Lecture 3
Big Data Processing Frameworks

Dr. Lydia Wahid

Agenda

- Basic Concepts
- Apache Hadoop
- Hadoop Subprojects
- Hadoop YARN
- Integrating R and Hadoop
- Apache Spark

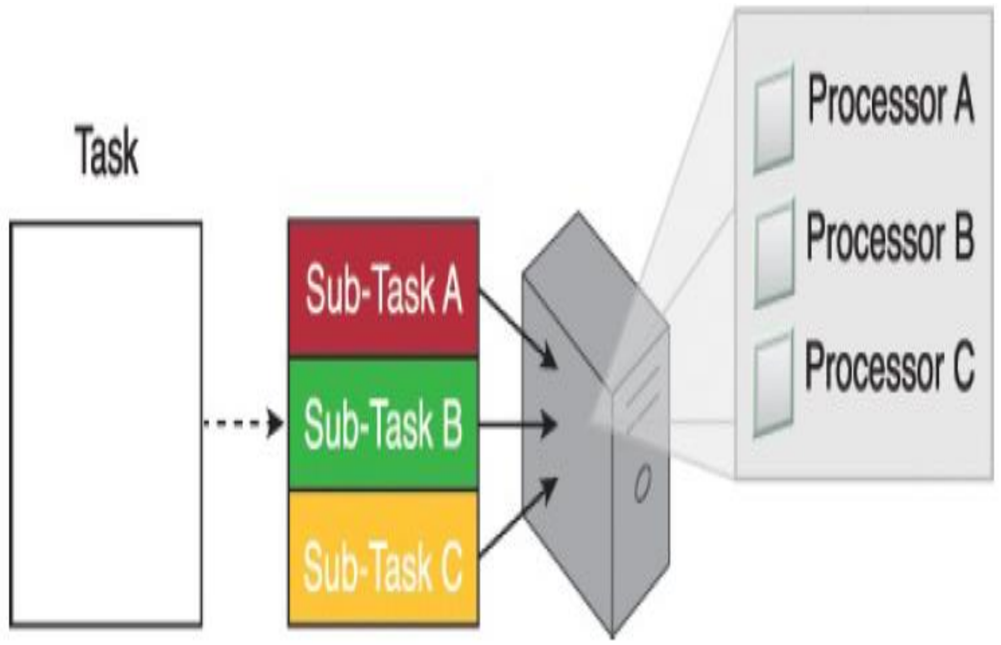
Basic Concepts

Basic Concepts

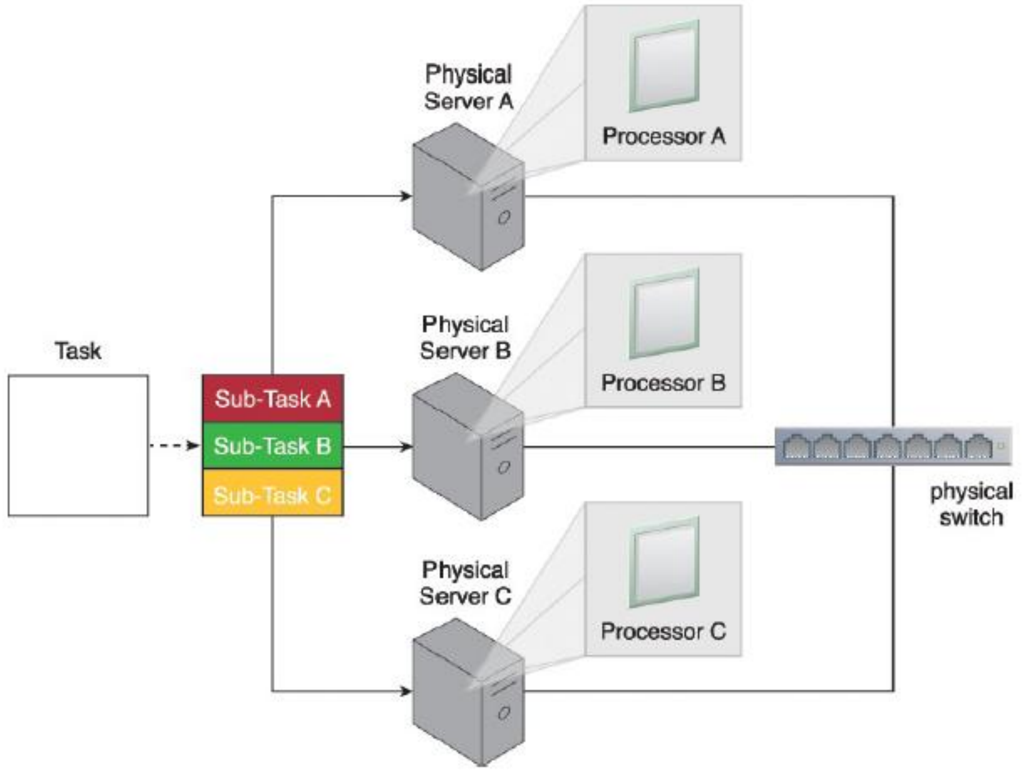
- **Parallel Data Processing vs Distributed Data Processing:**
  - Parallel data processing involves the **simultaneous execution** of multiple sub-tasks that collectively comprise a larger task.
  - The goal is to reduce the execution time by dividing a single larger task into multiple smaller tasks that run concurrently.
  - Although parallel data processing can be achieved through multiple networked machines, **it is more typically achieved within a single machine with multiple processors** or cores.
  - Distributed data processing is always **achieved through physically separate machines** that are networked together as a cluster.

Basic Concepts

Parallel Data Processing



Distributed Data Processing



Basic Concepts

➢ **Batch vs Interactive vs Real-time stream Processing**

- **Batch processing** is the processing of data in groups or batches. No user interaction is required once batch processing is happening.
- **Interactive processing** means that the person needs to provide the computer with instructions while it is doing the processing.
- **Real-time stream processing** is the processing of data at the time the data is generated. Processing times can be measured in microseconds rather than in hours or days.

Apache Hadoop

Apache Hadoop

- Hadoop is an Apache open-source framework written in java that allows distributed processing of large datasets across clusters of computers.

- A cluster is a configuration of nodes that interact to perform a specific task.

- Hadoop is designed to scale up from single server to thousands of machines, each offering local computation and storage.

Apache Hadoop - Architecture

➢ **Hadoop Architecture:**

- It consists of two main components:
    1. Processing/Computation (MapReduce)
        - Capable of processing enormous data in parallel on large clusters of computation nodes. It performs two main tasks: *map* and *reduce*.
    2. Storage (Hadoop Distributed File System)
        - MapReduce consume data from HDFS. HDFS creates multiple replicas of data blocks and distributes them on the nodes in a cluster. This distribution enables reliable and extremely rapid computations.

