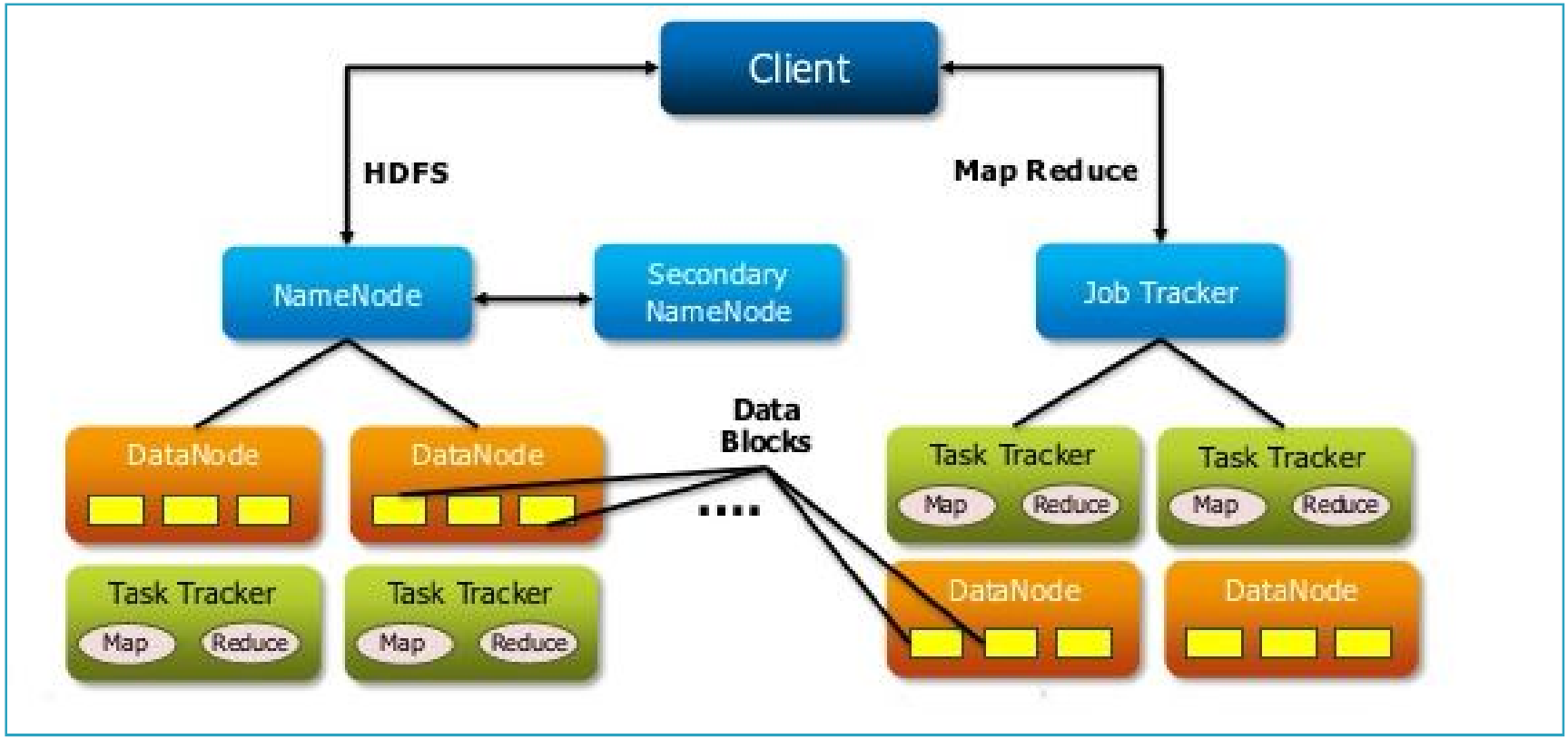
HDFS

- Distributed across "nodes"
- Natively redundant
- NameNode tracks locations

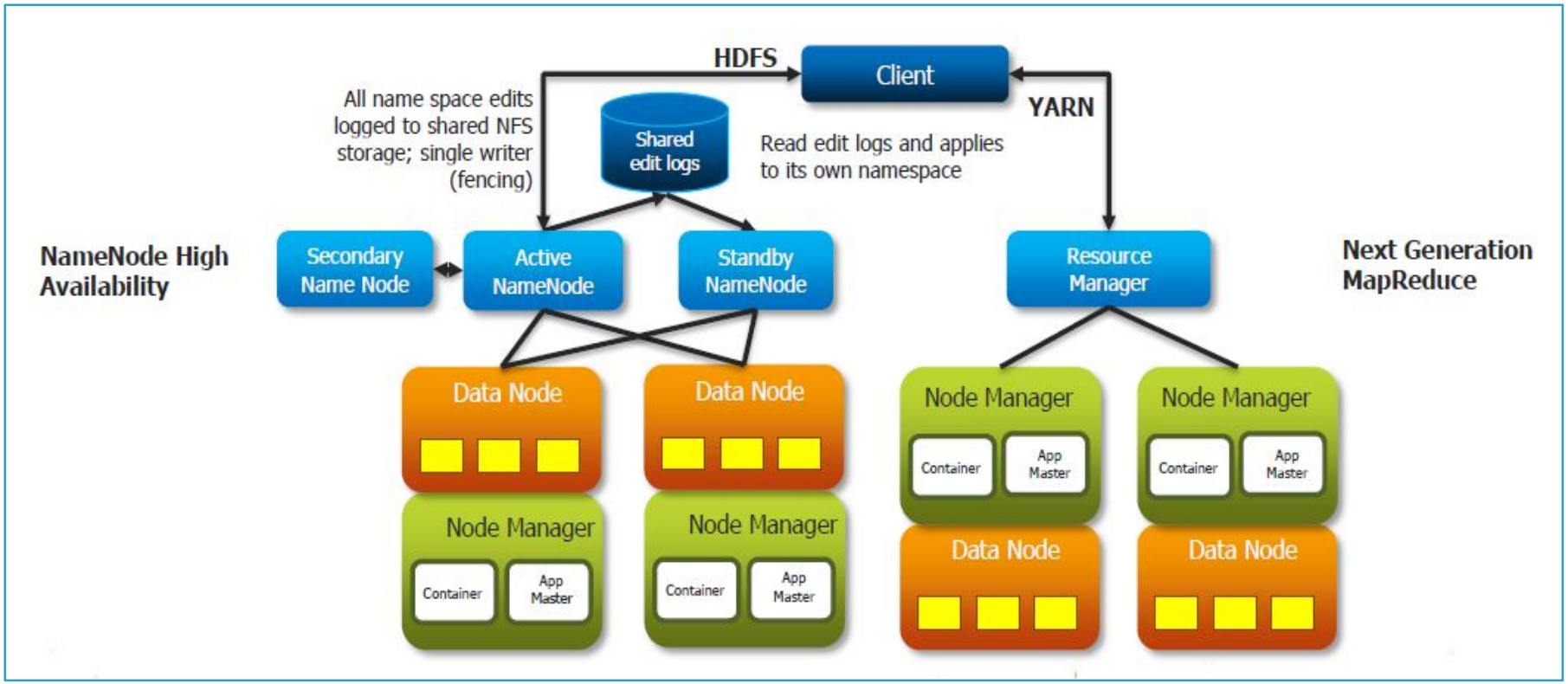
MapReduce

- Splits a task across processors
- "near" the data & assembles results
- Self-Healing, High Bandwidth
- Clustered storage
- JobTracker manages the TaskTrackers

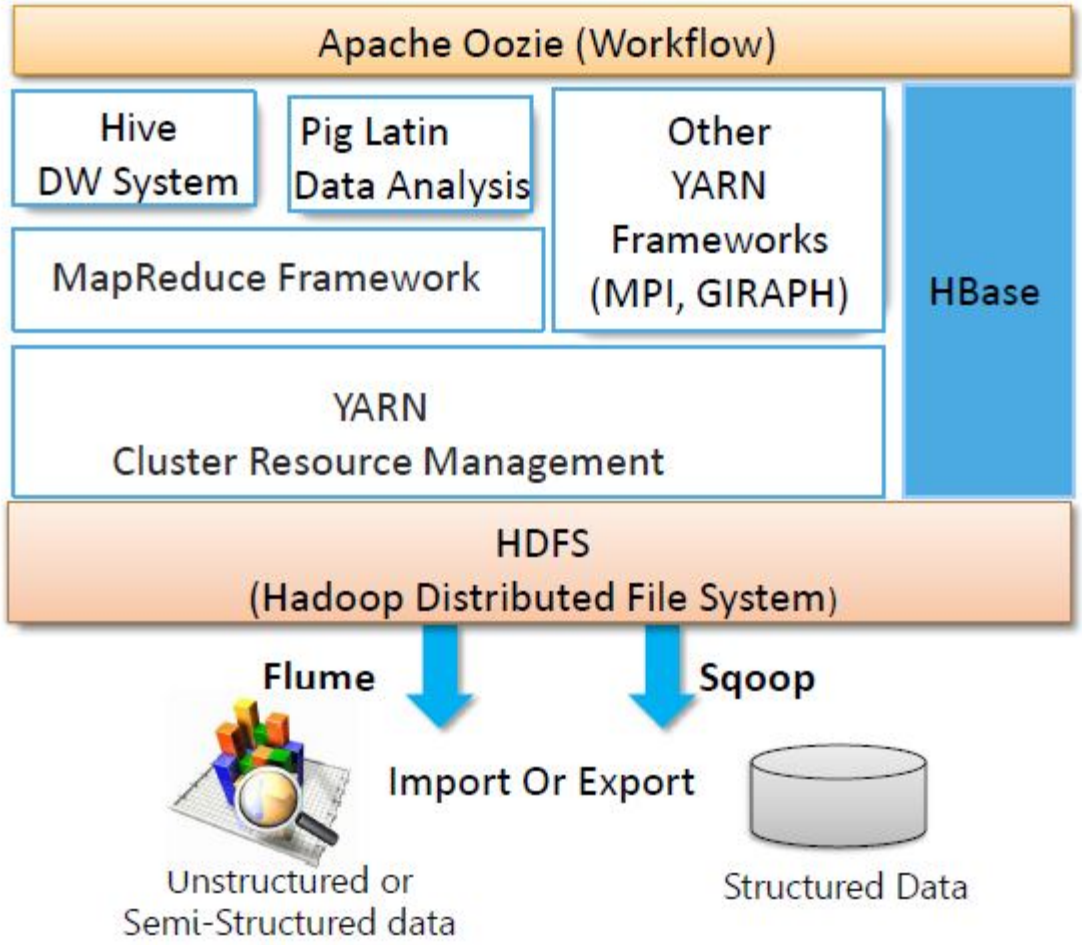
Hadoop Architecture (1.x)



