

Big data Introduction

Agenda

- What are the characteristics of Big data
- Generic concepts related to Datawarehouse platform
- Why traditional Datawarehouse platform are not consider suitable for Big data analytics
- Understanding the vast world of Big data platform
- Benefit of Hadoop file system and other cloud storage
- Mastering the concept of Parallel distributed Computing framework
- Component failure and recoveries
- Demystifying the Map Reduce

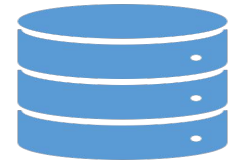
What is BigData?

- Big data is a phrase, used to describe a huge amount of both structured and unstructured data that is so large that it's practically impossible to store, process and access using traditional techniques
- Following are the characteristics exhibited by Big data
 - Massive Volume
 - Lots of unstructured data
 - Not possible to store this process using traditional RDBMS and ETL tools

Three Vs of Bigdata



Volume : Terabytes and petabytes (and even exabytes) of data



Velocity : Data flows into an organization at increasing rates

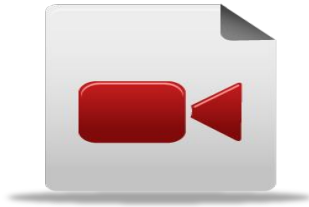


Variety : Any type of structured, Semi structured and unstructured data



Why Volume is increasing?

- Volume increasing exponentially with different Media format



- Lots of sensor, IOT and social media data



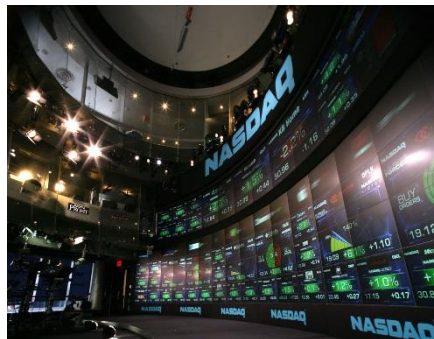
Why Velocity is increasing?

- Rate at which data is coming in
- Rate at which data needs to be analyzed
- Speed of processing should be faster than arrival of new data

2.7 billion LIKES/day



400 million tweets/day



Why Variety is increasing?

- Variety of data needs to be collected to perform advance analytics
- These data can from various sources such as RDBMS
- Application generated data and log file
- XML, JSON,CSV,TSV , Text file and Binary data stored on file systems



Structured data has predefined schema such as field name and data type

- CSV file with header
- Records from data bases

Unstructured data don't have predefined schema

- Text file with column names
- Audio Video
- Images

Common Types of Big Data?

- The advent of social medium, IOT devices and business-to-business transactions are introducing new file types and formats

New types of data

Sentiment

Understand how your customers feel about your brand and products – right now



Clickstream

Capture and analyze website visitors' data trails and optimize your website



Sensors

Discover patterns in data streaming automatically from remote sensors and machines



Geographic

Analyze location-based data to manage operations where they occur



Server Logs

Research logs to diagnose process failures and prevent security breaches



Unstructured

Understand patterns in files across millions of web pages, emails, and documents



Big Data Analytics

- Process of collecting, organizing and analyzing large amounts of data to gain insight
- Information and Data as Assets
- Improve decision making
- Enhance performance
- To be competitive
- To be two steps ahead of the competition
- Derive insights and drive growth

What is Data Warehouse?



- A Data Warehouse is a central location where consolidated data from multiple locations are stored.
- The end user accesses it whenever he needs some information.
- Data Warehouse is not loaded every time when new data is generated.
- There are timelines determined by the business as to when a Data Warehouse needs to be loaded – daily, monthly, once in a quarter etc.

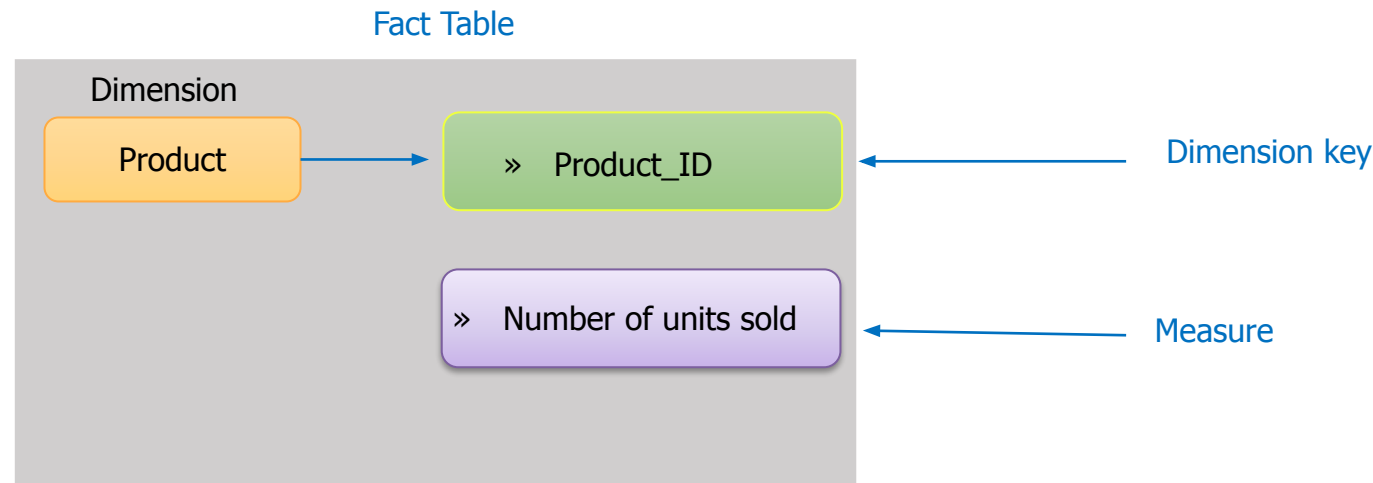
Dimension Table?

- The objects of the subject are called Dimensions.
- The tables that describe the dimensions involved are called dimension tables.
- Dividing a data warehouse project into dimensions, provides structured information for reporting purpose.

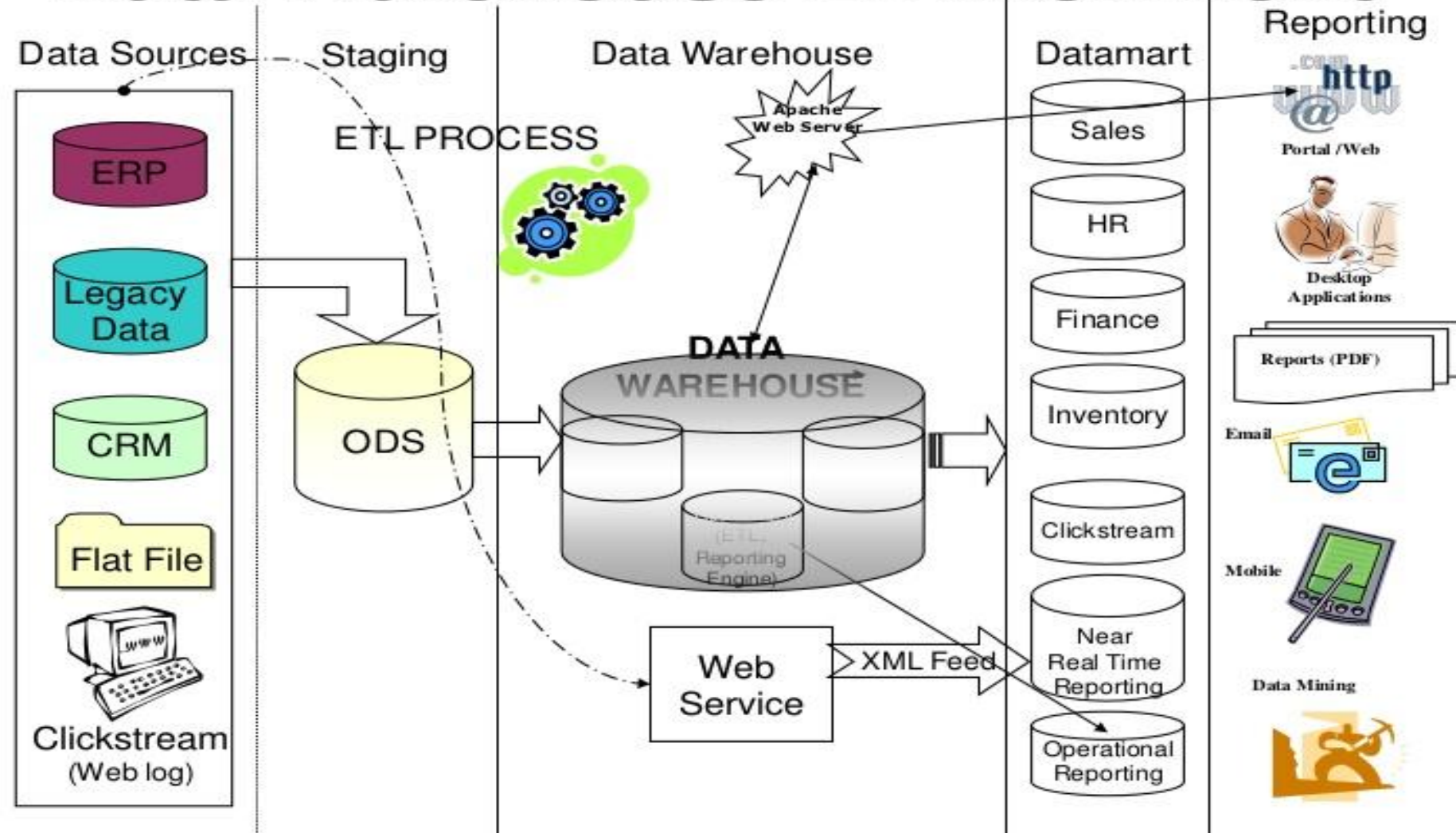
e-commerce Company									← Subject
Customer			Product			Date			← Dimensions
ID	Name	Address	ID	Name	Type	Order date	Shipment date	Delivery date	← Attributes

What are Facts?

- A fact is a measure that can be summed, averaged or manipulated. If a fact is manipulated, it has to be a measure that makes a business sense.
- A dimension is linked to a fact.
- A fact table contains 2 kinds of data – a dimension key and a measure.

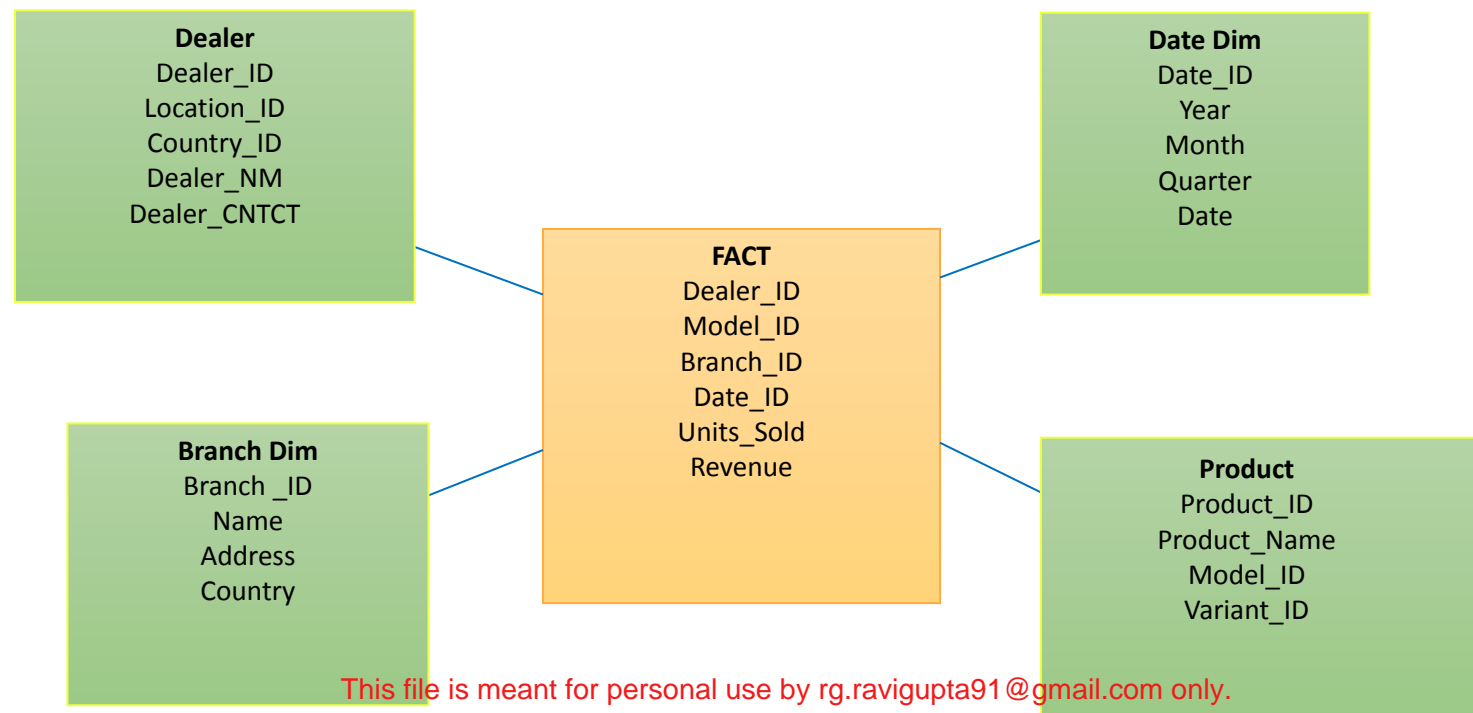


Data Warehouse Environment



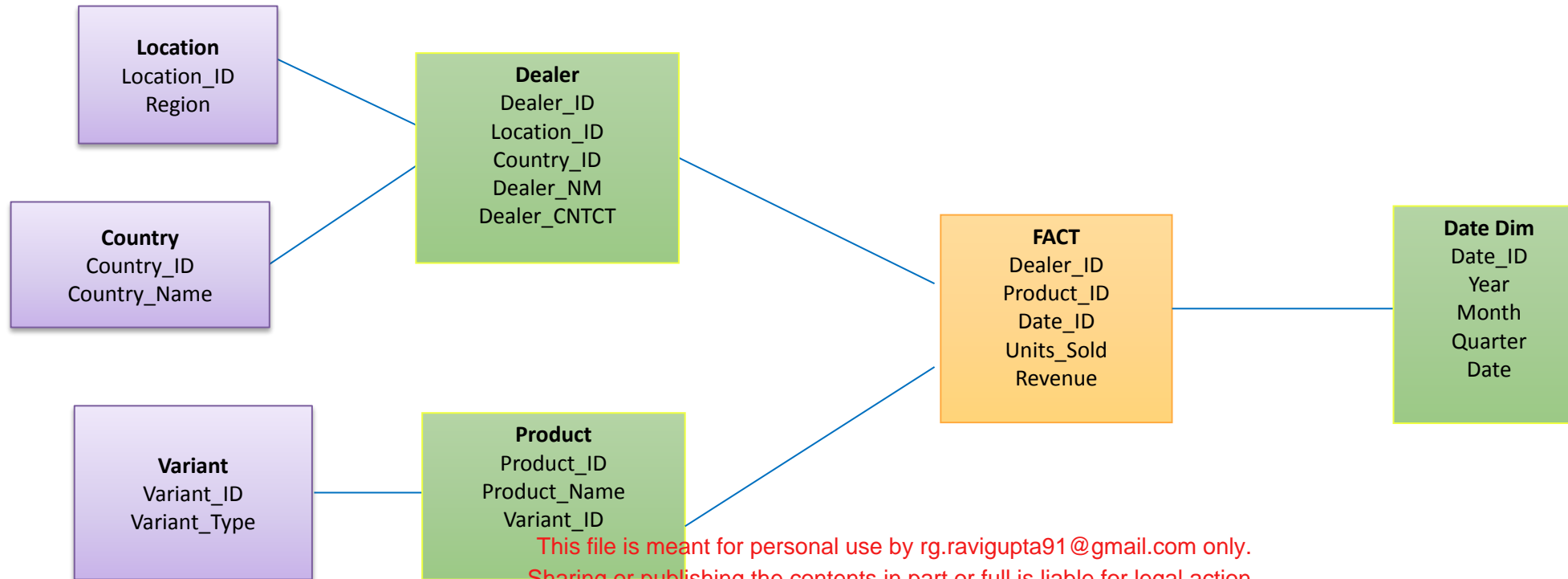
Star Schema

- A star schema is the simplest of all the data warehouse schemas.
- The dimensions are not normalized; Fact has keys from every dimension.
- Performance of the Star schema is good, but cost of storage is high



Snowflake Schema?

- Snow flaking is where we normalize the dimension tables of a star schema.
- In Snow Flake schema, Dimensions are normalized
- Query performance not so good as joins are more



Limitations of traditional data warehouse

- Not Possible to store huge data (Big Data)
- No Flexible schema
- Not suitable for storing un-structured data
- Limited scalability
- Takes huge time to perform ETL activity and load data into warehouse
- Not suitable for doing advance machine learning and AI
- Very expensive to develop, deploy and maintain the system

What is Dake Lake?



- A data lake is a centralized platform where all kinds of data can be stored at any scale.
- In Data lake, we store data in original form
- Data lakes are cluster of inexpensive and scalable commodity hardware.
- Unlike databases, A data lake works on a schema-on-read concept.
- Data scientists can access, prepare, and analyze data faster and with more accuracy using data lakes.
- This massive amount of data available in data lake can help companies to perform sentiment analysis or fraud detection.

Data Lake Vs Data Warehouse

Characteristics	Data Warehouse	Data Lake
Data	Relational from transactional systems, operational databases, and line of business applications	Non-relational and relational from IoT devices, web sites, mobile apps, social media, and corporate applications
Schema	Designed prior to the DW implementation (schema-on-write)	Written at the time of analysis (schema-on-read)
Price/Performance	Fastest query results using higher cost storage	Query results getting faster using low-cost commodity storage
Data Quality	Highly curated data	Any data that may or may not be curated (ie. raw data)
Users	Business analysts	Data scientists, Data developers, and Business analysts (using curated data)
Analytics	Batch reporting, BI and visualizations	Machine Learning, Predictive analytics, data discovery and profiling



Bigdata can be stored only on distributed platforms such as

- ✓ Distributed file systems
- ✓ Distributed data bases
- ✓ Distributed Messaging queue or PubSub platform
- ✓ Distributed Search Engines

Not only for storage even for processing we shall need distributed and parallel computing framework to process such data

- ✓ Map Reduce
- ✓ Spark

Storing Big data in file systems

- Big data can be stored in following file system platforms
- Distributed file system such as HDFS, AWS S3, Azure data lake gen(ADLS Gen2) and GCS
- These platforms are storing big data for analytics use cases as ETL Processing, Reporting, Machine learning and AI
- However, these platform will not be suitable if you want to perform CRUD(Create, Retrieve, Update and Delete) operation frequently



Benefits of storing Big data in file systems

- Big block size of 64 or 128 MB
- Bigger blocks enable reading more data in lesser disk IO Operations
- These file systems replicate the data and stored 3 copies of every block
- Replication helps these platforms to provide read scalability and data availability



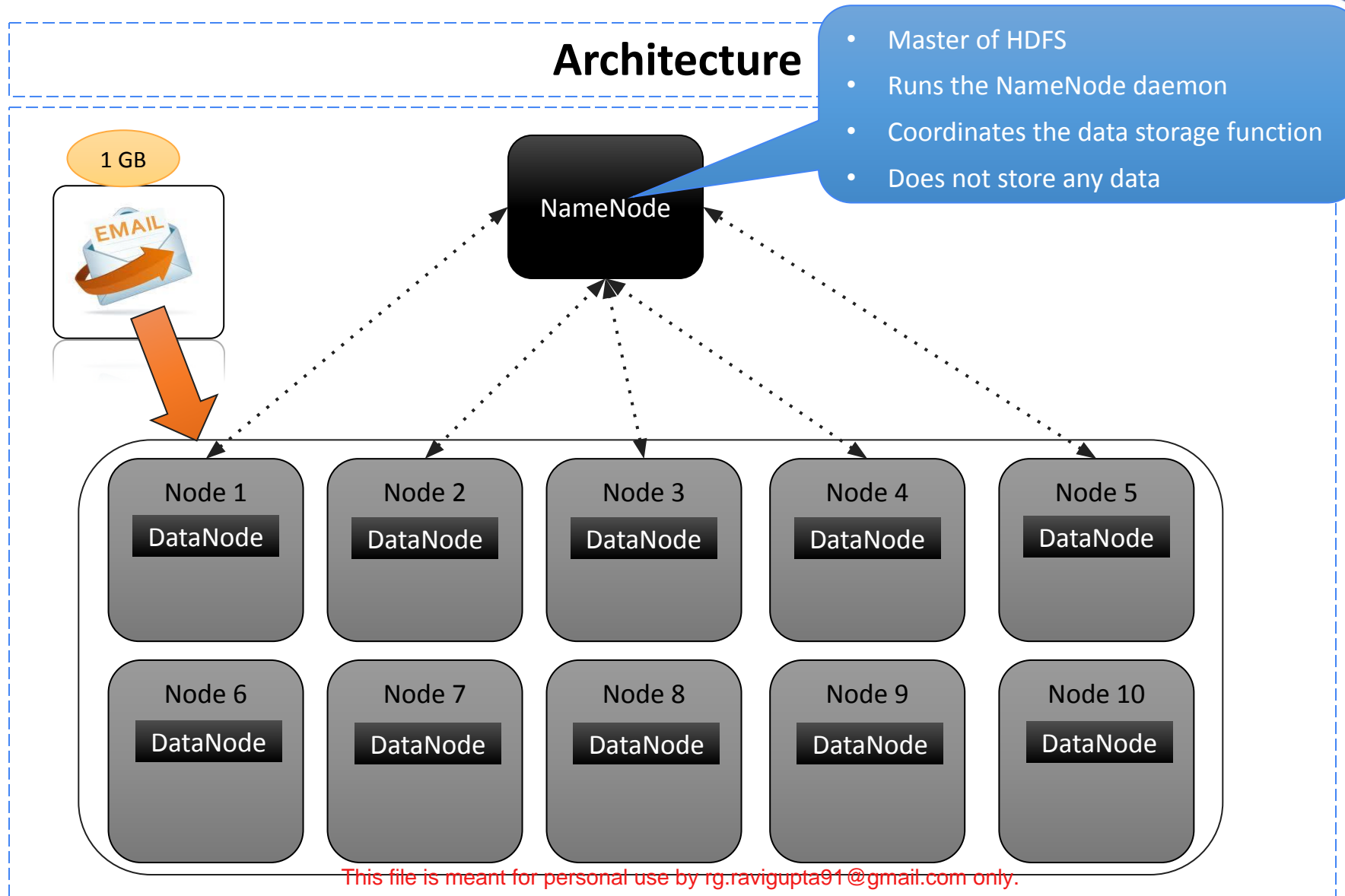
did you know?

- Apache Hadoop's MapReduce and HDFS components were inspired by Google papers on MapReduce and Google File System
- Doug Cutting and Mike Cafarella, were the authors of these paper that were published in October 2003

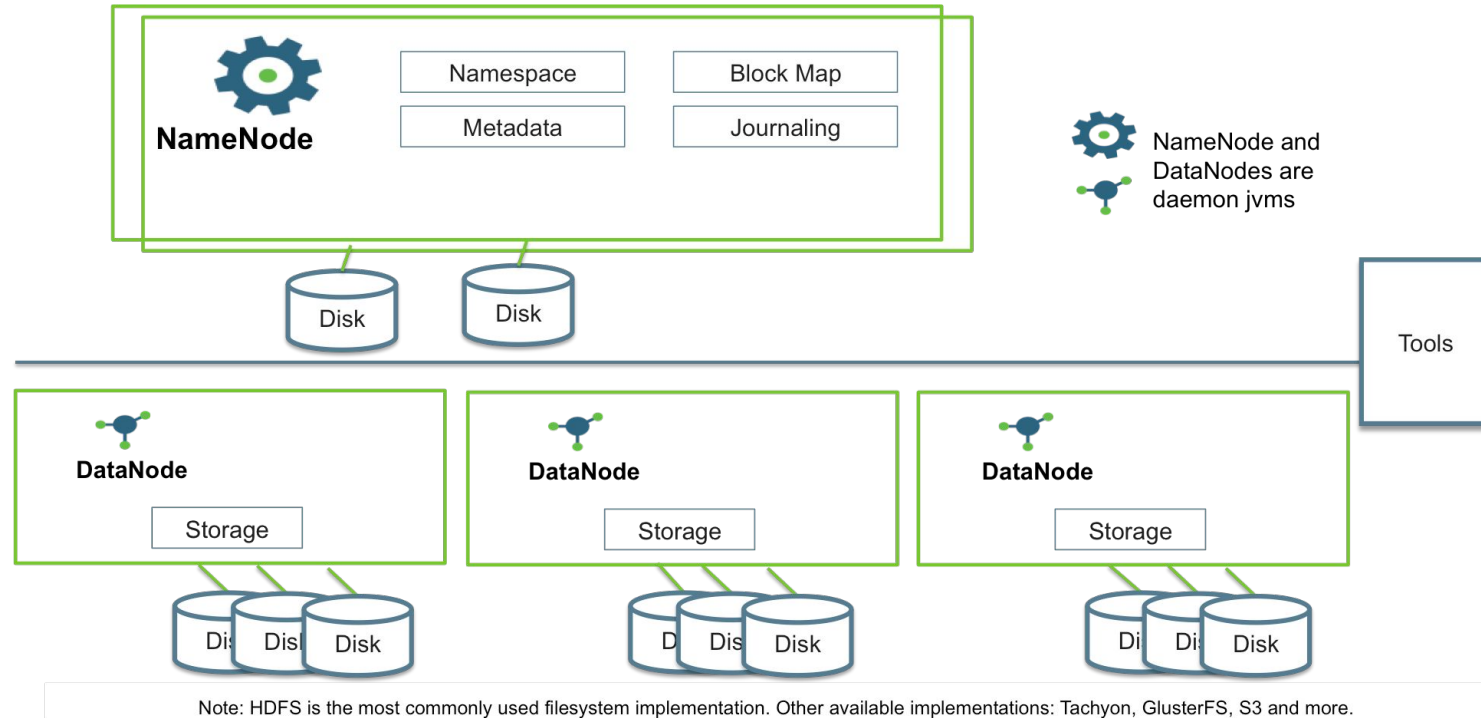
What is HDFS?

- Distributed
- Highly fault tolerant
- Runs on low-cost commodity hardware

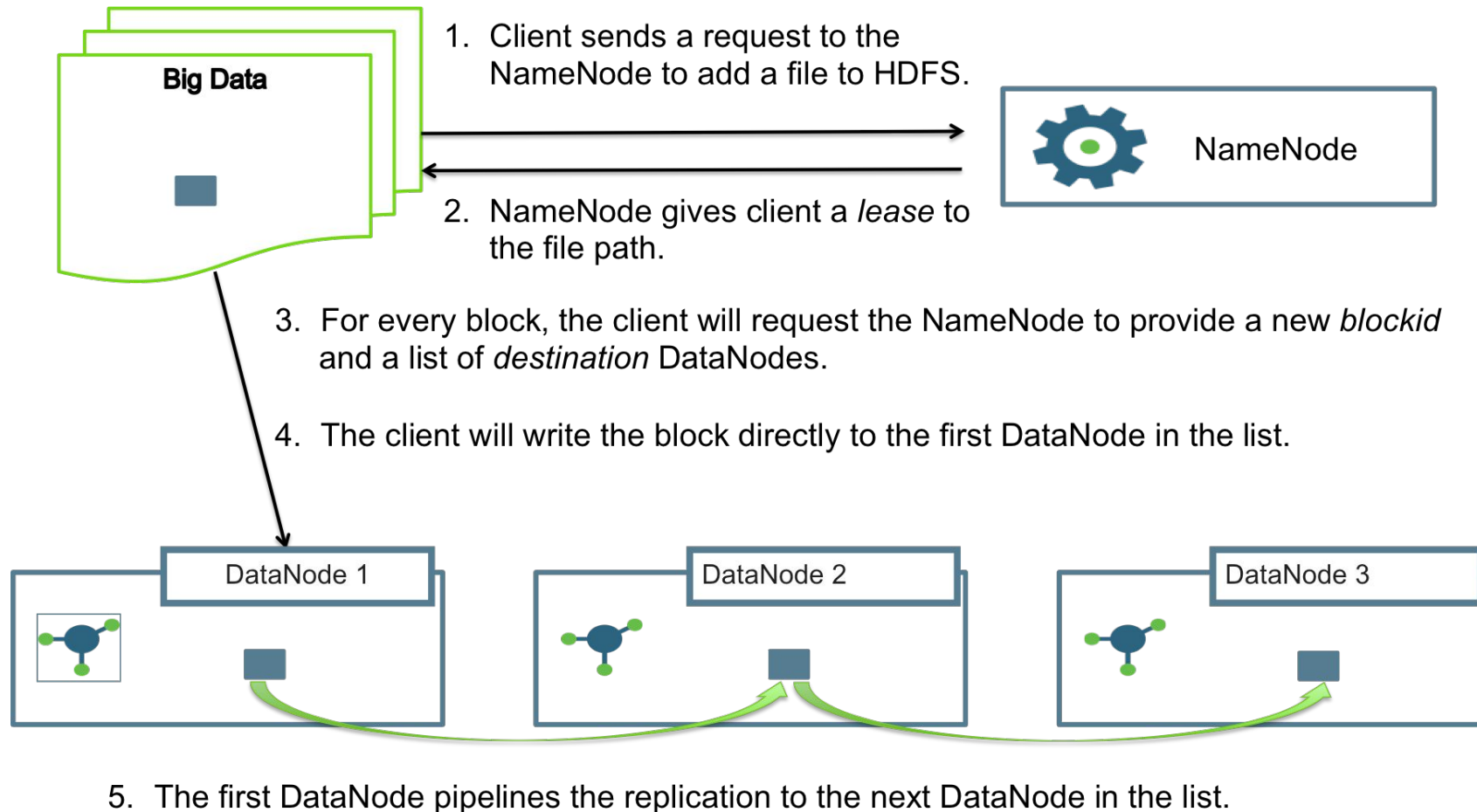
HDFS Architecture



HDFS Process



HDFS Write path

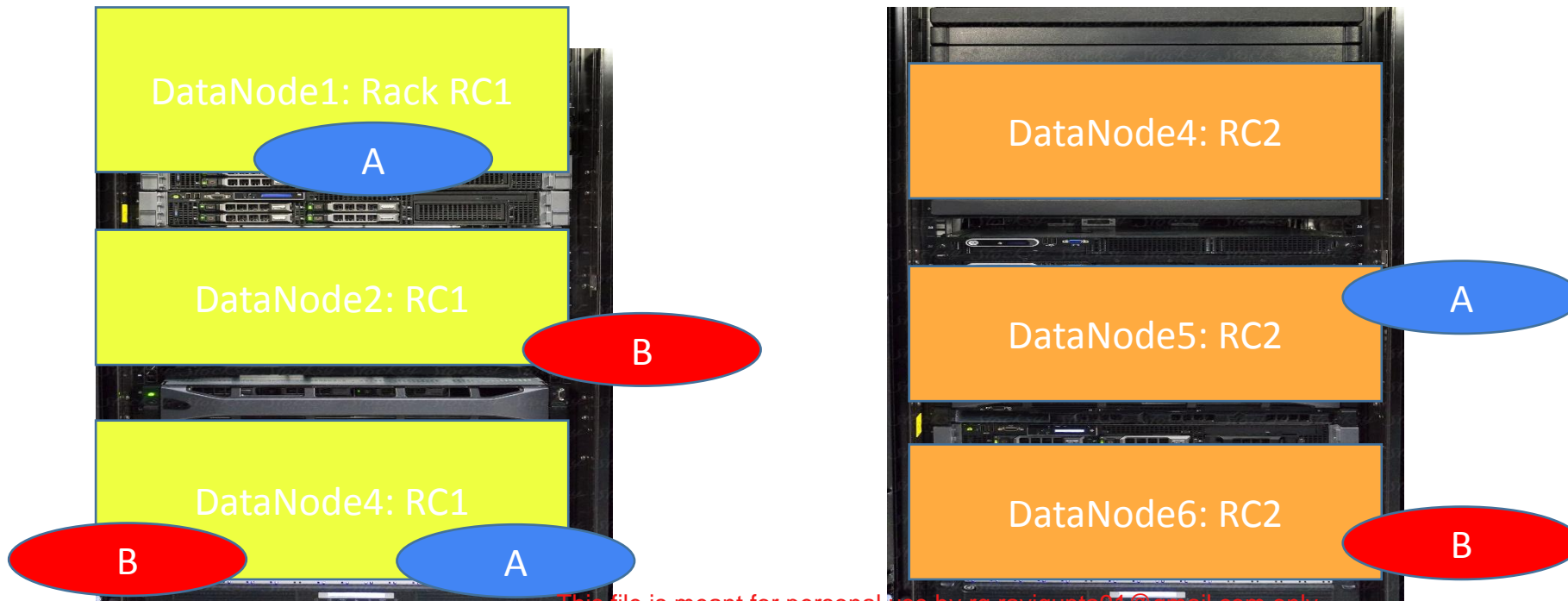


Component Failure & Recovery in HDFS

- NameNode Failure and Recovery
 - If Namenode process goes down, then HDFS metadata will not be available
 - In absence of metadata, client will be unable to read HDFS data
 - Namenode High availability(HA) configuration is designed to handle this
 - In Namenode HA configuration , one of the Namenode will be configured as standby
 - For HA configuration zookeeper is required
 - Zookeeper permots standby Namenode to Active Namenode if Namenode is not available
- DataNode Failure and Recovery
 - If DataNode process goes down, then some of HDFS blocks will not be available
 - If replication factor is set to 3 then 2 more datanodes will have same block stored
 - But what if all the 3 datanodes goes down in one shot?
 - Rack awareness, if configured will help in this scenario

Rack Awareness

- Once Rack awareness is configured then Namenode will make sure block gets replicated across the racks
- In below image, two copies of block A and B are stored on rack RC1 and one copy on rack RC2



What is YARN

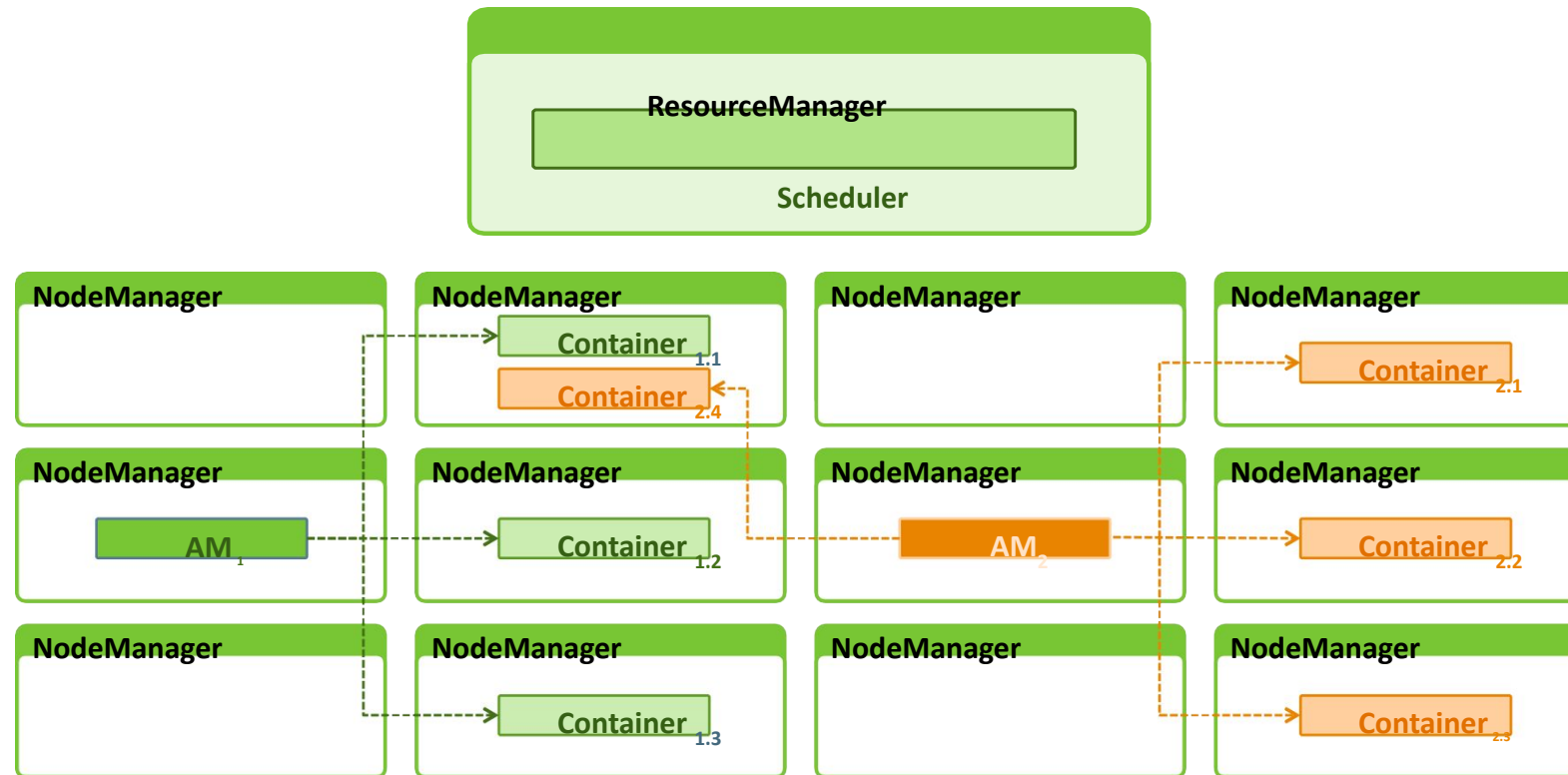


- YARN allocates resources to run any application inside Hadoop cluster
- Its full form is Yet Another Resource Negotiator
- It is an Operating System of Hadoop Cluster
- Part of core Hadoop
- Multi-node cluster has an aggregate pool of compute resources
- CPU and memory like resources that an operating system would manage