Apache Hadoop - Architecture

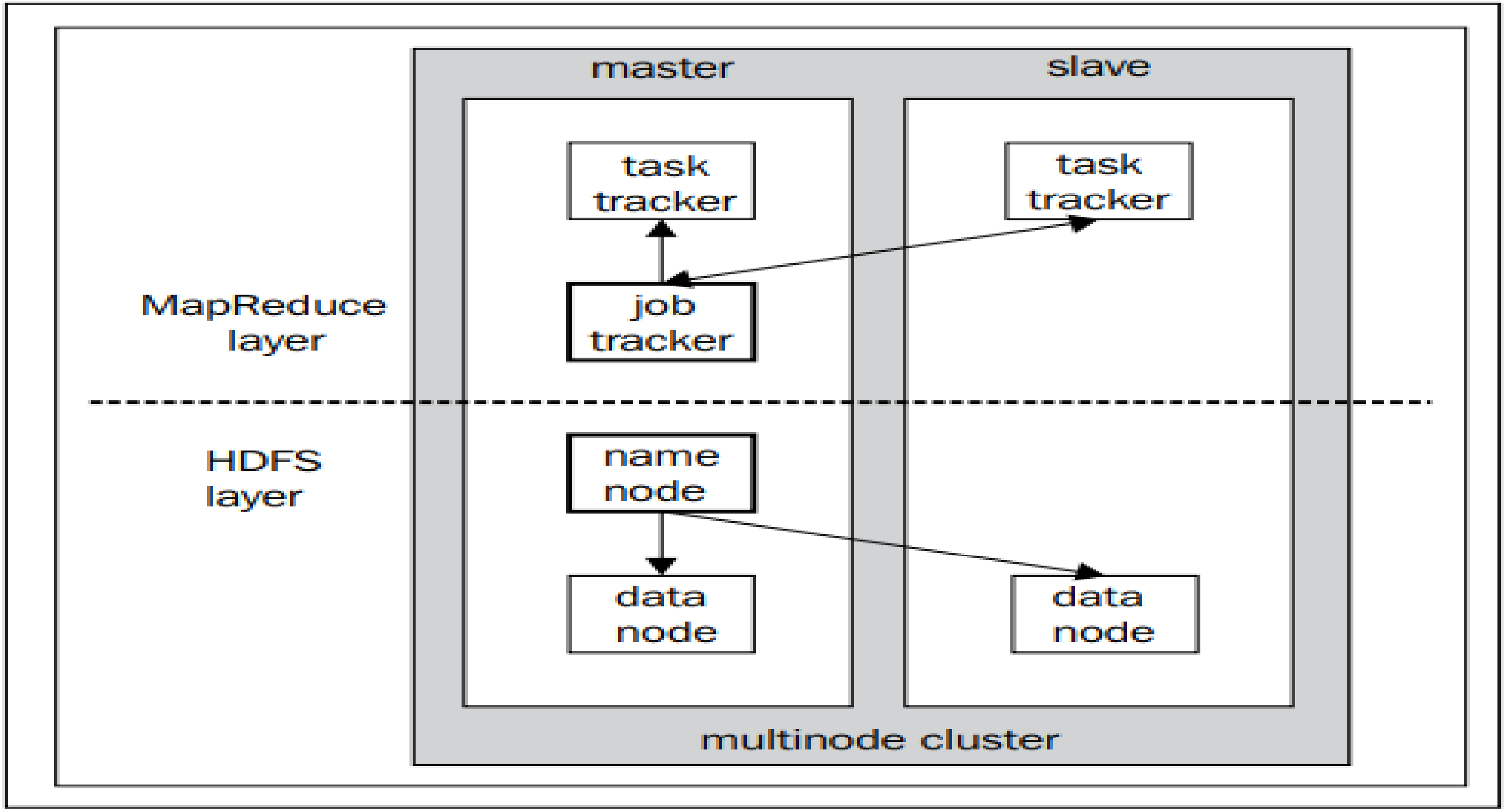
- MapReduce is managed with ***master-slave*** architecture included with the following components:
  - **JobTracker:** This is the *master* node of the MapReduce system, which manages the jobs and resources in the cluster. The JobTracker tries to schedule the maps to specific nodes in the cluster, ideally the nodes that have the data.
  - **TaskTracker:** These are the *slaves* that are deployed on each machine. They are responsible for running the map and reduce tasks as instructed by the JobTracker.

Apache Hadoop - Architecture

- Hadoop Distributed File System (HDFS) is managed with the ***master-slave*** architecture included with the following components:
  - **NameNode:** This is the *master* of the HDFS system. It maintains the file system tree and the metadata for all the files and directories present in the system.
  - **DataNode:** These are *slaves* that are deployed on each machine and provide actual storage. They are responsible for serving read-and-write data requests for the clients. The internal mechanism of HDFS divides the file into one or more blocks; these blocks are stored in a set of data nodes.

Note: **HDFS** is not a **database**, but it is a distributed file system that can store huge volume of data sets across a cluster of computers to be processed

Apache Hadoop - Architecture



The HDFS and MapReduce architecture

Apache Hadoop - Architecture

➢ **How MapReduce tasks are assigned to specific nodes in the cluster:**

1. Client applications submit jobs to the Jobtracker.
2. The JobTracker talks to the NameNode to determine the location of the data (DataNode)
3. The JobTracker locates TaskTracker nodes with available slots at or near the data (DataNode)
4. The JobTracker submits the work to the chosen TaskTracker nodes.
5. The TaskTracker nodes are monitored. If they do not submit signals often enough, they are considered to have failed and the work is scheduled on a different TaskTracker.
6. A TaskTracker will notify the JobTracker when a task fails. The JobTracker 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 TaskTracker as unreliable.
7. When the work is completed, the JobTracker updates its status.
8. Client applications can poll the JobTracker for information.

Apache Hadoop – HDFS Features

- **High Scalability -** HDFS is highly scalable as it can have hundreds of nodes in a single cluster.
- **Handling Huge datasets** – Due to the high scalability, HDFS can manage applications having huge datasets.
- **Distributed data storage -** This is one of the most important features of HDFS that makes Hadoop very powerful. Here, data is divided into multiple blocks and stored into nodes.
- **Hardware at data** – A requested task is done efficiently as the computation takes place near the data. This reduces the network traffic and increases the throughput, especially where huge datasets are involved.

Apache Hadoop – HDFS Features

- **Replication -** Due to some unfavorable conditions, the node containing the data may be lost. To overcome such problems, HDFS always maintains the copy of data on more than one machine.
- **Fault tolerance -** In HDFS, the fault tolerance signifies the robustness of the system in the event of failure. Due to Replication, HDFS is highly fault-tolerant that if any machine fails, the other machine containing the copy of that data automatically become active.
- **Portability -** HDFS is designed in such a way that it can be easily portable from platform to another.
- **Write-once-Read-many -** The application that runs on HDFS require to follow the write-once-read-many approach. So, a file once created need not to be changed. However, it can be appended and truncated.

