Spark Developer Training - 3 Days

Manaranjan Pradhan
manaranjan@enablecloud.com

Lab: Monitoring and Debugging Spark Applications

Things to learn

- Use of HDFS UI
- Use of Resource Manager UI
- Use of Appication Monitoring UI
- Understand Execution Workflow

In [2]:

```
sc
```

Out[2]:

```
<pyspark.context.SparkContext at 0x7f639419a390>
```

Open http://hadooplab.bigdataleap.com:50070/ (http://hadooplab.bigdataleap.com:50070/) on your browser for HDFS UI

This Spark Application is running on YARN. So, open the YARN Resource manager UI and verify if the application is running

- To open resource manager web UI, enter http://hadooplab.bigdataleap.com:8088/ (http://hadooplab.bigdataleap.com:8088/)

Open the http://hadooplab.bigdataleap.com:4040/ (http://hadooplab.bigdataleap.com:4040/) (or subsequent port) on the VM in firefox browser for application monitoring UI

Initialize SQLContext from SparkContext

In [3]:

```
from pyspark.sql import SQLContext
sqlContext = SQLContext(sc)
```

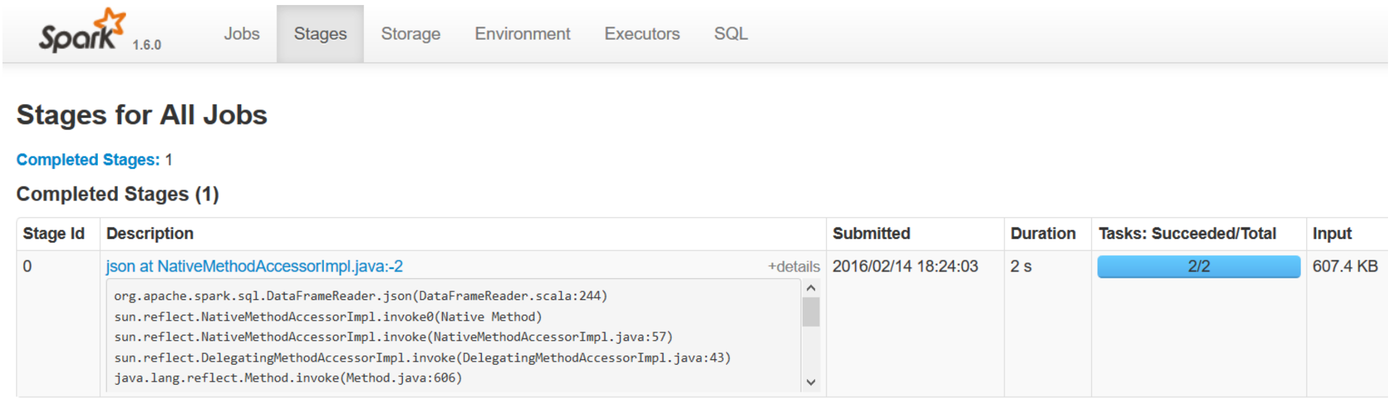
Reading JSON file from HDFS

In [4]:

```
## Read the json file from HDFS
txns = sqlContext.read.json( "hdfs://sparklab.awesomestats.in/sparklab/txnjsonsmall")
```

Go to the stages tab

- The data is read



Stages for All Jobs

Completed Stages: 1

Completed Stages (1)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input
0	json at NativeMethodAccessorImpl.java:-2 +details	2016/02/14 18:24:03	2 s	2/2	607.4 KB

```
org.apache.spark.sql.DataFrameReader.json(DataFrameReader.scala:244)
sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:57)
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
java.lang.reflect.Method.invoke(Method.java:606)
```

