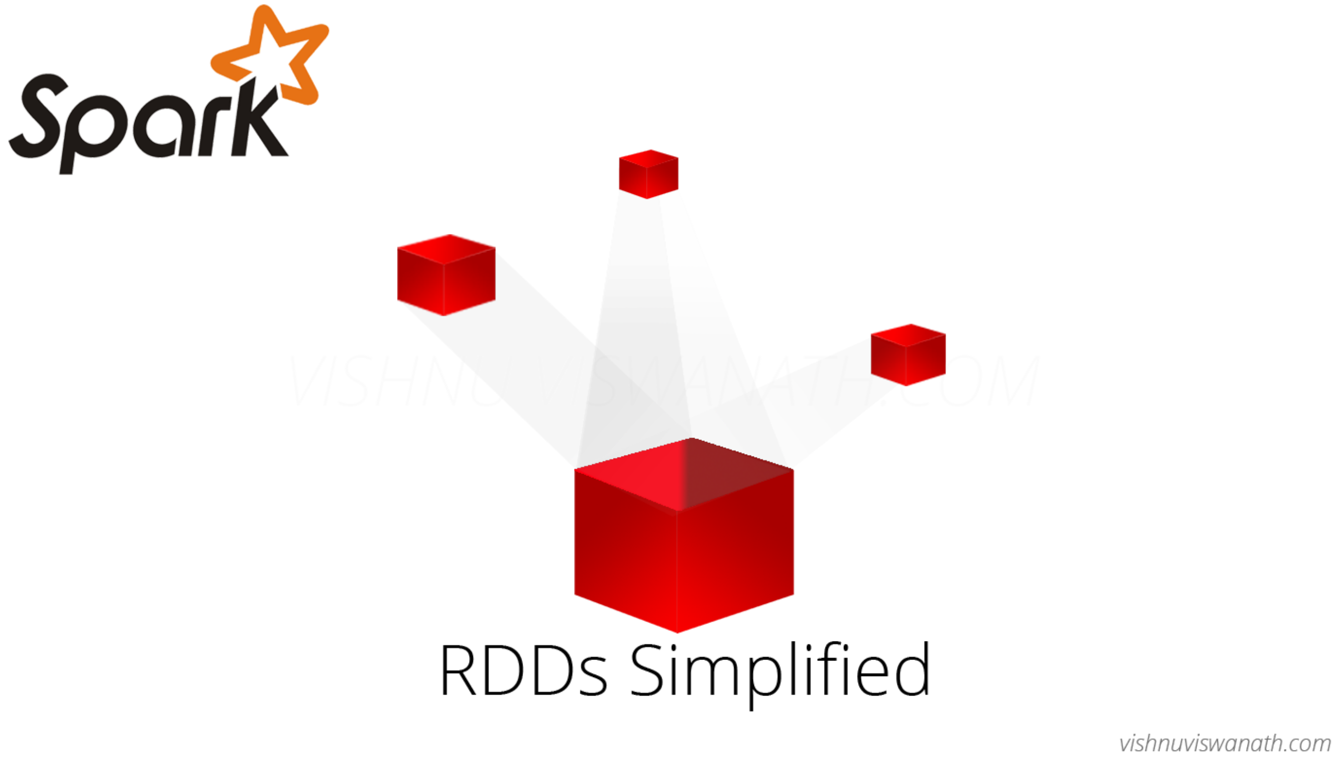
Search search

blog about tags

Spark RDDs Simplified

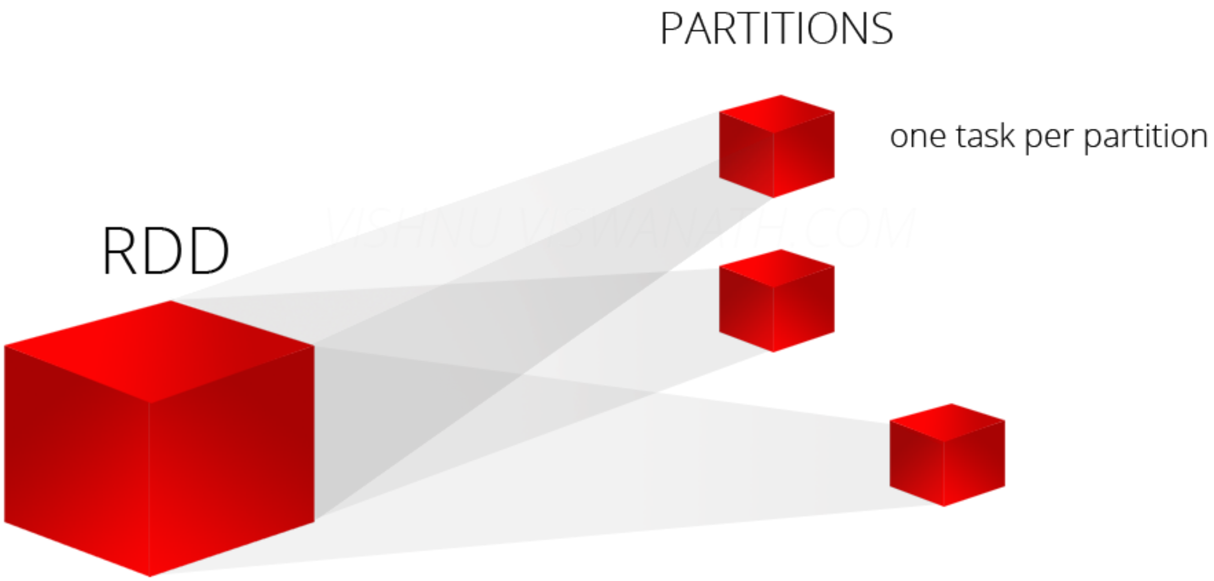
February 4, 2016. Estimated read time: 3 minutes



Spark RDDs are very simple at the same time very important concept in Apache Spark. Most of you might be knowing the full form of RDD, it is **Resilient Distributed Datasets**. ***Resilient*** because RDDs are immutable*(can't be modified once created)* and fault tolerant, ***Distributed*** because it is distributed across cluster and ***Dataset*** because it holds data.

So why RDD? Apache Spark lets you treat your input files almost like any other variable, which you cannot do in Hadoop MapReduce. RDDs are automatically distributed across the network by means of Partitions.

Partitions



RDDs are divided into smaller chunks called Partitions, and when you execute some action, a parallelism. Spark automatically decides the number of partitions that an RDD has to be divided into but you can also specify the number of partitions when creating an RDD. These partitions of an RDD is distributed across all the nodes in the network.

