Putting Big Data Analytics to the Test

A Comparison of Performance Benchmarks

databricks™

Introduction

The speed at which customers and markets are demanding value from organizations is increasing at an almost dizzying pace. Due to this, and the expectations of end users about how applications must perform, performance benchmarking has become a valuable way to rate and compare big data solutions. Keys to evaluating a big data analytics solution is to measure how it performs both with heavy load and at massive scale.

Understanding the performance of big data analytics platforms exhibit under various conditions is critical to choosing the right tool for the right job. This ebook introduces four essential benchmarks to rely on when selecting a big data analytics solution. Whether you are looking at open source Apache Spark or a fully managed platform such as Databricks' Unified Analytics Platform, you'll come away with a better understanding of what really matters when it comes to performance and which technologies come out on top:

RUNTIME PERFORMANCE
A performance standard from the Transaction Processing Performance Council (TPC) the TPC-DS benchmark is used to analyze the performance of decision support systems.

R WORKFLOWS
benchmark that shows accelerated common R workflows with improvements included in Databricks Runtime.

STREAMING
to test throughput, latency and completeness of streaming big data platforms.

SERVERLESS POOLS
benchmark test to demonstrate how the serverless pools fare when there is a concurrent and heterogeneous load.

Chapter 1: Runtime Performance

Complete benchmark analysis

ABOUT TPC-DS

The TPC-DS benchmark measures the performance of decision support systems including queries and data maintenance. This applies well to big data or big science systems that examine large volumes of data in an attempt to answer real world business questions (such as 'how will a change to the production volume of a good impact the business'). The benchmark measures the response times of 99 standard queries in: single user mode, query throughput in multi-user mode and data maintenance performance for a given hardware, operating system, and data processing system configuration under a controlled, complex, multi-user decision support workload.

The purpose of TPC benchmarks is to provide relevant, objective performance data to industry users. This is a very useful benchmark for the simple reason that for any business exploring big data platforms, SQL performance is going to be critical as it is the most commonly used language for big data analysis. The TPC-DS may be applied to any industry that must transform operational and external data into business intelligence.

ABOUT DATABRICKS RUNTIME

Databricks Runtime, part of the fully managed Databricks Unified Analytics platform, consists of Apache Spark and the Databricks I/O module (DBIO) which significantly improves the performance of Spark in the cloud. The infrastructure is serverless and highly elastic, designed to remove operational complexity while ensuring reliability and cost efficiency at scale.

BENCHMARK TEST

These benchmarks, led by Databricks co-founder and chief architect Reynold Xin, which included the 99 queries that make up the standard TPC-DS test, with an additional four queries which were pre-approved variants of four from the standard; and, an "s_max" query performing a full scan and aggregation of the biggest table in the test dataset (store_sales), for a total of 104 queries.

It should be noted that Spark SQL is one of the few open source SQL engines capable of running all 99 of the standard TPC-DS queries without modification.

VERSION AND CONFIGURATION

This benchmark compared Databricks Runtime 3.0 against the following:

- Vanilla open source Apache Spark in the cloud
- Presto in the cloud
- Apache Impala on-premises

DATABRICKS RUNTIME VS VANILLA OPEN SOURCE APACHE SPARK IN THE CLOUD

The queries used to test the performance of Databricks Runtime against Apache Spark on AWS were subdivided into the following query functional sub-groups: interactive queries, reporting queries, and deep analytics queries.

STEPS TAKEN: CONFIGURING THE HARDWARE

- Machine type: 11 r3.xlarge nodes (10 workers and 1 driver)
- CPU core count: 44 virtual cores (22 physical cores)
- System memory: 335 GB
- Total local disk space for shuffle: 880 GB
- Networking performance is described as “Moderate” by Amazon

STEPS TAKEN: LOAD THE DATASET

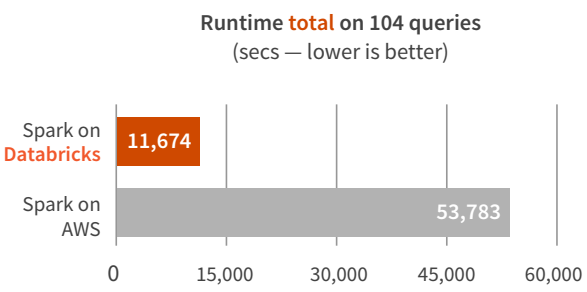
- The dataset used was the TPC-DS 1,000 scale factor on S3 (‘scale factor’ indicates raw data size in GB, in this case 1,000 GB)
- Row counts for tables scale realistically: fact tables grow linearly; and, dimension tables grow sub-linearly