Display the first 10 records

In [5]:

```
txns.show( 10 )
```

```
+------------+------------------+----------+--------------------+-----
--------+---------+---------+--------+
|CashOrCredit|      creditCardNo|customerNo|           lineItems| merc
hantCity|    state|    tDate|   txnNo|
+------------+------------------+----------+--------------------+-----
--------+---------+---------+--------+
|      credit|4971-xxxx-xxxx-5769|   4004819|[[015.82,Team Spo...|  Bro
wnsville|    Texas|06-27-2011|00000000|
|      credit|3787-xxxx-xxxx-6017|   4003459|[[089.28,Water Sp...|
Houston|    Texas|02-07-2011|00000001|
|      credit|5951-xxxx-xxxx-4036|   4009112|[[067.51,Exercise...|
Eugene|   Oregon|03-02-2011|00000002|
|      credit|3793-xxxx-xxxx-3180|   4009376|[[043.38,Water Sp...|
Paterson|New Jersey|01-23-2011|00000003|
|      credit|3913-xxxx-xxxx-4556|   4006758|[[193.65,Outdoor ...|
Gresham|   Oregon|05-07-2011|00000004|
|      credit|4629-xxxx-xxxx-3692|   4000951|[[104.47,Exercise...|   De
s Moines|     Iowa|12-07-2011|00000005|
|      credit|4032-xxxx-xxxx-1996|   4002494|[[093.97,Jumping,...|  St.
Louis  |  Missouri|05-02-2011|00000006|
|      credit|3551-xxxx-xxxx-0696|   4000599|[[197.33,Exercise...|
Phoenix|  Arizona|06-02-2011|00000007|
|      credit|3282-xxxx-xxxx-5190|   4007057|[[128.98,Winter S...|Overl
and Park|   Kansas|03-06-2011|00000008|
|      credit|4621-xxxx-xxxx-9258|   4005366|[[074.57,Water Sp...|
Fremont|California|06-22-2011|00000009|
+------------+------------------+----------+--------------------+-----
--------+---------+---------+--------+
only showing top 10 rows
```

In [6]:

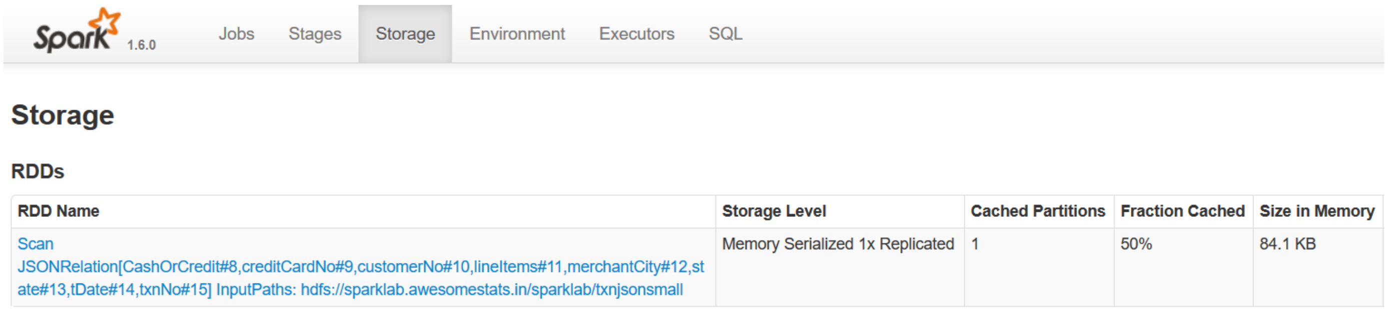
```
txns.persist()
```

Out[6]:

```
DataFrame[CashOrCredit: string, creditCardNo: string, customerNo: strin
g, lineItems: array<struct<amount:string,category:string,product:string
>>, merchantCity: string, state: string, tDate: string, txnNo: string]
```

Check the storage. The RDD should have been created.

- Check the size of the RDD. Number of partitions and percentage of cache.



Storage

RDDs

RDD Name	Storage Level	Cached Partitions	Fraction Cached	Size in Memory
Scan JSONRelation[CashOrCredit#8,creditCardNo#9,customerNo#10,lineItems#11,merchantCity#12,state#13,tDate#14,txnNo#15] InputPaths: hdfs://sparklab.awesomestats.in/sparklab/txnjsonsmall	Memory Serialized 1x Replicated	1	50%	84.1 KB

The line items are nested structure in each transaction. Display the lineitems of first 5 transactions

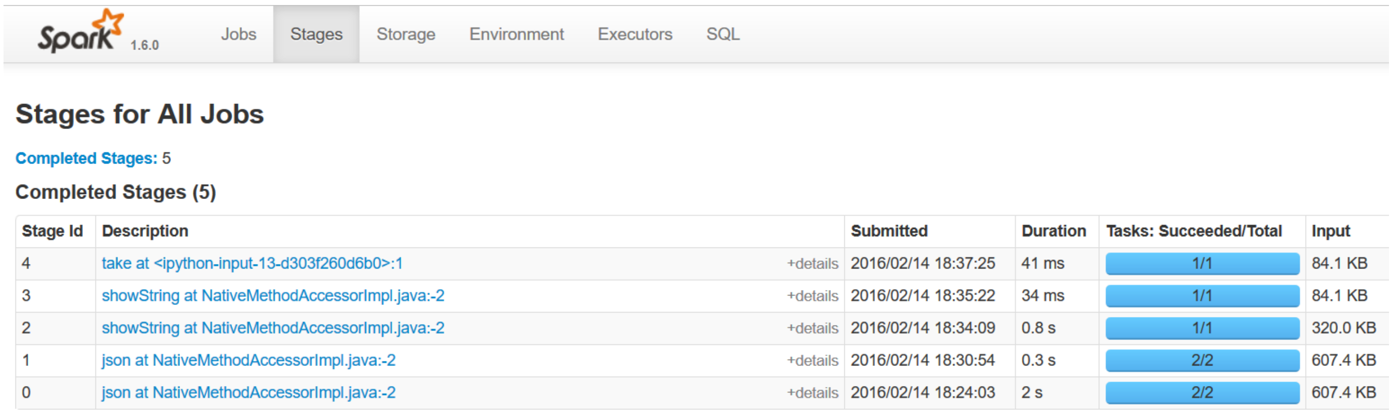
In [7]:

```
txns.select( "lineItems" ).take( 5 )
```

