Title : Exploratory Big Data visualization using an automated containerised Spark cluster

Project summary

What is project about

In this project, we created a Apache cluster in containerised services of masters and workers. We used docker containers for creating independent containers which communicates each other through a docker network.

Apache spark and hadoop are used to process big data. Hadoop is already popular among data scientist for big data processing. But there are some limitations of Hadoop over Apache Spark. Spark is built on Hadoop and way much faster than (100X) Hadoop.It is capable of processing steaming data.

Why?

challenges and problem statement

Pandas which is most popular library among data scientist and data analyst for data processing. However, it is not good performant for processing big data. Apache spark

solutions? Goals

How/ Implementation

Architecture

Project Summary

The goal of this project was to create an Apache Spark cluster using Docker containers. Apache Spark is a distributed computing framework that enables processing of large datasets in parallel across multiple machines. Docker is a popular platform for building, shipping, and running applications in containers. By combining these technologies, we aimed to create a scalable and portable Spark cluster that can be easily deployed on different environments.

Problem Statement

Big data processing has become increasingly important in various industries, from finance to healthcare to retail. However, setting up a reliable and efficient Spark cluster can be a daunting task, especially for small or medium-sized enterprises that do not have dedicated IT infrastructure. Moreover, deploying a Spark cluster on different environments, such as local machines or cloud providers, can be challenging due to dependencies and configuration issues. Therefore, we wanted to provide a solution that simplifies the process of creating and deploying a Spark cluster, while also ensuring high performance and scalability.

Goals

The main goals of this project were:

To create a Spark cluster using Docker containers, with a focus on modularity, scalability, and ease of use.

To optimize the performance of the Spark cluster by fine-tuning the Spark configurations and the Docker settings.

To provide a comprehensive documentation and examples that demonstrate how to use the Spark cluster for different use cases, such as data processing, machine learning, and streaming.

Implementation

To achieve these goals, we followed the following steps:

Choose a Docker image that includes the necessary components for running Spark, such as Java, Scala, and Hadoop. We used the official jupyter/pyspark-notebook image, which provides a Jupyter notebook interface for interacting with Spark.

Create a Dockerfile that extends the chosen image and installs additional dependencies and configurations. For example, we installed the findspark package, which allows running Spark on a local machine without installing it globally. We also set up the SPARK\_HOME and PYSPARK\_PYTHON environment variables, which are required for running Spark in Python.

Use Docker Compose to define a multi-container setup that includes a Spark master node and one or more Spark worker nodes. Docker Compose allows defining the network, volume, and environment settings for each container, as well as linking them together.

Configure the Spark cluster by editing the spark-defaults.conf file, which includes various settings related to memory, parallelism, and serialization. We also set up the log4j properties to customize the logging output.

Test the Spark cluster by running some sample Spark jobs, such as word count, SQL queries, or machine learning algorithms. We used the Jupyter notebook interface to write and execute Spark code, and to visualize the results.

Conclusion

In conclusion, we have successfully created an Apache Spark cluster using Docker containers, which can be easily deployed and scaled on different environments. We have also optimized the performance of the Spark cluster by fine-tuning the Spark configurations and the Docker settings. Finally, we have provided a comprehensive documentation and examples that demonstrate how to use the Spark cluster for different use cases, such as data processing, machine learning, and streaming. We believe that this project can be useful for anyone who needs to process big data using Spark, without the need for dedicated infrastructure or complex setups.

About Apache

* Apache Spark is a distributed computing framework designed to process large datasets in parallel across multiple computers in a cluster. It is an open-source project developed by the Apache Software Foundation and is written in Scala, Java, and Python.
* Provides API for well-known languages ( Java, R, Python, Scala ,SQL) eliminating the need for training for newcomers
* In-memory processing model enables to perform much faster (100X) than traditional big data processing frameworks like Hadoop

There are several reasons why we use Apache Spark. One of the primary reasons is its ability to handle large datasets efficiently. Spark's in-memory processing model enables it to perform much faster than traditional big data processing frameworks like Hadoop, especially when dealing with iterative algorithms and machine learning workloads. Spark also provides a rich set of libraries, such as Spark SQL, Spark Streaming, and MLlib, which make it easier to work with different types of data and perform complex data processing tasks