Hadoop Architecture (2.x)



Hadoop EcoSystem

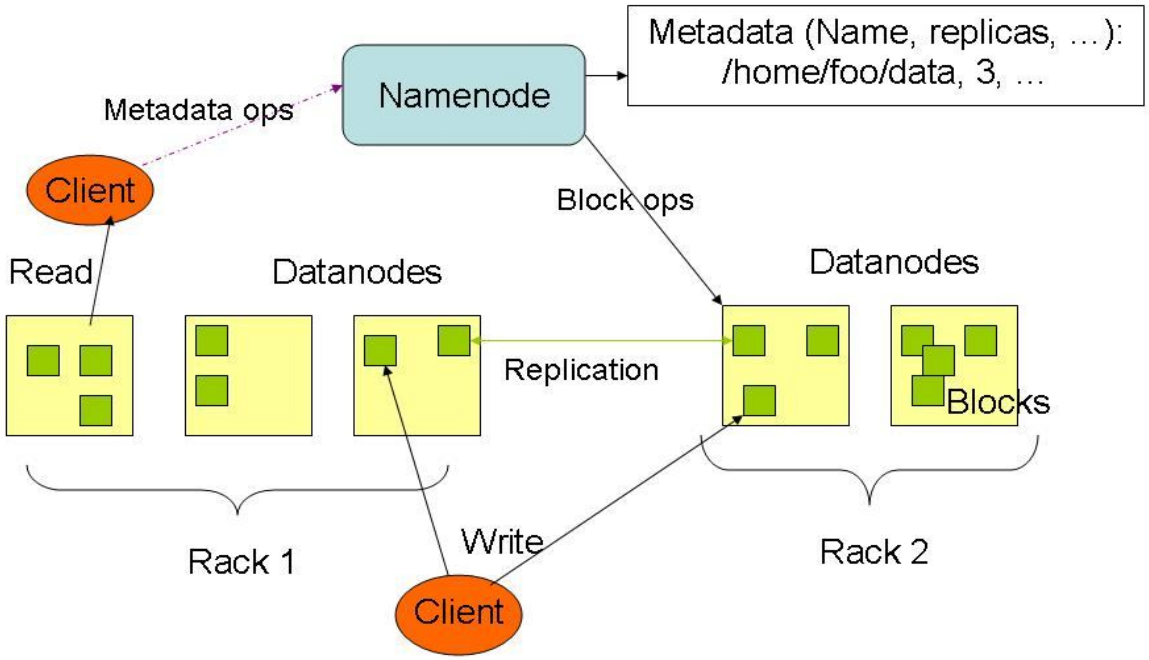


Hadoop 1.X	Hadoop 2.X
Limited to 4,000 nodes per cluster	Up to 10,000 nodes per cluster
'0' number of tasks in a cluster	'0' (Cluster Size)
Jobtracker bottleneck	Efficient cluster utilization - YARN
Has only one namespace for handling HDFS	Supports multiple namespace for handling HDFS
Map and Reduce slots are static	Not restricted to Java
Has only one job - to run MapReduce	Any application can integrate with Hadoop

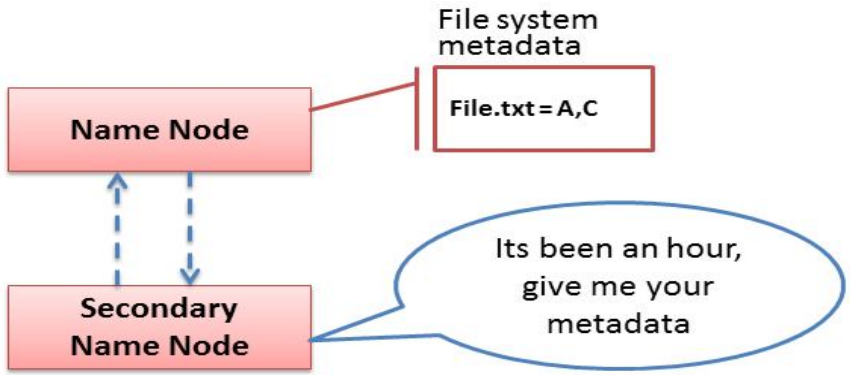
HDFS (Hadoop Distributed File System)

- HDFS Overview
- HDFS Architecture
- HDFS Components
  - Name Node
  - Secondary Name Node
  - Data Node
- HDFS File Write Anatomy
- HDFS File Read Anatomy
- Rack Awareness
- Hadoop File Formats
  - Text, Sequence, Avro, Parquet, ORC, etc.
- Hadoop Compression Techniques
  - Snappy, gzip, bgzip2, LZO, etc.
- Data Replication, Fault Tolerance, Resiliency, HA

