John Mahoney
Project

AWS Spark vs. Databricks Spark for Cloud Machine Learning

Executive Summary

Spark is one of the most popular cloud based tools for deploying machine learning models on data, and it is especially popular for big data machine learning. Spark clusters are typically deployed on Amazon Web Services (AWS) by directly building a cluster through the UI console. However, there is another option, Apache Databricks will link to your AWS account and build the cluster for you. To assess the differences in these systems, both performance and user experience, seven data sets were created of sizes: 1.7, 3.4, 6.8, 13.6, 27.2, 54.4, and 108.8 million rows each with 56 columns. These data sets were each fed to three machine learning models (logistic regression, logistic regression with SGD and K-means clustering) on clusters of sizes two, four and eight workers on both AWS Spark and Databricks Spark running the same machines (m5.xlarge: 16 GB memory, 4 cores). Databricks not only provides a host of features to make it easier to set-up and use a Spark cluster, but these clusters seem to outperform vanilla AWS Spark clusters in most settings. It needs to be mentioned that Databricks is a premium service that runs on top of AWS, so when working on Databricks a user will be billed by both AWS and Databricks.

Background

Machine Learning is an important tool for data scientists and increasingly data sets are becoming larger often too large to fit on an average machine. This is why data people are turning to Spark clusters for machine learning. Spark has built in packages such as MLlib for machine learning as well as the ability to run many third-party packages as well. However, Spark can be intimidating and difficult for new users which is why Databricks is so interesting.

Databricks is a higher-level platform that runs a proprietary version of Spark with updates and features not yet included in the open source build. It has an easy to use interactive UI which is great for ingesting data, Databricks will automatically pause clusters that are not being used and it allows for multiple users to use the same cluster and/or notebook simultaneously. Databricks is a complete end-to-end environment for working in Spark.

Data

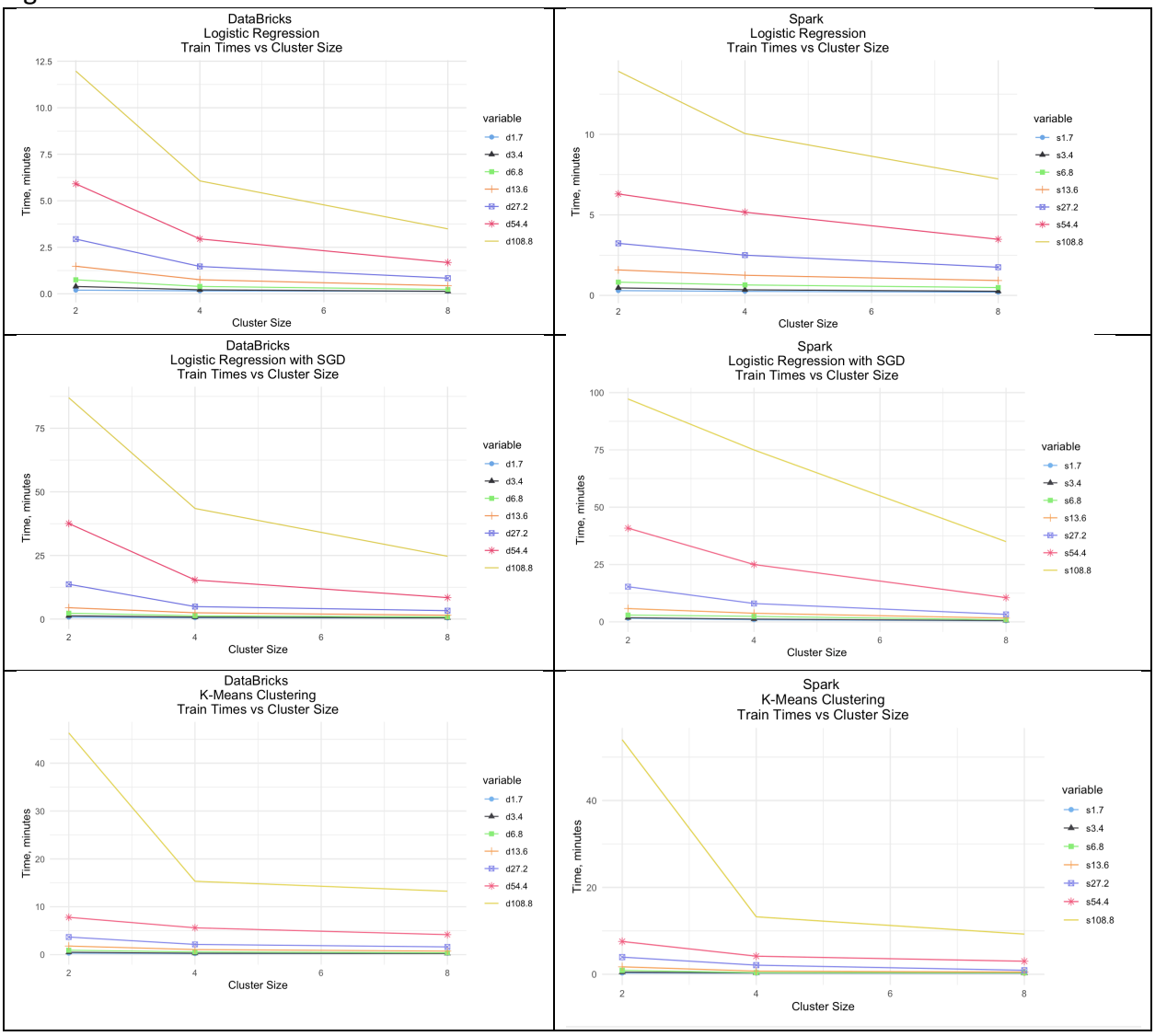
For these comparisons, I used the Seattle Public Library's collection inventory which is an open data set available from the city listing all the physical library materials for loan (no ebooks). This data set is about 11 GB, has 35.5 million rows and 13 columns. After cleaning and preparing the original data set was a little over 27 million rows and 124 columns. I then used

