Apache Hadoop YARN: Yet Another Resource Negotiator

Apache Hadoop is one of the open source which implements the MapReduce and mainly focused on handling the unusual scale which is required for indexing the web crawls. There are 10 requirements for managing the resource infrastructure for Hadoop which are:

1. **Scalability:** In Hadoop MapReduce, scalability is the original driver which is a major component of Apache Hadoop.
2. **Multi-tenancy:** The Hadoop YARN supports the multiple users and applications which can share all the resources of pool in a cluster.
3. **Serviceability:** One of the Hadoop platform features which this paper discusses is that it has an ability to upgrade the Hadoop platform itself without any need of all other application programs in order to simultaneously upgrade.
4. **Locality awareness:** The efficiency through the placement of computing the process closer to the needed data. It is a key requirement for YARN.
5. **High cluster utilization:** It is a top priority for YARN and it was the forceful reason for retiring HoD. The resource manager handles master application failures and node manager failures.
6. **Reliability/Availability:** A lot of effort requires for monitoring the jobs for dysfunction and avoids the requirement for manual inputs. It is very hard for operating a larger and multiple Hadoop clusters.
7. **Secure and auditable operation:** An authorization model generally provides the secure and auditable operation feature for handling the larger clusters which are mostly multi-tenant.
8. **Support for Programming Model Diversity:** Users can write MapReduce program with different programming languages which is handle by Hadoop YARN because of its support for programming model diversity feature
9. **Flexible resource model:** It is not just stick to 'map' and 'reduce' slots. The fungible resources can complicate the scheduling but also helps the allocator for cluster packing tightly.
10. **Backward compatibility:** For avoiding the traps of "second system syndrome", the new architecture reuses the old code from existing frameworks for managing the system in familiar patterns.

Apache Spark: A Unified Engine for Big Data Processing

Apache Spark project purpose was to design a unified engine for distributed data processing. The spark programming model is kind a similar to MapReduce but has an enhanced feature which helps in to manage the data sharing abstraction which is known as "Resilient Distributed Datasets" commonly known as "RDD". The RDD is the major component of Apache Spark as it can automatically recovers in case of failures. It has a fault tolerant collection of partitioned objects. Because of the RDD, it can support High-Level libraries such as MLlib. According to this paper if we compare the performance of logistic regression in Hadoop MapReduce vs Spark for 100GB of data on 50 m2.4xlarger EC2 nodes, for up to 20 number of iterations and 2500 secs of running time the Spark shows less amount of processing time with respect to the Hadoop. The Apache spark can support Batch processing as well as Stream processing as well making it more efficient than the Hadoop. The core of Spark API was majorly based on the functionality of programming over across the distributed collections which contains the objects such as Scala, Java, or Python. The performance optimizations is much better in Spark against the Hadoop. Also the local storage of Spark consists of each node memory of approximately 50GB/s of bandwidth, also 10 to 20 local disks for approximately 1GB/s to 2GB/s of disk bandwidth. The spark can handles the interactive queries more easily than Hadoop.

I pledge on my honor that I have not given or received any unauthorized assistance on this assignment/examination. I further pledge that I have not copied any material from a book, article, the Internet or any other source except where I have expressly cited the source.

Signature: Janmejay Mohanty Date: 21st February 2022