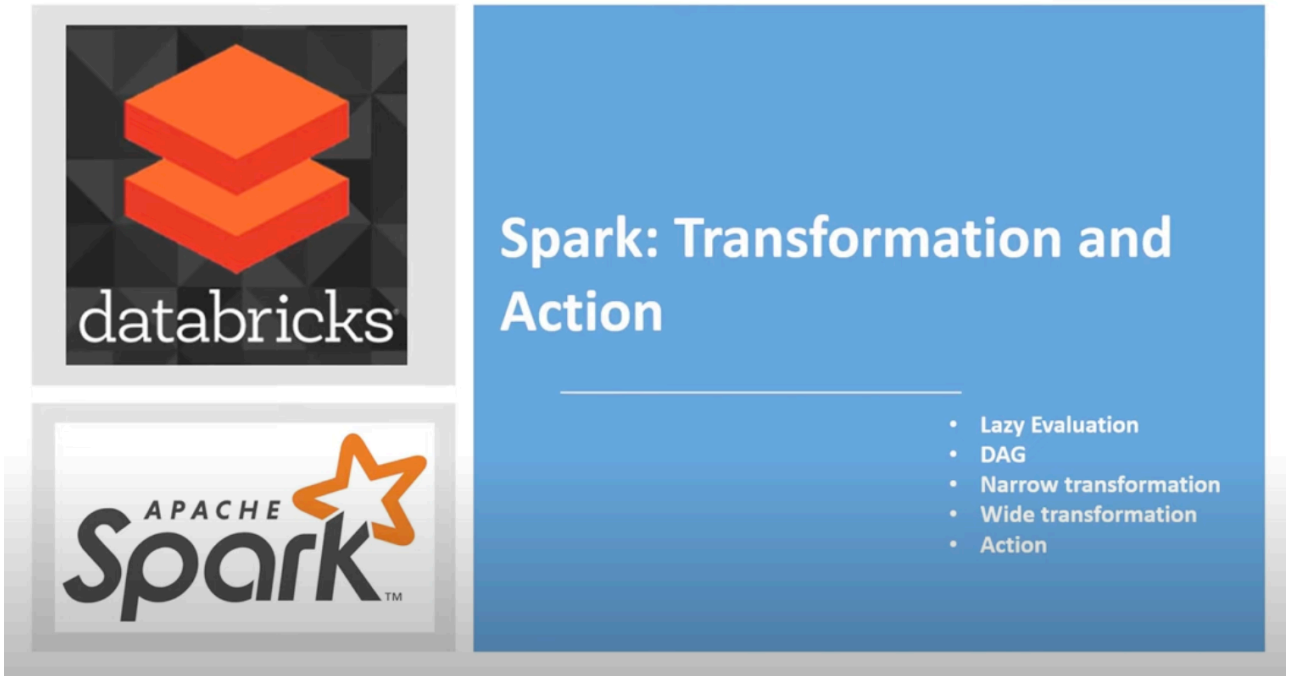
03. Databricks | PySpark: Transformation and Action

Monday, 4 November 2024 11:03 AM

[03. Databricks | PySpark: Transformation and Action]



In spark world, transformation and actions are operations used in spark

programming.

Transformation:

Transformation is kind of operation which will transform Dataframe from one form to another. We will get new Dataframe after execution of each transformation operation. Transformation is lazy evaluation based on DAG (Directed Acyclic Graph). Filter, Union etc.,

Dataframe is immutable, when transformation is applied new df is created.

Lazy evaluation - actual result data is not provided/produced immediately. Spark engine will only create the Logical plan for that transformation and then create DAG.
Logical plan of execution steps will be associated with DAG.

Until unless we call an action, this trf will not get executed. So only lazy evaluation will be performed to create the DAG. And then action is to be called to execute the transformation.

