# Introduction Apache Spark and spark core

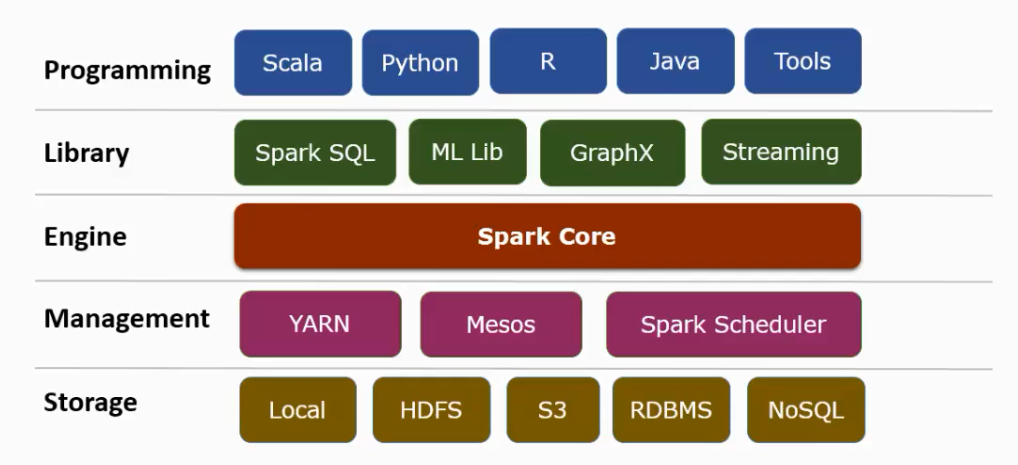
## What is Apache Spark?

Apache Spark is a cluster computing platform, designed for fast computation. Spark does **in memory computation**.

## What are all the languages supported by spark?

Java,Scala,Phython and R. Spark is built using scala

1. **Why apache spark? IMP**
2. It does in memory computation at a very fast rate.
3. Apache spark can do batch processing, real time stream processing, graph processing, Iterative processing.
4. It provides libraries for machine learning.
5. It supports multiple programming languages such as Java,Scala and python
6. **Eco system of Apache Spark? imp**

****

**Spark Core:**

1. **What are the basic functionalities of spark? Imp**

**Note**: remember in such a way that right from creating RDD to completion spark job.

Spark core is execution engine of spark and it does it in memory

1. APIs for building and manipulating RDD
2. Task scheduling
3. Memory management
4. Interacting with storage systems.
5. Fault recovery

**Spark SQL**

Spark SQL is a component on top of Spark Core that introduces a new data abstraction called SchemaRDD, which provides support for structured and semi-structured data.

**MLlib (Machine Learning Library)**

**TODO: Need to learn if required in future**

MLlib is a distributed machine learning framework above Spark because of the distributed memory-based Spark architecture. It is, according to benchmarks, done by the MLlib developers against the Alternating Least Squares (ALS) implementations. Spark MLlib is nine times as fast as the Hadoop disk-based version of Apache Mahout (before Mahout gained a Spark interface).

**GraphX**

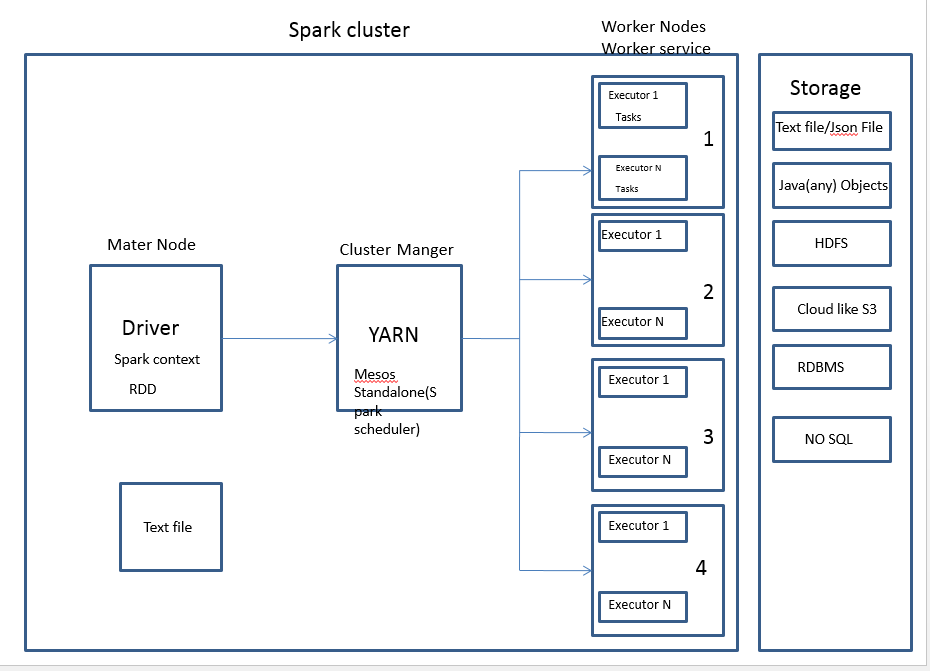
**TODO: Need to learn if required in future**

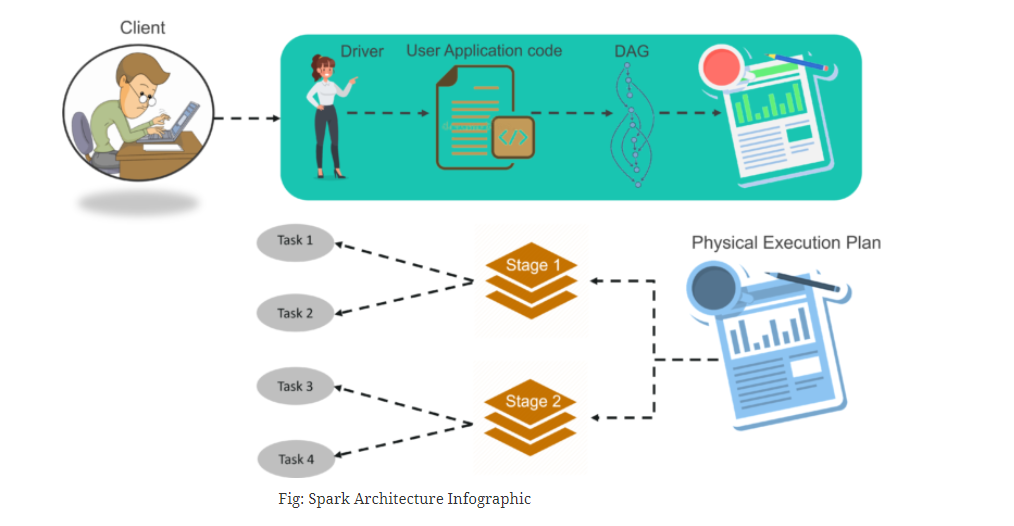
GraphX is a distributed graph-processing framework on top of Spark. It provides an API for expressing graph computation that can model the user-defined graphs by using Pregel abstraction API. It also provides an optimized runtime for this abstraction.

1. **What is the Difference between Spark and Hadoop? IMP**

|  |  |
| --- | --- |
| Spark | Hadoop |
| In memory processing | Disk based processing. |
| Spark is 10X to 100X faster than Hadoop | As a result Hadoop is not as fast as spark |
| Batch processing and real time processing | Batch processing.  Note: In hadoop also we can do real time processing but we have to do lot of administration |
| Fault tolerance: Spark uses RDD which guarantees fault tolerance. | Fault tolerance: In hadoop fault tolerance is achieved by replicating data in multiple copies. In case of machine goes down we use other copy |
| Spark provides most versatile API. The spark APIs can be used with multiple data sources and languages | Hadoop does not provide that much versatile APIs |

1. **Explain Spark Architecture? imp**





[**https://www.edureka.co/blog/spark-architecture/#Overview**](https://www.edureka.co/blog/spark-architecture/#Overview)

## Can we use spark independent of Hadoop?

Yes. We can run spark without Hadoop in standalone mode. But some of its functionality relies on Hadoop for example handling parquet file.

