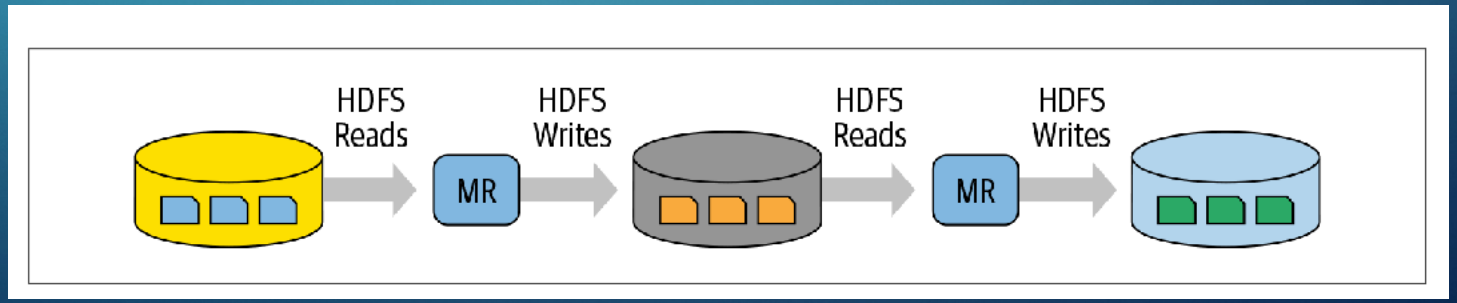
DISTRIBUTED AND CLOUD COMPUTING

LAB9 INTRO TO APACHE SPARK

THE GENESIS OF SPARK

- The Hadoop MapReduce framework on HDFS had a few shortcomings.
  - It was hard to manage and administer, with cumbersome operational complexity
  - The batch-processing MapReduce API was verbose.
  - Intermediate computed result is written to the local disk, not memory.
  - Not able to combine other workloads such as machine learning, streaming, or interactive SQL-like queries.



Workflow of Hadoop MapReduce: intermediate computed result is written to the local disk

APACHE SPARK

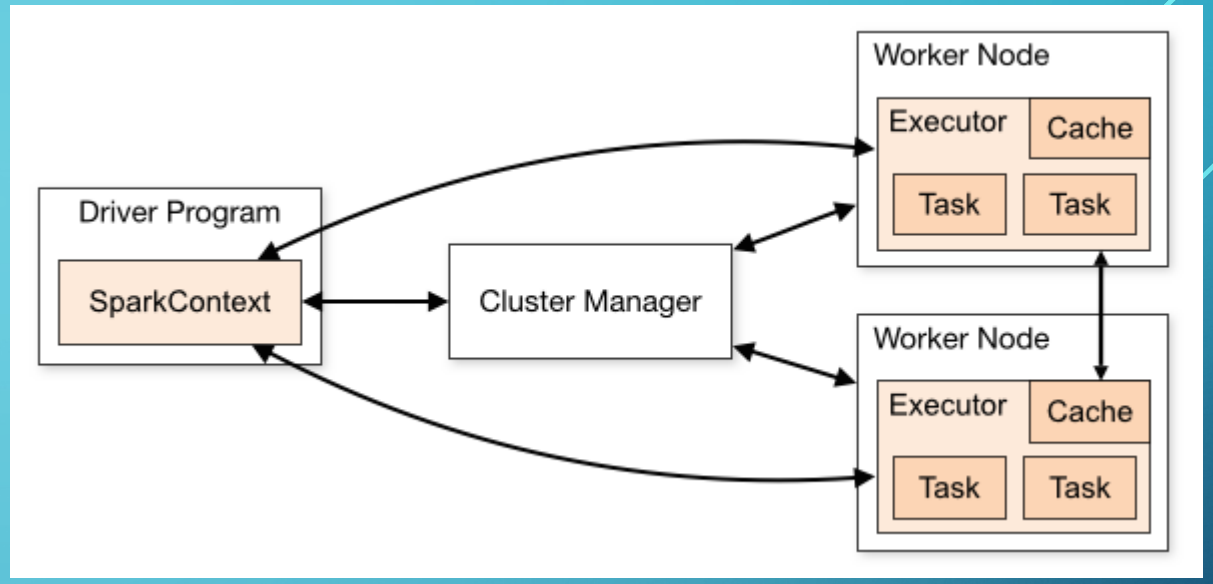
A distributed data processing engine with its components working collaboratively on a cluster of machines.

