with Big Data ***[Ge…](#) [Exclusive Offers o…](#) [Big Data Course…](#)***

1. Objective

The Spark SQL performance can be affected by some tunin… consideration. To represent our data efficiently, it uses the knowledge of types ve… effectively. Spark SQL

plays a great role in th optimization of querie This blog also covers what is Spark SQL performance tuning ar various factors to tune the Spark SQL performance in **Apache Spark**.

Before reading this blog I would recommend you to read **Spark Performance Tuning.** It will increase your understanding of Spark and help further in this blog.

Spark SQL Performance Tuning – Learn Spark SQL

2. What is Spark SQL Performance Tuning?

Spark SQL is the module of Spark for structured data processing. The high-level query language and additional type information makes Spark SQL more efficient. Spark SQL translates commands into codes that are processed by executors. Some tuning consideration can affect the Spark SQL performance. To represent our data efficiently, it also uses the knowledge of types very effectively. Spark SQL plays a great role in the optimization of queries. The Spark SQL makes use of **in-memory** columnar storage while caching data. The in-memory columnar is a feature that allows storing the data in a

columnar format, rather than row format. The columnar storage allows itself extremely well to analytic queries found in business intelligence product. Using columnar storage, the data takes less space when cached and if the query depends only on the subsets of data, thus Spark SQL minimizes the data read.

3. Options for Performance Tuning in Spark SQL

There are several different Spark SQL performance tuning options are available:

i. spark.sql.codegen

The default value of *spark.sql.codegen* is **false**. When the value of this is true, Spark SQL will compile each query to Java bytecode very quickly. Thus, improves the performance for large

queries. But the issue with codegen is that it slows down with very short queries. This happens because it has to run a compiler for each query.

ii. spark.sql.inMemorycolumnarStorage.compressed

The default value of *spark.sql.inMemorycolumnarStorage.compressed* is **true**. When the value is true we can compress the in-memory columnar storage automatically based on statistics of the data.

iii. spark.sql.inMemoryColumnarStorage.batchSize

The default value of *spark.sql.inMemoryColumnarStorage.batchSize* is **10000**. It is the batch size for columnar caching. The larger values can boost up memory utilization but causes an out-of-memory problem.

iv. spark.sql.parquet.compression.codec

The *spark.sql.parquet.compression.codec* uses default snappy compression. Snappy is a library which for compression/decompression. It mainly aims at very high speed and

reasonable compression. In most compression, the resultant file is 20 to 100% bigger than other inputs although it is the order of magnitude faster. Other possible option includes uncompressed, gzip and lzo.

Note:

In Spark SQL as more **optimizations** are performed automatically, it is possible that following options can get vanished in the further release:

- sql.files.maxPartitionBytes,
- sql.files.openCostI
- sql.autoBroadcastJoinThreshold,
- sql.shuffle.partitions,
- sql.broadcastTimeout.

4. Conclusion

In conclusion to Apache Spark SQL, caching of data in in-memory columnar storage improves the overall performance of the Spark SQL applications. Hence, Using the above mention

operations it's easy to achieve the optimization in Spark SQL.

See Also-

- RDD Persistence and Caching Mechanism in Spark
- Spark SQL DataFrame Tutorial
- Spark SQL DataSet Tutorial

Reference for Spark

You give me 15 seconds I promise you best tutorials

Please share your happy experience on **Google** | **Facebook**

Tags: apache spark Spark Spark SQL optimization Spark SQL Performance tuning spark-sql

LEAVE A REPLY

Comment

Name * **Email** *