For example Spark can run on mesos and S3

Also spark added native support to run on kubernetes

1. **What happens when a task fails, or executor goes down? Imp**

Spark is fault tolerant. So failed tasks will be rescheduling

DAG scheduler and cluster manager will notice failed task and reschedule it.

1. **What is processing paradigm of spark?**

**How Spark does processing?**

We create RDD from the data to be processed and we apply transformations and actions on RDDs.

One action corresponds to one Job.

Then the job is further divided to tasks and submitted to executors for processing.

DAG (Directed Acyclic Graph).

Once the program is written it will be converted to DAG.

1. **How is in-memory processing being significant in spark?**

Speed: It improves processing speed.

If partition fits into memory it will bring everything into memory.

Suppose if we are doing any iterative computation.

If we do in spark we can persist that RDD in memory. As it is residing in memory the computation is very fast.

If we do in Map Reduce, every iteration it launches MR job as a result every time it reads from disk most of the time is spent in disk IO only.

# Partitioning

https://www.talend.com/blog/2018/03/05/intro-apache-spark-partitioning-need-know/

## Does spark partition the data automatically?

Yes

## What are the default number of partitions if not mentioned? IMP

* **Number partitions will be equal to number of cores in all executors**: If the RDD is created newly by driver code say by loading some unpartitioned external data like file or by using internal collections.
* **It will be equal to number of partitions in external source**: for example, if the data is read from Kafka then number of partitions of RDD will be equal to number of partitions in Kafka.

## What are the characteristics of partitions in spark?

* Every node in a Spark cluster contains one or more partitions.
* One partition in spark will never be distributed to multiple nodes
* Tuples in the same partition are guaranteed to be on the same machine.

## What will happen if we have less partitions?

* causing less parallelism
* data skewing (single partition will have more data)
* improper resource utilization

## What will happen if we have too many partitions?

causing task scheduling to take more time than actual execution time.

## What is the relationship between number of partitions and tasks?

Spark assigns one task per partitions

## What are the types of partitions in spark? IMP

https://www.waitingforcode.com/apache-spark-sql/range-partitioning-apache-spark-sql/read

Type of partitioning

|  |  |
| --- | --- |
| Hash partition | Range partition |
| As the name suggests it is based on hashing technique, depending on the partition key data will be distributed across multiple partitions.  Object.hashCode method is used to determine the partition in Spark as partition = key.hashCode () % numPartitions. | Range partition works better when key is having range of values like age.  For example  0-20, 20-40 |
|  | We can call repartitionByRange method to repartition the data using range. |

## What are the factors that affect choosing number of partitions? IMP

* Number of available cores in cluster on which task will run
* Size of local collections Cassandra table or HDFS file , kafka partitions

## What is the relationship between spark partition and HDFS block size? IMP

1 HDFS block is equal to 1 Partition

## What is the relationship between spark partition and Parquet? IMP

One parquet file is one partition as it is not splitable.

## How many tasks we should assign per core of an Executor? IMP

As a performance best practice, we need to assign 2-3 tasks per Executor core.

## If you are reading parquet file or some of the big file how partition will happen?

Spark has configuration **spark.sql.files.maxPartitionBytes** One partition will have maximum these many bytes of data.

## Difference between repartition and coalesce? IMP

|  |  |
| --- | --- |
| Repartition | coalesce |
| Repartition creates new partition and does full shuffle. | coalesce uses existing partitions to minimize the amount of data that are shuffled |
| Because repartition creates new partitions, results in roughly equal sized data in partitions. | Coalesce reuses existing partitions, so partition data will be of different sizes |
| We can increase or decrease the number of partitions using repartition | We can only decrease number of partitions using coalesce |

* **Downloading and Getting started**

**Introduction to core spark concept**

1. **How Driver program accesses spark?**

Driver programs access Spark through a SparkContext object, spark context is a connection to a computing cluster.

SparkContext is automatically created for

us, as the variable called sc.

1. **How can you build RDDs?**

Using sparkContext we can build RDDs. Then we can run various operations on this RDD such as count etc.

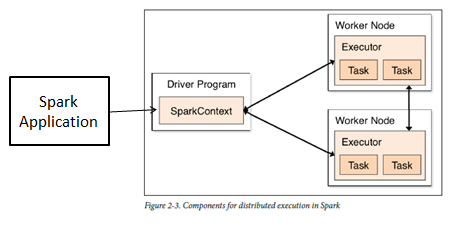
For example we call sc.textFile on a text file to create an RDD. In this case RDD is the lines of text in a file.

We can then run various operations on these lines, such as count ().

1. **Where the operations on RDDs executed?**

The operations on RDDs are executed at multiple executors.

For example, if we run count () operation on RDD on a cluster, different machines count lines in different ranges of the file. That is line from 1 to 10 might be executed in one machine and 11 to 20 in a different machine and so on.



1. **What is the client jar for spark core component?**

groupId = org.apache.spark

artifactId = spark-core\_2.10

version = 1.1.0

Version varies based on spark version

1. **How to initialize spark context in your spark application?**

import org.apache.spark.SparkConf;

import org.apache.spark.api.java.JavaSparkContext;

**SparkConf conf = new SparkConf();**

**conf.setMaster("local"); // cluster URL**

**conf.setAppName("My App"); // spark application name**

**JavaSparkContext sc = new JavaSparkContext(conf);**

//initialize spark context for java

* A cluster URL, namely “local” which tells Spark how to connect to a cluster.

“local” is a special value that runs Spark on one thread on the local machine, without connecting to a cluster.

* An application name: say “My App”.

This will help us identifying our application on the cluster manager’s UI if you connect to a cluster.

Additional parameters also exists using those we can configure how our application executes or adding code to be shipped to the cluster.

Once SparkContext is initialized, we can use all the methods to create RDDs (e.g. from a text file) and manipulate them.

1. **How to shutdown or stop spark application?**

sc. stop();

To shut down Spark, you can either call the stop() method on your SparkContext,

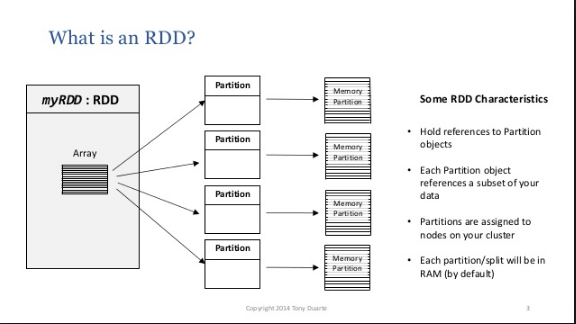
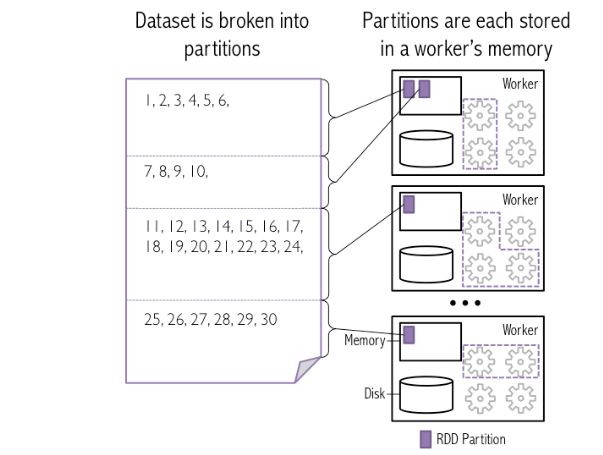
or simply exit the application (e.g. with System.exit(0) or sys.exit()).

