Fourth Industrial Summer School

Big Data Analytics

Apache Spark

Session Objectives

- ✓ Introduction
- ✓ Fundamentals
- ✓ Apache Spark

Spark

- It is the most active open source project in the big data world
- Successor to Hadoop MapReduce
- It is an ***in-memory*** cluster computing framework for processing and analyzing big data

Key Features

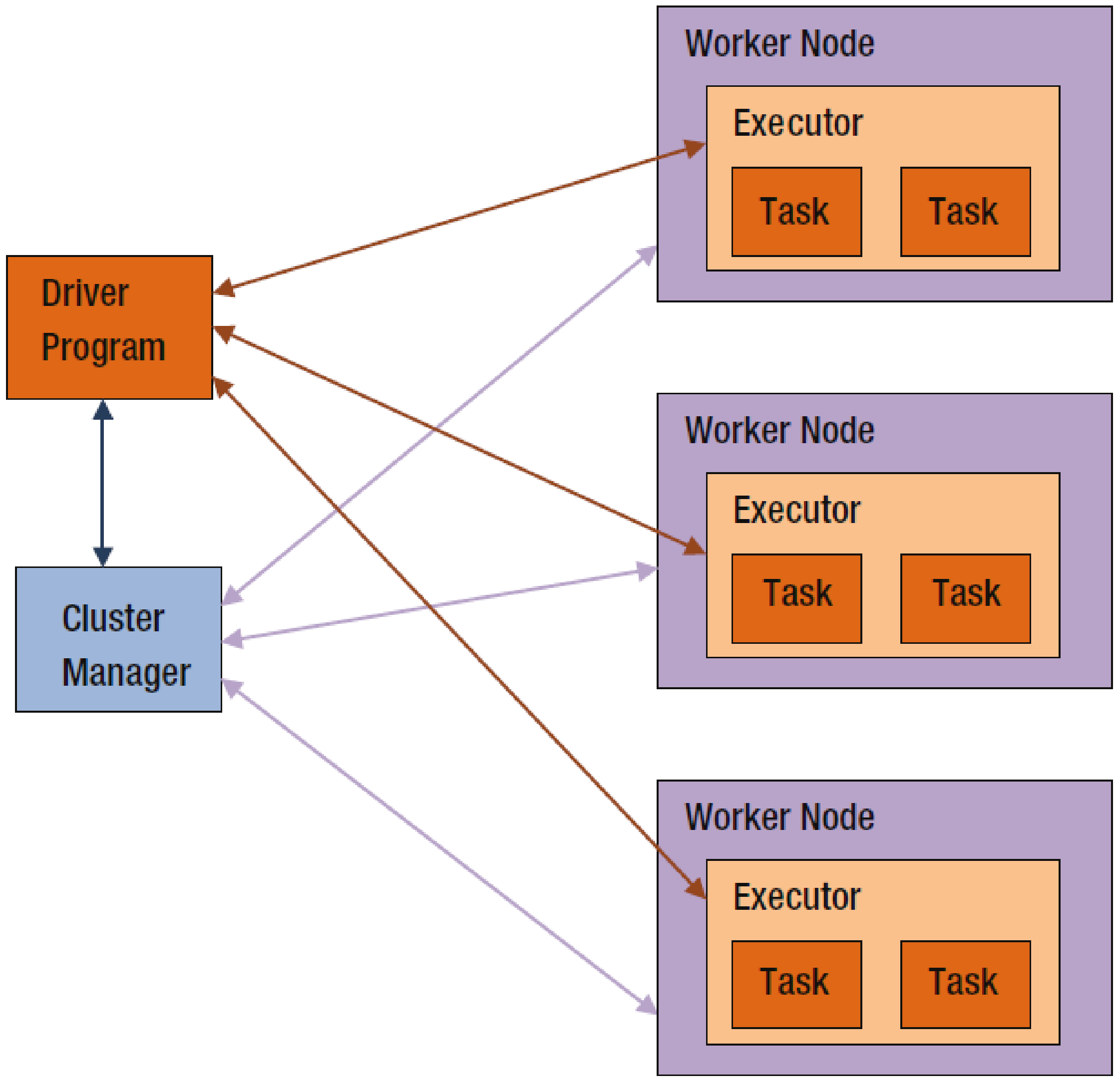
- Easy to Use
  - Simpler than MapReduce
  - Rich set of application programming interface (APIs)
- Fast
  - Orders of magnitude faster than MapReduce
  - For big data I/O latency is the most important contributor of execution time
  - It has advanced job execution engine

Key Features

- General Purpose
  - It can be used batch processing, interactive analysis, stream processing, machine learning, and graph computing
- Scalable and Fault Tolerant
- Ideally suited for Iterative Algorithms and Interactive Analysis

High-level Architecture

- Spark creates a task per data partition.
- More partitions mean more tasks processing data in parallel.



Application Execution

- Shuffle: A shuffle redistributes data among a cluster of nodes
- Job: A set of computations that Spark performs to return results to a driver program
- Stage: Collection of tasks. Spark splits a job into a DAG of stages

- An application is submitted as a job, it gets broken down into DAG stages. The executors run the tasks submitted by Spark in parallel. A stage consists of many tasks (one per partition of RDD)

Data sources

- Spark supports a variety of data sources: HDFS, HBase, Cassandra, and even local file system.

Application Programming Interface (API)

- Spark is written in Scala but provides APIs in many languages including Scala, Java, Python, and R.

The Spark API consists of two important abstractions:

- **SparkContext**: SparkContext is a class defined in the Spark library. It is the main entry point into the Spark library.

  sc = SparkContext()
- **Resilient Distributed Datasets** (**RDD**): RDD represents a collection of partitioned data elements that can be operated on in parallel. It supports lazy operations.
  - An RDD is an Interface, Immutable, Partitioned, Fault Tolerant, Strongly Typed, and In-Memory

RDDs Operations – Creation

Creating RDDs:

- parallelize
- textFile
- sequenceFile
- …

RDDs Operations –Transformations

A transformation method of an RDD creates a new RDD by performing a computation on the source RDD.

- map
- filter
- sample

RDDs Operations –Actions

Actions are RDD methods that return a value to a driver program

- collect
- count
- reduce

RDD materialization

- It follows a lazy materialization approach. In other words, RDD transformations are lazy
- It just creates an execution plan without executing, until necessary

Key-Value RDDs

- A collection of (key, value) pairs

Practice

For APIs, you can refer to:

https://spark.apache.org/docs/latest/api/python/index.html

References

- Guller, Mohammed. Big data analytics with Spark: A practitioner's guide to using Spark for large scale data analysis. Apress, 2015.
- Yoav Freund, Big Data Analytics using Spark, University of California San Diego.
- Hien Luu, Beginning Apache Spark 2: With Resilient Distributed Datasets, Spark SQL, Structured Streaming and Spark Machine Learning library, Apress, 2018.