Spark Performance tuning guidelines

1. *Avoid very small and very large executor.*
2. *Normally choose different CPU cores and memory size per executor has no big performance difference.*
3. *Normally 3 to 6 executors per node is a reasonable configuration, depends on the CPU cores and memory size per executor.*
4. *Leave some of the resources (cores and memory) to OS.*
5. *Compression*
6. *Kryo Serialization*

*Use the power of Tungsten-To make the most out of Tungsten we pay attention to the following:*

*-Use Dataset structures rather than DataFrames*

*-Avoid User-Defined Functions (UDFs) as much as possible*

*We regularly use small DataFrames, for example when we want to cross a billion auctions with a website list we choose to broadcast the latter to all the executors and avoid a shuffle.*

*Handling Skew in the Join*

*To handle skew in the join keys, you can specify the hint ` /\*+ SKEW ('<table\_name>') \*/ ` for a join that describes the column and the values upon which skew is expected. Based on that information, the engine automatically ensures that the skewed values are handled appropriately.*

*You can specify the hint in the following formats:*

*Format 1:*

*/\*+ SKEW('<tableName>') \*/*

*Copy to clipboard*

*This shows that all the columns in a given table are skewed and the value on which they are skewed is not known. With this hint, the Spark optimizer tries to identify the values on which the column involved in the join is skewed. This operation is performed when the Spark optimizer identifies that a column is involved in the join and then it samples the data on the table.*

*Example: In a query, suppose there is a table t1 where all columns involved in the join are skewed. But the skew values are unknown. In this case, you can specify the skew hint as ` /\*+ SKEW('t1') \*/ `.*

*Format 2:*

*/\*+ SKEW ('<tableName>', (<COLUMN-HINT>), (<ANOTHER-COLUMN-HINT>)) \*/*

*Configuring the Spark External Shuffle Service--spark.shuffle.service.enabled true*

*Continuously Running Spark Streaming Applications*

*Set yarn.resourcemanager.app.timeout.minutes=-1 as an Hadoop override at the Spark cluster leve*

*Spark Counters for Verifying Performance*

*test("wordcount perf") {*

*val listener = new PerfListener()*

*sc.addSparkListener(listener)*

*doWork(sc)*

*println(listener)*

*assert(listener.totalExecutorRunTime > 0)*

*assert(listener.totalExecutorRunTime < 10000)*

*}*

<https://mapr.com/blog/performance-tuning-apache-kafkaspark-streaming-system/>