Out[7]:

```
[Row(lineItems=[Row(amount='015.82', category='Team Sports', product='C
heerleading'), Row(amount='086.47', category='Water Sports', product='W
hitewater Rafting'), Row(amount='063.08', category='Exercise & Fitnes
s', product='Gym Mats'), Row(amount='068.80', category='Exercise & Fitn
ess', product='Weightlifting Machine Accessories'), Row(amount='092.4
9', category='Team Sports', product='Lacrosse'), Row(amount='083.92', c
ategory='Outdoor Recreation', product='Lawn Games')]),
 Row(lineItems=[Row(amount='089.28', category='Water Sports', product
='Water Tubing'), Row(amount='042.38', category='Water Sports', product
='Surfing'), Row(amount='062.80', category='Team Sports', product='Chee
rleading')]),
 Row(lineItems=[Row(amount='067.51', category='Exercise & Fitness', pro
duct='Exercise Bands'), Row(amount='154.57', category='Team Sports', pr
oduct='Rugby'), Row(amount='100.18', category='Outdoor Recreation', pro
duct='Skateboarding'), Row(amount='190.52', category='Exercise & Fitnes
s', product='Foam Rollers'), Row(amount='054.35', category='Water Sport
s', product='Kitesurfing')]),
 Row(lineItems=[Row(amount='043.38', category='Water Sports', product
='Boating'), Row(amount='106.27', category='Team Sports', product='Rugb
y'), Row(amount='164.86', category='Combat Sports', product='Fencing'),
Row(amount='164.94', category='Racquet Sports', product='Tennis'), Row
(amount='007.36', category='Exercise & Fitness', product='Gym Mats'), R
ow(amount='110.56', category='Outdoor Recreation', product='Skateboardi
ng')]),
 Row(lineItems=[Row(amount='193.65', category='Outdoor Recreation', pro
duct='Deck Shuffleboard'), Row(amount='135.98', category='Winter Sport
s', product='Snowshoeing'), Row(amount='063.27', category='Racquet Spor
ts', product='Racquetball'), Row(amount='151.53', category='Dancing', p
roduct='Ballet Bars'), Row(amount='088.69', category='Gymnastics', prod
uct='Balance Beams'), Row(amount='070.75', category='Outdoor Play Equip
ment', product='Swing Sets')])]
```