HDFS Architecture

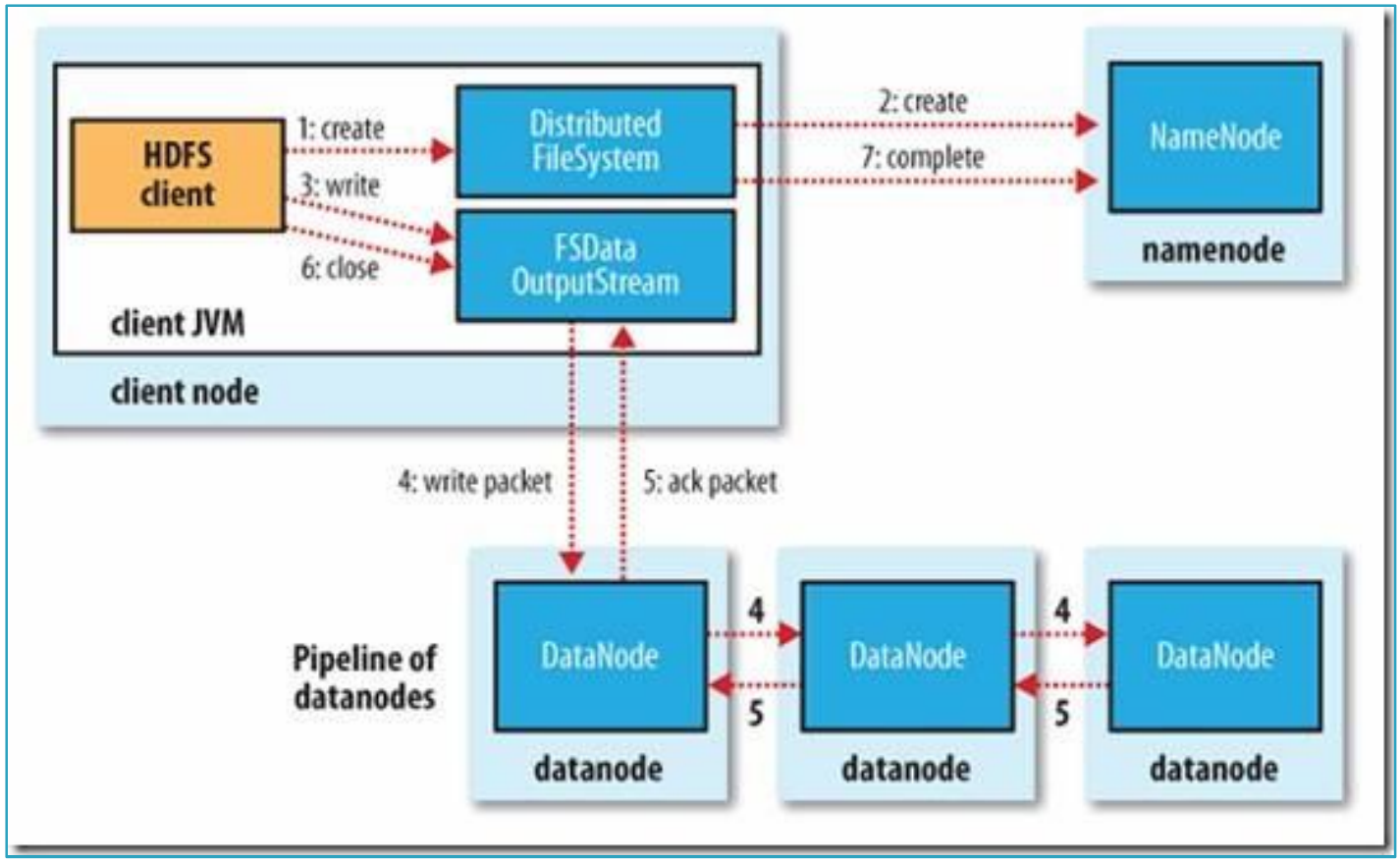


Secondary Name Node

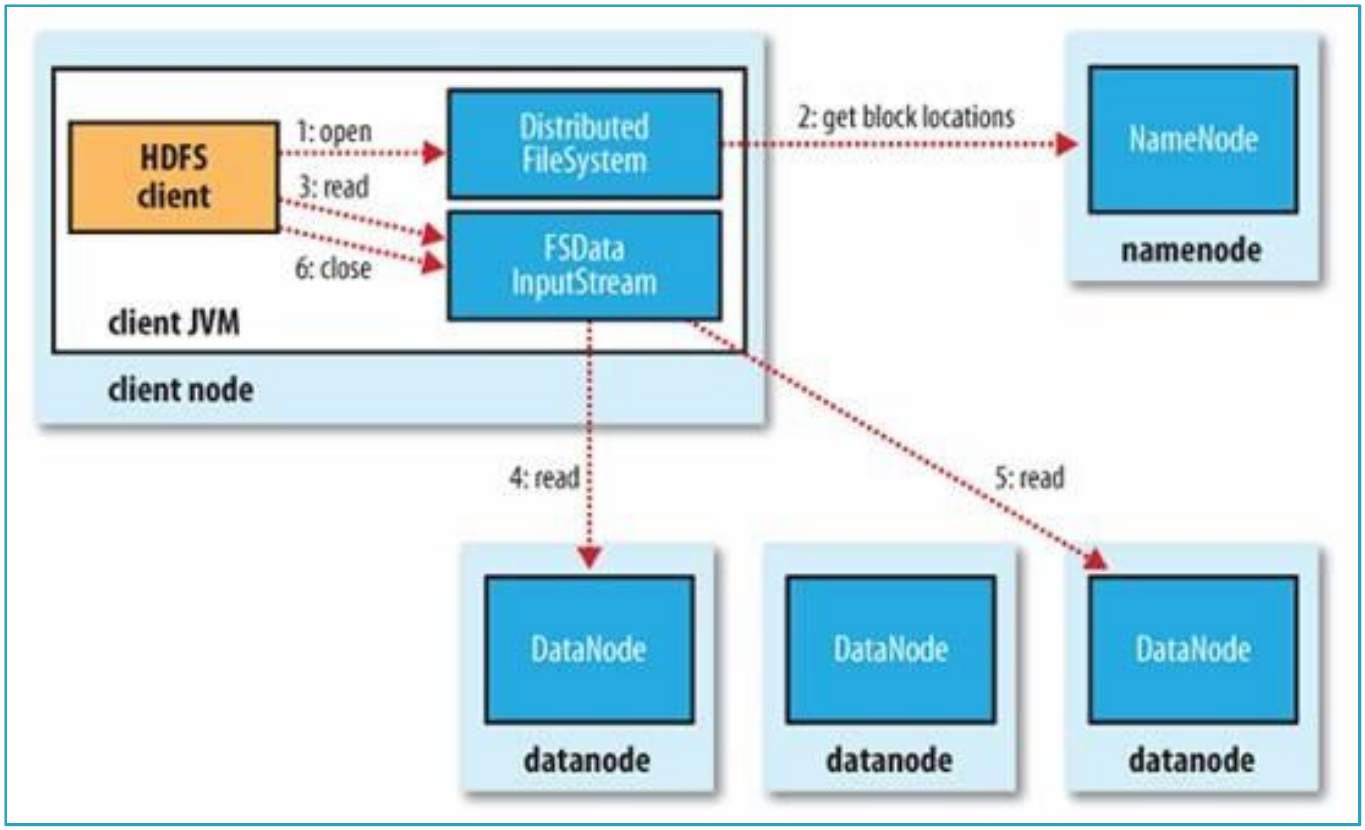


- Not a hot standby for the Name Node
- Connects to Name Node every hour*
- Housekeeping, backup of Name Node metadata
- Saved metadata can rebuild a failed Name Node