- Spark provides in-memory storage for intermediate computations, making it much faster than Hadoop MapReduce.
  - ~100x speedup compared to Hadoop MR
- It incorporates various libraries:
  - Machine learning (MLlib)
  - SQL for interactive queries (Spark SQL)
  - Stream processing (Structured Streaming) for interacting with real-time data
  - Graph processing (GraphX)
- Key characteristics:
  - Speed: it builds its query computations as a directed acyclic graph (DAG)
  - Ease of use: data structure called a Resilient Distributed Dataset (RDD); operations as *transformations* and *actions*
  - Modularity: the supported languages include Scala, Java, Python, SQL, R
  - Extensibility: it decouples storage and compute

INTO IN-MEMORY COMPUTING

- Why Spark is faster than Hadoop MR?
  - 1. Data is better cached in memory. Hadoop MR will write almost all result to disk
  - 2. Using of DAG and RDD to store intermediate result and quick recover
  - 3. Spark uses thread pool while Hadoop MR are processes
- Spark, in its core, is still MapReduce

SPARK ARCHITECTURE

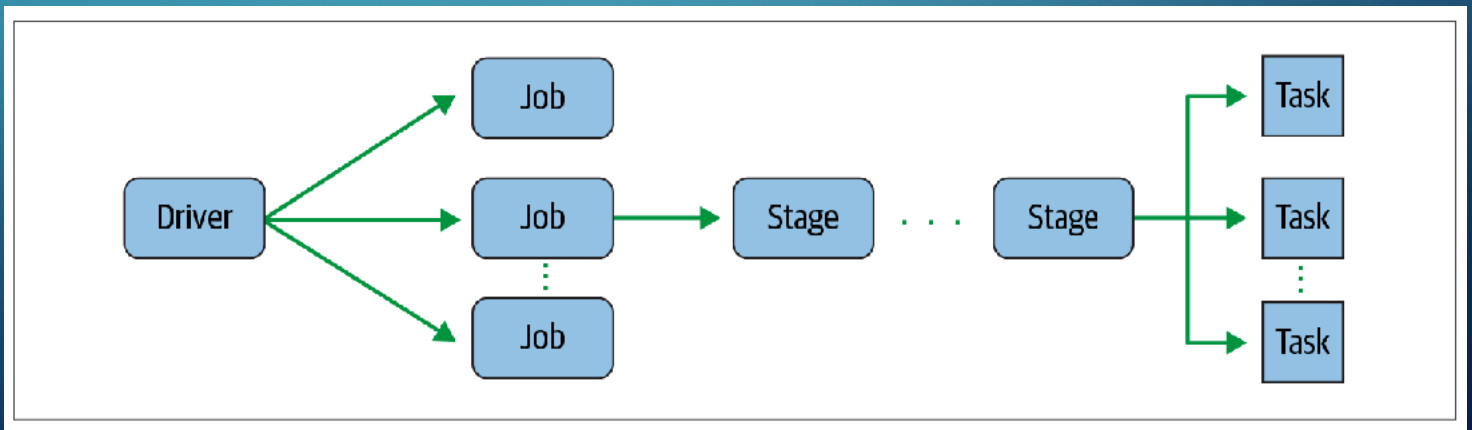


- Still a master-slave architecture
  - Every Spark application consists of a *driver program* that runs the user's main function and executes various *parallel operations* on a cluster.
- Driver: Execute main function of user program and create SparkContext
  - SparkContext: Main entrance for Spark functionalities. Basically a representation of Spark API
- Executor: Run specific task