Creating an RDD

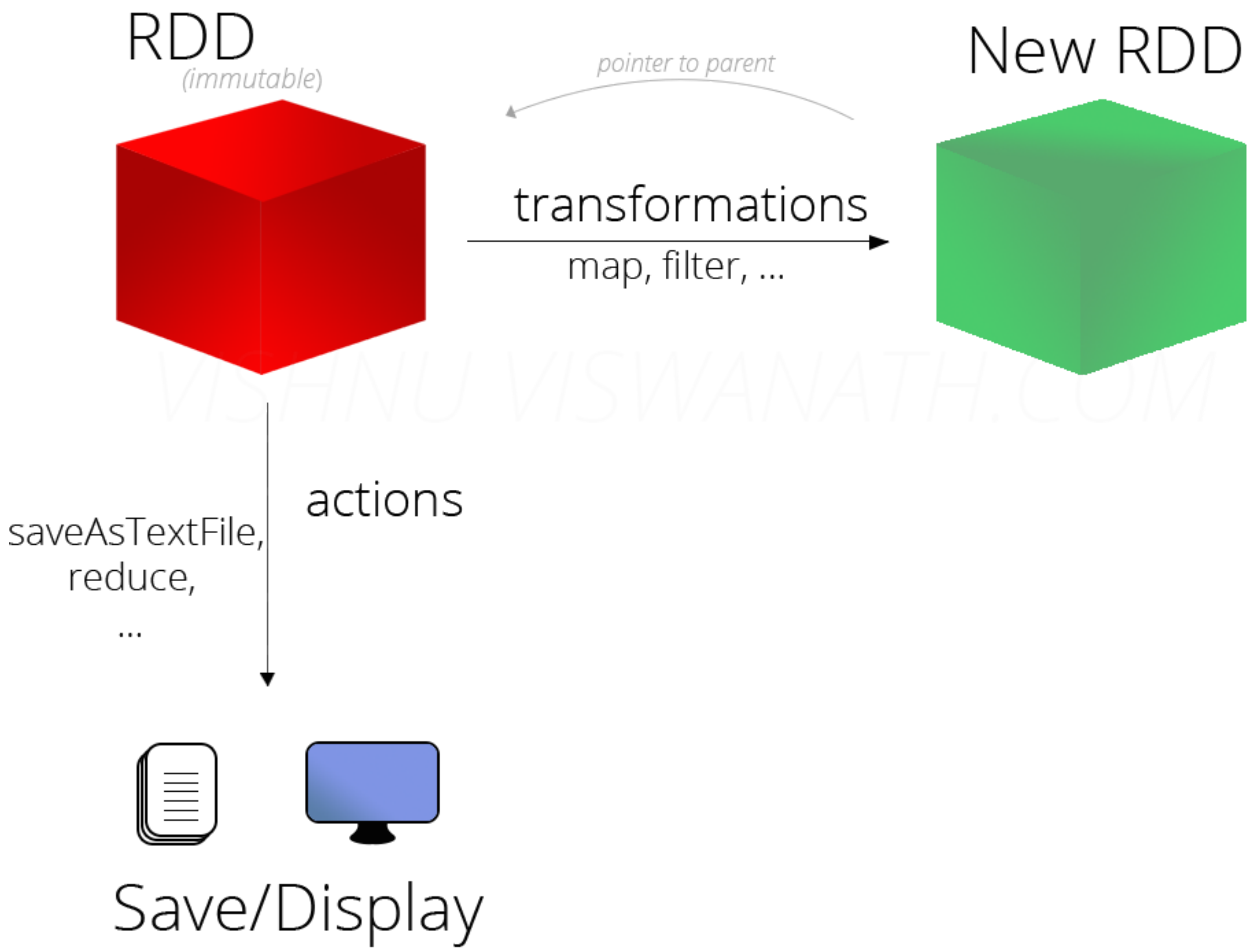
Creating an RDD is easy, it can be created either from an external file or by parallelizing collections in your driver. For example,

```
val rdd = sc.textFile("/some_file",3)
val lines = sc.parallelize(List("this is","an example"))
```

The first line creates an RDD from an external file, and the second line creates an RDD from a list of Strings. *Note that the argument '3' in the method call sc.textFile() specifies the number of partitions that has to be created. If you don't want to specify the number of partitions, then you can simply call sc.textFile("some_file").*

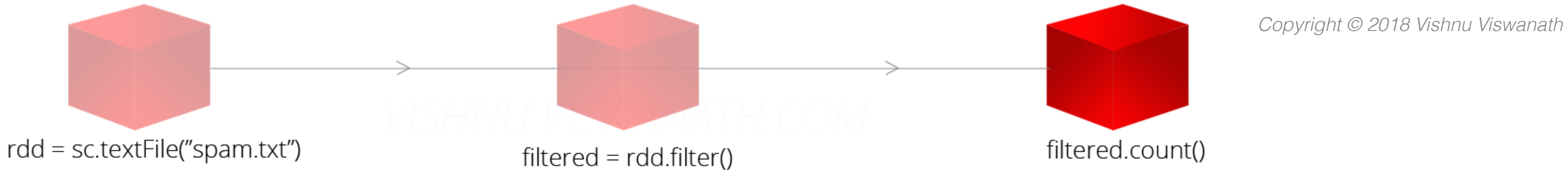
Actions/Transformations

There are two types of operations that you can perform on an RDD- *Transformations and Actions*. **Transformation** applies some function on a RDD and creates a new RDD, it does not modify the RDD that you apply the function on.*(Remember that RDDs are resilient/immutable).* Also, the new RDD keeps a pointer to it's parent RDD.



Copyright © 2018 Vishnu Viswanath

When you call a transformation, Spark does not execute it immediately, instead it creates a **lineage**. A lineage keeps track of what all transformations has to be applied on that RDD, including from where it has to read the data. For example, consider the below example


rdd = sc.textFile("spam.txt")
filtered = rdd.filter()
filtered.count()
Copyright © 2018 Vishnu Viswanath
VISHNU VISWANATH.COM