Example of transformations -> filter, union etc

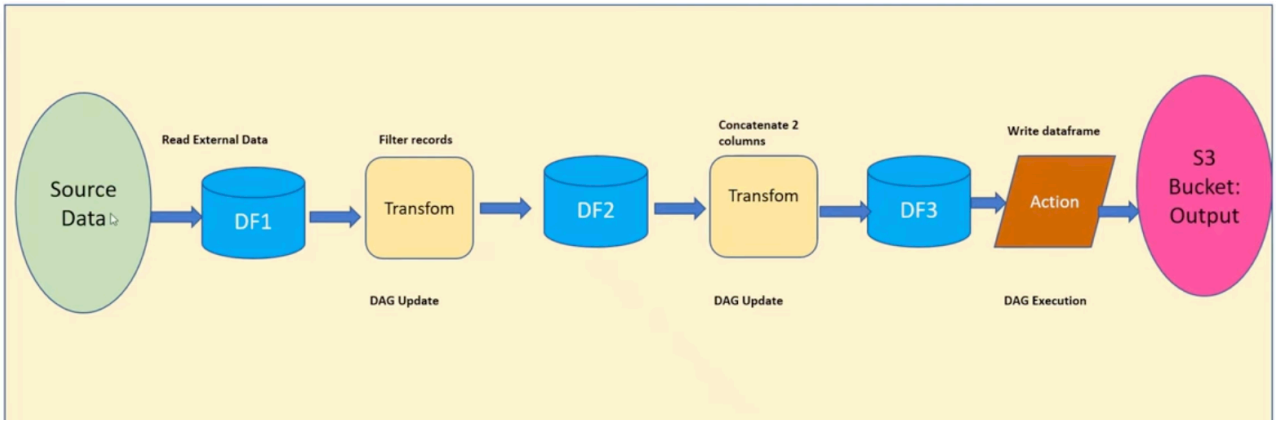
Action:

When we want to work with actual data, call the action. Action returns the data to driver for display or storage into storage layer. Count, collect, Save etc.,

Action would executed all the transformations that are associated with a DAG and return data to driver for display or stored it in designated storage location.

Exp -> count, collect, save etc

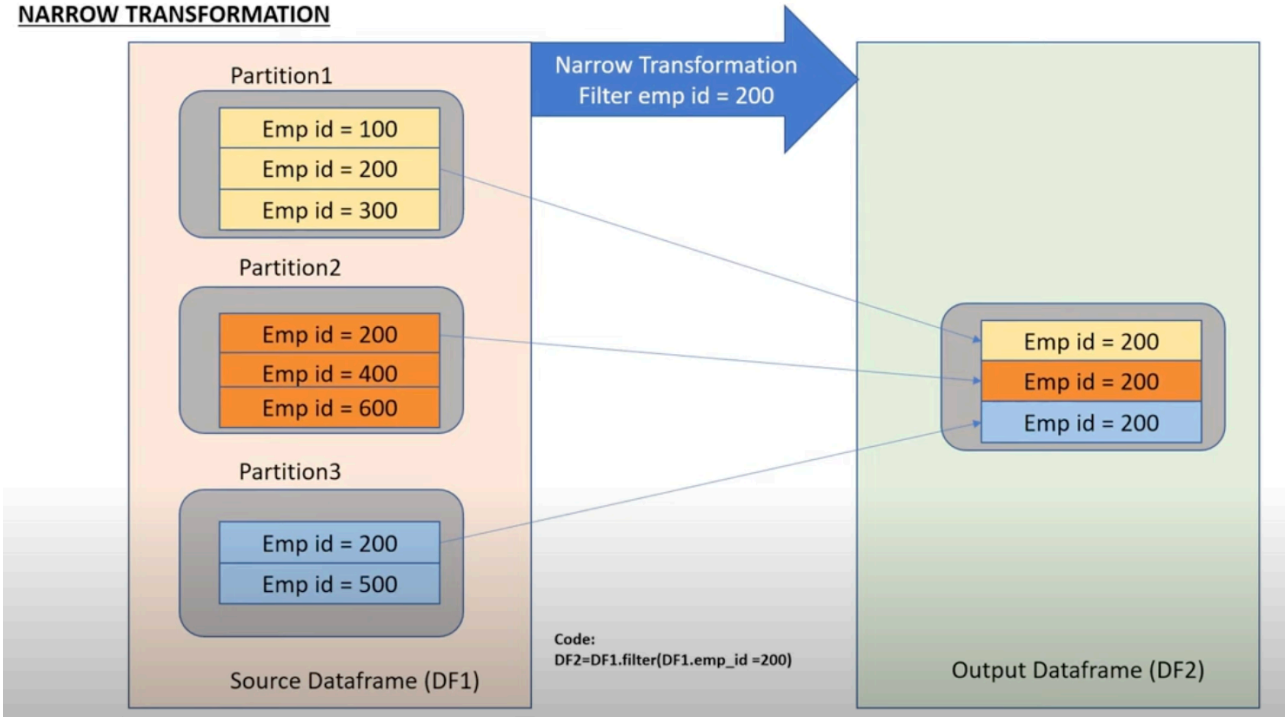
Lazy Evaluation & DAG



Initially when we apply transformation the logical plan is written in DAG, DF1 gets data from source, DF2 is second DF after filter(as dataframe is immutable new DF2 is created after DF1) then DF3 is created after second transformation. All this is written in DAG.

