How can the Spark Web UI aid in understanding Spark execution?

Authored by

stats writer

June 24, 2024

The Spark Web UI is a tool that provides a graphical representation of the execution process of Spark applications. It offers real-time monitoring of the various stages and tasks involved in the execution, allowing users to gain a deeper understanding of the underlying processes. This aids in troubleshooting and optimizing the performance of Spark applications. Additionally, the Web UI provides detailed metrics and statistics, making it easier to identify bottlenecks and improve overall efficiency. Overall, the Spark Web UI plays a crucial role in aiding users in comprehending the intricate execution of Spark, leading to improved productivity and better utilization of resources.

Apache Spark provides a suite of Web UI/User Interfaces (Jobs, Stages, Tasks, Storage, Environment, Executors, and SQL) to monitor the status of your Spark/PySpark application, resource consumption of Spark cluster, and Spark configurations.

To better understand how Spark executes the Spark/PySpark Jobs, this set of user interfaces comes in handy. In this article, I will run a small application and explain how Spark executes this by using different sections in Spark Web UI.

Before going into Spark UI first, learn about these two concepts.

Let me give a small brief on those two, Your application code is the set of instructions that instructs the driver to do a Spark Job and lets the driver decide how to achieve it with the help of executors.

Instructions to the driver are called Transformations and action will trigger the execution.

I had written a small application that does transformation and action.

```
//Transformation
val rawDF = spark.read //job(0) : Read
              .option("inferSchema", "true") //job(1) : InferSchema
              .option("header", "true")
              .csv( path = "data/survey.csv")
//Action
rawDF.count()// job(2): Get Count
```

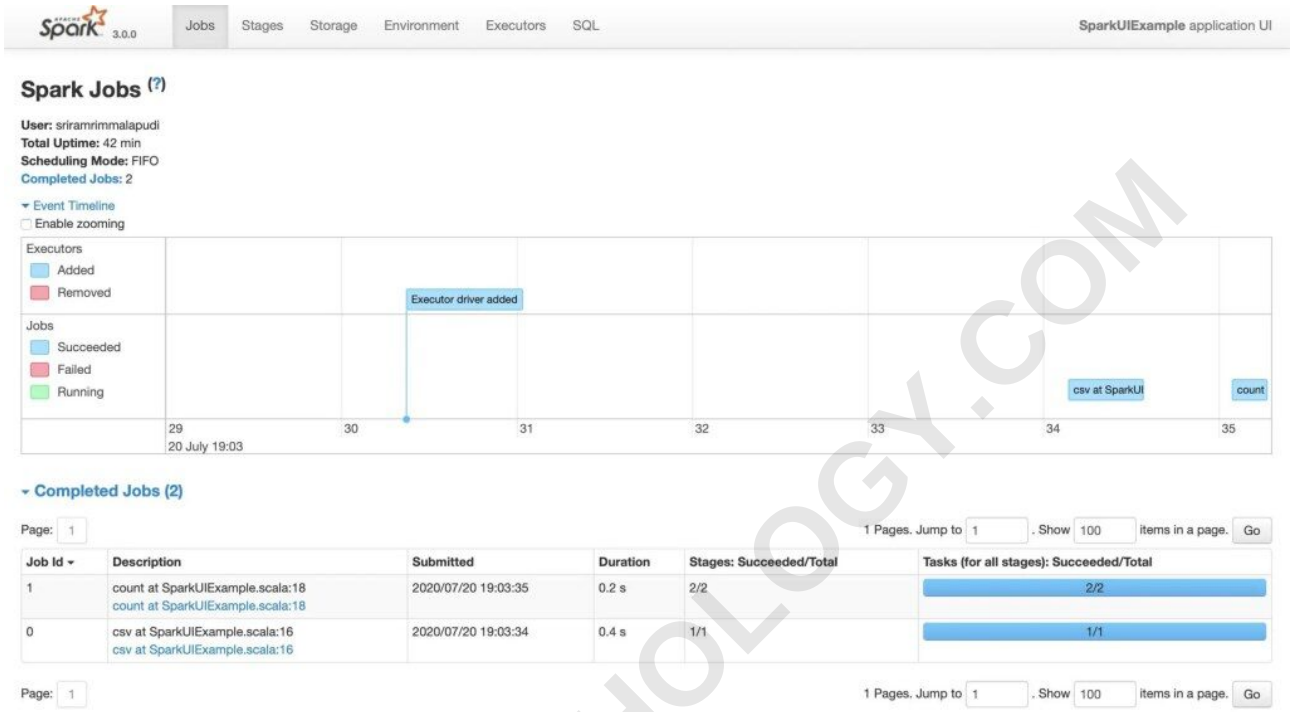
Application Code

Here we are creating a DataFrame by reading a .csv file and checking the count of the DataFrame. Let's understand how an application gets projected in Spark UI

Spark UI is separated into the below tabs.