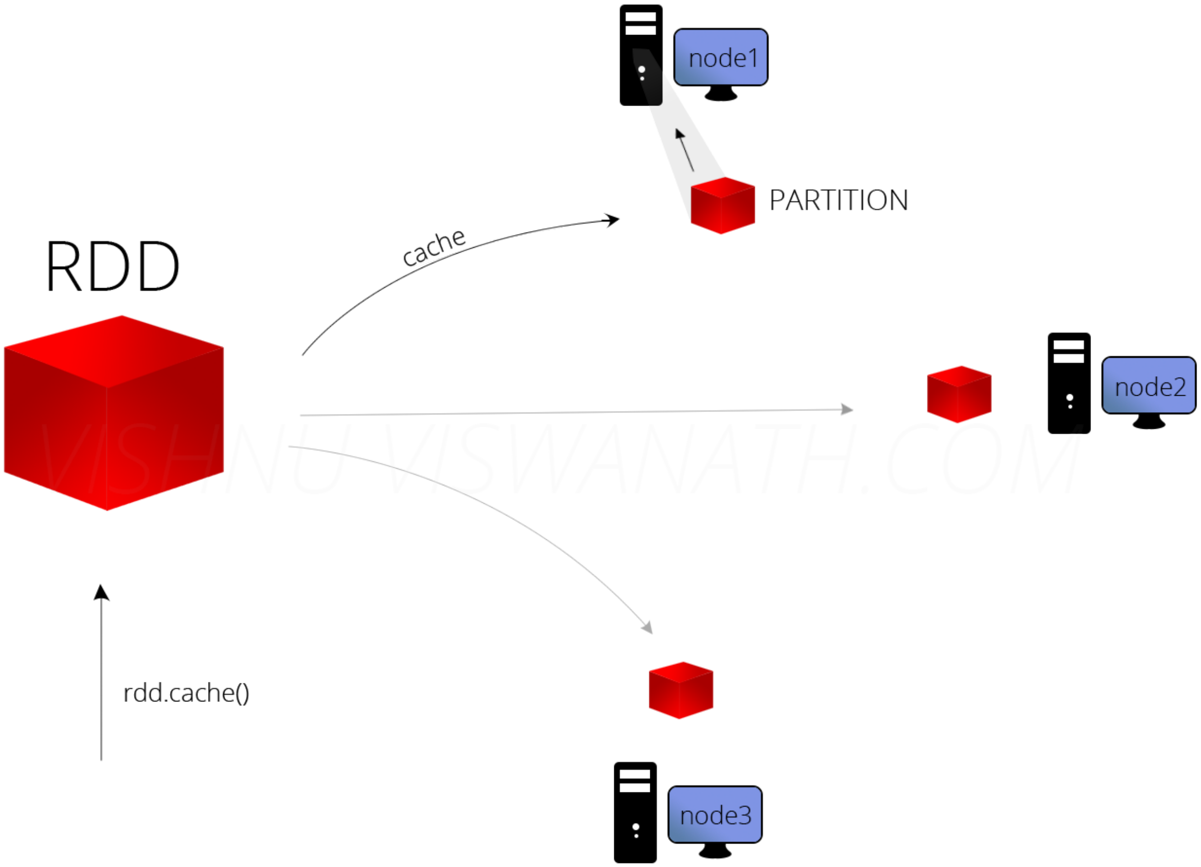
```
val rdd = sc.textFile("spam.txt")
val filtered = rdd.filter(line => line.contains("money"))
filtered.count()
```

sc.textFile() and rdd.filter() do not get executed immediately, it will only get executed once you call an *Action* on the RDD - here filtered.count(). An **Action** is used to either save result to some location or to display it. You can also print the RDD lineage information by using the command `filtered.toDebugString` *(filtered is the RDD here)*.

> *RDDs can also be thought of as a set of instructions that has to be executed, first instruction being the load instruction.*

Caching

You can cache an RDD in memory by calling `rdd.cache()` . When you cache an RDD, it's Partitions are loaded into memory of the nodes that hold it.



Caching can improve the performance of your application to a great extent. In the previous section you saw that when an action is performed on a RDD, it executes it's entire lineage. Now imagine you are going to perform an action multiple times on the same RDD which has a long

RDD in the memory thereby eliminating the need to recompute it every time. You can think of caching as if it is breaking the lineage, but it does remember the lineage so that it can be recomputed in case of a node failure.

This concludes the basics of RDD. If you would like to read more, Part2 talks about Persistence, Broadcast variables and Accumulators. Thanks for reading!