**Building standalone applications**

TODO: can refer learning spark book for the details.

## Spark Core: RDD

### What is RDD?

** **

1. RDD is fundamental data structure of Spark.
2. It is an immutable distributed collection of objects.
3. RDD is divided into logical partitions, each partition id processed on different nodes of the cluster.
4. RDD is a fault-tolerant collection of elements those can be operated on in parallel.
5. **What is the use of RDD in spark**

Spark uses RDD to achieve faster and efficient MapReduce operations.

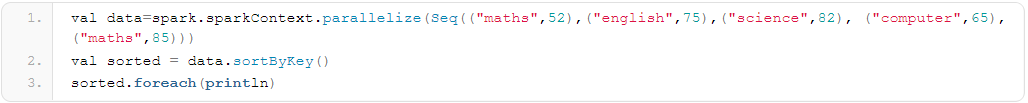
1. **Ways to create RDD?**

There are 3 ways to create RDD

* 1. **Parallelized collection (parallelizing)**

By taking an existing collection in the program and passing it to Spark Context’s parallelize () method.

JavaRDD<String> lines = jsc.parallelize(Arrays.asList("pandas", "i like pandas"));



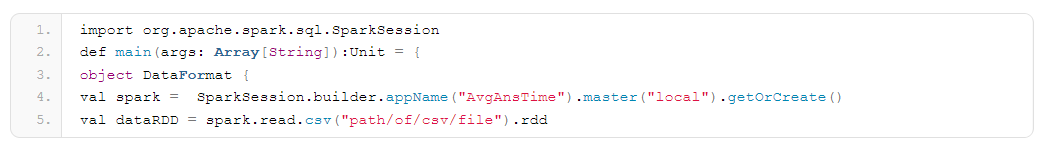
* 1. **External Datasets (Referencing a dataset)**

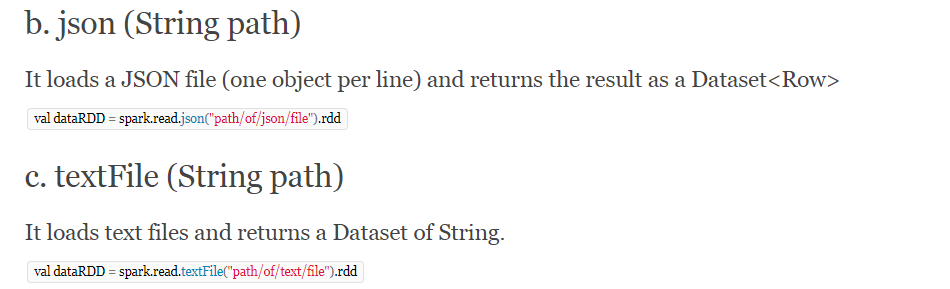
We can create RDD by loading data from the external sources like local file system, HDFS, Cassandra, and HBase.

There are two ways to create RDDs

1. csv (String path)

It loads a CSV file and returns the result as a Dataset<Row>.

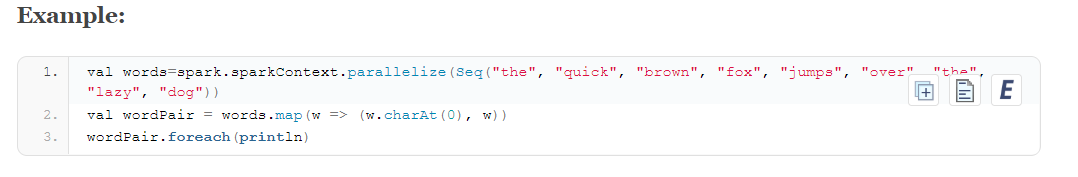




* 1. **Creating RDD from existing RDD**

Transformation changes one RDD into another RDD, thus transformation is the way to create an RDD from already existing RDD.

Some of the operations applied on RDD are: filter, count, distinct, Map, FlatMap etc.



1. **Storing the RDD?**

Spark provides functions to store the RDDs to variety of data storage like

Text file: saveAsTextFile()

JSON

Sequence Files

Collections.

RDBMS: move to local collection and store

1. **What are the operations that we can do on RDD once it is created?**

Transformations and Actions

1. **What are transformations?**
2. **Transformations:** construct a new RDD from an existing RDD.

Ex: Map,flatMap,Filter, union and intersection

Transformation can actually operate on any number of input RDDs

*Both new and existing RDDs will be there*

*Transformtions are lazily evaluated. Transformations will not be executed until an action is requested*

JavaRDD<String> inputRDD = sc.textFile("log.txt");

JavaRDD<String> errorsRDD = inputRDD.filter(

new Function<String, Boolean>() {

public Boolean call(String x) { return x.contains("error");

}

});

1. **Transformations on RDDs are lazily evaluated what does that mean?**

Transformations will be executed only when action is performed on RDD.

Lazy evaluation means that when we call a transformation on an RDD (for instance

calling map), the operation is not immediately performed. Instead, Spark internally records

meta-data to indicate this operation has been requested.Transformation will be performed once spark see an action.

Loading data into an RDD is lazily evaluated in the same way transformations are. So

when we call sc.textFile the data is not loaded until it is necessary. Like with transformations,

the operation (in this case reading the data) can occur multiple times.

1. **Explain MAP Transformation?**

newRdd = rdd.map(function);

1. Map acts on one element at a time. Function that is passed will be executed once for each row.

Generally, the operation can be anything checking condition, cleaning data.

1. Result RDD will have same number of elements as original RDD.
2. Output RDD can be of same type or of Different type. Source RDD can be string, result can have RDD of Some other object.
3. Use cases:

Data Standardization: First Name, Last Name

Data type conversion:

Element level computation-Tax computation

Add new attribute – Grade based on test score

1. **Explain FLATMAP Transformation?**
2. This is same as map only difference is

FlatMap can return more number of records than original RDD.

newRDD = rdd.flatMap(function)

Flatmap applies a function to all elements of this RDD and returns new RDD, and then flattening the results.

1. UseCase:

Some split operations. Because when we split multiple records will be returned.

1. **Explain FILTER Transformation?**

newRDD = rdd.filter(function);

1. Filter function acts on each element of the RDD newly returned RDD will have only elements which matches the condition.

Usually resulting RDD will have lesser elements than the original RDD.

1. **Explain SET Transformation?**

Set operations are performed on 2 RDDs

First

Union:

unionRDD = firstRDD.union(secondRDD);

When we do union of 2 RDD it returns all the records in both RDD as single RDD but records will not be duplicated.

Union does row level merging.

Intersection:

intersectionRDD = firstRDD. intersection(secondRDD);

Intersection returns common elements in first and second element.

1. **Drawbacks of by key transformations?**

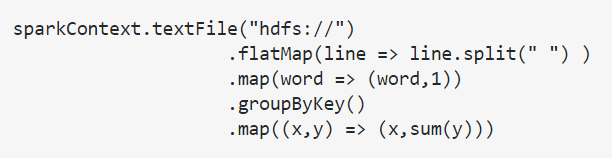
Bykey transformations are expensive operations as they involves shuffle as a result there will be lot of data exchange between the network hence low performance.