When action to write final result df to S3 is called, it will refer the DAG and execute all the steps from step1(DF1) to DF3, this is how lazy evaluation work.

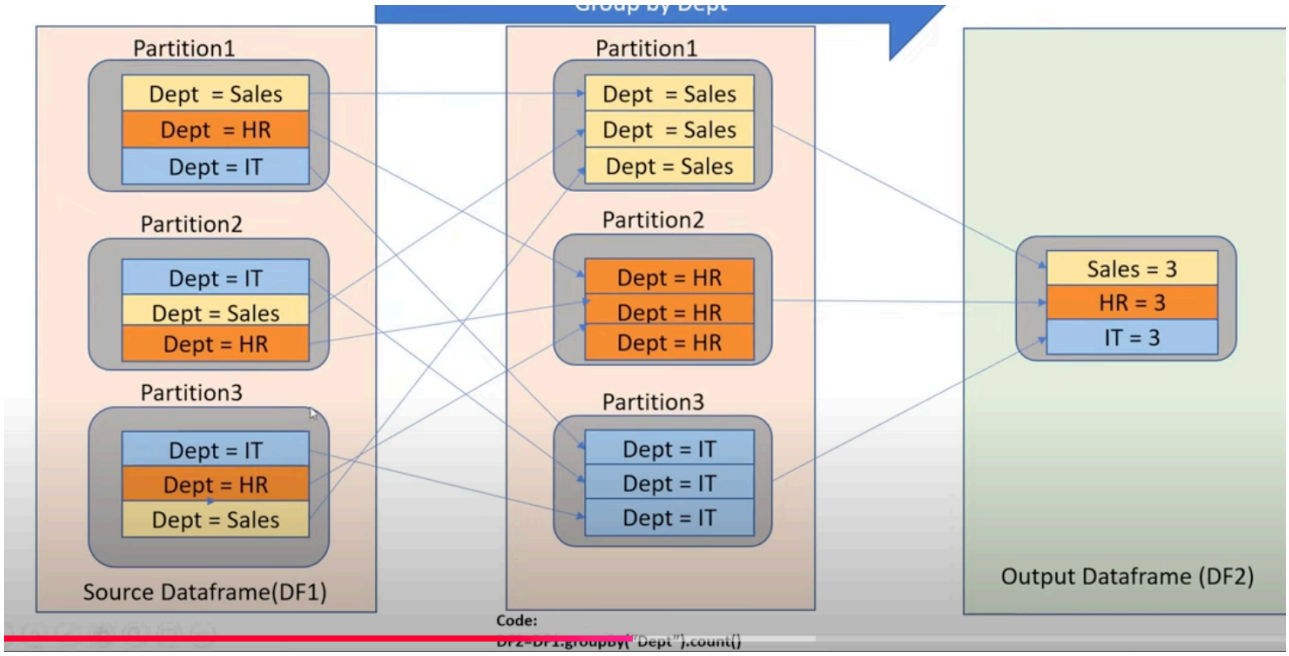
Narrow Transformation vs Wide Transformation



In **narrow transformation**, we do not need to shuffle data across the nodes. This is simple transformation and is inexpensive.

In above example - say we have one cluster and it has 3 executors, each executor has the above data partition on it.
We want to filter data where empid = 200, this logic is provided by driver to each executor and each executor will give filtered data to driver where it will concatenated to build the final result data.

Wide transformation



In wide transformation data is getting shuffled across nodes that is why it is expensive and hits performance.

We need to shuffle data as each partition is depending on other partition to get the right data/output.

Groupby is a wide transformation

Transformation List:

- map
- filter
- flatMap
- mapPartitions
- mapPartitionsWithIndex
- groupBy
- sortBy
- union
- intersection
- subtract
- distinct
- cartesian
- zip
- sample
- randomSplit
- keyBy
- zipWithIndex
- zipWithUniqueID
- zipPartitions
- coalesce
- repartition
- pipe

Action List:

- reduce
- collect
- aggregate
- fold
- first
- take
- forEach
- top
- treeAggregate
- treeReduce
- forEachPartition
- collectAsMap
- count
- takeSample
- max
- min
- sum
- histogram
- mean
- variance
- stdev
- sampleVariance
- countApprox
- countApproxDistinct