https://spark.apache.org/docs/latest/cluster-overview.html

WORKFLOW

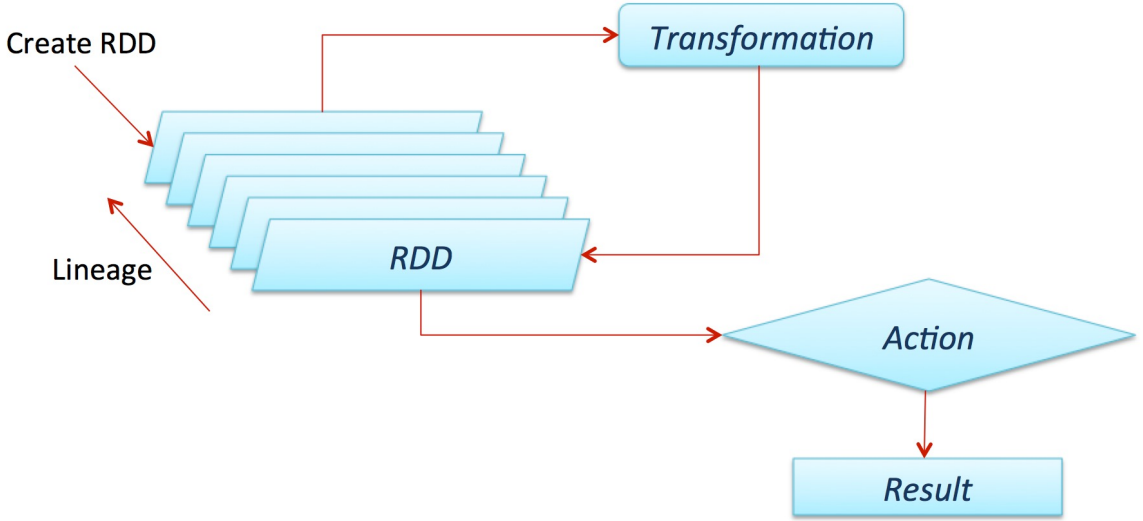
- Spark driver converts Spark application into one or more Spark jobs. It then transforms each job into a DAG (Directed Acyclic Graph).
- Stages are created based on what operations can be performed serially or in parallel
- Each task maps to a single core and works on a single partition of data



RESILIENT DISTRIBUTED DATASET (RDD)

- The fundamental data structure of Apache Spark
  - RDD is a collection of elements partitioned across the nodes of the cluster that can be operated on in parallel
  - RDD is a read-only and partitionable distributed dataset
- RDD can be cached in memory, either fully or partially (by partition)
- RDD can either be stored in memory, or stored in disk in case of insufficient memory
- Can be replaced with higher level APIs such as DataFrame, Dataset

