NGEE ANN POLYTECHNIC

School of InfoComm Technology

Distributed Data Pipelines

Diploma in Data Science (DS)
October 2023 Semester

INDIVIDUAL ASSIGNMENT 1

(30% of Distributed Data Pipelines Module)

Deadline for Submission:

15th Dec 2023 (Friday), 2359 Hours

Student Name :	Oh Ern Qi
Student Number :	S10243067K

Penalty for late submission:

10% of the marks will be deducted every day after the deadline.
NO submission will be accepted after 22nd Dec 2023, 23:59.

DDP Section A : HADOOP vs SPARK

Table of Contents

Comparing general qualities

Similarities :

- Distributed computing
- Open Source
- Used for Big Data Processing
- Extensive Ecosystems
- Fault Tolerant

Differences:

Processing Model:

Hadoop: ● Uses Map Reduce ● Batch Processing only	Spark: ● In-memory processing model ● Unified approach for batch, interactive and streaming data processing

Performance:

Hadoop: ● Uses disk based storage hence it is slower	Spark: ● Uses in memory processing,leading to faster performance ● 100 times faster than Hadoop Map Reduce

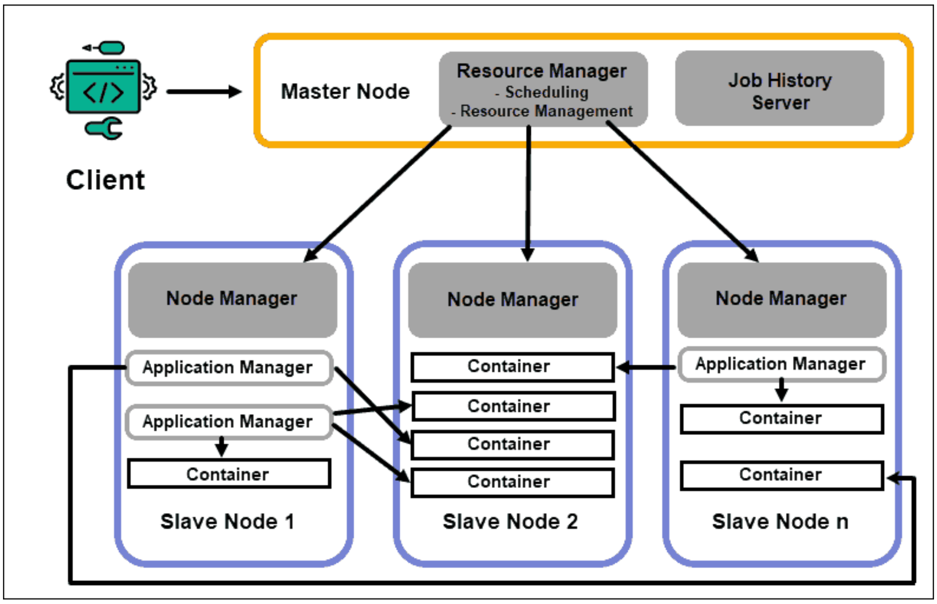
Ease of use:

Hadoop: ● Uses Java only ● Harder to use, greater learning curve	Spark: ● Provides support for multiple programming languages, including Java, Scala, Python, and R.

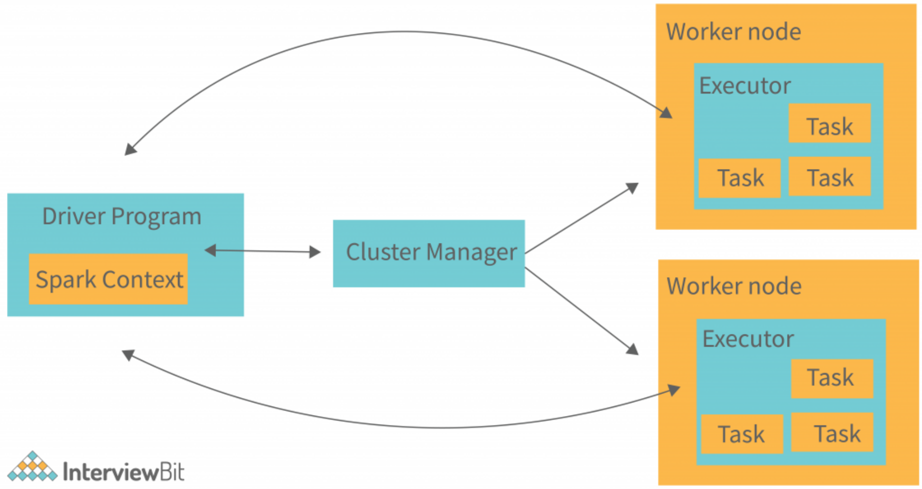
Spark and Hadoop are not mutually independent and can work hand in hand together.

