SPARK ADVANCED : DATA PARTITIONING

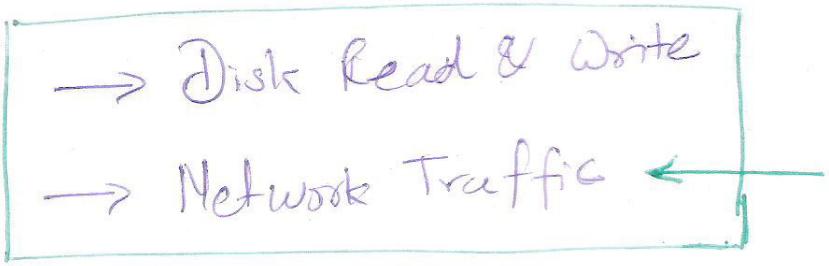
By www.HadoopExam.com

Note: These instructions should be used with the HadoopExam Apache Spark: Professional Trainings. Where it is executed and you can do hands on with trainer.

Spark: Data Partitioning

⇒ Distributing Data Across nodes in cluster, to improve efficiency.

⇒ To biggest performance constreint, while working on distributed platform.



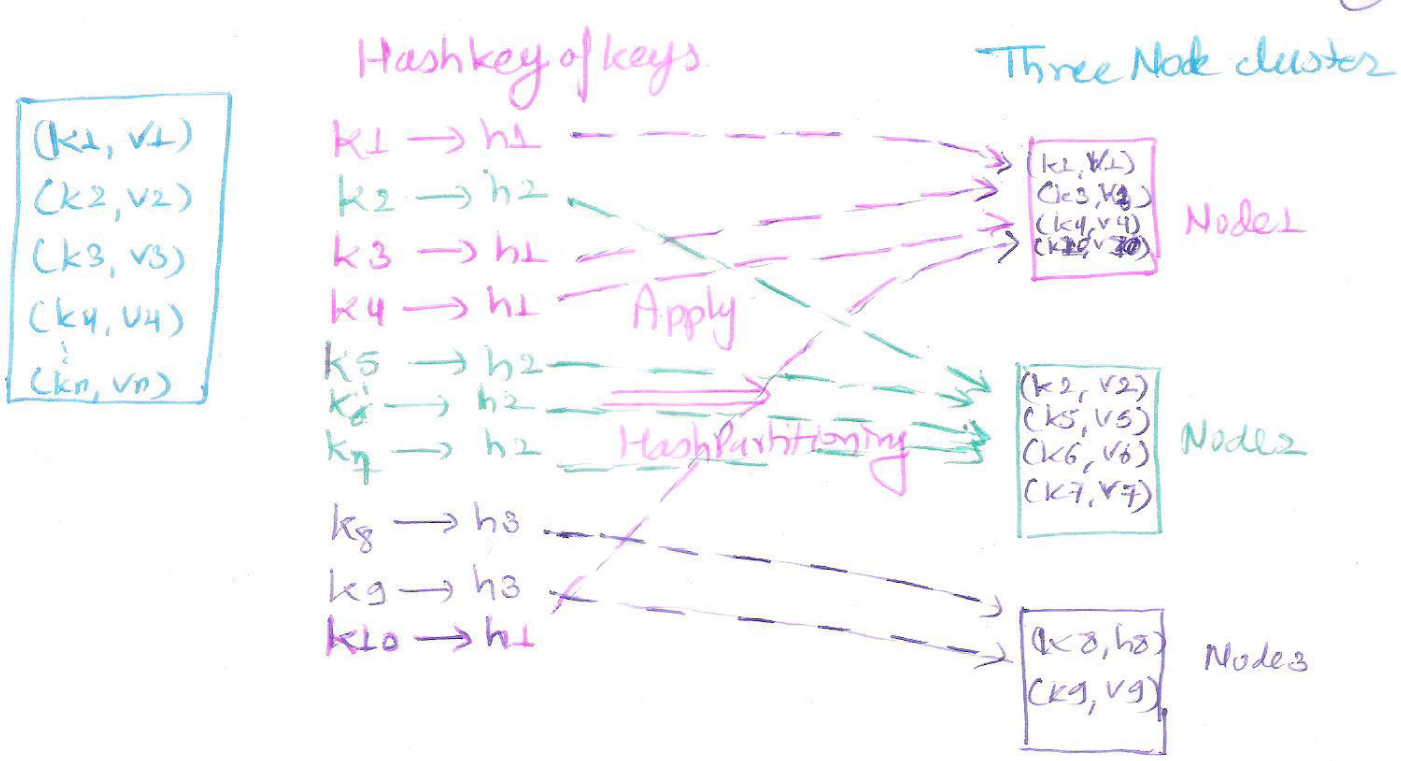
⇒ Data partitioning will reduce n/w traffic.