YARN Bird's Eye View

- ResourceManager (master)
 - Application management
 - Scheduling
 - Security
- NodeManager (worker)
 - Provides resources to lease
 - Memory, cpu
- Container is unit of processing
- ApplicationMaster manages jobs
- JobHistory Server/Application Timeline Server
 - Job history preserved in HDFS

YARN Architecture

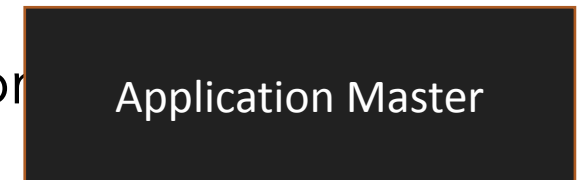
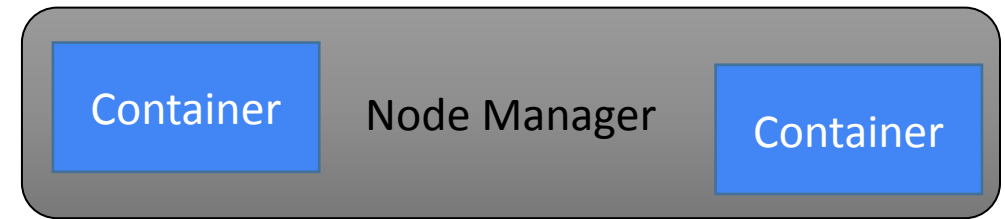
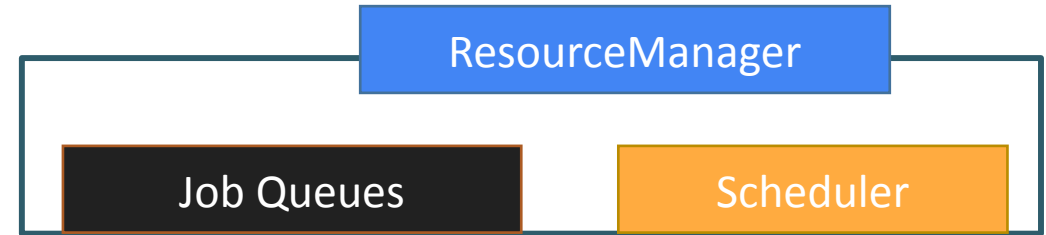


5 Key Benefits of YARN

- Scale 
- Improved Programming models and Services 
- Agility 
- Improved Cluster Utilization 
- Beyond JAVA 

Components of YARN

- ResourceManager
 - Global resource scheduler
 - Hierarchical queues
- NodeManager
 - Per-machine agent
 - Manages the life-cycle of container
 - Container resource monitoring
- ApplicationMaster
 - Per-application
 - Manages application scheduling and task execution
 - E.g. MapReduce Application Master



NodeManager

- Daemon process on same machine as DataNode process
- Container management
- Driven by authorized ApplicationMaster requests
- Start/stop/cleanup containers
- Container sizes are determined by ApplicationMasters
- Node-level resource management
- Memory and CPU as resources

Resource Manager

- ApplicationsManager
- ApplicationMaster launch
- Publishing resource capabilities to ApplicationMaster
- Scheduling
- Capacity Scheduler (default)
- Hierarchical
- Liveliness Monitoring
- NodeManager & ApplicationMasters
- Expects a heartbeat from a NodeManager within 10 minutes
- Heartbeats sent by NodeManagers every 1 second by default
- Expects a heartbeat from an ApplicationMaster within 10 minutes

Application Master

- There will single Application Master per YARN Application
- App Master will be parent of all the containers of the application(Job)
- App Master will be initiated by Resource Manager
- Every App Master will have unique application Id
- App Master runs on the Data Node machine

Component Failure & Recovery in YARN (1/2)

- Container Failure
 - Exceptions are propagated back to the ApplicationMaster
 - ApplicationMaster and NodeManager are responsible for restart or reschedule
 - ApplicationMaster will request new container(s) from ResourceManager
- ApplicationMaster Failure
 - It can recover the state of the running containers and completed containers from the failed ApplicationMaster
 - ResourceManager will start a new instance of the master running in a new container managed by a NodeManager

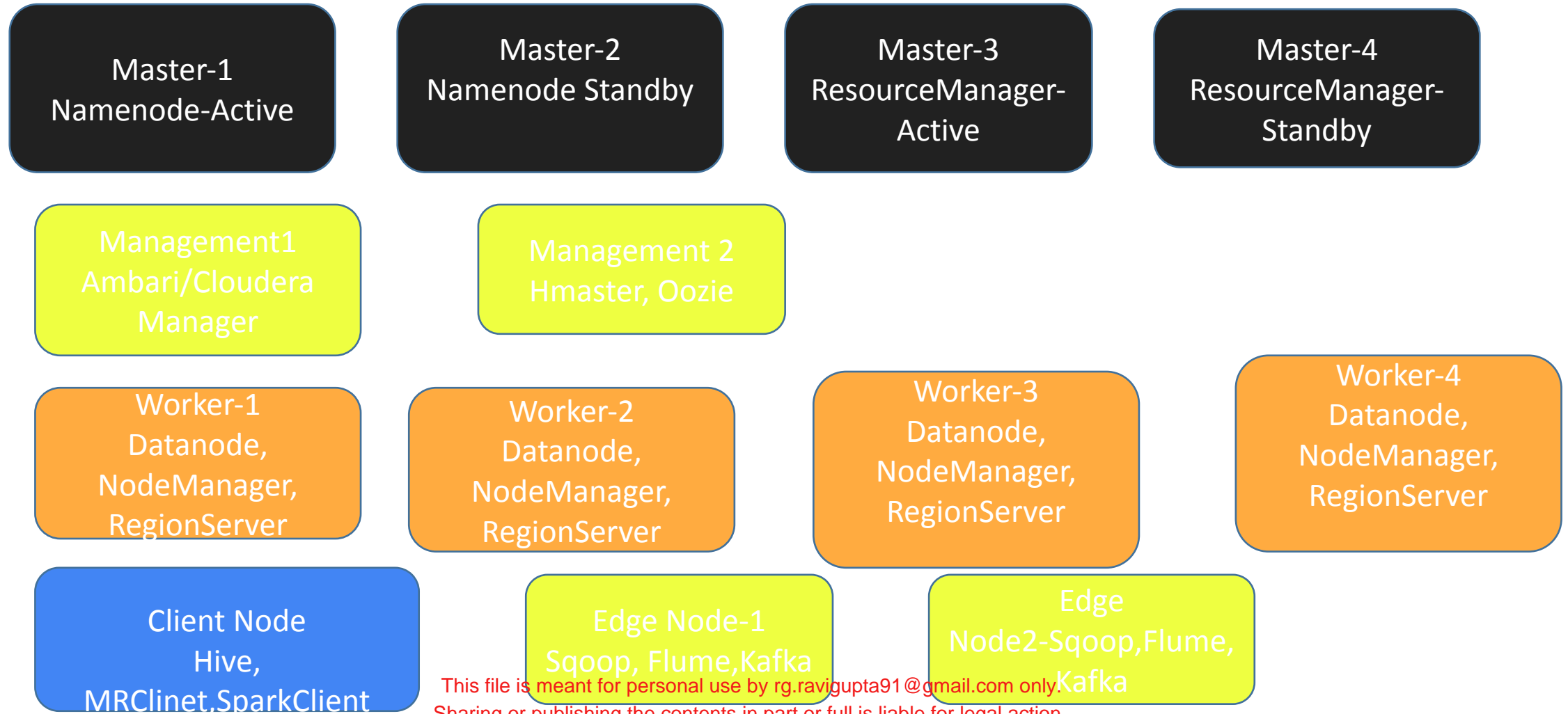
Component Failure Recovery in YARN(2/2)

- NodeManager Failure
 - Removed from ResourceManager's pool of available nodes
 - ApplicationMaster and containers running on the failed NodeManager can be recovered as described above
 - ApplicationMasters can blacklist NodeManagers
- ResourceManager Failure
 - Failover to standby ResourceManager or manually brought up by an administrator
 - Recovers from saved state stored by ZooKeeper
 - Cannot recover resource requests

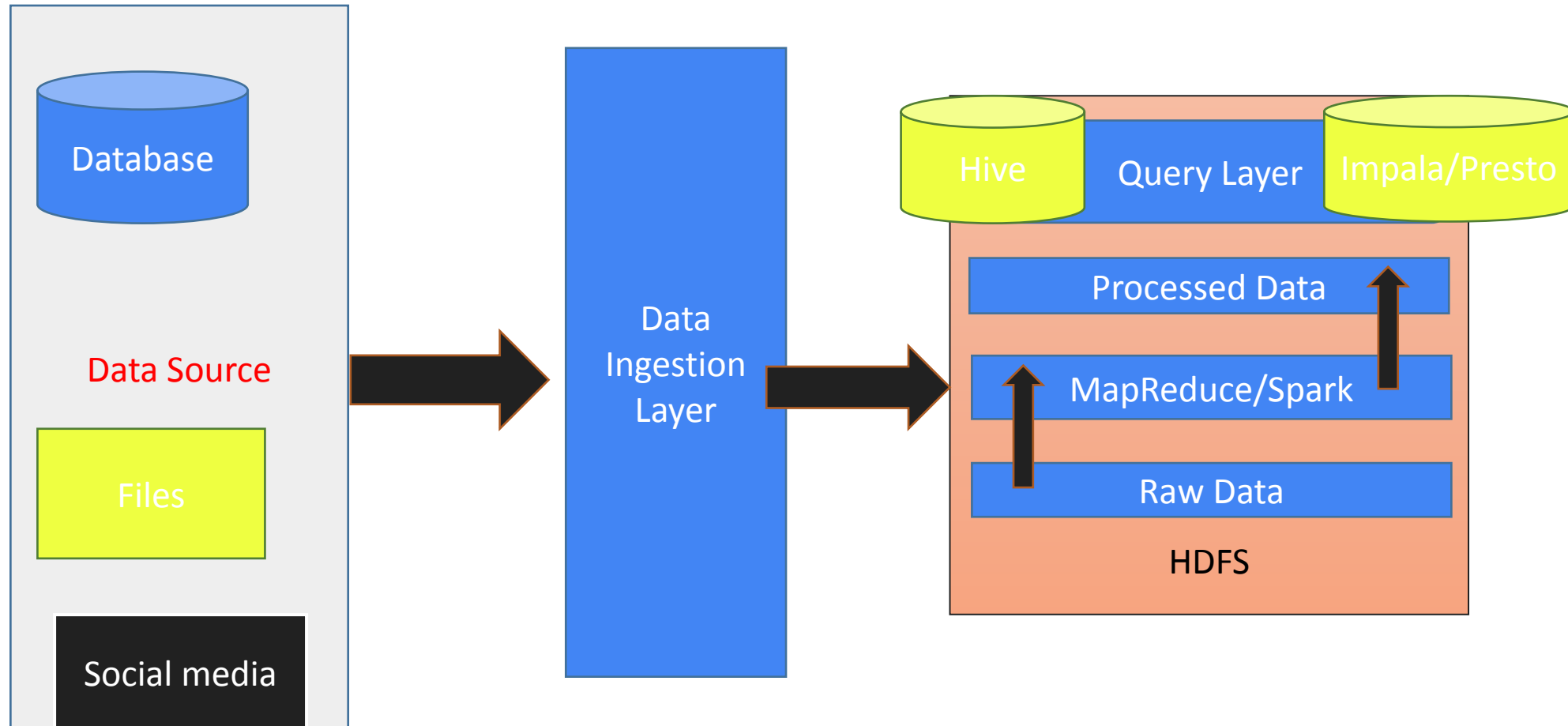
Types of nodes in Hadoop Cluster

- Master Nodes:
 - On these nodes master components of HDFS and YARN are installed
 - Namenode and Resource Manager process are master nodes
- Management Nodes:
 - On these nodes, mostly management software are installed
 - Ambari/Cloudera manager , WebCat Server, Zookeeper and JobHistory Server
- Workers Nodes:
 - On workers nodes, Datanode, NodeManager and other process runs
- Client Nodes:
 - Software such as hive, Spark client , MRClient are installed
- Edge Nodes:
 - Different data ingestion tools such as Flume, Sqoop and Kafka are installed

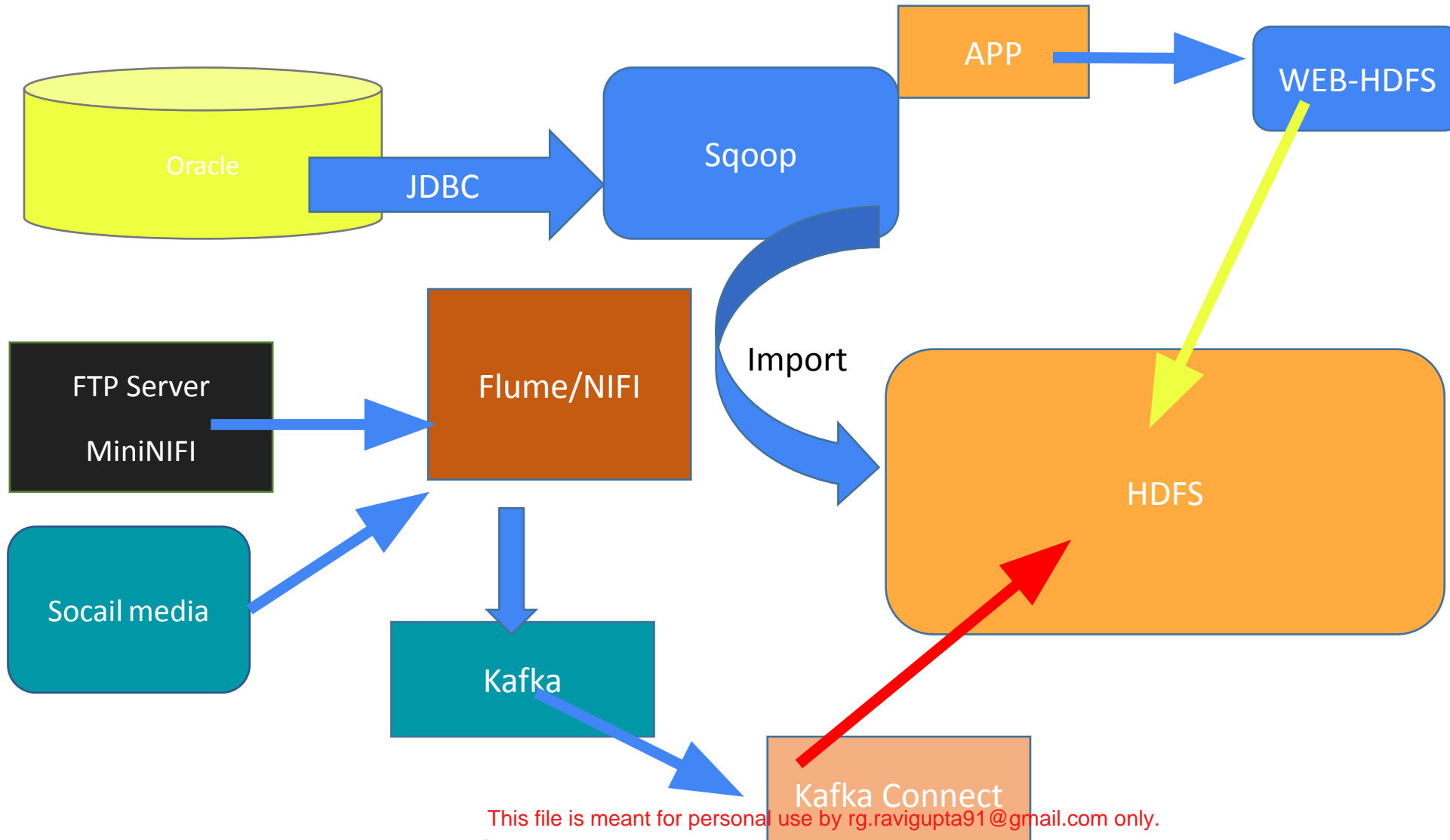
Hadoop Cluster Layout



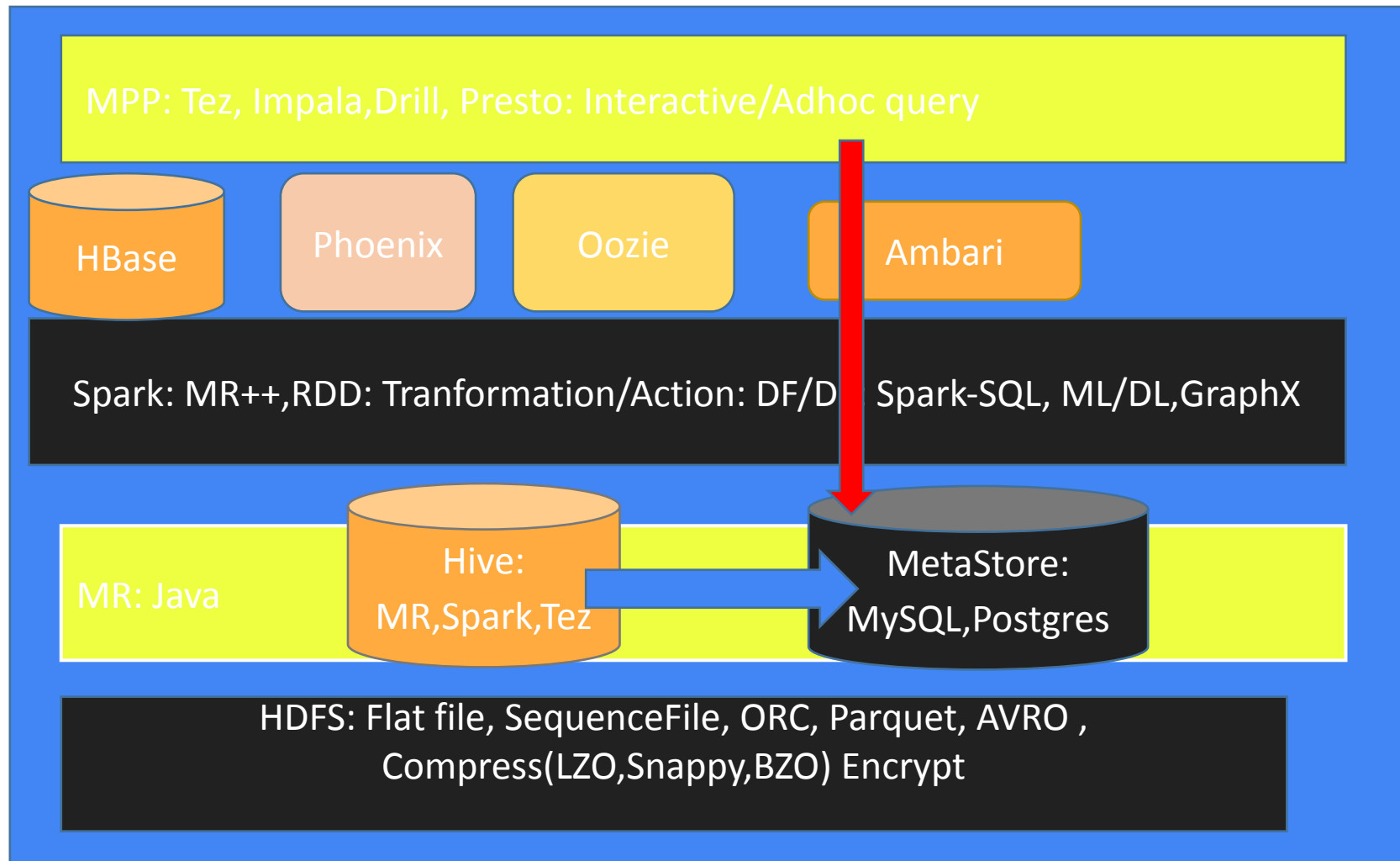
Hadoop As Data Lake



Data Ingestion Layer



Inside Hadoop Platform



Roles of Hadoop Ecosystem Components

- **Storage**
 - HDFS which is file system of Hadoop
 - Hbase , a NoSQL database of Hadoop
 - Kafka, a streaming platform for storing messages
- **Processing**
 - MapReduce
 - Apache Spark
- **MPP Query engine**
 - Impala
 - Presto
 - Tez
- **Data Ingestion**
 - Flume
 - Sqoop
 - NIFI

Data Ingestion : Apache Sqoop



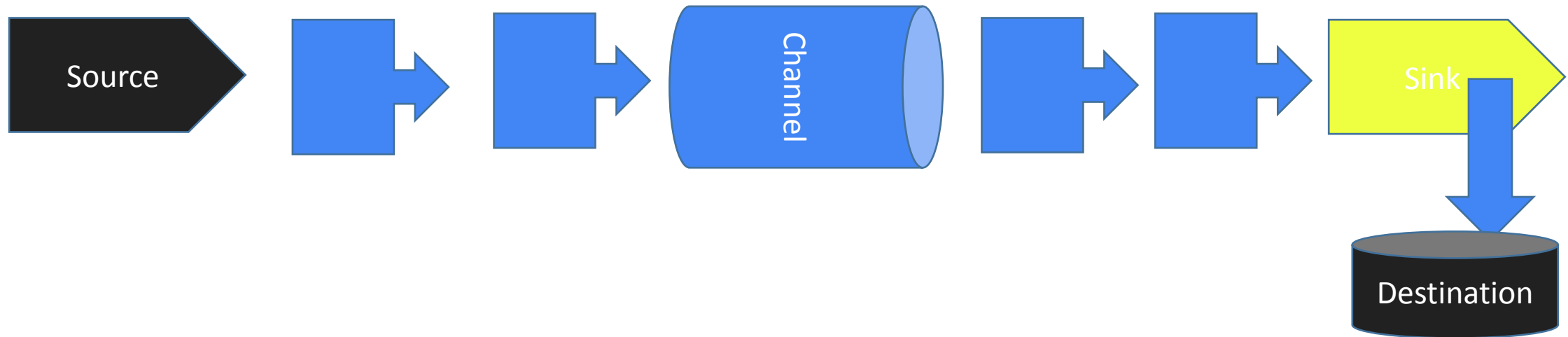
- Apache SQOOP is a tool designed to support bulk export and import of data into HDFS from RDBMS, enterprise data warehouses, and NoSQL systems.
- Sqoop has connectors for working with a range of popular relational databases, including MySQL, PostgreSQL, Oracle, SQL Server, and DB2.
- Each of these connectors knows how to interact with its associated RDBMS

Sqoop Features

- Import a single table
- Import the complete database
- Import selected tables
- Import selected columns from a particular table
- Filter out certain rows from certain tables, etc by using the query option
- Sqoop uses MapReduce(map only) job to fetch data from an RDBMS and stores that on HDFS.

Data Ingestion : Apache Flume

- Apache Flume is a data ingestion tool to ingest data from remote servers
- It collects , aggregate and direct stream into HDFS
- Flume works with different data sources to process and send data to defined destinations including HDFS



Hadoop 3.0 new features

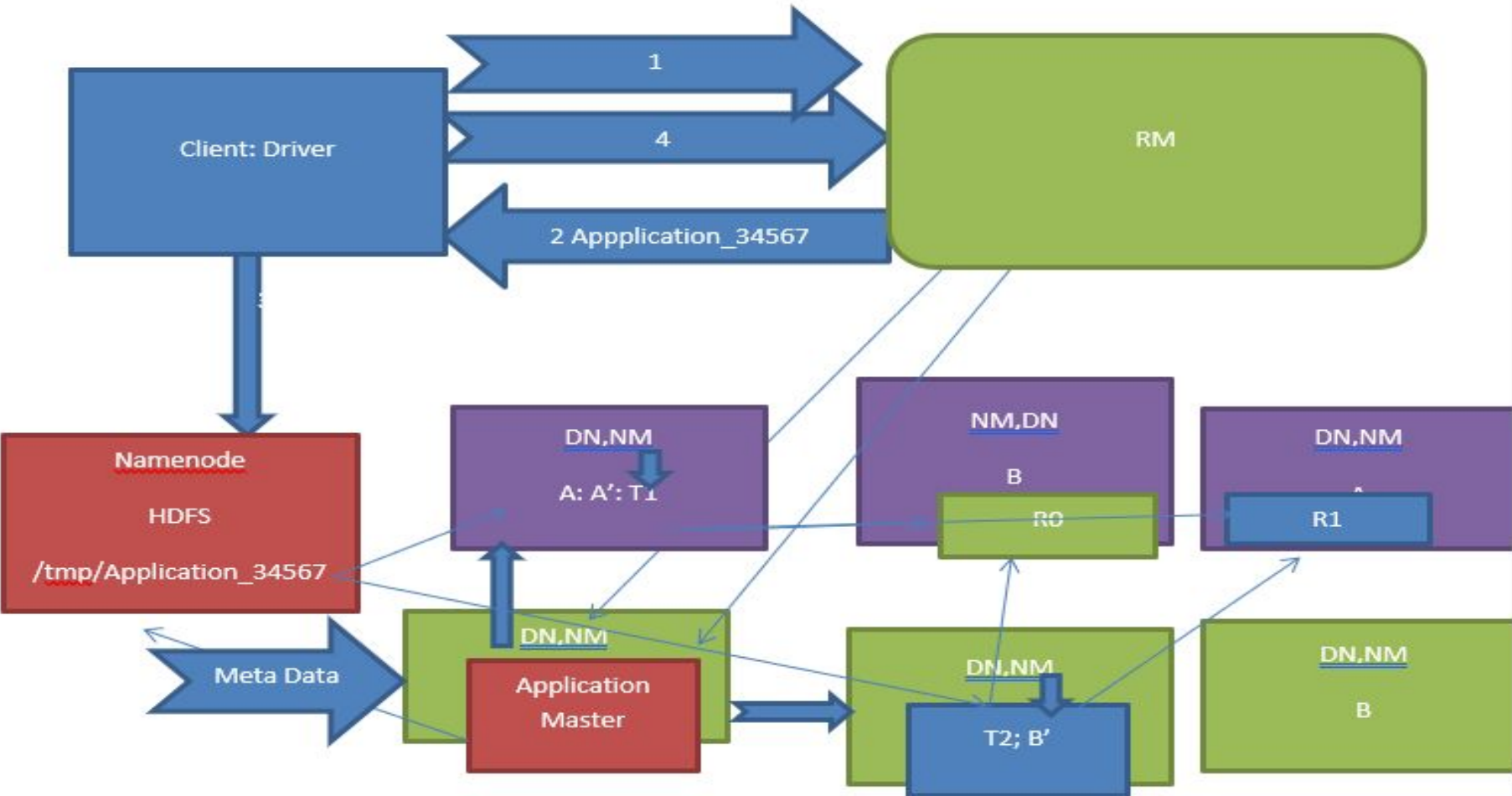
- More Than Two NameNodes Supported
- Support for erasure encoding in HDFS
- Support for Opportunistic Containers and Distributed Scheduling
- Intra-datanode balancer
- Filesystem Connector Support
- JDK 8.0 is the Minimum JAVA Version Supported by Hadoop 3.x



Let us perform some hands-on

- ✓ Understanding HDFS commands
- ✓ Executing MR Job
- ✓ Verifying the results
- ✓ Exploring MR Job history page

Execution Flow of YARN Application



MR Execution Flow (1/4)

- Submit the request to resource manager by the job object created in the client JVM. Note: Client JVM will be created in the client node of the hadoop cluster
- Then application manager of the resource manager will check whether application is a valid YARN application or not , and if it is a valid then generated a unique YARN application id
- Job object inside the Driver process , using this unique id, creates a staging folder `/tmp/application_2731` and copies all the jar, script, .sql and dependencies.
- Job object inside the Driver process, again submit the request to the resource manager

MR Execution Flow (2/4)

- Resource manager selects one of the data node(based on the availability of the resources CPU 1 core, 1 GB RAM) where Application master image will be copied
- Then resource manager will instruct NM of the respective node(server) to start the image . once image is started it will be called as App Master container
- App master wants to break the job into n tasks for this he will connect with the namenode and get metadata(blocks, locations) for the input folder specified while running the job

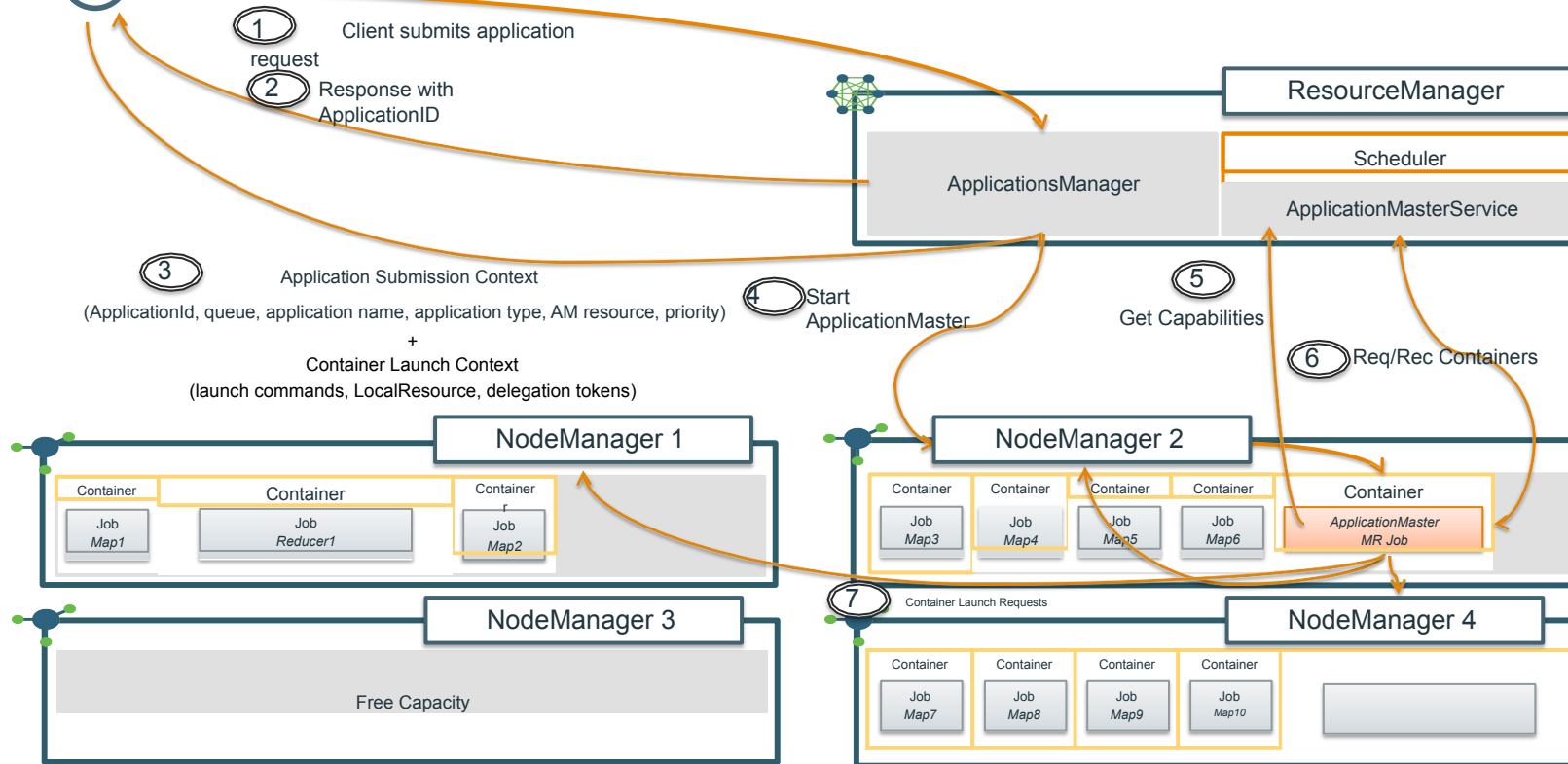
MR Execution Flow (3/4)

- App master in order to create N no of tasks, first it needs to find out number of input split with the help of InputFormatClasses . Input classes consider the data split and make the record logically complete in order to decide the no of splits
- Application master will decide the task assignment and request the resource manager to allocate the containers .
- Once resources are allocated then application master will request the respective NM to start the container using those resources. Once container is started mapper process will start in that container

MR Execution Flow (4/4)

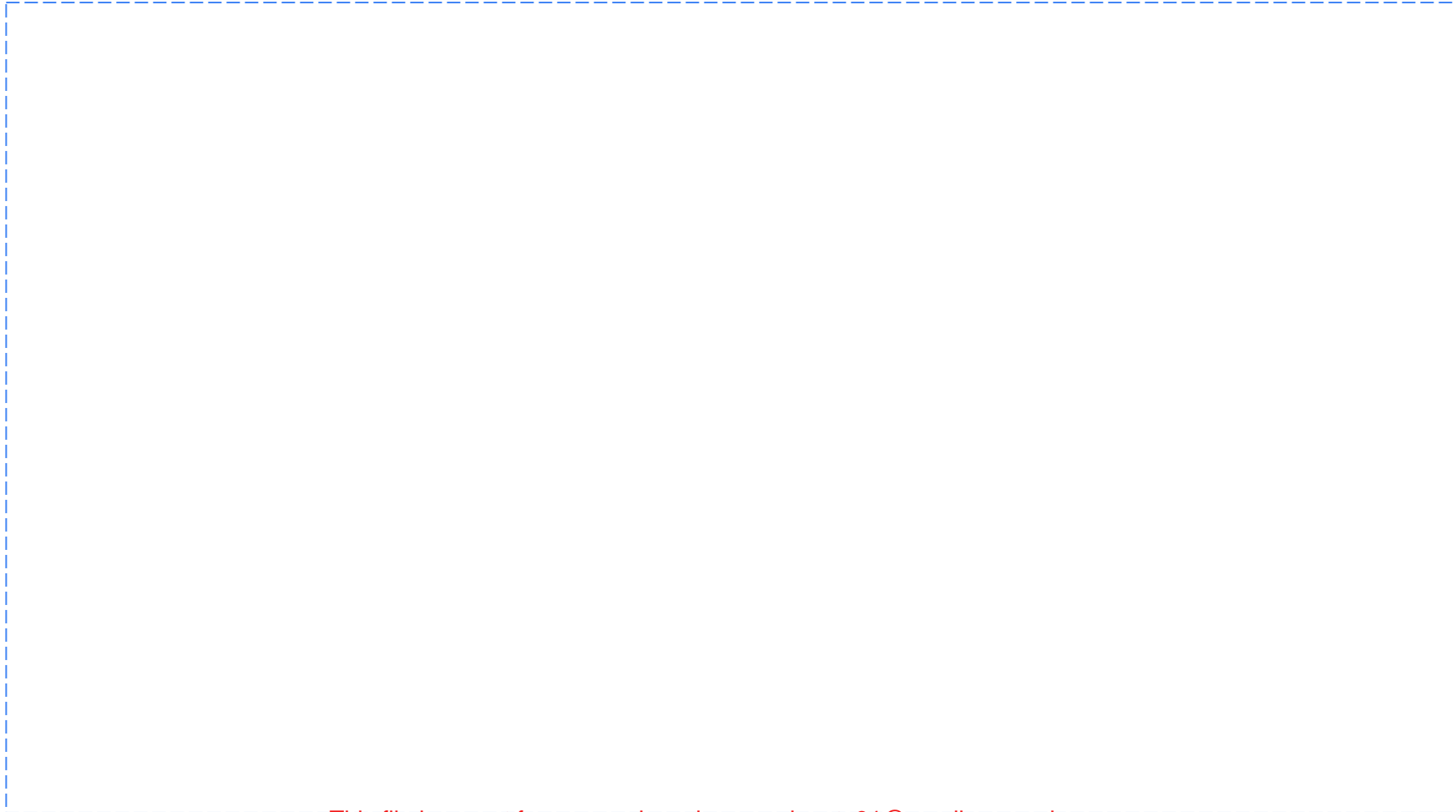
- Mapper task runs map method implemented in the Mapper java class.
- Once mapper is over then sorting shuffling will take place . As part of this phase, Hash partitioning will write the data (key value pair) into n part-m-* files here n depends upon no of reducer set in the application
- These part files will be copied to respective reducers which will perform merge sort to combine multiple values of the given key into list of values
- List of values and key will be fed into reduce method of Reducer class
- Output of the reducer will be stored in HDFS

Lifecycle of a YARN Application



MapReduce Introduction

How does MapReduce work?



How does MapReduce work?

Mapper Phase

InputSplits



Map Output

Reducer Phase

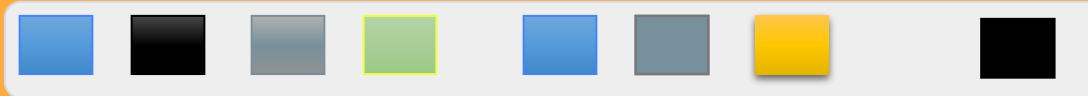
Reduce Tasks

Reduce Output

How does MapReduce work?

