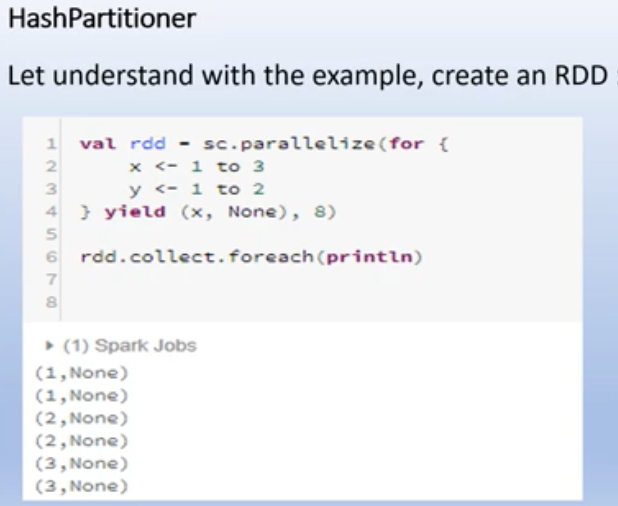
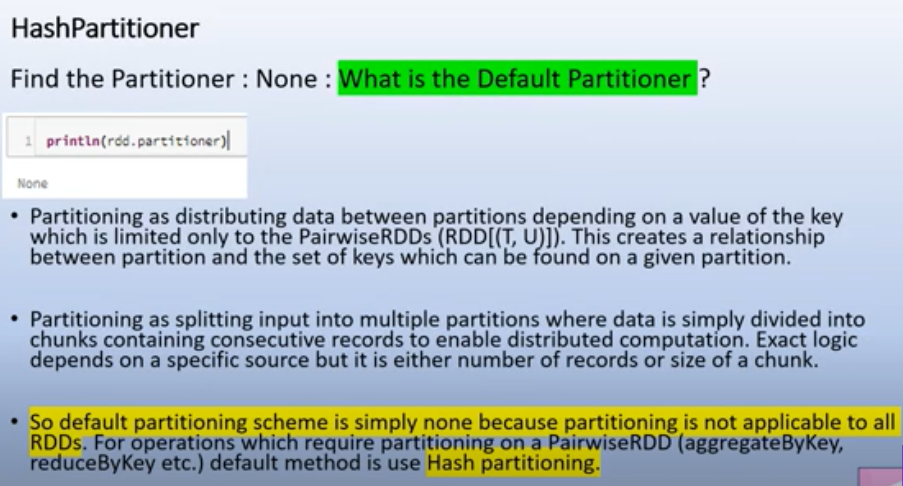
**Hashpartitioner in spark:**

Two partitioner provided in spark i.e. hashpartitioner and rangepratitioner



8 is the no. of partitioner



In above example we give 8 as a no. of partitioner but when we do rdd.partitioner we get as ***None***

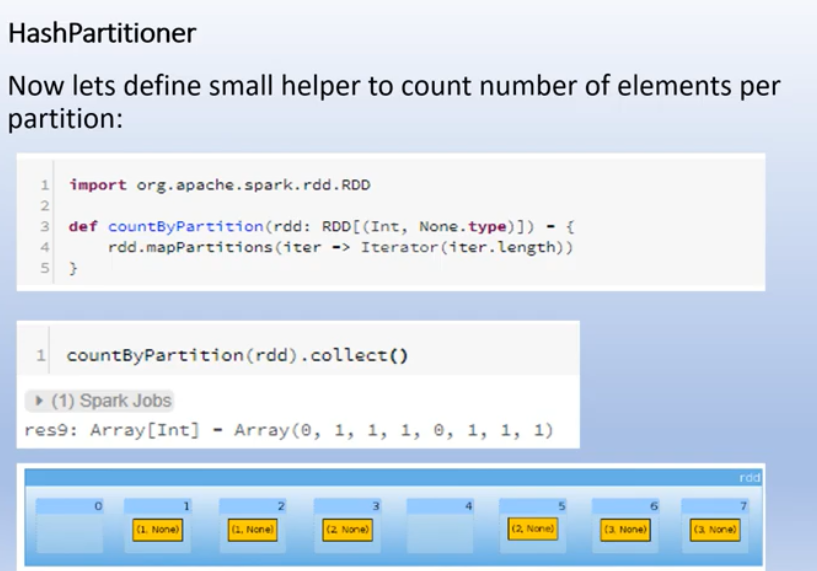
But if we do rdd.getNumpartition it will give the value as 8. So what is the default partition ??