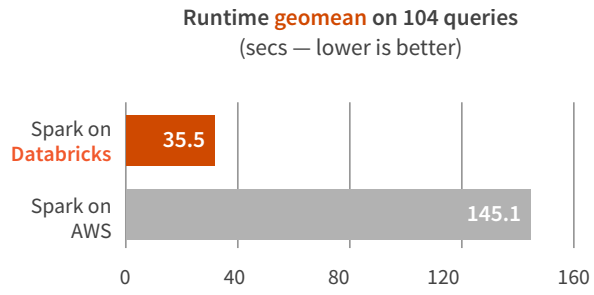
STEPS TAKEN: CONFIGURATION TUNING

- Databricks Runtime: out of the box configuration on Databricks
- Apache Spark: on AWS with additional configuration options that can be found here and here.

RESULTS

The runtime results included a recording for total runtime as well as the runtime geometric mean (geomean) for the 104 queries in this comparison. In the case of both, Databricks Runtime outperformed the competition.





Generally, the results show that Spark on Databricks performs roughly 5X better in total runtime and 4X better in geometric mean. A deeper dive into the results show the following performance differences of the query functional sub-groups:

- Interactive queries (completing within 1 min): **Databricks Runtime 3.0 is 3X faster**
- Reporting queries (completing within 3 mins): **Databricks Runtime 3.0 is 4X faster**
- Deep analytics queries (long running queries that can take an hour or more): **Databricks Runtime 3.0 is 5X faster**

DATABRICKS RUNTIME VS PRESTO IN THE CLOUD

Presto is an open-source distributed SQL query engine optimized for low-latency, ad-hoc analysis of big data. It supports the ANSI SQL standard, including complex queries, aggregations, joins, and window functions.

STEPS TAKEN: CONFIGURING THE HARDWARE

- The same hardware configuration used for the vanilla Spark comparison was used for the Databricks Runtime vs Presto (11 r3.xlarge nodes).

STEPS TAKEN: LOAD THE DATASET

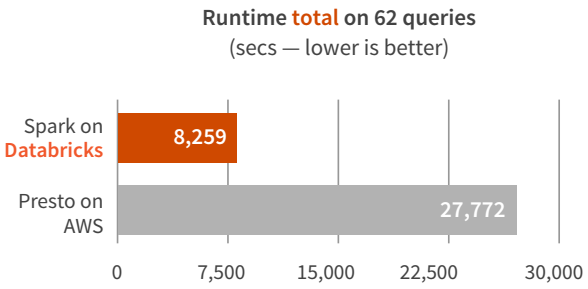
- TPC-DS 1,000 scale factor, on S3

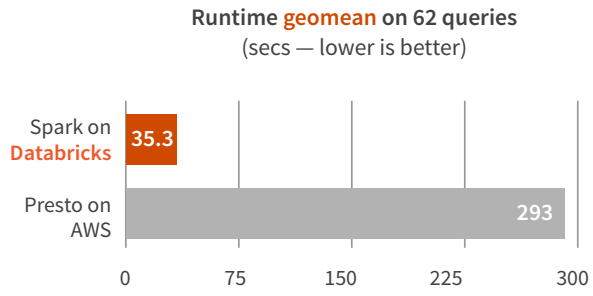
STEPS TAKEN: CONFIGURATION TUNING

- The only tuning required consisted of rewriting some queries for Presto due to the lack of support for the SQL grouping function for a rollup in the TPC-DS benchmark queries.

RESULTS

Only 62 queries could complete on Presto. The remainder of the queries (42) either crashed the system or returned no result. This information is needed to clarify why the total runtime on Presto is smaller than the total runtime for vanilla Spark from the previous comparison. The total runtime for Presto does not take into account the failing queries.





Comparing only the 62 queries Presto was able to complete, **Databricks Runtime performed 3x faster in total runtime** and was **8x better in geometric mean** than Presto.

DATABRICKS RUNTIME VS APACHE IMPALA ON-PREMISE

Apache Impala is an open source, massively parallel processing SQL query engine for data stored in clusters running Apache Hadoop (which is used primarily by Cloudera customers). Cloudera publishes benchmark numbers for the Impala engine and in the most recent benchmark only 77 queries of the 99 in the TPC-DS benchmark were used (this was determined due to 22 of the SQL statements in the benchmark using obscure SQL functionality).

This comparison examines how Databricks Runtime, which runs in the cloud, performs against Impala running on physical hardware tuned by the engineering team behind the product.