Mapper Phase

InputSplits



Mapper
Tasks

Map Output List

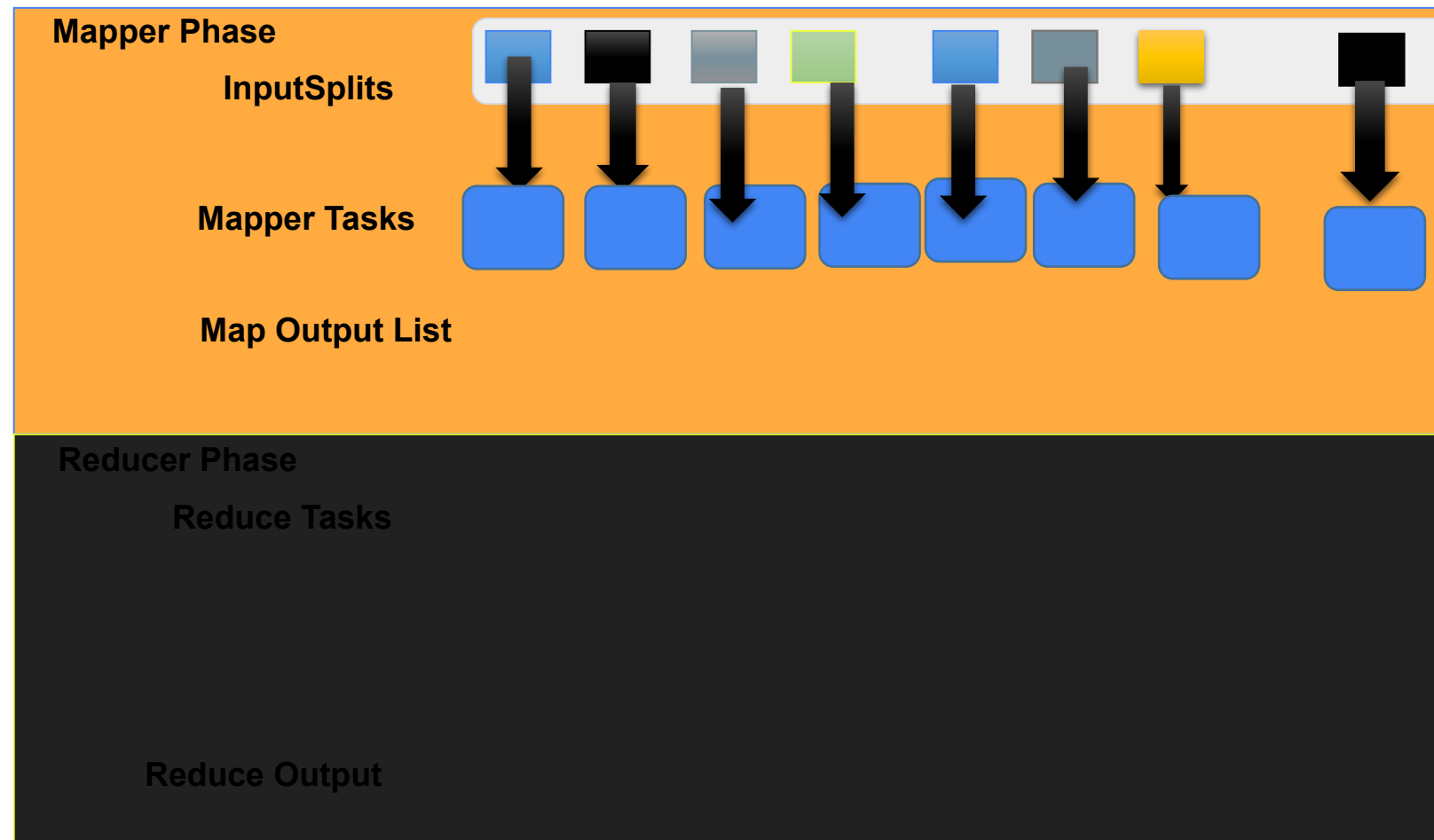
Reducer Phase

Reduce Tasks

Reduce Output

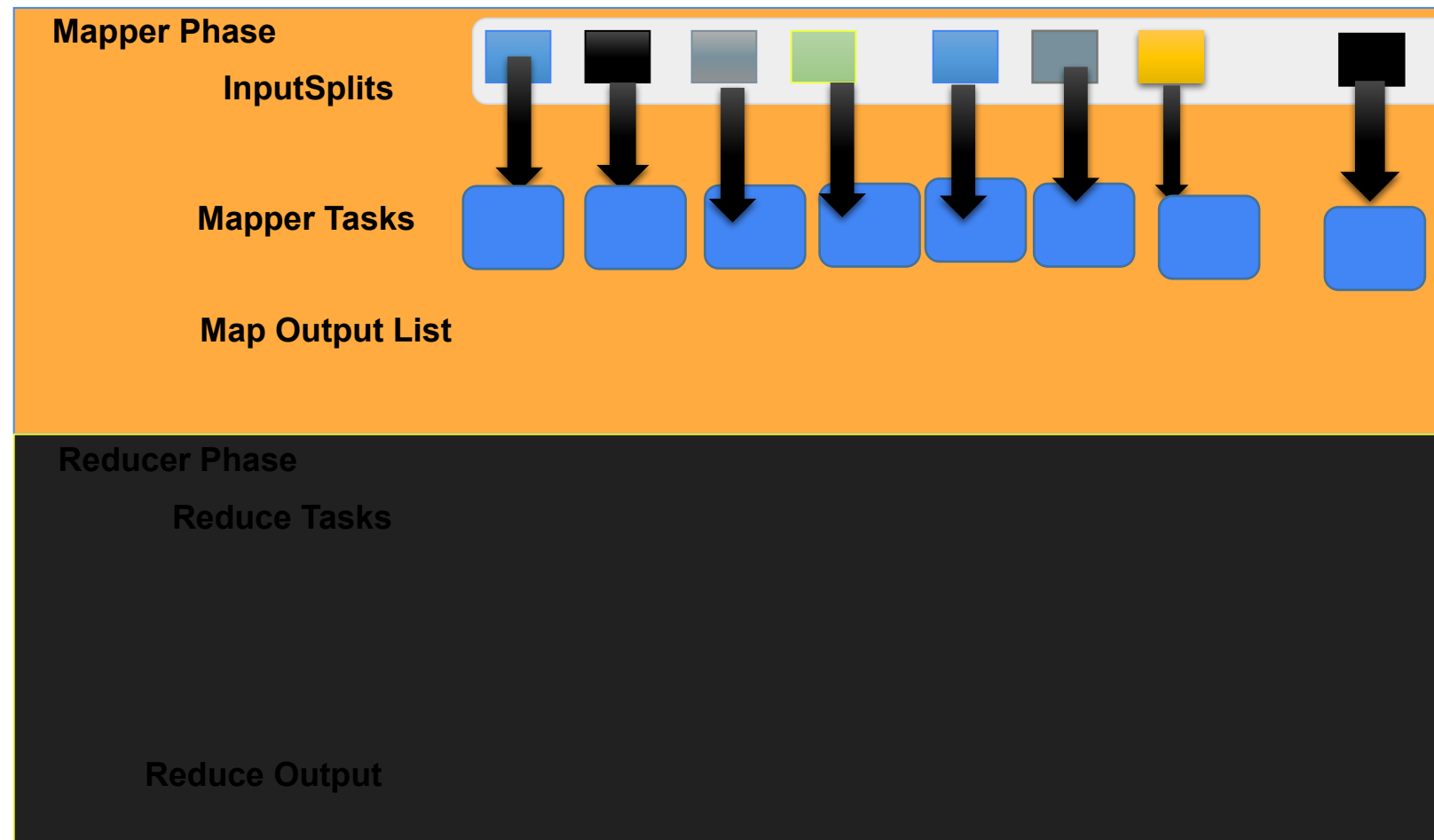
MapReduce

How does MapReduce work?



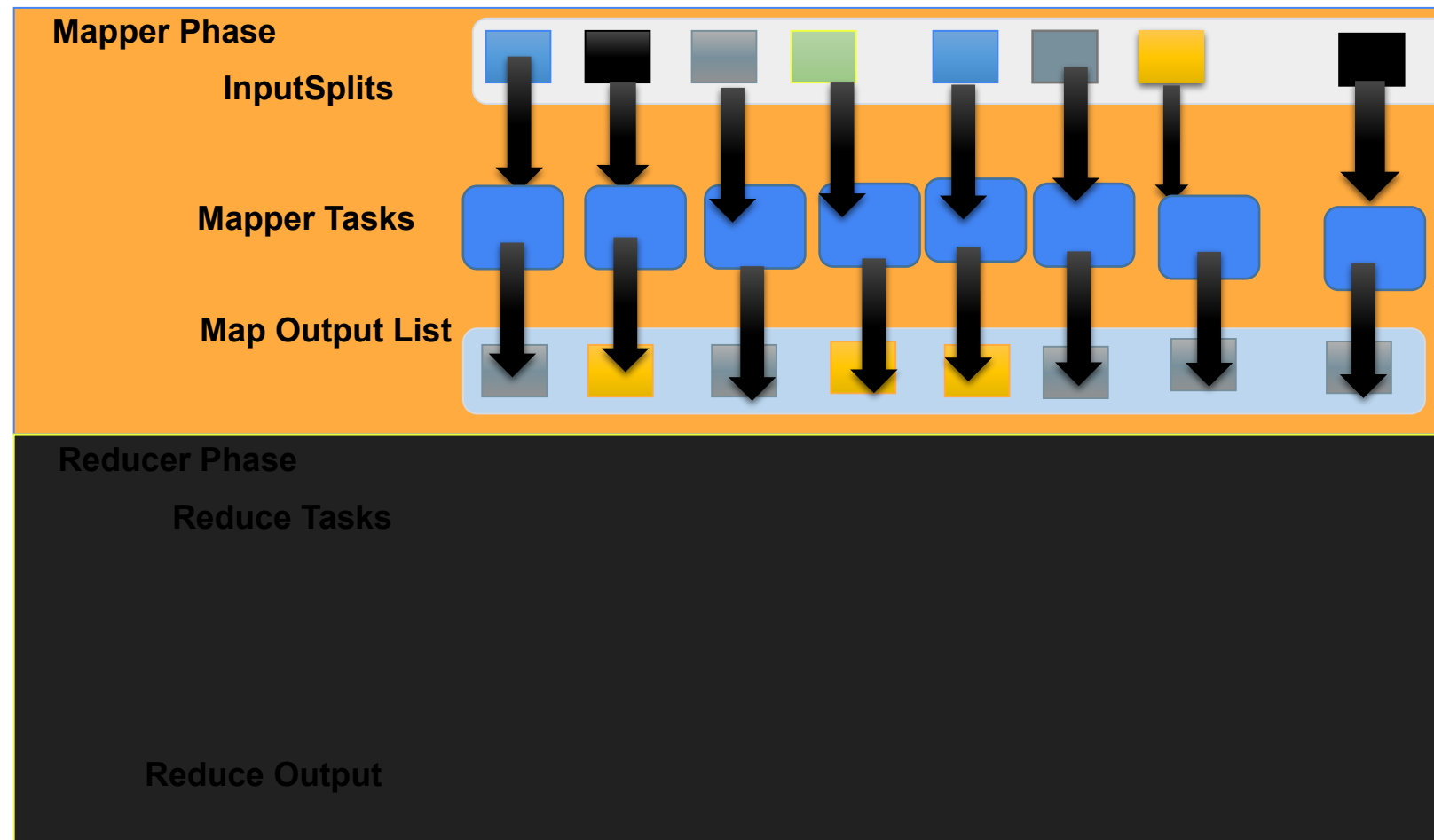
MapReduce

How does MapReduce work?



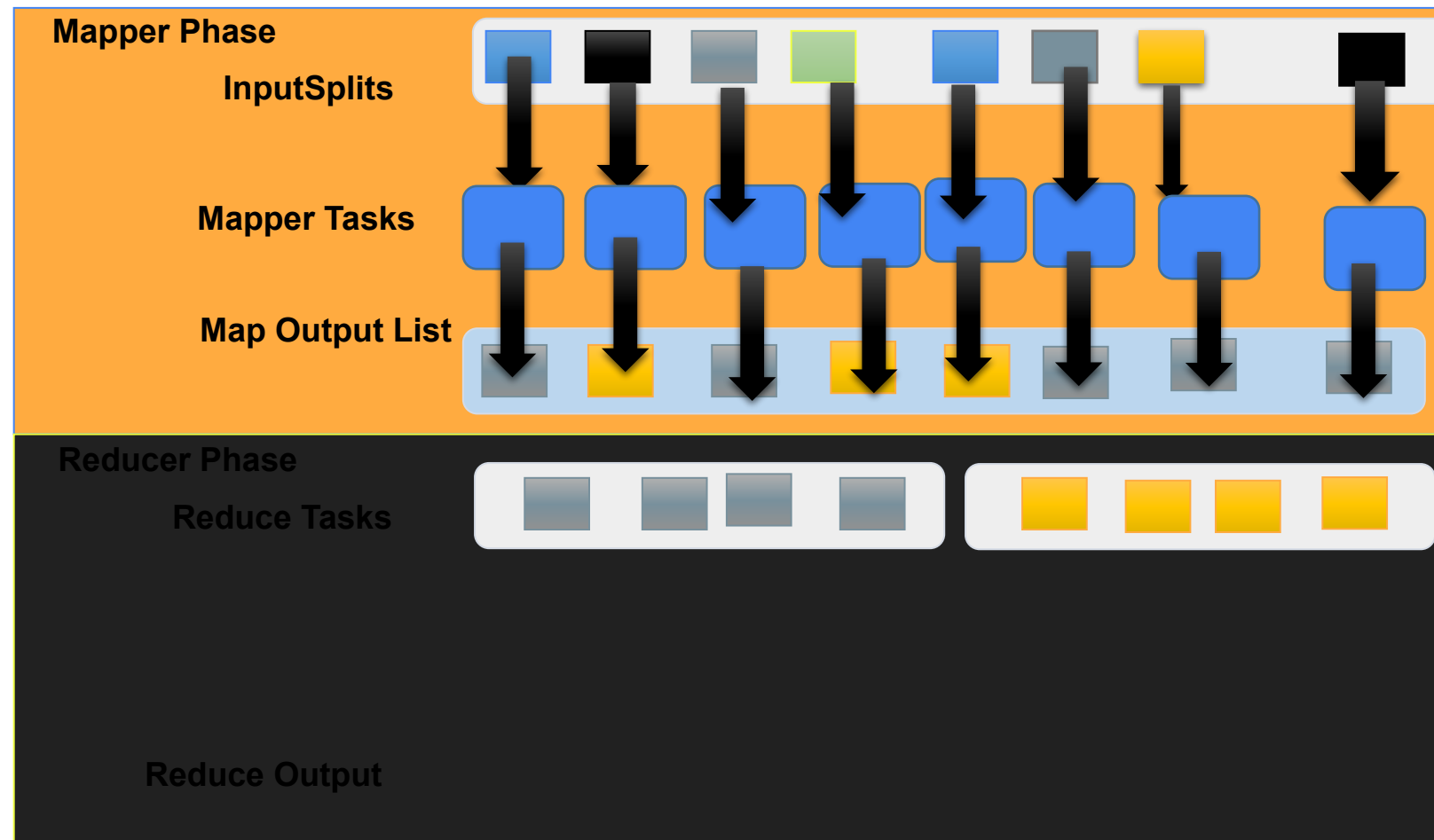
MapReduce

How does MapReduce work?



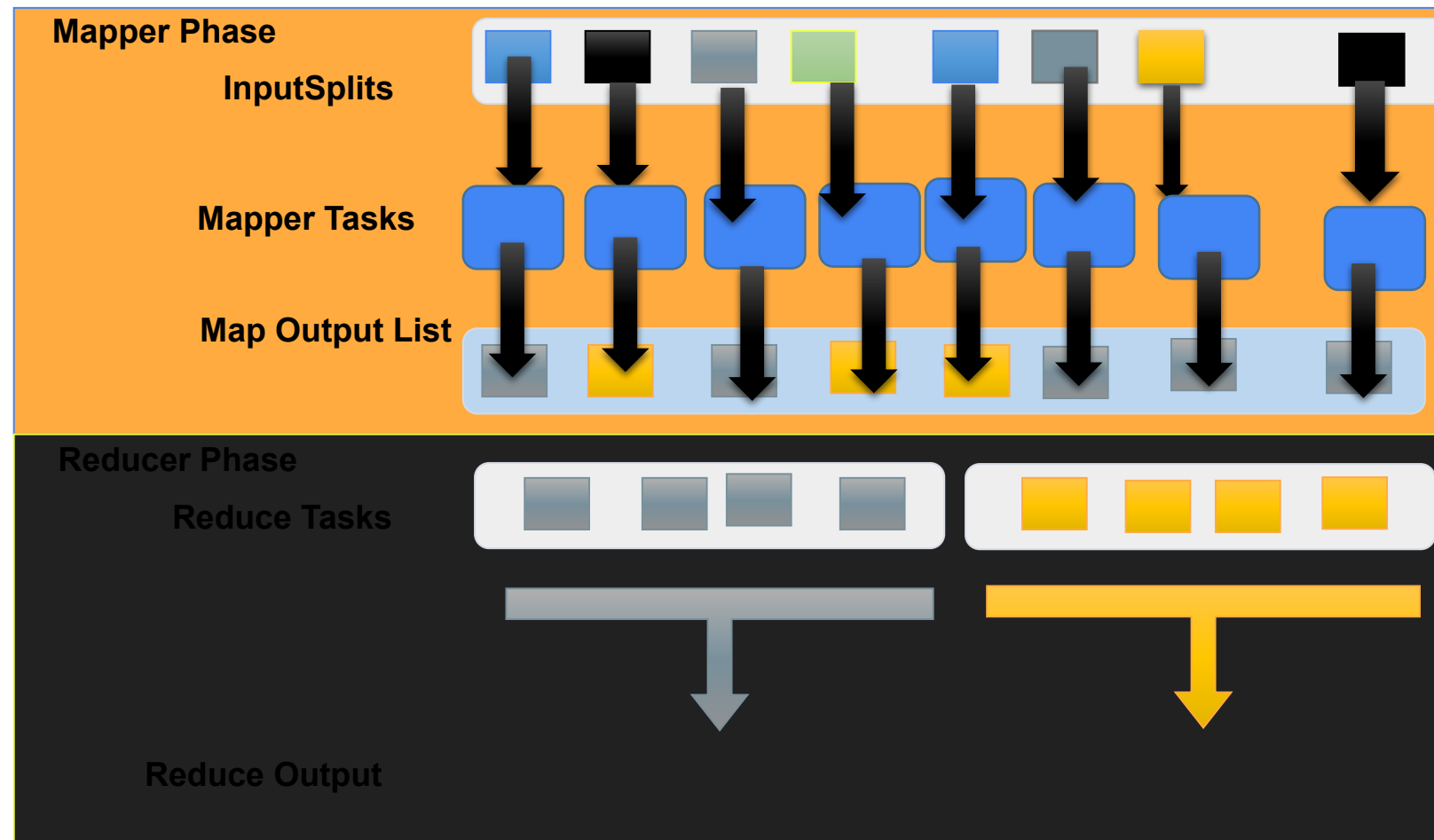
MapReduce

How does MapReduce work?



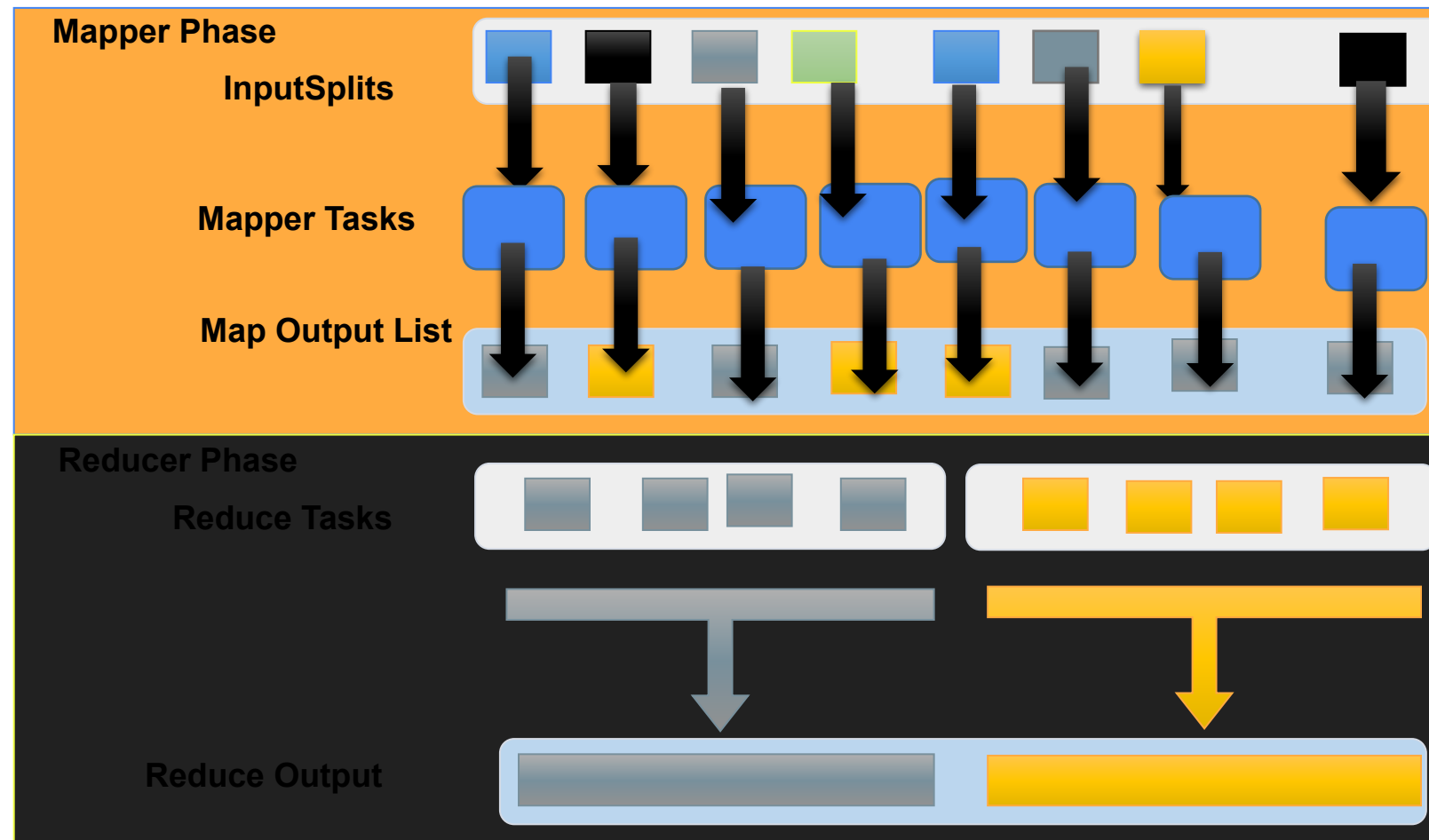
MapReduce

How does MapReduce work?



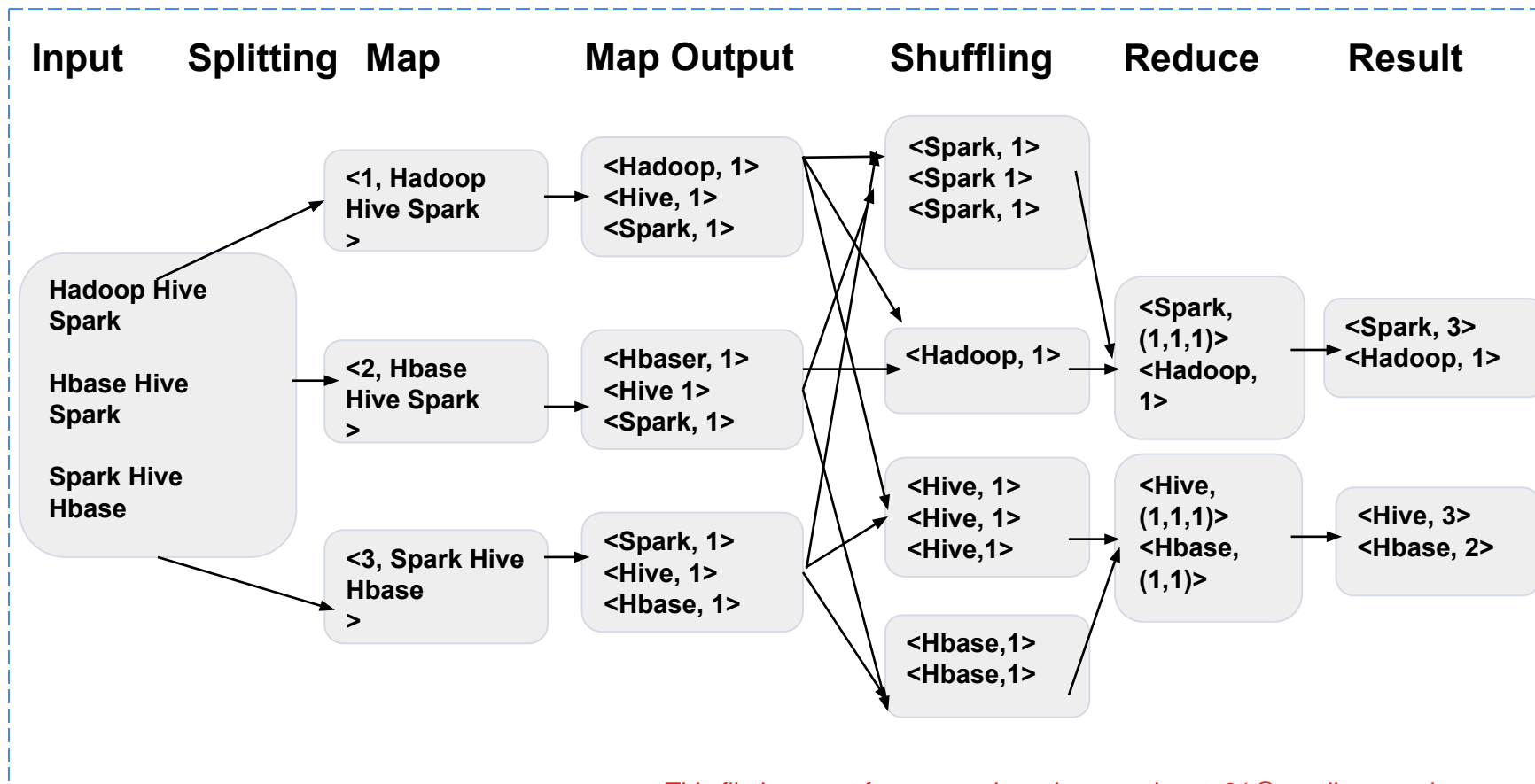
MapReduce

How does MapReduce work?