Check the next stage for select statement



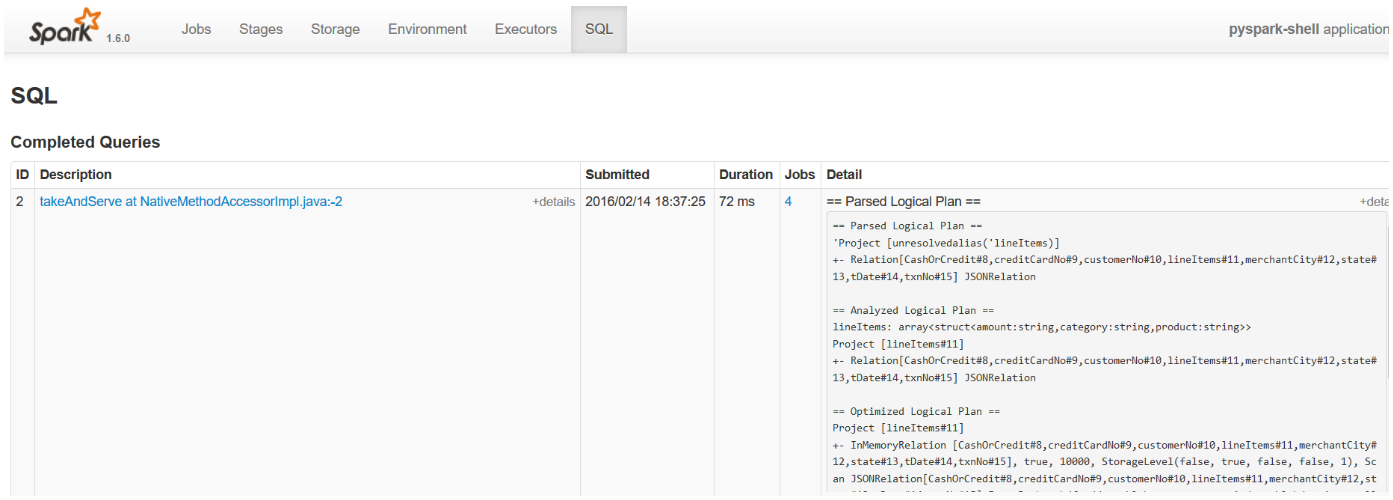
Stages for All Jobs

Completed Stages: 5

Completed Stages (5)

Stage Id	Description		Submitted	Duration	Tasks: Succeeded/Total	Input
4	take at <ipython-input-13-d303f260d6b0>:1	+details	2016/02/14 18:37:25	41 ms	1/1	84.1 KB
3	showString at NativeMethodAccessorImpl.java:-2	+details	2016/02/14 18:35:22	34 ms	1/1	84.1 KB
2	showString at NativeMethodAccessorImpl.java:-2	+details	2016/02/14 18:34:09	0.8 s	1/1	320.0 KB
1	json at NativeMethodAccessorImpl.java:-2	+details	2016/02/14 18:30:54	0.3 s	2/2	607.4 KB
0	json at NativeMethodAccessorImpl.java:-2	+details	2016/02/14 18:24:03	2 s	2/2	607.4 KB

Also the SQL section for detailed logical plan



SQL

Completed Queries

ID	Description		Submitted	Duration	Jobs	Detail
2	takeAndServe at NativeMethodAccessorImpl.java:-2	+details	2016/02/14 18:37:25	72 ms	4	== Parsed Logical Plan == +deta

```
== Parsed Logical Plan ==
'Project [unresolvedalias('lineItems)]
+- Relation[CashOrCredit#8,creditCardNo#9,customerNo#10,lineItems#11,merchantCity#12,state#
13,tDate#14,txnNo#15] JSONRelation

== Analyzed Logical Plan ==
lineItems: array<struct<amount:string,category:string,product:string>>
Project [lineItems#11]
+- Relation[CashOrCredit#8,creditCardNo#9,customerNo#10,lineItems#11,merchantCity#12,state#
13,tDate#14,txnNo#15] JSONRelation

== Optimized Logical Plan ==
Project [lineItems#11]
+- InMemoryRelation [CashOrCredit#8,creditCardNo#9,customerNo#10,lineItems#11,merchantCity#
12,state#13,tDate#14,txnNo#15], true, 10000, StorageLevel(false, true, false, false, 1), Sc
an JSONRelation[CashOrCredit#8,creditCardNo#9,customerNo#10,lineItems#11,merchantCity#12,st
```

Import the explode function to flatten the records

In [8]:

```
from pyspark.sql.functions import *
```

In [9]:

```
## Explode and flatten the nested structure into a set of columns
txns_new = txns.select( 'txnNo', 'tDate', 'customerNo', 'merchantCity', 'state', 'ite
                       'item.product', 'item.amount',
                       explode( txns.lineItems ).alias( 'item' ) ).drop( 'item')
```

In [10]:

```
# Show 10 records
txns_new.show( 10 )
```

```
+--------+----------+----------+------------+------+------------------+
--------------------+------+
|   txnNo|     tDate|customerNo|merchantCity| state|          category|
product|amount|
+--------+----------+----------+------------+------+------------------+
--------------------+------+
|00000000|06-27-2011|   4004819| Brownsville| Texas|       Team Sports|
Cheerleading|015.82|
|00000000|06-27-2011|   4004819| Brownsville| Texas|      Water Sports|
Whitewater Rafting|086.47|
|00000000|06-27-2011|   4004819| Brownsville| Texas|Exercise & Fitness|
Gym Mats|063.08|
|00000000|06-27-2011|   4004819| Brownsville| Texas|Exercise & Fitness|
Weightlifting Mac...|068.80|
|00000000|06-27-2011|   4004819| Brownsville| Texas|       Team Sports|
Lacrosse|092.49|
|00000000|06-27-2011|   4004819| Brownsville| Texas|Outdoor Recreation|
Lawn Games|083.92|
|00000001|02-07-2011|   4003459|     Houston| Texas|      Water Sports|
Water Tubing|089.28|
|00000001|02-07-2011|   4003459|     Houston| Texas|      Water Sports|
Surfing|042.38|
|00000001|02-07-2011|   4003459|     Houston| Texas|       Team Sports|
Cheerleading|062.80|
|00000002|03-02-2011|   4009112|      Eugene|Oregon|Exercise & Fitness|
Exercise Bands|067.51|
+--------+----------+----------+------------+------+------------------+
--------------------+------+
only showing top 10 rows
```

