Spark RDD Transformations

Example File contents: ( Text File ) read as RDD

Sachin, Cricket Sachin, Cricket

Ronaldo, Soccer 🡪 Ronaldo, Soccer

Dhanraj, Hockey Dhanraj, Hockey

Dhanraj, Hockey Dhanraj, Hockey

<Each element becomes an RDD>

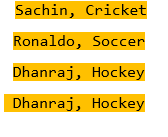
Text File RDD reads each line as RDD. This should be separated by line

JSON/XML files are read as wholetextfiles

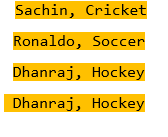
**MAP Function ( N elements input to N elements output )**

RDD(x) 🡪 f(x) 🡪 RDD

Each individual RDD is taken and a function is applied on top of it and the resultant element is an RDD.



Example : Each element is converted to small case, uppercase, date adjustment etc.

 🡪 sachin,cricket

ronaldo,soccer

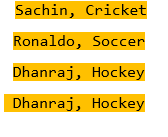
dhanraj,hockey dhanraj,hockey

**FLAT MAP Function ( N elements input to N or , N+ or N- elements output )**

RDD(x) 🡪 f(x) which can return a 0, 1 or multiple elements 🡪 RDD

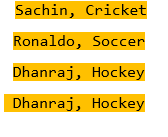
FlatMapFunction returns an Iterable element which can contain 0, 1 or many elements

Example : Each element is split by comma, or each element is uppercased etc.



* Sachin, Cricket, Ronaldo etc…

**FILTER Function ( N elements 🡪 output based on filter)**



🡪 Filter by Dhanraj 🡪



**Set Operations**

1. Union
2. Intersection
3. Subtraction – ( Inter leaving )
4. Distinct
5. Cartesian ( All possible combinations)

All the above is possible using Apache Spark APIS

**Actions**

**Reduce API**

Reduces the RDD elements . This will have to follow associative and commutative property

Commutative property

i.e a+b+c = a+(b+c)

or

Associative

a(bc) = (ab) c

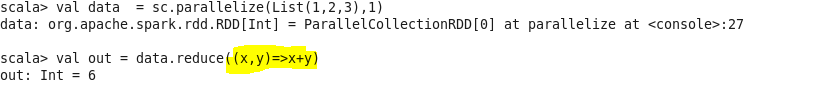
The same thing is not true when

a-b-c is not equal to a-(b-c)

7-4-3 = 0 which is not equal to 7-(4-3) is 6

**Care should be taken cared that what we are trying to reduce is applicable with the above properties**

**Example 1**



1 2 3

This the RDD

Each element gets substituted in place of x, y => x+y

1, 2 -> 1+2

3,3 -> 3+3

**Example 2**



Each element gets substituted in place of x, y => x+y+3

1, 2 -> 1+2 +3

6,3 -> 6+3 +3

**Collect**

Collects all the output from Transformations and sent back to driver program. Make sure this data set is not large because it may fail because of memory issues

**Take(n)**

Displays the first few outputs

**First**

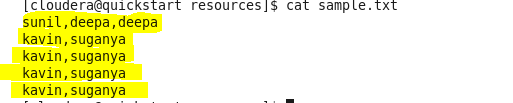
Displays the first output

**saveAS**

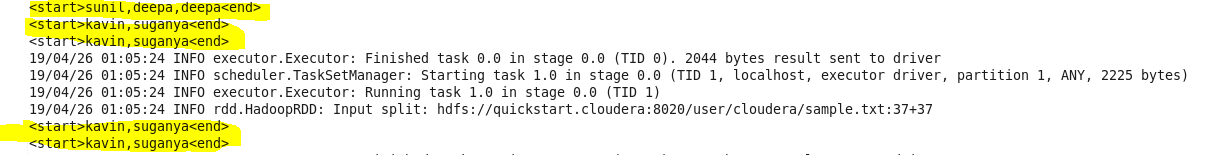
Outputs the file

**forEach Action**

Acts on each individual RDD



Each element is a RDD . Let us say if we apply a tag like <start> RDD <end> . The below is the following result



**Use cases**

* Writing into Kafka Topic
* Writing into DB/Hbase
* Updating the accumulator variable

**Pair RDD**

RDD comes with key value pairs

Common operations performed based on Key based are

* Aggregation
* Sorting
* Sum
* Average
* Count
* Joining etc.

1. PairRDD can be created using mapToPair function in Java

Basically the data structure of the paired RDD is similar to the rows in relational database

Paired RDD is nothing but a tuple 🡪 (Key,value)

**Reduce By Key**

Let us say the below record is split into 2 partitions

1,100

2,300

1,200

**Split into 2 partitions**

1,100 1,200

2,300

Key will be consolidated . One of the value gets eliminated

(x,y) -> y

In this case the output is

1,100

2,300

(x,y) -> x+y

This will consolidate it

**Group By Key**

Let us say the below record is split into 2 partitions

1,100

2,300

1,200

**Split into 2 partitions**

1,100 1,200

2,300

Key will be consolidated .

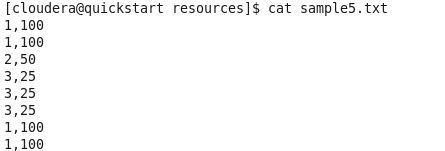
(x,y) -> y

1 , {100 ,200}

2, 300

**Aggregate By Key**

Let us say the below record is split into 2 partitions



**Split into 2 partitions**

1,100 3,25

1,100 3,25

2,50 1,100

3,25 1,100

AggreateByKey takes 3 inputs

1. Increment Value – can be 0 if there is nothing to start with
2. Sequence Function
3. Combiner Function

**Sequence Function :**

Takes element in 1st partition and apply the code

Ex: ( x,y) -> x+ y

**Partition 1**

For each keys , the output is X + Y

* For 1 = 100+100
* For 2 = 50 + 0
* For 3 = 25 + 0

**Partition 2**

For each keys , the output is X + Y

* For 3 = 25+25
* For 1 = 100+100

**Combiner Function**

Ex: ( x,y) -> x +y

For 1 🡺 200+200

For 2 🡺 50+0

For 3 🡺 25 + 50