this data set to create seven data sets of sizes 1.7, 3.4, 6.8, 13.6, 27.2, 54.4, and 108.8 million rows each with 56 columns. In CSV form the 108.8 million row data set was

Performance Results

One each system (AWS Spark and Databricks) three clusters were created of sizes two, four, and eight worker nodes. Each cluster on either system exclusively used m5.xlarge machines with 16 GB memory and 4 cores each as we wanted an apples to apples comparison. For each cluster three MLlib machine learning models were trained five times on all seven data sets and the median train time recorded. Spark is prone to large outliers for run times and the median seemed the most appropriate way to get a good estimate. The machine learning models were Logistic Regression, Logistic Regression with Stochastic Gradient Descent (SGD) and a K-Means Clustering method with seven means. The train times are shown in the plots below (Figure 1).

Figure 1



We can see that for each learning model type the median train times for the smaller data sets, 1.7 million rows to 13.6 million rows, show little variation between Spark and Databricks. We also notice that the train times for these small data sets seem to be very similar to each other within systems which suggests that the overhead of scheduling tasks in the cluster is a large part of the train time. As the data set size increases it becomes more challenging for each cluster, especially the two cluster sizes

Databricks seems to have a performance advantage over AWS Spark. So, we calculated the average percent improvement in median train times by data size and cluster size. We bucketed similar performing data set sizes together into three buckets. The smaller data sets of one million to 14 million rows, the medium data set sizes of 25 to 55 rows and the largest data set was in a bucket by itself, 108.8 million rows. Figure 2 shows the databricks performance advantage changing as a function of cluster size. Figure 3 shows the databricks performance advantage changing as a function of data set size.

