Lecture 2

Big Data Processing Frameworks (Apache Hadoop)

Dr. Lydia Wahid

Agenda

- Basic Concepts
- Introduction to MapReduce
- Apache Hadoop
- Hadoop ecosystem
- Hadoop YARN

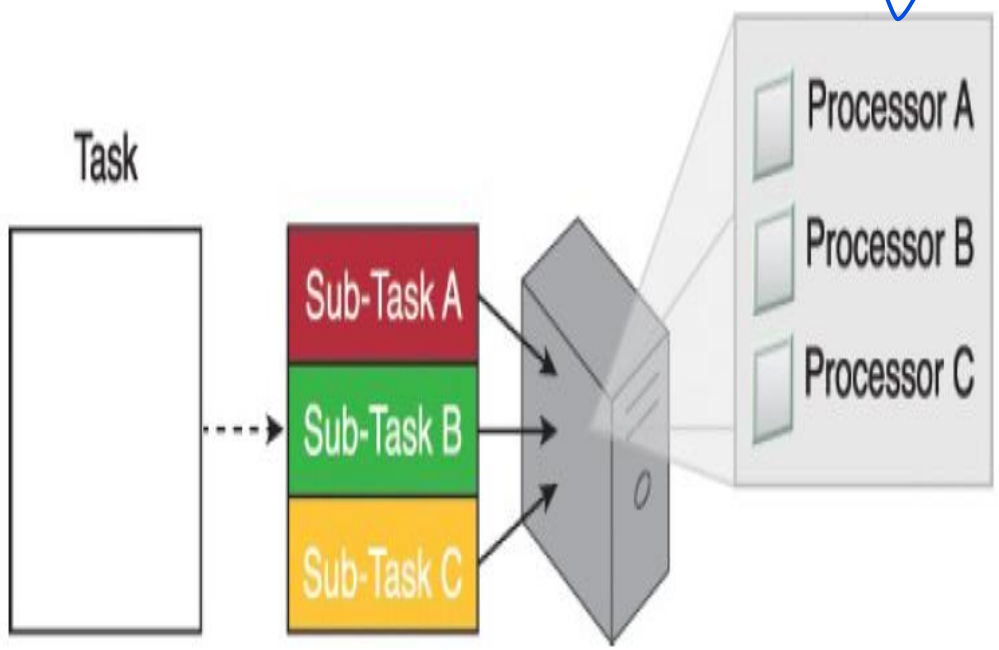
Basic Concepts

Basic Concepts

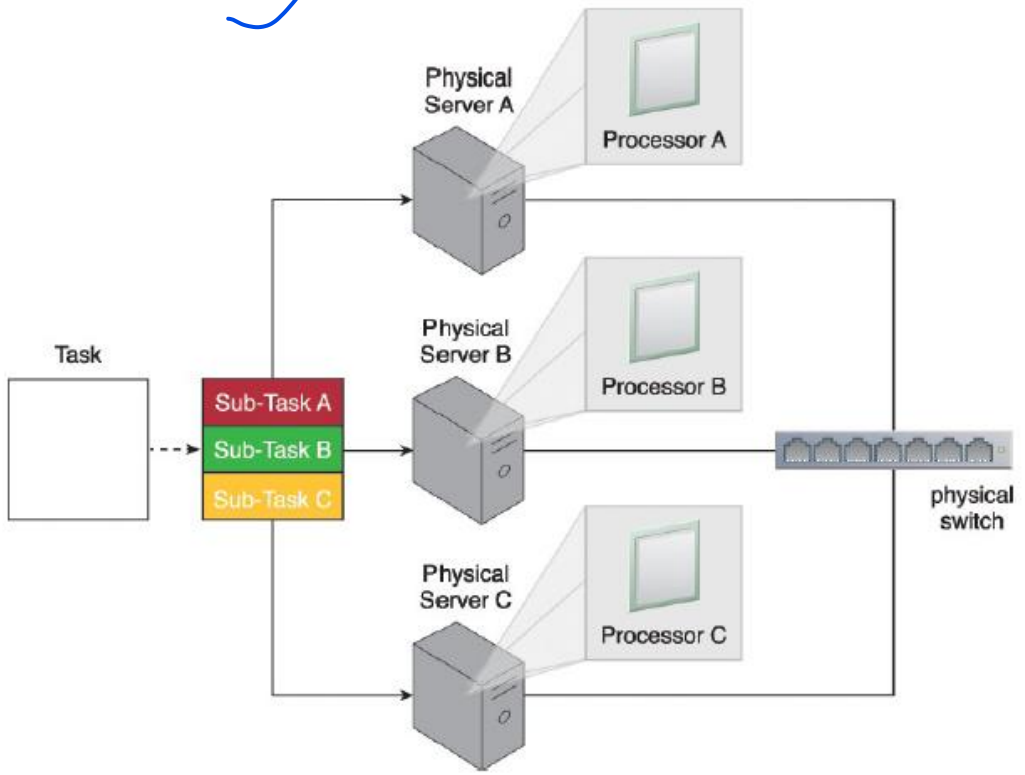
- **Parallel Data Processing vs Distributed Data Processing:**
  - Parallel data processing involves the **simultaneous execution** of multiple sub-tasks that collectively comprise a larger task.
  - The goal is to reduce the execution time by dividing a single larger task into multiple smaller tasks 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