⇒ Always useful : No

⇒ If given RDD is scanned only once, then no use of partitioning

⇒ When ⇒ In key-oreinted operations (maybe PairRDD), and dataset is re-used multiple times e.g. Lookup Data, used in Join operation.

⇒ Hashkey based partitioning :-



⇒ Now calling persist() method on RDD is Mandatory

Example:- Let's take an example from ~~one~~ Amazon kindle online, web reading application.

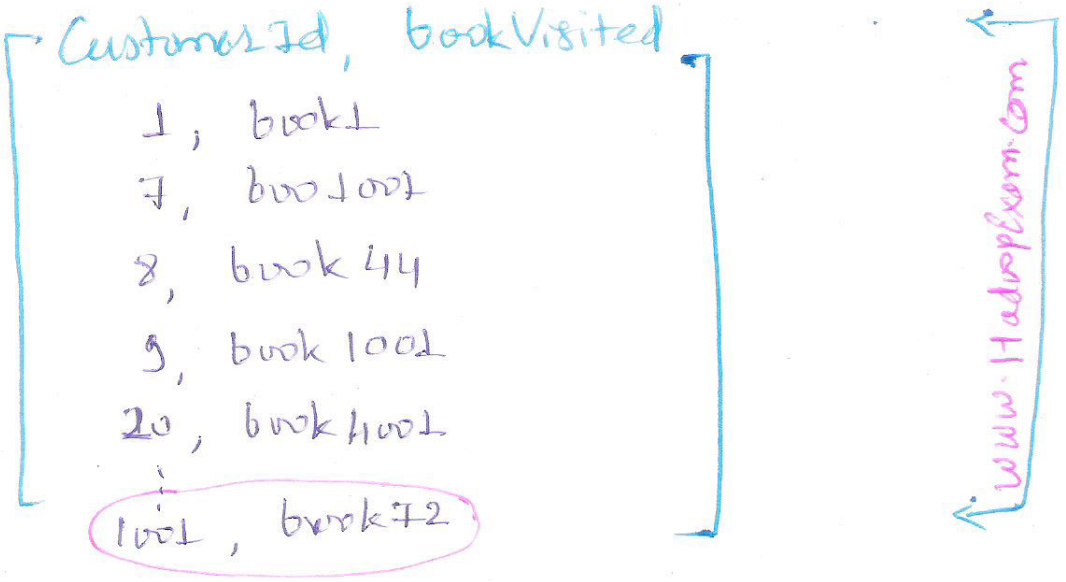
~~Learner~~

Customer_Info (RDD structure)

CustomerId	List [book1, book2, book3 ---- bookn]
1	[book1, book2, book7, book100 ---- book700]
2	[book2, book9, book400 ---------- book405]
----	----
1001	[book7, book8, book44 ---------- book1001]

⇒ CustomerInfo RDD is huge in size.

⇒ Another RDD, which we receive every 10 minutes using Streaming Application. With following structure.



Problem Statement :⇒

Find out the book name, which is not subscribed by a Customer, but he had visited.

For Example :- CustomerId → 1001
Book Visited → book72 (Not subscribed)

Join both the RDD, so Amazon® can find, which book is not subscribed by Customer, but he has interest so it can send recommendation in his kindle® box.