Spark JobsStagesTasksStorageEnvironmentExecutorsSQL

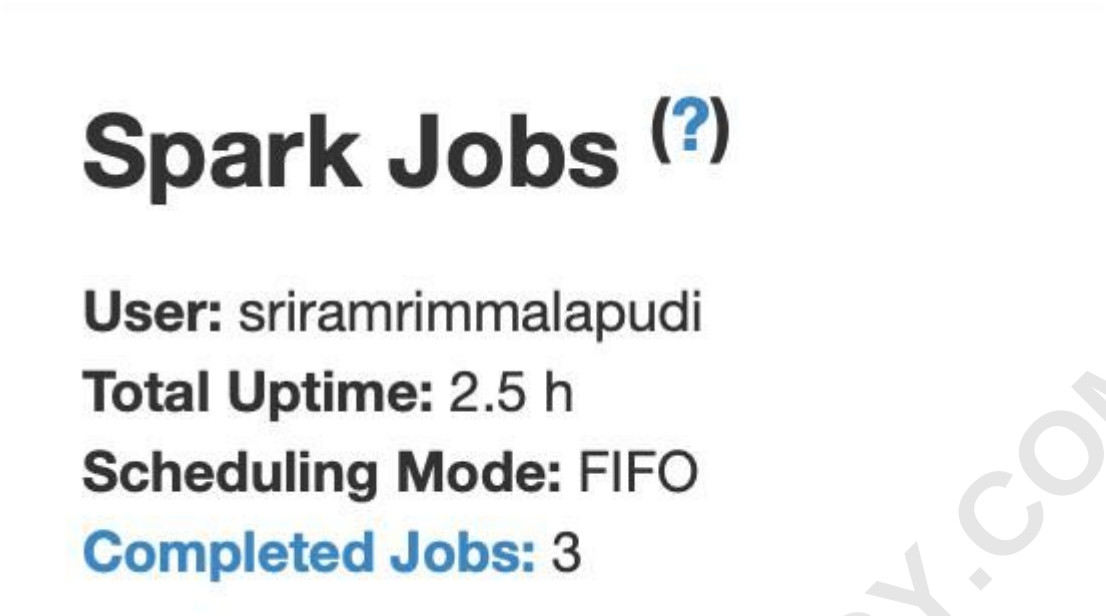
If you are running the Spark application locally, Spark UI can be accessed using the http://localhost:4040/ . Spark UI by default runs on port 4040 and below are some of the additional UI's that would be helpful to track the Spark application.



Spark Web UI

Note: To access these URLs, the Spark application should be in a running state. If you wanted to access this URL regardless of your Spark application status and wanted to access Spark UI all the time, you would need to start Spark History server.

1. Spark Jobs Tab



Jobs tab

The details that I want you to be aware of under the jobs section are **Schedulingmode**, the **number of Spark Jobs**, the **number of stages** it has, and **Description** in your Spark job.

1.1 Scheduling Mode

We have three Scheduling modes.

Standalone mode **YARN** mode **Mesos**

Spark Scheduling tab

As I was running on a local machine, I tried using `Standalone mode`

1.2 Number of Spark Jobs:

Always keep in mind, that the number of Spark jobs is equal to the number of actions in the application and each Spark job should have at least one Stage.
In our above application, we have performed 3 Spark jobs (0,1,2)

So if we look at the fig it clearly shows 3 Spark jobs result of 3 actions.