- Save

Most commonly used trfs

Filter
Groupby
Sortby
Union - narrow trf

Most commonly used Actions
Collect
First
Take
Foreach
Save

Create df from file

```
df = spark.read.format("csv")\
            .option("header", "true")\
            .option("sep", ",")\
            .option("inferschema", "true")\
            .option("mode", "FAILFAST")\
            .load("/rt_test/2010-summary.csv")
display(df)
```

```
df1 = df.filter(df["DEST_COUNTRY_NAME"] == "Honduras")
```

```
df2 = df.filter(df["DEST_COUNTRY_NAME"] == "Equatorial
Guinea")
```

```
df3 = df1.union(df2)

# all of the above queries are narrow transformations and there
will be lazy evaluation which means there will be only
execution plan created and DAG will be saved with steps to get
to df3, there will be no real data processing until Action is
called.
```

```
df4 = df3.groupby("DEST_COUNTRY_NAME").count()
```

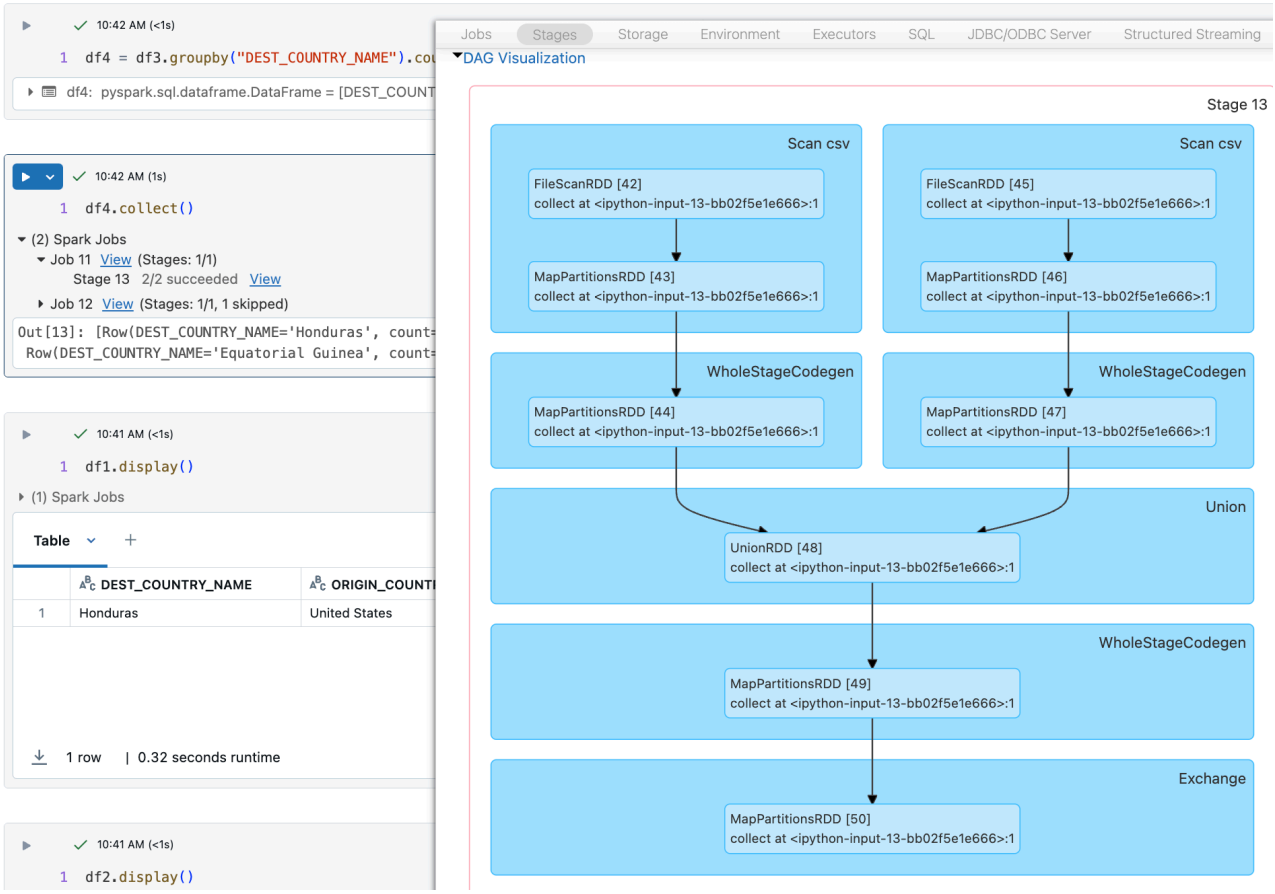
this is wide transformation and data needs to shuffle on executors.

```
df4.collect()
```

this is a action called. Here spark will refer to the DAG and will perform all the steps from df to df4 and then build output and display it on terminal.

If we run display() on any of other dataframes like df1 or df2,3 that will also run the steps as saved in DAG.

DAG can be viewed from spark stage.



Exchange stage means data shuffling happened to get the result.