Register the new table as temporary (in memory) table, so that we can run SQL Queries on it

In [11]:

```
# Register the dataframe as an temporary sql table into memory.. so that we can go an
txns_new.registerTempTable("txnrecords")
```

Find revenue generated by state and product

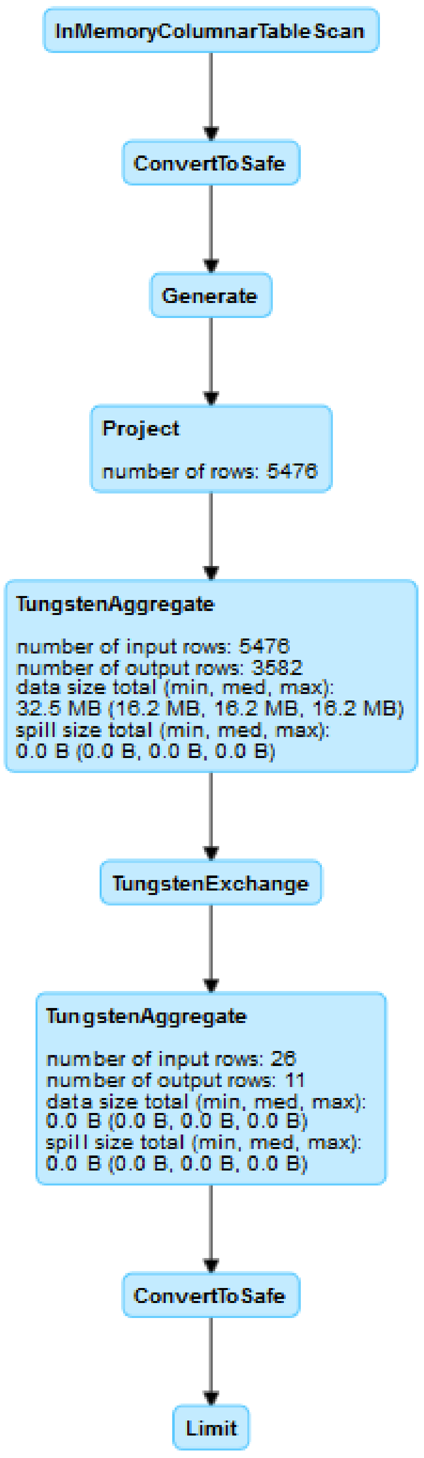
In [12]:

```
revenue_by_state = sqlContext.sql( "select state, product, sum( amount ) as total fro
In [13]:
```

```
# show the first 10 records
revenue_by_state.show( 10 )
```

```
+-------------+--------------------+------------------+
|        state|             product|             total|
+-------------+--------------------+------------------+
|       Oregon|               Rugby|             261.4|
|        Texas|          Parachutes|            706.58|
|       Oregon|Scuba Diving & Sn...|            264.55|
|         Utah|           Wrestling|207.79000000000002|
|     Kentucky|          Bobsledding|232.84999999999997|
|      Florida|        Foam Rollers|            387.27|
|Massachusetts|          Air Hockey|            120.66|
|      Alabama|          Windsurfing|296.17999999999995|
|      Arizona|      Jumping Stilts| 96.00999999999999|
| Pennsylvania|           Disc Golf|             28.08|
+-------------+--------------------+------------------+
only showing top 10 rows
```

DAG Visualization under SQL


InMemoryColumnarTableScan
ConvertToSafe
Generate
Project
number of rows: 5476
TungstenAggregate
number of input rows: 5476
number of output rows: 3582
data size total (min, med, max):
32.5 MB (16.2 MB, 16.2 MB, 16.2 MB)
spill size total (min, med, max):
0.0 B (0.0 B, 0.0 B, 0.0 B)
TungstenExchange
TungstenAggregate
number of input rows: 26
number of output rows: 11
data size total (min, med, max):
0.0 B (0.0 B, 0.0 B, 0.0 B)
spill size total (min, med, max):
0.0 B (0.0 B, 0.0 B, 0.0 B)
ConvertToSafe
Limit

In [14]:

```
# We can also register the result sets as temporary tables
revenue_by_state.registerTempTable('state_revenue')
```

Write an UDF (User defined function) to extract week day name from the date field