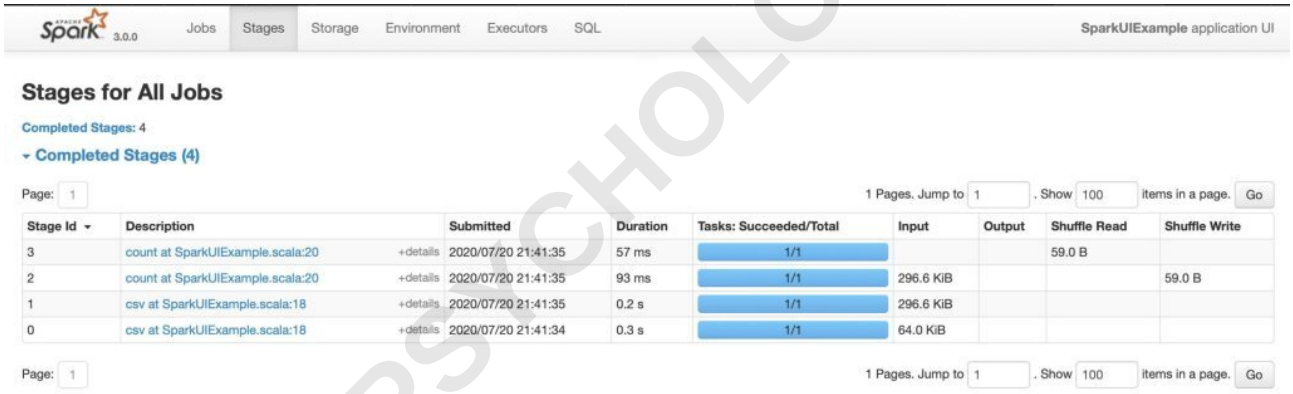
1.3 Number of Stages

Each Wide Transformation results in a separate Numberof Stages. In our case, Spark job0 and Spark job1 have individual single stages but when it comes to Spark job 3 we can see two stages that are because of the partition of data. Data is partitioned into two files by default.

1.4 Description

Description links the complete details of the associated SparkJob like Spark Job Status, DAG Visualization, Completed Stages
I have explained the description part in the coming part.

2. Stages Tab



Spark Stage Tab

We can navigate into the Stage Tab in two ways.

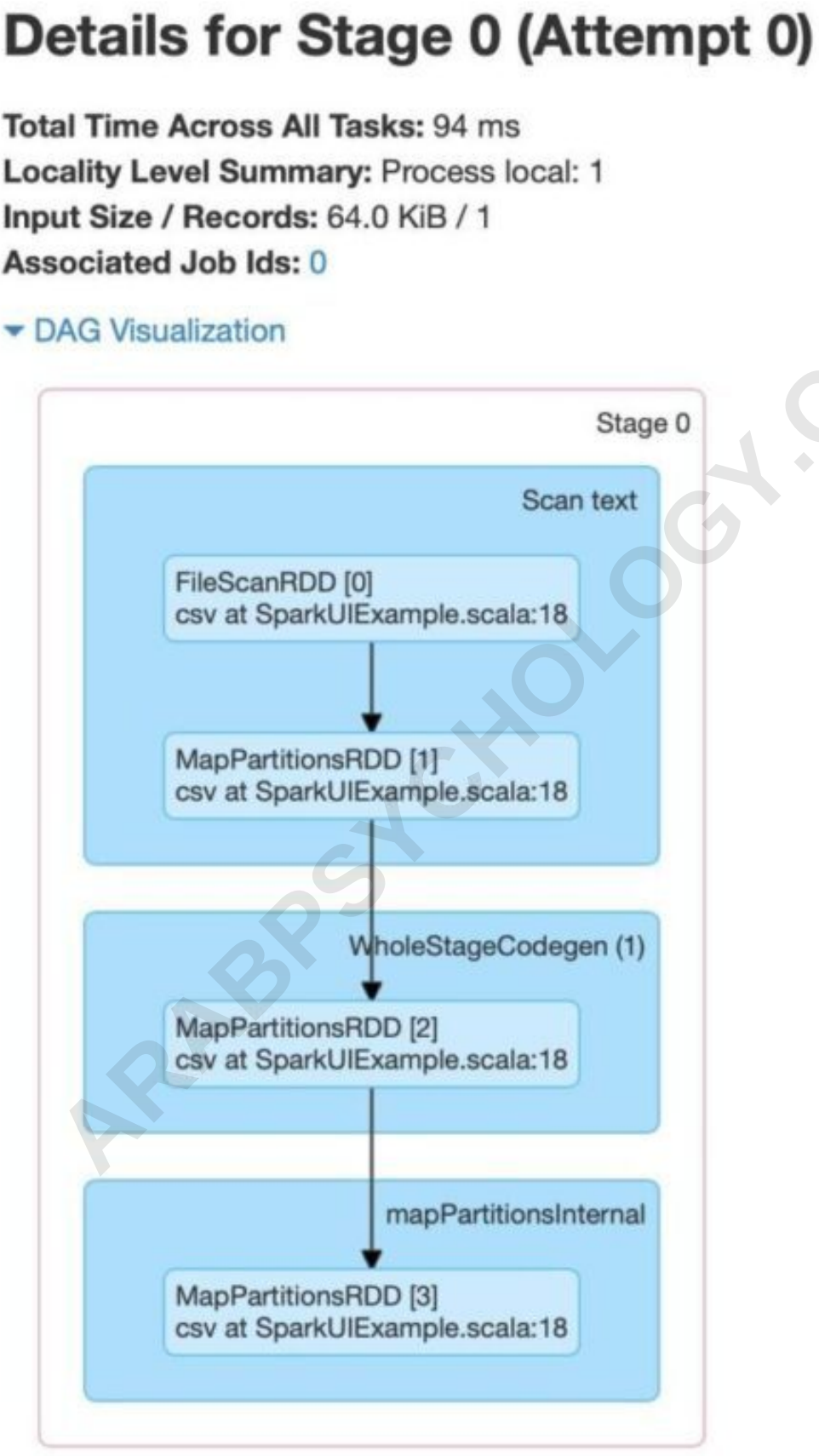
Select the Description of the respective Spark job (Shows stages only for the Spark job opted)On the top of the Spark Job tab select the Stages option (Shows all stages in the Application)