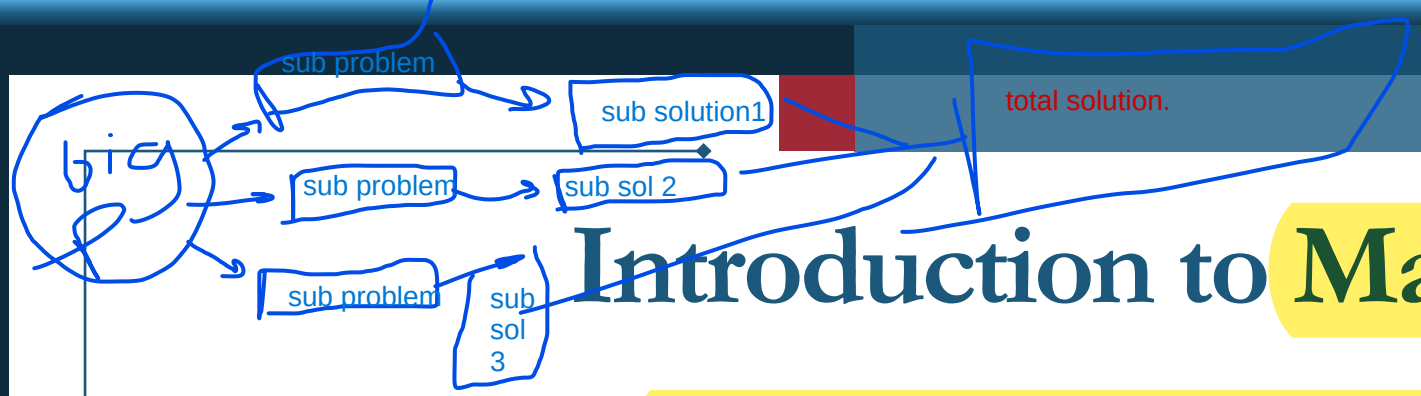
Distributed Data Processing



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.