1. **What does reduceByKey (This method is there in JavaPairRDD) do?**

Merge values of each key

## Difference between GroupByKey, ReduceByKey? Imp

**groupByKey**



groupByKey is just to group your dataset based on a key. It will result in data shuffling when RDD is not already partitioned.

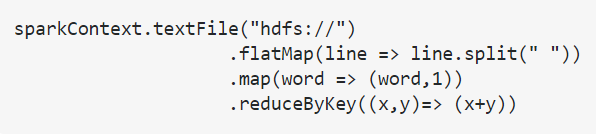
When a groupByKey is called on a RDD pair the data in the partitions are shuffled over the network to form a key and list of values.

The disadvantage of group by key is it will try to bring all the values into memory. Since list of values is in memory collection it will bring all values into memory and there is chance that it will throw out of memory exception.

Group by key there will be no aggregation function.

*JavaPairRDD<K,Iterable<V>> groupByKey()*

**reduceByKey**



Data are combined at each partition, only one output for one key at each partition to send over the network. reduceByKey required combining all your values into another value with the exact same type.

In reduce by key what happens is since we are defining an aggregation function aggregation function will be defined on each partition on every partition local aggregation happens and finally global aggregation so amount of data shuffled across the network is very less also it is not storing any kind of values in the in memory collection we will be getting aggregated value per key

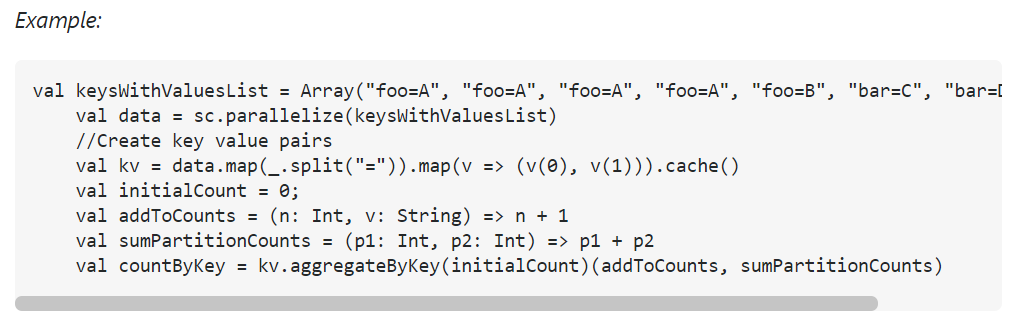
Reduce by key defines an aggregation function.

*JavaPairRDD<K,V> reduceByKey(Function2<V,V,V> func)*

**aggregateByKey:**

same as reduceByKey, which takes an initial value.

3 parameters as input i. initial value ii. Combiner logic iii. sequence op logic



output: Aggregate by Key sum Results bar -> 3 foo -> 5

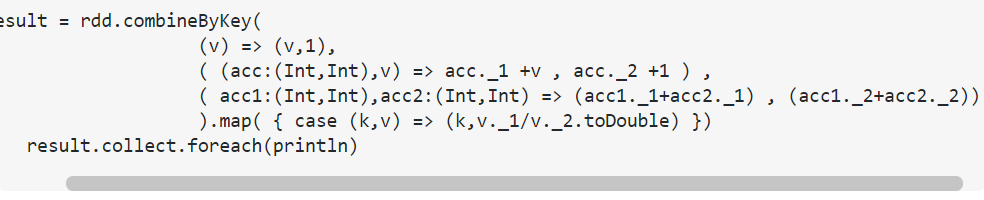
**combineByKey:**

3 parameters as input

Initial value: unlike aggregateByKey, need not pass constant always, we can pass a function that will return a new value.

merging function

combine function



1. **Performance impacts of groupby key?**

Since we are using in memory collection there can be out of memory exceptions

1. NEED TO LEARN OTHER TRANSFORMATIONS WHEN TIME PERMITS THERE ARE MANY

**Actions:**

1. **What are actions?**

Analyze RDD and produces result. Ex: collect, count, first, take, reduce

System.out.println("Input had " + badLinesRDD.count() + " concerning lines")

System.out.println("Here are 10 examples:")

for (String line: badLinesRDD.take(10)) {

System.out.println(line);

}

1. **Explain Reduce Action?**

**rdd.reduce(function);**

Perform an operation across all elements of RDD.

Here the operation is function which takes 2 input values this function is executed for all the elements of RDD.

Below is sample to show how the reduce works

**Input RDD = [a,b,c,d,e]** here a b c d e are numbers

And the **function is func(x,y)** say function adds x and y

The above function is executed as below.

Func(Func(Func (Func(a,b),c),d),e)

First time function will be called for a,b the next call will be result of the first call and c

We should write the function logic-

1. **Does the action execute on workers or on driver?**

An action executes on the worker nodes where each partition is stored and returns the result back to driver. Its driver’s responsibility to aggregate the result.

1. **Can a transformation be applied from action?**

No.

1. **Is it possible to have more than one action from same RDD?**

Yes

1. **Can we Persist RDD?**

Yes we can persist RDD to memory and to disk

**TODO: Write code to persist RDD to disk**

1. **What are the implications of persisting large RDD in memory? What happens if we persist large RDD in to memory?**
2. It eats up executor memory so limited memory for execution.
3. If RDD is too large it will occupy heap space
4. GC over head
5. In case if any failure occurs whole RDD will be lost
6. **What is the behavior of Persting RDD with storage level MEMORY\_AND\_DISK?**

When we persist RDD with storage level memory and disk RDD will be persisted to both memory and disk in such case even if some failure occurs like we lost executor RDD will be there in disk. So next time no need to compute RDD can be loaded from disk directly.

The only disadvantage in case of persisting RDD in both memory and disk memory can be full in case of large RDD.

**Fault Tolerance:**

Fault mean failure there are multiple types of failures like task, job, hardware Etc

1. **What happens when driver fails? IMP**

When driver fails all executors running in the cluster are killed along with the data in the memory. This data cannot be recovered even if driver is restarted.

1. **How to avoid the data loss happed when the driver is down?**

By using apache spark streaming’s write ahead logs feature we can avoid loss this is only in case of receiver based approach.

For kafka direct approach using checkpointing is enough because spark streaming knows which offset to which offset the data is read.

1. **How did fault tolerance is taken care by RDD?**

RDD is fault tolerant because they are **immutable.**

In case one of the tasks fails.

Fault tolerance is achieved by **Lineage graph.**

If any task fails, using lineage graph we can identify which transformation needs to be applied on previous RDD to arrive at this RDD.

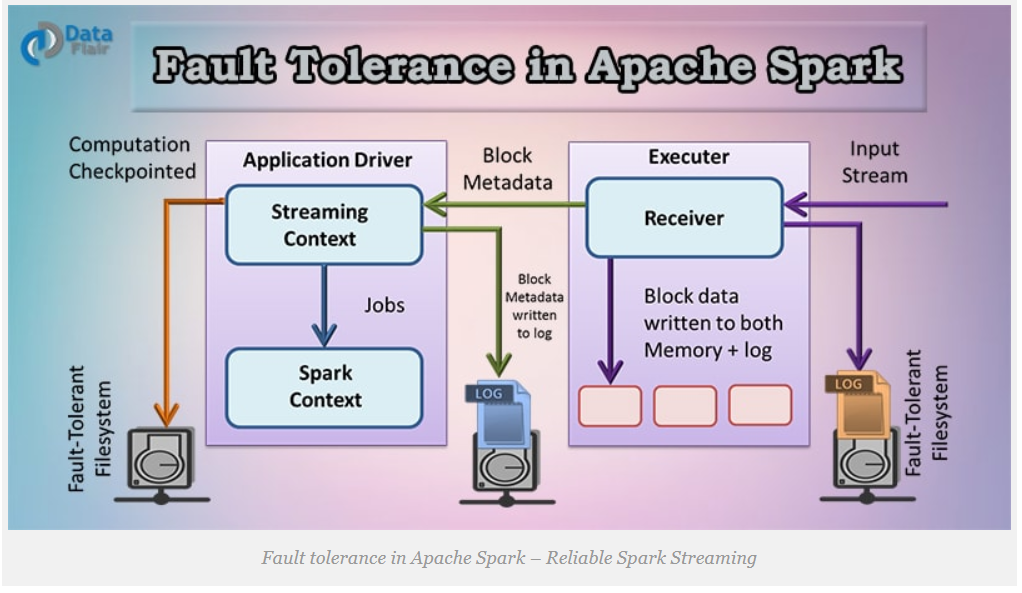
We check if previous RDD is existing in the memory or not if RDD exists we load RDD and compute the RDD to arrive at required RDD if not we go one step before and check previous operation.