Apache Hadoop – Hadoop Operation Modes

- **Hadoop can run in 3 different modes:**
  1. Standalone Mode (**Single** machine **Single** process):
     By default, Hadoop is configured to run in a non-distributed mode. It runs as a single Java process. Instead of HDFS, this mode utilizes the local file system. This mode is useful for debugging.
  2. Pseudo-Distributed Mode (**Single** machine **Multiple** processes):
     Hadoop can also run on a single machine in a Pseudo Distributed mode. In this mode, each Hadoop process runs as separate java process. Here HDFS is utilized for input and output.
  3. Fully Distributed Mode (**Multiple** machines **Multiple** processes):
     This is the production mode of Hadoop. In this mode, Hadoop is distributed across multiple machines. Therefore, separate Java processes are present. This mode offers fully distributed computing capability, reliability, fault tolerance and scalability.

Hadoop Subprojects

Apache Hadoop – Hadoop Subprojects

- **Mahout:**
  - It is a popular data mining library. It includes the most popular data mining scalable machine learning algorithms. Also, it is a scalable machine-learning library.
  - The following are some companies that are using Mahout: Amazon, Twitter, Yahoo, and LucidWorks Big Data (This is an analytics firm, which uses Mahout for clustering, duplicate document detection, phase extraction, and classification).

Apache Hadoop – Hadoop Subprojects

- **Apache Hbase:**
  - It is the Hadoop database, a distributed, scalable, big data store. This allows random, real-time read/write access to Big Data. Apache HBase is an open-source, distributed, non-relational database modeled after Google's Bigtable.
  - The following are some companies using HBase: Yahoo, Twitter, and stumbleupon (This is a personalized recommender system, realtime data storage, and data analytics platform).

Apache Hadoop – Hadoop Subprojects

- **Apache Hive**
  - It is a data warehouse software developed by Facebook that facilitates reading, writing, and managing large datasets residing in distributed storage using SQL.
  - DWs are central repositories of integrated data from one or more disparate sources.
  - It allows users to fire queries in SQL-like languages, such as **HiveQL**.

Apache Hadoop – Hadoop Subprojects

- **Apache Pig:**
  - It is platform for creating programs that run on Apache Hadoop. The language for this platform is called **Pig Latin**.
  - Apache Pig has been developed by Yahoo. Currently, Yahoo and Twitter are the primary users of Pig.

Apache Hadoop – Hadoop Subprojects

- **Apache Sqoop:**
  - Apache Sqoop is a mutual data tool for importing data from the relational databases to Hadoop HDFS and exporting data from HDFS to relational databases.
  - It works together with most modern relational databases, such as Microsoft SQL Server, MySQL, and Oracle.

Apache Hadoop – Hadoop Subprojects

- **Apache Solr:**
  - It is an open-source enterprise search platform.
  - Solr is highly reliable, scalable and fault tolerant, providing distributed indexing, replication and load-balanced querying, automated failover and recovery, centralized configuration and more.
  - This allows building web application with powerful search capabilities.
  - Solr powers the search and navigation features of many of the world's largest internet sites

Apache Hadoop – Hadoop Subprojects

- **Apache Zookeeper:**
  - It is an effort to develop and maintain an open-source server which enables highly reliable distributed coordination.
  - It is a centralized service for maintaining configuration information, naming, and providing distributed synchronization.

