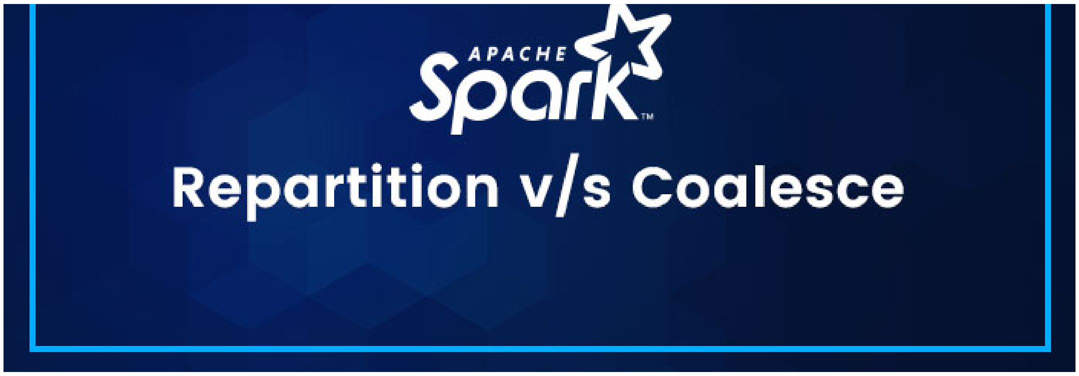
Apache Spark: Repartitioning v/s Coalesce



December 30, 2019 (https://blog.knoldus.com/apache-spark-repartitioning-v-s-coalesce/)

Divyansh Jain (https://blog.knoldus.com/author/divyanshjain837/)

Apache Spark (https://blog.knoldus.com/category/tech-blogs/big-data/apache-spark/), Big Data and Fast Data (https://blog.knoldus.com/category/tech-blogs/big-data/), Database (https://blog.knoldus.com/category/tech-blogs/database/), HDFS (https://blog.knoldus.com/category/tech-blogs/big-data/hdfs/), ML, AI and Data Engineering (https://blog.knoldus.com/category/tech-blogs/machine-learning/), NoSql (https://blog.knoldus.com/category/tech-blogs/database/nosql/), Scala (https://blog.knoldus.com/category/tech-blogs/functional-programming/scala/), Spark (https://blog.knoldus.com/category/tech-blogs/machine-learning/spark/)

Analytics (https://blog.knoldus.com/tag/analytics/), Apache Spark (https://blog.knoldus.com/tag/apache-spark/), Big Data Analytics (https://blog.knoldus.com/tag/big-data-analytics/), data analysis (https://blog.knoldus.com/tag/data-analysis/), fast data analytics (https://blog.knoldus.com/tag/fast-data-analytics/), partitioning in apache spark (https://blog.knoldus.com/tag/partitioning-in-apache-spark/), real time analytics (https://blog.knoldus.com/tag/real-time-analytics/), Spark SQL (https://blog.knoldus.com/tag/spark-sql/)

3 Comments (https://blog.knoldus.com/apache-spark-repartitioning-v-s-coalesce/#comments)

Reading Time: 3 minutes

Does partitioning help you increase/decrease the Job Performance?

Spark splits data into partitions and computation is done in parallel for each partition. It is very important to understand how data is partitioned and when you need to manually modify the partitioning to run spark applications efficiently.

Now, diving into our main topic i.e **Repartitioning v/s Coalesce**

What is Coalesce?

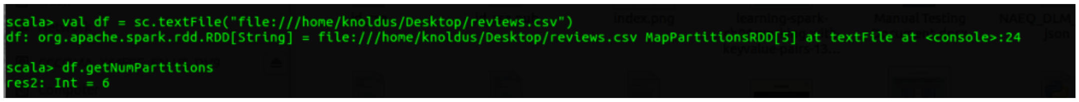
The coalesce method **reduces** the **number of partitions** in a DataFrame. Coalesce **avoids full shuffle**, instead of creating new partitions, it shuffles the data using Hash Partitioner (Default), and **adjusts into existing partitions**, this means it can **only decrease** the number of partitions.

What is Repartitioning?

out from existing partitions and **equally distributed** into **newly formed partitions**.

Where to use what?

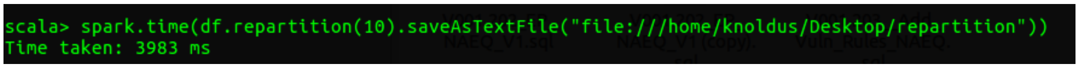
Let’s look at the below example for the answer.