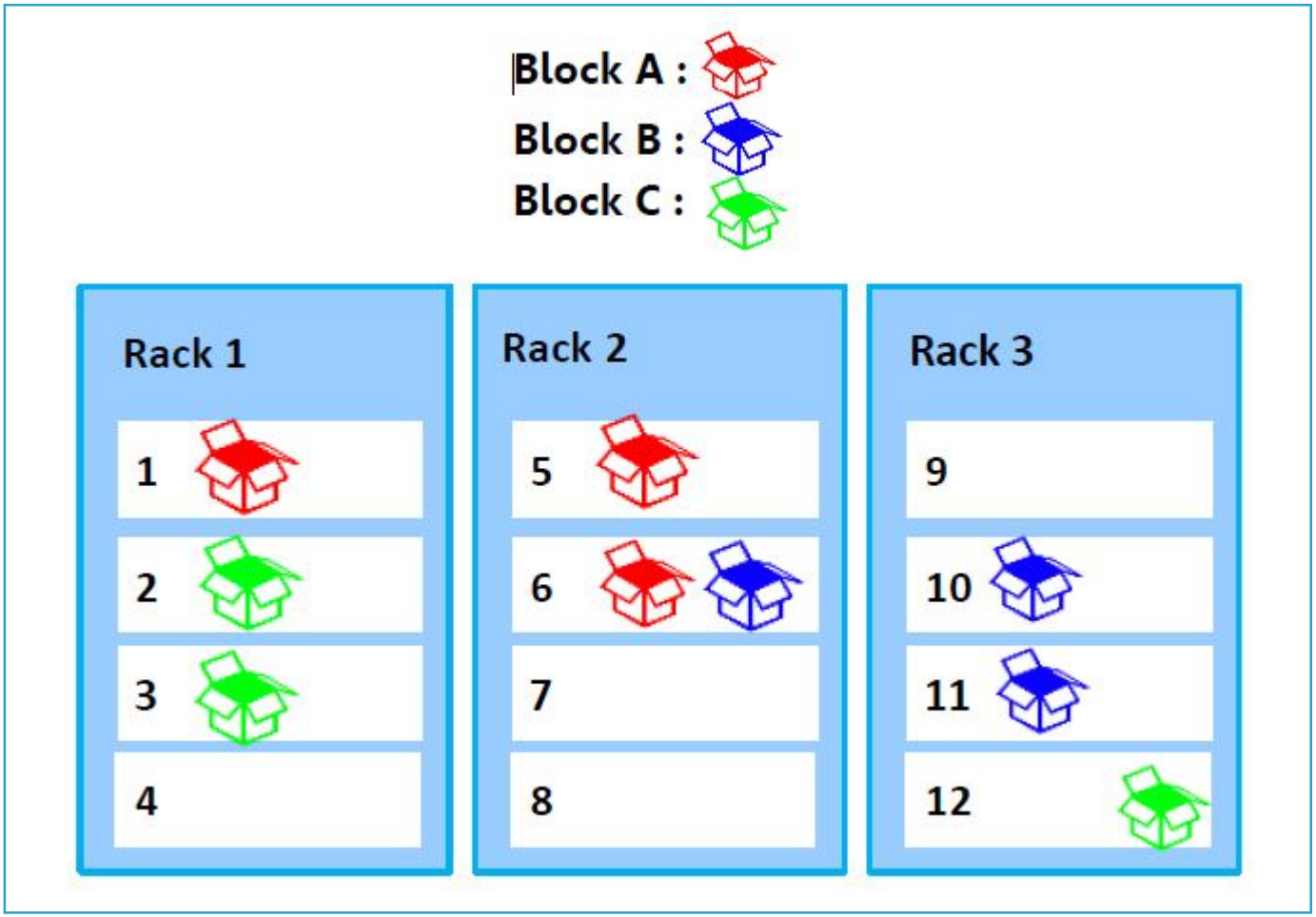
HDFS File Write Operation



HDFS File Read Operation



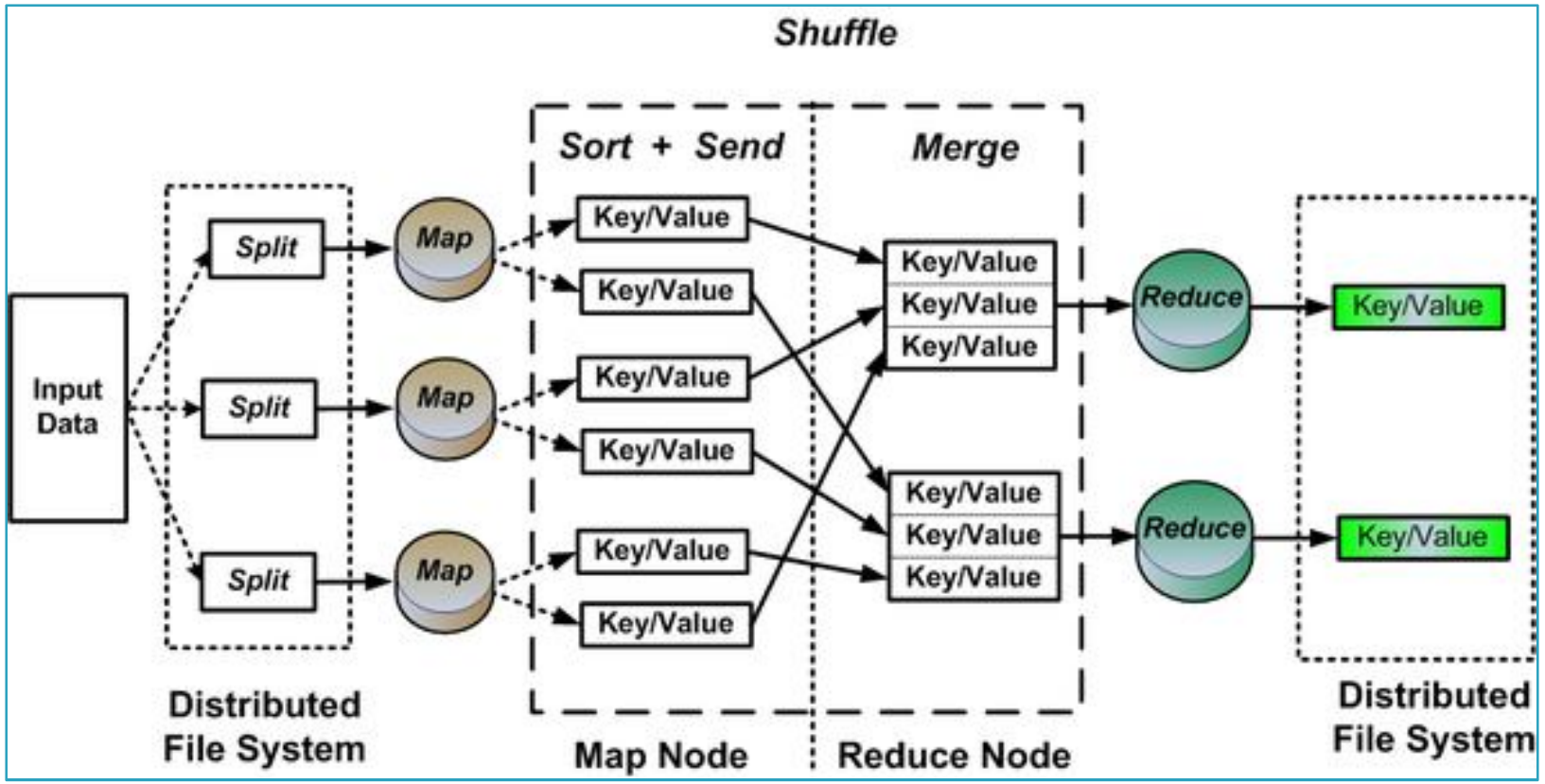
Rack Awareness



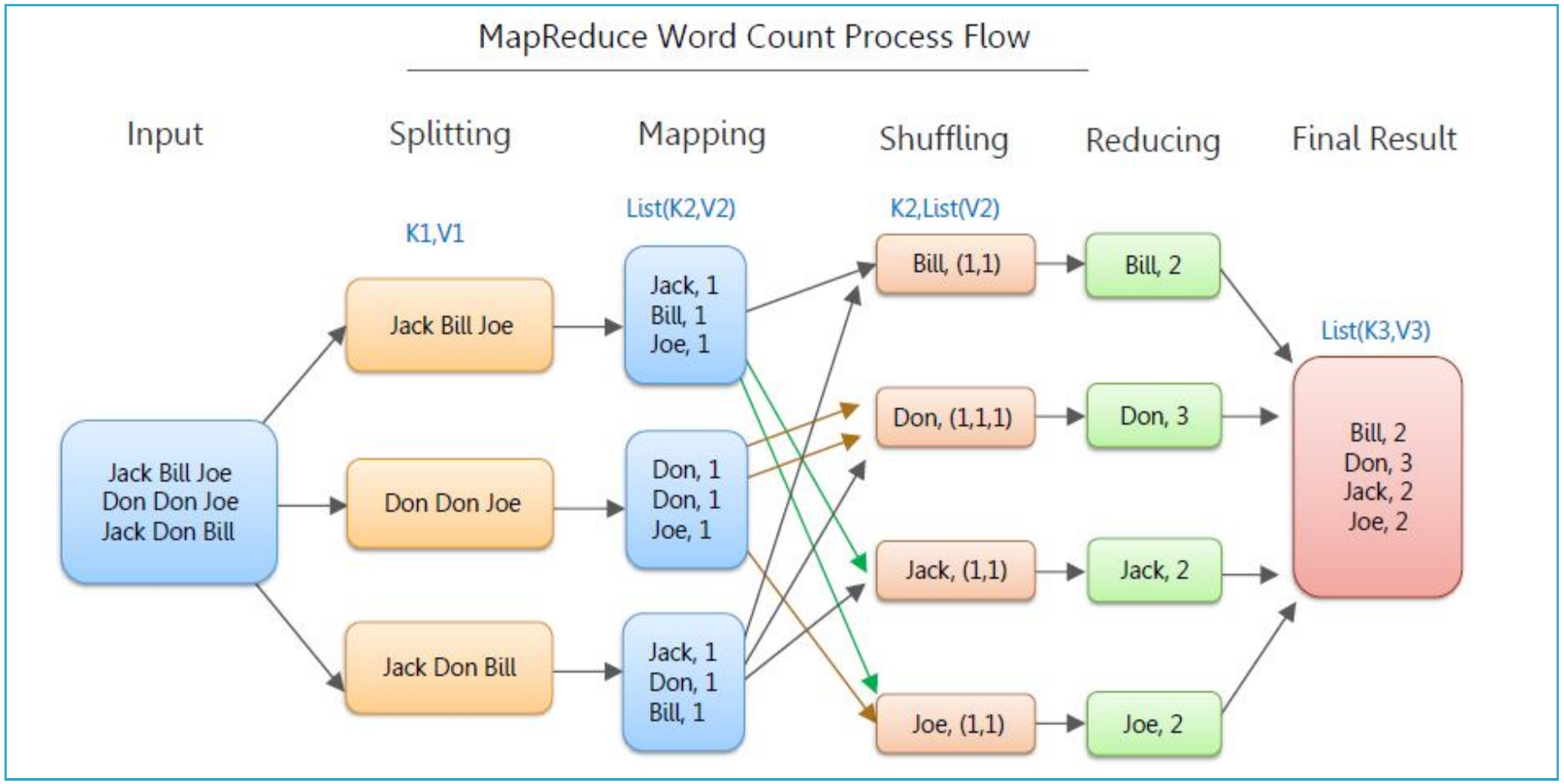
MapReduce Framework

- MapReduce Overview
- MapReduce Architecture
- MapReduce Concepts
  - Splits, Mappers, Reducers, Partitioners, Combiners and Counters
- Input / Output File Formats
- Map Side Join / Reduce Side Join
- YARN Overview
- YARN Components
  - Resource Manager, Node Manager, Container, App Master
- Job Scheduler
  - Fair Scheduler, Capacity Scheduler, etc.
- MR1 vs MR2
- MapReduce Job Execution
- Fault Tolerance, Resiliency, Reliability, HA