Hadoop YARN

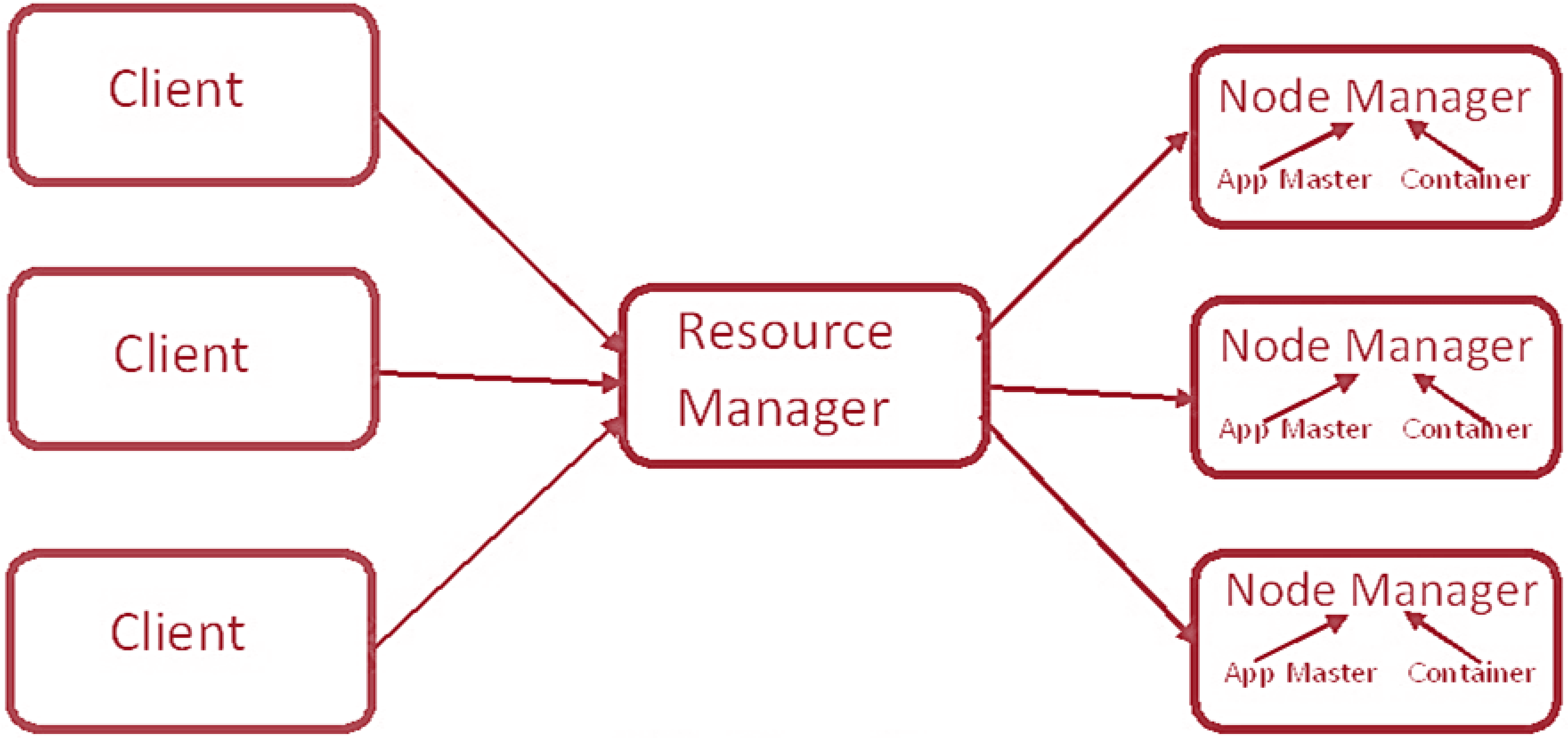
Hadoop YARN

- YARN stands for "***Yet Another Resource Negotiator***".
- It was introduced in Hadoop version 2.0.
- YARN allows different data processing engines like **batch**, **interactive**, and **real-time stream** processing of data.
- In the previous version of Hadoop that is Hadoop version 1.0, MapReduce performs both the task of processing and resource management by itself.
- The JobTracker in v1.0 is the single master that allocates resources for applications, performs scheduling for demand and also monitors the jobs of processing in the system.

Hadoop YARN

- The basic idea behind YARN is separating MapReduce from Resource Management and Job scheduling.
- Thus, YARN is now responsible for Job scheduling and Resource Management.
- Through its various components, YARN can dynamically allocate various resources and schedule the application processing.

Hadoop YARN - Architecture



Hadoop Yarn architecture

Hadoop YARN - Architecture

➢The main components of YARN architecture include:

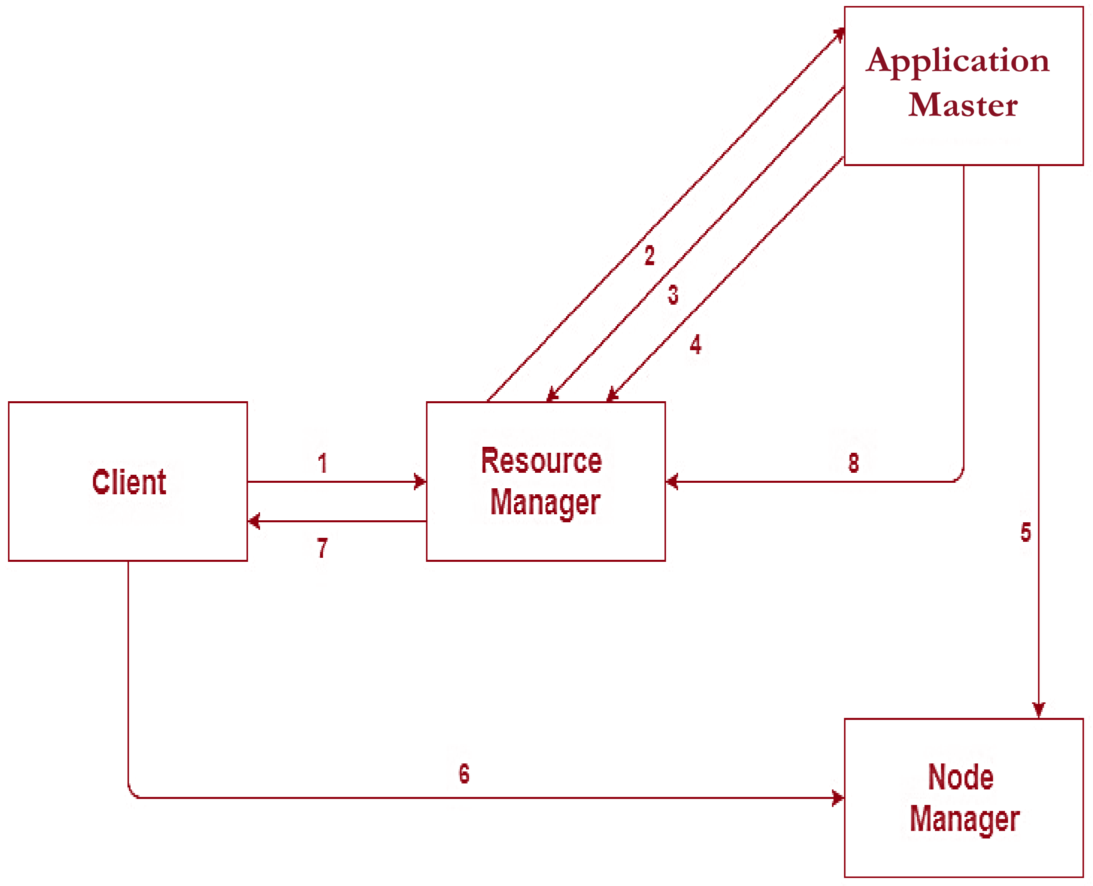
1. **Client:** It submits map-reduce jobs.
2. **Resource Manager:** It is the master of YARN and is responsible for **resource assignment and management** among all the applications. Whenever it receives a processing request, it forwards it to the corresponding node manager and allocates resources for the completion of the request accordingly.
3. **Container:** In the Container, there are the physical resources like a disk, CPU cores, RAM.

Hadoop YARN - Architecture

3. **Application Master:** An application is a single job submitted to a framework. The application master is responsible for **negotiating resources** with the resource manager, **tracking the status** and **monitoring progress** of a single application.
4. **Node manager:** It **sends each node's health status** to the Resource Manager, stating if the node process has finished working with the resource. Node manager is also responsible for **monitoring resource usage** by individual Container and reporting it to the Resource manager.

Hadoop YARN - Application workflow

1. Client submits an application
2. The Resource Manager allocates a container to start the Application Master
3. The Application Master registers itself with the Resource Manager
4. The Application Master negotiates containers from the Resource Manager
5. The Application Master notifies the Node Manager to launch containers
6. Application code is executed in the container
7. Client contacts Resource Manager or Application Master to monitor application's status
8. Once the processing is complete, the Application Master un-registers with the Resource Manager



Hadoop YARN - Features

- **Multi-tenancy:** YARN has allowed access to multiple data processing engines such batch, interactive, and real-time stream processing.
- **Scalability:** The scheduler in Resource manager of YARN architecture allows Hadoop to extend and manage thousands of nodes and clusters.
- **Cluster utilization:** YARN allocates all cluster resources in an efficient and dynamic manner, which leads to better utilization.
- **Compatibility:** YARN supports the existing map-reduce applications without disruptions thus making it compatible with Hadoop 1.0 as well.

