Resilient Distributed Datasets

RDDs have three important characteristics:

(i) Partitions (ii) Read only (iii) Lineage

To understand partitions. Let's go back to first what an RDD is An RDD is nothing but a collection of records that represent data which is kept in memory.

RDDs represent data in-memory

Obviously there is a limit to how much data you can keep in memory on a single machine.

So if you want to deal with a large amount of data. then you would need multiple machines.

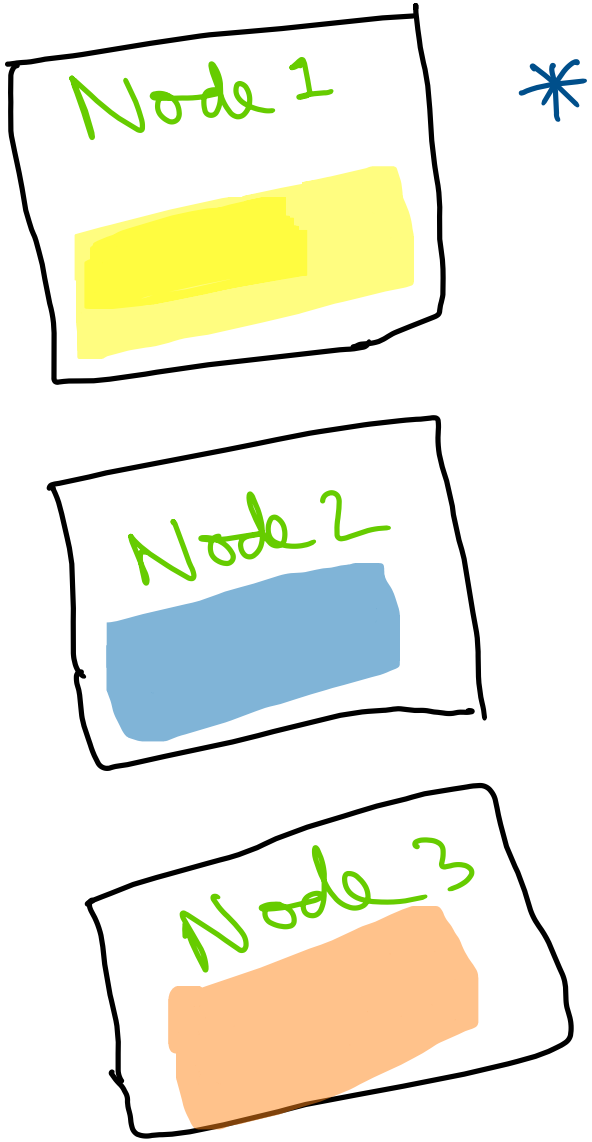
So this data in one RDD is actually divided into partitions that are kept in multiple machines. Each partition is stored on a different node.

Distributed to multiple machines, called nodes

Nodes are nothing but the individual machines in a cluster.

Partitions

Distributed to multiple machines called nodes.



* The advantage of this is that you can keep your entire data in memory, as long as the data size does not exceed the combined memory size of the cluster.

Because the data is kept in multiple machines. It can now be processed in parallel and this is where the speed of processing of spark comes from.

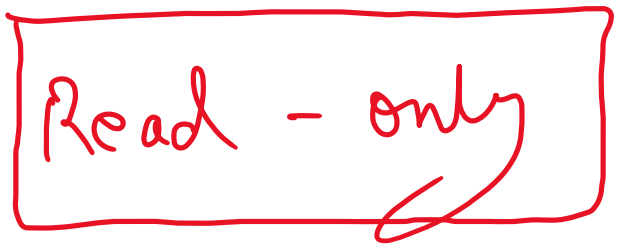
Nodes process data in Parallel.

The implications of this is that specific types of operations that can be performed in parallel are the ones that should be performed on RDDs.

Any operation that require you to keep track of which record in the dataset you are in and loop through the individual

records in the dataset are not really optimized for in this scenario.

The next important characteristic of an RDD is that it is read-only.



RDDs are immutable. They cannot be modified in place. If you want to do something with an RDD, there are only two types of operations that you can use.

Only Two Types of operations

Transformation

- which converts an RDD to another RDD
- Transform into another RDD

Action

- which requests a result or reads data from the RDD
- Request a result

So there are only two operations you can perform. you can not modify an RDD in place.

Transformations are used to transform records in a dataset, whether it is filtering out only certain rows or extracting specific fields or doing something with those fields.

once a dataset is loaded into an RDD. You might want to perform a chain of transformations before you actually look at some values. You can define this complete set of transformations on the dataset.

for instance, you might want to do three things:

1) Load the data

2) Pick only the 3rd Column

3) Sort the values

then after you sort you want to look at the top 10 rows.
Each of these will require a separate RDD operation and
each operation will convert one RDD to another RDD.

However, the actual data processing doesn't happen immediately. It happens only when the user actually requests the results. when you ask for the top 10 rows.

As you are defining each of these steps, intermediate RDDs are created. but those RDDs only contain metadata at this point. The data processing in actuality will happen only when an action is requested.

An action might involves reading a specific set of rows, Computing a count, Computing a Sum Either way an action

means that you want to see some value as a result of the data processing task.

Action

when this action is requested the chain of transformations that you defined earlier load the data, pick the 3rd value, sort the values all of these are performed in actuality at that time.

In meantime Spark is just keeping a record of all the transformations that you requested and when when the action is called upon it will group the

transformations in an efficient way and perform the data processing. utilizing the resources it has in the best manner.

Now if you want to understand how is it possible for Spark to do this? How does spark just keep a record of the list of transformations you require and then perform them only when you ask for the result?

This is done using characteristic called lineage.

Lineage

When an RDD is created. It just holds metadata. The metadata is the transformation that created the RDD and the parent RDD from which that RDD was created.

Therefore, every RDD knows where it came from.

((Lineage can be traced back all the way to the source))

RDD2's lineage is RDD1 and data.csv

when an action is requested on RDD2, first the furthest ancestor is materialized. This means the data is actually loaded into that RDD, so data from data.csv is loaded into RDD1 when an action is requested on RDD2 then RDD2 itself is materialized by processing the data from RDD1. This way an RDD and all its parent RDDs are materialized only when an action is requested on the RDD.

The characteristic of lineage is what allows RDDs to have resilience and Lazy evaluation

Lineage

Resilience → It means that even when there is a fault or if one of the nodes in the cluster crashes. you can reconstruct the RDD from the source data. because you know the complete set of transformations involved in creating that RDD.

In-built fault tolerance of something goes wrong reconstruct from source

Lazy-Evaluation

: Lazy evaluation allows you to perform an operation only when it is actually necessary that is when the user is looking for some result otherwise you just keep the series of transformations in memory as metadata and you do not have to hold or occupy the resources in memory until then.

Materialize only when necessary

Summary :

- Understood the role of Spark in data analysis
- Loading data from a file is a Spark transformation
- Reading data from an RDD is a Spark Action
- RDDs are partitioned, read-only collections which know their lineage
- Spark transformations are lazily evaluated