In [15]:

```
# Define a user defined function to be invoked from sql query. For example, deriving
import datetime
def getDayOfWeek( date):
    return datetime.datetime.strptime(date, "%m-%d-%Y").strftime('%A')
```

Register the function to SQL Context as new UDF

In [16]:

```
# Register the function
from pyspark.sql.types import StringType
sqlContext.udf.register("getDayOfWeek", lambda date: getDayOfWeek( date ), StringType
```

Invoke the UDF from the SQL

In [17]:

```
# Write a query to invoke the user defined function.. Calculate the total revenue by
revenue_by_state = sqlContext.sql( '''select weekday as weekday, round( sum( amount )
                                  from ( select getDayOfWeek( tDate ) as weekday, amo
                                  group by weekday order by total desc''' )
```

In [18]:

```
revenue_by_state.show( 10 )
```

```
+---------+--------+
|  weekday|   total|
+---------+--------+
| Thursday| 94549.2|
|Wednesday|85091.56|
|   Monday|81712.77|
|   Sunday|79634.08|
|  Tuesday|79594.51|
| Saturday|78114.84|
|   Friday| 71809.1|
+---------+--------+
```

Using jdbc to read from mysql table

In [19]:

```
## Read customer information from mysql table....
cust_df = sqlContext.read.format('jdbc').options(url='jdbc:mysql://localhost/retail?u
                                                 dbtable='customers').load()
```

In [20]:

```
cust_df.show( 10 )
```

```
+-------+---------+----------+---+--------------------+
| CustID|FirstName|  LastName|Age|          Profession|
+-------+---------+----------+---+--------------------+
|4000001| Kristina|     Chung| 55|               Pilot|
|4000002|    Paige|      Chen| 74|             Teacher|
|4000003|   Sherri|    Melton| 34|         Firefighter|
|4000004| Gretchen|      Hill| 66|Computer hardware...|
|4000005|    Karen|   Puckett| 74|              Lawyer|
|4000006|  Patrick|      Song| 42|        Veterinarian|
|4000007|    Elsie|  Hamilton| 43|               Pilot|
|4000008|    Hazel|    Bender| 63|           Carpenter|
|4000009|  Malcolm|    Wagner| 39|              Artist|
|4000010|  Dolores|McLaughlin| 60|              Writer|
+-------+---------+----------+---+--------------------+
only showing top 10 rows
```

Finding total money spent by each customers

In [21]:

```
top_10_custs = sqlContext.sql( '''select customerNo, round( sum( amount ), 2 )
                              as total from txnrecords group by customerNo
                              order by total desc limit 10''')
```

In [22]:

```
top_10_custs.registerTempTable( "top_10_custs" )
cust_df.registerTempTable( "custs" )
```

Joining with customer table to find top 10 customers based on total money spent

In [23]:

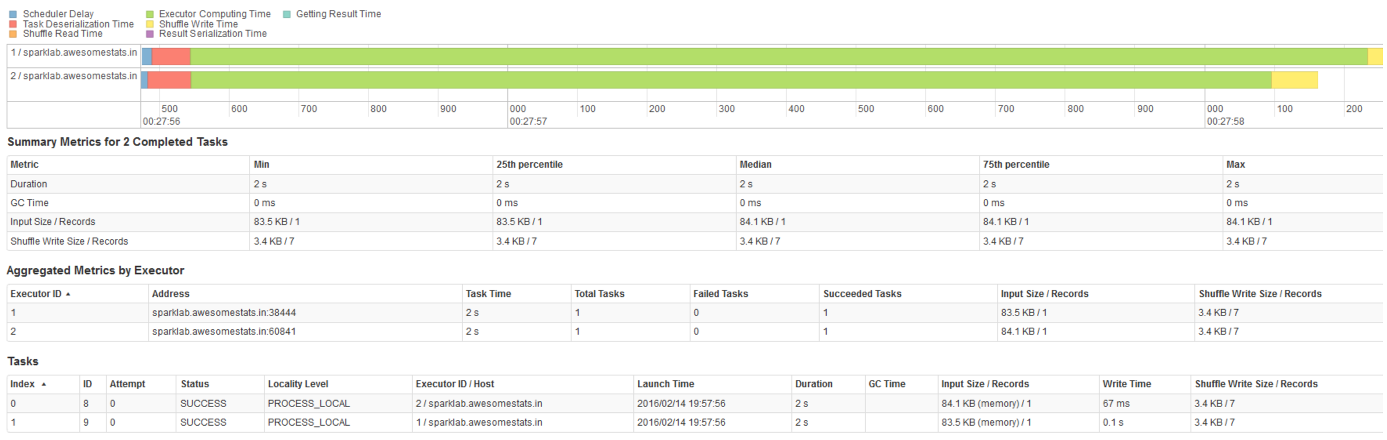
```
top_10_cust_names = sqlContext.sql( '''select a.CustID, a.FirstName, a.LastName,
                                   a.Age, b.total from custs a join top_10_custs  b
                                   on a.CustID = b.customerNo order by b.total desc''
```