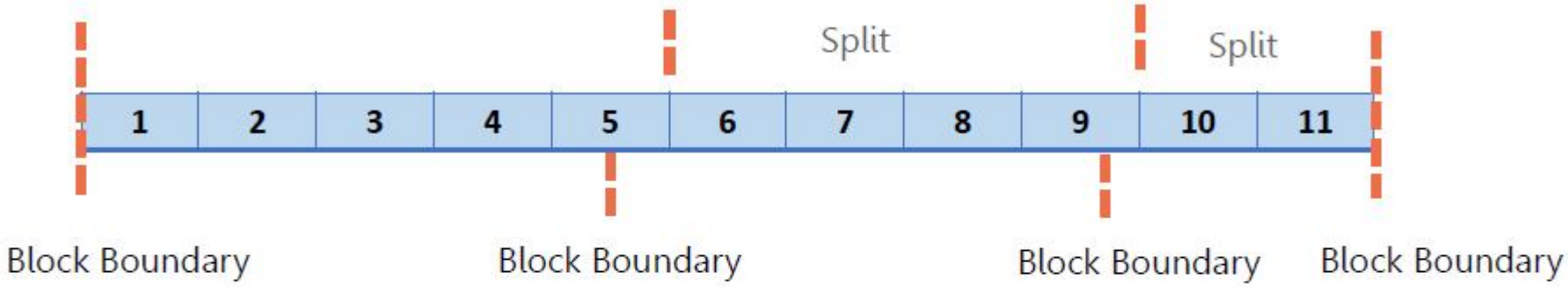
MapReduce Architecture



MapReduce Process Flow

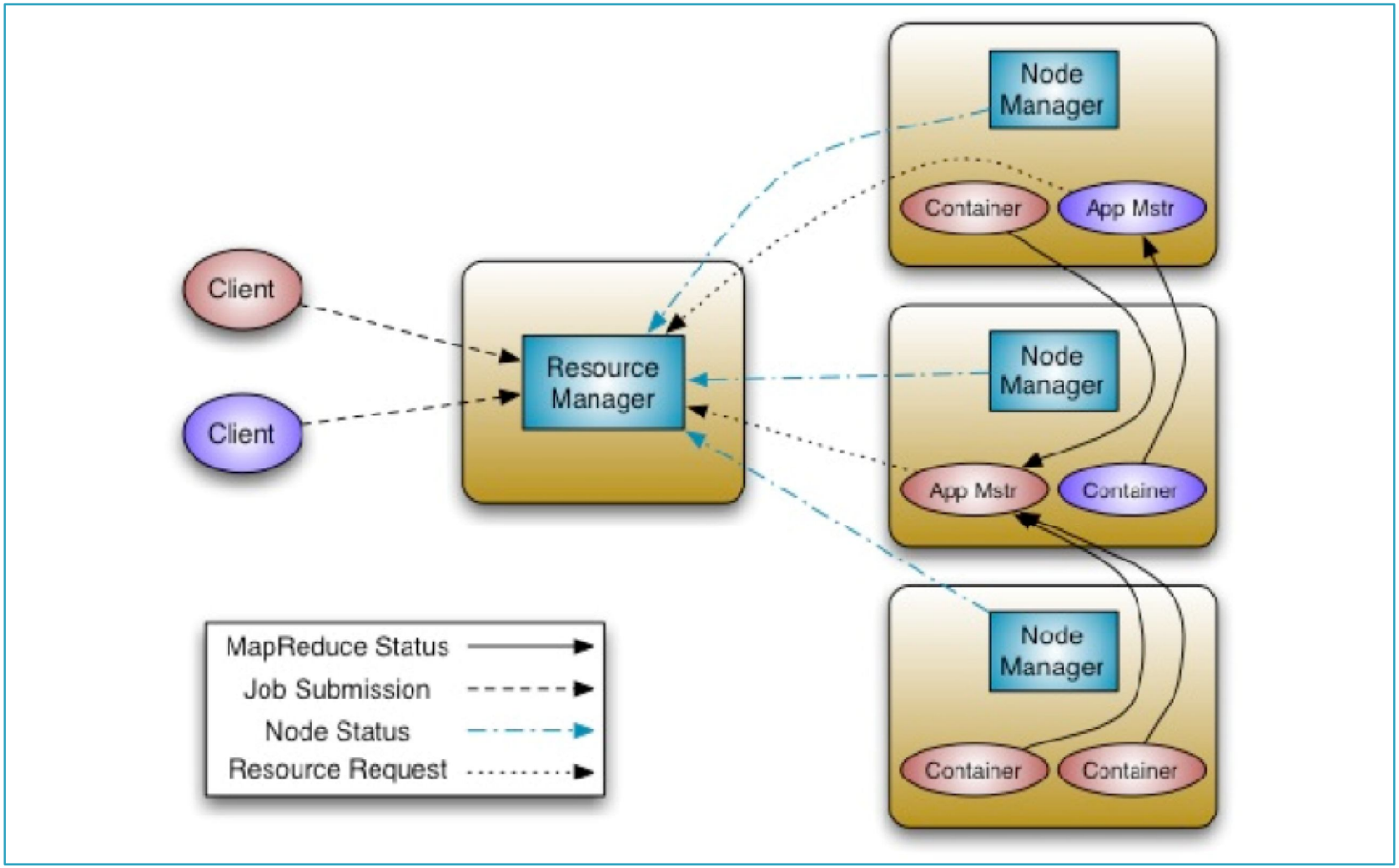


HDFS Blocks vs Input Splits

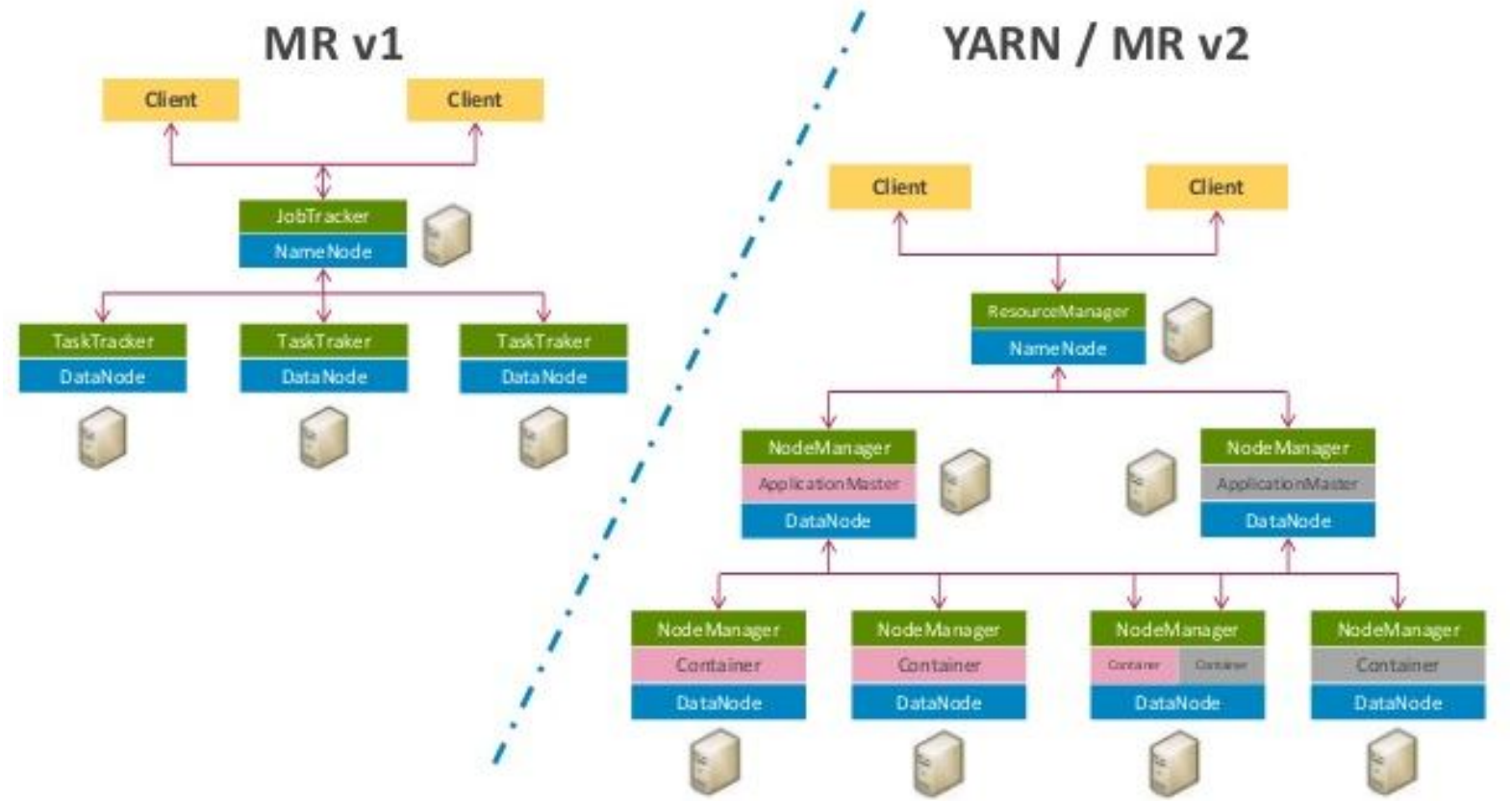


- In an ideal scenario, Block Size == Input Split Size
- However, Logical records rarely fit into HDFS blocks
- Logical Records may cross the block boundaries
- Deltas are read remotely

YARN Architecture

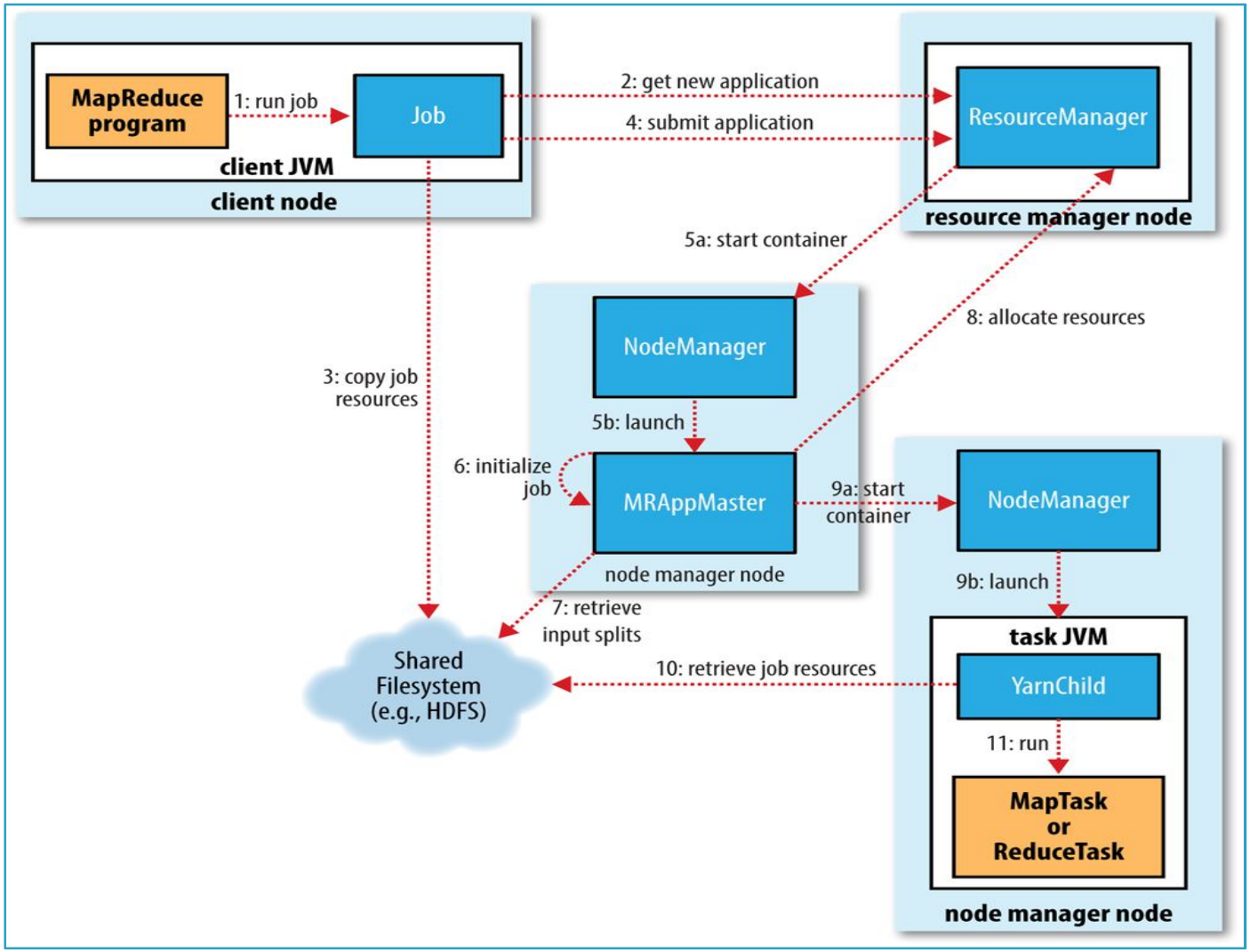


MR1 vs YARN/MR2 Architecture



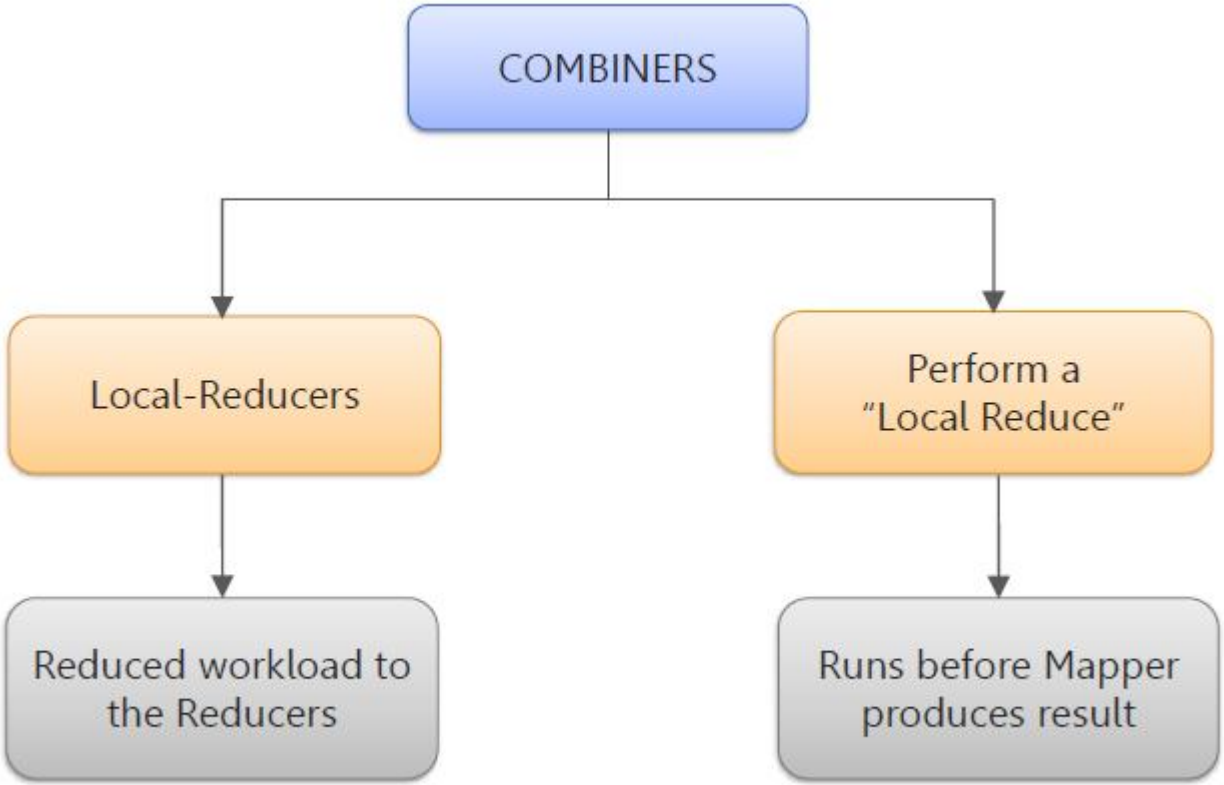
- **YARN** : **Y**et **A**nother **R**esource **N**egotiator
- **MR** : MapReduce