**Note:**

Spark replicates data of an RDD among multiple spark executors in worker nodes of cluster. So in this case we need to recover 2 types of data.

**Data received and replicated:** so data is already received and replicated in this case data can be retrieved from other nodes.

**Data is not buffered:** When data is not replicated. Only way to recover from data is to get the data from source again.



Just read

1. **In case of Spark SQL no need to worry much about fault tolerance or loss of data why?**

Because most of the time we will be operating the data that is stored in fault tolerant file system like S3 and HDFS.

1. **Is the RDD computation replicated in memory? Why not? Are we maintaining multiple copies of RDD?**

No.

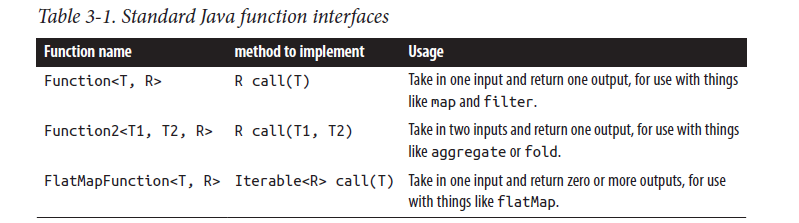
1. **Passing functions to spark?**

We need to pass functions to most of the spark transformations and actions those functions will be used by spark to compute data

1. **How do you pass functions to spark using java?**

In Java functions are specified as objects (class) these objects must implement one of the function interfaces that are present in the package org.apache.spark.api.java.function

Standard Java function interfaces



Java function passing with anonymous inner class.

RDD<String> errors = lines.filter(new Function<String, Boolean>() {

public Boolean call(String x) { return s.contains("error"); }

});

Java function passing with named class. Prefer this type as it is good practice

class ContainsError implements Function<String, Boolean>() {

public Boolean call(String x) { return x.contains("error"); }

}

RDD<String> errors = lines.filter(new ContainsError());

Java function class with parameters

class Contains implements Function<String, Boolean>() {

private String query;

public Contains(String query) { this.query = query; }

public Boolean call(String x) { return x.contains(query); }

}

RDD<String> errors = lines.filter(new Contains("error"));

Java function passing with lambda expression in Java 8

RDD<String> errors = lines.filter(s -> s.contains("error"));

1. **What are the common transformations and actions of RDD?**

**Transformations:**

1. **Map(func) :**

The map transformation takes function as input and applies that function to each element in the RDD. Result of the applied function will be new element for the RDD which will be formed newly from transformations.

Ex:

JavaRDD<Integer> rdd = sc.parallelize(Arrays.asList(1, 2, 3, 4));

JavaRDD<Integer> result = **rdd.map(**new Function<Integer, Integer>() {

public Integer call(Integer x) { return x\*x; }

});

System.out.println(StringUtils.join(result.collect(), ","));

Note: Return type of the map does not have to be the same as the input type.

1. **filter(func):** The filter transformation takes function as input and returns an RDD which only has elements that pass the filter function condition.

Ex: rdd.filter(x => x != 1) {2, 3, 3}

1. **Can a transformation be applied on action?**

No

**Lazy Evaluation:**

1. **What is Lazy Evaluation?**

Spark will load data, create data and performs transformations on RDD only when an action is performed.

Say I write a small program which does operations like

Load file into RDD.

Filter RDD

Count of number of elements.

In this case only when count function is called loading data to RDD and filtering RDD will be performed.

1. **Why lazy evaluation?**

It will help spark to optimize operations and resource usage.

Some points to watch out for lazy evaluation: If you find an error while executing action it is may be due to previous transformations.

**Spark execution model**

Create DAG of RDD to represent computation

Create logical execution plan for DAG

Schedule and execute individual task

The transformations that can be executed in parallel is one pipeline or stage

## Direct Acylic Graph in Apache Spark

### What is DAG in spark?

Set of Vertices and Edges, where vertices represent the RDDs and the edges represent the Operation to be applied on RDD. Each edge in the DAG is directed from one vertex to another .

### When does Spark creates DAG?

Spark creates DAG when an action is called.

### How DAG works in Spark?

At a high level, when an action is called on the RDD, Spark creates the DAG and submits the DAG to the DAG scheduler.

1. The DAG scheduler divides operators such as map, flatMap, and so on, into stages of tasks.
2. The result of a DAG scheduler is a set of stages.
3. The stages are passed on to the Task Scheduler.
4. The Task Scheduler launches tasks via Cluster Manager.
5. The worker executes the tasks.

### What does stage will have?

Stage consists of tasks based on partitions of the input data

### What are the transformations applied by spark to create DAG? How stages are decided? IMP

Two transformations are applied to create DAG.

1. *Narrow transformations (operators):* The operators that don’t require the data to be shuffled across the partitions are grouped together as a stage.

Examples are map, filter, and so on.

1. *Wide transformation (operators):* The operators that require the data to be shuffled are grouped together as a stage. An example is reduceByKey.

### Where you can see the DAG visually?

DAG visualization can be viewed through web UI

*http:/localhost:4040/jobs/*

### Explain DAG Visualization with an example?

Let’s take word count Example:

Data:

Hadoop is the Elephant King!

A yellow and elegant thing.

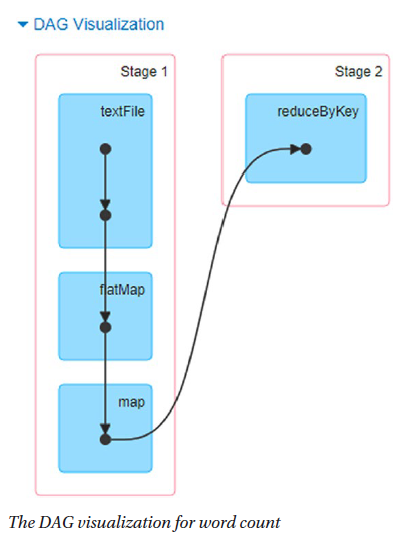
He never forgets

Useful data, or lets

An extraneous element cling!

sc.textFile("/home/usr/keywords.txt").flatMap(line => line.split(" ")).map(word => (word,1)).reduceByKey(\_+\_).collect()

For the DAG visualization of word count. The word count problem consists of two stages. The operators that do not require shuffling (flatMap() and map() in this case) are grouped together as Stage 1 and the operators that require shuffling (reduceByKey) are grouped together as Stage 2.



