SPARK SHARED VARIABLE: BROADCAST VARIABLE

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.

Apache Spark Shared Variable

① Broadcast Variable

② Accumulator

⇒ Broadcast Variable

⇒ Spark programs often need to access data that is not part of an RDD.

Example:

```
Val shared = Map(1→"a", 2→"e", 3→"i",
                 4→"o", 5→"u")

Val result = sc.parallelize(Array(1,2,3))
               .map(lookup(_))

assert(result.collect().toSet() === Set("a","e","i"))
```

⇒ In above program the variable lookup is serialized as a part of the closure passed to map().

⇒ It is not efficient way.

Reasons:

① Sending shared variable is closure is convenient, However.

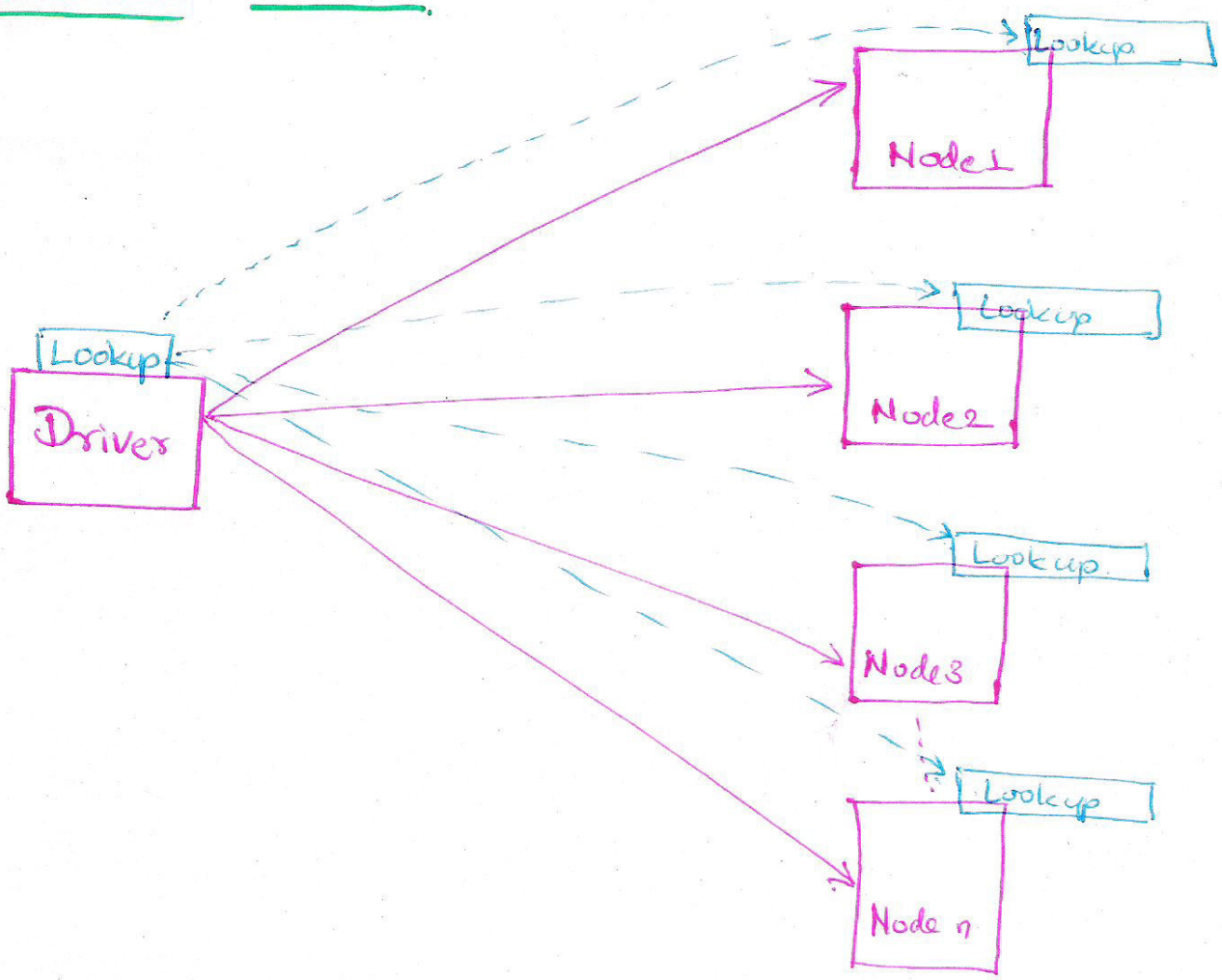
⇒ The default task launching mechanism is optimized for small task sizes.

⇒ You might in fact, use the same variable in multiple parallel operations, but spark will send it separately for each operation.

⇒ Assume in above program if lookup Array/table size is in MBs (Several Mega Bytes), then it is expensive to send from master alongside each task

⇒ Each call will send this variable again and again from master to task node, which will impact program time.

Broadcast Variable



⇒ Broadcast variable allows the program to efficiently send a large, read-only value to all the workers node for use in one or more Spark operations.

⇒ Generally such requirement appear in Machine Learning algorithm for large feature vector needs to be send across the cluster on all nodes.

=> A Broadcast Variable is simply an object of type

`Spark.broadcast.Broadcast[T]`

T => Datatype int, double, String etc.

=> T must be serializable.

=> To access the broadcast variable on node, you need to use value method, to fetch value..

=> Only once :- The value is sent to each node only once, using an efficient, Bit Torrent like Communication mechanism.

```
Val lookup: Broadcast[Map[Int, String]]
    = sc.broadcast(Map(1->a, 2->e, 3->i
                       4->o, 5->u))
Val result = sc.parallelize(Array(2, 1, 3)).
                 .map(lookup.value(_))

assert(result.collect().toSet === Set(a, e, i))
```

Program Using Broadcast Variable

⇒ As name suggest, broadcast variable are sent one way, from driver to task.

⇒ Read only:

⇒ There is no way to update Broadcast Variable and have the update propagated back to the driver, for that we need an Accumulator.

⇒ However, you can change the value of a broadcast variable, if it is not of primitive values e.g. int, boolean, double etc. or reference to an immutable object.

⇒ You can broadcast mutable objects as well.

⇒ If program sent mutable object as a broadcast variable, then it is upto him to maintain read-only condition.

Example:

```
Val lookup.BroadCast[Array["String]]
    = sc.broadcast(Array["A", "B", "C", "D"])
```

=> On one of the node, we do like this

```
Val lookupArray = lookup.value;
lookupArray(0) = "a" // new value
```

=> When above code is run on a worker node, that line will assign newvalue to the ~~first~~ element of lookupArray, only in the copy of the Array local to the worker node, running this code.

=> It will not change the contents of lookup.value on any of the other worker nodes.

=> Performance can further optimized by more efficient serialization mechanism.

=> As default java serialization is very slow, we can change it from below property.

spark.serializer e.g. Kryo