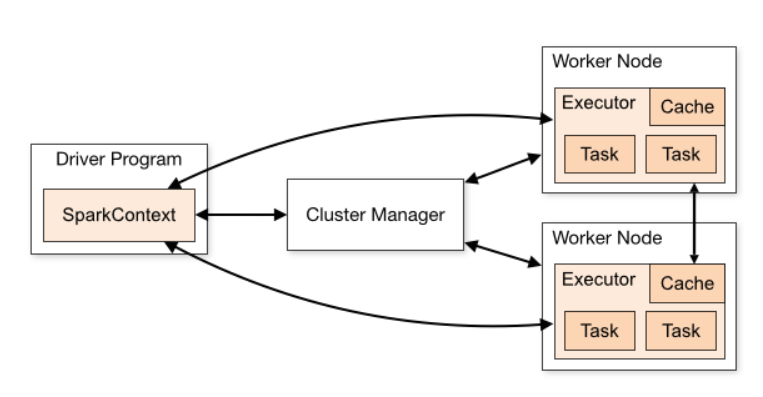
Assignment-12-2

1. Spark - how are worker, executor and task related to each other

Spark uses a master/slave architecture. It has one central coordinator (Driver) that communicates with many distributed workers (executors). The driver and each of the executors run in their own processes.



**DRIVER**

The driver is the process where the main method runs. First it converts the user program into tasks and after that it schedules the tasks on the executors.

You create a SparkContext object.

**EXECUTORS**

Executors are worker nodes' processes in charge of running individual tasks in a given Spark job. They are launched at the beginning of a Spark application and typically run for the entire lifetime of an application. Once they have run the task they send the results to the driver. They also provide in-memory storage for RDDs that are cached by user programs through Block Manager.

**CLUSTER MANAGER**

The layer in between the driver and the worker is the cluster manager. It manages resources of workers and other machines.

There are three options: 1. YARN 2. Standalone – Spark has its own cluster manager 3. Apache Mesos – open source

**How the three work can be explained by seeing the workflow:**

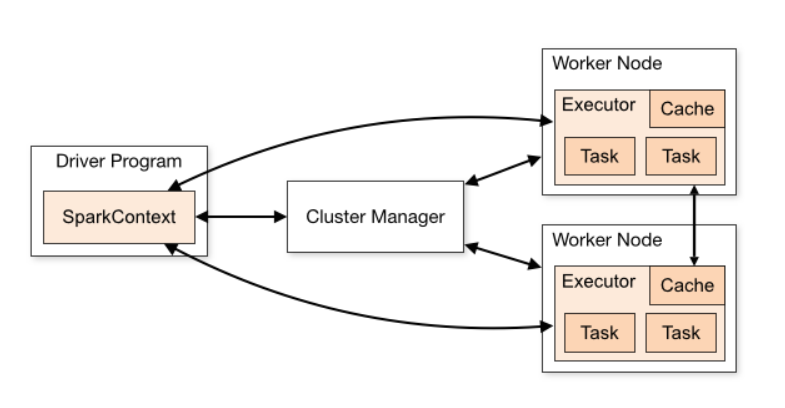
* A standalone application starts and instantiates a SparkContext instance (and it is only then when you can call the application a driver).
* The driver program ask for resources to the cluster manager to launch executors.
* The cluster manager launches executors.
* The driver process runs through the user application. Depending on the actions and transformations over RDDs task are sent to executors.
* Executors run the tasks and save the results.
* If any worker crashes, its tasks will be sent to different executors to be processed again. In the book "Learning Spark: Lightning-Fast Big Data Analysis" they talk about Spark and Fault Tolerance:
* With SparkContext.stop() from the driver or if the main method exits/crashes all the executors will be terminated and the cluster resources will be released by the cluster manager.

1. **What are the key features of Spark?**

* Spark is a cluster computing technology designed for fast computation
* Main feature is in-memory cluster computing – quite fast processing speed of application
* Speed − Spark helps to run an application in Hadoop cluster, up to 100 times faster in memory, and 10 times faster when running on disk. This is possible by reducing number of read/write operations to disk. It stores the intermediate processing data in memory.
* Supports multiple languages − Spark provides built-in APIs in Java, Scala, or Python. Therefore, you can write applications in different languages. Spark comes up with 80 high-level operators for interactive querying.
* Advanced Analytics − Spark not only supports ‘Map’ and ‘reduce’. It also supports SQL queries, Streaming data, Machine learning (ML), and Graph algorithms. Spark comes with tools for interactive query analysis, large-scale graph processing and analysis and real-time analysis – Spark Analytics Suite
* Resilient Distributed Datasets (RDD’s) – RDD’s are distributed objects that can be cached in-memory, across a cluster of compute nodes. They are the primary data objects used in Spark.

1. What is a Spark Driver?

The process running the main() function of the application and creating the SparkContext. A Spark "driver" is an application that creates a SparkContext for executing one or more jobs in the Spark cluster.



In the diagram above, the spark-master service is the Cluster Manager.

All jobs run on Spark using a driver process started by the API service.

The driver performs the following:

1. Connects to a cluster manager to allocate resources across applications

2. Acquires executors on cluster nodes processes run compute tasks, cache data

3. Sends app code to the executors

4. Sends tasks for the executors to run

1. What are the benefits of Spark over MapReduce?

|  |  |
| --- | --- |
| **Apache Spark** | **Apache Hadoop** |
| Easy to program and does not require any abstractions. | Difficult to program and requires abstractions. |
| Programmers can perform streaming, batch processing and machine learning ,all in the same cluster. | It is used for generating reports that help find answers to historical queries. |
| Has in-built interactive mode. | No in-built interactive mode except tools like Pig and Hive. |
| Executes jobs 10 to 100 times faster than Hadoop MapReduce. | Hadoop MapReduce does not leverage the memory of the hadoop cluster to the maximum. |
| Programmers can modify the data in real-time through Spark streaming. | Allows you to just process a batch of stored data. |

1. What is Spark Executor?

Executors are worker nodes' processes in charge of running individual tasks in a given Spark job. It is a process launched for an application on a worker node that runs tasks and keeps data in memory or disk storage across them. They are launched at the beginning of a Spark application and typically run for the entire lifetime of an application. Once they have run the task they send the results to the driver. They also provide in-memory storage for RDDs that are cached by user programs through Block Manager.

Executors run tasks. Task is a unit of work to execute.

Executor *typically* runs for the entire lifetime of a Spark application which is called **static allocation of executors** (but you could also opt in for dynamic allocation).

Executors report heart beat for active tasks to driver. They provide in-memory storage for RDDs that are cached in Spark applications. When an executor starts it first registers with the driver and communicates directly to execute tasks.