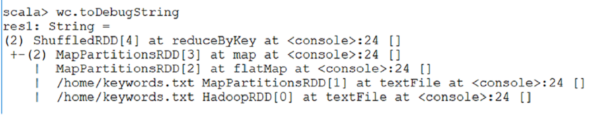
### How Spark Achieves Fault Tolerance Through DAG? IMP

Spark maintains each RDD’s lineage (that is previous RDD on which it depends) that is created in DAG to achieve fault tolerance.

When any node crashes, Spark Cluster Manager assigns another node to continue processing. In such case spark reconstructs RDD’s partition by executing series of operations on the source data partition.

### How to see the lineage graph of an RDD? Show for example of word count? IMP

To view the lineage, use *toDebugString*



### Is lineage graph being one or independent for each RDD that you create in your code?

Each RDD will have their own lineage graph.

Additional questions

### What is the relationship of task and core in Spark?

The number of cores of spark has no direct relationship to physical CPU cores, instead it is a logical counter which determines the number of concurrent tasks that are able to run on one executor

### What is the relationship between number of tasks in stage and partitions?

Mostly number of tasks will be equal to number of partitions.

### In stage will one task perform multiple transformations?

Yes. If there multiple transformations (operator) in one stage all transformations will be executed by same task

### How spark knows which operator is narrow operator and which is wide operator?

There is no fixed rule that a particular operator is always a narrow operator or wide operator it depends on situation mostly the previous operation performed on RDD and partitions.

Note: But few operator will always be narrow like map.

https://github.com/rohgar/scala-spark-4/wiki/Wide-vs-Narrow-Dependencies

### What does DAG optimizer do?

DAG optimizer helps to rearrange or combine operators (Operators are mostly transformations) whenever possible.

If we submit a spark job which has a map() operation followed by a filter operation. The DAG Optimizer will rearrange the order of these operators since filtering will reduce the number of records to undergo map operation.

### Advantages of DAG in Spark?

1. The lost RDD (mostly partition) can be recovered using the Directed Acyclic Graph. Hence helps to achieve fault tolerance
2. It can do a better global optimization than a system like Hadoop MapReduce.
3. Map Reduce has just two queries the map, and reduce but in DAG we have multiple levels. So to execute SQL query, DAG is more flexible.

### What is the difference between Lineage graph and Directed Acyclic graph (DAG)?

*Lineage Graph:* Lineage graph is associated with each RDD it is representation of dependencies of particular RDD with its parent RDDs.

*DAG (Directed Acyclic Graph):* DAG is split into stages where each stage will have tasks

Lineage graph is only till transformations whereas DAG shows complete task transformation +action

### What is the purpose of DAG scheduler?

1. The DAG scheduler divides operators such as map, flatMap, and so on, into stages of tasks.
2. Identifies preferred location to run each task.
3. Handles failures due to shuffle output files being lost.

# Persisting (Caching) RDD

## What is persisting or cache feature in spark? IMP

Persisting an RDD stores the computation result in memory and reuses it to perform other actions on that data set. This helps future actions to be performed much faster.

## How to Persist an RDD?

We can use persist () or cache () methods on RDD.

The default storage level is StorageLevel.MEMORY\_ONLY, we can be set by using the cache () method.

RDD can be persisted using a different storage level to do so we can pass the StorageLevel object (Scala, Java, and Python) to persist ()

For example persist (StorageLevel.MEMORY\_ONLY).

## What are the storage levels supported in spark? IMP

|  |  |
| --- | --- |
| **Storage level** | **Meaning** |
| MEMORY\_ONLY | Store RDD as deserialized Java objects in the JVM. If the RDD does not fit in memory, some partitions will not be cached and will be recomputed each time when they’re needed. This is the default level. |
| MEMORY\_AND\_DISK | Store RDD as deserialized Java objects in the JVM. If the RDD does not fit in memory, store the partitions that don’t fit into the disk, and read them from there when they’re needed. |
| MEMORY\_ONLY\_SER (Java and Scala) | Store RDD as serialized Java objects (one byte array per partition). This is generally more space-efficient when compared to deserialized objects, especially when using a fast serializer, but more CPU-intensive to read. |
| MEMORY\_AND\_DISK\_SER | Similar to MEMORY\_ONLY\_SER, but spill partitions that don’t fit in memory to disk instead of recomputing them on the fly each time they’re needed. |
| DISK\_ONLY | Store the RDD partitions only on disk. |
| MEMORY\_ONLY\_2, MEMORY\_AND\_DISK\_2, etc | Same as the levels above, but replicate each partition on two |

## Which is the default storage level in spark? IMP

MEMORY\_ONLY is default

## In what case spark does persisting by itself without calling persist or cache methods?

Shuffle operations like reuduceByKey are persisted with intermediate data automatically, by doing so spark avoids computation of entire input if a node fails during shuffle.

1. **What is the relationship between job and action?**

One action is one job.

1. **Is it possible to have more than one action from same RDD?**

Yes.

1. **What is default scheduling mechanism of spark job?**

Fair Scheduler.

1. **What is the role of task scheduler?**

Once the DAG scheduler identifies action, job, stages and number of task.

The only job of the

Task scheduler is to launch the tasks of particular stage in executors

1. **How many tasks does spark jobs have?**

Number of tasks will be equal to number of RDD partitions.

1. **On What basis stages are identified in job?**

Based on Shuffle.

1. **What will be there in stage?**

Each stage contains tasks, based on the partitions of the RDD

1. **Should be able to Write DAG for any Given Spark Program practice it?**

**RDD lineage in Spark: ToDebugString Method**

**Note:** RDD lineage graph is also called as RDD operator graph or RDD dependency graph

1. **What is lineage graph in spark?**

Generally in spark all the dependencies between RDDs will be logged in a graph (like how an RDD will be created suppose if we are creating an RDD from existing RDD what are its dependencies). This graph is called lineage graph.

1. **What is the use of Lineage graph?**

Using lineage graph we can recompute missing or damaged partitions due to node failure.

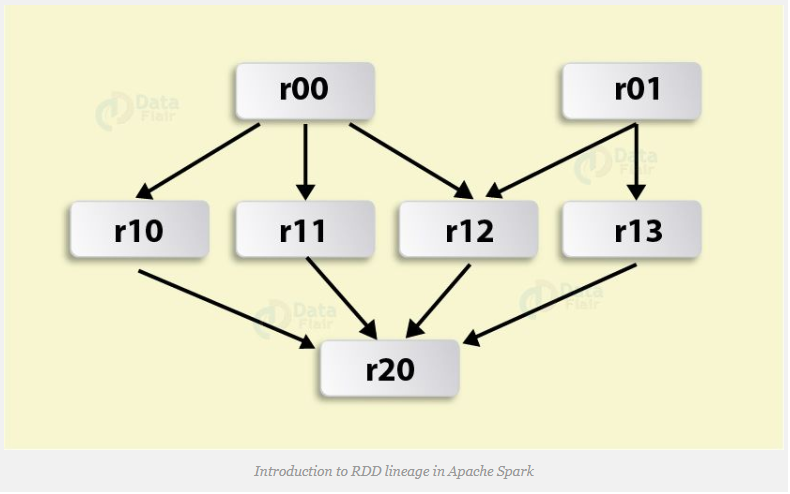
1. **Explain RDD Lineage Graph with an Example?**

We know that we need to perform series of transformation to create an final RDD and perform action on it and all the transformations are lazily evaluated that means they will not get executed till action is performed.

When we create new RDD from an Existing RDD new RDD need to have pointer to its parent RDD and these dependencies will be logged into a graph rather than the actual data and this graph is called as Lineage graph.