MR Job Execution Flow

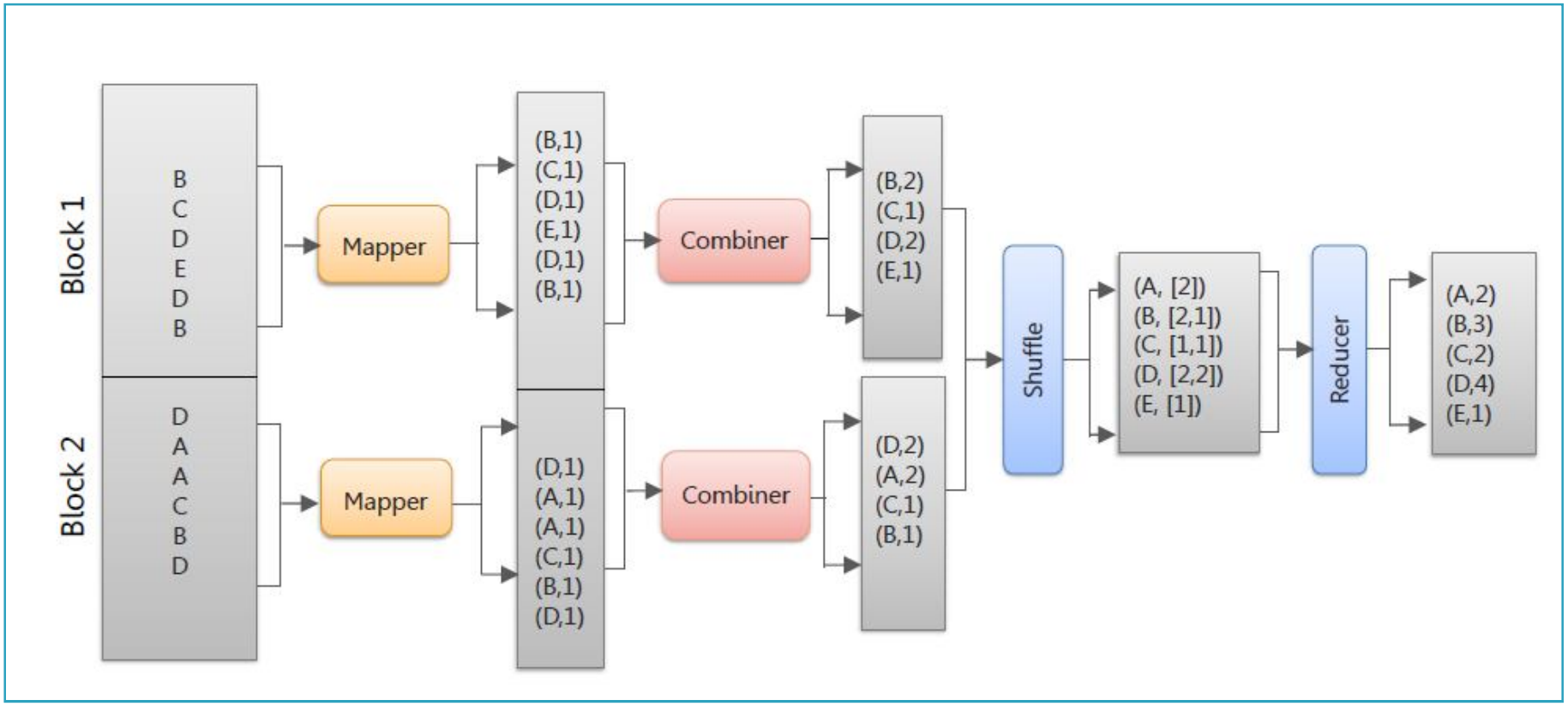


MR Advanced Concepts

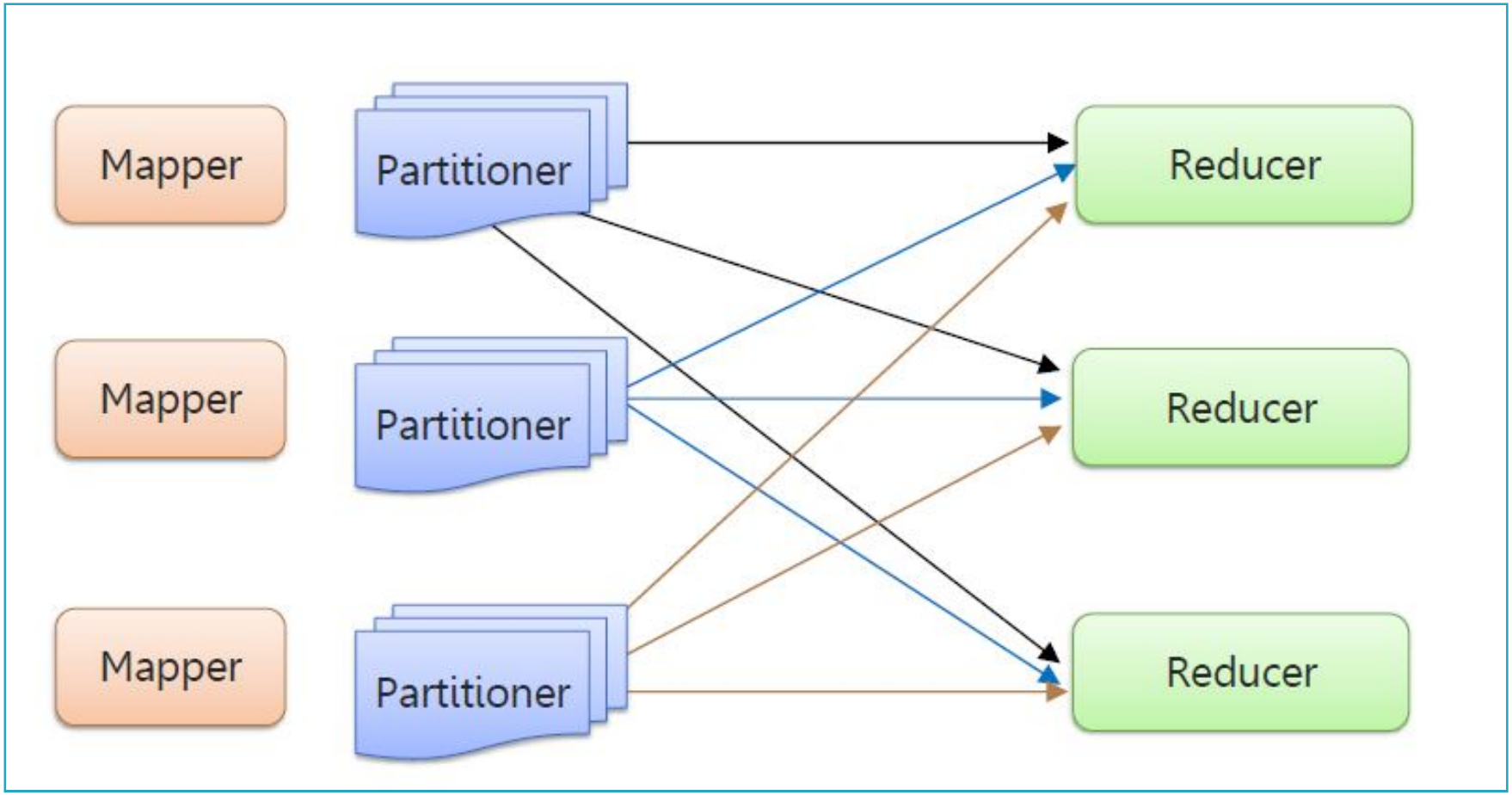
Combiner


COMBINERS
Local-Reducers
Perform a
"Local Reduce"
Reduced workload to
the Reducers
Runs before Mapper
produces result