In [25]:

```
top_10_cust_names.show( 10 )
```

```
+-------+---------+--------+---+-------+
| CustID|FirstName| LastName|Age|  total|
+-------+---------+--------+---+-------+
|4007510|  Kristin|    Levin| 73|2204.79|
|4003293|   Martha|   Warner| 45|2024.67|
|4003971|   Donald|     Lamm| 34|1869.43|
|4004260| Courtney|    Rubin| 54|1869.12|
|4001058|   Gloria| Matthews| 53|1791.99|
|4008914| Samantha|Batchelor| 41|1652.06|
|4004491|     Rita|    Parks| 44|1649.74|
|4007168|  Carolyn|      Han| 52|1610.76|
|4001253|    Peter| McNamara| 74|1516.77|
|4009672|   Samuel|     Kidd| 61|1485.23|
+-------+---------+--------+---+-------+
```

Timeline



Summary Metrics for 2 Completed Tasks

Metric	Min	25th percentile	Median	75th percentile	Max
Duration	2 s	2 s	2 s	2 s	2 s
GC Time	0 ms	0 ms	0 ms	0 ms	0 ms
Input Size / Records	83.5 KB / 1	83.5 KB / 1	84.1 KB / 1	84.1 KB / 1	84.1 KB / 1
Shuffle Write Size / Records	3.4 KB / 7	3.4 KB / 7	3.4 KB / 7	3.4 KB / 7	3.4 KB / 7

Aggregated Metrics by Executor

Executor ID ▲	Address	Task Time	Total Tasks	Failed Tasks	Succeeded Tasks	Input Size / Records	Shuffle Write Size / Records
1	sparklab.awesomestats.in:38444	2 s	1	0	1	83.5 KB / 1	3.4 KB / 7
2	sparklab.awesomestats.in:60841	2 s	1	0	1	84.1 KB / 1	3.4 KB / 7

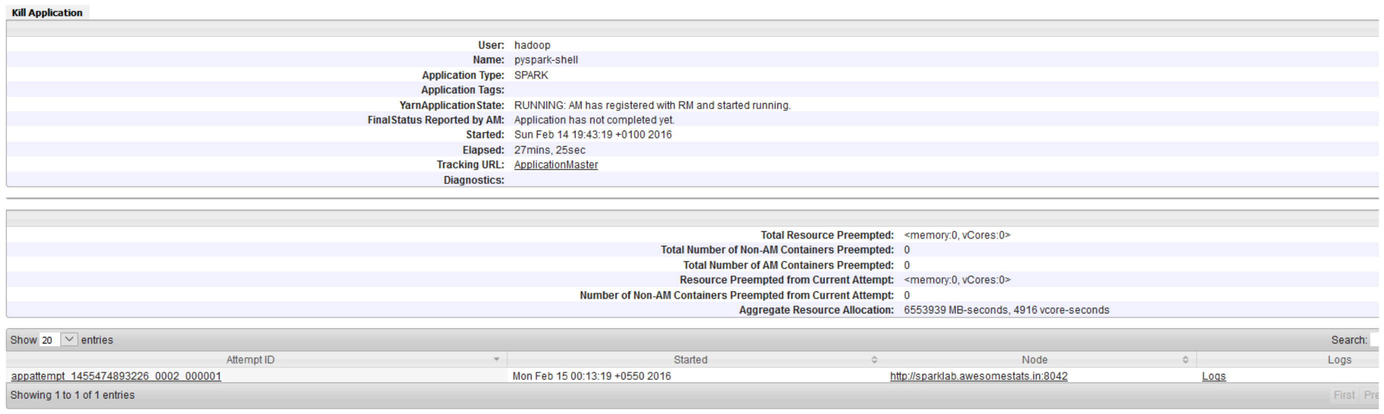
Tasks

Index ▲	ID	Attempt	Status	Locality Level	Executor ID / Host	Launch Time	Duration	GC Time	Input Size / Records	Write Time	Shuffle Write Size / Records
0	8	0	SUCCESS	PROCESS_LOCAL	2 / sparklab.awesomestats.in	2016/02/14 19:57:56	2 s		84.1 KB (memory) / 1	67 ms	3.4 KB / 7
1	9	0	SUCCESS	PROCESS_LOCAL	1 / sparklab.awesomestats.in	2016/02/14 19:57:56	2 s		83.5 KB (memory) / 1	0.1 s	3.4 KB / 7

