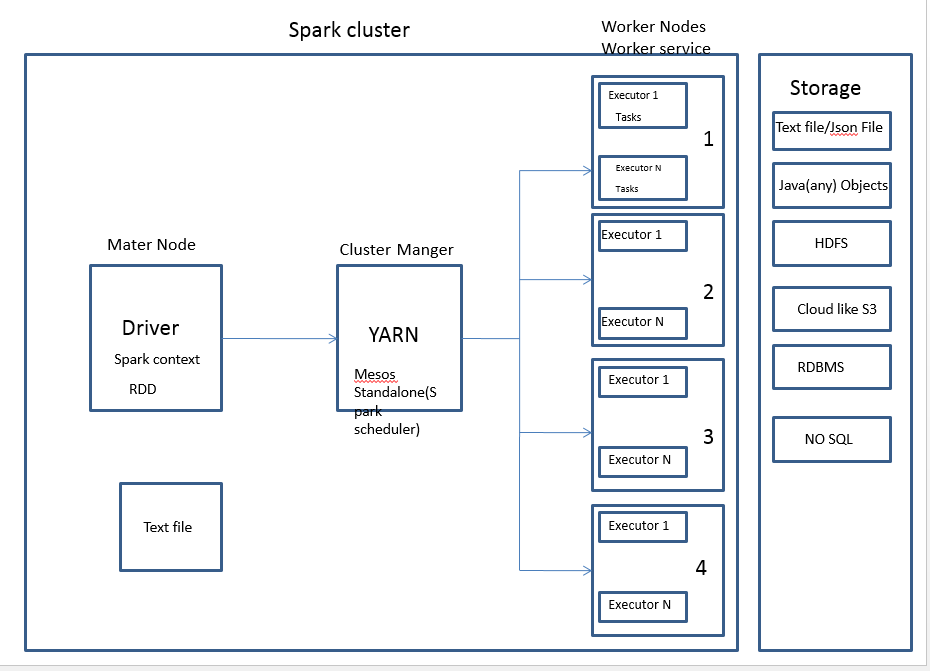
* **Spark key interview questions**

1. **What is the Difference between Spark andHadoop?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**



1. **Ask about there clusters?**
   1. **Which distribution**
   2. **Spark versoin**
   3. **Cluster size**
   4. **Number of executors**
2. **What is RDD?**
3. RDD is fundamental data structure of Spark.
4. It is an immutable distributed collection of objects.
5. RDD is divided into logical partitions, each partition id processed on different nodes of the cluster.
6. RDD is a fault-tolerant collection of elements those can be operated on in parallel.
7. **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 two ways to create RDDs

1. Text file:

Ex:

jsc.textFile("data/movietweets.csv")

1. **Parallelizing:** Creating RDDs from existing collection.

Ex:

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

1. **Loading external data**: Reading data stored in external storage system like HDFS,RDBMS,Local Storage.

From RDBMS we can load data to local collection and then convert it to RDD.

Ex:

JavaRDD<String> lines = sc.textFile("/path/to/README.md");

1. **Word count example in spark with java?**

**package com.journaldev.sparkdemo;**

**import org.apache.spark.SparkConf;**

**import org.apache.spark.api.java.JavaPairRDD;**

**import org.apache.spark.api.java.JavaRDD;**

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

**import scala.Tuple2;**

**import java.util.Arrays;**

**public class WordCounter {**

**private static void wordCount(String fileName) {**

**SparkConf sparkConf = new SparkConf().setMaster("local").setAppName("JD Word Counter");**

**JavaSparkContext sparkContext = new JavaSparkContext(sparkConf);**

**JavaRDD<String> inputFile = sparkContext.textFile(fileName);**

**JavaRDD<String> wordsFromFile = inputFile.flatMap(content -> Arrays.asList(content.split(" ")));**

**JavaPairRDD countData = wordsFromFile.mapToPair(t -> new Tuple2(t, 1)).reduceByKey((x, y) -> (int) x + (int) y);**

**countData.saveAsTextFile("CountData");**

**}**

**public static void main(String[] args) {**

**if (args.length == 0) {**

**System.out.println("No files provided.");**

**System.exit(0);**

**}**

**wordCount(args[0]);**

**}**

**}**

}

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.

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 is the relationship between job and action?**

One job is one action

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

Spark is fault tolerant. So failed tasks will be reschedules

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

### 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 submitsthe 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?

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 beshuffled 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.

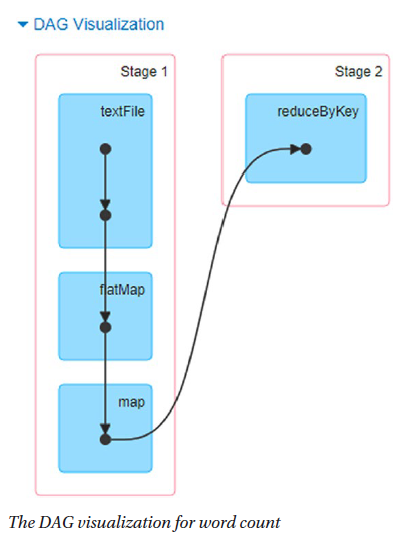
He never forgets

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.



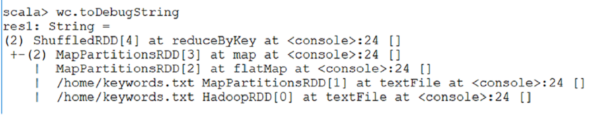
### How Spark Achieves Fault Tolerance Through DAG?

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?

To view the lineage, use *toDebugString*

****

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

Each RDD will have their own lineage graph.

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

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).

### 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 are the storage levels supported in spark?**

|  |  |
| --- | --- |
| **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 on 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 |

**What is the difference between map and flat map?**

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

One action is one job.

1. **How do you calculate executor memory?**

[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**

1. First 1 core and 1 GB is needed for OS and Hadoop Daemons, so available are 15 cores, 63 GB RAM for each node.
2. 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.

1. How to choose number of Executors.

Because we choose 5 core 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 forAM (Application Manager) in YARN.

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

1. 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.

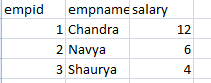
Number of executors requested in each round increased exponentially from the previous round.For instance, an application will add 1 executor in the first round, and then 2, 4, 8 and so on executors in the subsequent rounds. At a specific point, the above max comes into picture

* 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.

1. **Say you have table**

****

**You need to add new column to it if salary is between 1 to 5 then it is grade0, if it is between 5 to 10 then grade 2 if it is above 10 then it is grade 3**

1. **What is the difference between dataframe and dataset?**
2. **What is the difference between reduce by key and group by key?**
3. **What is the difference between repartition and coalesce?**
4. **How will you calculate moving average and cumulative sum in spark SQL?**

Using window functions.

We can easily calculate moving average or cumulative sum or we can also reference a value in the previous column

1. **How can you reference value of previous row of a table in spark?**

Using Window functions.

Note: Window functions are complementary to existing Data Frame operations.

1. **What is data skew?IMP**

Uneven distribution of data in a distributed environment is called skew.

1. **How to solve data skew issue in general?IMP**
2. When we setup cluster we need to make sure that all nodes in the clustershould be homogeneous that is all nodes in the cluster should have same configuration in terms of cpu, memory and disk
3. Should try to dedicate Nodes to one service(team) if Node is shared then again we might face same issue
4. There will be techniques provided in each of the technologies like spark sql, hive etc we need to know and apply it properly