Comparing Architectures

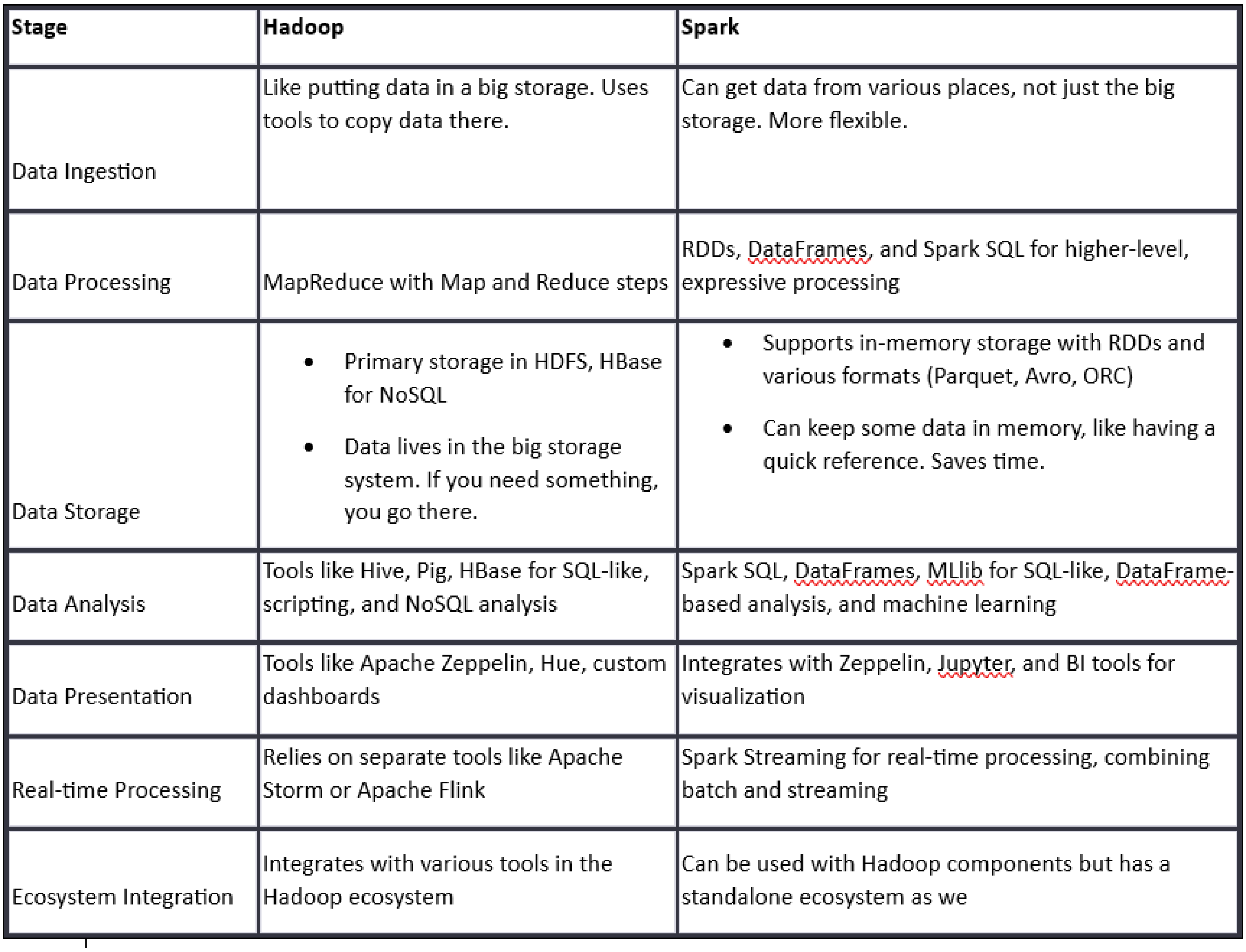
Hadoop:



