Home (/) About (https://sparkbyexamples.com/about-sparkbyexamples/) | { One stop for all Spark Examples }

Spark by {Examples} (https://sparkbyexamples.com/)

Spark (https://sparkbyexamples.com/)
PySpark (https://sparkbyexamples.com/pyspark-tutorial/)
Hive (https://sparkbyexamples.com/apache-hive-tutorial/)
HBase (https://sparkbyexamples.com/apache-hbase-tutorial/)
Kafka (https://sparkbyexamples.com/apache-kafka-tutorials-with-examples/)
FAQ's (https://sparkbyexamples.com/spark-questions/)
More (https://sparkbyexamples.com/)

Spark DataFrame Cache and Persist Explained

NNK (https://sparkbyexamples.com/author/admin/) - Apache Spark (https://sparkbyexamples.com/category/spark/)

Spark *Cache* and *Persist* are optimization techniques in DataFrame / Dataset for iterative and interactive Spark applications to improve the performance of Jobs. In this article, you will learn What is Spark `cache()` and `persist()`, how to use it in DataFrame, understanding the difference between `Caching` and `Persistance` (https://sparkbyexamples.com/spark/spark-difference-between-cache-and-persist/) and how to use these two with DataFrame, and Dataset using Scala examples.

Spark RDD – Read CSV (https://sparkbyexamples.com/apache-spark-rdd/spark-load-csv-file-into-rdd/)

Spark RDD – Create RDD (https://sparkbyexamples.com/apache-spark-rdd/different-ways-to-create-spark-rdd/)

Spark RDD – Create Empty RDD (https://sparkbyexamples.com/apache-spark-rdd/spark-how-to-create-an-empty-rdd/)

Spark RDD – Transformations (https://sparkbyexamples.com/apache-spark-rdd/spark-rdd-transformations/)

Spark RDD – Actions (https://sparkbyexamples.com/apache-spark-rdd/spark-rdd-actions/)

Spark RDD – Pair Functions (https://sparkbyexamples.com/apache-spark-rdd/spark-pair-rdd-functions/)

Spark RDD – Repartition and Coalesce (https://sparkbyexamples.com/spark/spark-repartition-vs-coalesce/)

Spark RDD – Shuffle Partitions (https://sparkbyexamples.com/spark/spark-shuffle-partitions/)

Spark RDD – Cache vs Persist (https://sparkbyexamples.com/spark/spark-difference-between-cache-and-persist/)

Spark RDD – Persistance Storage Levels (https://sparkbyexamples.com/spark/spark-persistence-storage-levels/)

Spark RDD – Broadcast Variables (https://sparkbyexamples.com/spark/spark-broadcast-variables/)

Spark RDD – Accumulator Variables (https://sparkbyexamples.com/spark/spark-accumulators/)

Spark RDD – Convert RDD to DataFrame (https://sparkbyexamples.com/

Though Spark provides computation 100 x times faster than traditional Map Reduce jobs, If you have not designed the jobs to reuse the repeating computations you will see degrade in performance when you are dealing with billions or trillions of data. Hence, we may need to look at the stages and use optimization techniques as one of the ways to improve performance.

Using `cache()` and `persist()` methods, Spark provides an optimization mechanism to store the intermediate computation of a Spark DataFrame so they can be reused in subsequent actions.

When you persist a dataset, each node stores it's partitioned data in memory and reuses them in other actions on that dataset. And Spark's persisted data on nodes are fault-tolerant meaning if any partition of a Dataset is lost, it will automatically be recomputed using the original transformations that created it.