Hadoop Version 2.0 vs Version 1.0

Criteria	Version 2.0	Version 1.0
Components	• HDFS • MapReduce • YARN	• HDFS • MapReduce
Type of processing	Real-time, batch, and interactive processing	Batch processing
Suitable for	MapReduce and non-MapReduce applications	Only MapReduce applications
Managing cluster resource	Done by YARN	Done by JobTracker
Cluster resource optimization	Excellent due to central resource management	Average due to fixed Map and Reduce slots

Integrating R and Hadoop

Integrating R and Hadoop

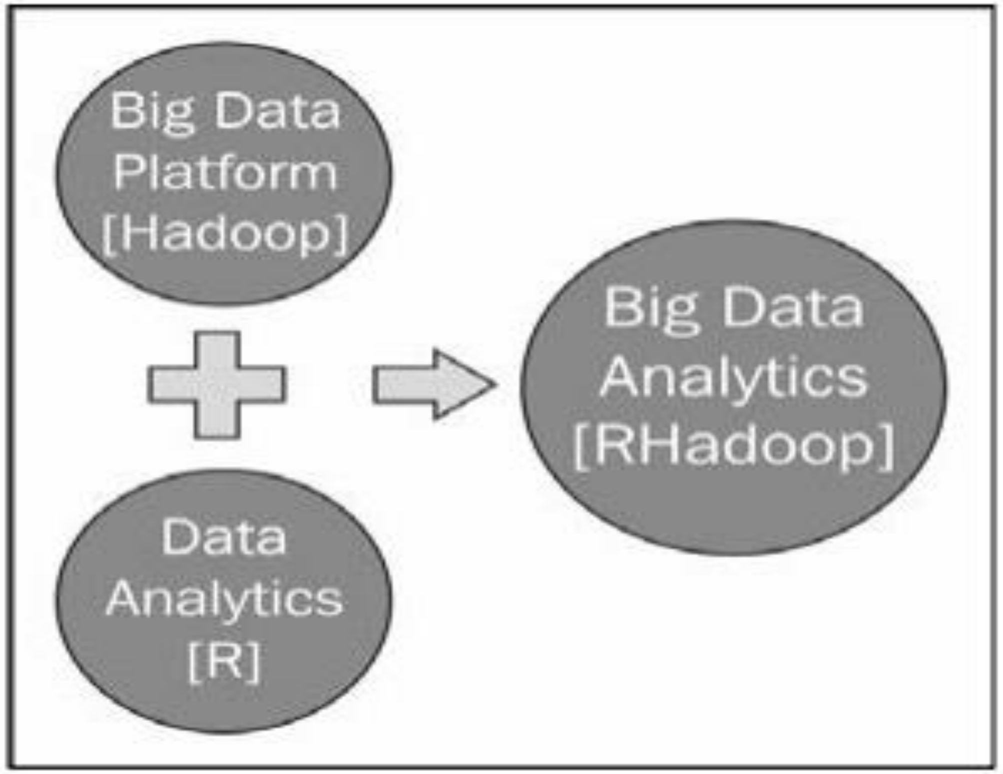
- ➢The advantages of R and Hadoop integration within an organization:
  - R programming language is the preferred choice amongst data analysts and data scientists because of its rich ecosystem catering to the essential ingredients of a big data project.
  - Since data analysts and data scientists frequently use the R tool for data exploration as well as data analytics, Hadoop integration is a big boon for processing large-sized data.
  - Similarly, data engineers who use Hadoop tools can perform analytical operations to get informative insights that are actionable by integrating with R tool.

Integrating R and Hadoop

- ➢There are mainly three ways to link R and Hadoop as follows:
  - RHadoop
  - RHIPE
  - R and Hadoop streaming
- ➢Note: RHIVE is used for launching Hive queries from R interface. It provides functionalities for retrieving metadata like database names, column names, and table names from Apache Hive.

Integrating R and Hadoop - RHadoop

- RHadoop in R allows **performing data analytics with the Hadoop platform via R functions**.
- The integration of such tools and technologies can build a powerful scalable system that has features of both of them.



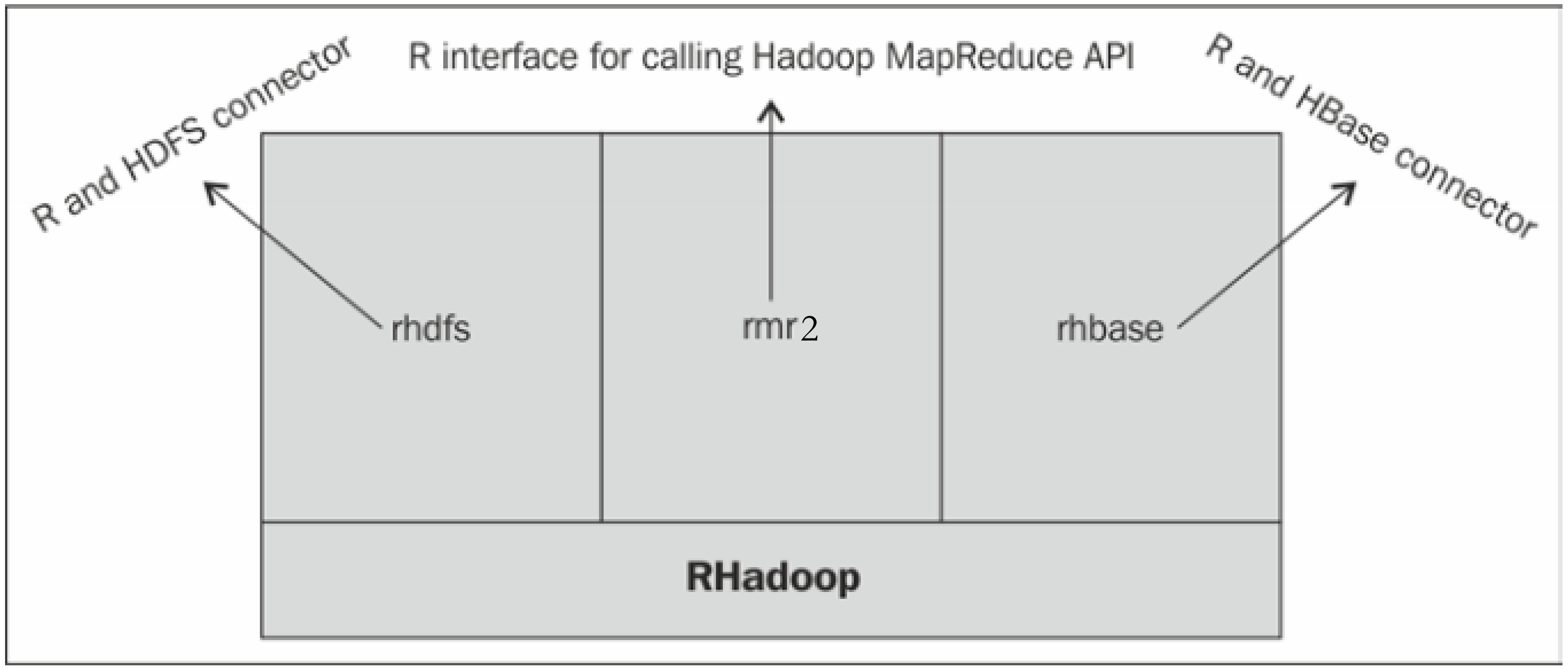
Integrating R and Hadoop - RHadoop

- The RHadoop packages have been designed on Hadoop's two main features: **HDFS** and **MapReduce.**

- The RHadoop project has mainly three different R packages: **rhdfs**, **rmr2**, and **rhbase.**
    - **rhdfs** is an R interface for providing the HDFS usability from R.
    - **rmr2** is an R interface for providing Hadoop MapReduce facility inside R.
    - **rhbase** is an R package for handling data at HBase database through R.

