

# What is the difference between spark.sql.shuffle.partitions and spark.default.parallelism?

Asked 7 years, 2 months ago Modified 1 year, 4 months ago Viewed 227k times

What's the difference between `spark.sql.shuffle.partitions` and `spark.default.parallelism`?

I have tried to set both of them in `SparkSQL`, but the task number of the second stage is always 200.

134

performance apache-spark hadoop apache-spark-sql

Share Edit Follow

edited May 1, 2020 at 16:58

 user4157124  
2,904 ● 14 ● 30 ● 44

asked Aug 16, 2017 at 2:22

 Edison  
1,355 ● 2 ● 10 ● 8

## 4 Answers

Sorted by: Highest score (default)

From the answer [here](#), `spark.sql.shuffle.partitions` configures the number of partitions that are used when shuffling data for joins or aggregations.

151

`spark.default.parallelism` is the default number of partitions in `RDD`'s returned by transformations like `join`, `reduceByKey`, and `parallelize` when not set explicitly by the user. Note that `spark.default.parallelism` seems to only be working for raw `RDD` and is ignored when working with dataframes.

If the task you are performing is not a join or aggregation and you are working with dataframes then setting these will not have any effect. You could, however, set the number of partitions yourself by calling `df.repartition(numOfPartitions)` (don't forget to assign it to a new `val`) in your code.

To change the settings in your code you can simply do:

```
sqlContext.setConf("spark.sql.shuffle.partitions", "300")
sqlContext.setConf("spark.default.parallelism", "300")
```

Alternatively, you can make the change when submitting the job to a cluster with `spark-submit`:

```
./bin/spark-submit --conf spark.sql.shuffle.partitions=300 --conf
spark.default.parallelism=300
```

Share Edit Follow

edited Aug 24, 2017 at 2:48

answered Aug 16, 2017 at 3:16



Shaído

28.2k ● 25 ● 74 ● 80

11 Any advice on what this number should be...? – CpILL Mar 21, 2019 at 23:08

1 @CpILL: It depends on your situation, some more information and useful links can be found here: [stackoverflow.com/questions/35800795/...](https://stackoverflow.com/questions/35800795/...). In particular, the Spark [documentation on tuning](#) recommends 2-3 tasks per CPU core in the cluster. – Shaído Mar 22, 2019 at 1:21

1 The default number of partition you can decide by available ram across the node that you can provide to executors. So here is the simple formula no. of partition =(available RAM in MB)/256 that means 256MB per partition. – Amit khandelwal Aug 7, 2019 at 0:59 ✎

7 A little confused here. spark.sql.shuffle.partitions configures the partitions used for joins or aggregations. You then say spark.default.parallelism is used for transformations like join, reduceByKey. Arent those joins or aggregations as well? – vi\_ral Jun 1, 2020 at 14:50 ✎

@Shaído Even if I apply this --conf spark.sql.shuffle.partitions=300 , still I see a lot of stages being generated and most of them does not have records , which is leading to Java heap OOM exception , how to handle this situation? – BdEngineer Dec 18, 2020 at 4:43



21

**spark.default.parallelism** is the default number of partition set by spark which is by default 200. and if you want to increase the number of partition than you can apply the property **spark.sql.shuffle.partitions** to set number of partition in the spark configuration or while running spark SQL.



Normally this **spark.sql.shuffle.partitions** it is being used when we have a memory congestion and we see below error: spark error:java.lang.IllegalArgumentException: Size exceeds Integer.MAX\_VALUE

so set your can allocate a partition as 256 MB per partition and that you can use to set for your processes.

also If number of partitions is near to 2000 then increase it to more than 2000. As spark applies different logic for partition < 2000 and > 2000 which will increase your code performance by decreasing the memory footprint as data default is highly compressed if >2000.

Share Edit Follow

edited Aug 7, 2019 at 1:15

answered Aug 7, 2019 at 1:03



Amit khandelwal

534 ● 5 ● 7

2 Hi. A bit late, but do you have any source concerning the different behavior of spark (2.1.0 if possible :D) when the number of partitions is above 2000 ? I can't find anything. – Itération 122442 Jun 23, 2021 7:10

I have seen this mentioned before and the only reference I could find was the source itself, here: [github.com/apache/spark/blob/...](https://github.com/apache/spark/blob/...) – Andy Kershaw Oct 1, 2021 at 13:01

Note that the code listed here has changed and now relies on `config.SHUFFLE_MIN_NUM_PARTS_TO_HIGHLY_COMPRESS` which can be configured by `spark.shuffle.minNumPartitionsToHighlyCompress` (see [github.com/apache/spark/blob/master/core/src/main/scala/org/...](https://github.com/apache/spark/blob/master/core/src/main/scala/org/...)). This blog post is one of the few references on how to use it [waitingforcode.com/apache-spark/...](https://waitingforcode.com/apache-spark/...) – papirrin Dec 21, 2023 at 23:32



To add onto what some great answers have already posted:

## 17 Summary



- `spark.sql.shuffle.partitions`:
  - Determines how many output partitions you will have after doing wide transformations on **Dataframes/Datasets** by default.
  - Its default value is 200.
- `spark.default.parallelism`:
  - Is a more complicated parameter that lives more "deep down" in Spark. It influences:
    - how many partitions you will have after doing wide transformations on **RDDs** if you don't specify an amount
    - how many partitions `sc.parallelize` creates
    - how many partitions are read in when doing `spark.read.csv`, ...
  - Its default value depends on what type of operation you do on which type of cluster.