Advantages for Caching and Persistence of DataFrame

Below are the advantages of using Spark Cache and Persist methods.

apache-spark-rdd/convert-spark-rdd-to-dataframe-dataset/)

Spark SQL Tutorial

Spark SQL – Create DataFrame (https://sparkbyexamples.com/spark/different-ways-to-create-a-spark-dataframe/)

Spark SQL – Select Columns (https://sparkbyexamples.com/spark/spark-select-columns-from-dataframe/)

Spark SQL – Add and Update Column (withColumn) (https://sparkbyexamples.com/spark/spark-dataframe-withcolumn/)

Spark SQL – Rename Nested Column (https://sparkbyexamples.com/spark/rename-a-column-on-spark-dataframes/)

Spark SQL – Drop column (https://sparkbyexamples.com/spark/spark-drop-column-from-dataframe-dataset/)

Spark SQL – Where | Filter (https://sparkbyexamples.com/spark/spark-dataframe-where-filter/)

Spark SQL – When Otherwise (https://sparkbyexamples.com/spark/spark-case-when-otherwise-example/)

Spark SQL – Collect data to Driver (https://sparkbyexamples.com/spark/spark-dataframe-collect/)

Spark SQL – Distinct (https://sparkbyexamples.com/spark/spark-remove-duplicate-rows/)

Spark SQL- Pivot Table DataFrame (https://sparkbyexamples.com/spark/how-to-pivot-table-and-unpivot-a-spark-dataframe/)

Spark SQL – Data Types (https://sparkbyexamples.com/spark/spark-sql-dataframe-data-types/)

Cost efficient – Spark computations are very expensive hence reusing the computations are used to save cost.

Time efficient – Reusing the repeated computations saves lots of time.

Execution time – Saves execution time of the job and we can perform more jobs on the same cluster.

Spark Cache Syntax and Example

Spark DataFrame or Dataset **cache()** method by default saves it to storage level `**MEMORY_AND_DISK**` because recomputing the in-memory columnar representation of the underlying table is expensive. Note that this is different from the default cache level of `RDD.cache()` which is ‘**MEMORY_ONLY**‘.

Syntax

```
cache() : Dataset.this.type
```

Spark SQL – StructType | StructField (https://sparkbyexamples.com/spark/spark-sql-structtype-on-dataframe/)

Spark SQL – Schema (https://sparkbyexamples.com/spark/spark-schema-explained-with-examples/)

Spark SQL – Groupby (https://sparkbyexamples.com/spark/using-groupby-on-dataframe/)

Spark SQL – Sort DataFrame (https://sparkbyexamples.com/spark/spark-how-to-sort-dataframe-column-explained/)

Spark SQL – Join Types (https://sparkbyexamples.com/spark/spark-sql-dataframe-join/)

Spark SQL – Union and UnionAll (https://sparkbyexamples.com/spark/spark-dataframe-union-and-union-all/)

Spark SQL – map() vs mapPartitions() (https://sparkbyexamples.com/spark/spark-map-vs-mappartitions-transformation/)

Spark SQL – foreach() vs foreachPartition() (https://sparkbyexamples.com/spark/spark-foreachpartition-vs-foreach-explained/)

Spark SQL – map() vs flatMap() (https://sparkbyexamples.com/spark/spark-map-vs-flatmap-with-examples/)

Spark SQL – Persist and Cache (https://sparkbyexamples.com/spark/spark-dataframe-cache-and-persist-explained/)

Spark SQL – UDF (User Defined Functions) (https://sparkbyexamples.com/spark/spark-sql-udf/)

Spark SQL – Array (ArrayType) Column (https://sparkbyexamples.com/spark/spark-array-arraytype-dataframe-column/)

Spark `cache()` method in Dataset class internally calls `persist()` method which in turn uses `sparkSession.sharedState.cacheManager.cacheQuery` to cache the result set of DataFrame or Dataset. Let's look at an example.

Example

```
val spark:SparkSession = Spar
  .master("local[1]")
  .appName("SparkByExamples.c
  .getOrCreate()

//read csv with options
val df = spark.read.options(M
  .csv("src/main/resources/zi

val df2 = df.where(col("State
df2.show(false)

println(df2.count())

val df3 = df2.where(col("Zipc

println(df2.count())
```

DataFrame Persist Syntax and Example

Spark persist() method is used to store the DataFrame or Dataset to one of the storage levels **`MEMORY_ONLY`**, **`MEMORY_AND_DISK`**, **`MEMORY_ONLY_SER`**, **`MEMORY_AND_DISK_SER`**, **`DISK_ONLY`**, **`MEMORY_ONLY_2`**, **`MEMORY_AND_DISK_2`** and more.