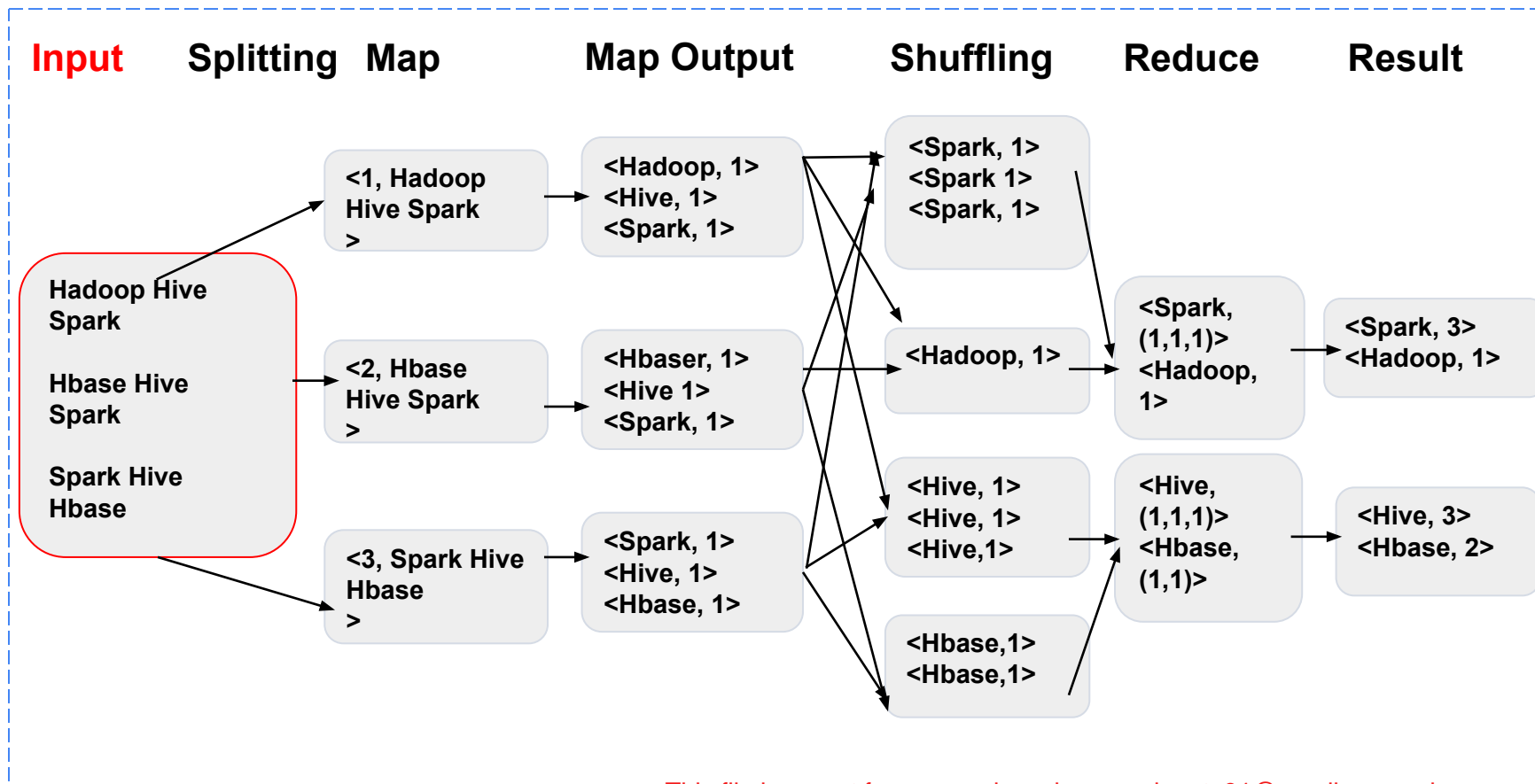
MapReduce

Hadoop MapReduce



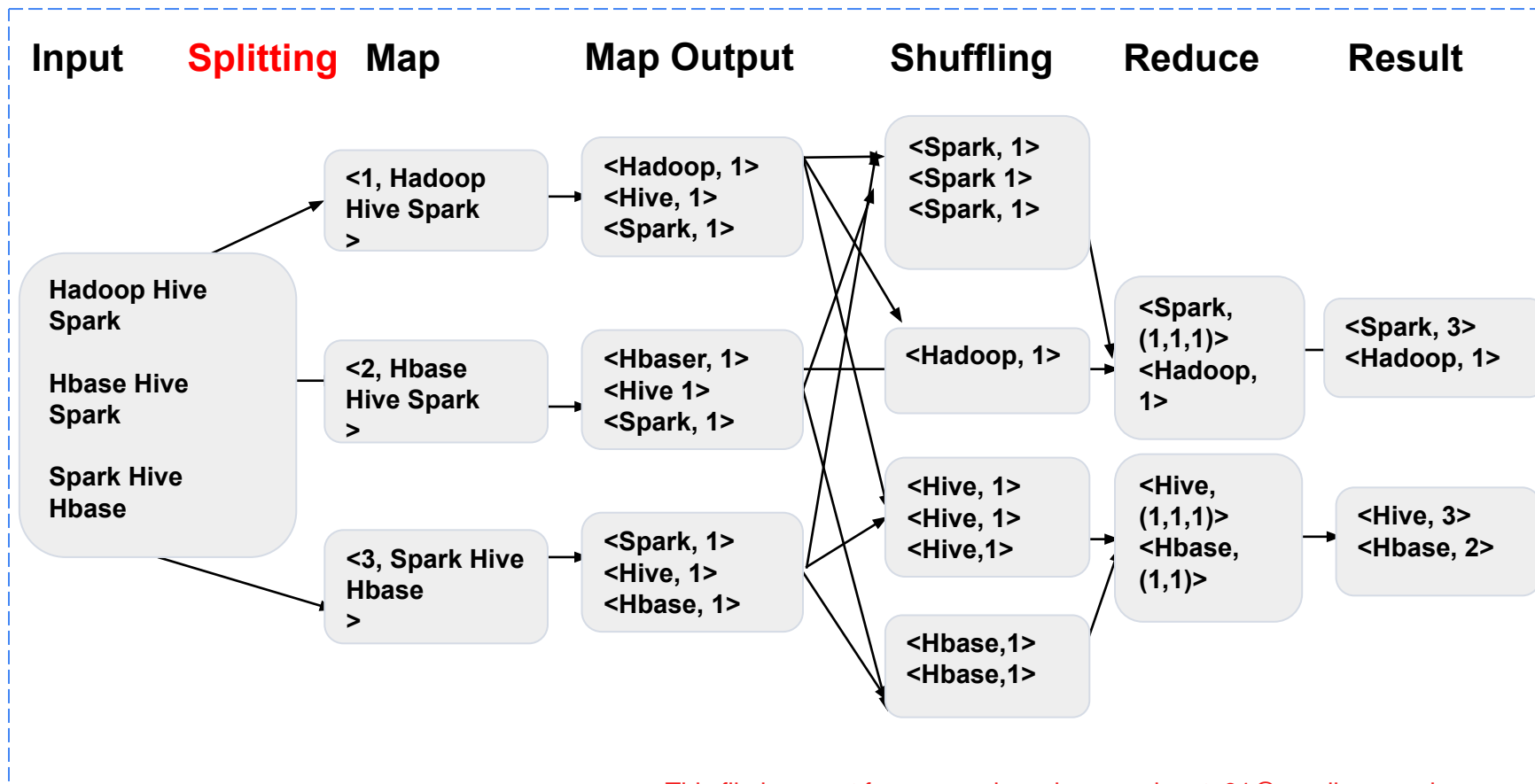
MapReduce

Hadoop MapReduce



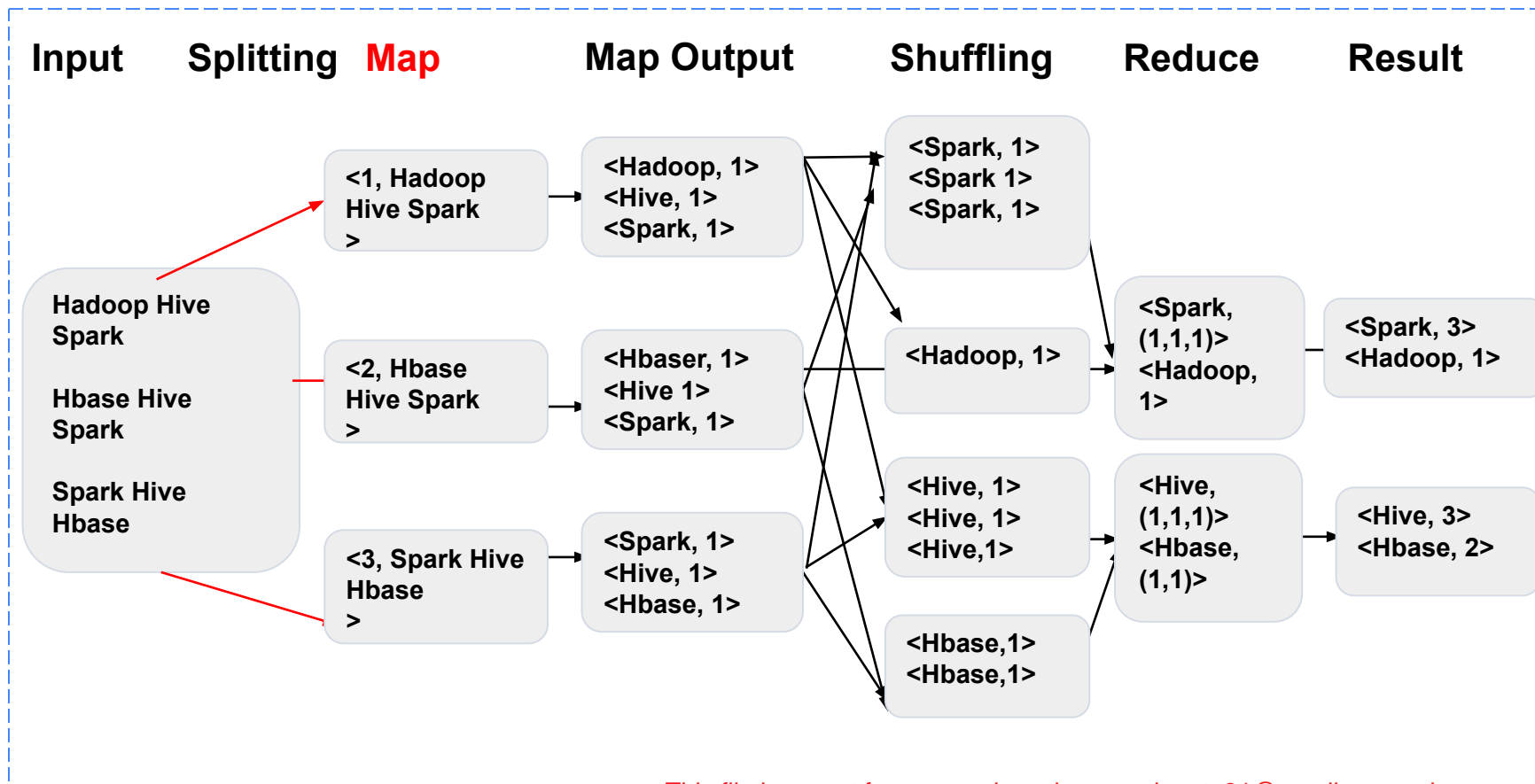
MapReduce

Hadoop MapReduce



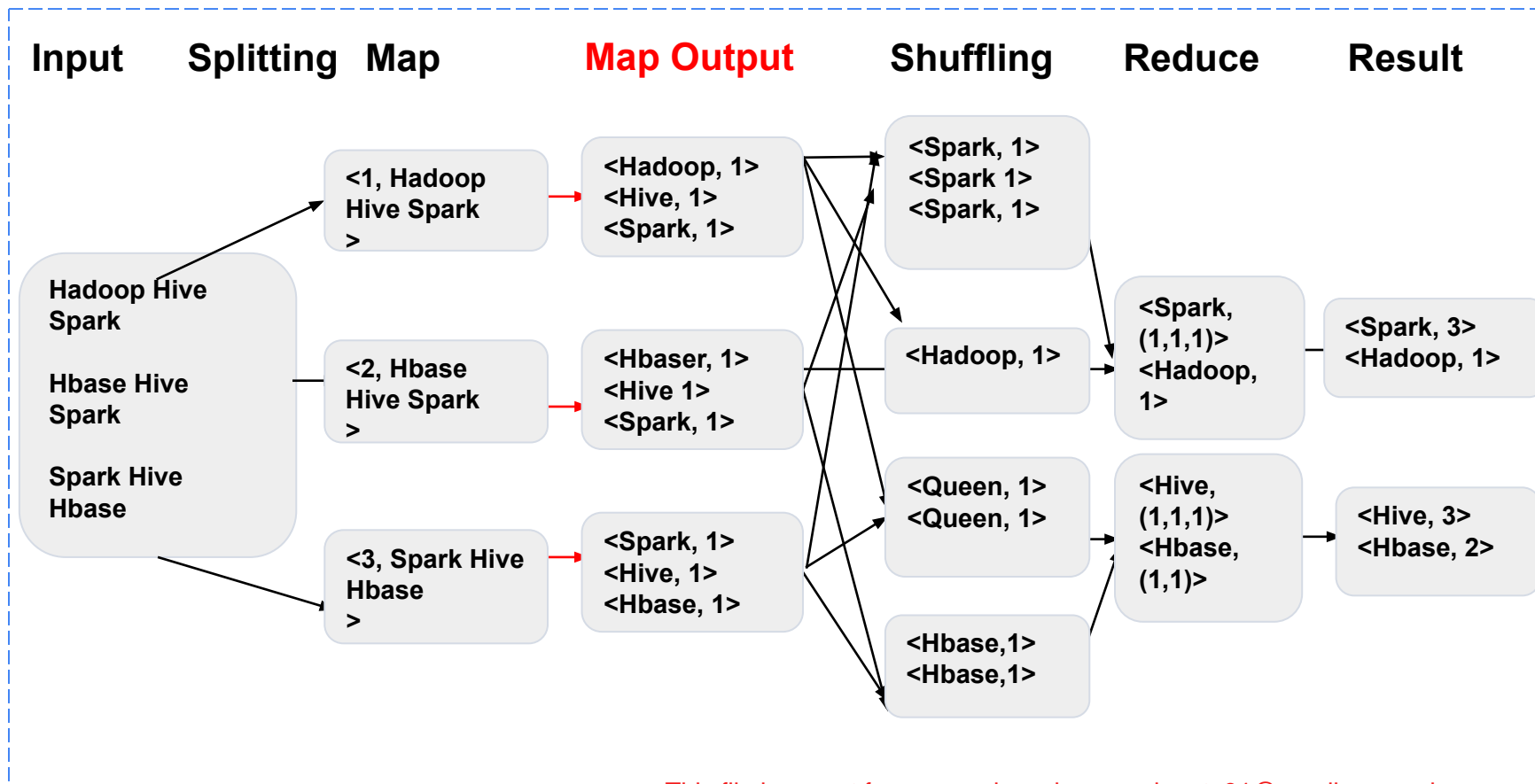
MapReduce

Hadoop MapReduce



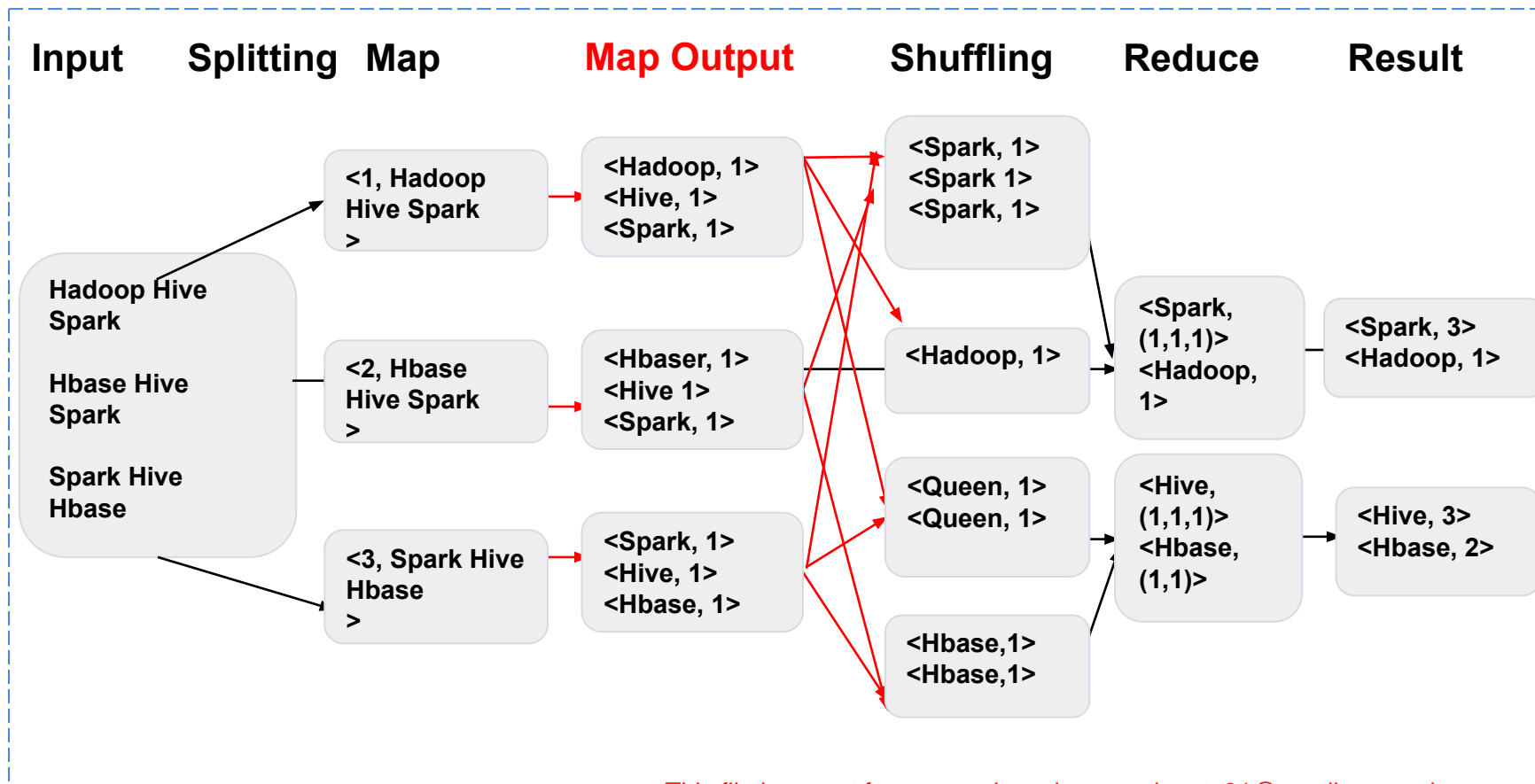
MapReduce

Hadoop MapReduce



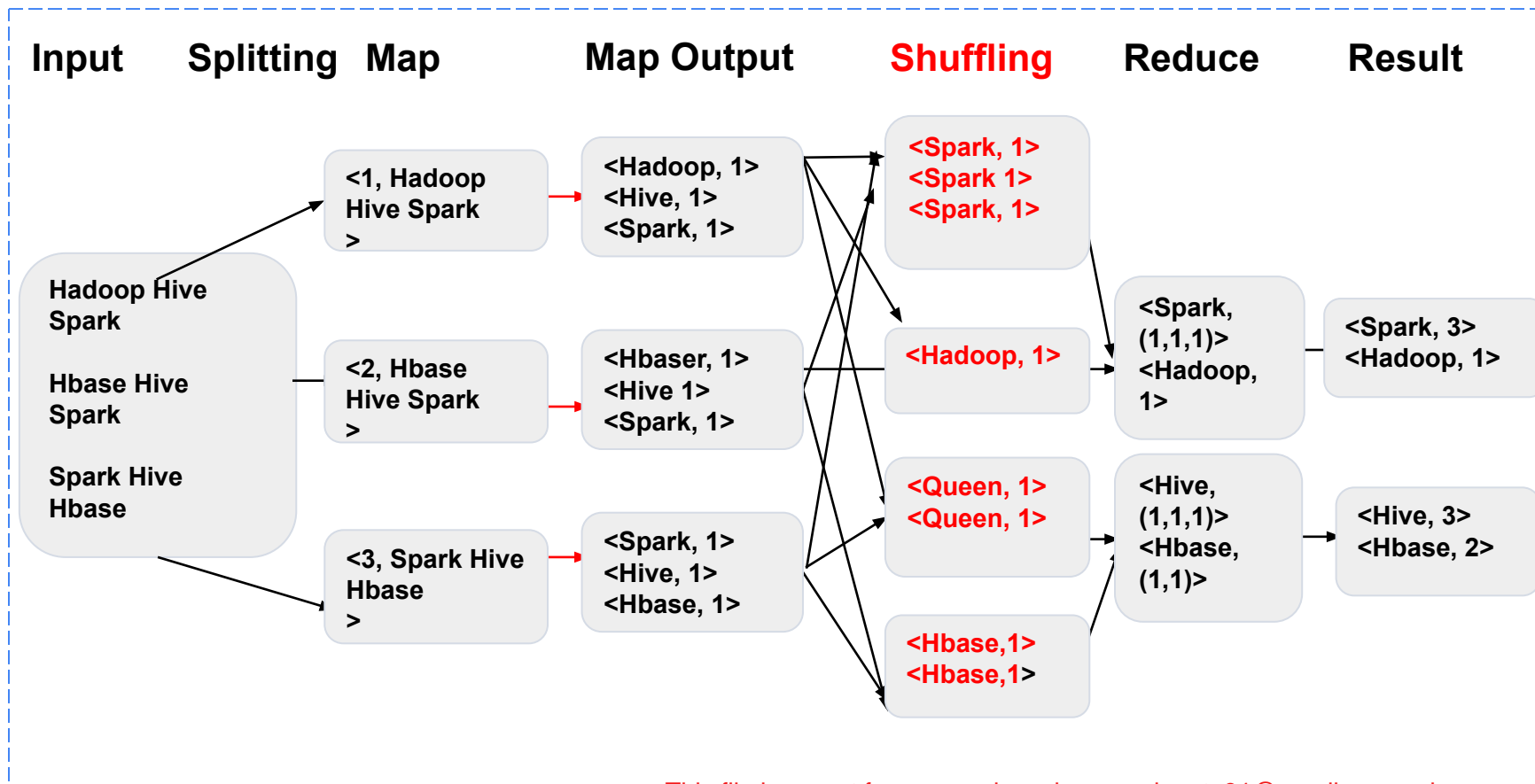
MapReduce

Hadoop MapReduce



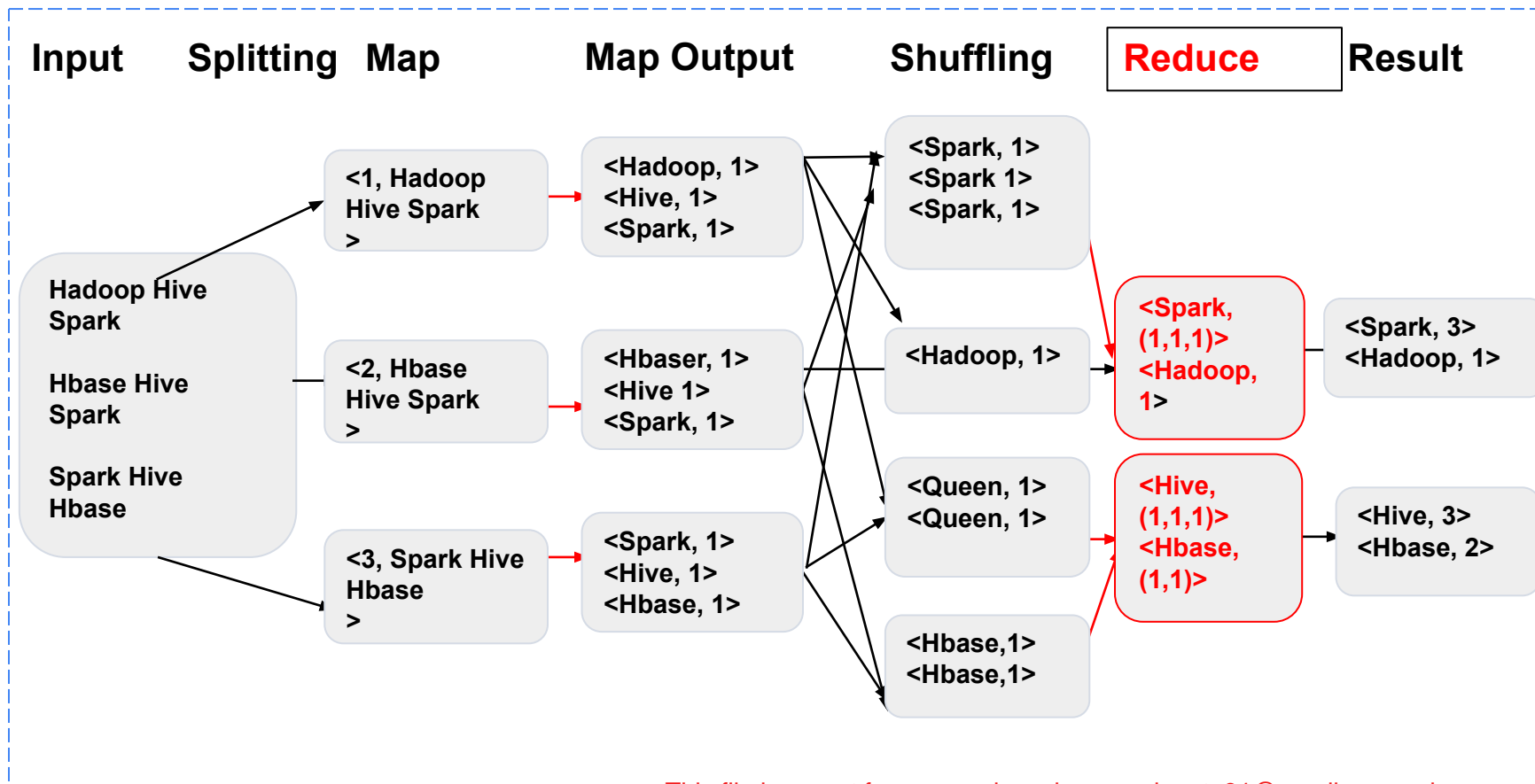
MapReduce

Hadoop MapReduce



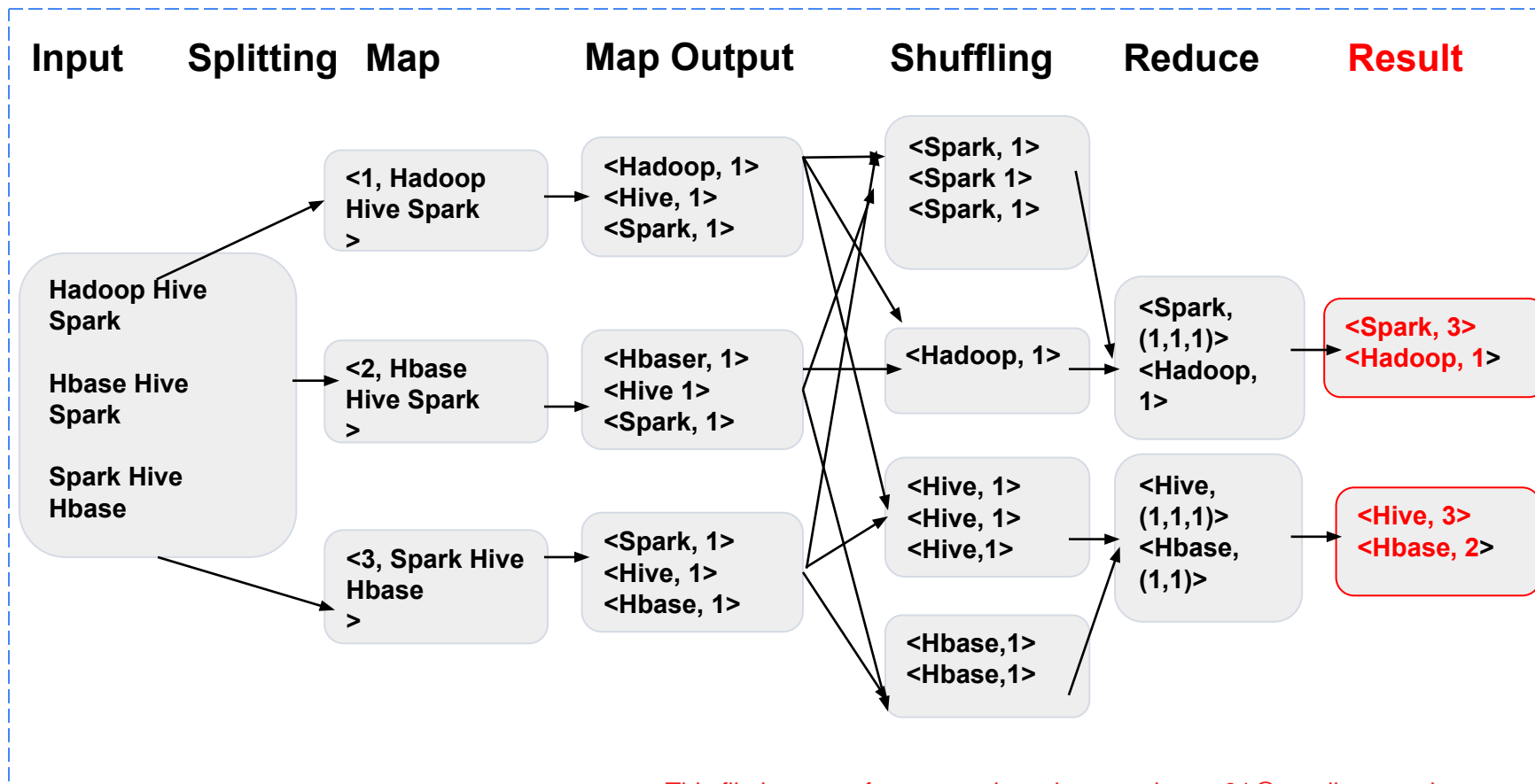
MapReduce

Hadoop MapReduce



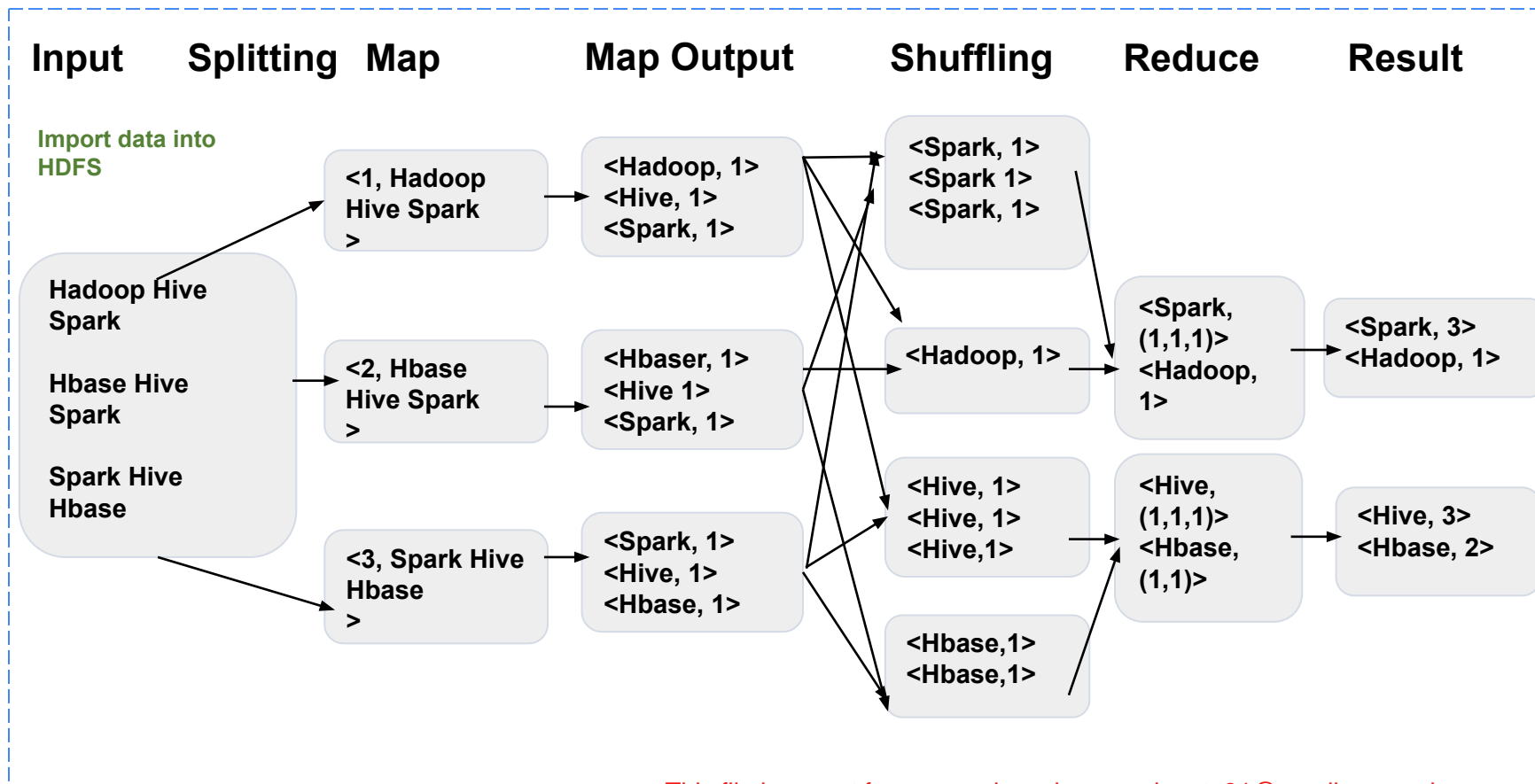
MapReduce

Hadoop MapReduce



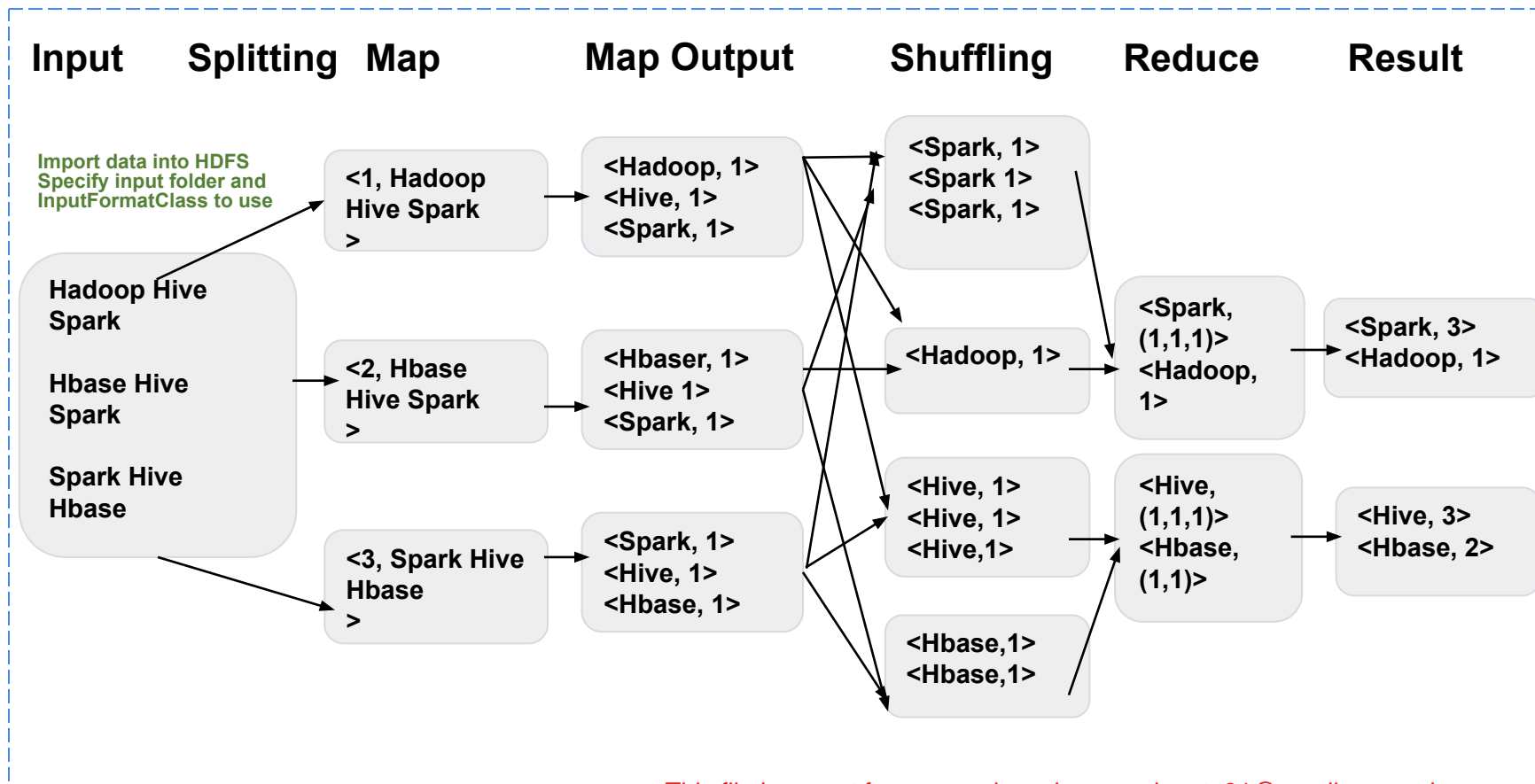
MapReduce

Hadoop MapReduce



MapReduce

Hadoop MapReduce

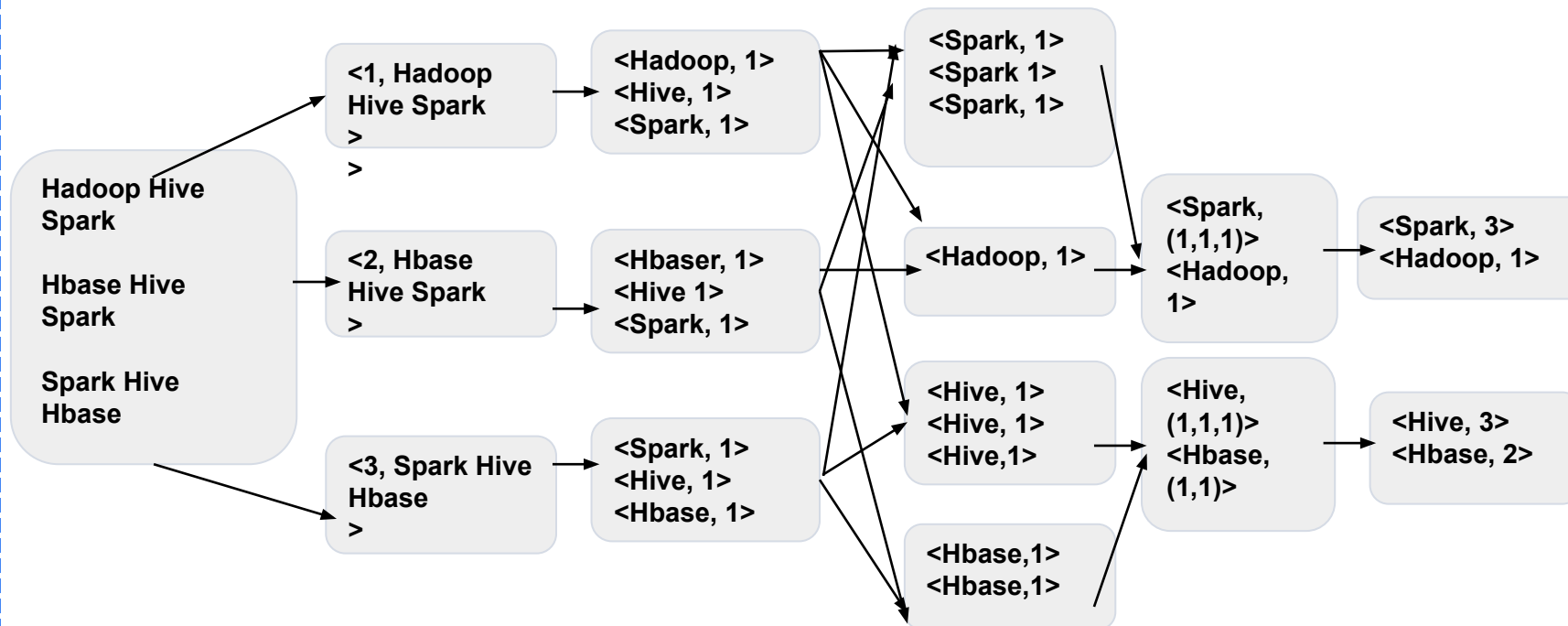


MapReduce

Hadoop MapReduce – Roles: User vs. Framework

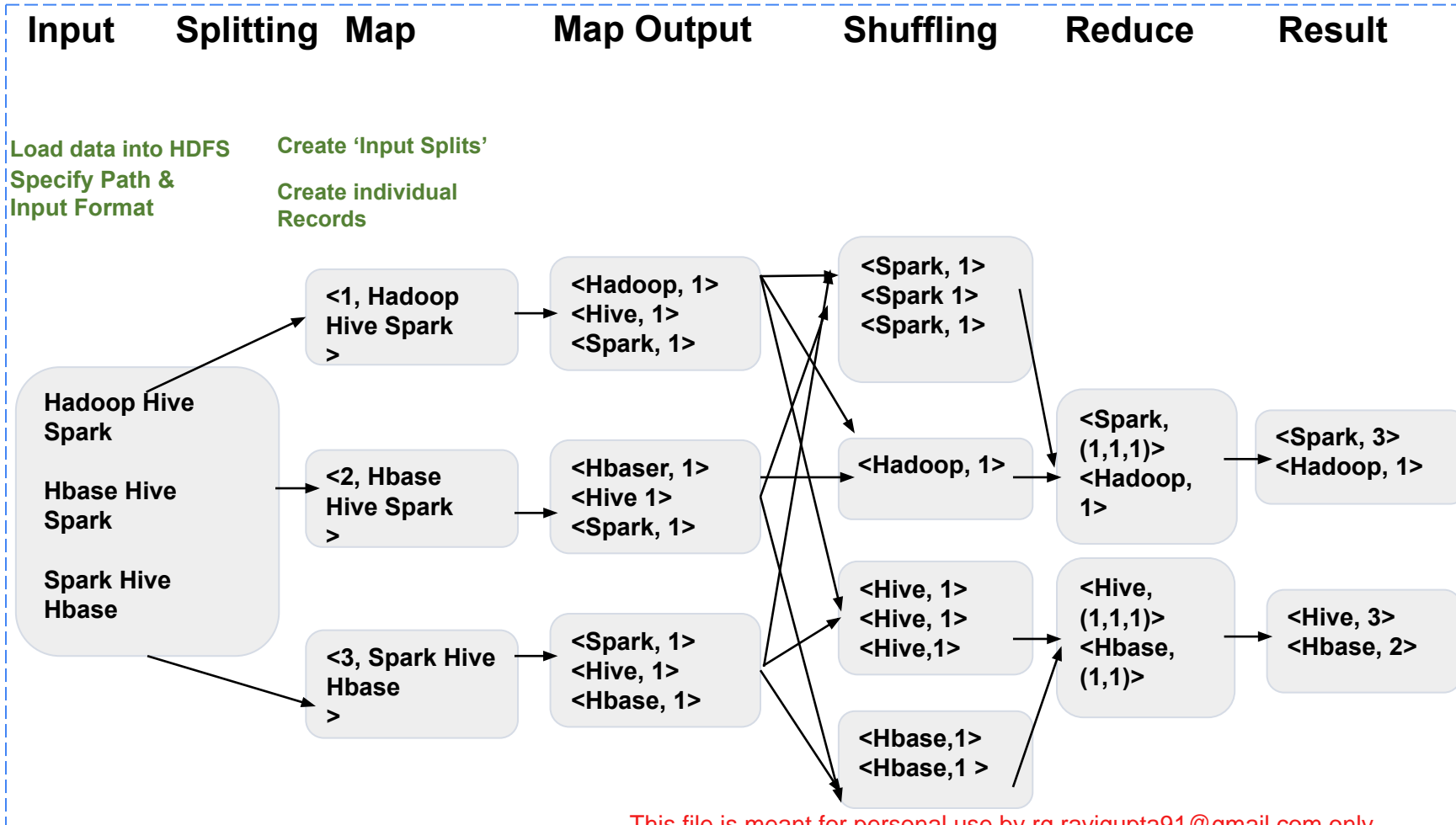
Input Splitting Map Map Output Shuffling Reduce Result

