**Cluster Computing with Spark** Gregoire Cousin : 18204188

22.10.2018 02:30:44 PM

[gregoire.cousin@ucdconnect.com](mailto:gregoire.cousin@ucdconnect.com)

*({1}: comment from paper)* Apache Spark is an expeditious, in memory data processing engine which allows data workers to efficiently execute streaming, machine learning or SQL workloads. Spark offers fast iterative access to datasets. Essentially Spark is a computational engine which can schedule and distribute applications consisting of many computational tasks across a variety of machines. Speed is extremely critical in handling substantial datasets. As it implies exploring datasets intelligently and seamlessly. Spark has various points of interest and detriments. A standout amongst the most imperative viewpoints that Spark has is its capacity to run calculations in memory. Spark additionally has a solid DAG execution engine that supports data flow in memory computing. This, in the end, makes processing speed much more modular and reliable. Spark empowers applications in Hadoop clusters to run up to a hundred times quicker in memory and its architecture has the ability to run ten times faster on disk.

Hadoop is a processing library that works with databases. A traditional SQL database mechanism will work on a linear method. Where one set of data needs to meet its identifier and vice versa. the rules of how these databases relate is stored as the identity of the database. In a Hadoop system the rules are based on a java program generated by the user or the database. The magic behind Hadoop is that data is spread across many different servers creating a cluster of data. The java program is also known to have the ability to map out where these files are located and what dependencies are related to the particular set of data when making a query. Hadoop is not the only processing library that can be implemented by Spark. *({2}: comment from paper)* Spark is also widely used in conjunction of Mesos, Cassandra, OpenStack and MySQL. However, you can now see why using a Spark system might be a great option for time management and optimization. Especially Since data is spread among multiple servers with each server being able to process data at its own time and in real-time. A developer has an extensive variety of languages to choose from made available by Spark. Sparks API is accessible in Scala, Java, Python and R. *({3}: comment from paper)* Spark introduces what is known as a Spark Session which is a passage point to access Sparks functionality. This enables software engineers and database developers to utilize Sparks DataFrame and Dataset APIs. Sparks DataFrame is distributed in a collection of datasets organized into name columns and rows. Which its syntax is close to a traditional DataFrame. *({4}: comment* *from paper)* These APIs have rich optimization making getting information quicker than a typical RDD system.

There are many more models provided by Spark. *({5}: comment from paper)* Spark offers multiple models to work with datasets such as: Data Frame API and SQL API. Both have their own syntax which a programmer can choose whether they would like to go towards a more programing approach which is similar to querying in a NoSQL environment or choose a more traditional approach which you will find similarities in MySQL based languages. However, you should get the same performance/results with either method. Spark has the capacity to run in multiple types of workflows making it one of the most modular platforms to use. Spark lets developers choose arbitrary operators such as mappers, reducers, joins, grid bites and filters. This computation makes it less demanding for any engineer to accomplish a wide array of computations including complex queries, iterative machine learning, and batch processing. Previously, this has only been achievable through multiple distributed systems. By doing this Spark makes it easier to process multiple models in a single platform seamlessly. By supporting these work processes in the same motor, Spark makes it less demanding to process numerous automation like processes in the same engine when running big data queries seamlessly. This is critical when needing to deal with annalistic pipelines. For instance, we can write a Spark application that orders information progressively through a Spark machine learning library. At the point when the data is injected from streaming sources via Spark streaming, the data scientist can then seamlessly query the resulting data through Spark SQL.

***References***

1. *Mohd RehanGhazi, Hadoop MapReduce and HDFS*, viewed 29 September 2018, <https://www.sciencedirect.com/science/article/pii/S1877050915006171#!>
2. *What is Apache Spark,* viewed 28 September 2018, <https://databricks.com/spark/about>
3. *Apache Spark*, viewed 27 September 2018, <http://spark.apache.org/>

Redo operations -