⇒ Why partitioning will help here.

Partitioning of Customer-Info RDD, is required. Because this RDD, we will be using in every 10 minutes to calculate customer Interest.

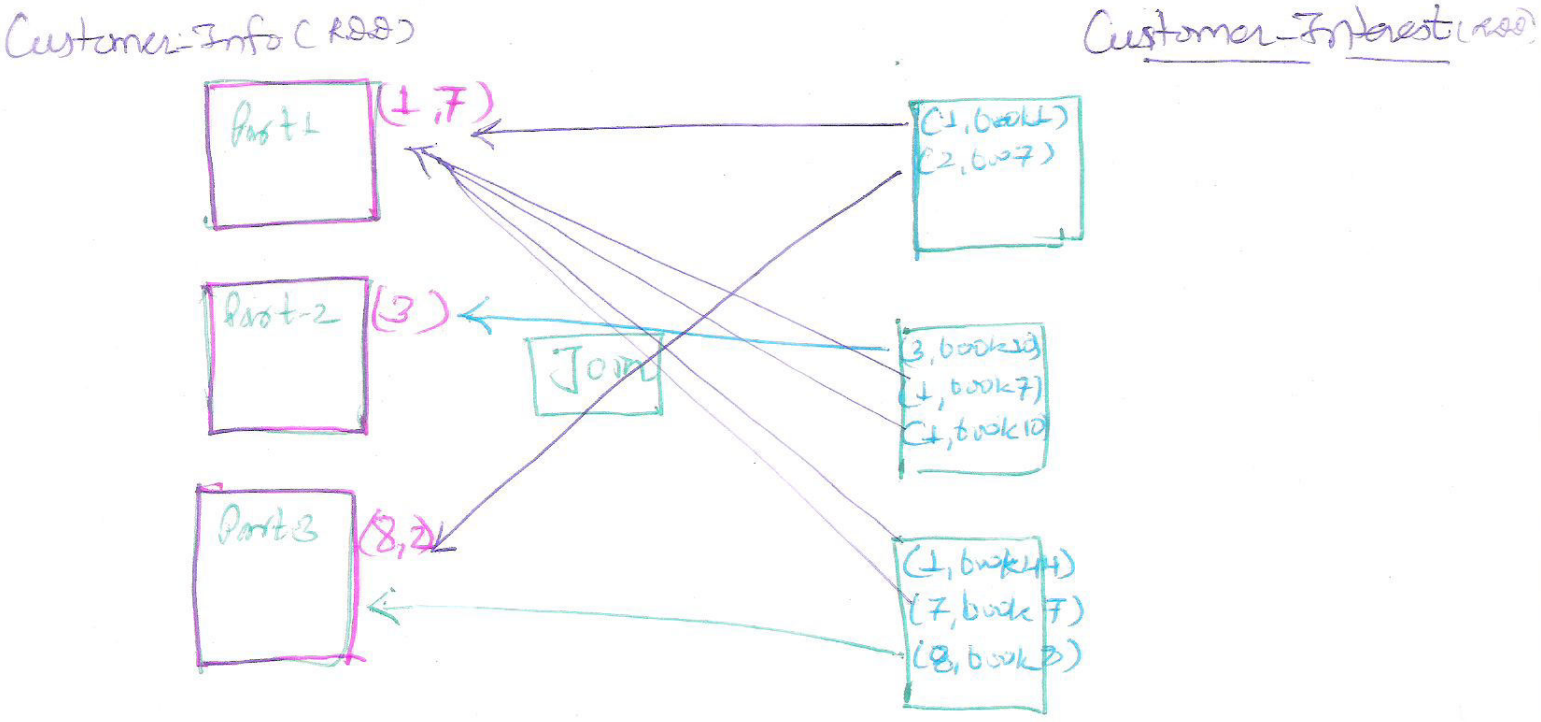
Two RDDs

Customer_Info → Customer_Interest

Big Table
(No need to re-calculate again and again)

Re-generated every 10 minutes from Streaming application.