Caching or persisting of Spark DataFrame or Dataset is a lazy operation, meaning a DataFrame will not be cached until you trigger an action.

Syntax

[Spark SQL – Map (MapType) column (https://sparkbyexamples.com/spark/spark-dataframe-map-maptype-column/)](https://sparkbyexamples.com/spark/spark-dataframe-map-maptype-column/)

[Spark SQL – Flatten Nested Struct Column (https://sparkbyexamples.com/spark/spark-flatten-nested-struct-column/)](https://sparkbyexamples.com/spark/spark-flatten-nested-struct-column/)

[Spark SQL – Flatten Nested Array Column (https://sparkbyexamples.com/spark/spark-flatten-nested-array-column-to-single-column/)](https://sparkbyexamples.com/spark/spark-flatten-nested-array-column-to-single-column/)

[Spark SQL – Explode Array & Map Columns (https://sparkbyexamples.com/spark/explode-spark-array-and-map-dataframe-column/)](https://sparkbyexamples.com/spark/explode-spark-array-and-map-dataframe-column/)

[Spark SQL – Sampling (https://sparkbyexamples.com/spark/spark-sampling-with-examples/)](https://sparkbyexamples.com/spark/spark-sampling-with-examples/)

[Spark SQL – Partitioning (https://sparkbyexamples.com/spark/spark-partitioning-understanding/)](https://sparkbyexamples.com/spark/spark-partitioning-understanding/)

Spark SQL Functions

[Spark SQL String Functions (https://sparkbyexamples.com/spark/usage-of-spark-sql-string-functions/)](https://sparkbyexamples.com/spark/usage-of-spark-sql-string-functions/)

[Spark SQL Date and Timestamp Functions (https://sparkbyexamples.com/spark/spark-sql-date-and-time-functions/)](https://sparkbyexamples.com/spark/spark-sql-date-and-time-functions/)

[Spark SQL Array Functions (https://sparkbyexamples.com/spark/spark-sql-array-functions/)](https://sparkbyexamples.com/spark/spark-sql-array-functions/)

[Spark SQL Map Functions (https://sparkbyexamples.com/spark/spark-sql-map-functions/)](https://sparkbyexamples.com/spark/spark-sql-map-functions/)

[Spark SQL Sort Functions (https://sparkbyexamples.com/spark/spark-sql-sort-functions/)](https://sparkbyexamples.com/spark/spark-sql-sort-functions/)

[Spark SQL Aggregate Functions (https://sparkbyexamples.com/](https://sparkbyexamples.com/)

```
1) persist() : Dataset.this.typ
2) persist(newLevel : org.apach
```

Spark persist has two signature first signature doesn't take any argument which by default saves it to **`MEMORY_AND_DISK`** storage level and the second signature which takes `StorageLevel` as an argument to store it to different storage levels.

Example

```
  val dfPersist = df.persist()
  dfPersist.show(false)
```

Using the second signature you can save DataFrame/Dataset to any storage levels.

```
  val dfPersist = df.persist(St
  dfPersist.show(false)
```

This stores DataFrame/Dataset into Memory.

Note that Dataset `cache()` is an alias for `persist(StorageLevel.MEMORY_AND_DISK)`

Unpersist syntax and Example

Spark automatically monitors every persist() and cache() calls you make and it checks usage on each node and drops persisted data if not used or by using least-recently-used (LRU) algorithm. You can also manually remove using `unpersist()` method. unpersist() marks the Dataset as non-persistent, and remove all blocks for it from memory and disk.