Combiner in Action

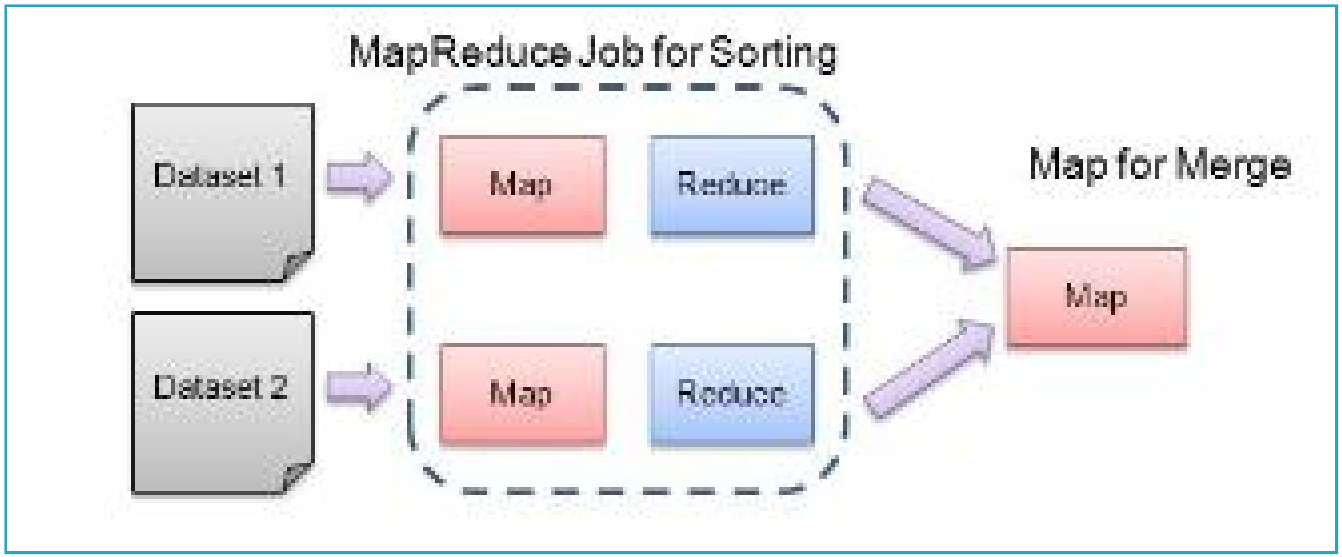


Partitioner

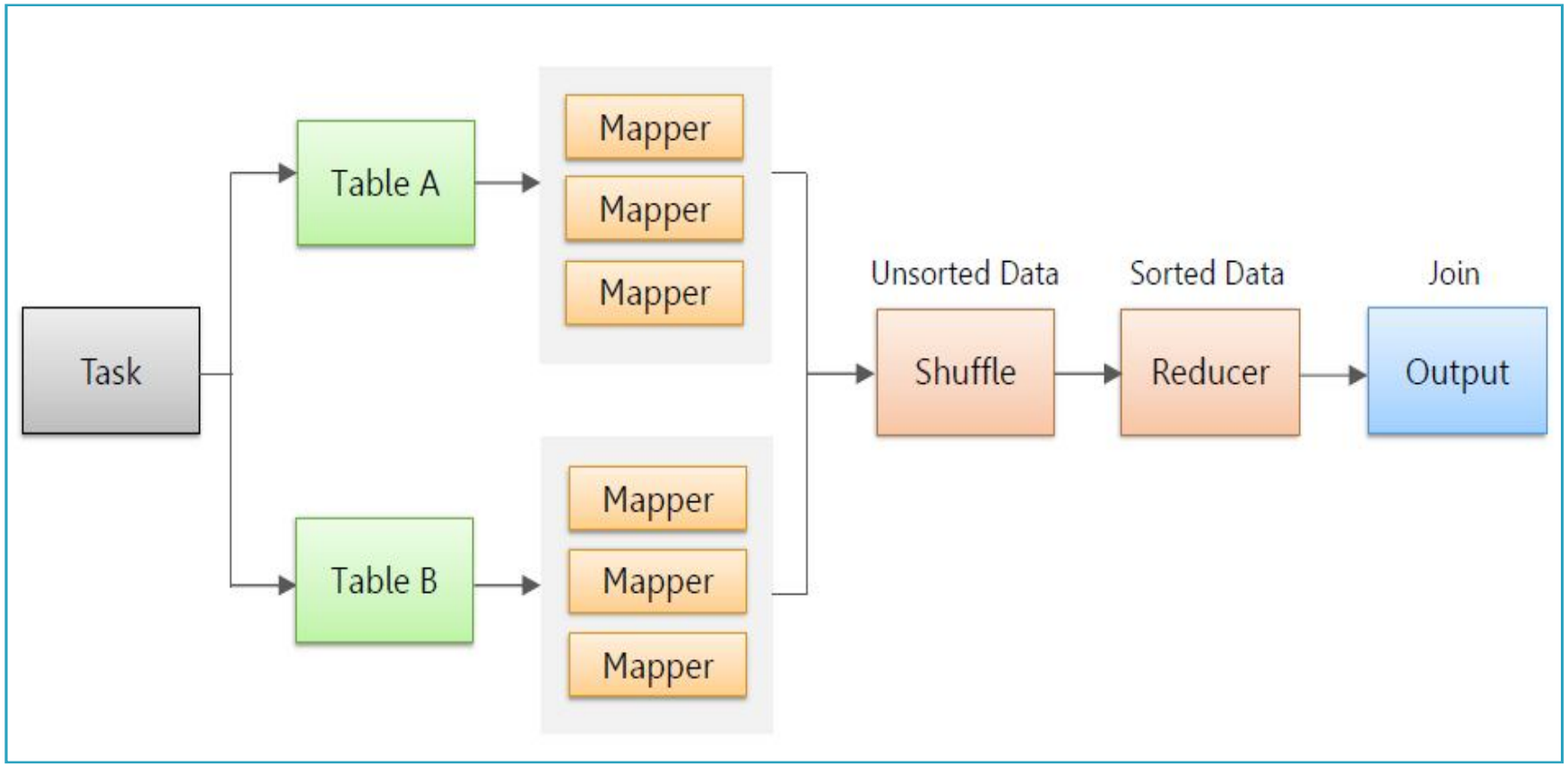


Map Side Join

- A map-side join works by performing the join before the data reaches the map function
- Requirements
  - Each input dataset must be divided into the same number of partitions
  - It must be sorted by the same key (the join key) in each source
  - All the records for a particular key must reside in the same partition
- Above requirements actually fit the description of the output of a MapReduce job
  - A map-side join can be used to join the outputs of several jobs that had the same number of reducers, the same keys, and output files that are not splittable



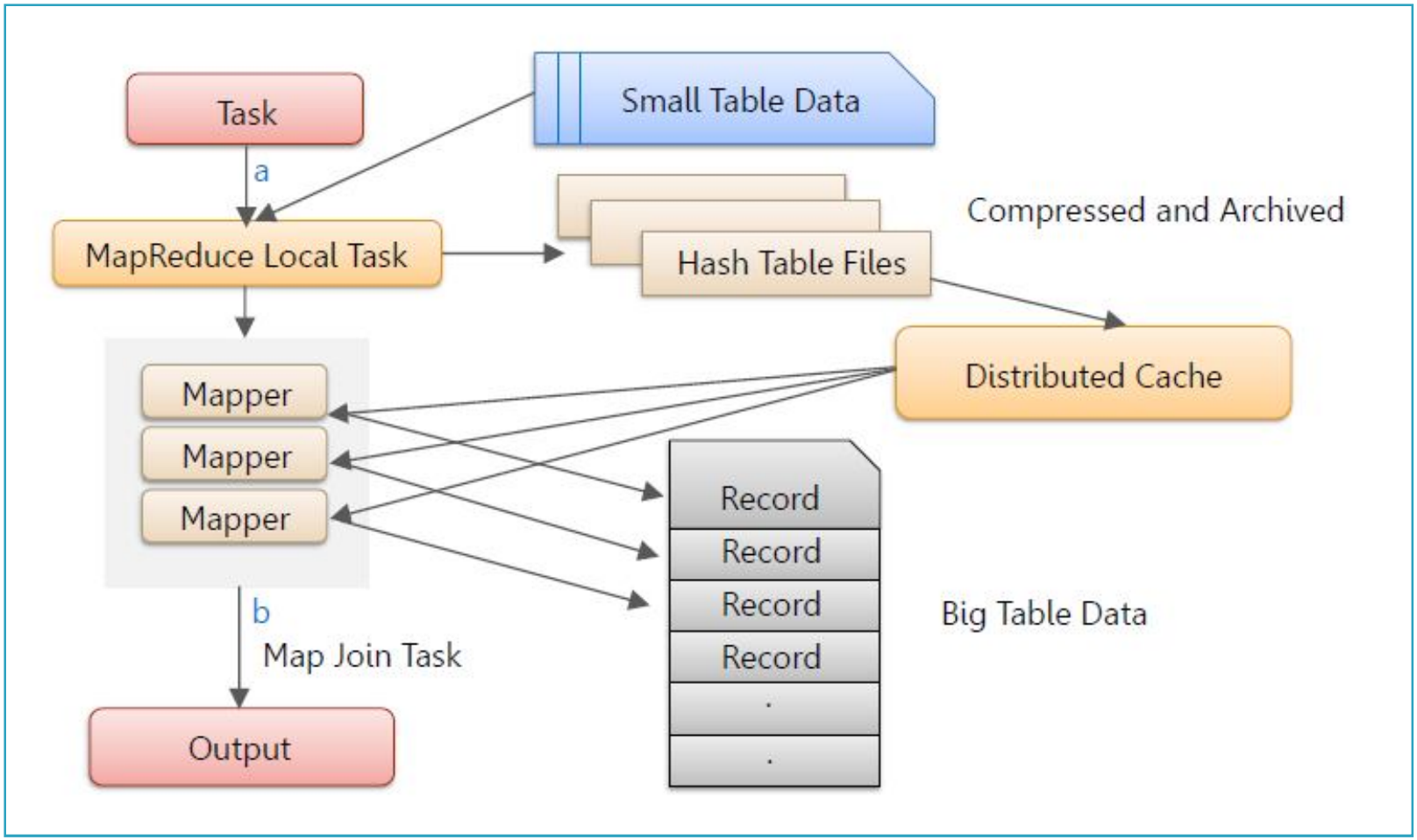
Reduce Side Join



Distributed Cache

- At times, we might need to refer a relatively smaller amount of data in Map Reduce jobs
  Examples: Master Files, Lookup files, Cross Reference Files etc.
- Map Reduce provides Distributed Cache facility. This allows to cache files (text, archives, jars etc.) needed by applications
- Files are copied only once per job and should not be modified by the application or externally while the job is executing
- Both Mappers and reducers can access these files
- DistributedCache can be used to distribute simple, read-only data/text files and/or more complex types such as archives, jars etc via the JobConf
- Programmer need to be careful to not use too many cache files
- The ideal size for distributed cache file is <= 100 to 150 MB

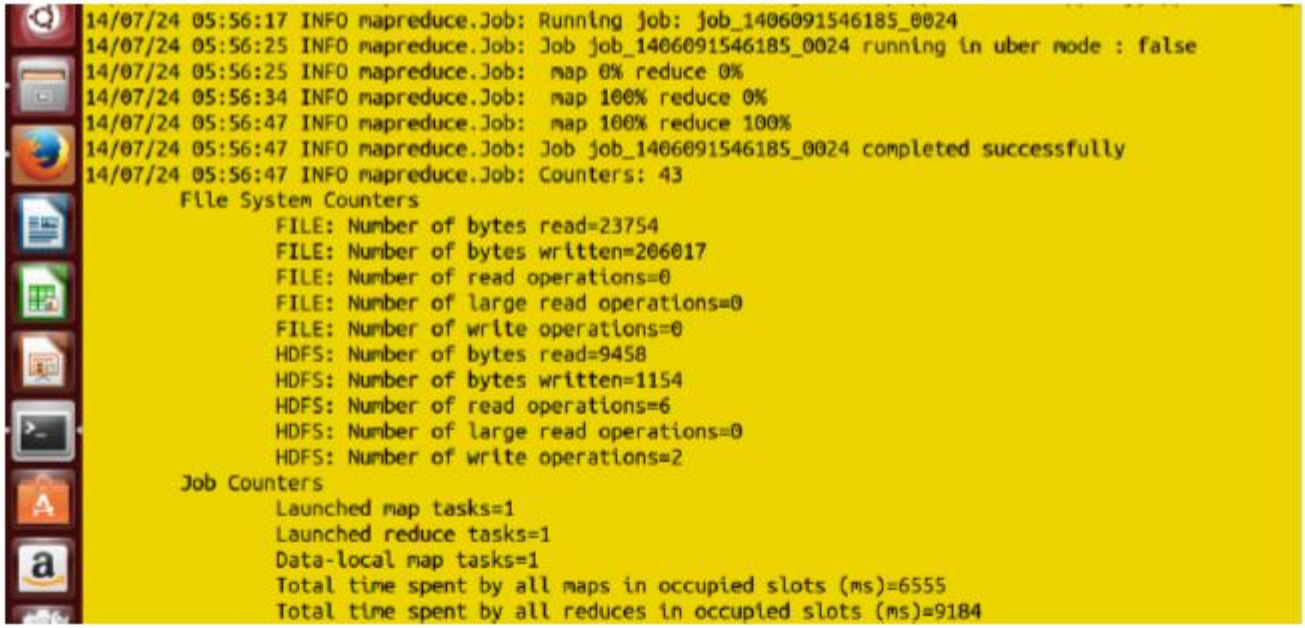
Distributed Cache (contd.)