So when we do partitioning in distributed environment is meant to split the data in multiple bucketing.

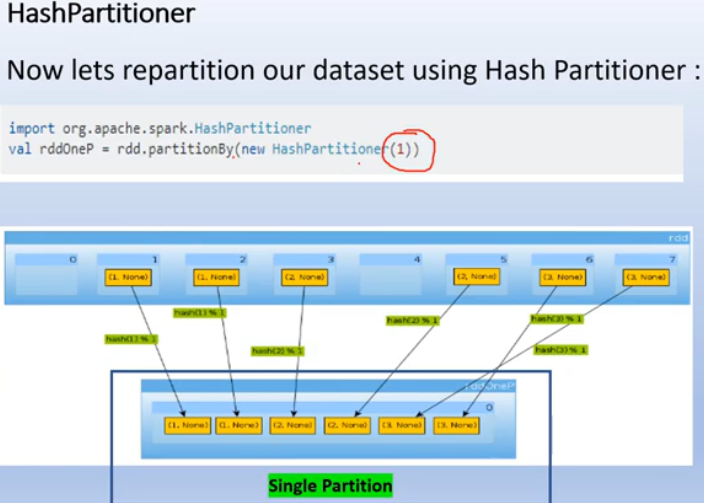
In spark when we don’t defined explicitly any partitioner, it will take account the no. of records and put it in consecutive nodes and once finish the range of cluster it will again come back and try to split in round robin fashion.

The concept of hash partitioner comes only when we have pairedRDDs where we already have key defines there , that key can be used to defined the hash code.

pairedRdds are present in reducebykey, groupbykey.

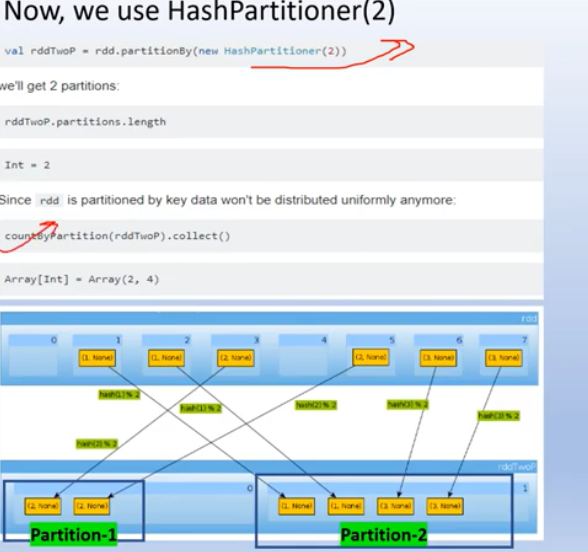


In above example we have 8 partitions in which 2 partitions are empty and 6 partitions have one element each.