Continue reading

Tags: ApacheSpark Scala BigData Hadoop

25 Comments **vishnuviswanath.com** 1 **Login**

♡ **Recommend** 4 Tweet Share **Sort by Best**

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

subwaymatch • 2 years ago

Very nice post and amazing illustrations, thanks!

1 ^ | ˅ • Reply • Share ›

Vishnu Viswanath Mod ➦ subwaymatch • 2 years ago

Thank you!

^ | ˅ • Reply • Share ›

Email address Subscribe

I recently worked on sparklyr .. where I had to do transformations on Data multiple times. Since it is spark, it has lazy evaluation. As discussed in the post, as the lineage increases, the work increases, so, we use cache to make it faster. My question is, is it similar to creating a checkpoint?
For example, If my lineage has 10 transformations? I would create 2 checkpoints and use th
later again whenever necessary. Is this possible with cache? Can I have multiple caches in the same lineage and access a particular cache to do another analysis/transformations?

1 ^ | ˅ • Reply • Share ›

Vishnu Viswanath Mod ➦ Raj • 2 years ago

Thank you and good question. Caching and checkpointing have some similarities but are not the same. Caching tells spark to store the computed value in memory/external storage. Caching does not break the lineage, i.e., the RDD still have the lineage info.
RDD.checkpoint() stores the computed value to the storage and it breaks the lineage.
While applying performing an action, spark first checks if the RDD is cached, if not it checks if the RDD is checkpointed, if not it computes the values from the source by applying all the transformations before that point.
Yes, you can have multiple caches in the same linage and use it for doing another transformation.

^ | ˅ • Reply • Share ›

Lyubcho Dimov • 2 years ago

Amazingly well explained!! Kudos! :)

1 ^ | ˅ • Reply • Share ›

Vishnu Viswanath Mod ➦ Lyubcho Dimov • 2 years ago

Thank you :)

^ | ˅ • Reply • Share ›

Rajesh • 3 years ago

Remember that RDD.cache() is also lazy. Let's see following code -

Email address Subscribe

L3 filteredRDD.cache() // This is lazy as well
L4 filteredRDD.count()
L5 some code
L6 some code
L7 filteredRDD.count()

Note:
One might think line L3 has loaded the file and cached filtered data in memory, how does it happen when there was no action performed on this RDD. Don't assume that line L3 (filteredRDD.cache()) has read the file, loaded and cached. This line is also lazy and it only prepared instructions on what to do when an action is performed on the filteredRDD.

Line L4 is an action (1st time the filteredRDD is executed), when this is executed - filteredRDD is loaded, cached, and counted.

When line L7 is executed (2nd time the filteredRDD is executed) - the operation will take the data from the cache and count the lines.

1 ^ | ˅ • Reply • Share ›

Vishnu Viswanath Mod ➦ Rajesh • 3 years ago

good point Rajesh.

^ | ˅ • Reply • Share ›

Prasiddh Nandola • 4 years ago

I have one doubt: we have only rdd.persist() and rdd.cache() methods. But, i do not want to persist entire rdd, instead i want to persist only some partitions of a particular rdd. how can i do that?

1 ^ | ˅ • Reply • Share ›

Tapan Behera ➦ Prasiddh Nandola • 2 years ago

create one more RDD based on filteration and cached it. So it will cahced the new RDD not the old one.

1 ^ | ˅ • Reply • Share ›

can split your RDD into multiple RDDs based on some condition and choose to cache only the RDD you want.

Reply • Share ›

Prasiddh Nandola → Vishnu Viswanath • 4 years ago

I want only some part of the
rdd to persist because, Each partition has either high/low incremental maintenance overhead. e.g. partition 1 has high overhead since the updates (resulting in updates on partition 1)on the source side are heavy, while partition 2 has low overhead. Thus, it makes sense to only materialize partition2 in memory not partition 1.

Reply • Share ›

Prasiddh Nandola • 4 years ago

nice one

1 • Reply • Share ›

Vishnu Viswanath Mod → Prasiddh Nandola • 4 years ago

Thank you

Reply • Share ›

Matt Gardner • 4 years ago

Great blog post to explain the basic concepts - I shared this with my intro Data Science class I am teaching. Thank you.

1 • Reply • Share ›

Vishnu Viswanath Mod → Matt Gardner • 4 years ago

Glad you liked it, and hopefully it will help the students that you are teaching. Thanks!