Integrating R and Hadoop - RHadoop

➤The architecture of RHadoop is shown in the following diagram:

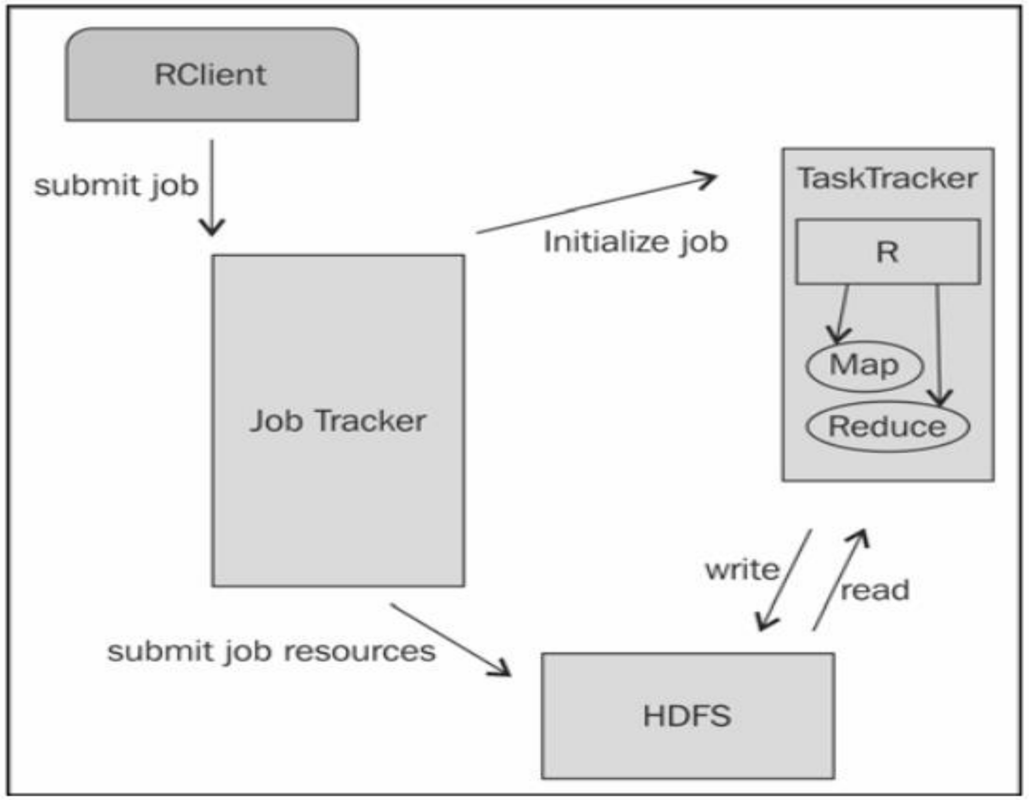


Integrating R and Hadoop - RHIPE

- RHIPE (“R and Hadoop Integrated Programming Environment”) is a package of R that enables the use of API in Hadoop.
- RHIPE in R allows users to run Hadoop MapReduce jobs within R programming language.
- R programmers just have to write R map and R reduce functions and the RHIPE library will transfer them and invoke the corresponding Hadoop Map and Hadoop Reduce tasks.

Integrating R and Hadoop - RHIPE

➢RHIPE library package architecture:



- **RClient** is an R application that calls the **JobTracker** to execute the job.
- The **JobTracker** initializes and monitors the MapReduce jobs over the Hadoop cluster.
- The **TaskTracker** executes the MapReduce jobs as per the orders given by **JobTracker**, retrieve the input data chunks, and run R-specific Mapper and Reducer over it.
- Finally, the output will be written on the **HDFS** directory.

Integrating R and Hadoop - R and Hadoop streaming

- Hadoop streaming is a Hadoop utility for running the Hadoop MapReduce job with executable scripts such as Mapper and Reducer.

- With this, the text input file is printed on stream (stdin), which is provided as an input to Mapper and the output (stdout) of Mapper is provided as an input to Reducer.

- Finally, Reducer writes the output to the HDFS directory.

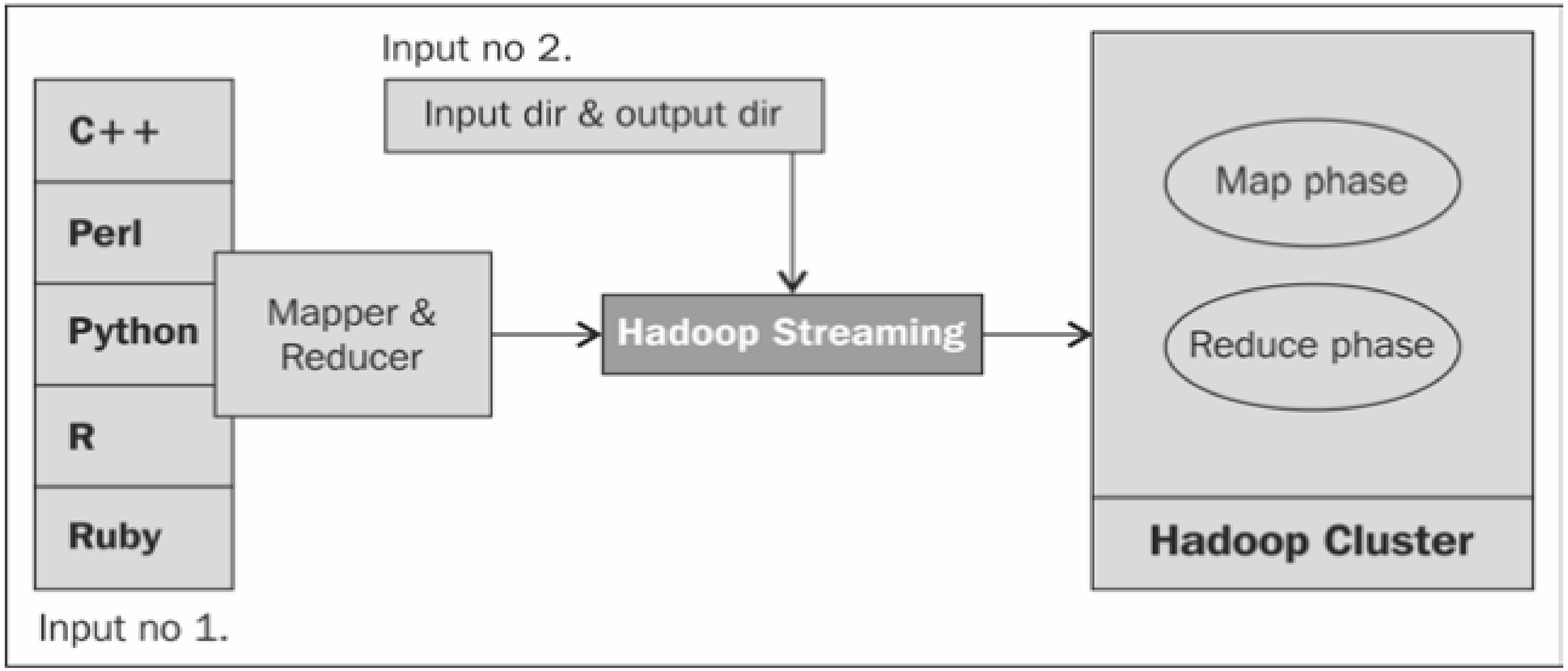
Integrating R and Hadoop - R and Hadoop streaming