Spark:



Comparing the Entire Pipeline Process:



Stage	Hadoop	Spark
Data Ingestion	Like putting data in a big storage. Uses tools to copy data there.	Can get data from various places, not just the big storage. More flexible.
Data Processing	MapReduce with Map and Reduce steps	RDDs, DataFrames, and Spark SQL for higher-level, expressive processing
Data Storage	• Primary storage in HDFS, HBase for NoSQL • Data lives in the big storage system. If you need something, you go there.	• Supports in-memory storage with RDDs and various formats (Parquet, Avro, ORC) • Can keep some data in memory, like having a quick reference. Saves time.
Data Analysis	Tools like Hive, Pig, HBase for SQL-like, scripting, and NoSQL analysis	Spark SQL, DataFrames, MLlib for SQL-like, DataFrame-based analysis, and machine learning
Data Presentation	Tools like Apache Zeppelin, Hue, custom dashboards	Integrates with Zeppelin, Jupyter, and BI tools for visualization
Real-time Processing	Relies on separate tools like Apache Storm or Apache Flink	Spark Streaming for real-time processing, combining batch and streaming
Ecosystem Integration	Integrates with various tools in the Hadoop ecosystem	Can be used with Hadoop components but has a standalone ecosystem as we

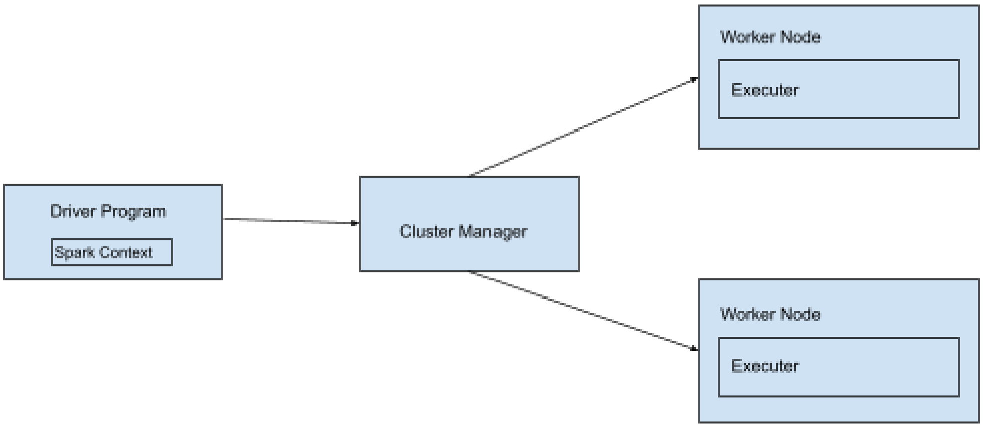
Comparing Resource Management

Resource Manager:

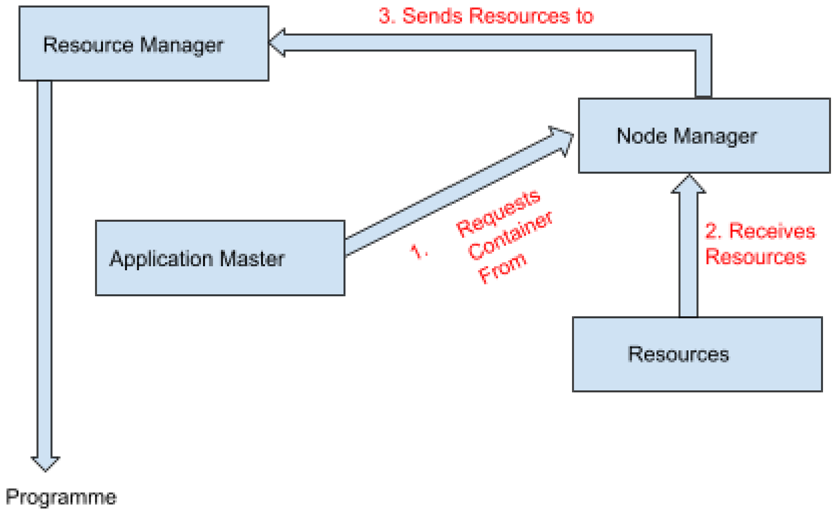
Hadoop:	Spark:
- Manages and schedule resources across the clusters allowing multiple applications to share resources efficiently - Uses Hadoop YARN (Yet Another Resource Negotiator)	- Responsible for acquiring resources such as CPU and memory across the cluster and allocating them to Spark applications. - Uses either Apache Mesos,Hadoop YARN or Spark standalone cluster manager