Reply • Share ›

likitha prakash • 10 months ago

Clear and simple explanation!!! Saved lot of my time :) Appreciate it!!!

amit shah • 3 years ago

Nice post. I like the simplicity with which the concept is explained. I have a question about RDD's - At what instance in time would the rdd be in-memory? From the above explanation it seems that the RDD would be discarded from the RAM after the lineage execution is completed. To keep it in-memory, we need to call rdd.persist() or rdd.cache(). Is my understanding right?

^ | ˅ • Reply • Share ›

Vishnu Viswanath Mod ➦ amit shah • 3 years ago

Hi Amit, Thank you. Glad you liked the post.
You are right, an RDD is kept in memory if you call cache() on it. That way the processing that has be done on that RDD to reach that particular state need not be recomputed every time. This is usually done when building ML models since during the learning stage of ML, you usually need to perform repeated operation on the same RDD.
But note that your cluster should have enough memory to hold the RDD in memory, otherwise you will have to do some other form of persistence (may be memory + disk).
If you don't do any caching, during the processing of the RDD, each partition of the RDD will be brought into memory and will be processed.

^ | ˅ • Reply • Share ›

amit shah ➦ Vishnu Viswanath • 3 years ago

Got it. Thanks for the detailed reply. While reading more about RDD's I have some new questions which could be worth discussing

1. What is the underlying physical representation of a Spark RDD?
2. What is a pairRDD? While reading about RDD's I understand that spark provides different types of RDD's meant for specific needs. I am looking for more details.
3. How can we execute SQL queries on a RDD? I understand that the dataframe or the dataset api is generally used for this but I want to know if it's possible to execute SQL queries on RDD's. I read a bit about SchemaRDD's but wasn't able to grasp it completely.

Vishnu Viswanath Mod → amit shah • 3 years ago

1. As far as I know, Spark by default uses Java Serialization, you can also use Kryo serializer. So the physical representation would depend on the type of Serializer used. (https://spark.apache.org/do...

2. A pairRDD is an RDD with key and value. This is useful in cases where you want to perform operations based on the key. E.g., aggregateByKey, countByKey, reduceByKey etc. (http://spark.apache.org/doc...

3. You cannot directly run SQL query on RDD. You need to convert that RDD into schemaRDD. A schemaRDD is an RDD of Rows with Schema information. If your RDD is an RDD of case classes, then you can register that RDD as a tempTable and run SQL queries on it. Note: The api for temp table registration has change from spark 2.0

Hope this helps.

^ | ⌄ • Reply • Share ›

Wanderer • 4 years ago • edited

I have series of question. When I execute "val rdd = sc.textFile("spam.txt")" a new rdd is created and is partitioned automatically by spark. Now were would the partitioned data is stored in the cluster? Does they store in worker node memory or worker node disk? If it is stored in worker node memory, what is the need of cache? Like Hadoop does spark has replication of data in its cluster?. Also if a node fails in spark, how does the computation is handled for the data in that particular failed node?

^ | ⌄ • Reply • Share ›

Vishnu Viswanath Mod → Wanderer • 4 years ago

Regarding the failure case, if a node fails then Spark knows which part of the transformation/action failed and it can ask some other node to execute the failed task. All Spark has to know is where to read data from and what to do with it, which is available in the execution plan/graph.

An RDD is essentially a set of transformations and and action, so when you create an RDD and apply some transformation on it, it is not loaded/executed until you call an action on it. When an action is called on an RDD, spark loads the data and keeps it in memory during the execution and discards it. So when you call some action on the same RDD, all it's lineage has to be executed again. The point of cache is to avoid this, i.e., if you call cache on an RDD and you execute an action on it, then the result of the transformation upto the point where you called cache is stored in memory. So when you call an action on the cached RDD again, it is not loaded again and also the transformations are not applied again.

^ | ⌄ • Reply • Share ›

Wanderer ➦ Vishnu Viswanath • 4 years ago

Can you create a new thread to know deep understanding of how worker nodes split its cores, like how many cores alloted for spark usage. Understanding about executor, threads and how many partitions for each cores.

^ | ⌄ • Reply • Share ›

✉ **Subscribe** **Add Disqus to your siteAdd DisqusAdd** 🔒 **Disqus' Privacy PolicyPrivacy PolicyPrivacy**

Email address Subscribe