Load data into HDFS Create 'Input Splits'
Specify Path &
Input Format



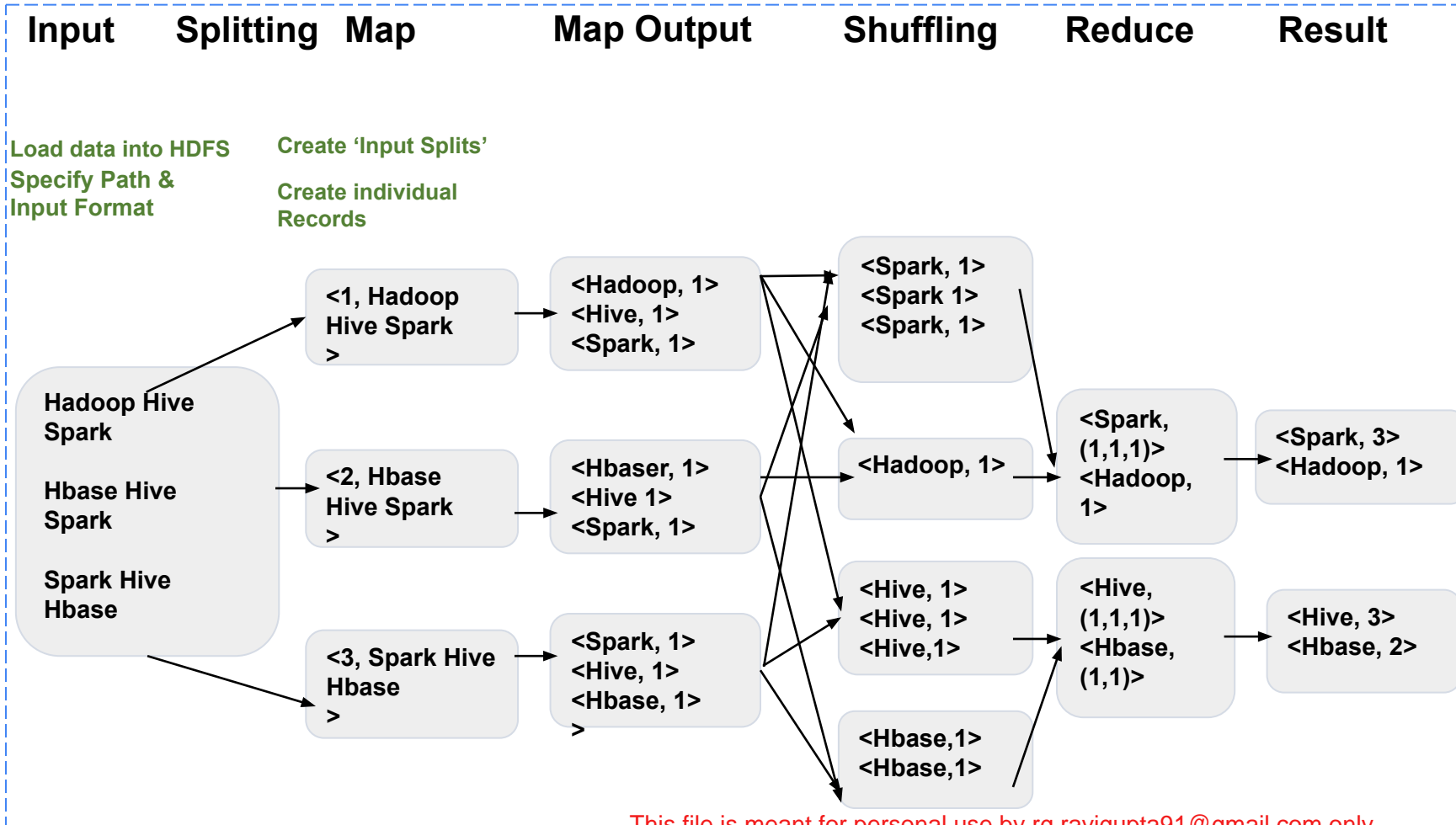
MapReduce

Hadoop MapReduce – Roles: User vs. Framework



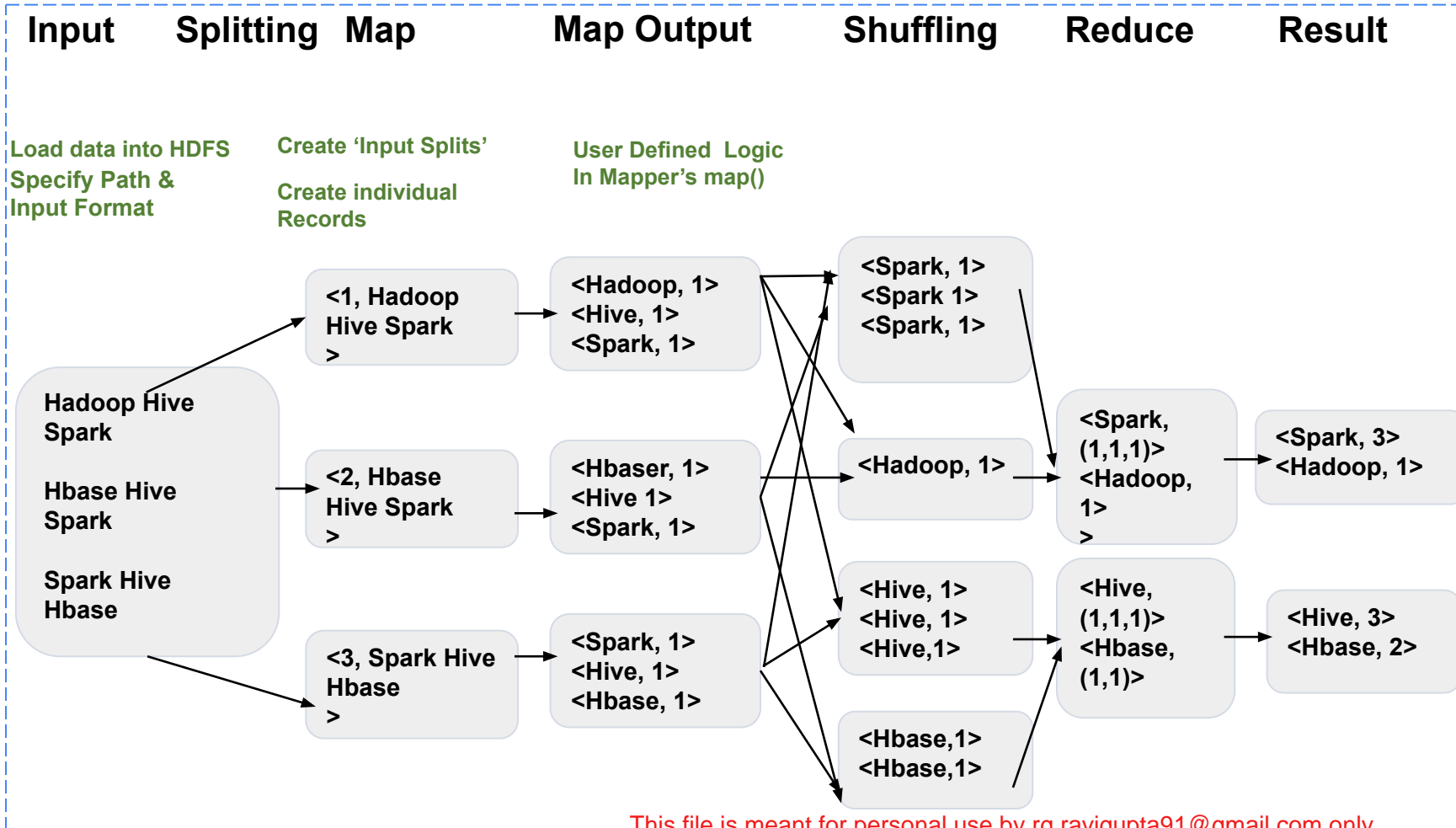
MapReduce

Hadoop MapReduce – Roles: User vs. Framework



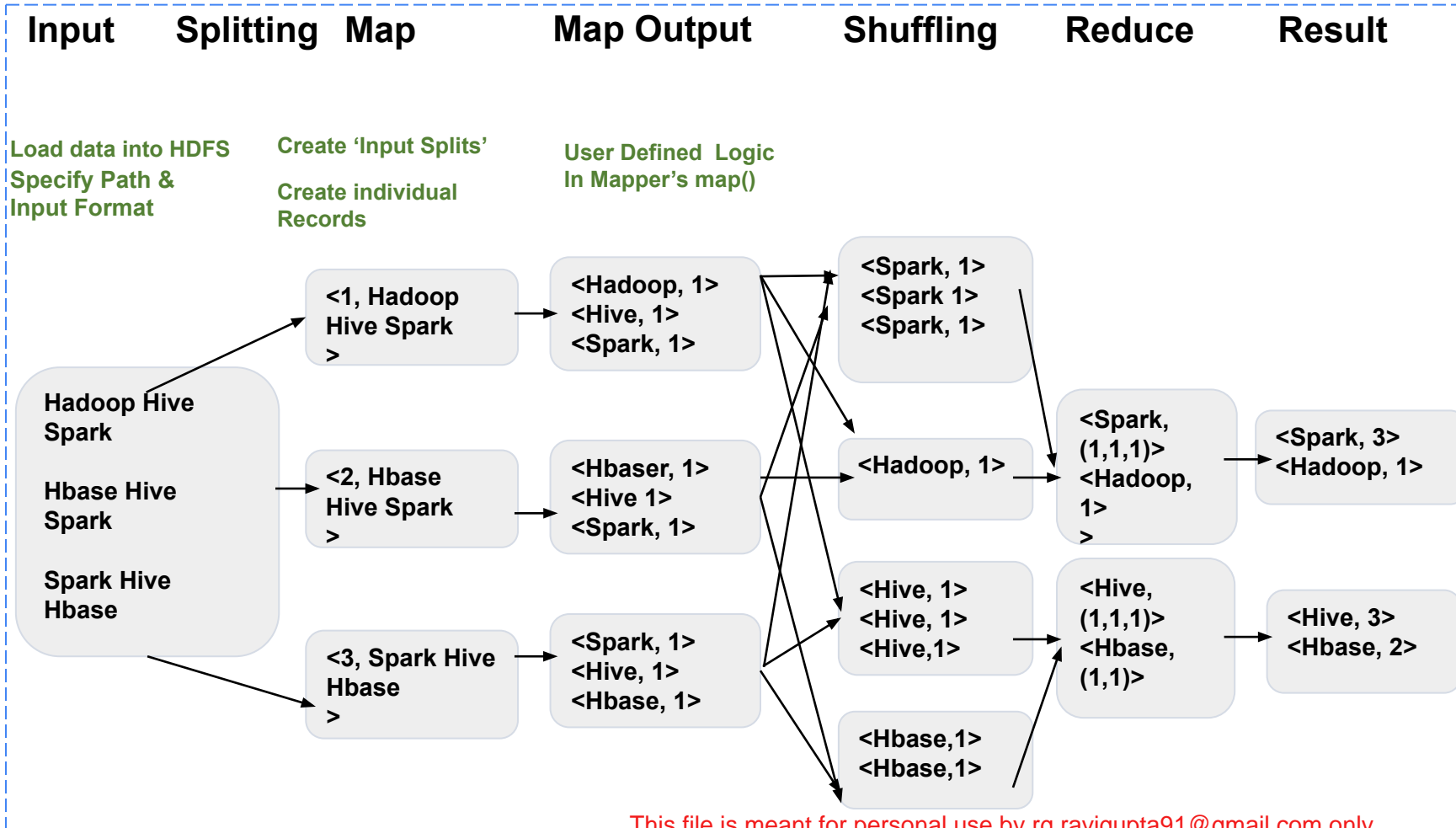
MapReduce

Hadoop MapReduce – Roles: User vs. Framework



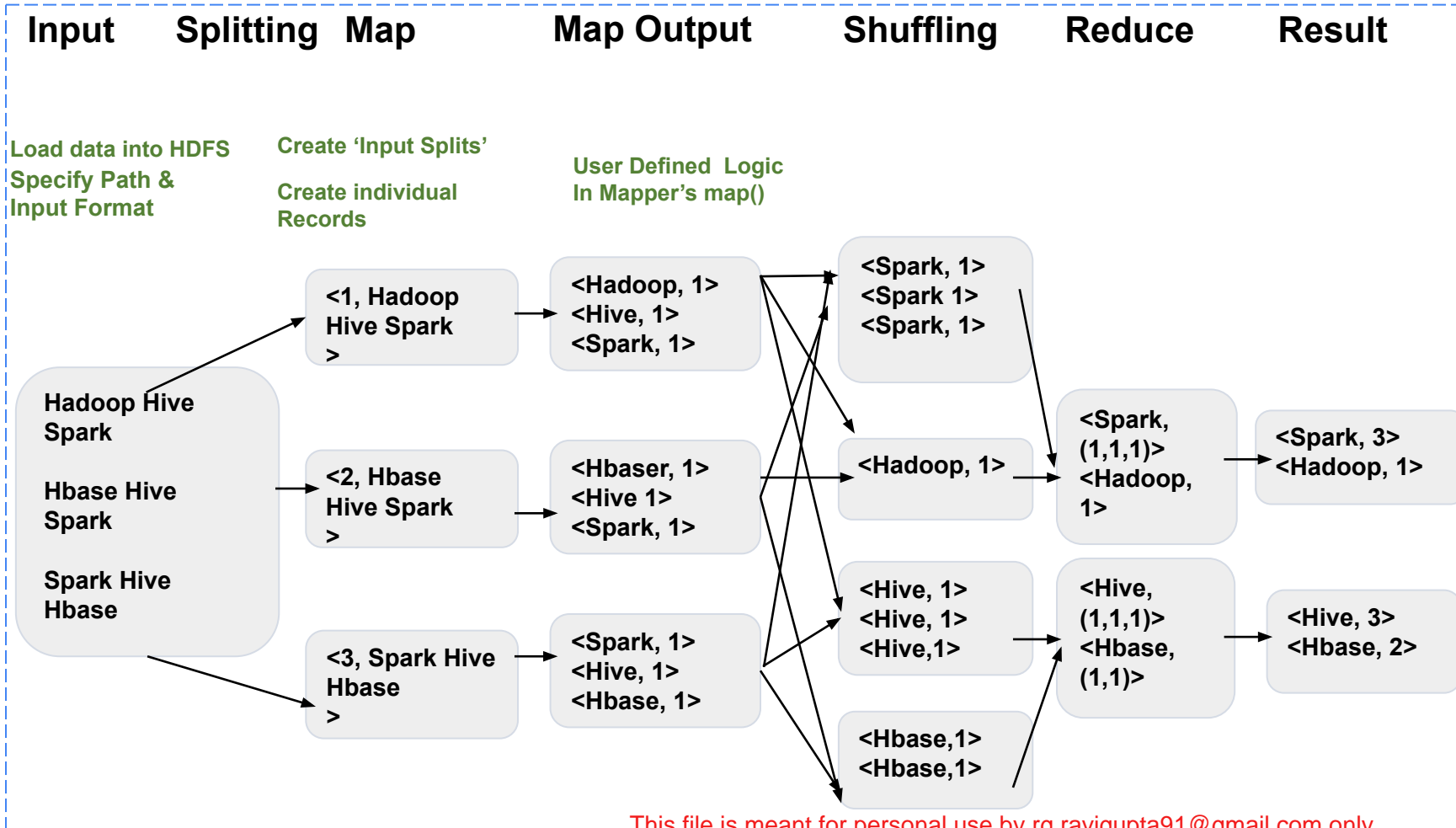
MapReduce

Hadoop MapReduce – Roles: User vs. Framework



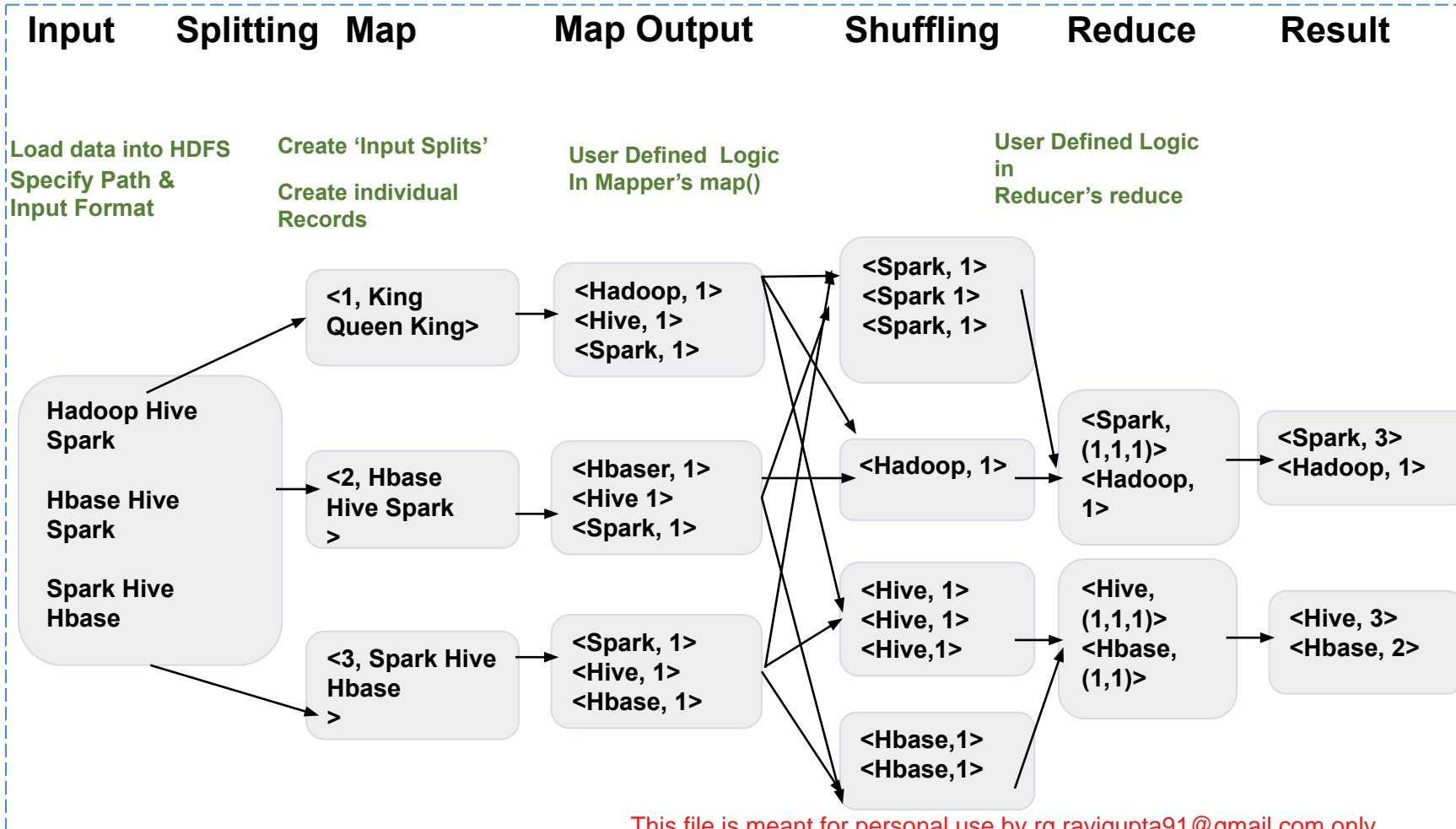
MapReduce

Hadoop MapReduce – Roles: User vs. Framework



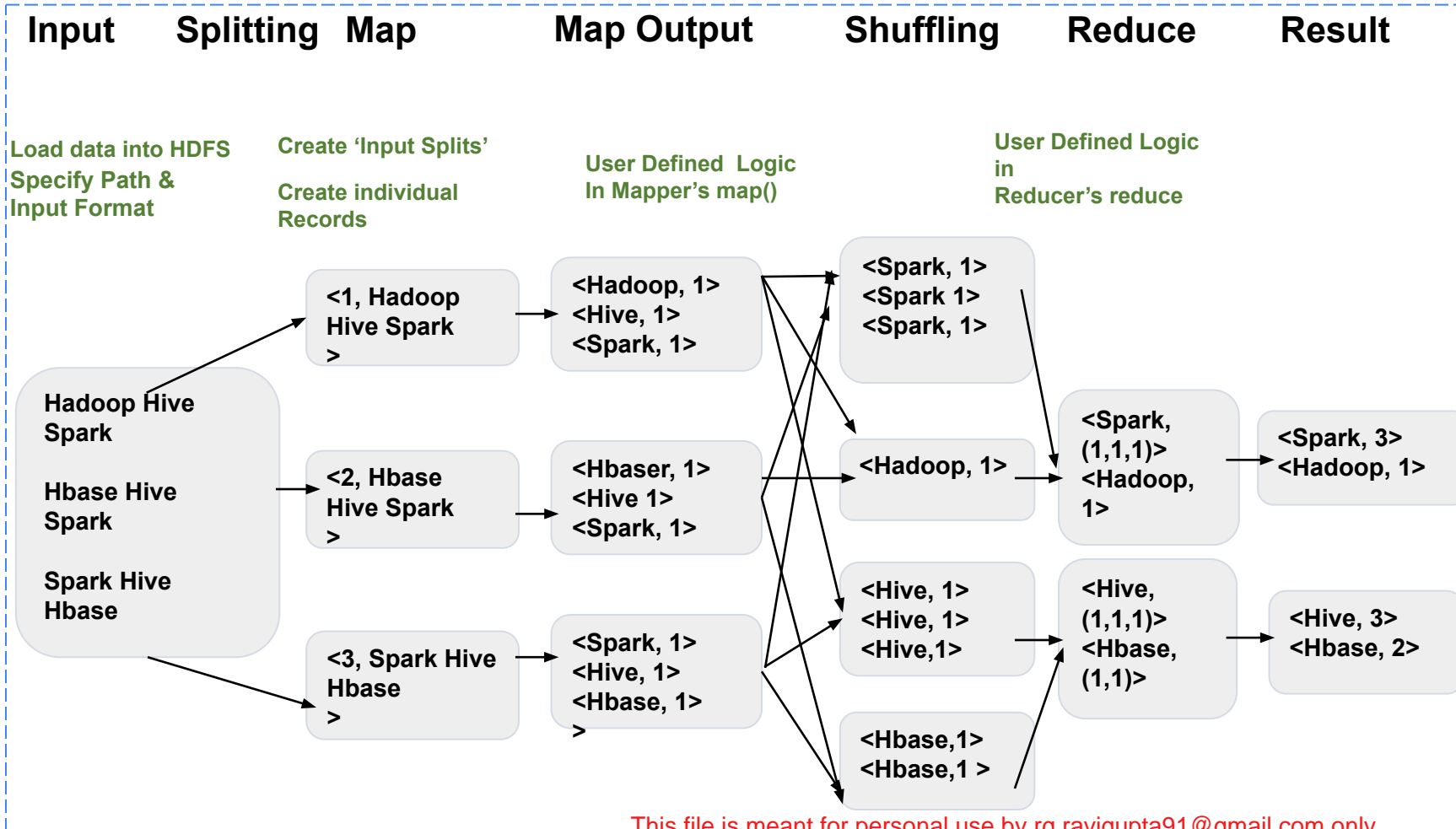
MapReduce

Hadoop MapReduce – Roles: User vs. Framework



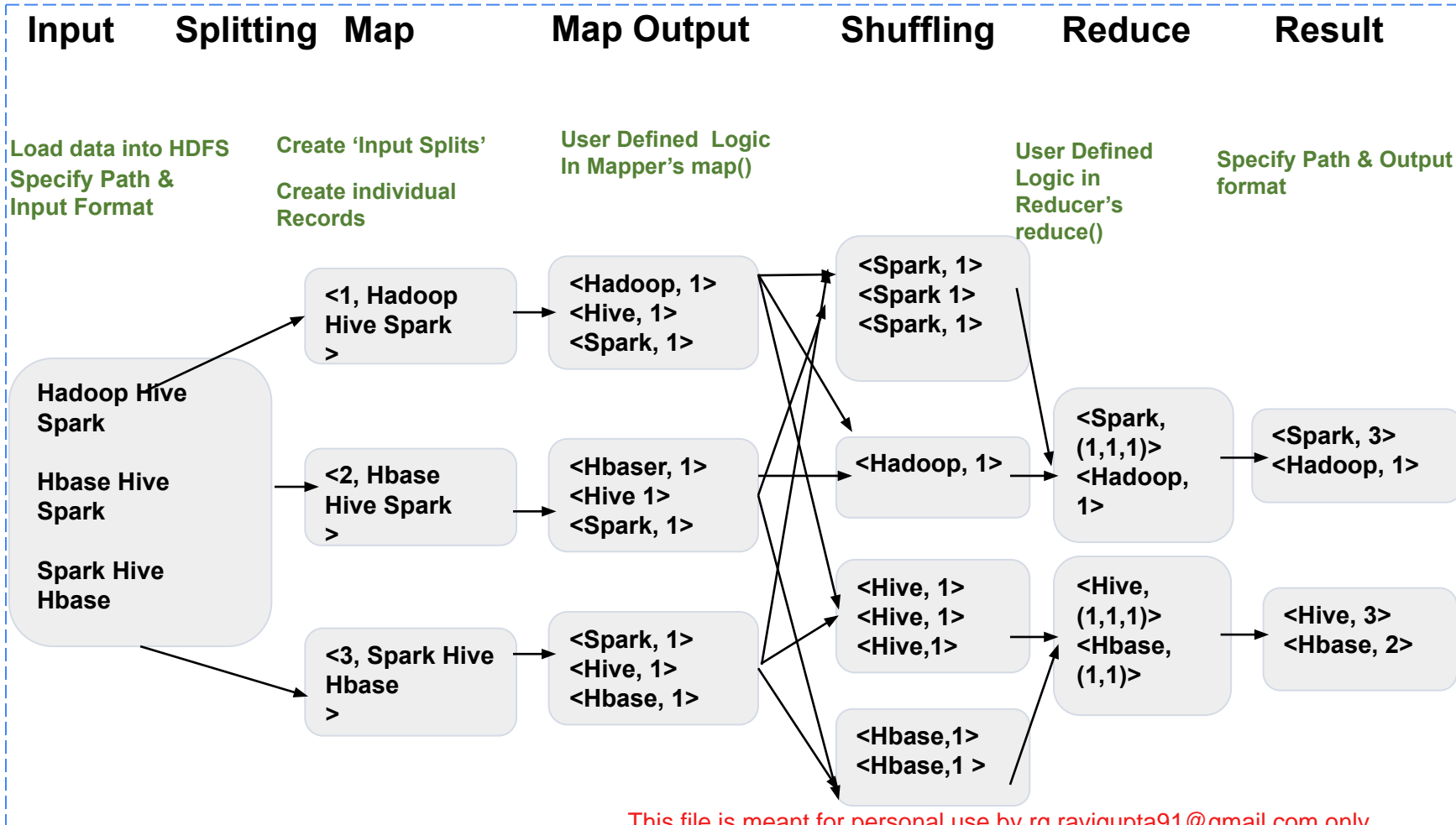
MapReduce

Hadoop MapReduce – Roles: User vs. Framework



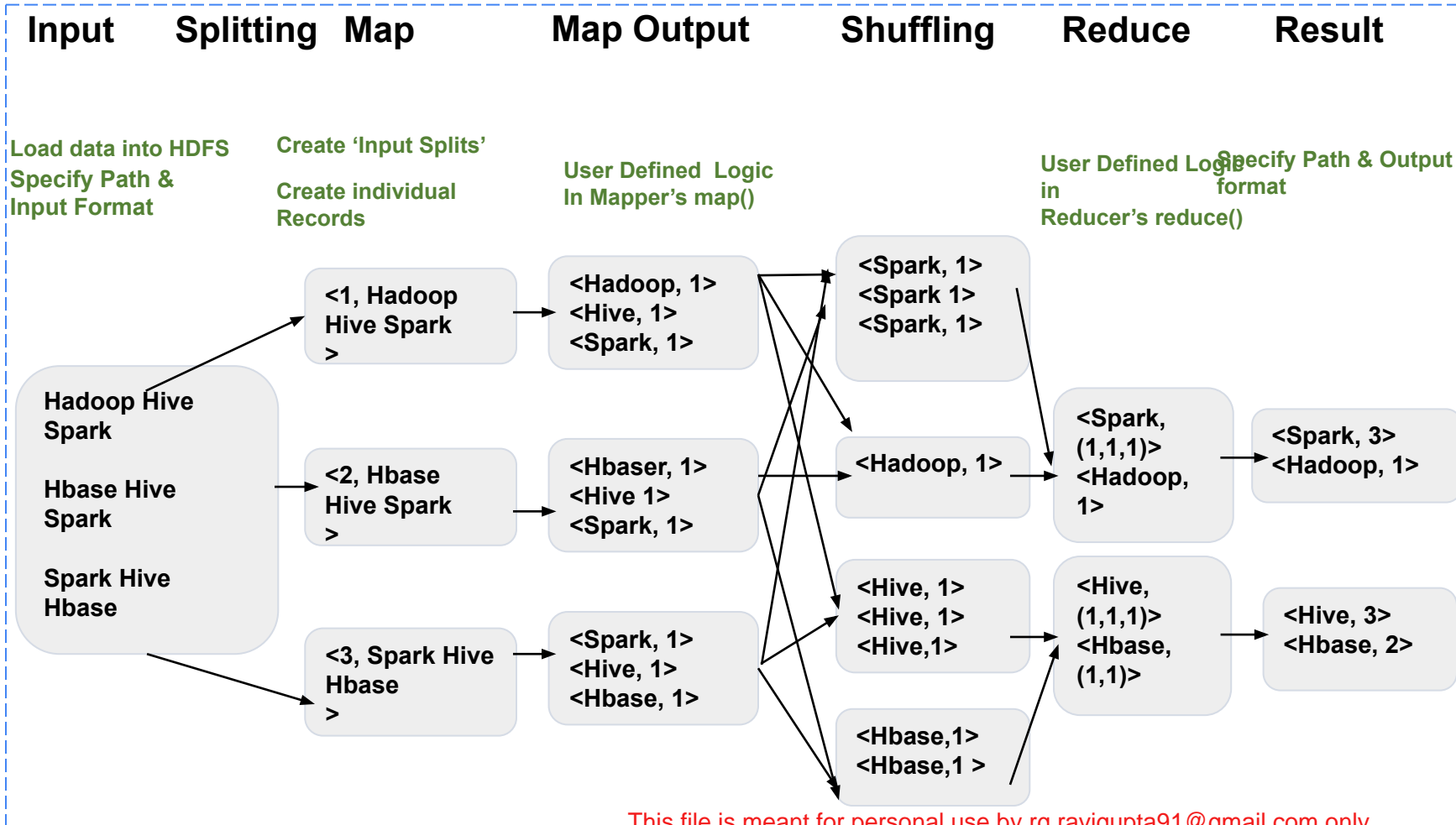
MapReduce

Hadoop MapReduce – Roles: User vs. Framework



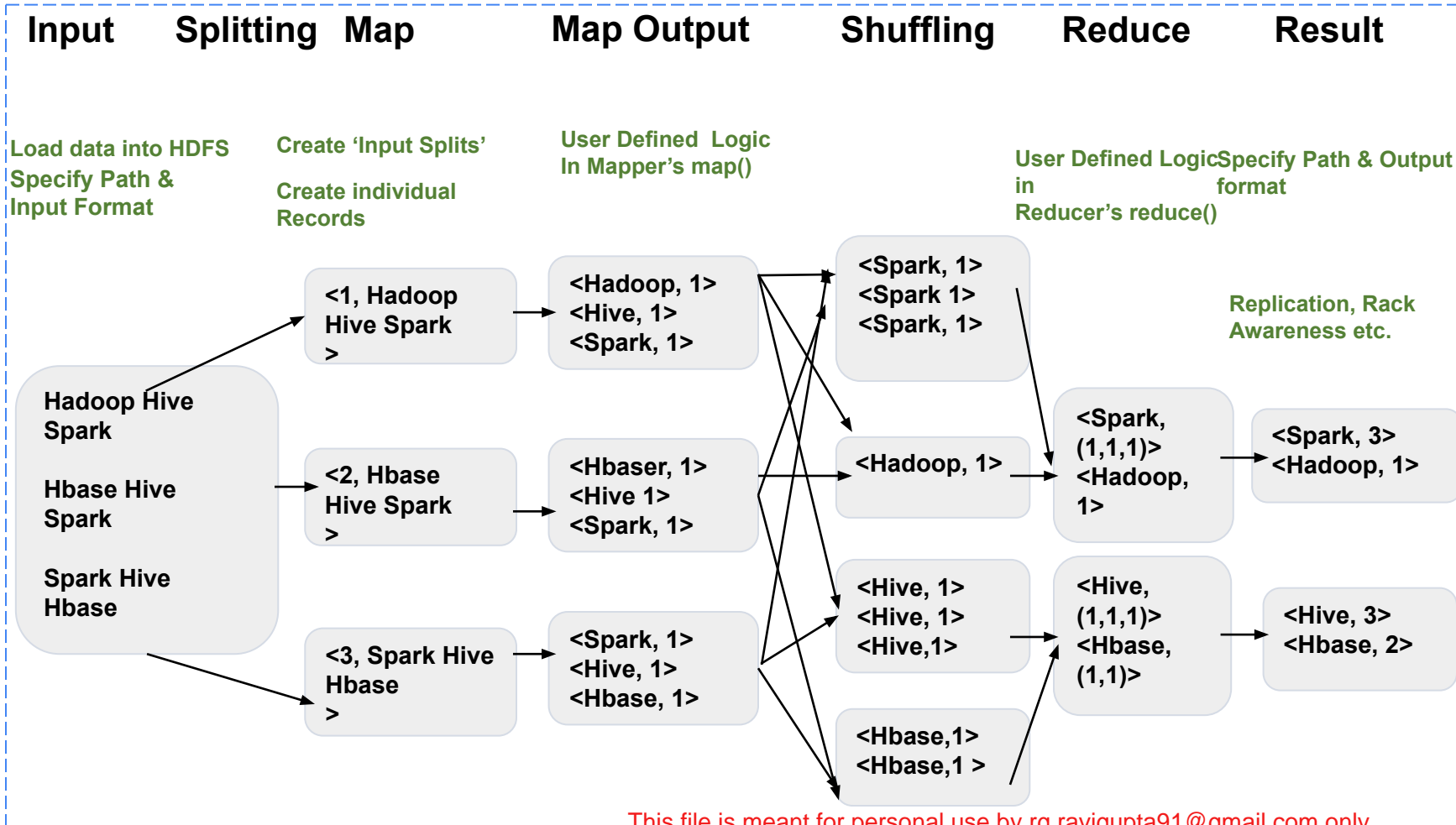
MapReduce

Hadoop MapReduce – Roles: User vs. Framework



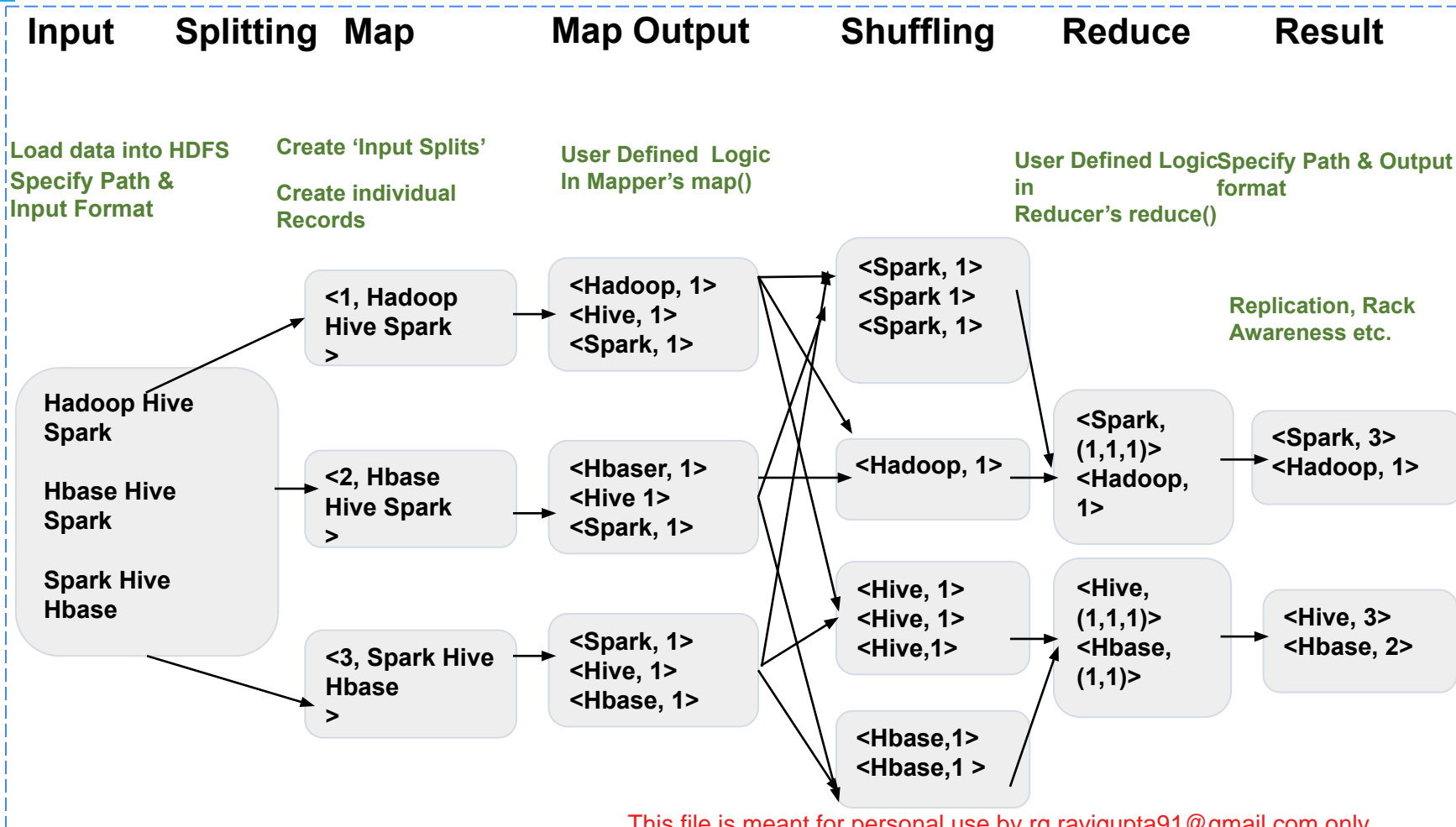
MapReduce

Hadoop MapReduce – Roles: User vs. Framework



MapReduce

Hadoop MapReduce – Roles: User vs. Framework





- *MR Job can contain either map reduce both or just mappers*
- *No of Mappers in the Job depends upon the input split*
- *No of reducer in the Job needs to be set by the developer*
- *By default, no of reducer in the Job is just one*
- *Otherwise , recommended value of the reducers in the job is $\frac{1}{4}$ of mappers tasks*
- *Each Map and Reduce task is separate container of JVM*
- *By default , Map and reduce JVM have 1 GB Memory and 1 CPU core*



- *Driver is the process which runs inside the Application Master JVM*
- *Each Mapper are Reducer tasks are being run as separate JVM by the Node Manager*
- *Each task can be executed maximum 4 times*
- *Each task is being assigned by Application Master to specific Node Manager*
- *InputFormat class is Java class which has intelligence of making the record complete*