Checking logs on YARN

For Driver logs, click on logs link on down right

For Executors logs, click on the attempt id on down left



Kill Application

User:	hadoop
Name:	pyspark-shell
Application Type:	SPARK
Application Tags:	
YarnApplicationState:	RUNNING: AM has registered with RM and started running.
FinalStatus Reported by AM:	Application has not completed yet.
Started:	Sun Feb 14 19:43:19 +0100 2016
Elapsed:	27mins, 25sec
Tracking URL:	ApplicationMaster
Diagnostics:	

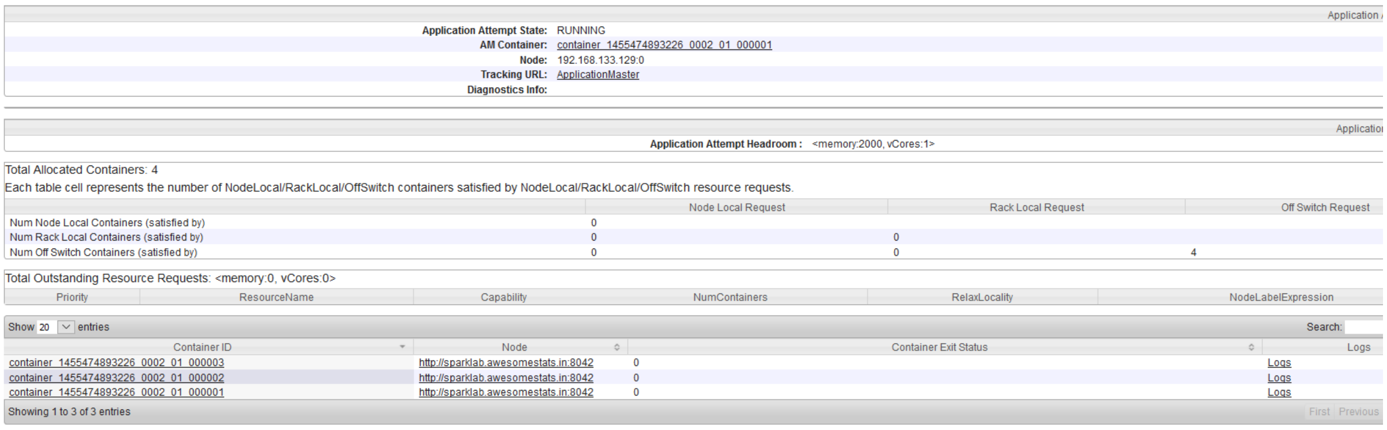
Total Resource Preempted:	<memory:0, vCores:0>
Total Number of Non-AM Containers Preempted:	0
Total Number of AM Containers Preempted:	0
Resource Preempted from Current Attempt:	<memory:0, vCores:0>
Number of Non-AM Containers Preempted from Current Attempt:	0
Aggregate Resource Allocation:	6553939 MB-seconds, 4916 vcore-seconds

Show 20 entries Search:

Attempt ID	Started	Node	Logs
appattempt_1455474893226_0002_000001	Mon Feb 15 00:13:19 +0550 2016	http://sparklab.awesomestats.in:8042	Logs

Showing 1 to 1 of 1 entries First Pre

For each Executor logs, click on logs link on each containers



Application

Application Attempt State:	RUNNING
AM Container:	container_1455474893226_0002_01_000001
Node:	192.168.133.129:0
Tracking URL:	ApplicationMaster
Diagnostics Info:	

Applicatio

Application Attempt Headroom : <memory:2000, vCores:1>

Total Allocated Containers: 4

Each table cell represents the number of NodeLocal/RackLocal/OffSwitch containers satisfied by NodeLocal/RackLocal/OffSwitch resource requests.

	Node Local Request	Rack Local Request	Off Switch Request
Num Node Local Containers (satisfied by)	0		
Num Rack Local Containers (satisfied by)	0	0	
Num Off Switch Containers (satisfied by)	0	0	4

Total Outstanding Resource Requests: <memory:0, vCores:0>

Priority	ResourceName	Capability	NumContainers	RelaxLocality	NodeLabelExpression

Show 20 entries Search:

Container ID	Node	Container Exit Status	Logs
container_1455474893226_0002_01_000003	http://sparklab.awesomestats.in:8042	0	Logs
container_1455474893226_0002_01_000002	http://sparklab.awesomestats.in:8042	0	Logs
container_1455474893226_0002_01_000001	http://sparklab.awesomestats.in:8042	0	Logs

Showing 1 to 3 of 3 entries First Previous

Make a note of things you learnt in the exercise.