	DATABRICKS RUNTIME	CLOUDERA IMPALA
CPU Core Count	144 (288 AWS v CPUs)	280
Memory (GB)	2196	1792
Local Disk (TB)	68	112
Data Storage	S3 (decoupled storage and compute)	HDFS (local disks)
Machine Details	18 cloud i3.4xlarge	7 on-prem nodes

STEPS TAKEN: LOAD THE DATASET

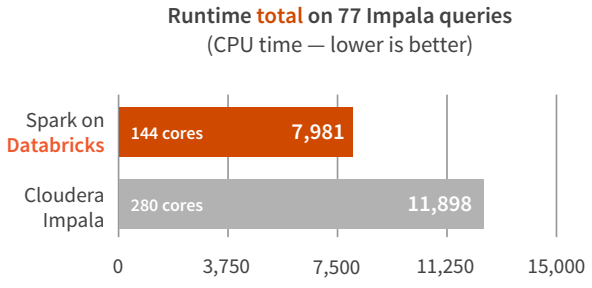
- TPC-DS 10,000 scale factor (10,000 GB)
  - On S3 for Databricks
  - On HDFS for Impala

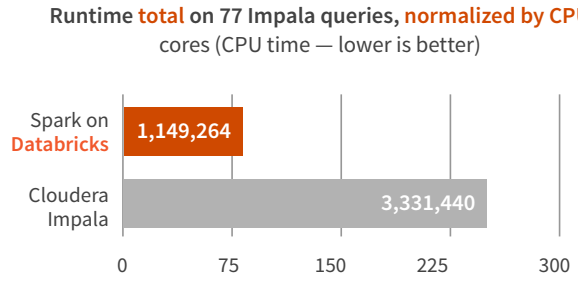
STEPS TAKEN: CONFIGURATION TUNING

- Databricks ran “out-of-the-box” with no configuration changes
- Impala configuration modifications were not reported in the Cloudera benchmark

RESULTS

Only 62 queries could complete on Presto. The remainder of the queries (42) either crashed the system or returned no result. This information is needed to clarify why the total runtime on Presto is smaller than the total runtime for vanilla Spark from the previous comparison. The total runtime for Presto does not take into account the failing queries.





The runtime results for the 77 queries specified in the Cloudera report show **Databricks Runtime is 1.5x faster than the same queries running on Impala on-premise**.

When taking into account the number of CPUs as a normalization factor, **Databricks Runtime, using commodity hardware in the cloud, is 3x more efficient than Impala on-premise**.

Chapter 2: R Workflows

Complete benchmark analysis

ABOUT THE R LANGUAGE

R is an open source language and environment used for advanced statistical computing and graphics. It is the most popular statistical software in use today and provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, …) and graphical techniques, and is highly extensible. It is used by millions of Data Scientists today, and its usage continues to grow rapidly.

More data professionals are using R with Spark, and the Databricks Unified Analytics Platform, with Databricks Runtime at its core, both accelerates and unifies the strengths of these two technologies providing incredibly fast and powerful advanced analytics capabilities.

Hossein Falaki and Xiangrui Meng from the Databricks engineering team conducted a benchmark test to show how the Databricks I/O (DBIO) accelerator package improved performance of `SparkR::collect()` and `SparkR::createDataFrame()` in Databricks Runtime version 3.3 vs 3.0. These two APIs are the bridges between single-node R and distributed Spark applications and are among the most frequently used functions of SparkR.

BENCHMARK TEST

For the benchmark Falaki and Meng used an airlines dataset that contains flight arrival and departure details for all commercial flights within the USA, from October 1987 to April 2008. It is a large dataset and takes up 1.6 gigabytes of space compressed and 12 gigabytes when uncompressed. It consists of over 120 million rows and 29 columns of numeric and character types in CSV format, which is common and popular among R users.

They progressively used larger fractions of the dataset to evaluate throughput and latency with varying data size and also found the limit after which the calls fail.

VERSION AND CONFIGURATION

Compared Databricks Runtime 3.3 with the older version of Databricks Runtime 3.0 on clusters comprising of four i3.xlarge workers.

STEPS TAKEN: MEASURING `SparkR::collect()` PERFORMANCE

- Load data using Spark's CSV data source with automatic schema inference.
- Cache and materialize the Spark `DataFrame`
- Collect it to a local R data.frame .
- Measure the total elapsed time of the collect step.

STEPS TAKEN: MEASURING `SparkR::createDataFrame()` PERFORMANCE

- Load the file from the local file system using R's tabular data reading functionality with a default configuration, which automatically infers schema.
- Parallelize the `data.frame` with `SparkR::createDataFrame()` and count the number of rows.
- Measure elapsed time of last two steps combined.