In our application, we have a total of **4 *Stages***.

The Stage tab displays a summary page that shows the current state of all stages of all Spark jobs in the Spark application

The number of tasks you can see in each stage is the number of partitions that Spark is going to work on and each task inside a stage is the same work that will be done by Spark but on a different

partition of data.



Stage 0

Stage detail

Details of the stage showcase the Directed Acyclic Graph (DAG) of this stage, where vertices represent the RDDs or DataFrame and edges represent an operation to be applied.

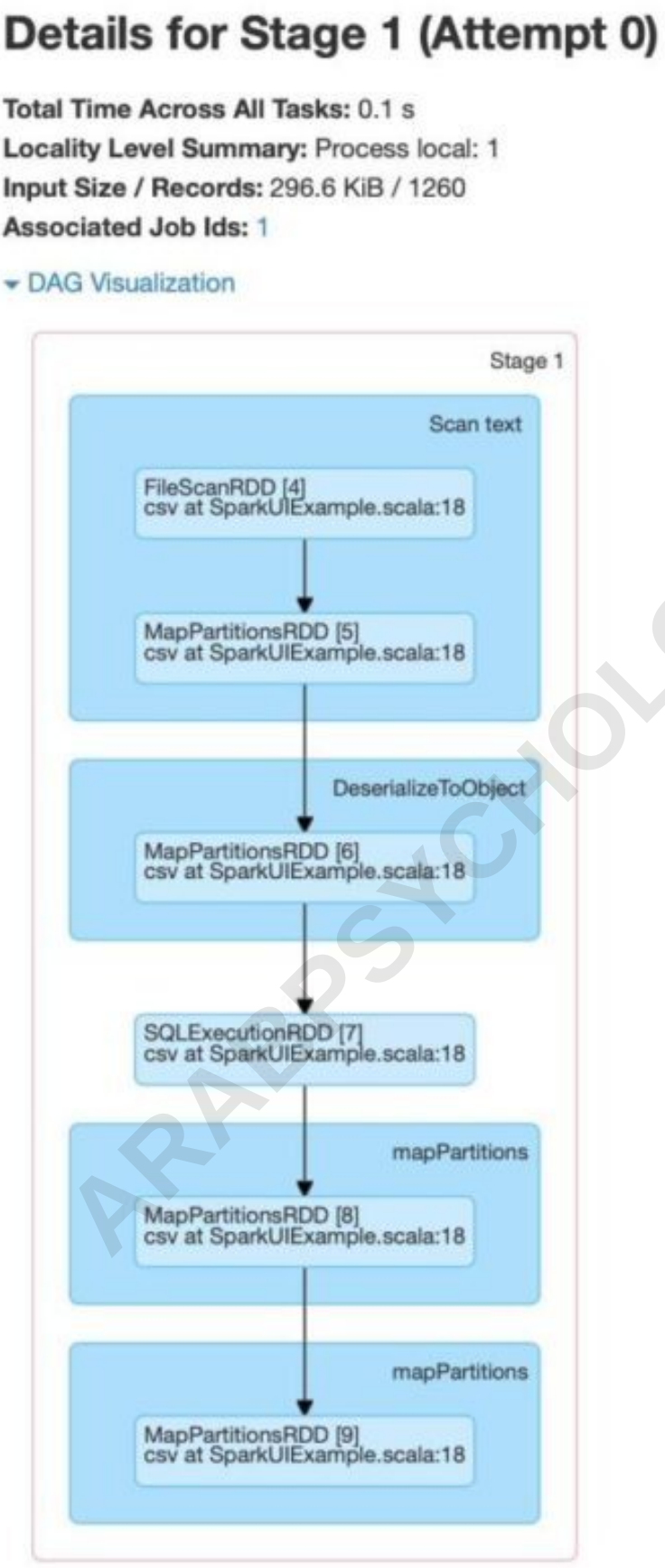
let us analyze operations in Stages
Operations in Stage0 are
1.FileScanRDD
2.MapPartitionsRDD