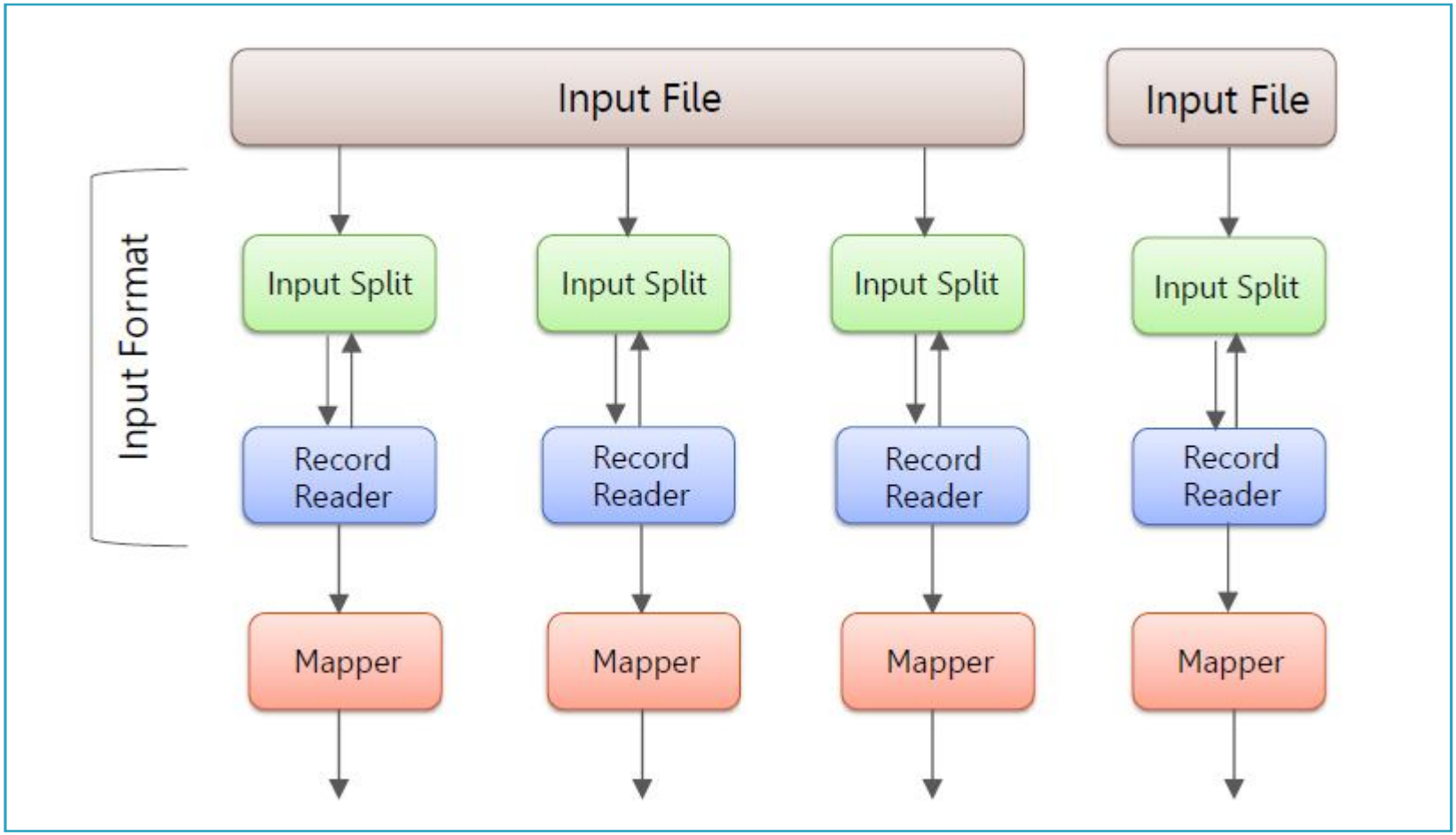
Counters

- Counters are lightweight objects in Hadoop that allow you to keep track of system progress in both the map and reduce stages of processing

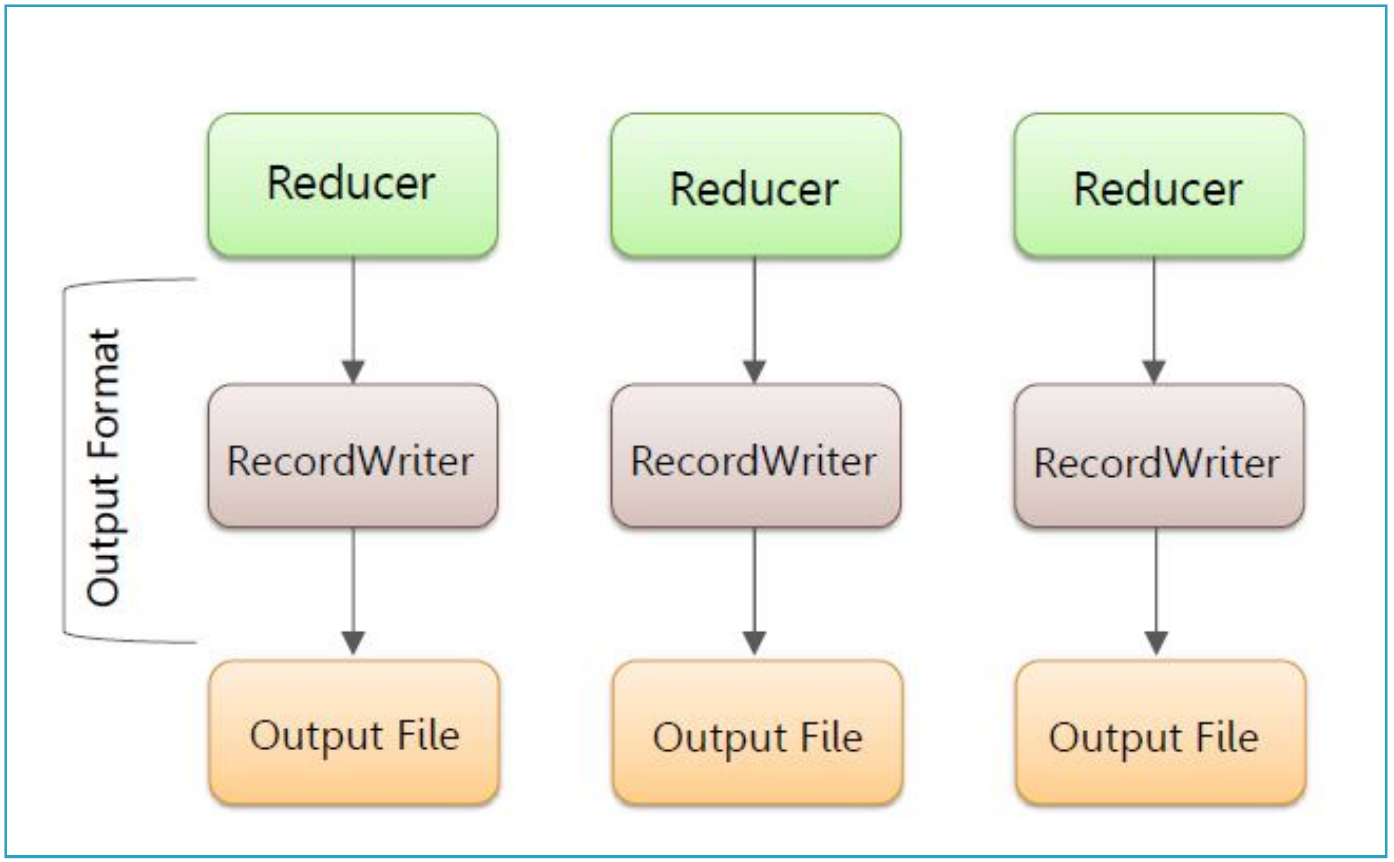
- Counters are used to gather information about the data we are analysing, like how many types of records were processed, how many invalid records were found while running the job, and so on



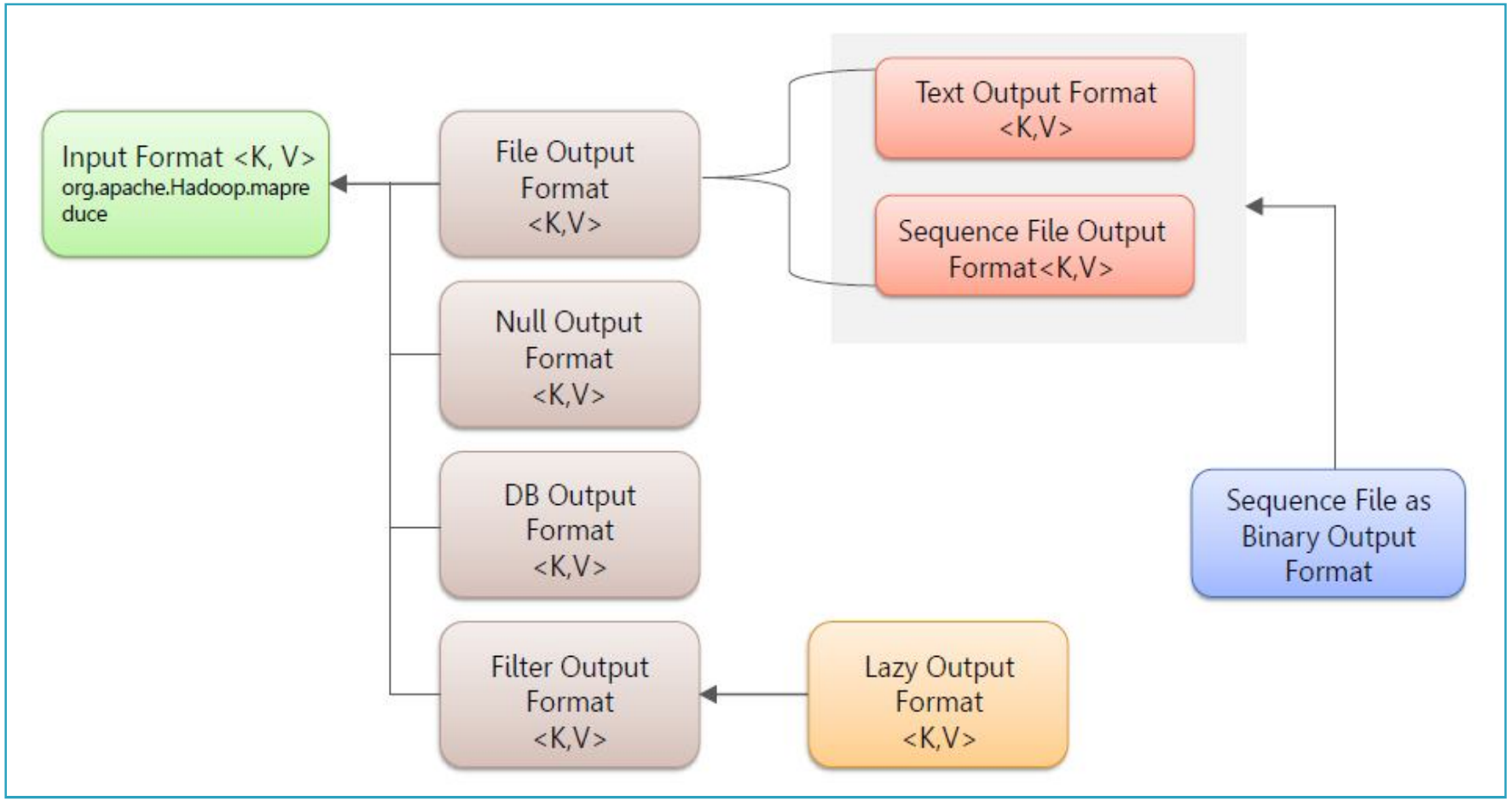
Hadoop Input Formats



Hadoop Output Formats



Hadoop Output Formats – Class Hierarchy



Thank You!