Apache Spark Tech Review

What is Apache Spark? The big data platform that crushed Hadoop

Fast, flexible, and developer-friendly, Apache Spark is the leading platform for large-scale SQL, batch processing, stream processing, and machine learning

Apache Spark is a data processing framework that can quickly perform processing tasks on very large data sets, and can also distribute data processing tasks across multiple computers, either on its own or in tandem with other distributed computing tools. These two qualities are key to the worlds of big data and machine learning, which require the marshalling of massive computing power to crunch through large data stores. Spark also takes some of the programming burdens of these tasks off the shoulders of developers with an easy-to-use API that abstracts away much of the grunt work of distributed computing and big data processing.

Apache Spark architecture

At a fundamental level, an Apache Spark application consists of two main components: a driver, which converts the user's code into multiple tasks that can be distributed across worker nodes, and executors, which run on those nodes and execute the tasks assigned to them. Some form of cluster manager is necessary to mediate between the two.

Out of the box, Spark can run in a standalone cluster mode that simply requires the Apache Spark framework and a JVM on each machine in your cluster. However, it's more likely you'll want to take advantage of a more robust resource or cluster management system to take care of allocating workers on demand for you. In the enterprise, this will normally mean running on Hadoop YARN (this is how the Cloudera and Hortonworks distributions run Spark jobs), but Apache Spark can also run on Apache Mesos, Kubernetes, and Docker Swarm.

If you seek a managed solution, then Apache Spark can be found as part of Amazon EMR, Google Cloud Dataproc, and Microsoft Azure HDInsight. Databricks, the company that employs the founders of Apache Spark, also offers the Databricks Unified Analytics Platform, which is a comprehensive managed service that offers Apache Spark clusters, streaming support, integrated web-based notebook development, and optimized cloud I/O performance over a standard Apache Spark distribution.

Apache Spark builds the user's data processing commands into a Directed Acyclic Graph, or DAG. The DAG is Apache Spark's scheduling layer; it determines what tasks are executed on what nodes and in what sequence.

Spark vs. Hadoop: Why use Apache Spark?

It's worth pointing out that Apache Spark vs. Apache Hadoop is a bit of a misnomer. You'll find Spark included in most Hadoop distributions these days. But due to two big advantages, Spark has become the framework of choice when processing big data, overtaking the old MapReduce paradigm that brought Hadoop to prominence.

The first advantage is speed. Spark's in-memory data engine means that it can perform tasks up to one hundred times faster than MapReduce in certain situations, particularly when compared with multi-stage jobs that require the writing of state back out to disk between stages. In essence, MapReduce creates a two-stage execution graph consisting of data mapping and reducing, whereas Apache Spark's DAG has multiple stages that can be distributed more efficiently. Even Apache Spark jobs where the data cannot be completely contained within memory tend to be around 10 times faster than their MapReduce counterpart.

The second advantage is the developer-friendly Spark API. As important as Spark's speedup is, one could argue that the friendliness of the Spark API is even more important.

Spark SQL

Originally known as Shark, Spark SQL has become more and more important to the Apache Spark project. It is likely the interface most commonly used by today's developers when creating applications. Spark SQL is focused on the processing of structured data, using a dataframe approach borrowed from R and Python (in Pandas). But as the name suggests, Spark SQL also provides a SQL2003-compliant interface for querying data, bringing the power of Apache Spark to analysts as well as developers.

Alongside standard SQL support, Spark SQL provides a standard interface for reading from and writing to other datastores including JSON, HDFS, Apache Hive, JDBC, Apache ORC, and Apache Parquet, all of which are supported out of the box. Other popular stores—Apache Cassandra, MongoDB, Apache HBase, and many others—can be used by pulling in separate connectors from the Spark Packages ecosystem.

Migration from Hadoop Hive to Spark

In normal hive production grade code, it is always in .hql format, which are getting triggered by some shell script or using Hadoop scheduler 'oozie' or by using some external scheduling tools like ControlM. To migrate hive to spark in very simple, i.e. no need to change the origin .hql codes. We can write parsing in spark to read .hql files and convert them to spark sql. Then deploy the spark code using spark submit with proper environment properties, like number of

vcore, number of executers, executer memory and driver memory. This will not only make the code optimize but also help in debugging the code very easily.

Improve Spark Performace

- Use DataFrame's instead of RDDs
- Avoid using Regex's
- Put the Largest Dataset on the Left
- Utilize Broadcast Joining for joining Smaller Datasets to Larger Ones
- Use Caching
- Set the spark.sql.shuffle.partitions Configuration Parameter
- COMPUTE STATISTICS of Tables Before Processing

Reference:

https://spark.apache.org/
https://www.infoworld.com/article/3236869/what-is-apache-spark-the-big-data-platform-that-crushed-hadoop.html