FileScanRDD

FileScan represents reading the data from a file.
It gives FilePartitions that are custom RDD partitions with PartitionedFiles (file blocks)
In our scenario, the *CSV file is read*

MapPartitionsRDD

MapPartitionsRDD will be created when you use map Partition transformation



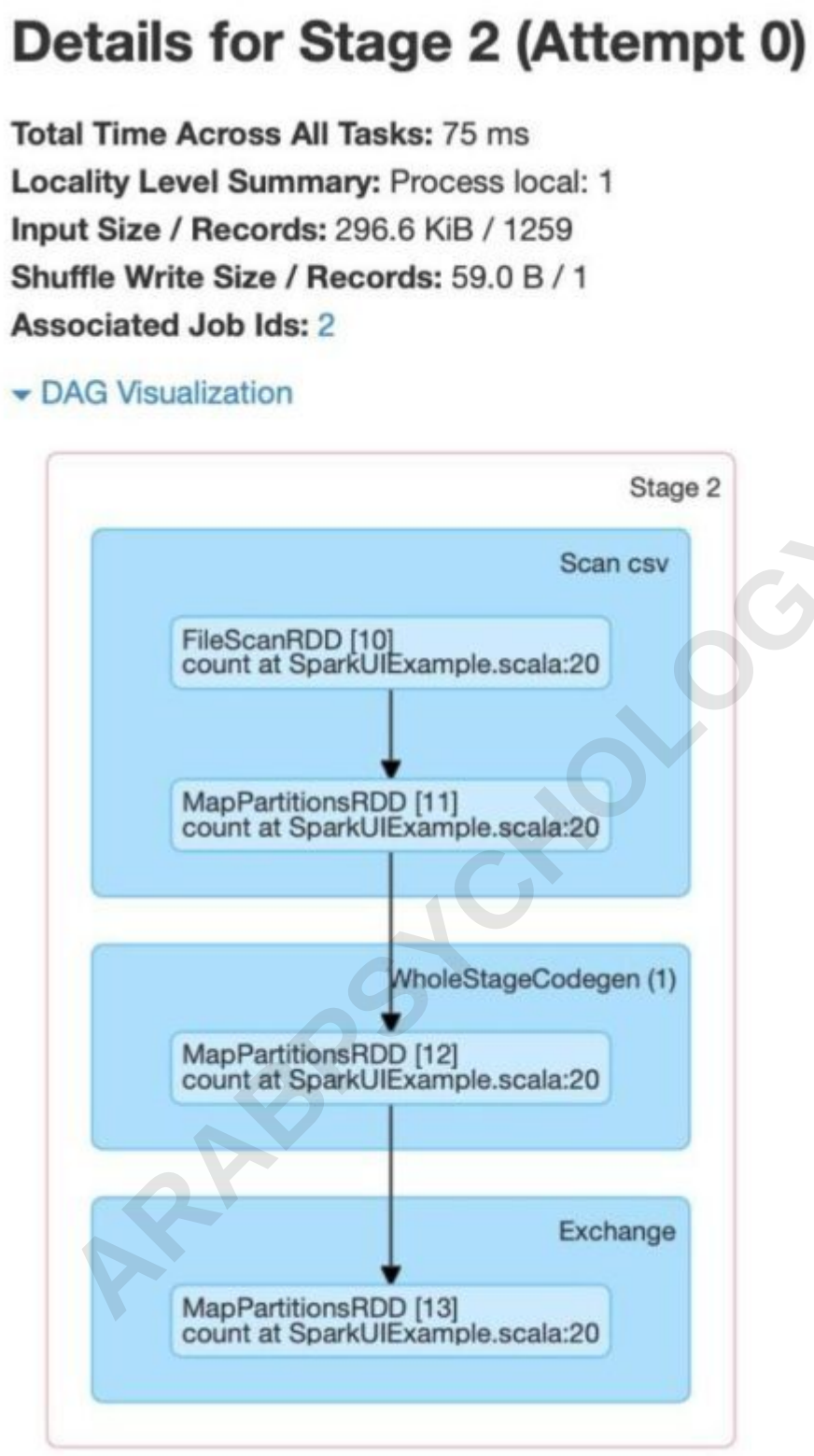
Stage1

Operation in Stage(1) are
1.FileScanRDD
2.MapPartitionsRDD
3.SQLExecutionRDD

As File Scan and `MapPartitionsRDD` is already explained, let us look at `SQLExecutionRDD`

SQLExecutionRDD

`SQLExecutionRDD` is Spark property that is used to track multiple Spark jobs that should all together constitute a single structured query execution.



Stage 2

Operation in Stage(2) and Stage(3) are

1.FileScanRDD

2.MapPartitionsRDD

3.WholeStageCodegen
4.Exchange

Wholestagecodegen