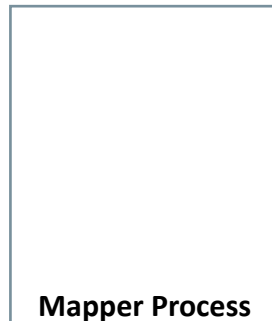
MapReduce



MapReduce Execution Framework

MapReduce

MapReduce Execution Framework



MapReduce

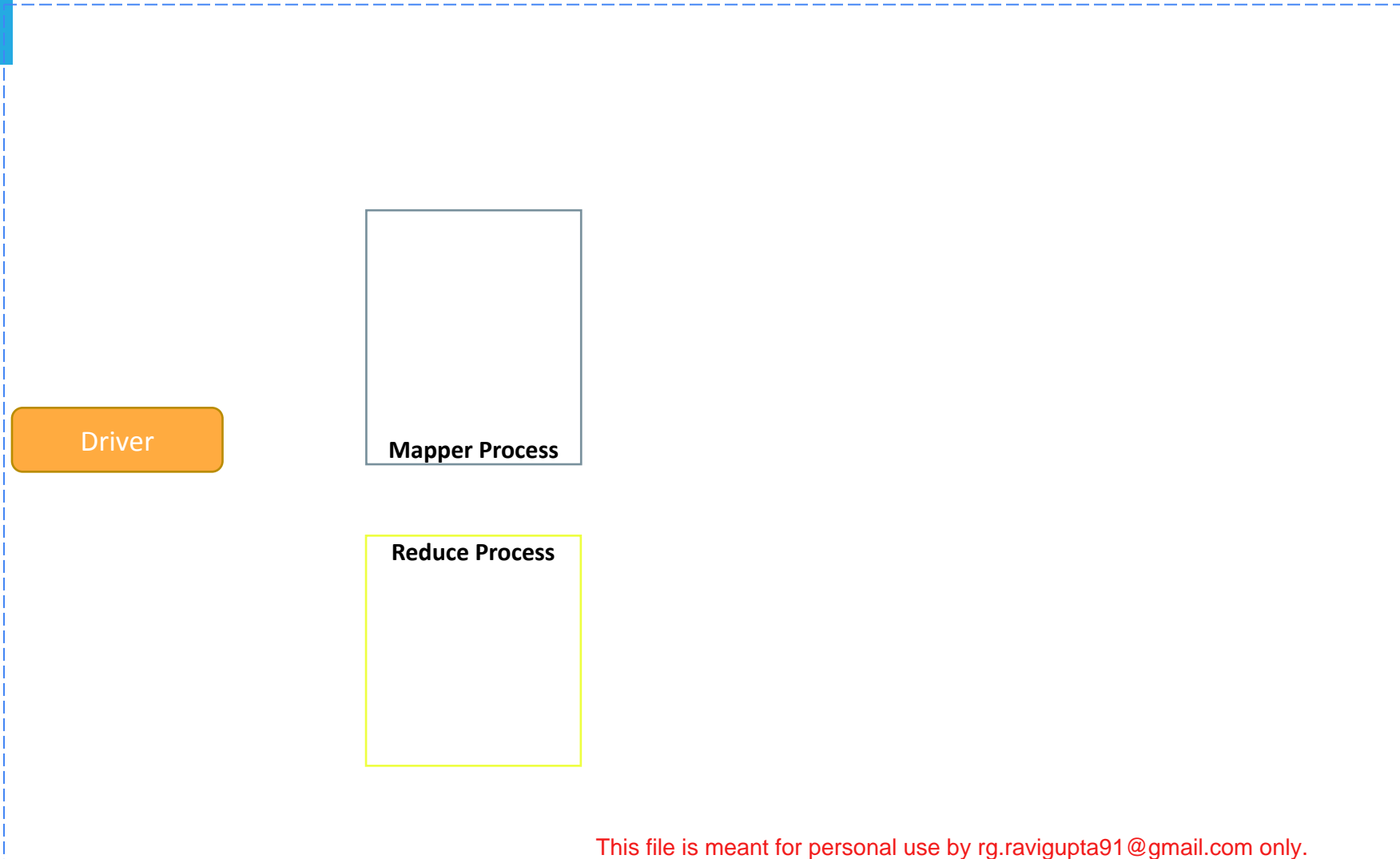
MapReduce Execution Framework

Mapper Process

Reduce Process

MapReduce

MapReduce Execution Framework



MapReduce

MapReduce Execution Framework

Input HDFS File - inputFile.txt

Driver

Mapper Process

Reduce Process

MapReduce

MapReduce Execution Framework

Input HDFS File - inputFile.txt

Block A

Block B

Block C

Driver

Mapper Process

Reduce Process

MapReduce

MapReduce Execution Framework

Input HDFS File - inputFile.txt

Block A

Block B

Block C

InputFormat

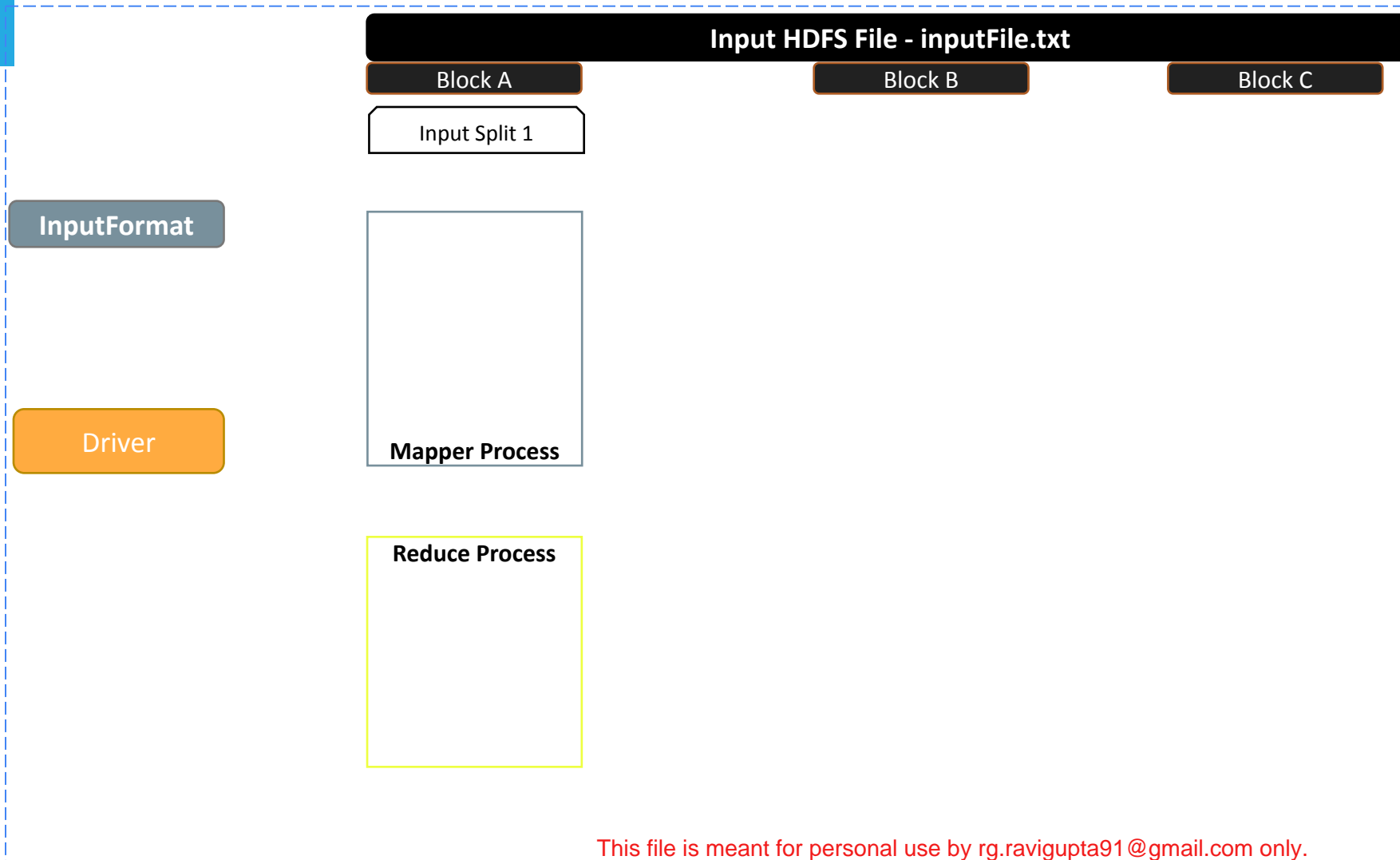
Driver

Mapper Process

Reduce Process

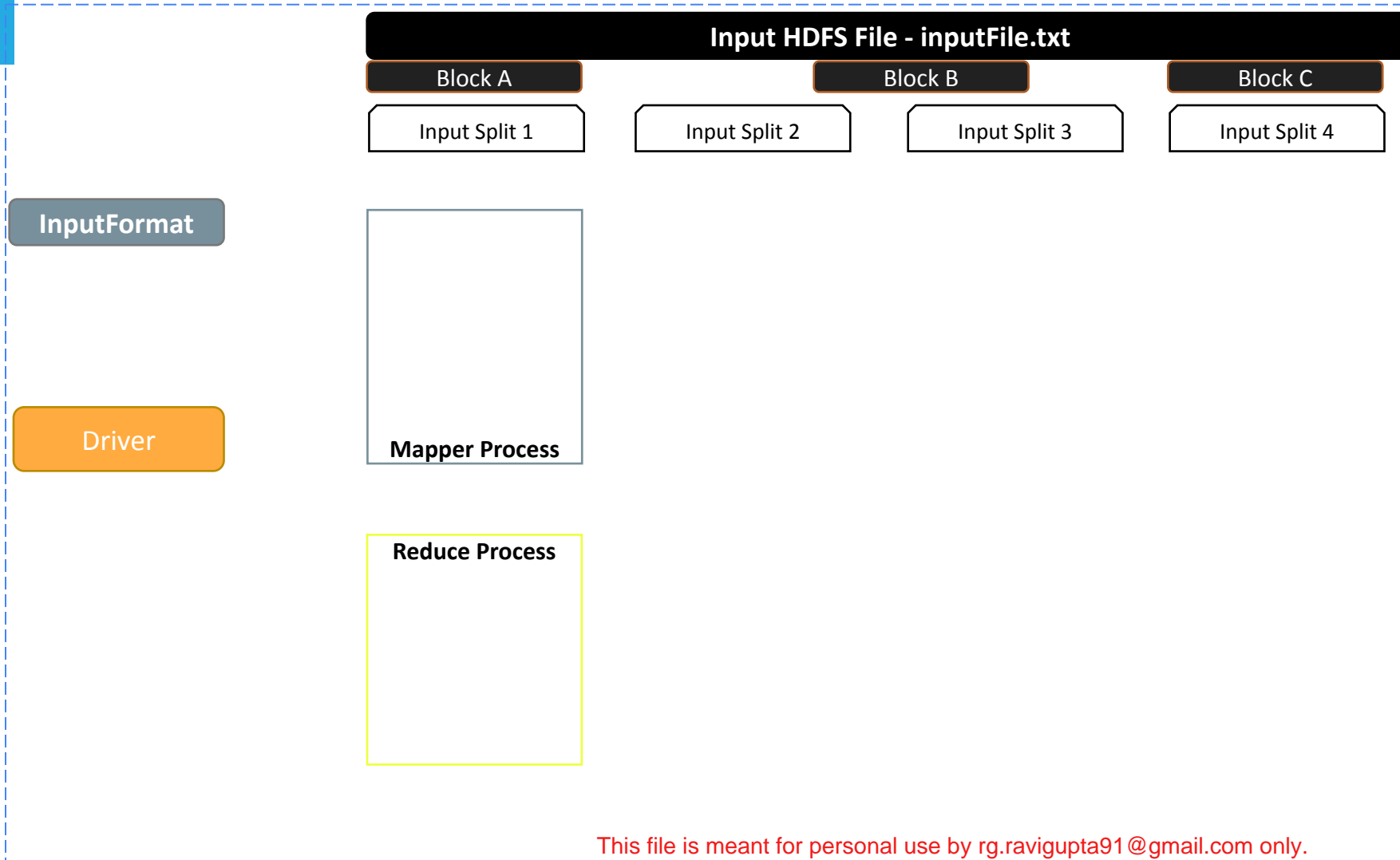
MapReduce

MapReduce Execution Framework



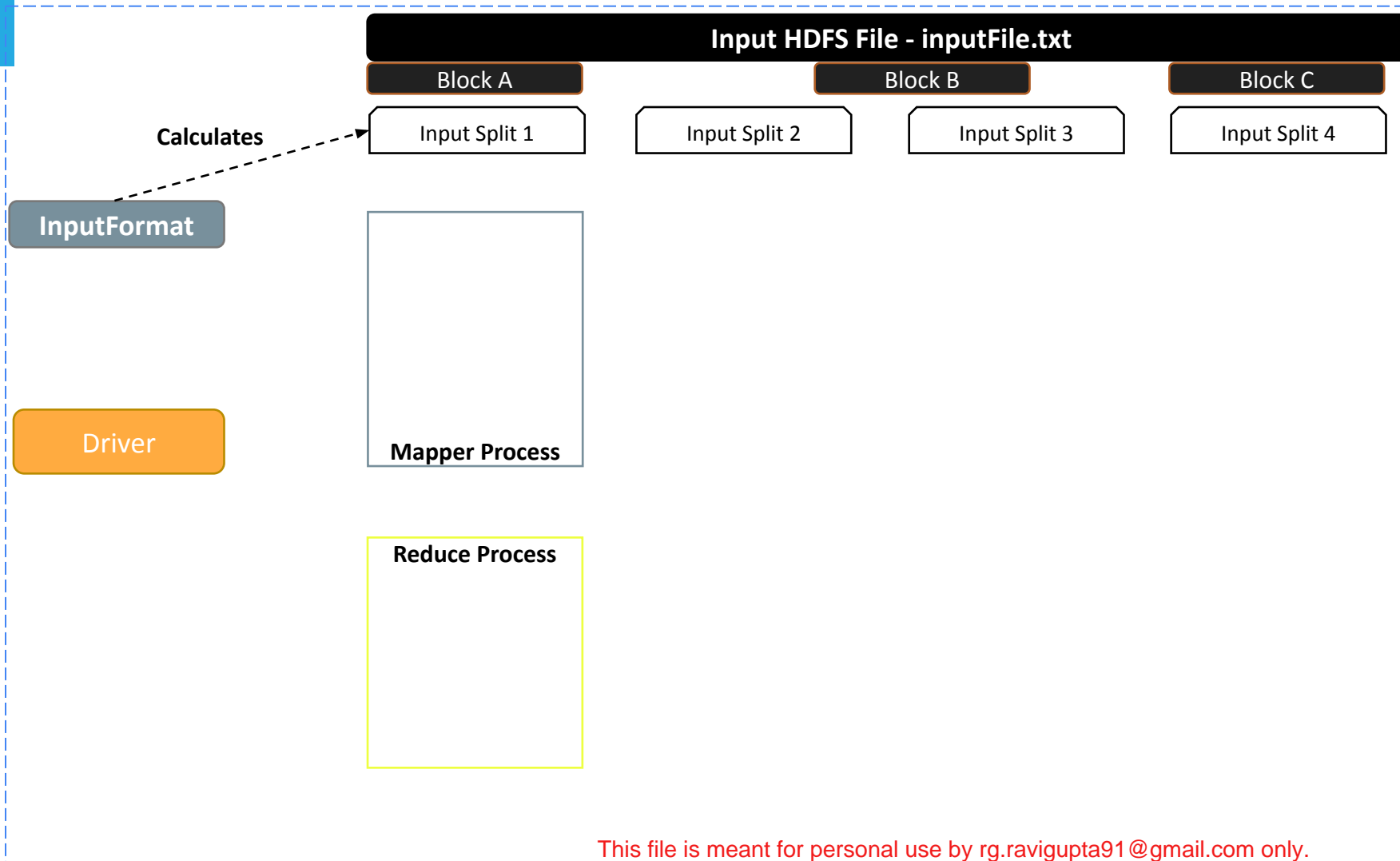
MapReduce

MapReduce Execution Framework



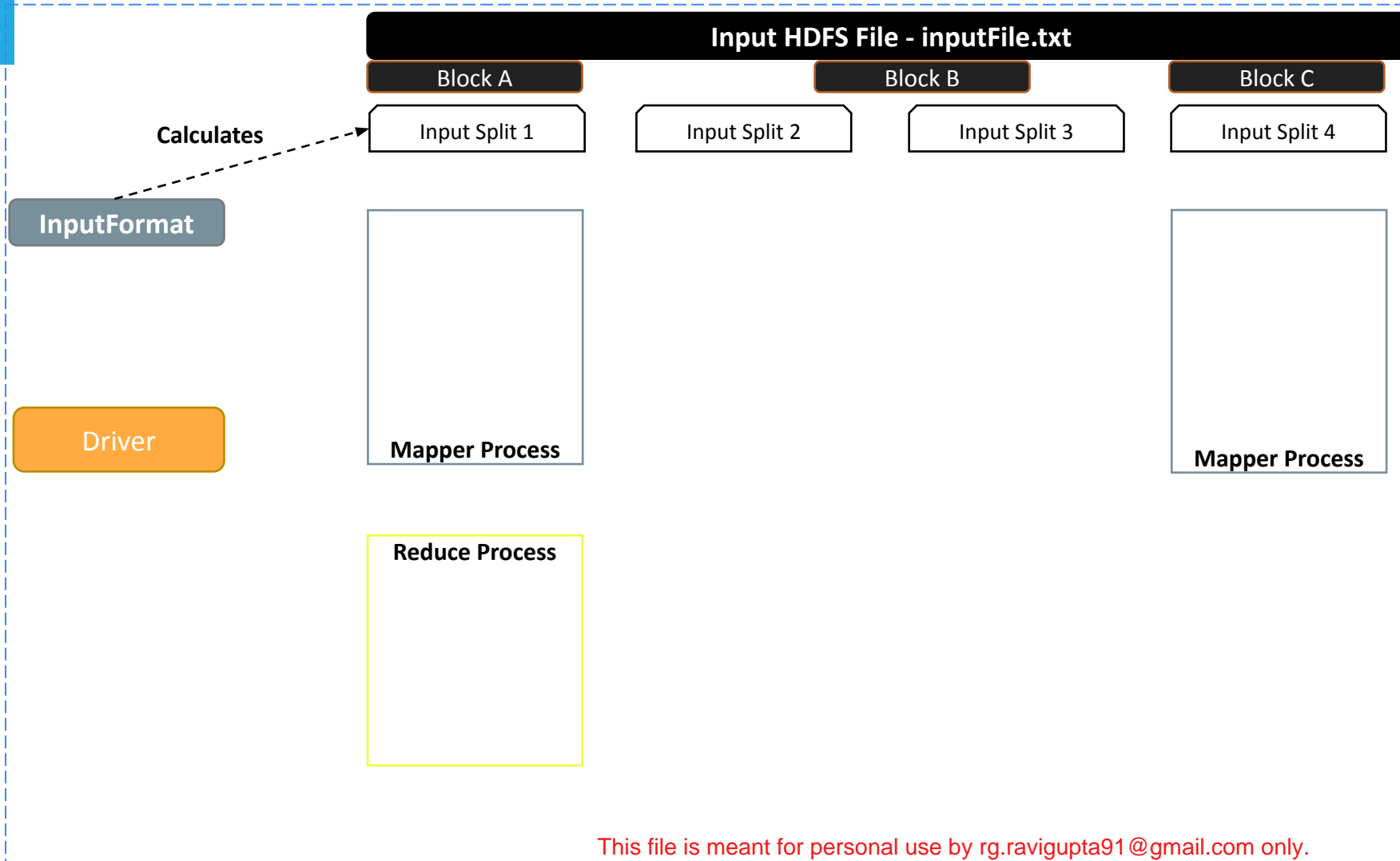
MapReduce

MapReduce Execution Framework



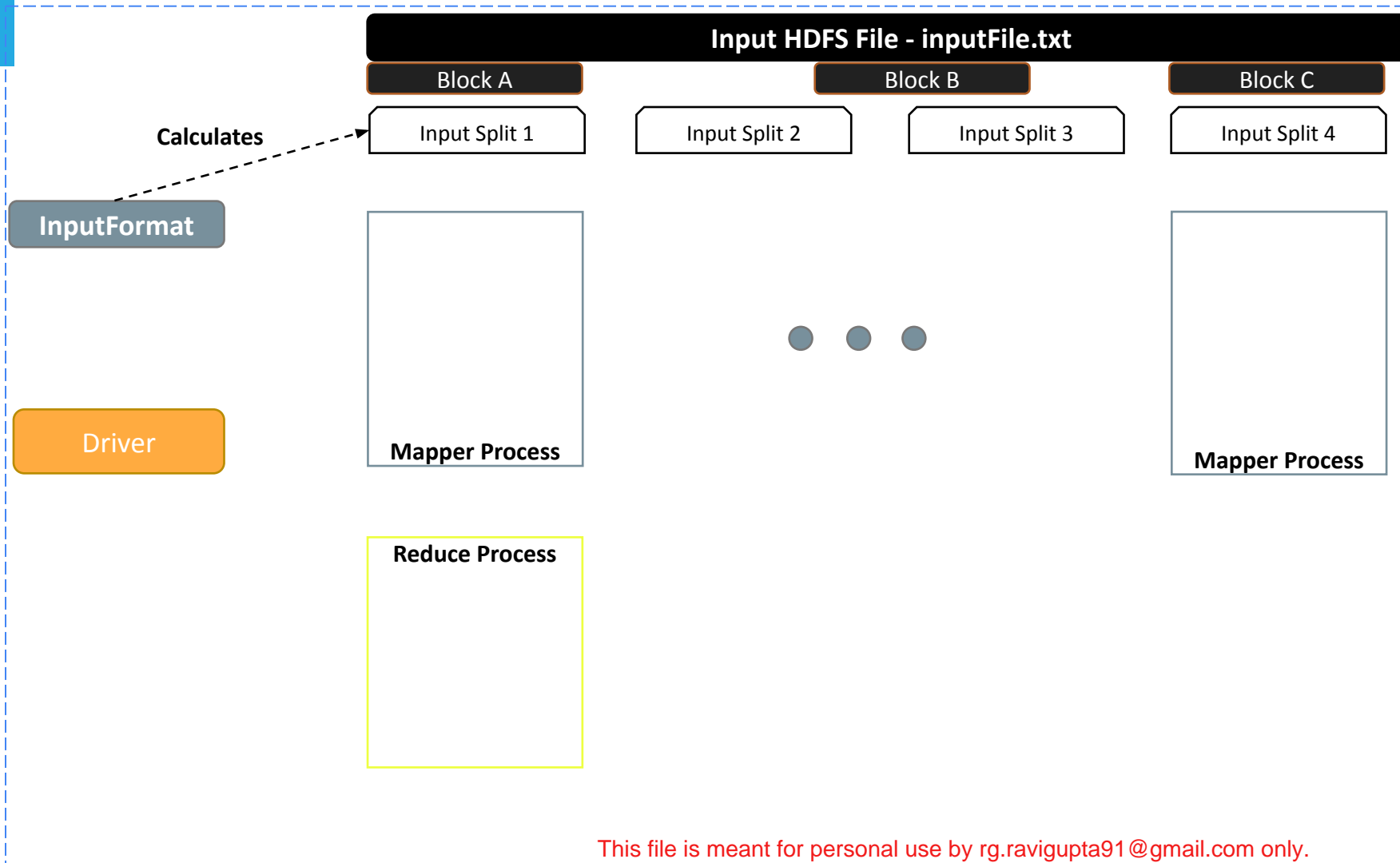
MapReduce

MapReduce Execution Framework



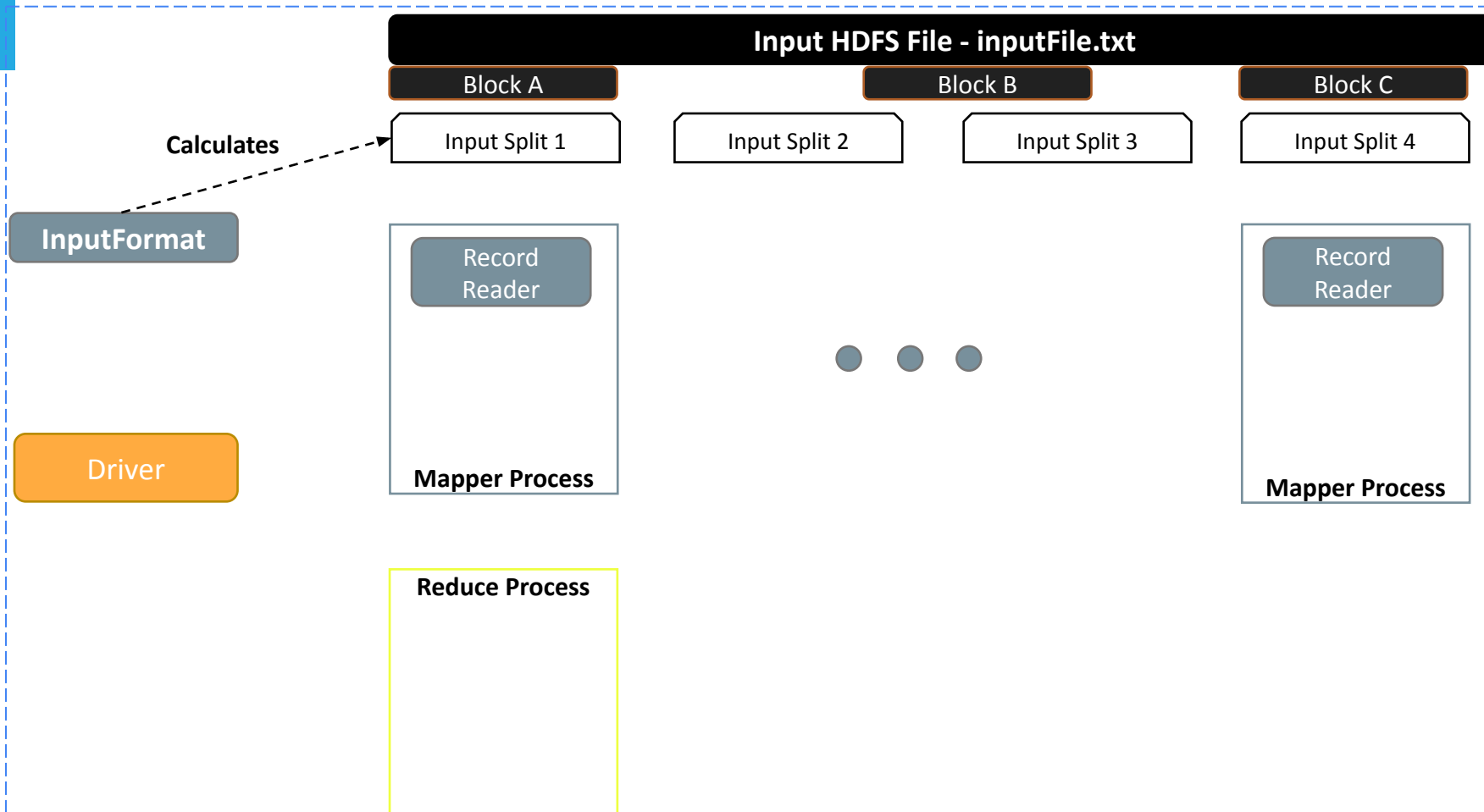
MapReduce

MapReduce Execution Framework



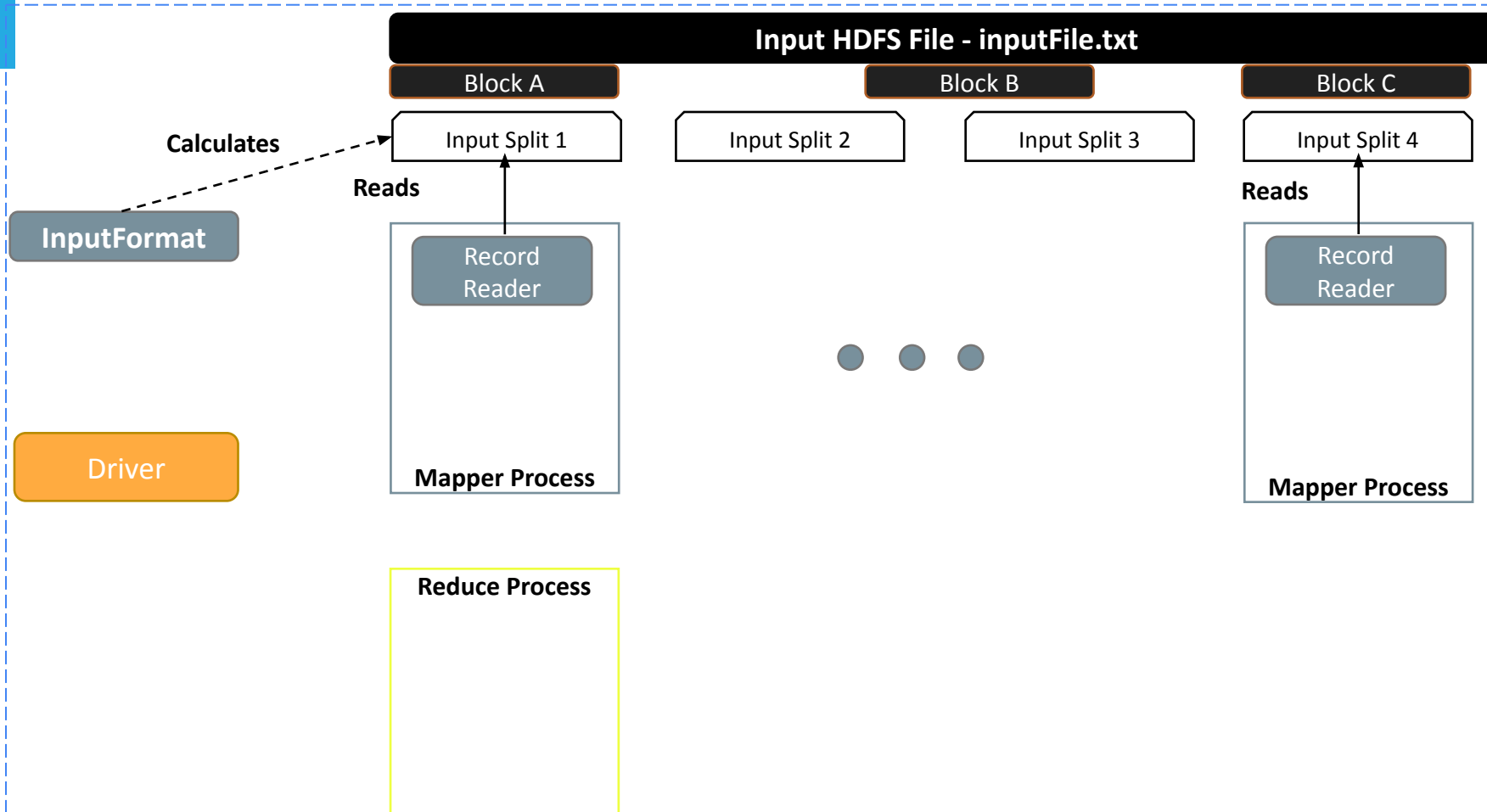
