

CSC 735 – Data Analytics

Chapter 2

A Gentle Introduction to Spark

Images from Guller, Mohammed. *Big data analytics with Spark: A practitioner's guide to using Spark for large scale data analysis*. Apress, 2015.

Goal

- Overview of Spark's core architecture of a cluster
- Overview of a Spark application
- Overview of Spark's structured APIs using DataFrames and SQL

Spark's Basic Architecture

- Using a single machine to process large data is not efficient

Spark's Basic Architecture

- Using a single machine to process large data is not efficient
- A cluster pools the resources of many machines together

Spark's Basic Architecture

- Using a single machine to process large data is not efficient
- A cluster pools the resources of many machines together
- We need a framework to coordinate work across these machines

Spark's Basic Architecture

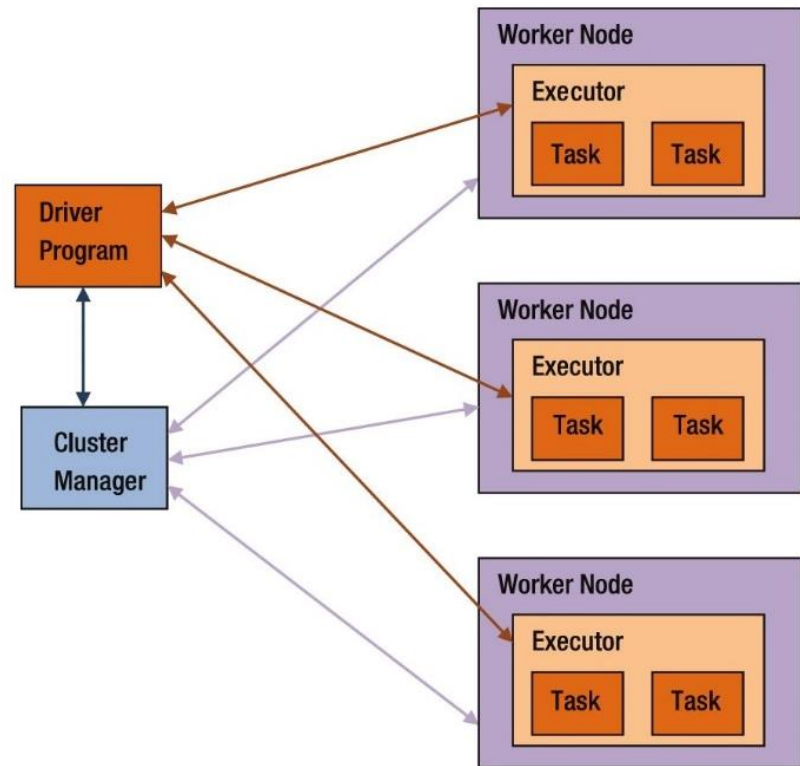
- Using a single machine to process large data is not efficient
- A cluster pools the resources of many machines together
- We need a framework to coordinate work across these machines
- Spark provides such a framework
 - it manages and coordinates the execution of tasks on data across a cluster of machines

Spark's Basic Architecture (cont.)

- A Spark application is submitted to a Spark cluster
- The cluster manager takes care of allocating and coordinating resources on the cluster to run the application
- Examples of cluster managers: Spark's standalone cluster manager, YARN, or Mesos

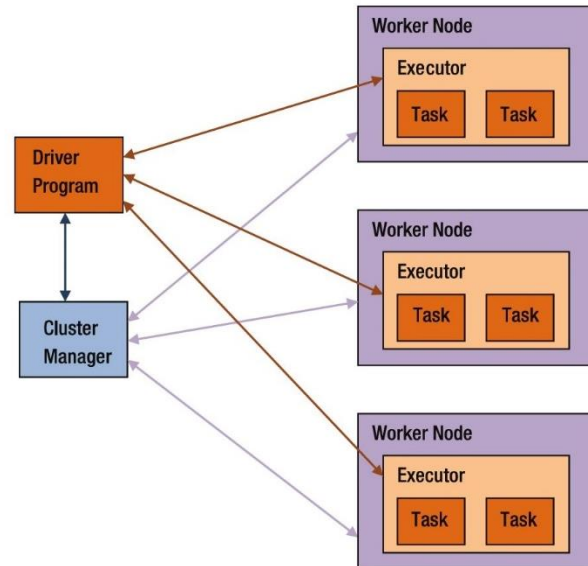
Spark Application

- A Spark application involves five entities:
 1. driver program
 2. cluster manager
 3. workers
 4. executors
 5. tasks



Workers

- Nodes on the cluster
- A worker provides CPU, memory, and storage resources
- Application runs on workers



Cluster Manager

- Spark uses a cluster manager to acquire and manage cluster resources to run an application
- It enables multiple applications to share cluster resources and run on the same worker nodes

Driver Program

- A Spark application consists of a driver program and a set of executors
- Think of a driver program as the pilot of a plane
- It contains the data processing code that Spark executes on the worker nodes

Driver Program (cont.)