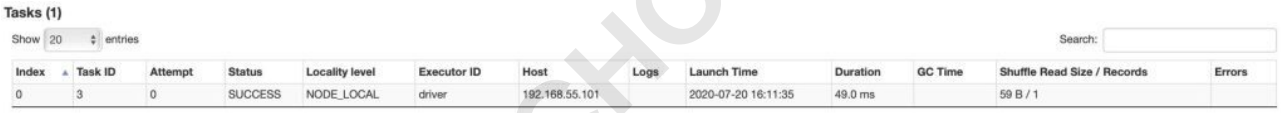
A physical query optimizer in Spark SQL that fuses multiple physical operators

Exchange

Exchange is performed because of the COUNT method.
As data is divided into partitions and shared among executors, to get count there should be adding of the count of from individual partition.

Represents the shuffle i.e data movement across the cluster(Executors).
It is the most expensive operation and if number of partitions is more exchange of data between executors will also be more.

3. Tasks


Tasks (1)
Show 20 entries Search:

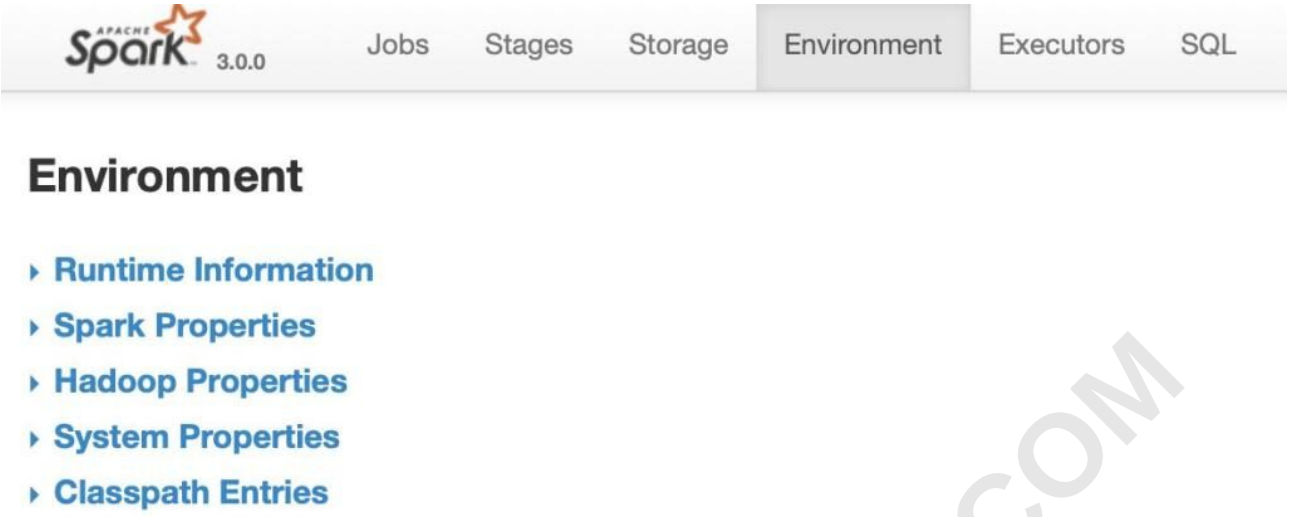
Index	Task ID	Attempt	Status	Locality level	Executor ID	Host	Logs	Launch Time	Duration	GC Time	Shuffle Read Size / Records	Errors
0	3	0	SUCCESS	NODE_LOCAL	driver	192.168.55.101		2020-07-20 16:11:35	49.0 ms		59 B / 1	