Comparing Resource Manager Architecture

Spark Cluster Manager:



Hadoop YARN:



Similarities:

- Cluster Resource Allocation:.
    - Both systems aim to efficiently use the computing resources available in a cluster.

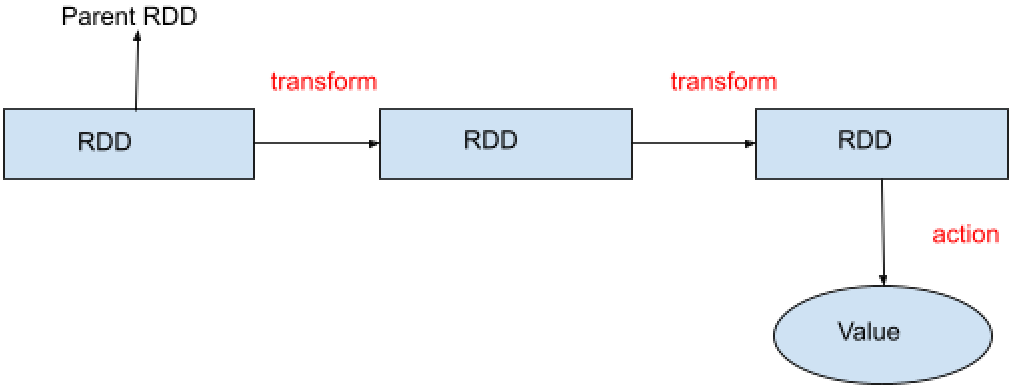
Differences:

- Dynamic Resource Allocation:
    - Hadoop:
        - YARN typically uses a static allocation model with fixed container sizes.
    - Spark:
        - Supports dynamic allocation, allowing it to request additional resources when needed and release resources when they are no longer required.
        - This leads to better utilisation of resources and improved efficiency.

Comparing Data Transformation:

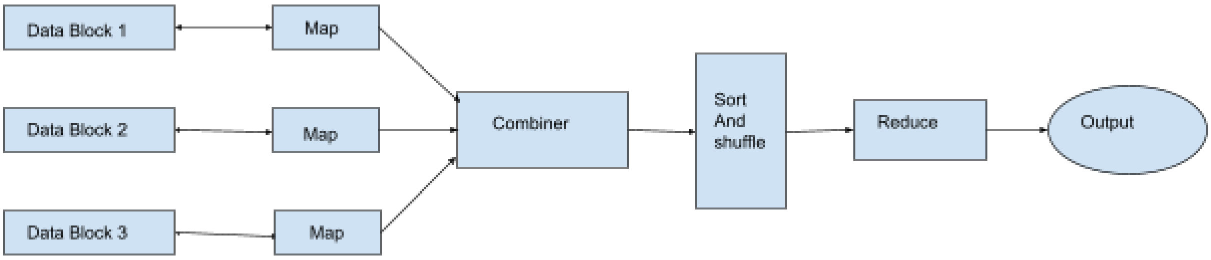
Hadoop : ● Using MapReduce ● MapReduce process allows for parallel and distributed processing of large datasets across a Hadoop cluster.	Spark : ● Using RDDs ● Transformations on RDDs are performed using a set of functional programming operations, such as map, filter, reduceByKey, groupBy, etc.