RDD lineage is nothing but the graph of all the parent RDDs of an RDD

Example



Above figure depicts an RDD graph, which is the result of the following series of transformations:

val r00 = sc.parallelize(0 to 9)

val r01 = sc.parallelize(0 to 90 by 10)

val r10 = r00 cartesian df01

val r11 = r00.map(n => (n, n))

val r12 = r00 zip df01

val r13 = r01.keyBy(\_ / 20)

val r20 = Seq(r11, r12, r13).foldLeft(r10)(\_ union \_)

After an action has been called, this is a graph of what transformations need to be executed.

If we say, on an

RDD val b=a.map()

Hence, RDD b keeps a reference to its parent RDD a. That is a sort of an RDD lineage.

1. **What is data lineage and how it is computed?**

Data lineage tells what are the transformations to be performed to complete this action. So whenever we perform an actionit will backfire for the transformations which is to be done to complete actions based on that it computes whole pipe line.

**Directed Acyclic Graph DAG in Apache Spark**

1. **What is major factor which creates difference between map reduce and spark?**

DAG

1. **How does spark job gets scheduled on YARN (Yet Another Resource Negotiator)?**

In yarn we have resource managers we have node managers. First thing is Resource manager will lauch container for application master when we have submitted a spark job In application master spark driver will be launched Application manager will talk to resource manager for negotiation of the resources for launching its spark execution once the negotiations are done node managers will launch containers in their respective machines in those containers application master will launch the executors in that executors we will have tasks.

Yarn cluster vs yarn-client

1. **Tuning spark jobs(Executor memory,executor core,number of executors)?**
2. We need to reduce number of shuffles
3. We need to reduce unnecessary persistence of caching If we are persisting RDD make sure we are persisting RDD for definite purpose.
4. Avoid using in-memory collections because they eat away executor memory
5. **Need for serialization?**

If we are not serializing using any serialization framework we are using objects as it is. Serialization converts data into byte format amount of data occupied by the data or our logic is very less. Otherwise it will occupy more memory.

Best serialization for spark is KRYO. If we do not use kryo it will use java serialization by default

1. **Difference between array of RDDs and RDD of arrays?**

Array of RDD Array[RDD]: Array of RDD is collection of RDD array will be existing in driver node. Each array element is an RDD Reference of RDD is in Driver actual RDD is in worker

RDD[Array[String]]: It is one complete RDD where each element is array so one complete RDD reference is in driver and RDD is in the worker.

1. **What are the comparing objects in spark?**
2. **What are the methods available in spark list?**
3. **Why is spark termed as one stop solution for batch and realtime solutions?**

Spark provides spark streaming component for real time data processing. Spark steaming has few streaming relating fucntionalities and APIs . Once they recive the data aging the execution engine will be spark core only.

Any way we can perform batch processing using RDDs

1. **How is spark bridging gap between data Engineers and data scientists?**

**MLlibs, Spark R**

Usually the data scientists community is based out of R and python for executing their workloads. The libraries they use is cycitlan for phython and R uses r packages. What ever is the case the basic fundamental abstraction is dataframe

**Advantages of Spark over MapReduce?**

<https://www.tutorialspoint.com/apache_spark/apache_spark_rdd.htm>

**Submitting Applications in spark:**

**How to launch application on cluster in spark?**

Using **Spark-submit** command: This script in Spark’s bin directory.

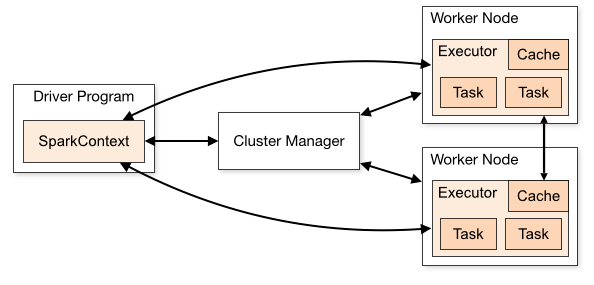
It can use all of Spark’s supported cluster managers through a uniform interface so you don’t have to configure your application especially for each one

**Options of spark submit:**

**--class**: Applications main class (for java/scala support)

**Cluster Mode Overview:**

Spark applications run as independent sets of processes on a cluster, coordinated by the SparkContext object in your main program (called the driver program).

****

To run on a cluster, the SparkContext can connect to cluster managers

Once connected, Spark acquires executors on nodes in the cluster. Next, it sends your application code (defined by JAR or Python files passed to SparkContext) to the executors. Finally, SparkContext sends tasks to the executors to run.

**Cluster Manager**: job of cluster manager is to allocate resources across applications.

**Executors**: are processes that run computations and store data for your application.

About spark submit command

Which version of spark you are using: 2.4

Which version of kafka you are using: kafka\_2.11 0.10.1.1

spark-submit

--master yarn

--deploy-mode cluster

--queue spark\_queue

--class org.dell.saie.analytics.SACSaAgentFileExtractor

--name $job\_name

--files config/application.yaml,config/generic.properties

--verbose

--num-executors 9

--executor-memory 24g

--executor-cores 5

--conf spark.driver.memory=8g

--conf spark.yarn.executor.memoryOverhead=4096m

$job\_name-1.3.0.jar application.yaml generic.properties

&> ${LOG}

Notes:

1. The machine where you submit spark application will act as Driver node and uses other nodes in the cluster as worker nodes.

How will you

## What is Executor memory in Spark?

Memory allocated to each executor for processing spark tasks.

## How do you calculate executor memory or how will you calculate number of executors? IMP

[https://stackoverflow.com/questions/37871194/how-to-tune-spark-executor-number-cores-and-executor-memory#](https://stackoverflow.com/questions/37871194/how-to-tune-spark-executor-number-cores-and-executor-memory)

**Case 1 Hardware - 6 Nodes, and Each node 16 cores, 64 GB RAM**

* First 1 core and 1 GB is needed for OS and Hadoop Daemons, so available are 15 cores, 63 GB RAM for each node.
* How to choose number of cores:

We know that Number of cores = Concurrent tasks as executor can run

Ideally, we should choose number of cores as 5. Any application with more than 5 cores will not perform properly as per proven results.

The number 5 came from ability of the executor not based on how many cores the machine has.

* How to choose number of Executors.

Because we choose 5 cores per executors, we have 15 cores per node. So, we can have 3 executors per node. 3 \* 6 = 18 Executors per node.

Out of 18 executors we need 1 Executor for AM (Application Manager) in YARN.

So, 17 is the number of Executor we need to set.

* How to calculate memory for each executor.

So, we choose to have 3 executors per node and available memory per node is 63GB

63/3 = 21 GB

However small overhead memory is also needed to calculate overhead below is the formula.

0.07 \* spark.executor.memory (0.07\*21 = 1.47)

So overhead memory required in this case is 1.47GB

So executor memory is 21-1.47 = 19.53 ~ 19GB

Final numbers - Executors - 17, Cores 5, Executor Memory - 19 GB

1. **What Dynamic allocation? IMP**

As the name suggests it is dynamic allocation of the executors when there are waiting tasks.

* 1. How to enable dynamic allocation in spark

Set configuration *spark.dynamicAllocation.enabled* = true.

When this configuration is true we no need to mention number of executors.

* 1. How will you set initial number of Executors

Set the configuration *spark.dynamicAllocation.initialExecutors* to number of executors to be started initially.

* 1. How many executors to request based on the Tasks pending (Load)

We can set minimum or maximum number of executors that can be running at any point of time for an application.

spark.dynamicAllocation.minExecutors

spark.dynamicAllocation.maxExecutors

* 1. When Executors should be allocated

If there are pending tasks for the duration set in the configuration

spark.dynamicAllocation.schedulerBacklogTimeout is crossed then new executor will be allocated.