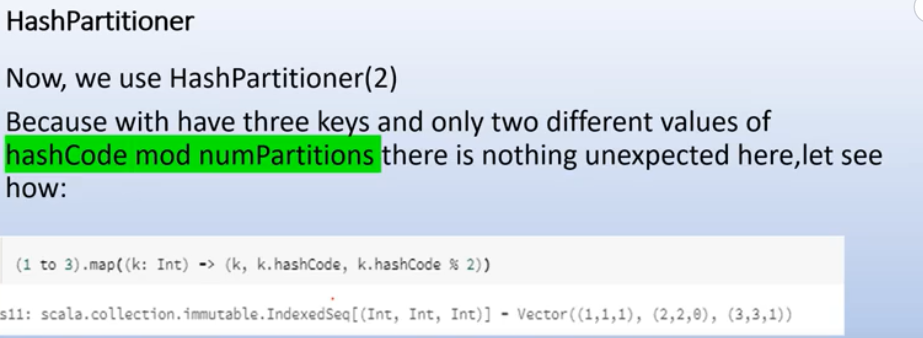


Now we use hash partitioner in which we define partition value as 1, so single partition is created,

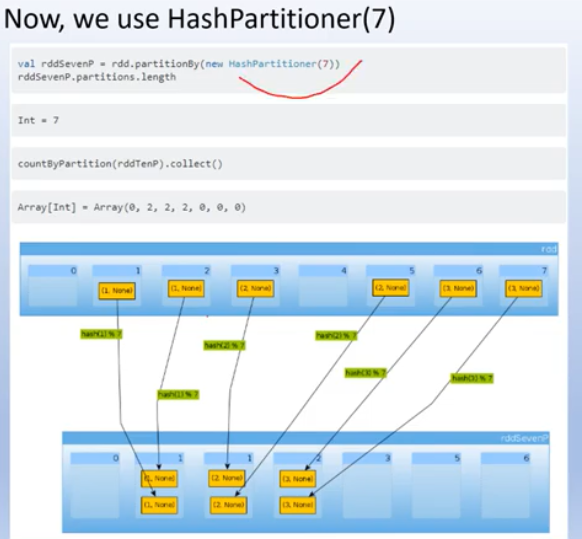
So fetch all partition and put it in one.

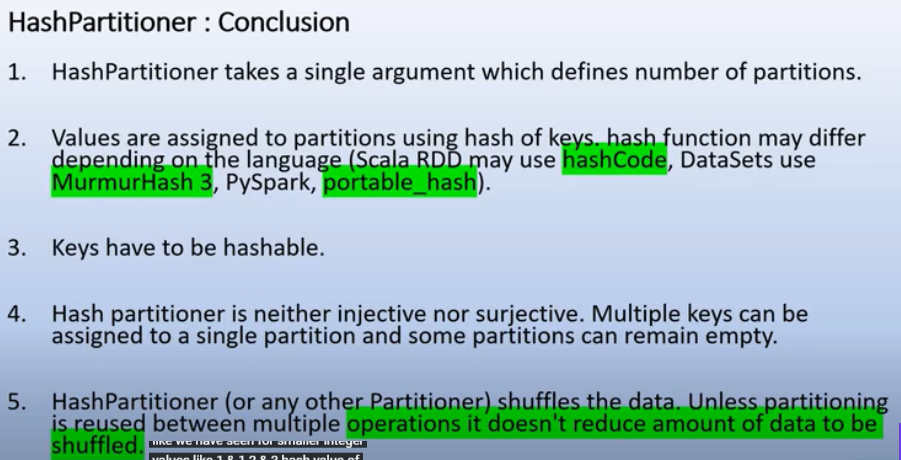


If we define hashpartitioner value as 2 , in first partion we have 2 values and in second partition we have 4 values, they are not evenly distributed.



This will done under the hood and elements are distributed , for key value 1 and 3 the hascode module value comes as 1 so these value goes to the same partition and for the key value 2 , we got hsahcode module value as 0.





Hash Partitioning: Example

Consider a Pair RDD, with keys [8, 96, 240, 400, 401, 800], and the desired number of partitions of 4.  
Furthermore, suppose that hashCode() is the identity (n.hash€ode() ==n).  
In this case, hash partitioning distributes the keys as follows among the partitions:

– partition 0: [8, 96, 240, 400, 800]

– partition 1: [401]

– partition 2: []

– partition 3: []

The result is a very unbalanced distribution which hurts performance.