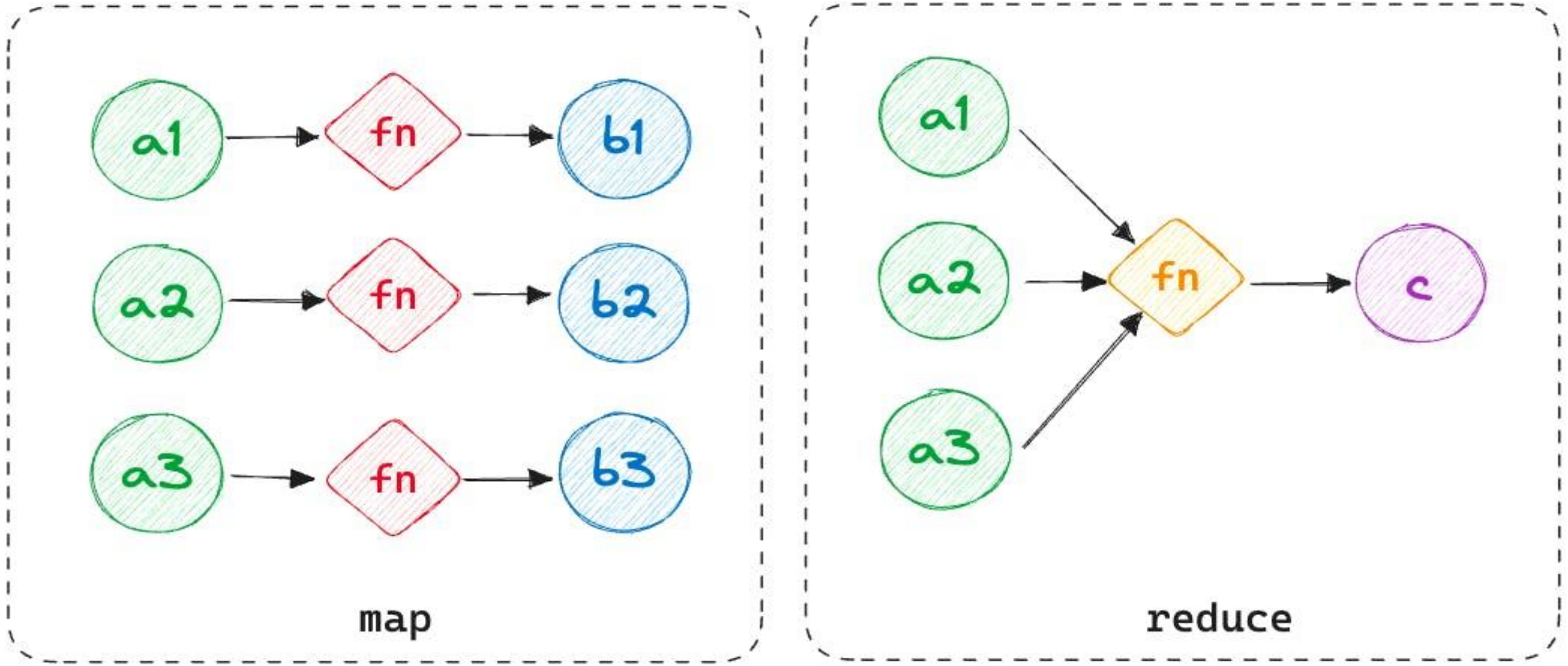
Introduction to MapReduce

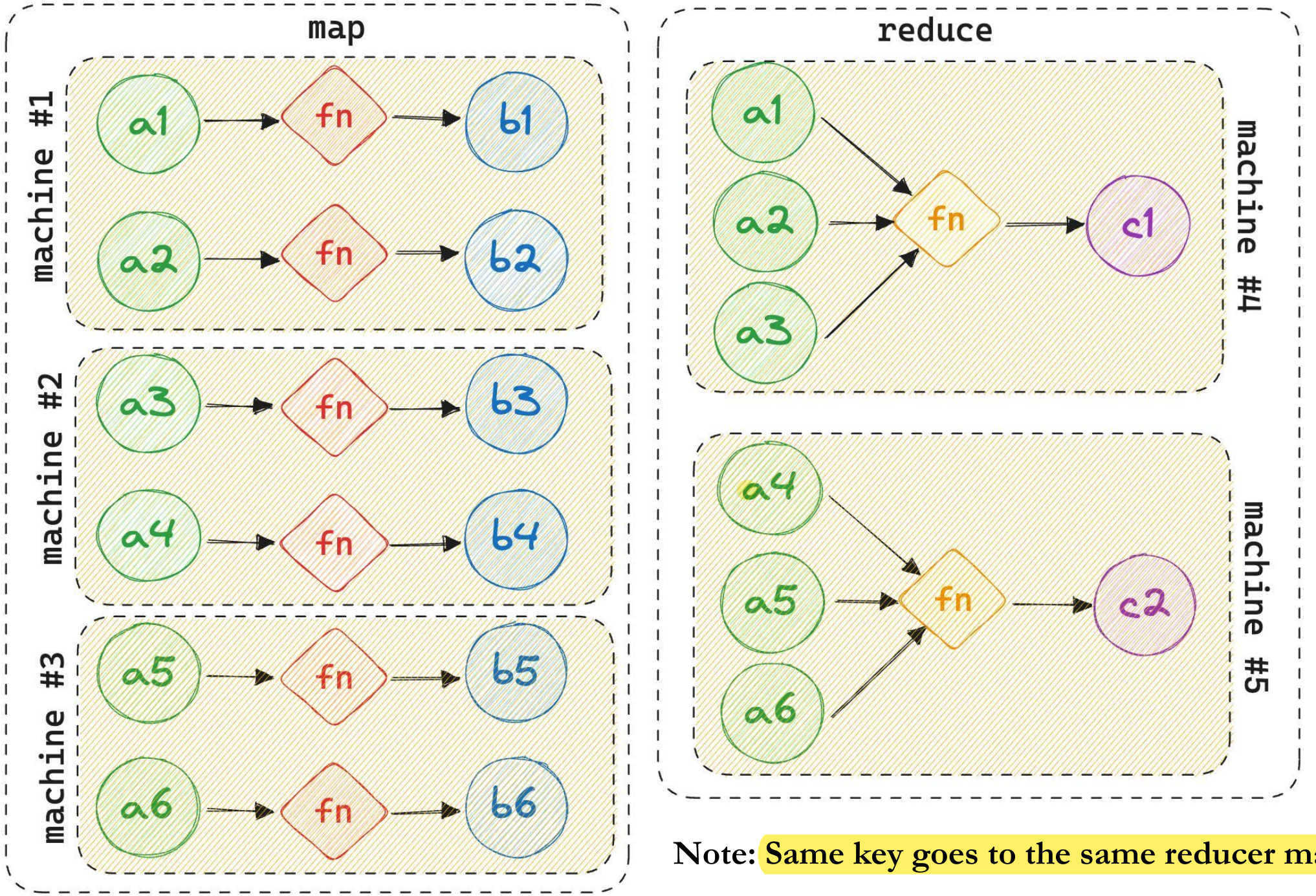


Introduction to MapReduce

- MapReduce is a programming paradigm to perform large-scale computation across computing clusters.
- It is mainly composed of two stages; Map and Reduce:
  - **Map:** takes a sequence of values and iteratively applies a mapper function to each value in the sequence and produces <key, value> pairs as output.
  - **Reduce:** takes <key, value> pairs and combines all the elements having the same key through a reducer function.

Introduction to MapReduce





Note: Same key goes to the same reducer machine

Introduction to MapReduce

- Example: Word count.