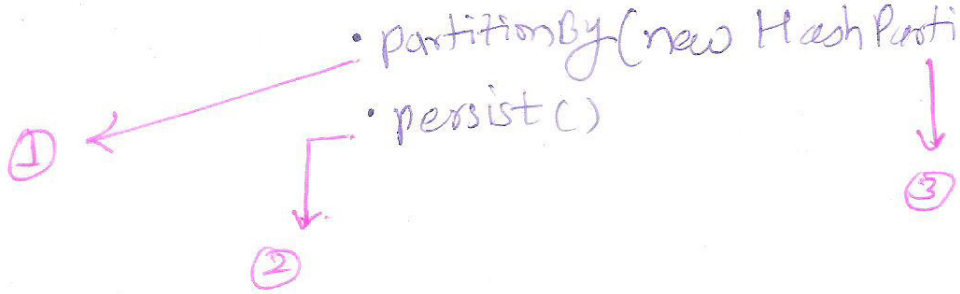
Partition this RDD & persist() its state



Application Code Looklike

~~val user~~

```
val customerData = sc.textFile[custId, list[book]]
    .partitionBy(new HashPartitioner(100))
    .persist()
```



1 → partitionBy is a method to position data.

2 → Algorithm (technique) used to do partitioning.

3 → Save the position created (Dont miss it)

```
def generateData (streamedFile: String) {

    val row = sc.textFile[custId, book] (streamFile)

val result = customerData.join(row)

    // (CustId, [List(Book), Book)] RDD structure joined

val newVisits = result.filter{

    case (custId, (list, book)) =>

        ! list.books.Contains (book.name)

}

newVisits.saveAsTextFile("path");
```

⇒ Operations create Partitioned RDD :-

There are many operations which will create partitioned RDD.

sortBykey() ⟶ Range Partition.

groupBykey() ⟶ hash-partitioned RDD.

map() ⟶ Forget Parent RDD partitions

Why ? ⇒ This map() operation can modify the key

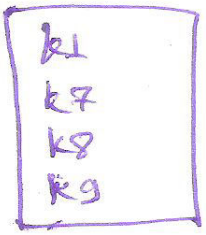
⇒ Operations that Benefit from Partitioning :-

→ Operations which requires shuffling data by key across network.

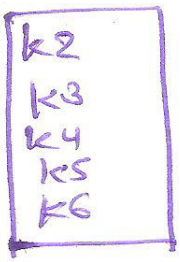
→

cogroup()	rightOuterJoin()	lookup()
groupWith()	groupBykey()	
join()	reduceBykey()	
leftOuterJoin()	combineBykey()	

→ reduceBykey() :- will work only on single RDD.



Part-1 (do operation locally in each partition) ①



Part-2

② Send Result to master node from each node.

Operation That affect Partitioning :-

Calling reduceByKey() after join()

Significantly faster.

Why? Join will create hash-partitioned RDD and reduceByKey() will take benefit of this.

=> map() can change key.

=> mapValues() & flatMapValues() : will not change the key.

Binary Operation :-

① RDD1 → Partitioned by Hash; RDD2; join(); result RDD [it will be partitioned by Hash]

② RDD1 → CustomPartitioner; RDD2 → HashPartitioner; join(); resultRDD [It will be with Custom partitioner]

③ RDD1 ["No-partition"]; RDD2 [No-partition]; join(); result RDD [Default Partition: Hash-partition]

cloudera CCA175 Is Now Available , with Hands On Sessions

CCA Spark and Hadoop Developer Certification

HadoopExam Learning Resource provides the following material for the Advanced Technologies.
Please visit www.HadoopExam.com for more detail this is just a few products from portfolio.

Price start for training with Just $79/3500INR

Apache Spark Professional Training with HandsOn Session + Certification Material

Hadoop Professional Training with HandsOn Sessioon + Certification Material

HBase Professional Training with HandsOn Session + Certification Material

Certification Material

Certification Material

Certification Material

Certification Material

Certification Material

Certification Material

Certification Material