Tasks are located at the bottom space in the respective stage.
Key things to look task page are:
1. Input Size - Input for the Stage
2. Shuffle Write-Output is the stage written.

4. Storage

The Storage tab displays the persisted RDDs and DataFrames, if any, in the application. The summary page shows the storage levels, sizes, and partitions of all RDDs, and the details page shows the sizes and executors for all partitions in an RDD or DataFrame.

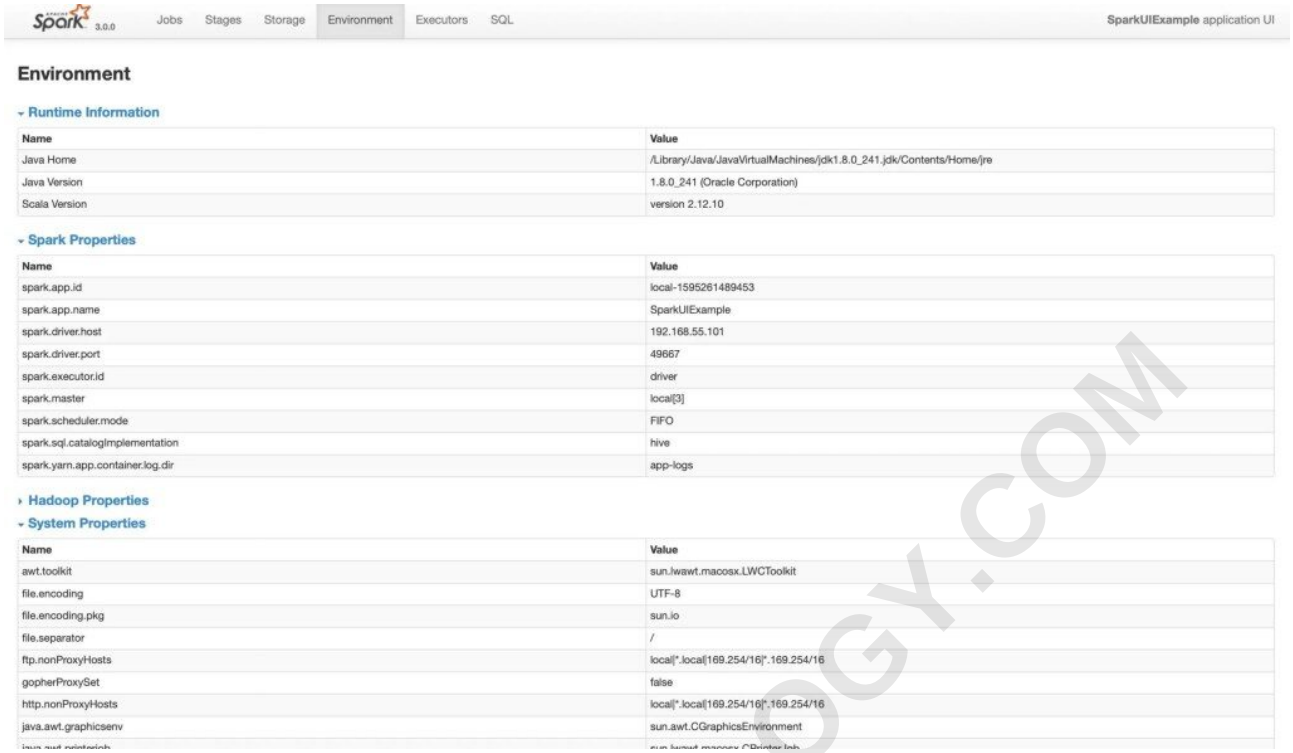
5. Environment Tab



Spark Environment Tab

This environment page has five parts. It is a useful place to check whether your properties have been set correctly.