```
1  Sub-DataSet-1
2  ----------------
3  Red Blue Red Blue Green Red Blue Green
4  White Black
5  Red White Black
```

```
1  Sub-DataSet-2
2  ----------------
3  Orange Green
4  Red Blue Red
5  Blue Green Red Blue
6  Green White Black
```

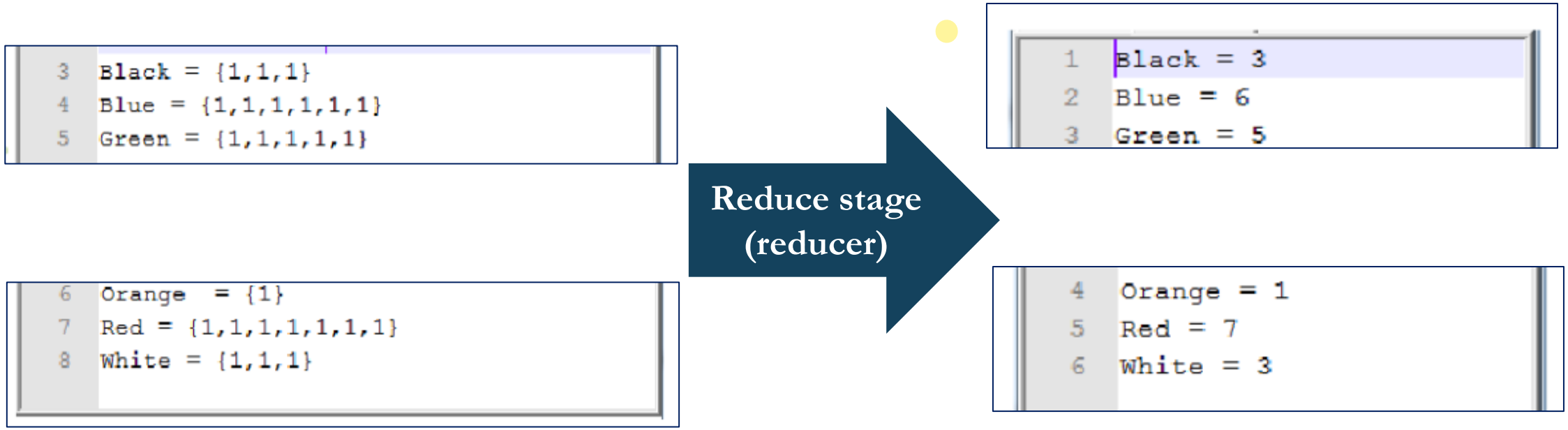
Map stage (mapper)

```
1  Sub-DataSet-1
2  ----------------
3  Red = 1
4  Blue = 1
5  Red = 1
6  Blue = 1
7  Green = 1
8  Red = 1
9  Blue = 1
10 Green = 1
11 White = 1
12 Black = 1
13 Red = 1
14 White = 1
15 Black = 1
```

```
1  Sub-DataSet-2
2  ----------------
3  Orange = 1
4  Green = 1
5  Red = 1
6  Blue = 1
7  Red = 1
8  Blue = 1
9  Green = 1
10 Red = 1
11 Blue = 1
12 Green = 1
13 White = 1
14 Black = 1
```