Number of executors requested in each round increased exponentially from the previous round. For instance, an application will add 1 executor in the first round, and then 2, 4, 8 and so on executors in the subsequent rounds. At a specific point, the above max comes into picture

* 1. When do we give away executors?

If any executor is idle for configured time in

spark.dynamicAllocation.executorIdleTimeout is reached then that particular executor will be released.

* **Working with Key/Value Pairs**

1. What is pair RDD?

Special type of RDD that can store key value pair

1. How you will create pair RDD?

Pair RDDs can be created through regular map operations

All transformations available for regular RDDs are also available for pair RDDs

1. Some of the transformation to handle the pair RDD?

mapValues: Transform each value without changing the key.

flatmapValues: generate multiple values with same key.

1. Pair RDD actions?

countByKey: produces a count(number of values) by each key in the RDD

groupByKey: perform aggregation like sum, average by key.

reduceByKey:perform reduce, but by key

aggregateByKey: perform aggregate by key

* **Loading and Saving Your Data**
* **Advanced spark Programming**

1. How local variable works in spark?

Spark makes copies of code (one per partition) executes them

Variables we create in base programming language are local to cluster.

Duplicate copies of local variables for each cluster.

1. How to create global variable and share across cluster?

We can achieve this in spark using

Broadcast Variables

Accumulators

1. What is broadcast variable?

It is read-only variable that is shared across all nodes.

If we want to create small look up object/table we can us broadcast variable.

1. Have you used broadcast variable in your project?

Yes. We have used to broadcast audience drl and Category data object.

1. What are accumulators in Spark?

The variable which is shared across all the executors this variable can be updated by all the executors

1. Have you used accumulators in your project?

Yes. We have used accumulator to accumulate unique users matched for a particular audience.

1. What is Partitioning?

By default all RDDs partitioned.

By default number of partitions created is equal to number of CPU cores available in cluster. For Example in our laptop if we have 2 cores spark will create RDDs with 2 partitions by default..

We can control the number of partitions using spark.default.parallelism parameter.

If the cluster is large then it is suggested to configure this value

1. Can we mention number of partitions to be created explicitly?

Yes we can specify number of partitions during RDD creation.

1. What is Persistence?

By default spark loads the RDD whenever it is required. It drops RDD once the action is completed. If any more action is performed on RDD spark loads and re-computes (transformation) RDD chain.

So persistence allows intermediate RDD to be persisted to the memory so that it need not to recomputed it again.

There is a method persisit(): This method can persist the RDD in memory,disk, shared or in other third party sync.

There is a method cache() – By default it persist RDD in memory

* **Running Spark on a Cluster**

**Cluster manager**

Spark is designed in such a way that it can be scalped up from one to many thousands of compute nodes.

1. **Explain how Spark’s distributed execution works?**



Note: The above figure can be assumed as spark application

* In Spark cluster there will be a Master node and Set of worker nodes In master node we run driver program which is nothing but main method of the application.
* In Driver program we create spark context which is the connection object to spark cluster and gateway to entire spark functionality
* Then there are worker nodes which work as per the instructions of the master. Inside worker node spark executor programs will be keeping running. Executor programs are programs that will be waiting for instructions from the master.
* Driver programs and the executor programs will be connected and controlled by cluster manager

com.myspark.helloworld.ExplainArchitectureExample.java

1. Say we are trying to create RDD from an ArrayList which has 4 rows (containing some names).
2. We can tell Spark context that I want to create RDD from this List . Spark context takes those 4 rows then it works with cluster manager and partition these records based on number of worker node and creates RDD in Worker node.

Say we have 4 nodes then the entire RDD will be partitioned to 4 partitions each partition will go to one worker node. Each partition will be locally stored in worker node.

1. Now if I want to convert all records from small to capital letter we need to apply transformation.

When Transformation is performed Spark context tells executors via cluster manager that this particular transformations should be performed.

Executors then creates local task. Then starts executing the task here task is nothing but converting name to upper case. Each task converts data to upper case in locally stored partition. Then we can ask spark context to return back the converted RDDs then spark context informs all executor through the cluster manager a task will be run and all converted records will be aggregated and Returned back to the driver program.

Spark has one central co-ordinator called as **driver.** The driver then communicates with large number of distributed workers called as **Executors** to get the work done

The Driver runs in its own java process

Each executors runs in their own java process

1. **What is driver program and what is it job?**

Driver program,

Is main function of spark application.

Runs in master node of the cluster

Responsible for controlling and co coordinating all spark operations in the cluster

Defines RDD on the cluster, and then applies operations on them. Launches various parallel tasks on a cluster.

Each driver program execution is called as JOB. Jobs will be converted as tasks in worker nodes

1. **What is SparkContext in Spark?**

Spark context is gateway for all spark functionality.

Spark context is the connection to the spark cluster .

1. Used to create RDDs,partition RDD and distribute them in cluster with help of cluster manager,
2. Also used create accumulators and broadcast variables on that cluster.
3. Split the job into task and executes them in multiple worker node. Collects the results and provide back to driver program.

Manages Executors running in worker node

1. **What are the modes in which I can run spark application?**

**Batch mode:** We can create a application and start it It runs and performs assigned job and end. We can also schedule this job using Quartz,TCC like schedulers.

For Example:Time RTS

**Streaming mode:** The program will be continuously running and listening for incoming data.

For Example: Event RTS

**Interactive mode:** It is like executing spark command one by one in shell (REPL). This mode we can use for local debugging or to test sample programs.

**For Example: spark-shell**

1. **Can the sparkConetxt be shipped to workers i.e used on workers ? Can we pass spark context to workers as parameters**

No.

Spark Context is not serializable. It is Transient variable it exists only on Driver.

1. **What is Driver?What are the operations performed by driver?**

The driver is the process where the main () method of our program runs.

Driver runs the code that creates a SparkContext, creates RDDs, and performs transformations and actions.

Once the driver terminates the application is finished

1. **What is the difference between executor and worker?**

Worker is service running on node

Executor is the process launched per application

1. **Spark Job execution details?**

1. **Does spark provides its own cluster manager?**

Yes spark has built in cluster manager called as **Standalone Scheduler.**

If we install Spark on an empty set of machines, by default the Standalone Scheduler will be the cluster manager.

It means cluster manager is configurable in spark

1. **On which all cluster managers spark can run?**

Hadoop YARN,Apache Mesos

* **Tuning and Debugging Spark**

1. **What is the purpose of DAG scheduler?**
2. **How many tasks does spark job have?**

**Number of partitions in RDD is equal to number of tasks**

1. **What is the role of Task Scheduler?**
2. **What is data lineage how it is computed?**
3. **What is the relationship between job and action?**

One action is equal to one job.

1. **In MR map tasks are JVM processes where as in spark tasks are threads what is the motivation behind changing execution design from process to threads?**

Launching the process is heavy but thread launching is easy because threads are children of process.

Threads can reuse resources allocated to process like memory, cores etc

1. **What is the difference between RDD check pointing and persist?**

RDD Check pointing is different from spark streaming checkpointing.

Note: RDD that is marked for checkpointing should be cached.

Note: It is strongly recommended that this RDD is persisted in memory, otherwise saving it on a file will require recomputation.

|  |  |
| --- | --- |
| **Check pointing** | **Persisting/Cache** |
| Breaks the lineage.  Check pointing stores the RDD in HDFS. It deletes the lineage which created it. | Keeps the lineage intact.  When we persist RDD with DISK\_ONLY storage level the RDD gets stored in a location. We can recomputed the RDD later |
| Once the spark job is completed checkpoint files are not deleted | Once spark job is completed the files are destroyed and cache is cleared. |
|  |  |