Spark RDDs:



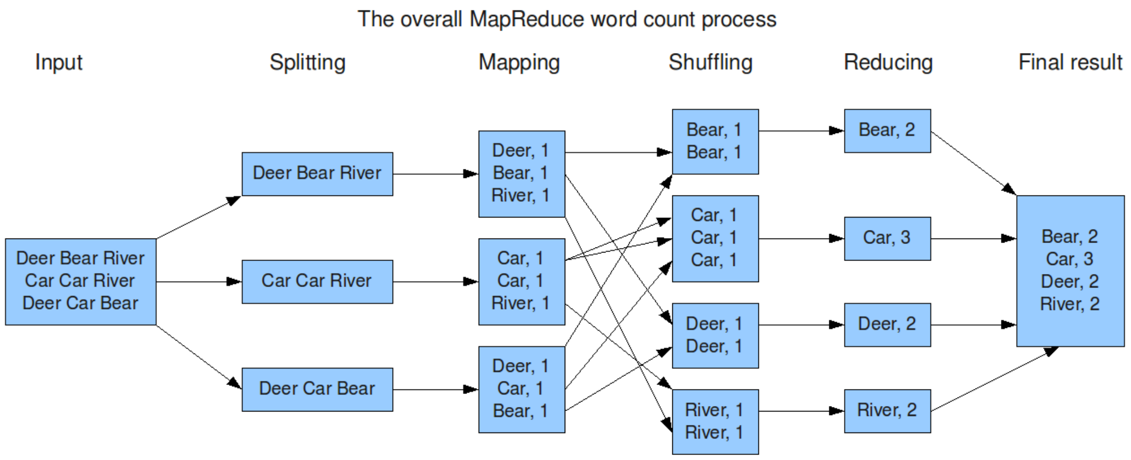
- Transformations are represented as a directed acyclic graph (DAG) of stages.
- The transformation operation produces a new RDD that embodies the changes applied by the transformation.
- A value is produced when an action is applied on the RDD

Hadoop MapReduce:



- MapReduce is a programming model for processing and generating large datasets. It involves two phases: Map, which processes and transforms data into key-value pairs, and Reduce, which aggregates and analyses the results.

Map reduce example:



Similarities:

- Transformations on Distributed Data:
  - Both frameworks provide a set of transformations to process distributed data, dividing the workload across the cluster.

Differences:

- In-Memory vs Disk-Based Data Transformation:
  - In-Memory (Spark):
    - Faster processing using in-memory operations.
  - Disk-Based (Hadoop):
    - MapReduce processing with disk involvement.

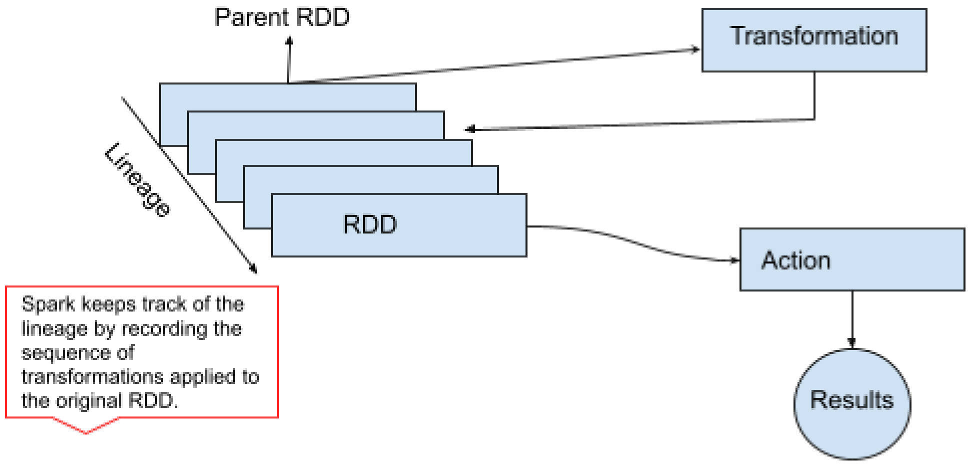
- Output of Transformation:
  - Hadoop MapReduce: Reduce phase produces key-value pairs as final results.
  - Spark RDDs: Transformation produces a new RDD capturing changes; actual computation occurs during actions.