Syntax

spark/spark-sql-aggregate-functions/)

Spark SQL Window Functions (https://sparkbyexamples.com/spark/spark-sql-window-functions/)

Spark SQL JSON Functions (https://sparkbyexamples.com/spark/spark-most-used-json-functions-with-examples/)

Spark Data Source API

Spark – Read & Write CSV file (https://sparkbyexamples.com/spark/spark-read-csv-file-into-dataframe/)

Spark – Read and Write JSON file (https://sparkbyexamples.com/spark/spark-read-and-write-json-file/)

Spark – Read & Write Parquet file (https://sparkbyexamples.com/spark/spark-read-write-dataframe-parquet-example/)

Spark – Read & Write XML file (https://sparkbyexamples.com/spark/spark-read-write-xml/)

Spark – Read & Write Avro files (https://sparkbyexamples.com/spark/read-write-avro-file-spark-dataframe/)

Spark – Read & Write Avro files (Spark version 2.3.x or earlier) (https://sparkbyexamples.com/spark/using-avro-data-files-from-spark-sql-2-3-x/)

Spark – Read & Write HBase using "hbase-spark" Connector (https://sparkbyexamples.com/spark/spark-read-write-using-hbase-spark-connector/)

Spark – Read & Write from HBase using Hortonworks (https://sparkbyexamples.com/spark/create-spark-dataframe-from-hbase-using-hortonworks/)

Spark – Read & Write ORC file (https://sparkbyexamples.com/spark/spark-read-orc-file-into-dataframe/)

```
unpersist() : Dataset.this.type
unpersist(blocking : scala.Bool
```

Example

```
  val dfPersist = dfPersist.unp
```

unpersist(Boolean) with boolean as argument blocks until all blocks are deleted.

Spark Persist storage levels

All different storage level Spark supports are available at `org.apache.spark.storage.StorageLevel` class. The storage level specifies how and where to persist or cache a Spark DataFrame and Dataset.

`MEMORY_ONLY` – This is the default behavior of the RDD `cache()` method and stores the RDD or DataFrame as deserialized objects to JVM memory. When there is no enough memory available it will not save DataFrame of some partitions and these will be re-computed as and when required. This takes more memory. but unlike RDD, this would be slower than **MEMORY_AND_DISK** level as it recomputes the unsaved partitions and recomputing the in-memory columnar representation of the underlying table is expensive

`MEMORY_ONLY_SER` – This is the same as `MEMORY_ONLY` but the difference being it stores RDD as serialized objects to JVM memory. It takes lesser memory (space-efficient) then MEMORY_ONLY as it saves objects as serialized and takes an additional few more CPU cycles in order to deserialize.

[Spark – Read Binary File (https://sparkbyexamples.com/spark/spark-read-binary-file-into-dataframe/)](https://sparkbyexamples.com/spark/spark-read-binary-file-into-dataframe/)

Spark Streaming & Kafka

[Spark Streaming – OutputModes (https://sparkbyexamples.com/spark/spark-streaming-outputmode/)](https://sparkbyexamples.com/spark/spark-streaming-outputmode/)

[Spark Streaming – Reading Files From Directory (https://sparkbyexamples.com/spark/spark-streaming-read-json-files-from-directory/)](https://sparkbyexamples.com/spark/spark-streaming-read-json-files-from-directory/)

[Spark Streaming – Reading Data From TCP Socket (https://sparkbyexamples.com/spark/spark-streaming-from-tcp-socket/)](https://sparkbyexamples.com/spark/spark-streaming-from-tcp-socket/)

[Spark Streaming – Processing Kafka Messages in JSON Format (https://sparkbyexamples.com/spark/spark-streaming-with-kafka/)](https://sparkbyexamples.com/spark/spark-streaming-with-kafka/)

[Spark Streaming – Processing Kafka messages in AVRO Format (https://sparkbyexamples.com/spark/spark-streaming-consume-and-produce-kafka-messages-in-avro-format/)](https://sparkbyexamples.com/spark/spark-streaming-consume-and-produce-kafka-messages-in-avro-format/)

[Spark SQL Batch – Consume & Produce Kafka Message (https://sparkbyexamples.com/spark/spark-batch-processing-produce-consume-kafka-topic/)](https://sparkbyexamples.com/spark/spark-batch-processing-produce-consume-kafka-topic/)

`MEMORY_ONLY_2` – Same as `MEMORY_ONLY` storage level but replicate each partition to two cluster nodes.