- The Hadoop streaming supports the Perl, Python, PHP, R, and C++ programming languages. Also, application written in other programming languages can be run.

- Now, assume we have implemented our Mapper and Reducer as code_mapper.R and code_reducer.R. These can be run with the Hadoop streaming command.

Integrating R and Hadoop - R and Hadoop streaming

➢R and Hadoop streaming architecture:



Apache Spark

Apache Spark

- Apache Spark is a data processing framework that can quickly perform processing tasks on very large data sets, and can also distribute data processing tasks across multiple computers.

- Spark also takes some of the programming burdens of these tasks off the shoulders of developers with an easy-to-use API that abstracts away much of the work of distributed computing and big data processing.

Apache Spark - Features

- **In-memory computing** – The main feature of Spark is its in-memory cluster computing that increases the processing speed of an application.
- **Speed** – Spark helps to run an application in Hadoop cluster faster when running on disk and even faster when run in memory. This is possible by reducing number of read/write operations to disk. It stores the intermediate processing data in memory.
- **Supports multiple languages** – Spark provides built-in APIs in Java, R, Python and Scala.
- **Advanced Analytics** – Spark not only supports Map and reduce, but also supports SQL queries, Streaming data, Machine learning (ML), and Graph algorithms.

Apache Spark - RDD

- **Resilient Distributed Datasets (RDD)** is a fundamental data structure of Spark. It is a distributed collection of objects.
- 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
- Each dataset is divided into logical partitions, which may be computed on different nodes of the cluster.

Apache Spark - RDD

➢ RDD is a read-only, partitioned collection of records. RDD is a fault-tolerant collection of elements that can be operated on in parallel.

➢ There are two ways to create RDDs:

- **Parallelizing** an existing collection in your driver program.
- **Referencing a dataset** in an external storage system, such as a shared file system, HDFS, HBase, or other data sources.

Apache Spark - RDD

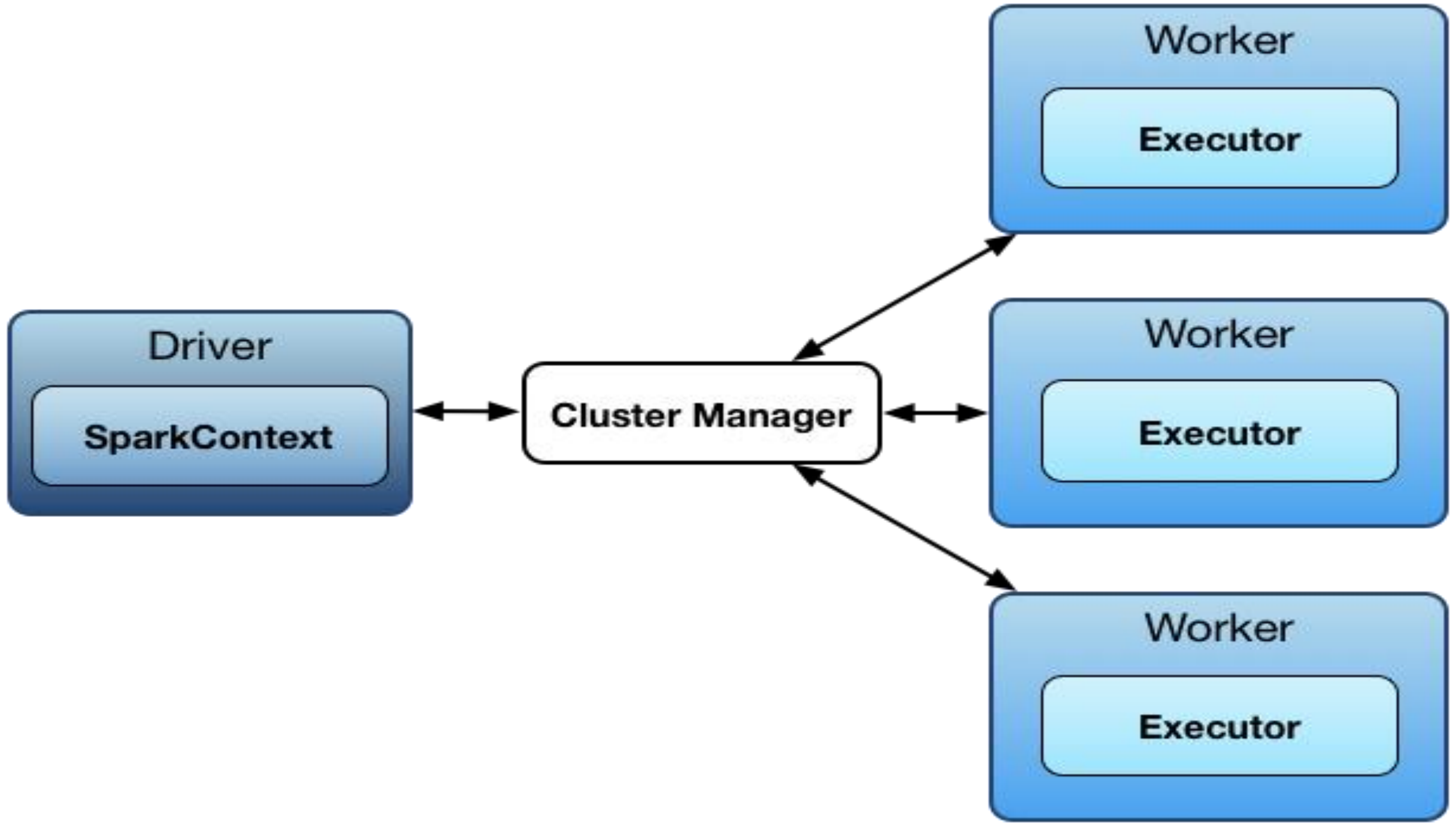
- Spark makes use of the concept of RDD to achieve faster and efficient MapReduce operations.