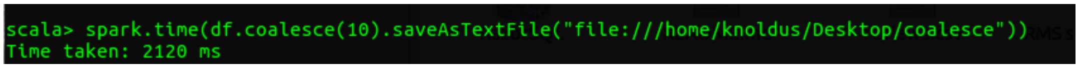
```
scala> val df = sc.textFile("file:///home/knoldus/Desktop/reviews.csv")
df: org.apache.spark.rdd.RDD[String] = file:///home/knoldus/Desktop/reviews.csv MapPartitionsRDD[5] at textFile at <console>:24

scala> df.getNumPartitions
res2: Int = 6
```

Now, if I manually pass the number of partitions to 10, see how the data gets distributed:



```
scala> spark.time(df.repartition(10).saveAsTextFile("file:///home/knoldus/Desktop/repartition"))
Time taken: 3983 ms
```



```
scala> spark.time(df.coalesce(10).saveAsTextFile("file:///home/knoldus/Desktop/coalesce"))
Time taken: 2120 ms
```

Comparatively, **coalesce** took less time as compared with **repartitioning**. And the data gets partitioned as below:

Repartitioning	Coalesce
19M repartition/part-00000	
19M repartition/part-00001	
19M repartition/part-00002	33M coalesce/part-00000
19M repartition/part-00003	29M coalesce/part-00001
19M repartition/part-00004	30M coalesce/part-00002
19M repartition/part-00005	31M coalesce/part-00003
19M repartition/part-00006	32M coalesce/part-00004
19M repartition/part-00007	33M coalesce/part-00005
19M repartition/part-00008	
19M repartition/part-00009	

If you observe above table when **repartitioned**, data over all the partitions are **equally populated**, but when we used **coalesce** the data is **not equally distributed**.

Also, if you observed above coalesce didn’t partition your data to **10 partitions** instead it created **6 partitions**. That means even if you provide a large number of partitions, it partitions your data to the default one in the above case it is **6**.

Now we understand the behavior and hence back to our initial question, where to use which function?

followed by others but the executor with **part-00005** will be still running meanwhile 1st executor will be idle. **Hence, the load is not balanced on executors equally.**

Repartition use case: All the **executors finish** the job at the same time, and the **resources** are **consumed equally** because all input partitions have the same size.

So, here is the answer:

- If you have loaded a dataset, includes huge data, and a lot of transformations that need an equal distribution of load on executors, you need to use **Repartition**.
- Once all the transformations are applied and you want to save all the data into fewer files(no. of files = no.of partitions) instead of many files, use **coalesce**.

So, this was all about **Repartitioning** & **Coalesce**. Hope to take the inputs from this blog you gonna better partition your data now to increase your Job performance.

In our next blog, we will be discussing **Windows Operations in Spark SQL.**

If you like this blog, please do show your appreciation by hitting like button and sharing this blog. Also, drop any comments about the post & improvements if needed. Till then HAPPY LEARNING.

in Share Share 12 Tweet

WhatsApp (https://blog.knoldus.com/apache-spark-repartitioning-v-s-coalesce/?share=jetpack-whatsapp&nb=1)

Telegram (https://blog.knoldus.com/apache-spark-repartitioning-v-s-coalesce/?share=telegram&nb=1)

More

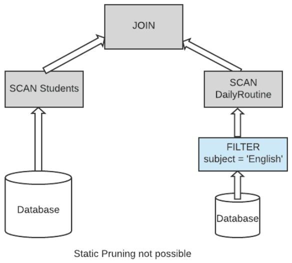
Related

reduce_by (/shuffling-and-repartitioning-of-rdds-in-apache-spark/?relatedposts_hit=1&relatedposts_origin=65773&relatedposts_position=0&relatedposts_hit=1&relatedposts_origin=65773&relatedposts_position=0&relatedposts_hit=1&relatedposts_origin=65773&relatedposts_position=0)

Shuffling and repartitioning of RDD's in apache spark (/shuffling-and-repartitioning-of-rdds-in-apache-spark/?relatedposts_hit=1&relatedposts_origin=65773&relatedposts_position=0&relatedposts_hit=1&relatedposts_origin=65773&relatedposts_position=0&relatedposts_hit=1&relatedposts_origin=65773&relatedposts_position=0)

June 19, 2015

In "Apache Spark"

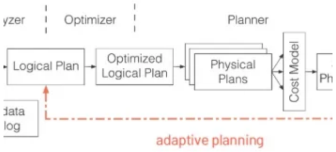


(/dynamic-partition-pruning-in-spark-3-0/?relatedposts_hit=1&relatedposts_origin=65773&relatedposts_position=1&relatedposts_hit=1&relatedposts_origin=65773&relatedposts_position=1&relatedposts_hit=1&relatedposts_origin=65773&relatedposts_position=1)

Dynamic Partition Pruning in Spark 3.0 (/dynamic-partition-pruning-in-spark-3-0/?relatedposts_hit=1&relatedposts_origin=65773&relatedposts_position=1&relatedposts_hit=1&relatedposts_origin=65773&relatedposts_position=1&relatedposts_hit=1&relatedposts_origin=65773&relatedposts_position=1)

May 16, 2020

In "Apache Spark"