- It interacts with the resource manager and the executors
- It is responsible for managing and distributing work across the executors

Driver Program (cont.)

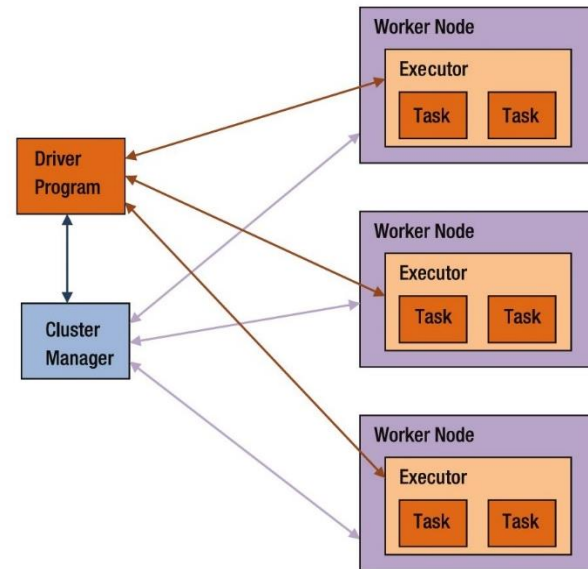
- It interacts with the resource manager and the executors
- It is responsible for managing and distributing work across the executors
- If the code requires the display of computed results to the user, the driver will collect the computed results from the executors and merge them together

Driver Program (cont.)

- It interacts with the resource manager and the executors
- It is responsible for managing and distributing work across the executors
- If the code requires the display of computed results to the user, the driver will collect the computed results from the executors and merge them together
- It can run on a **worker** node on the cluster or on the **client** node that launched the application

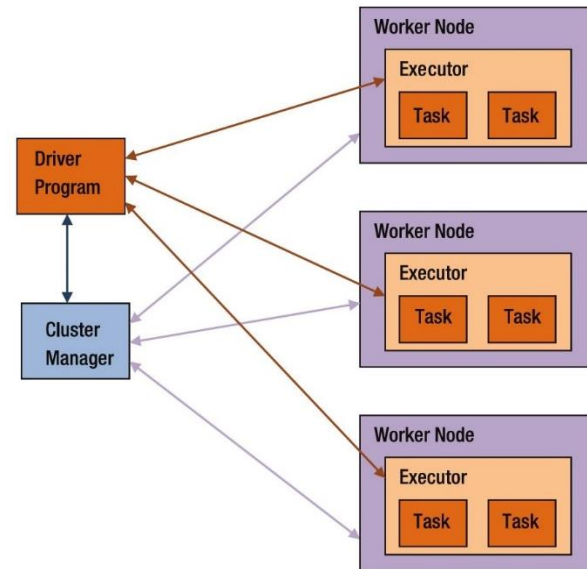
Executors

- Executors are JVM processes that Spark creates on worker nodes to run Spark applications



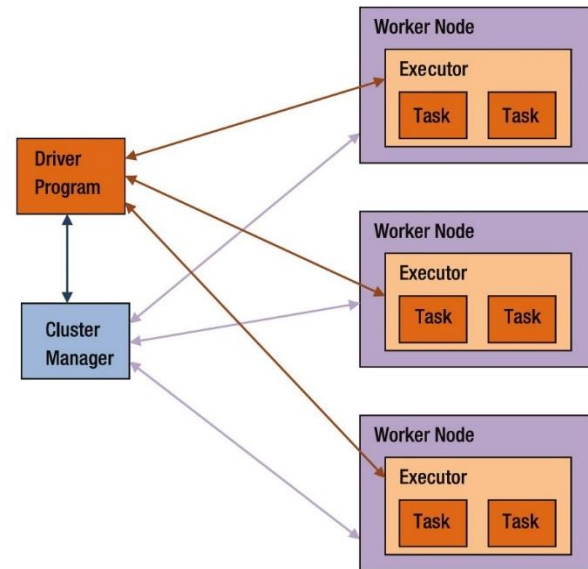
Executors

- Executors are JVM processes that Spark creates on worker nodes to run Spark applications
- Each executor caches portion of the data in memory and/or disk