RDD LAZY EVALUATION



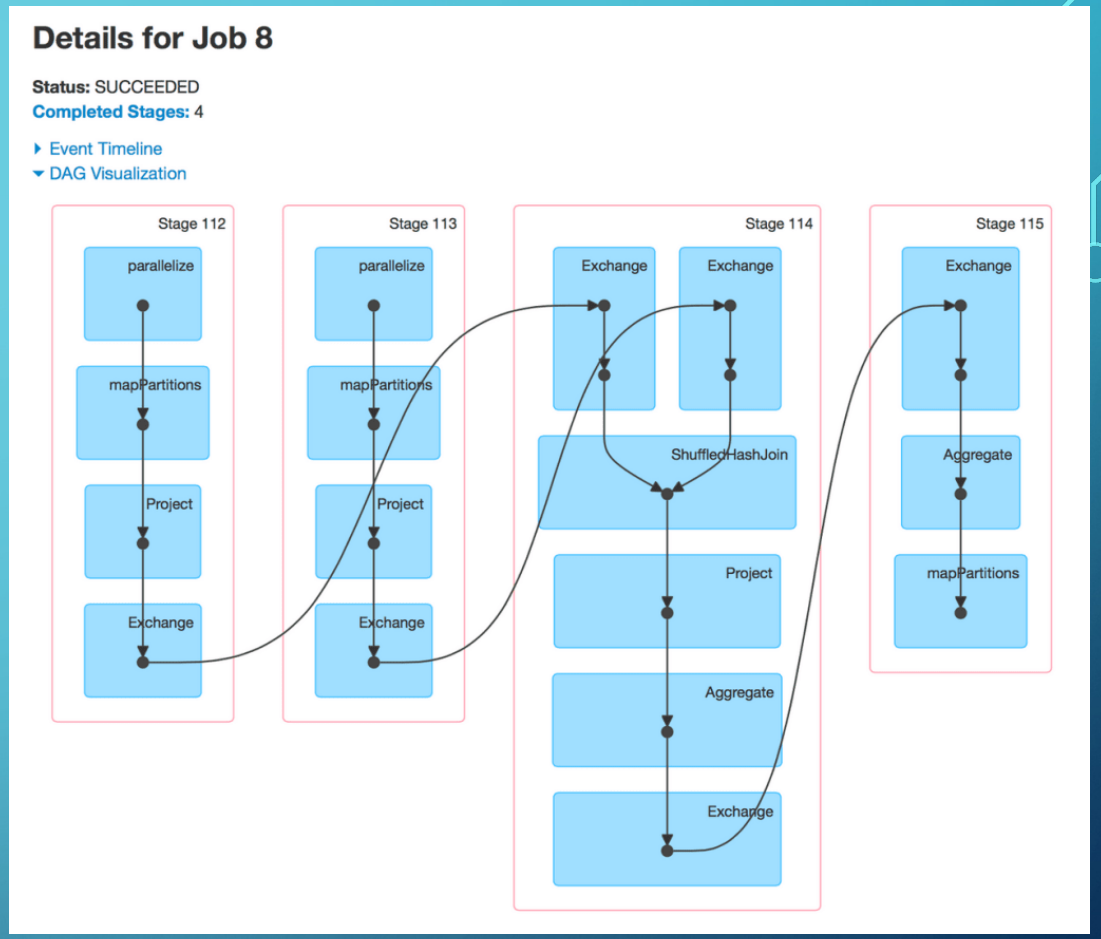
- Spark has two types of operations
  - Transformation (e.g., map, select, filter, etc.): doesn't alter the original data; returns the transformed results
  - Action (e.g., reduce)
- Lazy evaluation: transformations of RDDs are not computed immediately
  - They are recorded or remembered as a lineage (through DAG).
  - They will be computed until an action is invoked or data is "touched" (read from or written to disk).
  - Why? For optimisation purposes.
    - Spark can later rearrange certain transformations, coalesce them, or optimise transformations into stages for more efficient execution.

Transformations	Actions
orderBy()	show()
groupBy()	take()
filter()	count()
select()	collect()
join()	save()

https://data-flair.training/blogs/apache-spark-lazy-evaluation/

SPARK DAG

- Directed acyclic graph
- Vertices are RDDs, edges are operations
- Task, Taskset, Stage, Job, Application
- DAGScheduler: Construct Stage
- TaskScheduler: Submit Taskset to worker



https://medium.com/@goyalsaurabh66/spark-basics-rdds-stages-tasks-and-dag-8da0f52f0454

DEPLOYING SPARK

- Spark is written in Scala (still runs in JVM), and provides high-level APIs in Java, Scala, Python and R
- Spark still uses HDFS as storage
- Spark Core is the core component we are going to use
- Spark works with YARN/Mesos

PRACTICE

- Deploy Spark on a VM cluster
  - https://spark.apache.org/docs/latest/cluster-overview.html
  - https://spark.apache.org/docs/latest/quick-start.html
- You need to:
  - Download Spark (3.3.2) : https://spark.apache.org/downloads.html
  - Deploy and configure Hadoop (should have already been done)
  - Deploy Spark

REFERENCE

- *Jules S. Damji, Brooke Wenig, Tathagata Das, and Denny Lee, "Learning Spark: Lightning-Fast Data Analytic. 2nd Edition", O'Reilly, 2020*
  - *[Available on Blackboard]*