(/adaptive-query-execution-aqe-in-spark-3-0/?relatedposts_hit=1&relatedposts_origin=65773&relatedposts_position=2&relatedposts_hit=1&relatedposts_origin=65773&relatedposts_position=2&relatedposts_hit=1&relatedposts_origin=65773&relatedposts_position=2)

Adaptive Query Execution (AQE) in Spark 3.0 (/adaptive-query-execution-aqe-in-spark-3-0/?relatedposts_hit=1&relatedposts_origin=65773&relatedposts_position=2&relatedposts_hit=1&relatedposts_origin=65773&relatedposts_position=2&relatedposts_hit=1&relatedposts_origin=65773&relatedposts_position=2)

June 30, 2020

In "Apache Spark"

Written by [Divyansh Jain](https://blog.knoldus.com/author/divyanshjain837/)

Divyansh Jain is a Software Consultant with experience of 1 years. He has a deep understanding of Big Data Technologies, Hadoop, Spark, Tableau & also in Web Development. He is an amazing team player with self-learning skills and a self-motivated professional. He also worked as Freelance Web Developer. He loves to play & explore with Real-time problems, Big Data. In his leisure time, he prefers doing LAN Gaming & watch movies.

3 thoughts on "Apache Spark: Repartitioning v/s Coalesce 4 min read "

Pingback: Repartitioning and Coalescing in Spark – Curated SQL (https://curatedsql.com/2019/12/31/repartitioning-and-coalescing-in-spark/)

Shannkar says:

January 4, 2020 at 1:13 PM (https://blog.knoldus.com/apache-spark-repartitioning-v-s-coalesce/#comment-39417)

Are you provide any big data training

Like

Divyansh Jain says:

January 8, 2020 at 10:03 AM (https://blog.knoldus.com/apache-spark-repartitioning-v-s-coalesce/#comment-39419)

Loading...

Comments are closed.

OUR OFFERINGS

High Performace System (https://www.knoldus.com/services/high-performance-systems)

Data Strategy & Analytics (https://www.knoldus.com/services/data-strategy-and-analytics)

Intelligence Driven Decisioning (https://www.knoldus.com/services/intelligence-driven-decisioning)

Platform Strategy (https://www.knoldus.com/services/platform-strategy)

Blockchain (https://www.knoldus.com/services/blockchain)

Expert Services (https://www.knoldus.com/services/expert-services)

Academy, Audit & Consulting (https://www.knoldus.com/services/academy)

KDP (https://www.knoldus.com/work/platforms-and-products/digital-platform)

KDSP (https://www.knoldus.com/work/platforms-and-products/data-science-platform)

PremonR (https://www.knoldus.com/work/platforms-and-products/premonr)

COMPANY

About Knoldus (https://www.knoldus.com/about-us)

Knolway (https://www.knoldus.com/about/process)

Case Studies (https://www.knoldus.com/work/case-studies)

Testimonials (https://www.knoldus.com/about/testimonials)

Careers (https://www.knoldus.com/company/careers)

Tech Expertise (https://www.knoldus.com/work/technologies)

News Room (https://www.knoldus.com/news)

Our Partners (https://www.knoldus.com/partners)

CSR (https://www.knoldus.com/about/corporate-social-responsibility)

LEARN

Why Knoldus? (https://www.knoldus.com/connect/why-knoldus)

Blog (https://blog.knoldus.com/)

Resources (https://www.knoldus.com/learn/resources)

KnolX (http://knolx.knoldus.com/session)

Webinars and Videos (https://www.knoldus.com/learn/webinars)

Podcast (https://www.knoldus.com/learn/podcasts)

Rust Times (https://rusttimes.com/)

Innovation Labs (https://www.knoldus.com/about/innovation-labs)

Knoldus Tech Hub (https://techhub.knoldus.com/)

Open Source Contributions (https://www.knoldus.com/work/open-source)

CONNECT

Contact Us (https://www.knoldus.com/connect/contact-us)

Engagement Model (https://www.knoldus.com/connect/engagement-models)

Follow us Here:

in (https://in.linkedin.com/company/knoldus) You Tube (https://www.youtube.com/channel/UCP4g5qGeUSY7OokXfim1QCQ) (https://twitter.com/Knolspeak?ref_src=twsrc%5Egoogle%7Ctwcamp%5Eserp%7Ctwgr%5Eauthor) (https://www.facebook.com/KnoldusSoftware/) (https://www.slideshare.net/knoldus) (https://www.instagram.com/knoldus_inc/?hl=en)

Subscribe to our newsletter

SUBSCRIBE (https://share.hsforms.com/1oRAE_GdIQsi3niRYzwv65g2u6gi)

Partners

_(https://www.lightbend.com/partners/system-integrators)

_(https://databricks.com/company/partners)

_(https://www.confluent.io/partners/)

_(https://www.docker.com/)

_(https://www.hashicorp.com/ecosystem/find-a-partner/)

_(https://www.ibm.com/partnerworld/public)

Privacy Policy (https://www.knoldus.com/privacy-policy) | Sitemap (https://www.knoldus.com/sitemap)