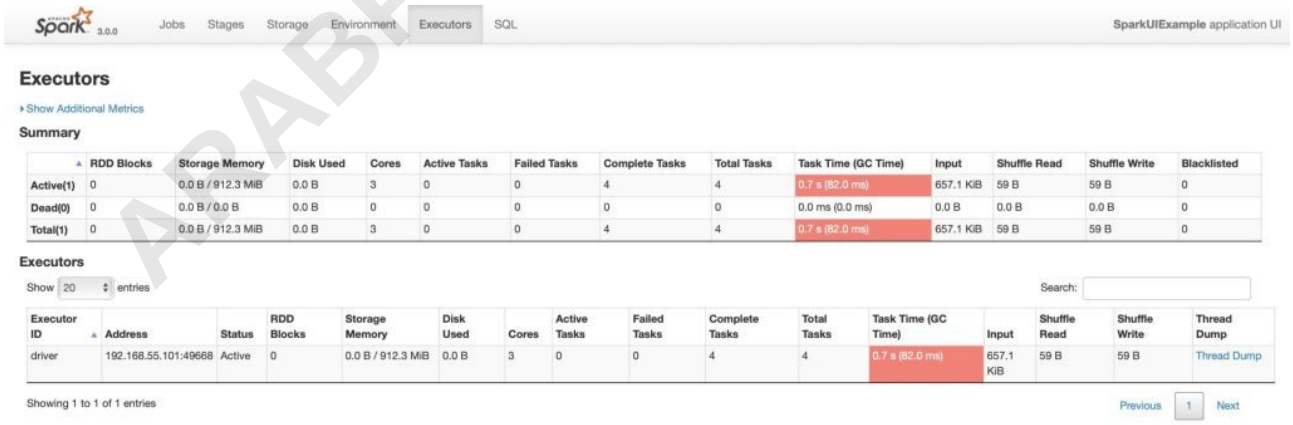
Runtime Information: simply contains the runtime properties like versions of Java and Scala.**Spark Properties**: lists the application properties like 'spark.app.name' and '`spark.driver.memory`'.**Hadoop Properties**: displays properties relative to Hadoop and YARN. **Note**: Properties like '`spark.hadoop`' are shown not in this part but in 'Spark Properties'.**System Properties**: shows more details about the JVM.**Classpath Entries**: lists the classes loaded from different sources, which is very useful to resolve class conflicts.



Spark Environment properties

The Environment tab displays the values for the different environment and configuration variables, including JVM, Spark, and system properties.

6. Executors Tab

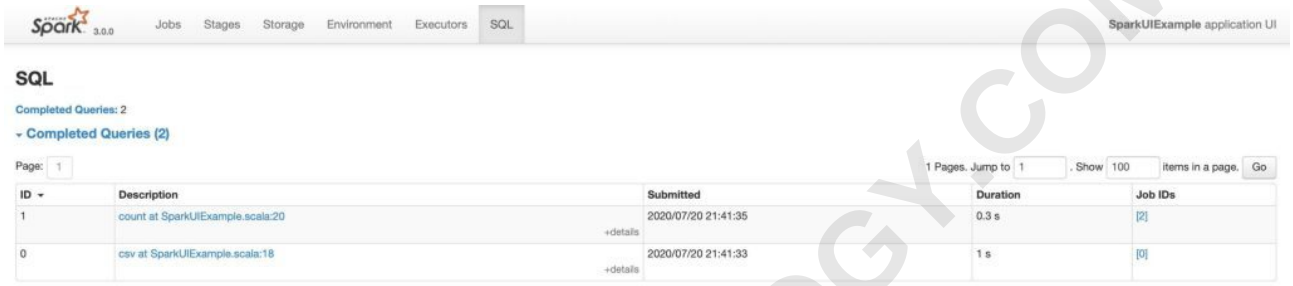


The Executors tab displays summary information about the executors that were created for the application, including memory and disk usage and task and shuffle information. The Storage Memory column shows the amount of memory used and reserved for caching data.

The Executors tab provides not only resource information like the amount of memory, disk, and cores used by each executor but also performance information.

In Executors
Number of cores = 3 as I gave master as local with 3 threads
Number of tasks = 4

7. SQL Tab



If the application executes Spark SQL queries then the SQL tab displays information, such as the duration, Spark jobs, and physical and logical plans for the queries.

In our application, we performed read and count operations on files and DataFrame. So both read and count are listed SQL Tab

Some of the resources are gathered from https://spark.apache.org/ thanks for the information.

"..... Keep learning and keep growing....."

Related Articles