Figure 2

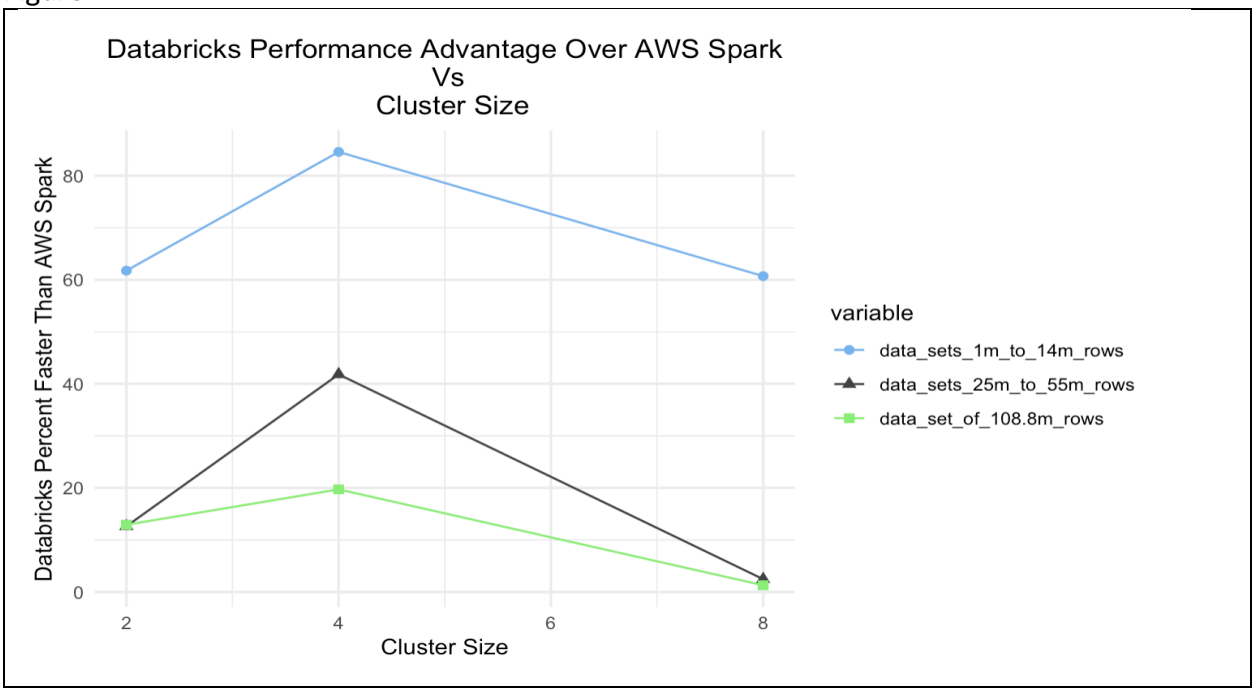
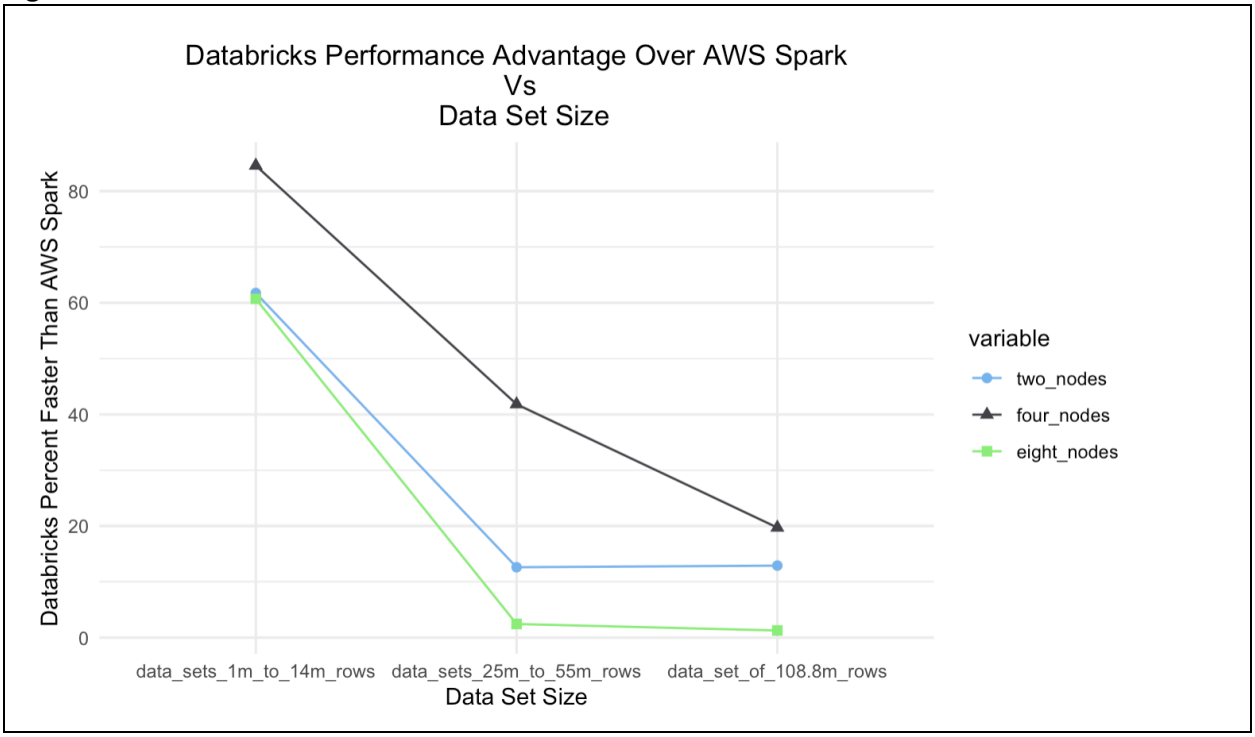


Figure 2 shows a definite performance advantage for Databricks over Spark for clusters of sizes two and four workers. That advantage becomes much smaller as the cluster size increases. It is possible that Databricks has a more efficient optimizer than AWS Spark and the performance difference between Databricks and Spark is most notable when the two optimizers are fully engaged. The performance advantage of the small data set (1-14 million rows) is consistently large however we must remember that we are using percent faster as our metric and 60% faster could be a very small amount of time.

Figure 3



In Figure 3 we see that for all cluster sizes the performance advantage of the Databricks system in terms of percent faster than Spark diminishes rapidly as data sets become larger. The performance advantage as data set size increases seems to be approaching a limit of around 12 percent for two node clusters, about 20 percent for the four node clusters and no advantage for eight node clusters.

The four-node cluster seems to be the optimal configuration for Databricks and our data sets. When we use clusters with two nodes the large data sets are too much and the small data sets are too little. Similarly, when using eight node clusters resources are too plentiful to really challenge either system.

User Experience

As mentioned Databricks is the easiest platform to create a new cluster. Everything is done in the browser UI and the documentation is great. The AWS Spark cluster is a bit more challenging to create. Another thing that came up was that the AWS Spark cluster would disconnect from my machine time to time, often when training models on the large data sets. Restarting can significantly increase the time required by the user to train a learning model but this time is not reflected in the measured train times. Databricks had no difficulties maintaining a connection to the cluster.

Conclusion

Databricks is the superior system, it is both easier to use and more performant. However, it charges about 50 cents an hour extra on top of the AWS charges which should be factored into any choices. For my money, the ease of use combined with stable connection to the cluster and the performance advantage make Databricks my preferred system, even when accounting for the extra cost.