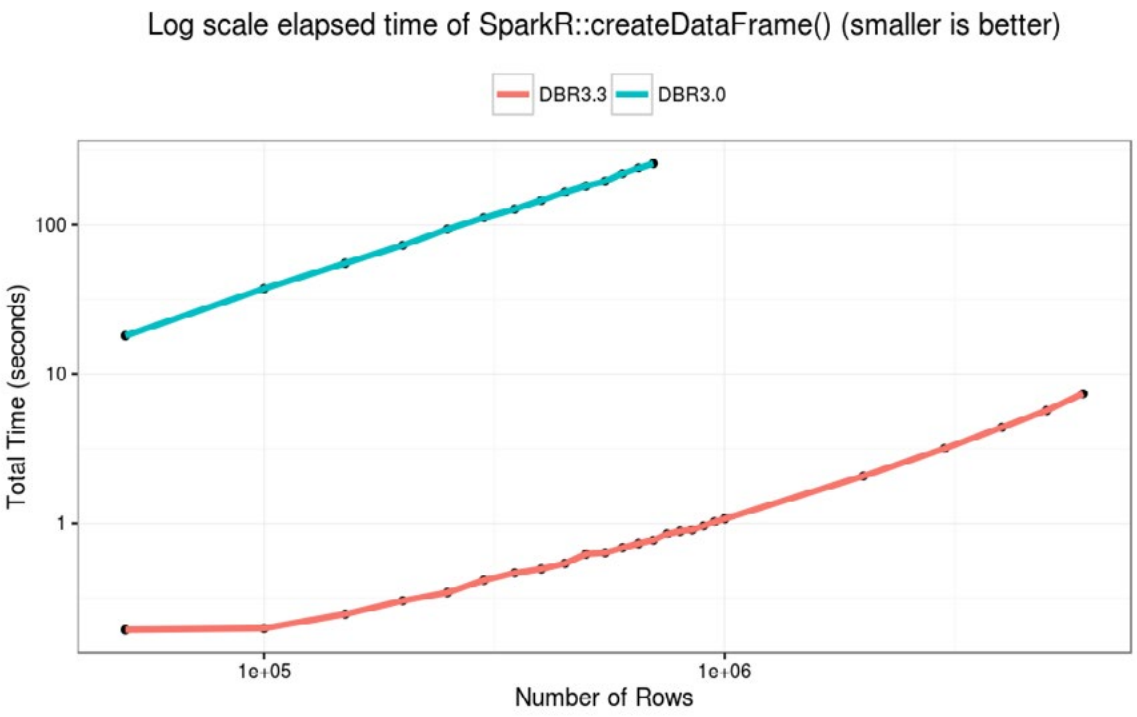
RESULTS

- Comparing average throughput of parallelizing R `data.frames` on DBR 3.3 and DBR 3.0. DBIO can achieve 300x higher average throughput in DBR 3.3 compared to DBR 3.0.
- When collecting Spark DataFrames, 24X higher average throughput on DBR 3.3 compared to the older version.

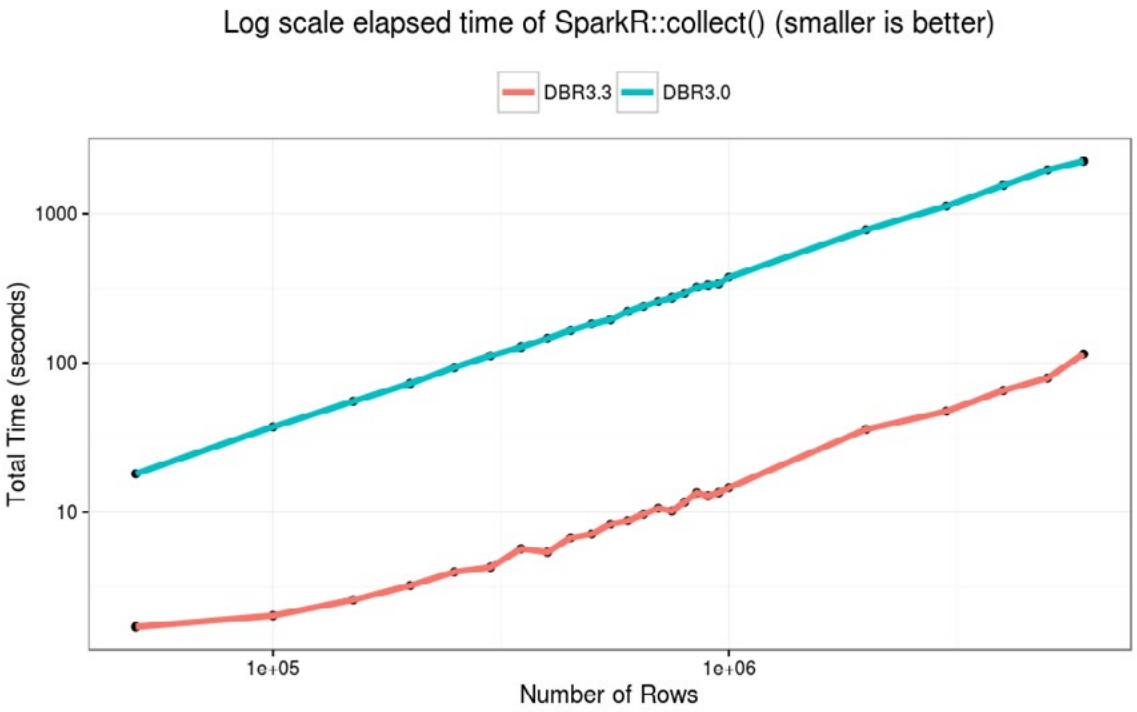
The plots below show end-to-end latency. This is what users perceive when calling SparkR API.

