Big Data: Spark vs. Hadoop

Both Spark and Hadoop are major players in the big data handling fields. On the one hand, Spark is faster, uses memory and is relatively easy to use. However, Hadoop uses some processes in managing data making are relatively slower than the spark. Additionally, it is essential to understand the subtle differences in these two tools before ultimately deciding which one to use in your big data management.

Neither Spark nor Hadoop is mutually exclusive of each other. The pairing of these two data management tools can be used together making them a great management of big data applications that are continuously maturing with the invention of new technology. Here is a list of core characteristics and how they contrast with each other with the two tools.

1. Distributed Storage System

Does Spark provide its own distributed storage system? No! Distributed storage is a core function many of the modern big data projects need to fill in in their bucket list. It allows a vast multi-petabyte data storage across an infinite number of computer hard drives; an advantage to which you accrue than when having all your data stored in one big device. Having distributed storage means you can add more drives to the parent network as data grows in size.

Hadoop gives its users distributed data collection across several nodes within a cluster of commodity servers; meaning they can always enjoy the benefits that accompany it. Spark on the other hand will require a third party data distribution storage system as it does not have its file distribution system. As a consequence, its users will need to install Hadoop on top of it to allow its advanced analytics apps to make use of the data stored using Hadoop Distributed File System (HDFS).

1. Speed

Spark is reported to work 100 times faster than Hadoop in given circumstances. While Spark operates on the whole data in one swoop, Hadoop incorporates the use of MapReduce in processing data. MapReduce analyzes data in steps making it relatively slower than the process in Spark.

When using Hadoop, the data workflow begins by reading data from cluster, performing an operation, writing the results of the operation to the cluster, reading the updated data from the cluster then a whole lot of other processes succeed these; making it lengthy than in Spark where similar operations are performed but in a single step, for instance, it reads data, performs operation on data then writes the data back into the database.

Also, spark includes its visual compilation system to view data both as graphs and collections. You may not need Spark if you only need static data and can wait for the batch operations to complete.

1. Fault Tolerance

Spark and Hadoop use different approaches for their fault tolerance operations.

* It employs MapReduce that in turn uses TaskTrackers that provide heartbeats to the JobTracker.
* Should a heartbeat be missed, the JobTracker assesses all pending and in progress operations then reschedules them to a different TaskTracker to continue with the process.
* Though the process can significantly increase the completion times for activities that have a single failure, it improves efficiency in fault tolerance.

On the other hand, Spark employs Resilient Distributed Datasets (RDDs). RDDs are collections of elements that are tolerant and can be operated on in parallel. It contains five main properties including;

* List of partitions,
* A function computing each split,
* List of dependencies,
* a free partitioner for key-value RDDs and
* An optional list of preferred locations to calculate each split.

They can also reference datasets in secondary storage systems such as external drives, HDFS, a shared file system, and HBakey-values or any data source that can offer a Hadoop input format.

The differences in the two data tools create a data failure recovery system that is also different. Hadoop is resilient to failures and system faults because every operation involves data being written to a disk while Spark allows for full data recovery from errors and failures.

1. Scalability and Security

The debate does not always erupt when the scalability of Hadoop and Spark are brought into question. MapReduce and Spark are scalable using the HDFS.

* Yahoo's Hadoop cluster has 42,000 nodes and speculation has it that it will continue growing. It is difficult to estimate what magnitude this can reach.
* Spark's most significant known cluster is five times smaller than that of Hadoop at 8,000 nodes.

Big data is continuously growing, and it is expected that cluster sizes will increase to maintain throughput expectations.

While reports by some of it users claim using Kerberos authentication by Hadoop is somehow challenging to manage. It is worth noting that organizations can leverage LDAP and Active Directory Kerberos for authentication offered by third parties, who also provide encryption for data at rest and in-flight.

Hadoop's Distributed File System uses a traditional file permission model to support Access Control Lists (ACLs). Hadoop also provides Service Level Authorization to clients to ensure they have the right permissions to access data. On the other hand, Sparks Security is still at its baby steps and only supports authentication via password authentication (shared secret).

Spark's security is a bit infant as it currently only supports authentication via shared secret (password authentication). A security perk Spark can enjoy the ability to run in HDFS. It can also use HDFS ACL and Kerberos authentication too.

It might seem that Spark is the outright tool to use for big data application at first glance because of:

* Its agility,
* Speed,
* Relative ease of use among other factors.

However, it is also prudent to note that Hadoop's MapReduce creates an environment conducive for low-cost operations. Nevertheless, it also provides features not provided by Spark enabling it to make inroads into the big data market for medium-sized and big businesses that require control of big data. If you take a closer look at both Spark and Hadoop, one cannot be faulted in thinking that their designers made them be used together but parallel to each other.