## A bit more detail

`spark.sql.shuffle.partitions`

From the [docs](#):

The default number of partitions to use when shuffling data for joins or aggregations. Note: For structured streaming, this configuration cannot be changed between query restarts from the same checkpoint location.

As can be seen in Spark 3.3.1's (newest version at the time of this post) [SQLConf.scala](#), `spark.sql.shuffle.partitions` has a default value of 200.

```
val SHUFFLE_PARTITIONS = buildConf("spark.sql.shuffle.partitions")
    .doc("The default number of partitions to use when shuffling data for joins or
aggregations. " +
```



```
"Note: For structured streaming, this configuration cannot be changed between
query " +
  "restarts from the same checkpoint location.")
.version("1.1.0")
.intConf
.checkValue(_ > 0, "The value of spark.sql.shuffle.partitions must be
positive")
.createWithDefault(200)
```

**Conclusion:** This is a value with which you can have immediate impact on your wide transformations (join, sort, ...) on **Dataframes/Datasets**. You'll be able to configure the amount of output partitions of those wide transformations with this parameter.

### `spark.default.parallelism`

From the [docs](#):

Default number of partitions in RDDs returned by transformations like join, reduceByKey, and parallelize when not set by user.

About its default value:

For distributed shuffle operations like `reduceByKey` and `join`, the largest number of partitions in a parent RDD. For operations like `parallelize` with no parent RDDs, it depends on the cluster manager:

- Local mode: number of cores on the local machine
- Mesos fine grained mode: 8
- Others: total number of cores on all executor nodes or 2, whichever is larger

So we already see this parameter is a bit more complicated. It has no real default value, but you can set it. This becomes clearer if we look at a bit of code.

In [SparkContext.scala](#), we see how `defaultParallelism` is defined:

```
/** Default level of parallelism to use when not given by user (e.g. parallelize
and makeRDD). */
def defaultParallelism: Int = {
  assertNotStopped()
  taskScheduler.defaultParallelism
}
```

So we see that this `defaultParallelism` is dependent on the type of taskScheduler (like the docs state). Let's have a look at those:



- [local mode](#):

```
override def defaultParallelism(): Int =
  scheduler.conf.getInt("spark.default.parallelism", totalCores)
```

- [Mesos fine grained mode](#):

```
override def defaultParallelism(): Int =
  sc.conf.getInt("spark.default.parallelism", 8)
```

- [Others](#) (`CoarseGrainSchedulerBackend`):

```
override def defaultParallelism(): Int = {
  conf.getInt("spark.default.parallelism", math.max(totalCoreCount.get(), 2))
}
```

Ok, so now we understand that this value is a bit more complicated let's try to figure out when it is relevant:

- When doing wide transformations on **RDDs**, [join](#), [reduceByKey](#), [groupByKey](#), etc. all make use of a `defaultPartitioner` if no partitioning is given as input parameter. In that [defaultPartitioner](#), `spark.default.parallelism` is used to determine the amount of partitions.
- When calling [sc.parallelize](#) on a `Seq`. Depending on your cluster manager, you'll get a number of output partitions as explained above.
- When reading in data (for example `spark.read.csv`), it will have an impact on how many partitions will be read in. In [DataSourceScanExec](#)'s `createReadRDD` function, the amount of output partitions that will be read in are influenced by the `maxSplitBytes` function, which is in itself influenced by `spark.default.parallelism` as explained in [this SO answer](#).
- I'm sure it's used in more place but I hope this already gives more intuition about this parameter.

Share Edit Follow

edited Jun 24, 2023 at 0:33

answered Dec 28, 2022 at 11:19



Koedlt

5,893 ● 9 ● 21 ● 45

3

In case someone who might want to know, there's indeed a special situation when the setting `spark.sql.shuffle.partitions` might become invalid. When you restart a structured streaming spark application with the same checkpoint location, changing this term does not take effect. See more at <https://spark.apache.org/docs/latest/configuration.html#runtime-sql-configuration>

Share Edit Follow

edited Sep 19, 2022 at 10:52

answered Sep 19, 2022 at 1



lize su

31 ● 5

---

wow this is great information but this is really bad on spark side. stopping application and clearing checkpoints means lost data. so it is not acceptable. – [Dariusz Krynicki](#) Oct 11, 2022 at 14:24

---