On each DBR version, input size was progressively increased until the call failed. Elapsed time was measured for each successful run. On DBR 3.0 `SparkR::createDataFrame()` failed with data larger than 750K rows (about 70MB). On DBR 3.3 the call did not fail for any R `data.frame`.

Overall `createDataFrame()` is 100X faster on DBR 3.3— and can handle much larger data.



The failure point of `SparkR::collect()` has not changed. At about 6M rows (550MB) the R process cannot handle the single in-memory object, and we observed failure when collecting Spark DataFrames. In this experiment, DBR 3.3 is 10x faster than older versions across varying input size.



In this benchmark test the DBIO in Databricks Runtime 3.3 significantly accelerates the performance of two of the most important SparkR calls: `SparkR::collect()` and `SparkR::createDataFrame()`.

Chapter 3: Structured Streaming

Complete benchmark analysis

ABOUT STREAMING

When working with big data, the main challenge that organizations must deal with are the massive volumes of data that are collected and what exactly to do with it. This is where data streaming becomes a critical part of a big data platform. When a significant amount of data needs to be quickly processed in near real time to gain insights, a streaming solution is the best answer.

Streaming data becomes valuable when it is able to be analyzed in real-time while in motion. It's very important that analyses can be carried out by entire data science teams in these short time windows, especially at scale, as any delay in the analysis can seriously reduce the reliability and therefore the value of the data itself. This is why being able to benchmark the performance of streaming data systems is so important. With so many industries becoming reliant on the intelligence that highly performant data streaming can provide for consumer, financial, and industrial applications, understanding which platforms perform under stringent conditions allows organizations to make the right investments in big data analytics platforms.

BENCHMARK TEST

Benchmarking the performance of the streaming systems tested was done by using the Yahoo Streaming Benchmark, which is an industry standard often used for evaluating these systems.

VERSION AND CONFIGURATION

The streaming systems evaluated were Databricks Runtime 3.1, Apache Flink 1.2.1, and Kafka Streams 0.10.2.1. The environment for benchmarking was created using Databricks Notebooks and cluster management which enabled the generation of consistent and reproducible scenarios for fair comparisons of performance.

STEPS TAKEN: SETTING UP THE TEST ENVIRONMENT

All of the systems to be evaluated can be done through Databricks Community Edition.

- Login to Databricks Community Edition. You can create an account here.
- Import the benchmark using the GitHub URL
- Launch a cluster
- Follow the instructions in the Main notebook regarding the installation of libraries and how to run the benchmark.

STEPS TAKEN: SET UP THE BENCHMARK TEST CASE

If you are not familiar with the specifics, the Yahoo Benchmark emulates an advertising application where a stream of ad events are consumed with the goal of computing event-time windowed counts of "viewed" ad campaigns. The order of operations is:

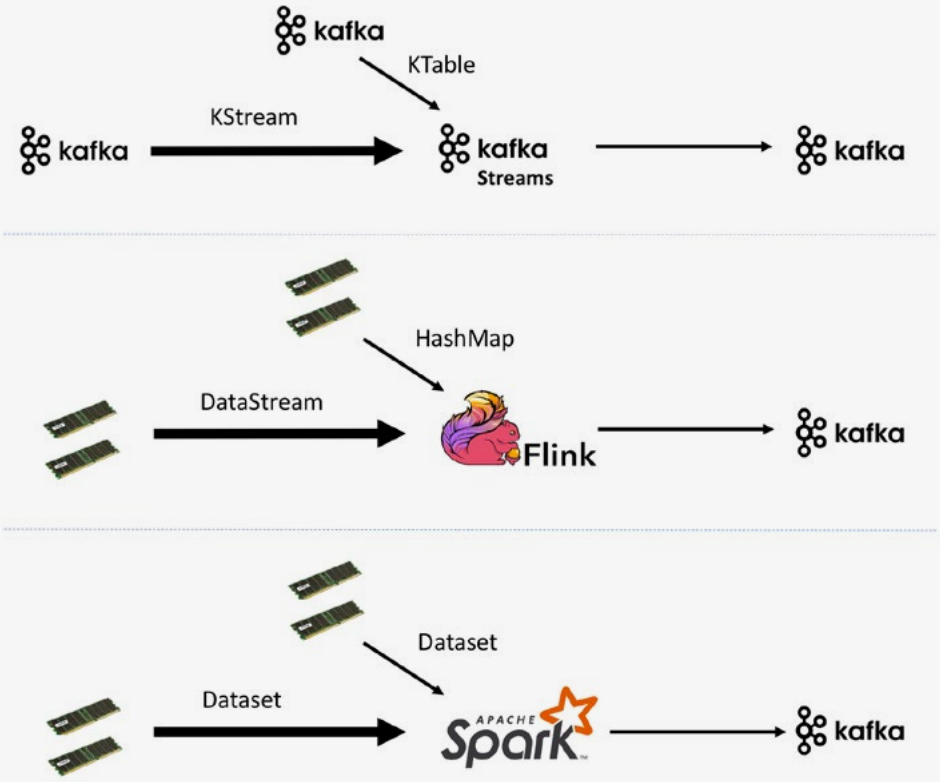
- Read JSON data from the data stream
- Filter events that we are interested in (view) based on the event_type field
- Take a projection of the relevant fields (ad_id and event_time)
- Join each event by ad_id with its associated campaign_id and store in a static table
- Take a windowed count of views per campaign and store each window with a timestamp of the time the last updated window. This step must be able to handle late events.

STEPS TAKEN: CONFIGURATION MODIFICATIONS

In order to ensure that the system being benchmarked was the bottleneck and not the result of an external service, the following modifications were made (and which are a replication of the modifications done in the benchmark published by data Artisans on Flink:

- The static table was joined with data streams as follows:
  - In Kafka Streams, we read the static table from Kafka as a KTable.
  - Flink doesn't support joins (in version 1.2.1) of a Datastream with a Dataset. Therefore, a hashmap lookup is performed
  - In Spark, we join with a static local Dataset.
- Data is generated as follows:
  - For Flink and Spark, we generate data in memory
  - Kafka Streams requires data to be stored persistently in Kafka, so we generate data using Spark and write it out to Kafka
- Data is written out to Kafka and Kafka timestamps were used to determine the timestamp of the last update for the window.

This process is illustrated on the following page:



STEPS TAKEN: CALCULATING LATENCY AND THROUGHPUT

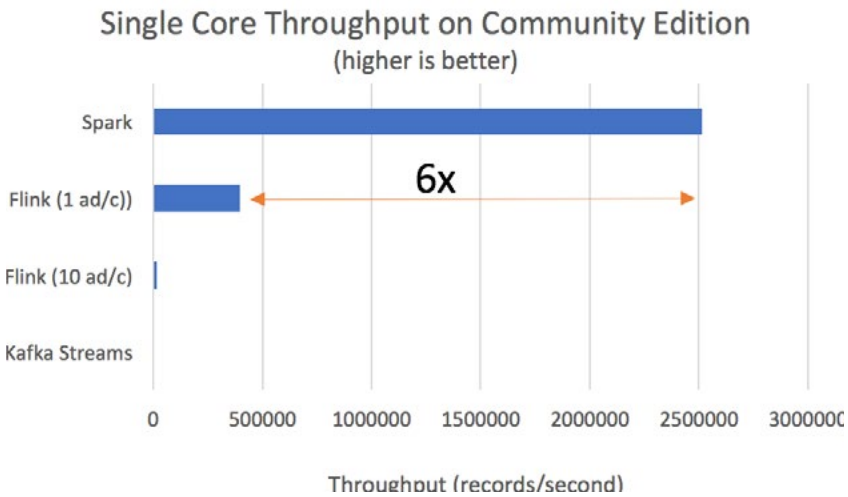
Latency was determined by taking the timestamp of the latest record that was received for a given (campaign, event-time-window) pair during the windowed counts phase. Then the difference between this timestamp and the Kafka ingestion timestamp of the output was used to calculate latency.