MapReduce

MapReduce Execution Framework



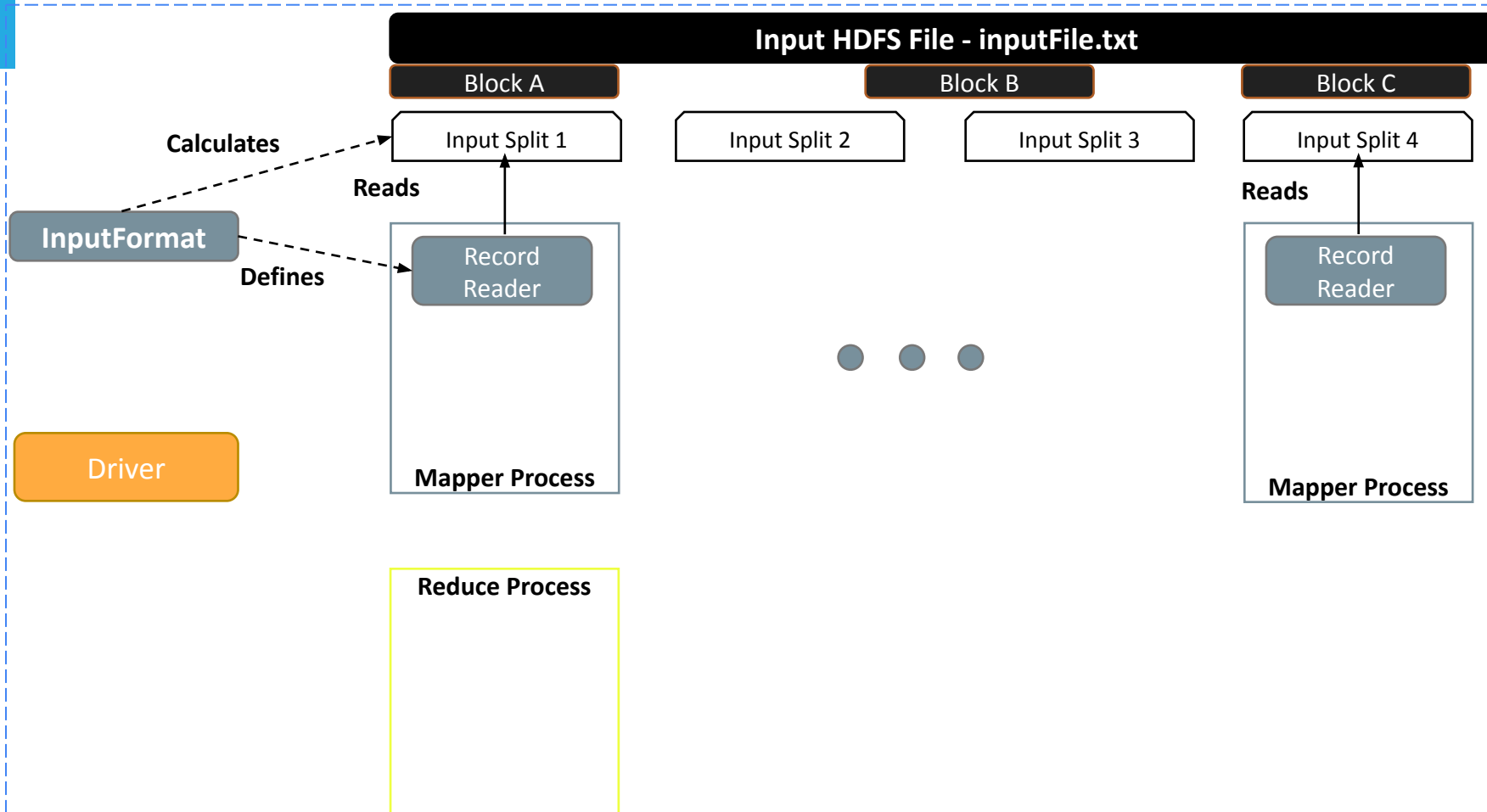
MapReduce

MapReduce Execution Framework



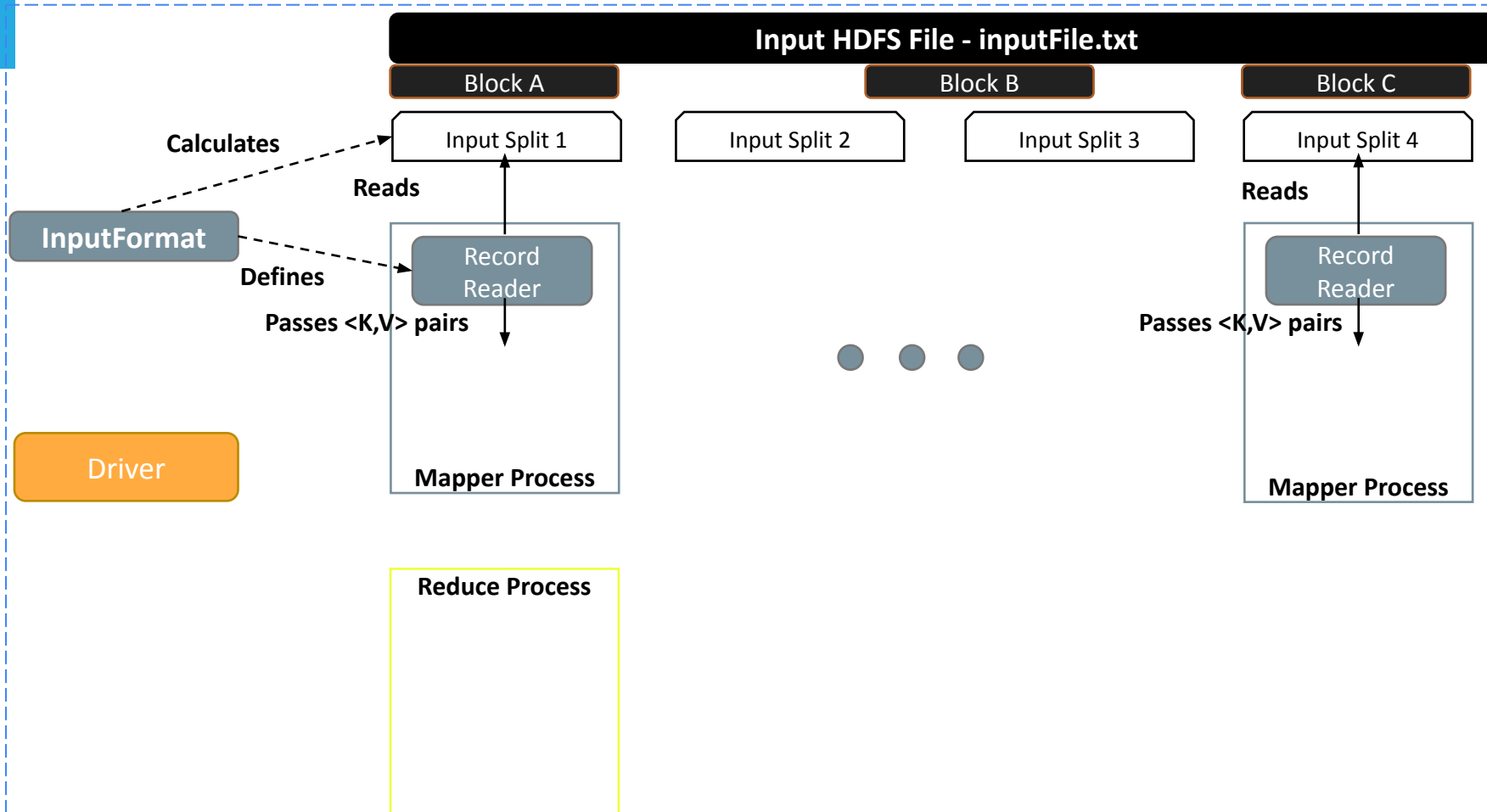
MapReduce

MapReduce Execution Framework



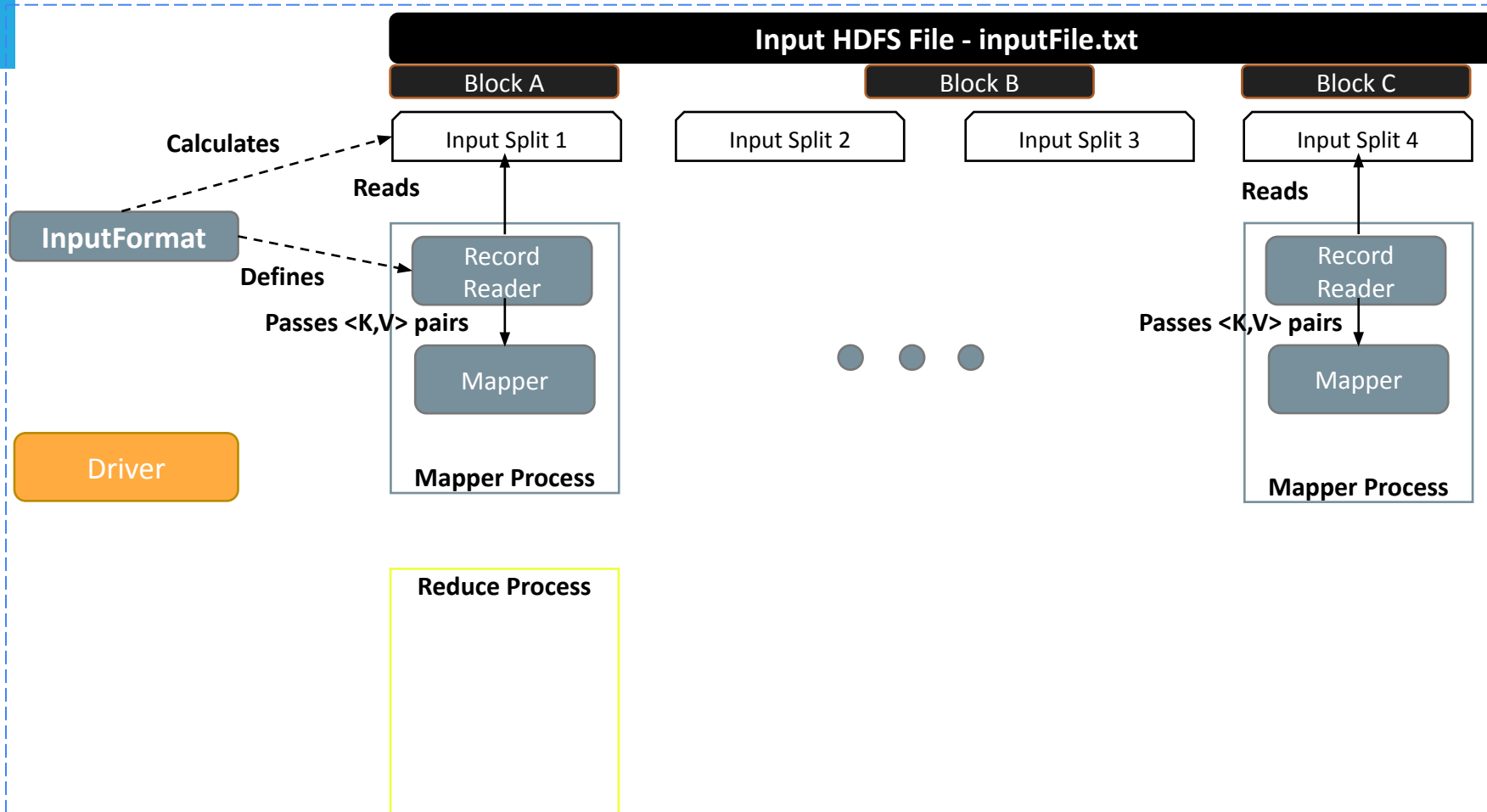
MapReduce

MapReduce Execution Framework



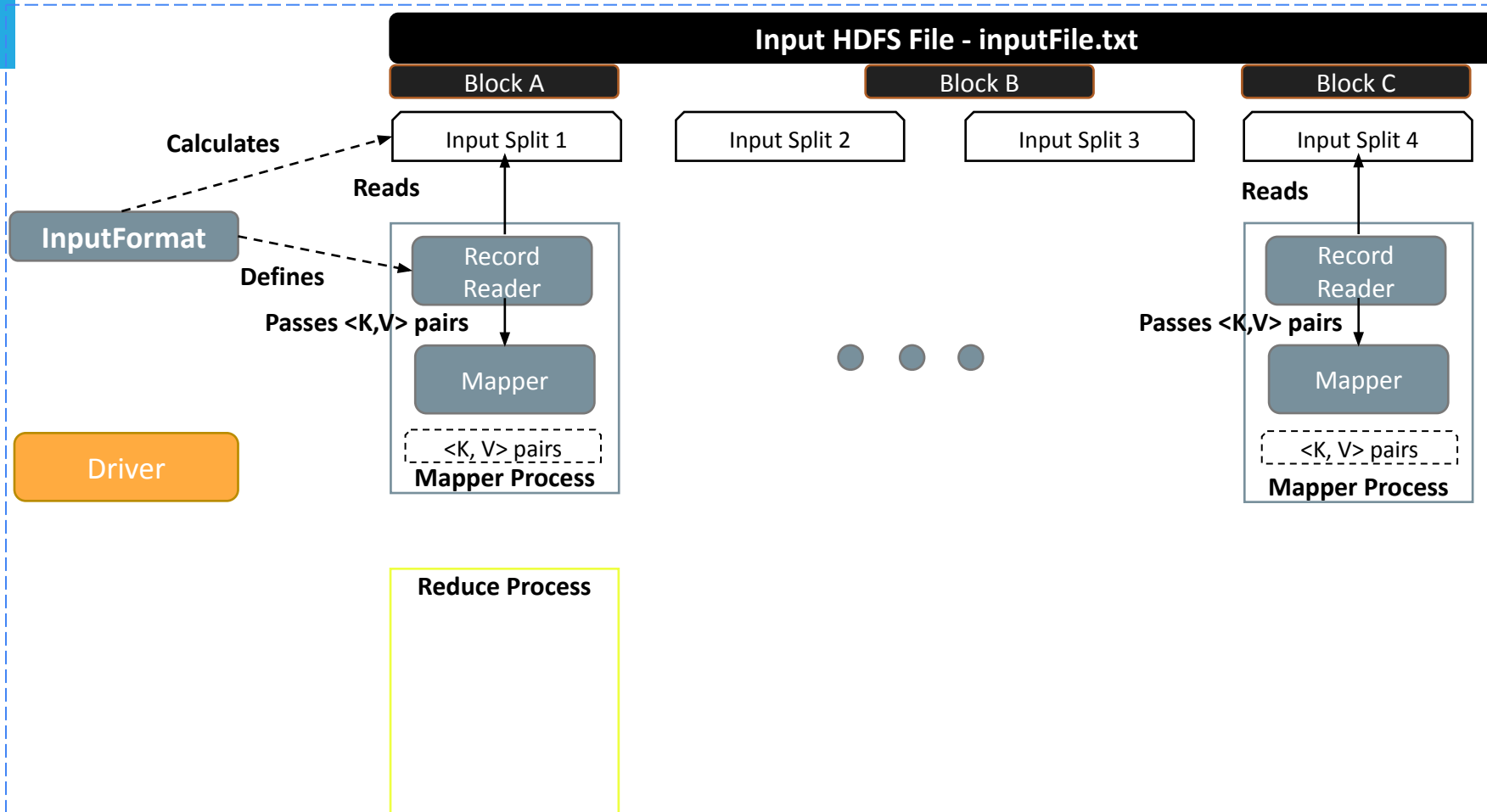
MapReduce

MapReduce Execution Framework



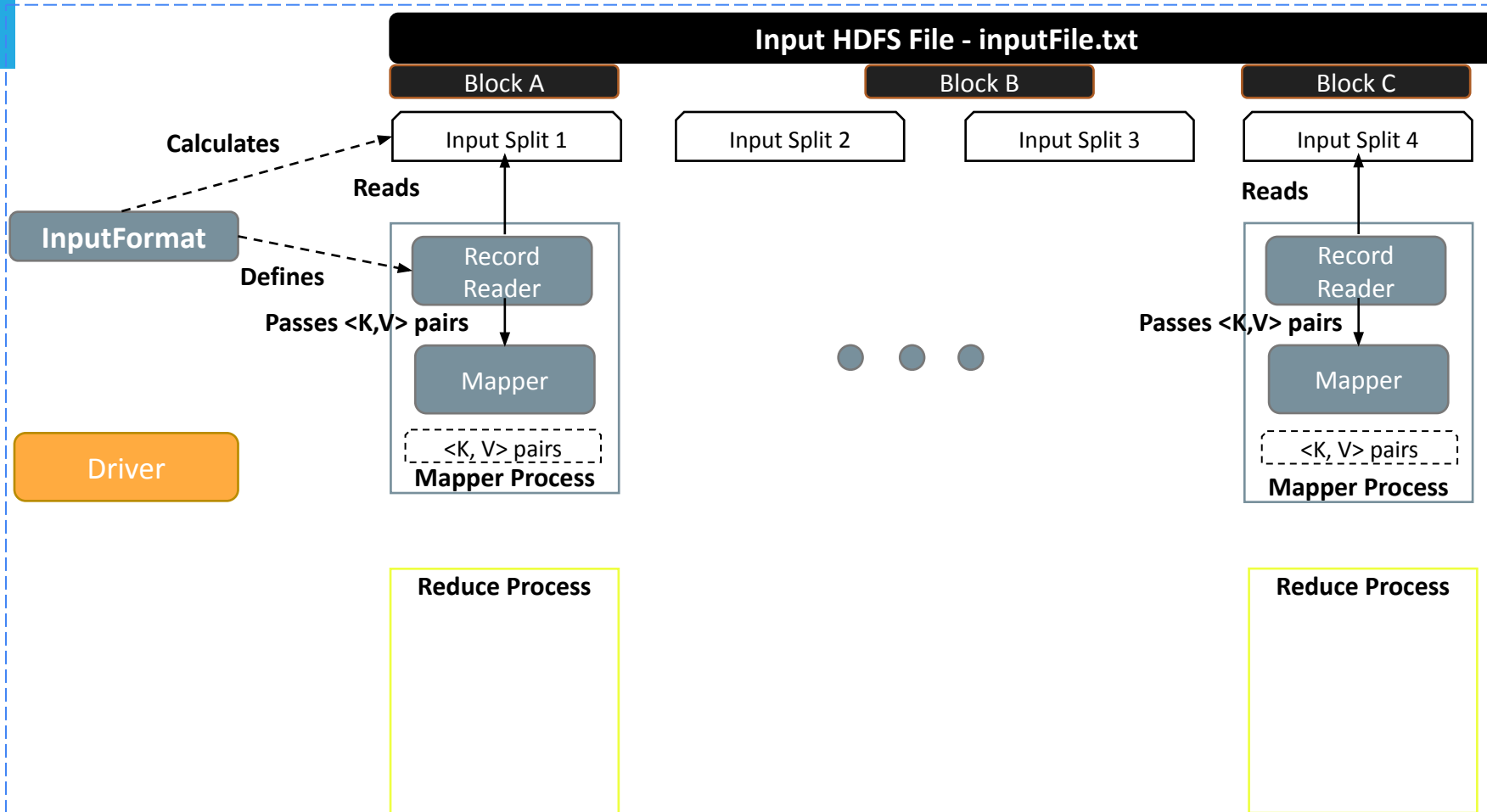
MapReduce

MapReduce Execution Framework



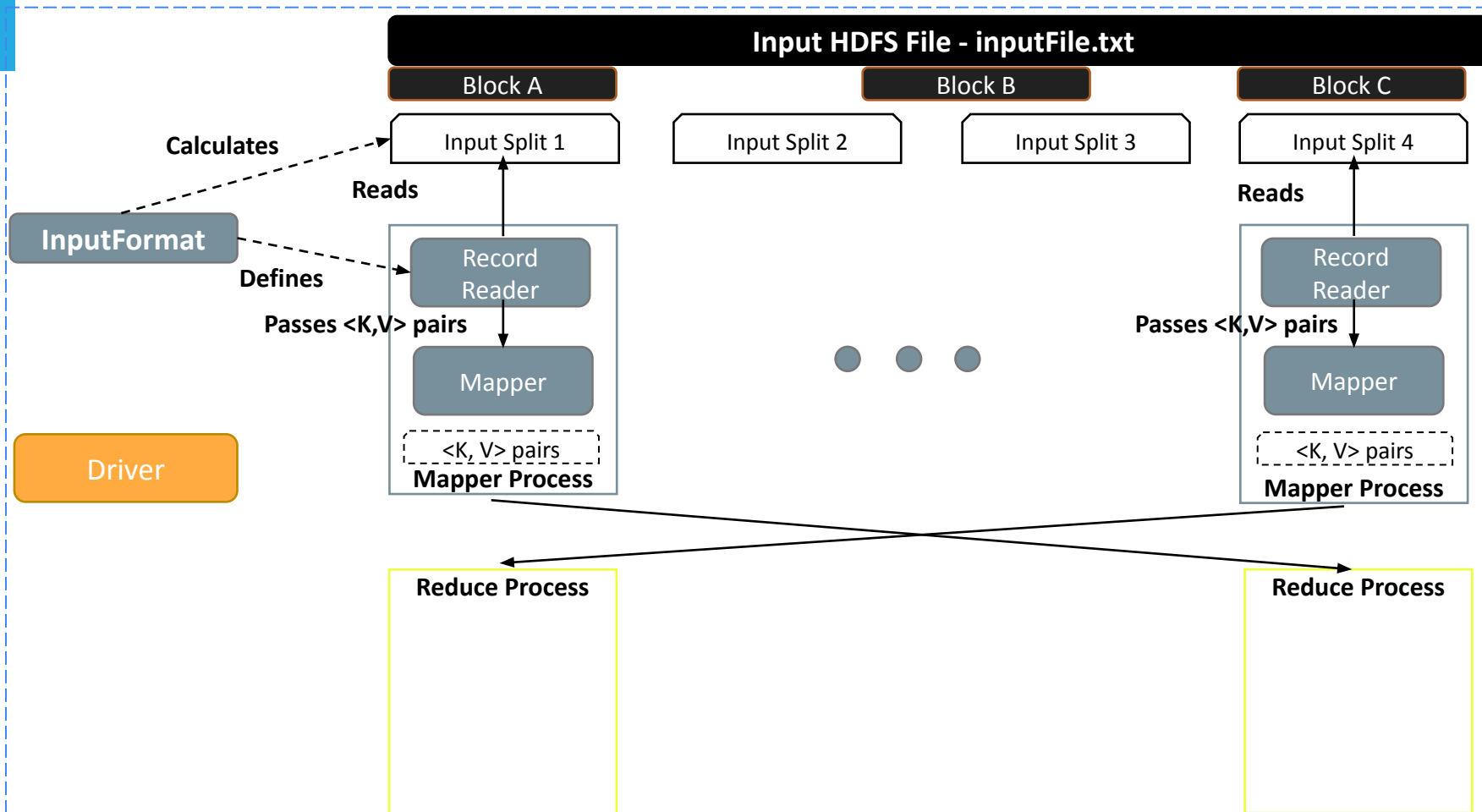
MapReduce

MapReduce Execution Framework



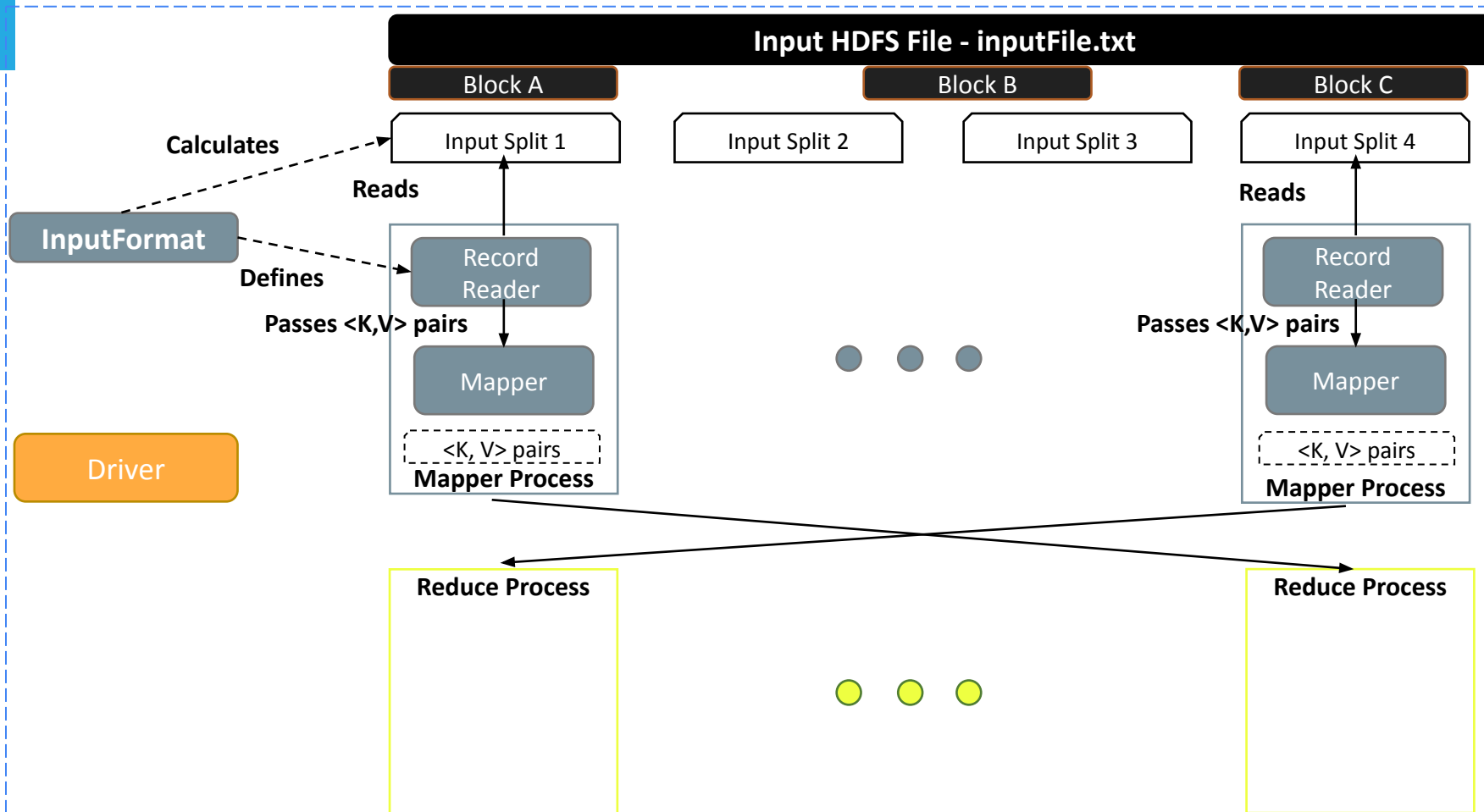
MapReduce

MapReduce Execution Framework



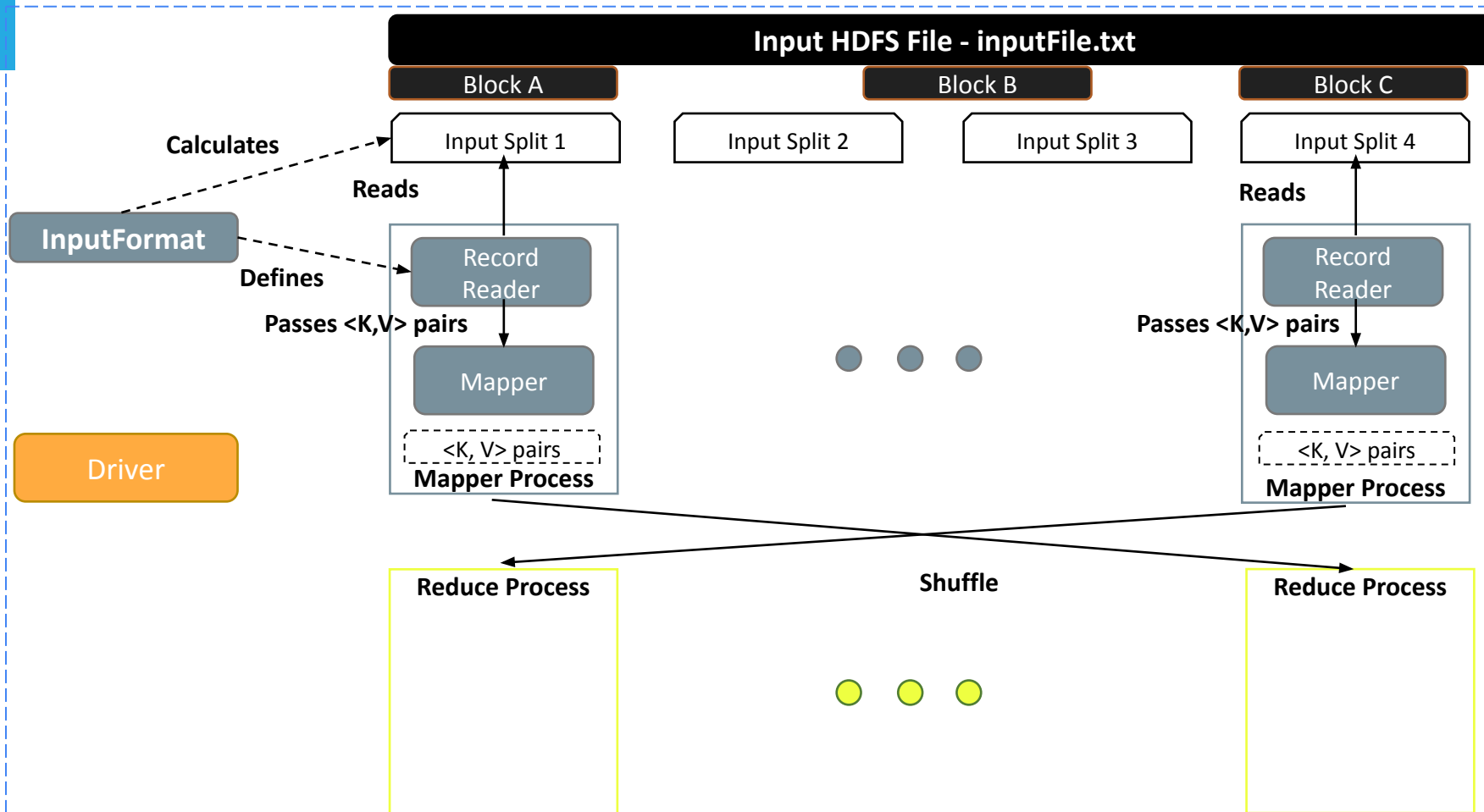
MapReduce

MapReduce Execution Framework



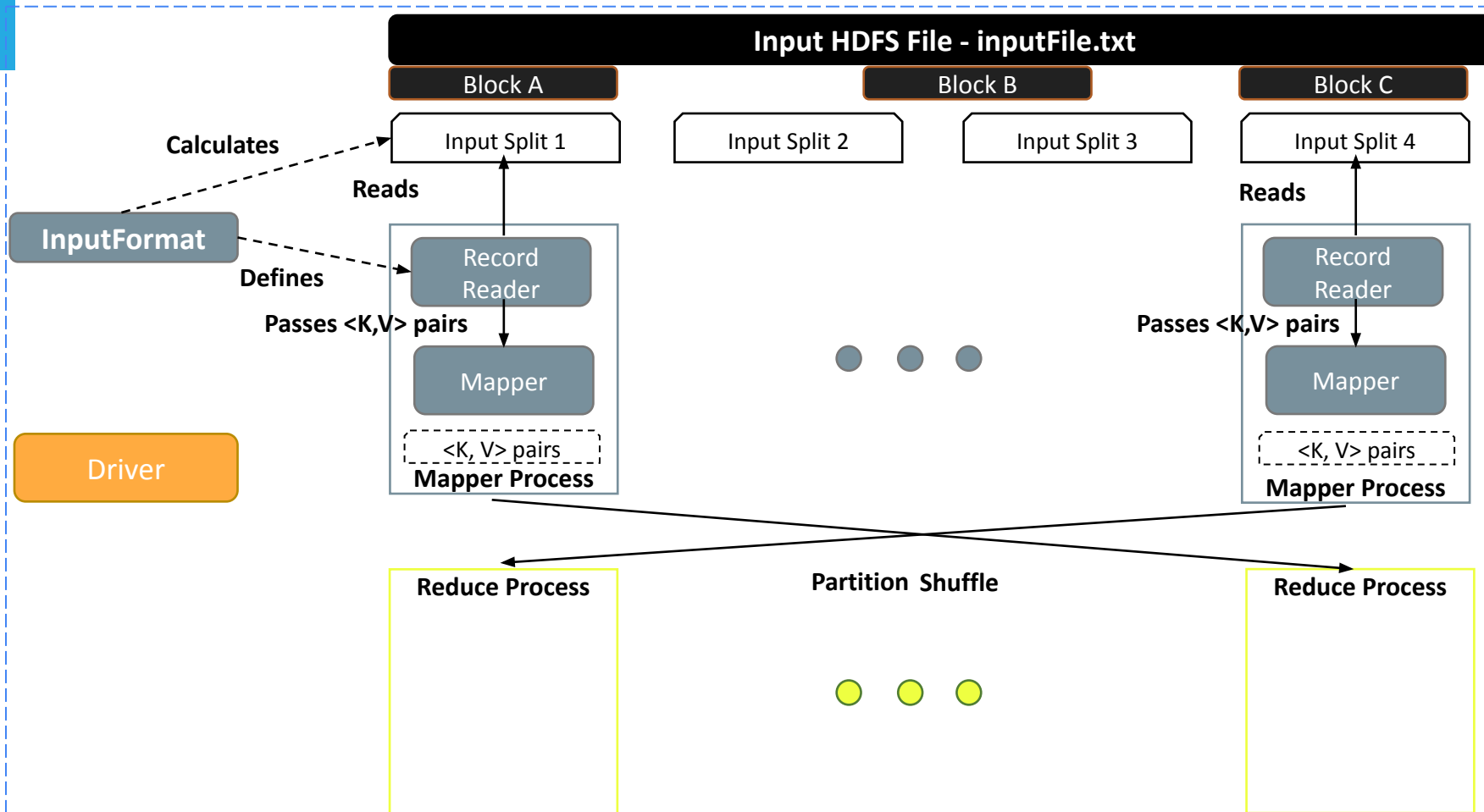
MapReduce

MapReduce Execution Framework



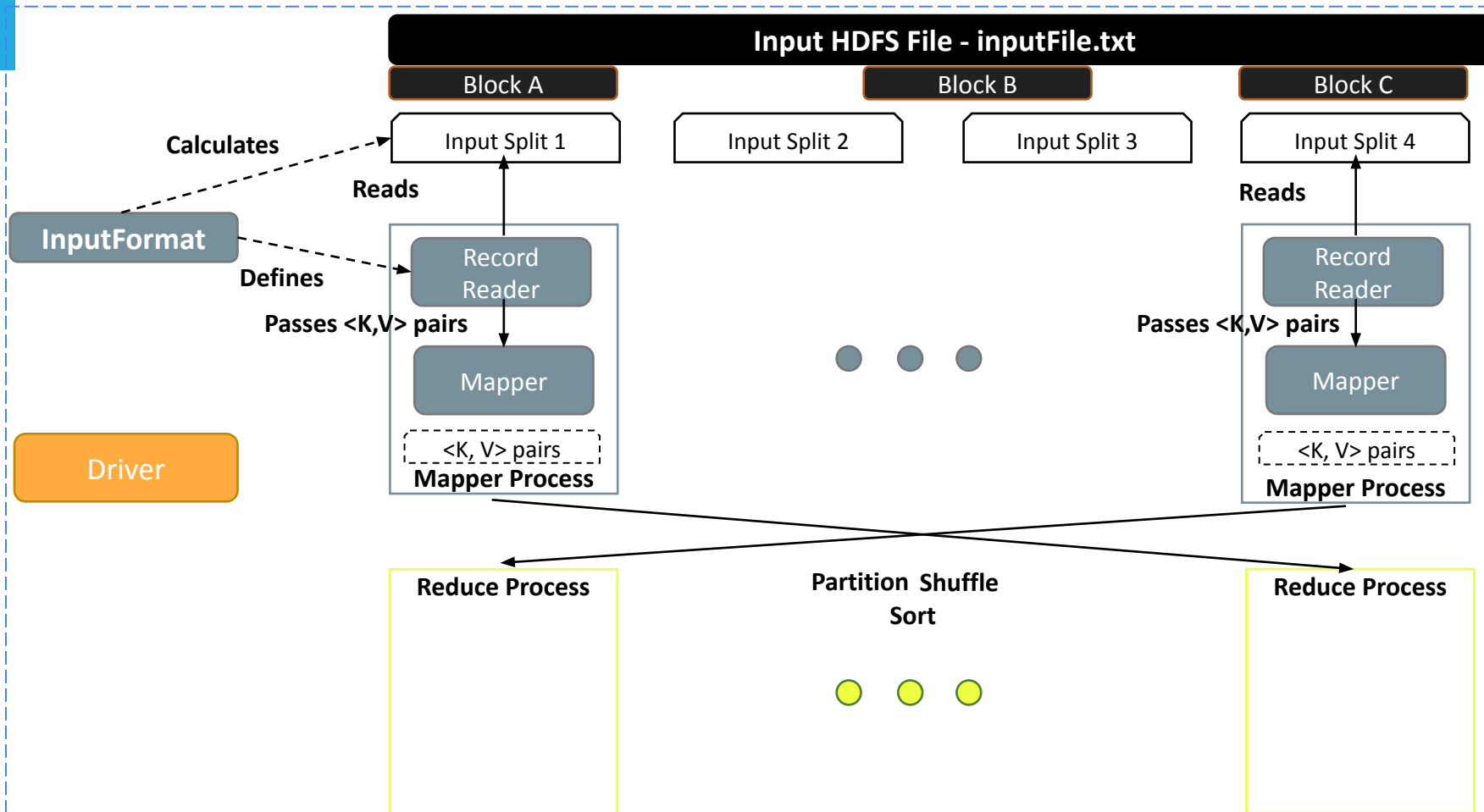
MapReduce

MapReduce Execution Framework



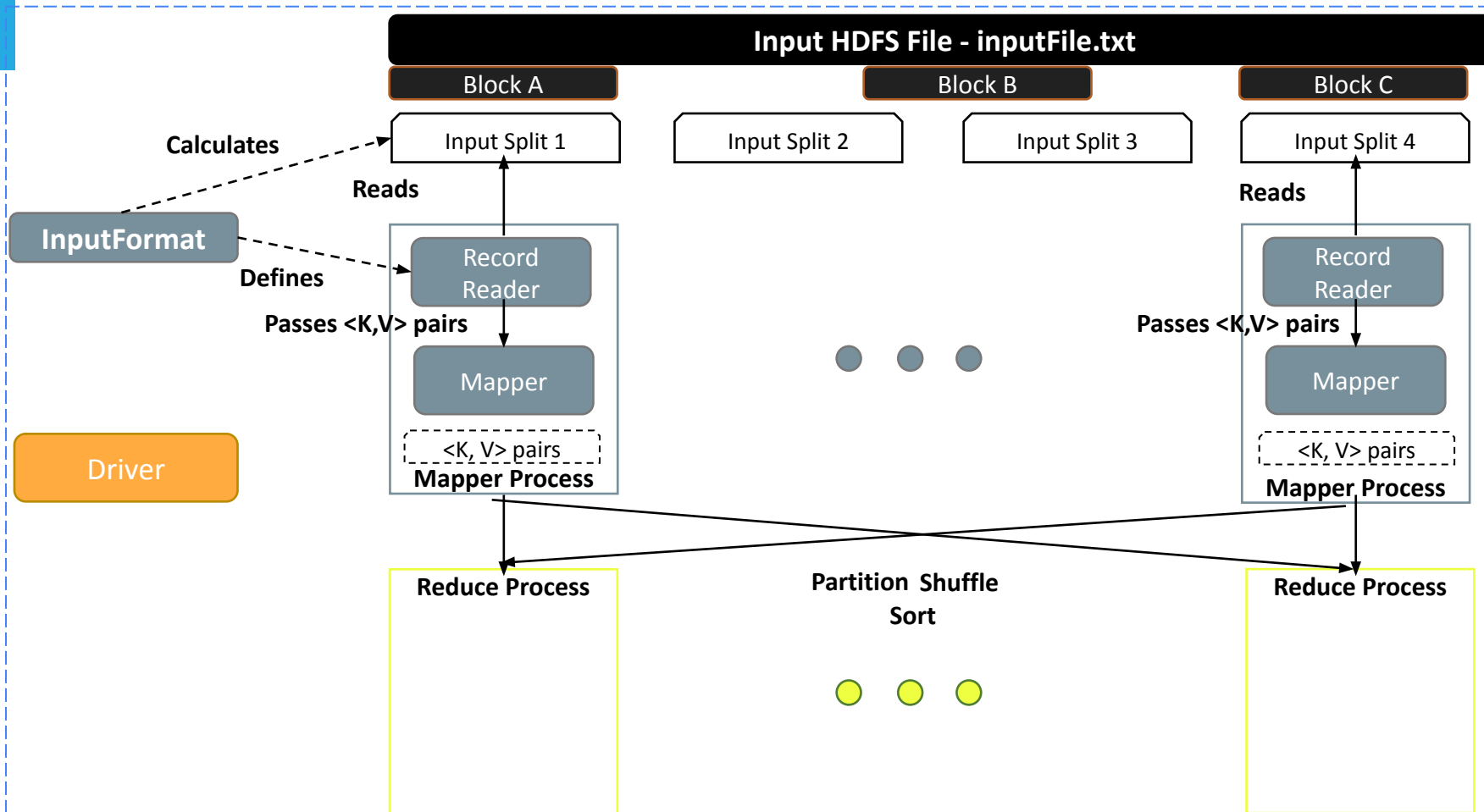
MapReduce

MapReduce Execution Framework



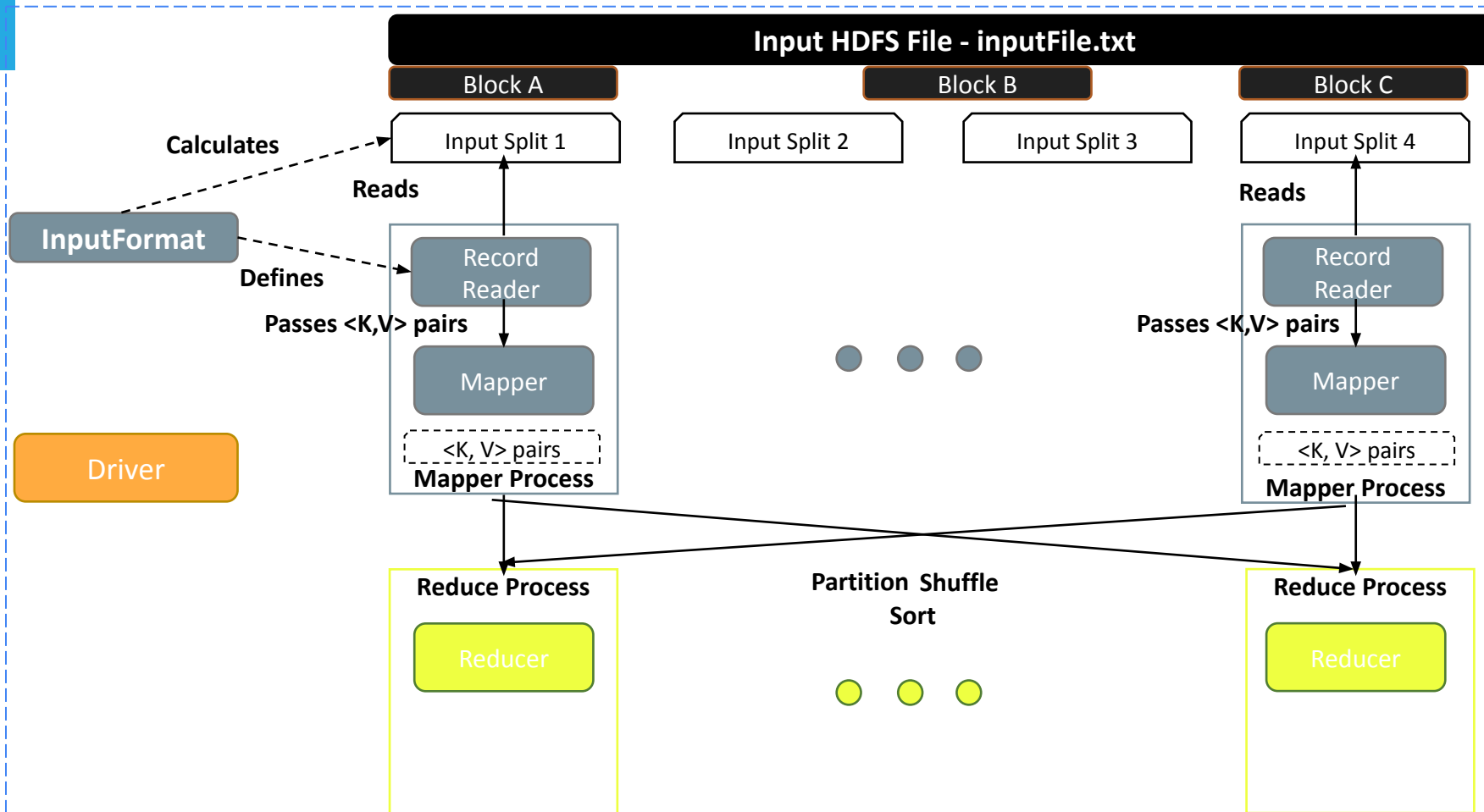
MapReduce

MapReduce Execution Framework



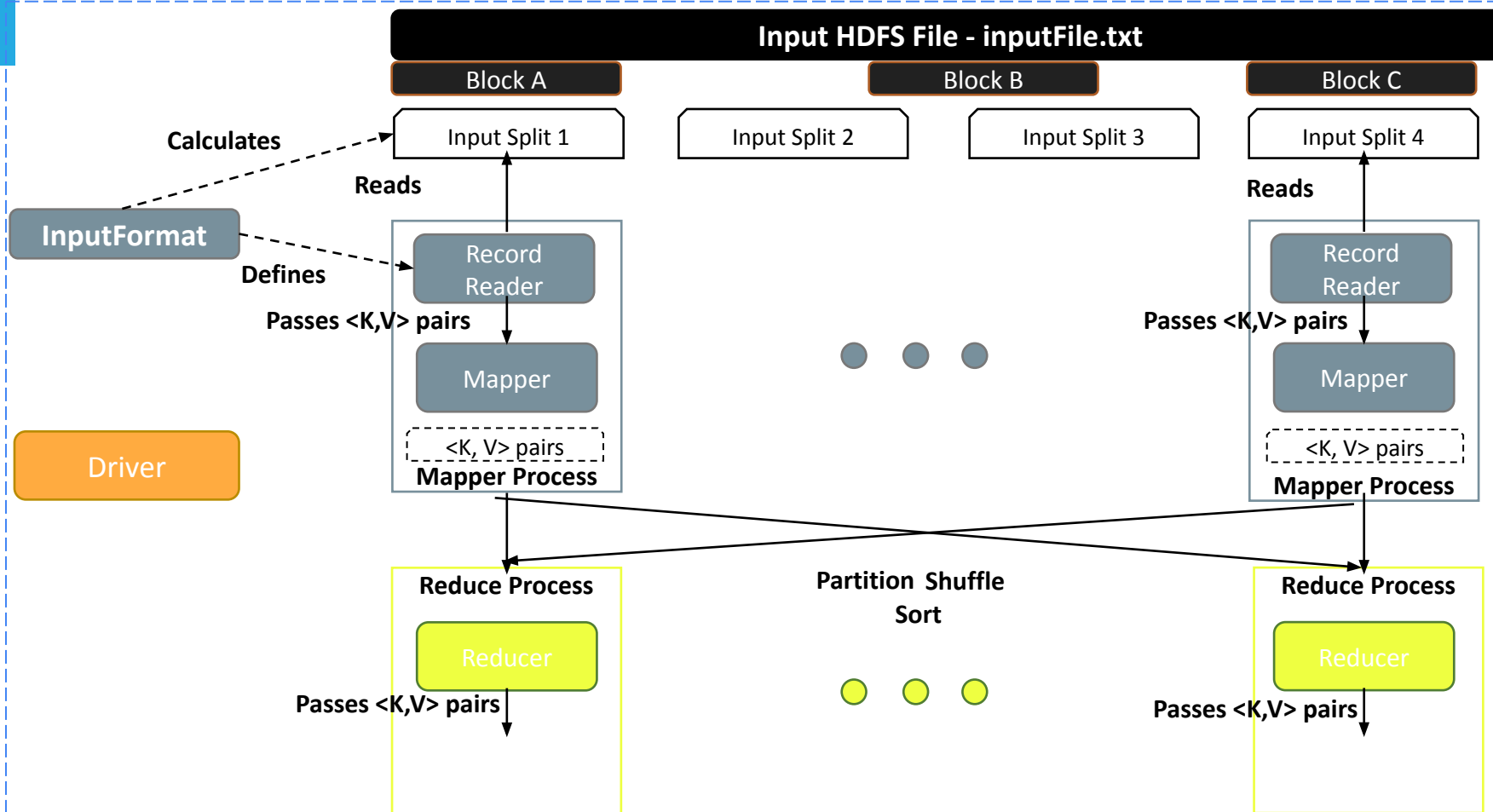
MapReduce

MapReduce Execution Framework



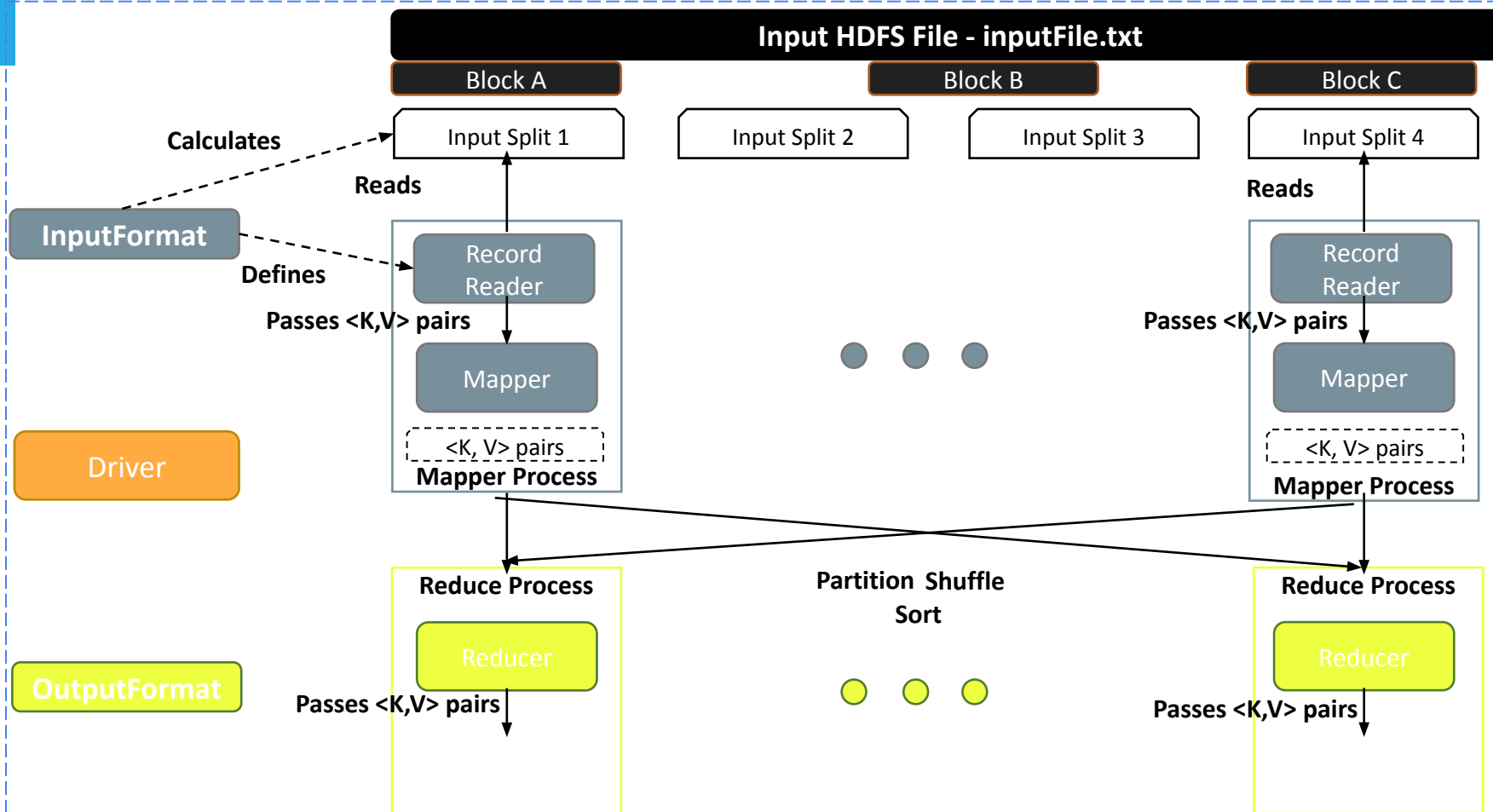
MapReduce