`MEMORY_ONLY_SER_2` – Same as `MEMORY_ONLY_SER` storage level but replicate each partition to two cluster nodes.

`MEMORY_AND_DISK` – This is the default behavior of the DataFrame or Dataset. In this Storage Level, The DataFrame will be stored in JVM memory as a deserialized objects. When required storage is greater than available memory, it stores some of the excess partitions into disk and reads the data from disk when it required. It is slower as there is I/O involved.

`MEMORY_AND_DISK_SER` – This is same as `MEMORY_AND_DISK` storage level difference being it serializes the DataFrame objects in memory and on disk when space not available.

`MEMORY_AND_DISK_2` – Same as `MEMORY_AND_DISK` storage level but replicate each partition to two cluster nodes.

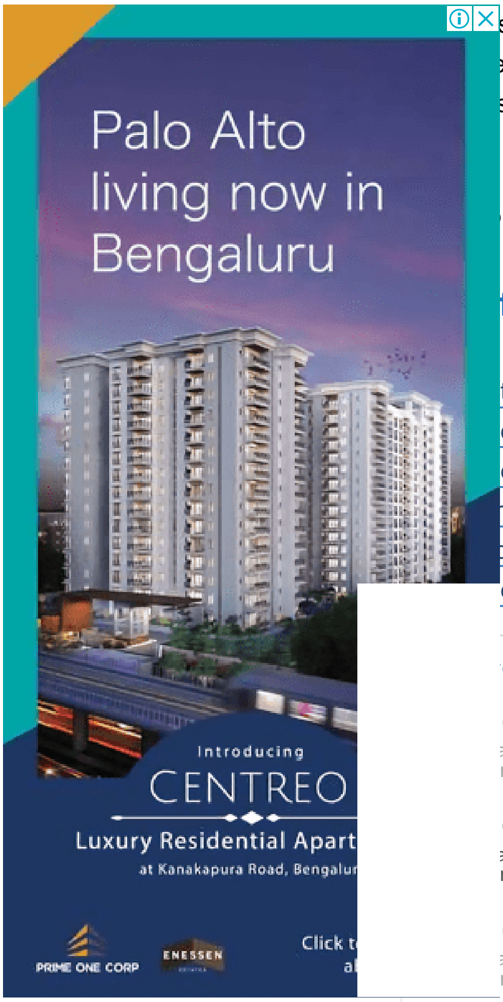
`MEMORY_AND_DISK_SER_2` – Same as `MEMORY_AND_DISK_SER` storage level but replicate each partition to two cluster nodes.

`DISK_ONLY` – In this storage level, DataFrame is stored only on disk and the CPU computation time is high as I/O involved.

`K_ONLY_2` – Same
`)ISK_ONLY` storage level but
icate each partition to two cluster
es.

nclusion

is article, you have learned Spark `he()` and `persist()` methods are used as optimization techniques to save interim computation results of DataFrame or Dataset and reuse them



sequently and learned what is the
erence between Spark Cache and
sist and finally saw their syntaxes
usages with Scala examples.

py Learning !!

ference

ttps://spark.apache.org/docs/latest/r
d-programming-guide.html#rdd-
ersistence
https://spark.apache.org/docs/latest/
dd-programming-guide.html#rdd-
ersistence)

e this:

(https://sparkbyexamples.com/spark/spark-
aframe-cache-and-persist-explained/?
re=facebook&nb=1)

(https://sparkbyexamples.com/spark/spark-
aframe-cache-and-persist-explained/?
re=reddit&nb=1)

(https://sparkbyexamples.com/spark/spark-
aframe-cache-and-persist-explained/?
re=pinterest&nb=1)

t (https://sparkbyexamples.com/spark/spark-dataframe-cache-and-persist-explained/?share=tumblr&nb=1)

(https://sparkbyexamples.com/spark/spark-dataframe-cache-and-persist-explained/?share=pocket&nb=1)

in (https://sparkbyexamples.com/spark/spark-dataframe-cache-and-persist-explained/?share=linkedin&nb=1)