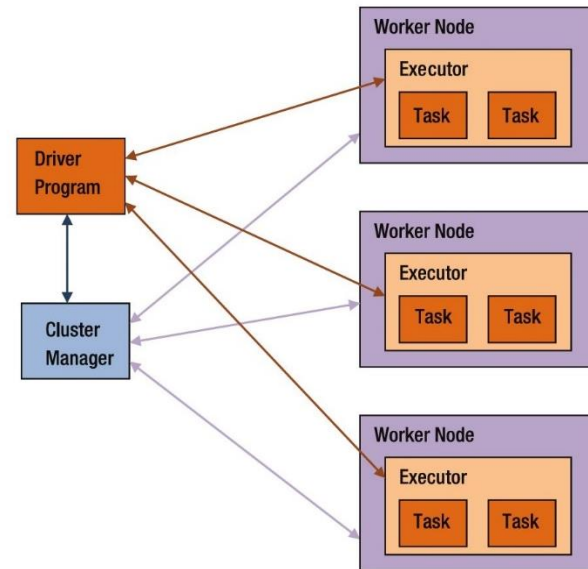
Executors

- Executors are JVM processes that Spark creates on worker nodes to run Spark applications
- Each executor caches portion of the data in memory and/or disk
- When a Spark application terminates, all the executors created for it also terminate



Executors

- Executors are JVM processes that Spark creates on worker nodes to run Spark applications
- Each executor caches portion of the data in memory and/or disk
- When a Spark application terminates, all the executors created for it also terminate
- A node can have multiple executors



Tasks

- A task is the smallest unit of work that driver sends to an executor
- It is executed in an executor on a worker node
- Each task performs some computations
- Result can be sent back to driver
- Spark creates a task per data partition
- An executor runs one or more tasks concurrently


SparkSession

- SparkSession is a Spark object that represents your entrance point to writing Spark code
- It is the driver process of your Spark application
- In interactive mode, a SparkSession object, named **spark**, is automatically created
- There is a one-to-one correspondence between a SparkSession and a Spark Application

SparkSession

C:\Windows\system32\cmd.exe - spark-shell

```
C:\Users\M>spark-shell
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context Web UI available at http://10.0.0.250:4040
Spark context available as 'sc' (master = local[*], app id = local-1694049401599).
Spark session available as 'spark'.
Welcome to
```



version 3.4.1

Command Prompt - spark-shell

```
scala> spark
res1: org.apache.spark.sql.SparkSession = org.apache.spark.sql.SparkSession@32b112a1
scala>
```

SparkSession

- Creating a range of numbers

```
scala> val myRange = spark.range(1000).toDF("number")  
myRange: org.apache.spark.sql.DataFrame = [number:  
bigint]
```

SparkSession

- Creating a range of numbers

```
scala> val myRange = spark.range(1000).toDF("number")  
myRange: org.apache.spark.sql.DataFrame = [number: bigint]
```

- DF represents a distributed collection
- DF is partitioned and stored on different executors

DataFrame

- A DataFrame is a Structured API
- It represents a table of data with rows and columns
- A schema specifies the columns and their types
- A DataFrame can span thousands of computers

Partitions

- To allow parallelism, Spark breaks up the data into chunks called partitions

Partitions

- To allow parallelism, Spark breaks up the data into chunks called partitions
- A partition is part of the data that sits on one computer

Partitions

- To allow parallelism, Spark breaks up the data into chunks called partitions
- A partition is part of the data that sits on one computer
- If you have **one partition** and many executors, Spark will have a parallelism of **only one**

Partitions

- To allow parallelism, Spark breaks up the data into chunks called partitions
- A partition is part of the data that sits on one computer
- If you have one partition and many executors, Spark will have a parallelism of only one
- If you have many partitions but only one executor, Spark will still have a parallelism of only one

Partitions

- To allow parallelism, Spark breaks up the data into chunks called partitions
- A partition is part of the data that sits on one computer
- If you have one partition and many executors, Spark will have a parallelism of only one
- If you have many partitions but only one executor, Spark will still have a parallelism of only one
- When we write code, we do not manipulate the individual partitions