- With RDDs, you can perform two types of operations:
  - **Transformations:** They are the operations that are applied to create a new RDD.
  - **Actions:** They are applied on an RDD to instruct Apache Spark to apply computation and pass the result back to the driver.

Apache Spark - Architecture

- At a fundamental level, an Apache Spark follows a master/slave architecture with two main components: a *driver* and *workers*.
  - **Driver (Master Node):** It is the entry point that runs the main () function of the application and is responsible for the translation of spark user code into actual spark jobs executed on the cluster.
  - **Workers (Slave Nodes):** They contain the *executers* that run tasks assigned to them. These tasks are executed on the partitioned RDDs in the worker node. Executors perform all the data processing.
- There is also the **Cluster Manager** which is an external service responsible for acquiring resources on the spark cluster and allocating them to a spark job.

Apache Spark - Architecture



Spark context is a gateway to all the Spark functionalities. It is similar to your database connection.

Apache Spark - Workflow

- **STEP 1:** The client submits spark user application code. When an application code is submitted, the driver implicitly converts user code into a logically *directed acyclic graph* called ***DAG.*** At this stage, it also performs optimizations.

- **STEP 2:** After that, it converts the logical graph called DAG into physical execution plan with many stages. After converting into a physical execution plan, it creates physical execution units called tasks under each stage. Then the tasks are bundled and sent to the cluster.

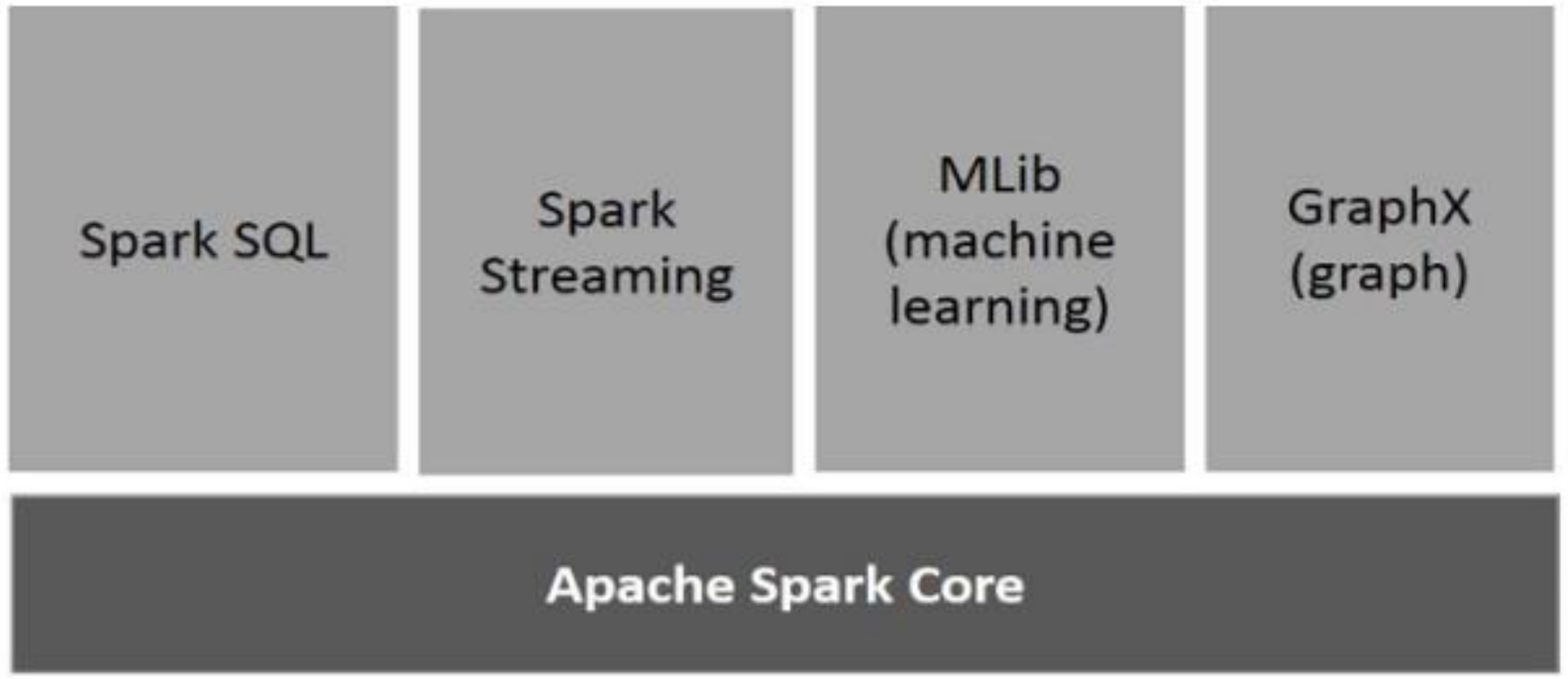
Apache Spark - Workflow

- **STEP 3:** Now the driver talks to the cluster manager and negotiates the resources. Cluster manager launches executors in worker nodes on behalf of the driver. At this point, the driver will send the tasks to the executors based on data placement. The driver will have a complete view of executors that are executing the task.

- **STEP 4:** During the course of execution of tasks, driver will monitor the set of executors that runs. Driver node also schedules future tasks based on data placement.

Apache Spark - Components

➢The following illustration depicts the different components of Spark:



Apache Spark - Components

- Apache Spark Core:
  - Spark Core is the underlying general execution engine for spark platform that all other functionality is built upon. It provides In-Memory computing and referencing datasets in external storage systems.
- Spark SQL:
  - Spark SQL enables users to run SQL queries. Alongside standard SQL support, Spark SQL provides a standard interface for reading from and writing to other datastores including JSON, HDFS, Apache Hive and others.

Apache Spark - Components

- ➢Spark Streaming:
  - Spark Streaming extended the Apache Spark concept of batch processing into streaming by breaking the stream down into a continuous series of microbatches, which could then be manipulated using the Apache Spark API.
- ➢MLlib (Machine Learning Library)
  - Machine learning library delivers both efficiencies as well as the high-quality algorithms. It is capable of in-memory data processing, that improves the performance of iterative algorithm drastically.
- ➢GraphX
  - Spark GraphX is the graph computation engine built on top of Apache Spark that enables to process graph data at scale.

Thank You