MapReduce Execution Framework



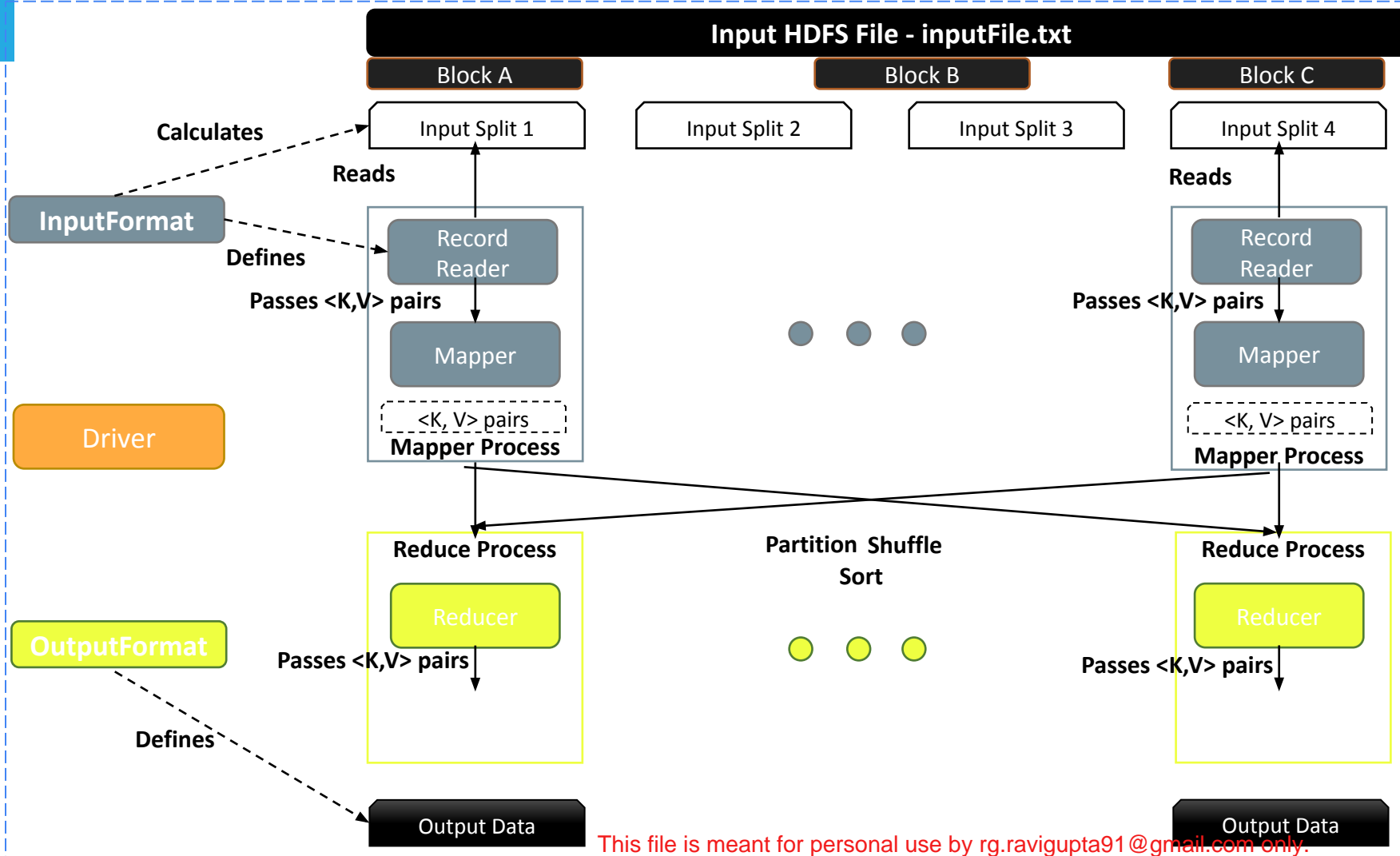
MapReduce

MapReduce Execution Framework



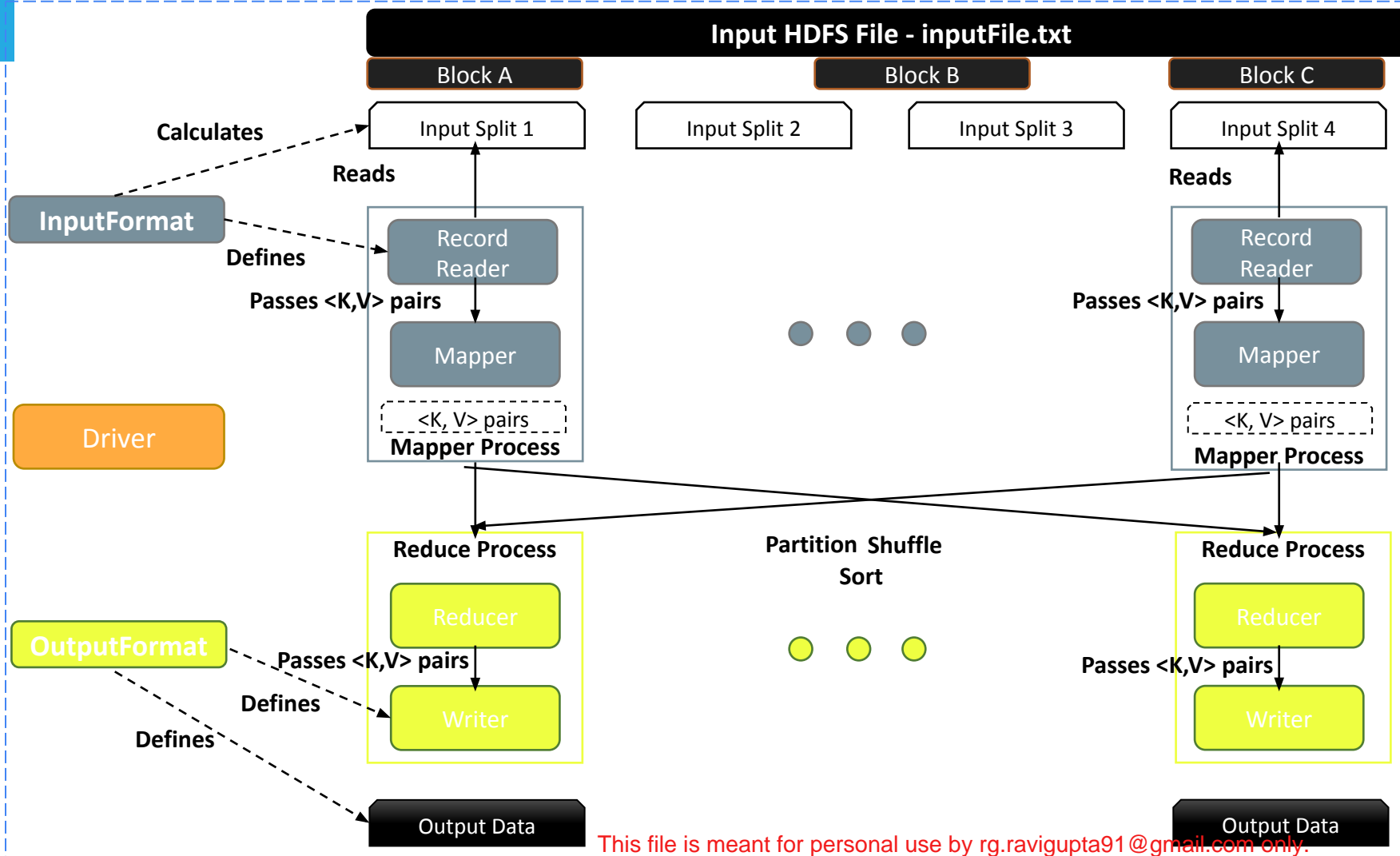
MapReduce

MapReduce Execution Framework



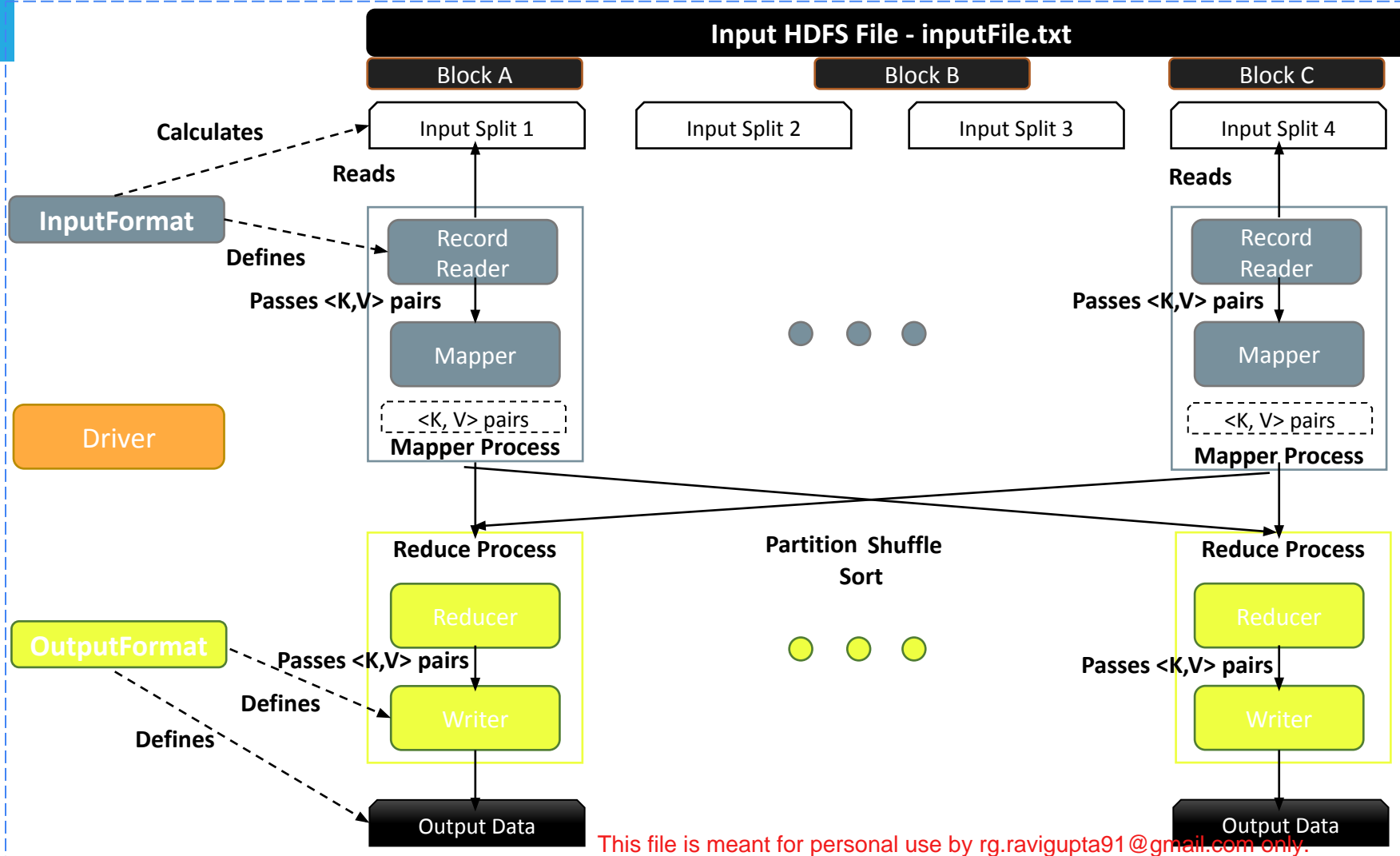
MapReduce

MapReduce Execution Framework



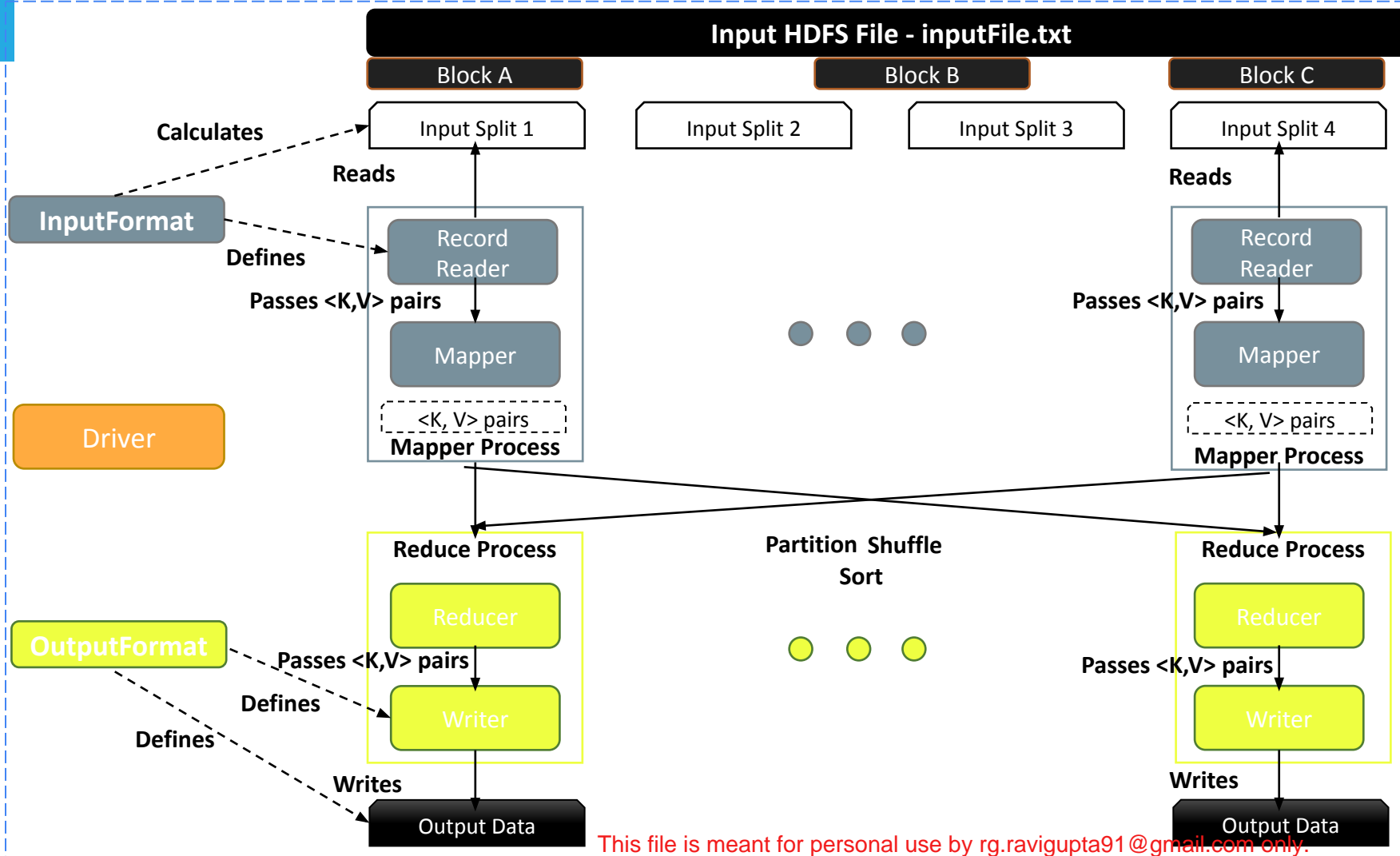
MapReduce

MapReduce Execution Framework



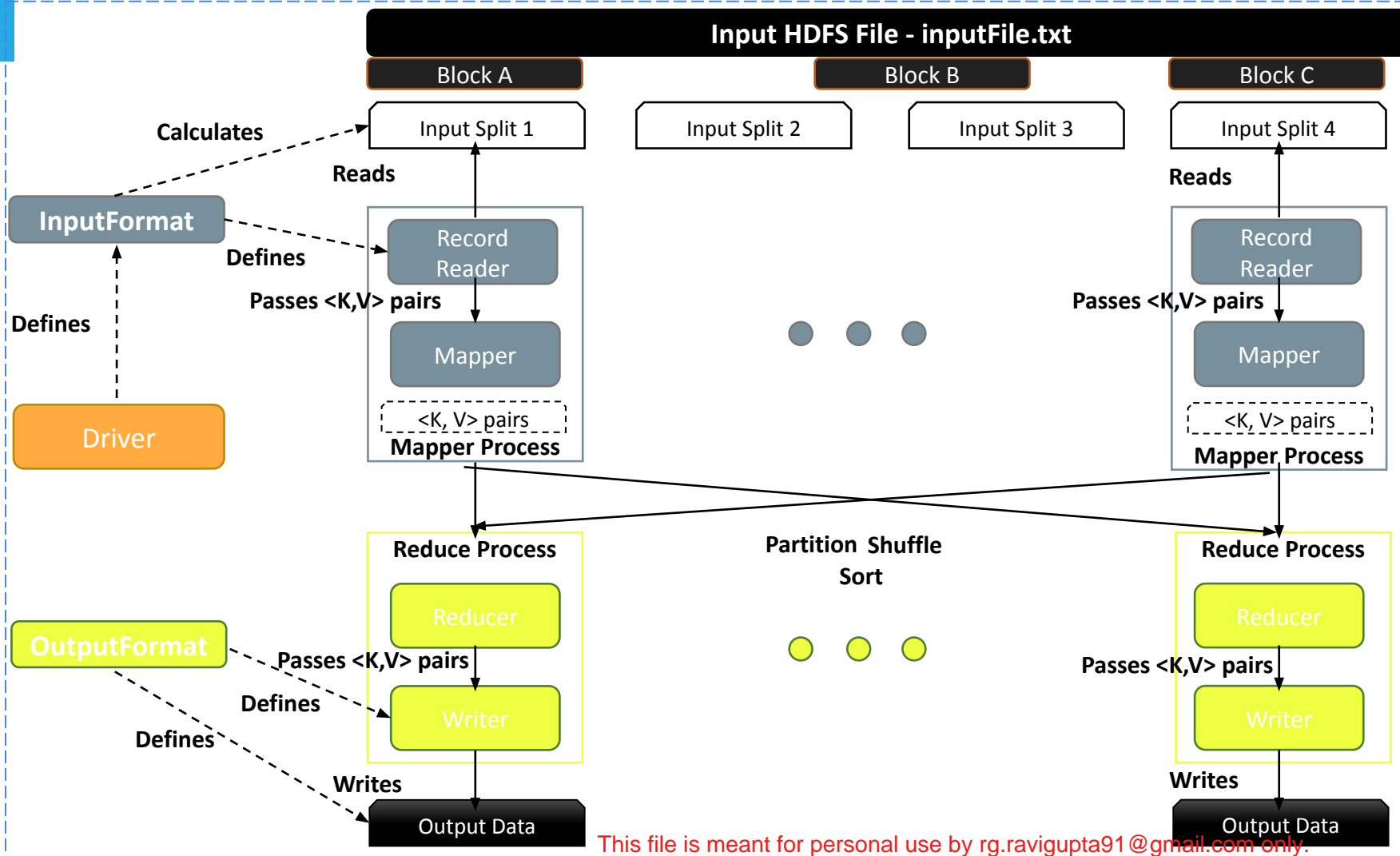
MapReduce

MapReduce Execution Framework



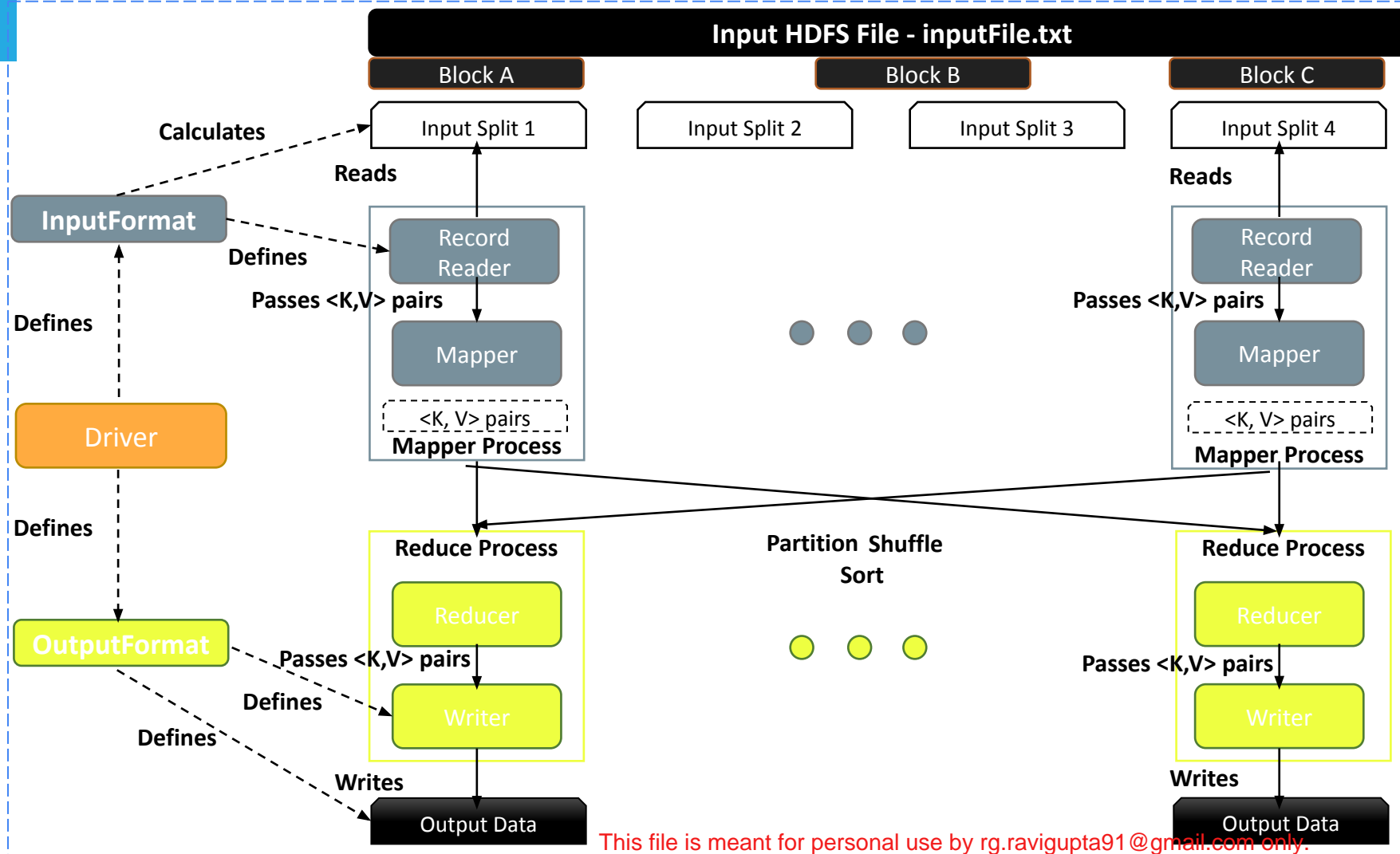
MapReduce

MapReduce Execution Framework



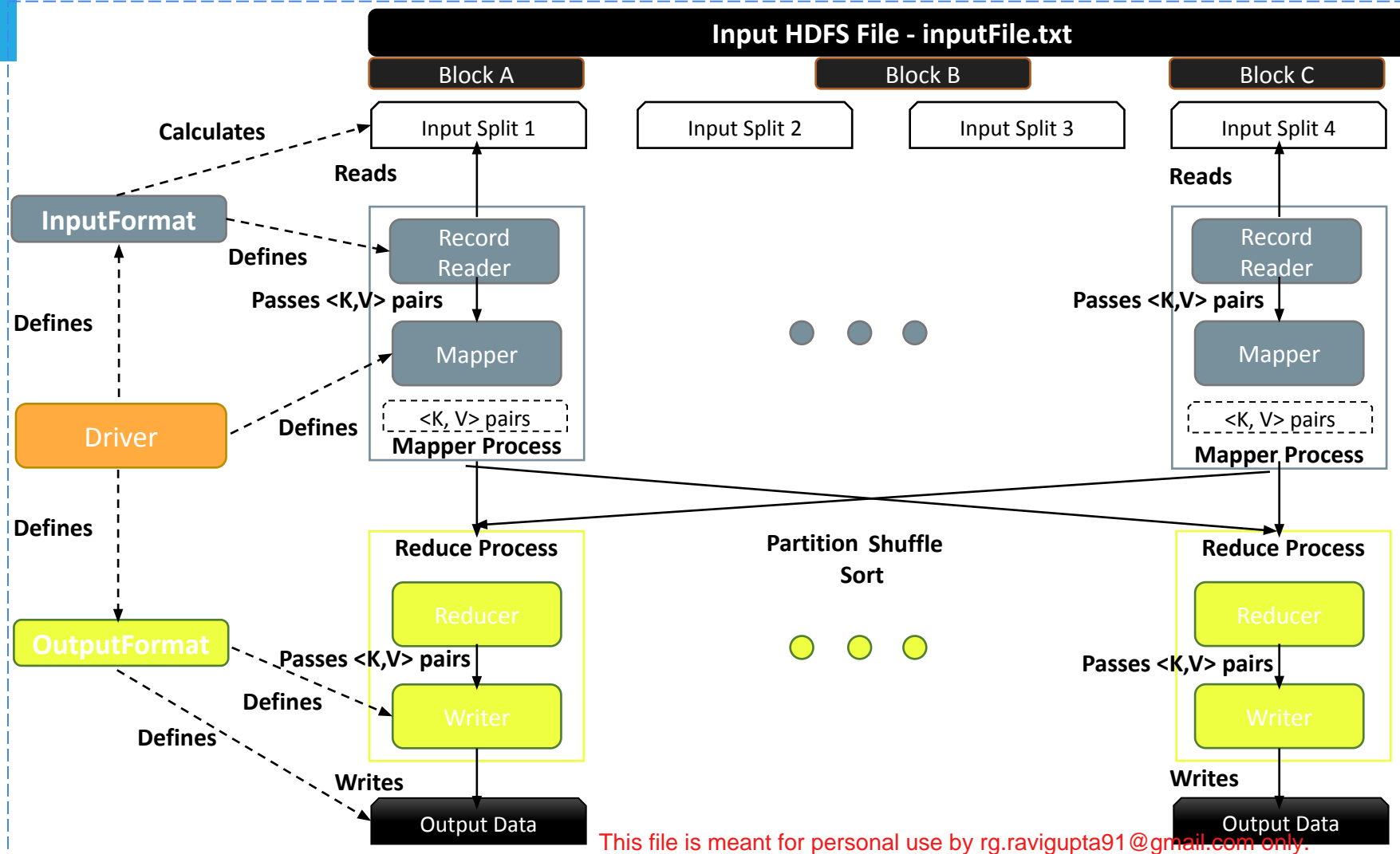
MapReduce

MapReduce Execution Framework



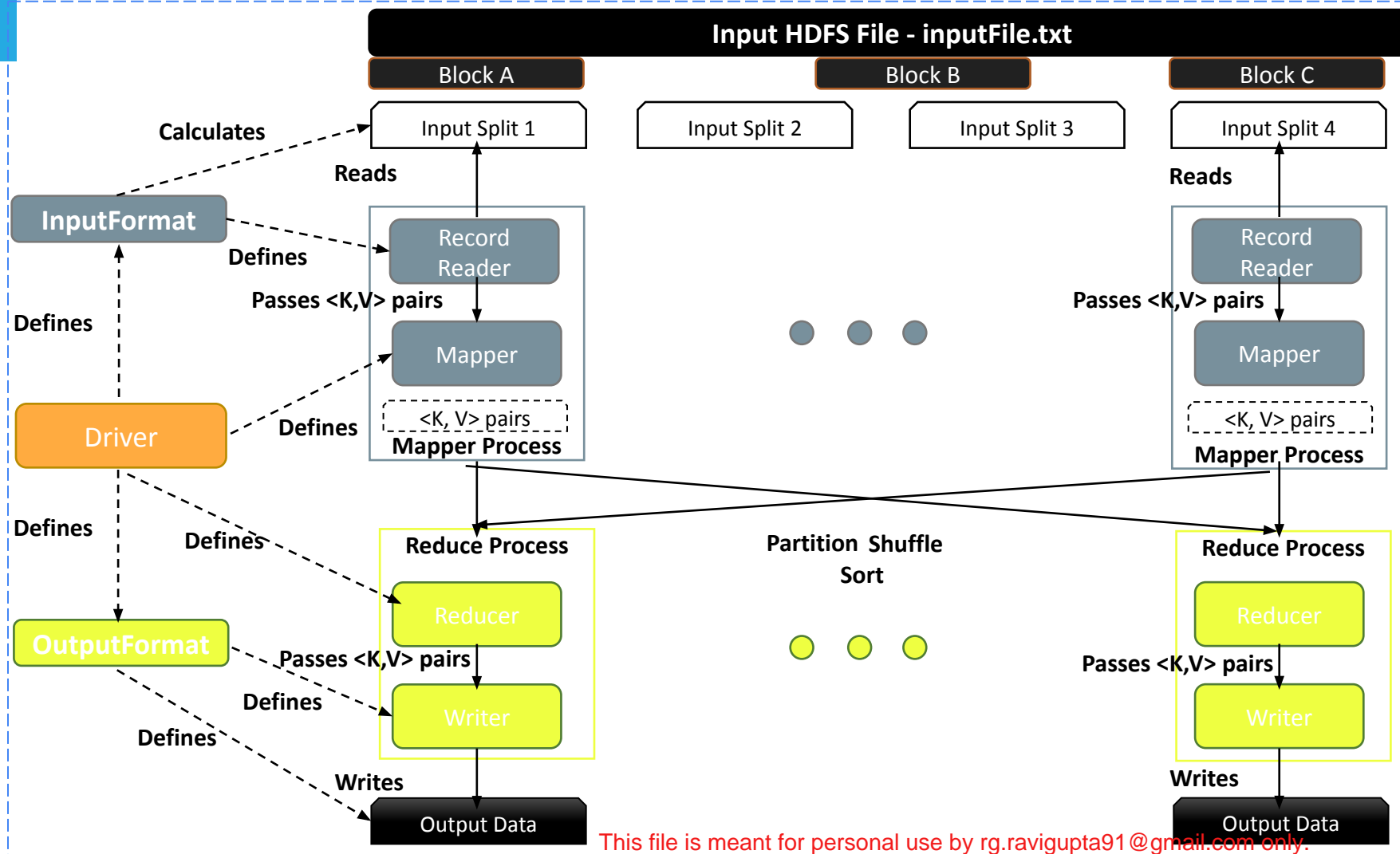
MapReduce

MapReduce Execution Framework



MapReduce

MapReduce Execution Framework





- *MR Job can contain either map reduce both or just mappers*
- *No of Mappers in the Job depends upon the input split*
- *No of reducer in the Job needs to be set by the developer*
- *By default, no of reducer in the Job is just one*
- *Otherwise , recommended value of the reducers in the job is $\frac{1}{4}$ of mappers tasks*
- *Each Map and Reduce task is separate container of JVM*
- *By default , Map and reduce JVM have 1 GB Memory and 1 CPU core*

Thank You