Elastic Map Reduce (EMR) vs Databricks – Development examples

Elastic Map Reduce (EMR) can run Spark workloads on persistent or transient clusters. It is an AWS offering.

Databricks is a platform created by the original developers of Spark; and is also used to run Spark workloads. Databricks can be used in either the AWS cloud, Microsoft Azure or in the Google cloud, but today I am only considering the AWS cloud. Databricks is very similar in all environments.

EMR is a minimally featured offering, designed to be used in conjunction with other AWS offerings, such as Apache Airflow, or AWS Lambda or Step functions for job scheduling and automated cluster management, and Athena or Redshift for querying data. Databricks on the other hand is a fully featured offering with development tools, database browsing, cluster management and job scheduling built in.

Both offerings use Cloud Storage, in AWS this would typically be S3.

There are plenty of articles comparing EMR and Databricks. Most of them mention that EMR has a much steeper learning curve than Databricks. This is true but doesn't really give you much insight into what it is like to work with either EMR or Databricks, so I want to take you through a few summary examples of how a developer might work with each of them.

Caveat: there are multiple ways to use tools such as EMR and Databricks. This article draws on my personal experience and the methods that I am most familiar with. There are more ways to perform tasks than the ones described below.

Example 1: Using a Jupyter Notebook

EMR

EMR requires some configuration to use Jupyter notebooks. It is necessary to establish a new cluster or connect to a running cluster with the right configuration each time a notebook is run. It can take from 10 to 30 minutes for EMR clusters to start

Databricks

In the Databricks console under Workspaces, create a new notebook. You don't need to have a cluster running to create or open a workbook, but you will need to attach to a running cluster to run code. You can attach to and start clusters from within the notebook.

Example 2: Running code

EMR

You can run code via a notebook (see Example 1), but if you want to run code created as a module, you will need to store your code on an S3 bucket that EMR has access to. Create a cluster or locate a running cluster and add a step to the cluster. Wait for your step to run, then look at the log files to see the outputs.

Databricks

a new notebook (see example 1). You can start writing code straight away, but you will need to attach to a cluster to run the code. You can then either run each cell individually, getting status and results after each step, or you can run the whole notebook. In my experience clusters usually take from 2 to 5 minutes to start, much faster than EMR.

Example 3: Exploring data

EMR

You could use a Notebook and load files into datasets to see the contents. However, it would make more sense to use either Athena or Redshift to do any data exploration. Use EMR to create the Delta files, and other tools to explore them.

Databricks

As with EMR, you could create the Delta files using Databricks, then explore them via Athena or Redshift. However, Databricks has inbuilt data exploration capability as well. To make the most of data exploration in Databricks, use the Unity Catalog feature.

Example 4: Automating a job

EMR

EMR does not have any automation capacity. Instead, you would use another product, such as AWS Step functions, AWS Lambda functions, or Apache Airflow. A tested python code module would be passed to an EMR cluster, either by adding a step to an existing cluster, or by starting a new one.

Databricks

A notebook can be scheduled to run using the Workflow feature. You can see the results of each job run, and see a copy of the scheduled notebook, including the output of all the cells.

Example 5 (Summary): Debugging an existing automated job

EMR

If a step fails on an EMR cluster, you will need to look at the logs on S3 to work out what has happened. To locate the logs, you will need the cluster identifier and the step id.

Databricks

Click on the Workflow tab and locate your job and click on the job name to see the job runs. Find the execution that failed and click on the link. You will see a copy of the notebook that was scheduled, and the outputs of each cell, like if you had run the notebook manually.

Some questions to ask when comparing EMR and Databricks for your company's use case

- Will only experienced developers be using your solution, or do Data Scientists and Analysts need to access it as well?
- Is your development team already very comfortable with a development language used by the tool, such as Python, or is this new to them? You will need to use Python or Scala for both EMR and Databricks, however Databricks provides more assistance in getting started.
- For the other users (eg. Data Scientists and Analysts) using the solution, how comfortable are they using languages such as Python or Scala vs SQL? In Databricks, the Catalog function means that you can have Data Engineers set up Delta Tables (or other storage mechanisms) to allow users to query the data using SQL syntax. This can limit the need for other users to work with Python or Scala and allow them to upskill more gradually.
- How familiar is your team with the other tools that might be needed for an EMR solution (eg. Apache Airflow, Lambda functions, Step Function, Athena, Redshift)?
- How complex is the solution you are trying to implement? The more complex the solution, the more the additional overhead of using EMR will impact on your development and support.

Conclusion

From these examples, I hope you have a better understanding of what the different "learning curves" between EMR and Databricks entail. You can do a lot of the same things in both products, but the experience in Databricks is far more streamlined and intuitive. Even once you know how to use it, EMR takes more steps and more time to wait for resources. This can add up to a lot of time per developer per day.

When you compare costs between the two options, I recommend that you take the increased developer time and effort of EMR into account, as well as the costs of the additional systems (eg.

Apache Airflow, Lambda functions, Step functions, Athena, Redshift) that you may need for a complete solution.

If you would like assistance in establishing an Apache Spark capability at your organisation then please contact our Fujitsu Data & AI specialist now.

Contact
Fujitsu Data & AI
+61 3 9924 3000