Introduction to MapReduce

- Example: Word count.
  - Assume there are two reducer machines



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. 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.
    2. Processing/Computation (MapReduce)
       - Capable of processing enormous data in parallel on large clusters of computation nodes. It performs two main tasks: *map* and *reduce*

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

- 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

- **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.

Number of Maps

- The number of maps is usually driven by the number of blocks of the input files.
- 10TB of input data and a blocksize of 128MB, will end up with 81,920 maps, unless **Configuration.set(MRJobConfig.NUM_MAPS, int)** is used to set it even higher.
- The right level of parallelism for maps seems to be around **10-100 maps per-node**, although it has been set up to 300 maps for very cpu-light map tasks.
- Block size can also be changed to adjust the number of blocks.

Number of Reducers

- In Hadoop, the default number of reducers is **one**.
- In this phase the reduce(WritableComparable, Iterable<Writable>, Context) method is called for each <key, (list of values)> pair in the grouped inputs.
- The number of reducers for the job is set by the user via **Job.setNumReduceTasks(int)**
- Increasing the number of reducers increases the framework overhead, but increases load balancing and lowers the cost of failures.
- It is legal to set the number of reduce-tasks to zero if no reduction is desired.

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.

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 ecosystem

Apache Hadoop – Hadoop ecosystem

- **Apache Hive**
  - It is a software developed by Facebook that facilitates reading, writing, and managing large datasets residing in distributed storage using SQL.
  - It allows users to fire queries in SQL-like languages, such as **HiveQL**.

- **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 ecosystem

- **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 stumble upon (This is a personalized recommender system, realtime data storage, and data analytics platform).

Apache Hadoop – Hadoop ecosystem

- **Apache Impala:**
  - With Impala, you can query data, whether stored in HDFS or Apache HBase – including SELECT, JOIN, and aggregate functions – in real time.
  - Impala directly access the data through a specialized distributed query engine. The result is order-of-magnitude faster performance than Hive
- **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 ecosystem

- **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 ecosystem

- **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 ecosystem

- **Apache Zookeeper:**
  - It is a service that enables highly reliable distributed coordination.
  - It is a centralized service for maintaining configuration information, naming, and providing distributed synchronization.

- …..and others

Hadoop YARN

Hadoop YARN

- 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. (master for resources + processing)

- YARN stands for "***Yet Another Resource Negotiator***". It was introduced in Hadoop version 2.0.

- **The fundamental idea of YARN is to split up the functionalities of resource management and job scheduling/monitoring.**

Hadoop YARN

- Thus, YARN is now responsible for Resource Management and Job scheduling.
- Through its various components, YARN can dynamically allocate various resources and schedule the application processing.
- The idea is to have a global ResourceManager (*RM*) and per-application ApplicationMaster (*AM*). An application is either a single job or a DAG of jobs.

Hadoop YARN - Architecture

- The main components of YARN architecture include:

1. **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.
   - The ResourceManager has two main components: *Scheduler* and *ApplicationsManager*.