Comparing Fault Tolerance

Hadoop: Using HDFS(Hadoop Distributed File System): 1. **Data Replication :** Replicating data across multiple nodes in the cluster. HDFS replicates each data block 3 times, storing the copies on different nodes. 2. **Heartbeat and reassignment:** DataNodes periodically send heartbeat signals to the NameNode. If the NameNode doesn't receive a heartbeat, it considers the DataNode as failed and triggers re-replication of data from failed nodes to healthy nodes.	Spark: Using RDD (Resilient Distributed Dataset): 1. **Lineage information:** Lineage records the sequence of transformations applied to the base dataset, allowing for recovery of lost data. 2. **Data partitioning & replication:** Provides fault tolerance by having redundant copies of data. If a partition is lost, it can be recovered from a replica. 3. **Task re-execution:** If a task fails on a particular partition, Spark can re-execute the task on another available node.

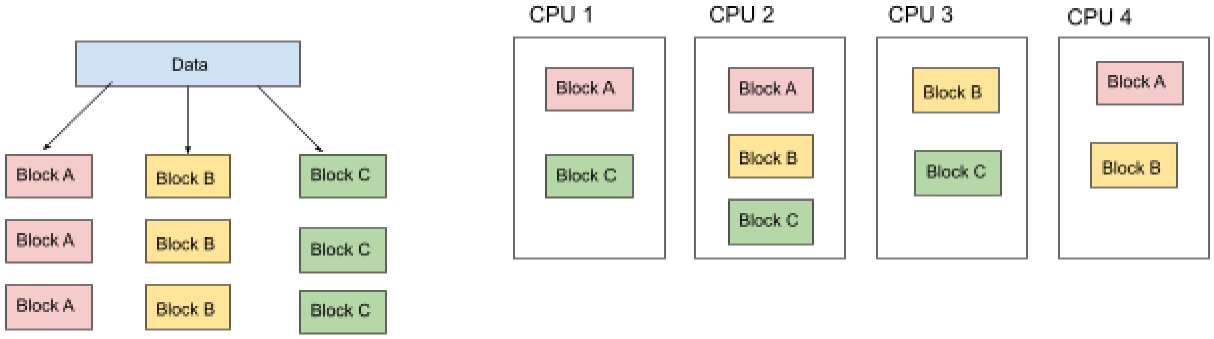
Comparing Fault Tolerant Architecture

Using RDDs in Spark



Spark keeps track of the lineage by recording the sequence of transformations applied to the original RDD.

Using HDFS in Hadoop



Similarities:

- Data Replication:
  - Both HDFS and Spark RDDs employ data replication to mitigate the impact of node failures.
  - Data is stored across multiple nodes, ensuring availability in case of individual node faults.
- Automatic Recovery:
  - Both systems feature mechanisms for automatic recovery from node failures.

Differences:

- Computation Recovery:
  - Spark RDDs offer fault tolerance at the computation level by using lineage information.
  - HDFS primarily focuses on data durability and recovery at the storage level.

Image References:

Hadoop Architecture:
https://www.google.com/url?sa=i&url=https%3A%2F%2Fphoenixnap.com%2Fkb%2Fapache-hadoop-architecture-explained&psig=AOvVaw3f9cvE2zYAD1RjDIVuVX96&ust=1702107907935000&source=images&cd=vfe&opi=89978449&ved=0CBIQjRxqFwoTCKjj65ys_4IDFQAAAAAdAAAAABAD

Spark Architecture:
https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.interviewbit.com%2Fblog%2Fapache-spark-architecture%2F&psig=AOvVaw35LUpe0LojGRsDr0DPkIM2&ust=1702107945673000&source=images&cd=vfe&opi=89978449&ved=0CBIQjRxqFwoTCJD8xt6s_4IDFQAAAAAdAAAAABAP

Mapreduce Example:
https://www.google.com/url?sa=i&url=http%3A%2F%2Fwww.todaysoftmag.com%2Farticle%2F1358%2Fhadoop-mapreduce-deep-diving-and-tuning&psig=AOvVaw1zQ_UBMZVefi1BWSdHqxje&ust=1702139271430000&source=images&cd=vfe&opi=89978449&ved=0CBIQjRxqFwoTCKii4YehgIMDFQAAAAAdAAAAABAD