The methods used for calculating throughput was:

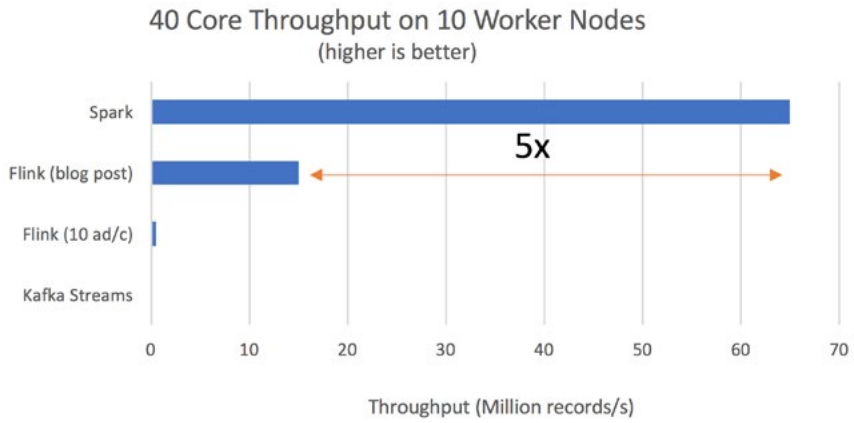
- For Spark, the `StreamingQueryListener` was used to record the start and end timestamps and the number of records processed.
- Kafka Streams was launched inside long-running Spark tasks. Right after starting the stream, a record would be sent to Kafka to mark the beginning timestamp, the number of records processed in each task once the stream is stopped was sent to Kafka as well. Then take the sum of the count of records, the minimum starting timestamp, the maximum ending timestamp, and then divide the sum of records by the duration.
- There was no good way to introspect the Flink job automatically, the number of records processed was periodically logged. This log was then used to figure out the start and end timestamps and the total records processed. This calculation is approximate, but the error is negligible on the scales we ran the benchmark.

RESULTS

The benchmark setup, which was ran on Databricks Community Edition, showed that Spark had 6x more throughput than Flink:



Once we ran the benchmark at scale (10 worker nodes) for each system, we observed the following results:



In the spirit of reproducible experiments and methodology, all of the scripts to reproduce these benchmarks are published in the benchmark GitHub repository.

Chapter 4: Serverless Pools

ABOUT SERVERLESS

Serverless is a hot new technology that enables organizations to build, deploy and run applications and microservices without thinking about servers at all.

Serverless applications don't require an IT team to provision, scale, and manage any servers, which means that developers can focus all of their efforts on developing products instead of worrying about managing servers or runtimes. The main benefit to this approach is increased efficiency in developing, deploying and operating applications.

Data science teams within large organizations spend too much time securing and provisioning hardware to run simulations, build predictive models, and perform data analysis. They also spend time and effort tuning hardware and fiddling with software settings for optimized performance. None of these tasks is central to their role as data scientists, and this leads to a loss of productivity overall.

With the Databricks Unified Analytics Platform users get the power of Databricks Serverless, a new initiative to offer serverless computing for complex data science and Apache Spark workloads. With Databricks Serverless, organizations can use a single, automatically managed pool of resources and get best-in-class performance for all users at dramatically lower costs.