(https://sparkbyexamples.com/spark/spark-dataframe-cache-and-persist-explained/?share=twitter&nb=1)

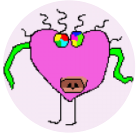
TAGS: CACHE (HTTPS://SPARKBYEXAMPLES.COM/TAG/CACHE/), OPTIMIZATION (HTTPS://SPARKBYEXAMPLES.COM/TAG/OPTIMIZATION/), PERSIST (HTTPS://SPARKBYEXAMPLES.COM/TAG/PERSIST/)

(https://sparkbyexamples.com/author/admin/)

NNK (Https://Sparkbyexamples.Com/Author/Admin/)

SparkByExamples.com is a Big Data and Spark examples community page, all examples are simple and easy to understand and well tested in our development environment Read more .. (https://sparkbyexamples.com/about-sparkbyexamples/)

› THIS POST HAS 4 COMMENTS

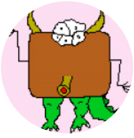


Arun 24 MAR 2021 REPLY

Please update color of code snippets to dark shades current ones are not clearly visible.

NNK 24 MAR 2021 REPLY

Hi Arun, I don't understand, right now I have a black background and code in white/gray. Could you please let me know what is not clearly visible?



Anonymous

15 JUN 2020 REPLY

please remove the autoscroll. Its not helping.

NNK 15 JUN 2020 REPLY

Could you please let me know what browser you are using? I am not seeing auto scroll on Chrome?

Leave a Reply

Categories

Apache Hadoop (https://sparkbyexamples.com/category/hadoop/)

Apache Spark (https://sparkbyexamples.com/category/spark/)

Apache Spark Streaming (https://sparkbyexamples.com/category/spark/apache-spark-streaming/)

Apache Kafka (https://sparkbyexamples.com/category/kafka/)

Apache HBase (https://sparkbyexamples.com/category/hbase/)

Apache Cassandra (https://sparkbyexamples.com/category/cassandra/)

Snowflake Database (https://sparkbyexamples.com/category/snowflake/)

H2O Sparkling Water (https://sparkbyexamples.com/category/h2o-sparkling-water/)

PySpark (https://sparkbyexamples.com/category/pyspark/)

Recent Posts

Spark regexp_replace() – Replace String Value (https://sparkbyexamples.com/spark/spark-regexp_replace-replace-string-value/)

How to Run a PySpark Script from Python? (https://sparkbyexamples.com/pyspark/run-pyspark-script-from-python-subprocess/)

Spark SQL like() Using Wildcard Example (https://sparkbyexamples.com/spark/spark-sql-like-using-wildcard-example/)

Spark isin() & IS NOT IN Operator Example (https://sparkbyexamples.com/spark/spark-isin-is-not-in-operator-example/)

Spark – Get Size/Length of Array & Map Column (https://sparkbyexamples.com/spark/spark-get-size-length-of-array-map-column/)

Spark Using Length/Size Of a DataFrame Column (https://sparkbyexamples.com/spark/spark-using-length-size-of-a-dataframe-column/)

Spark rlike() Working with Regex Matching Examples (https://sparkbyexamples.com/spark/spark-rlike-regex-matching-examples/)

Spark Check String Column Has Numeric Values (https://sparkbyexamples.com/spark/sp

About SparkByExamples.Com

SparkByExamples.com is a Big Data and Spark examples community page, all examples are simple and easy to understand, and well tested in our development environment Read more .. (https://sparkbyexamples.com/about-sparkbyexamples/)

Follow Us

(https://twitter.com/sparkbyexamples) (https://www.facebook.com/sparkbyexamples/) (https://www.linkedin.com/in/n-nk-b860a8193/) (https://github.com/spark-examples/)

ark-check-string-column-has-numeric-values/)

Spark Check Column Data Type is Integer or String (https://sparkbyexamples.com/spark/spark-check-column-data-type-is-integer-or-string/)