Hadoop YARN - Architecture

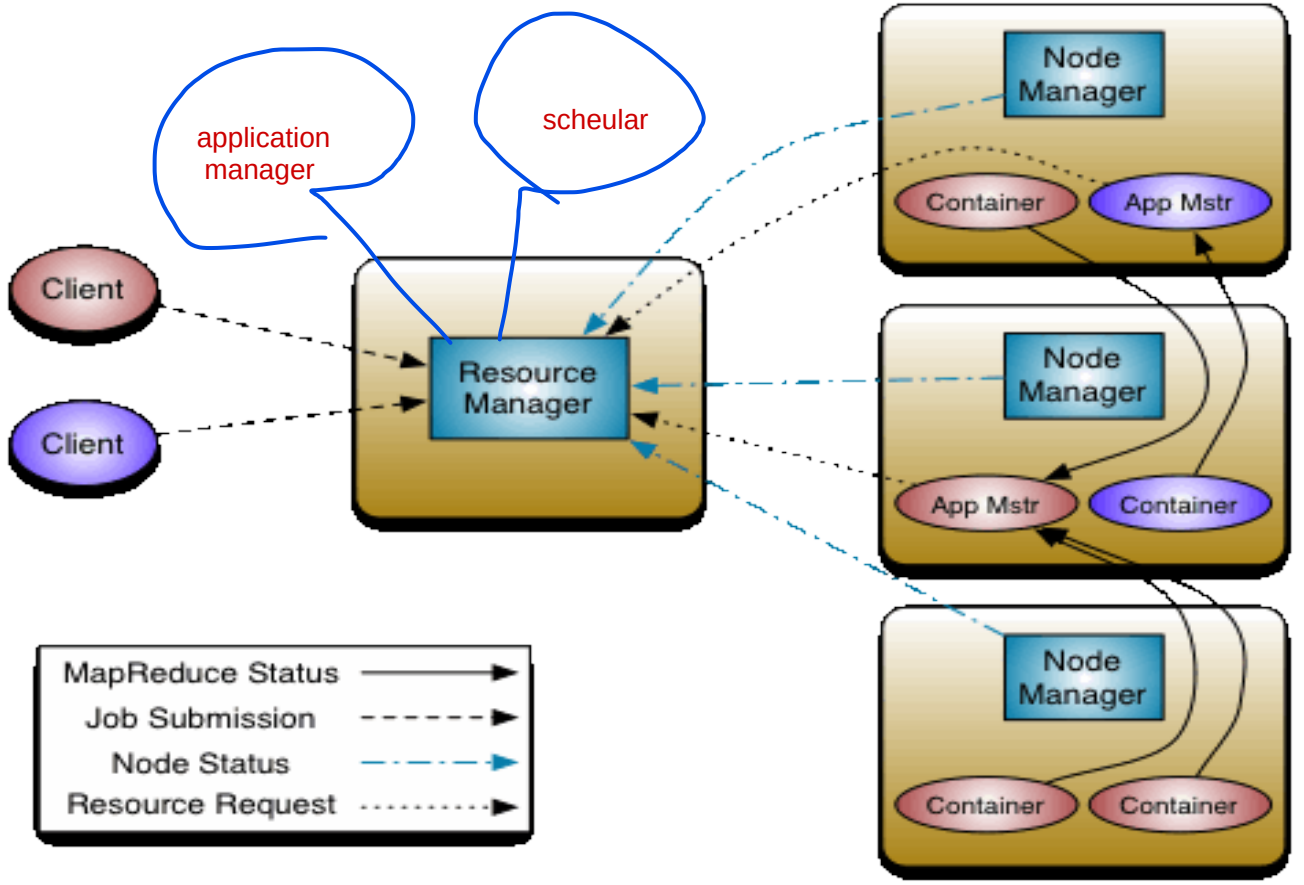
- The main components of YARN architecture include:

1. **Resource Manager:**
   - **ApplicationsManager** is responsible for accepting job-submissions, negotiating the first container for executing the application specific ApplicationMaster and provides the service for restarting the ApplicationMaster container on failure.
   - **Scheduler** is responsible for allocating resources to the various running application. It performs its scheduling function based on the resource requirements of the applications

Hadoop YARN - Architecture

2. **Container:** In the Container, there are the physical resources like a disk, CPU cores, RAM.
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, **executing, 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 - Architecture



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
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

Thank You