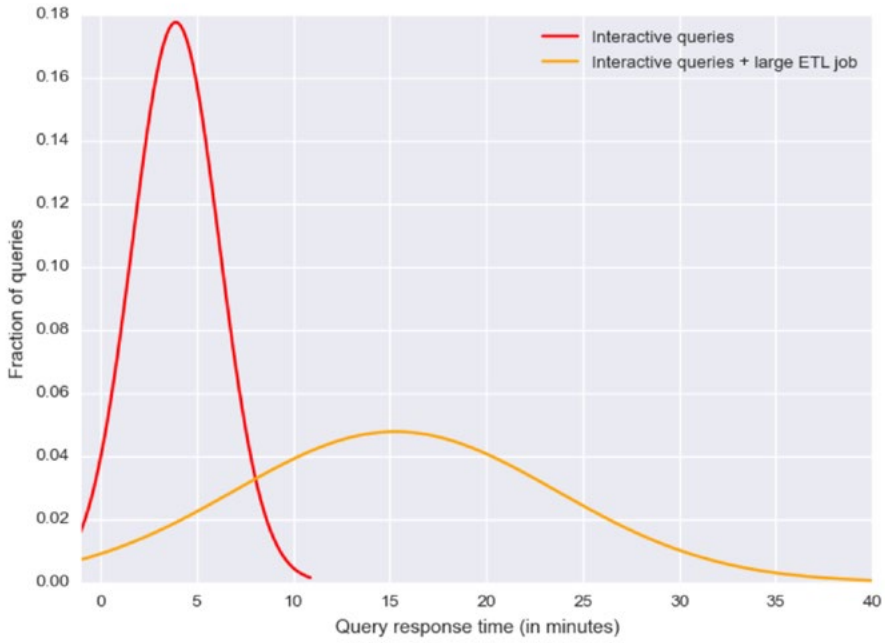
BENCHMARK TEST

Because serverless is such a new technology and approach to computing there are not a lot of existing benchmarks and performance evaluations to guide users in knowing whether this is the right tool for the job. Greg Owen, Eric Liang, Prakash Chockalingam and Srinath Shankar on the Databricks engineering team conducted a performance evaluation to show how Databricks serverless performs in common situations vs a standard cluster.

SCENARIO 1 - STANDARD CLUSTER:

Many data scientists are running Spark queries on a standard cluster. These are short-running interactive jobs that last at most a few minutes. Now a large ETL workload is introduced to the same cluster.

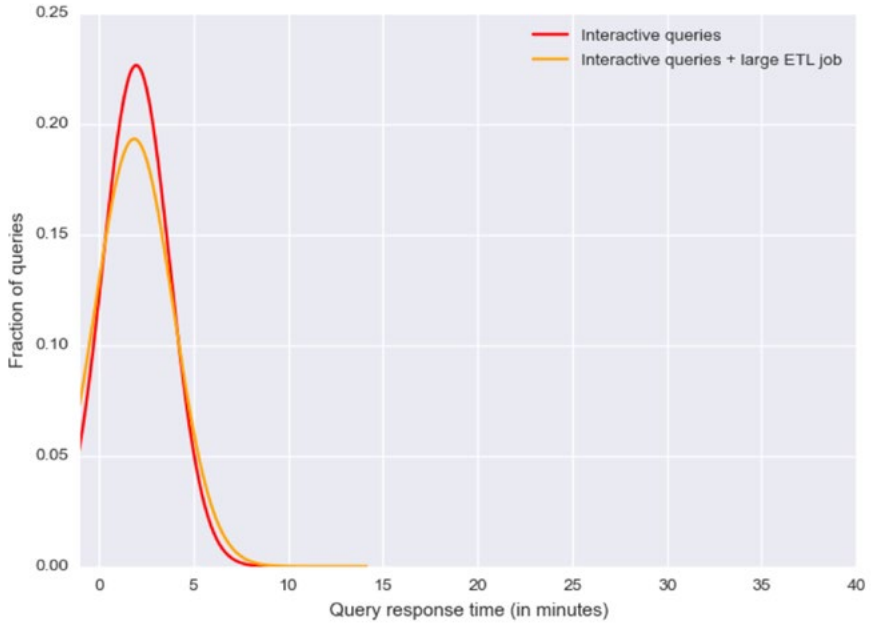
In this scenario that is shown in the graph below there are 20 users on a standard cluster. When the ETL jobs are added, average response times increase from 5 minutes (red line) to 15 (orange line), and in the worst case more than 40 minutes.



SCENARIO 2 - SERVERLESS POOL

There are again 20 users, this time on a serverless pool. The ETL jobs are introduced and the interactive queries get a little slower when the jobs start. However, the Databricks scheduler is able to guarantee performance isolation and limit their impact. The ETL jobs run in the background, efficiently utilizing idle resources.

In this scenario shown in the graph below users get excellent performance for both workloads without having to run a second cluster.



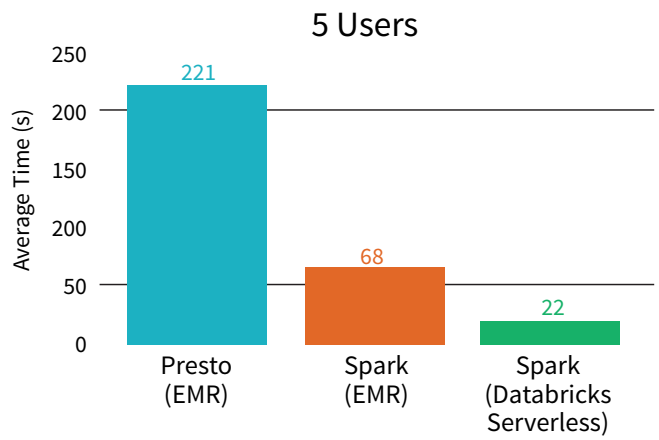
COMPARISON WITH OTHER SYSTEMS

The team also decided to compare the performance of larger, concurrent TPC-DS workloads running in three different environments:

1. Presto on AWS
2. Apache Spark on AWS
3. Databricks Serverless

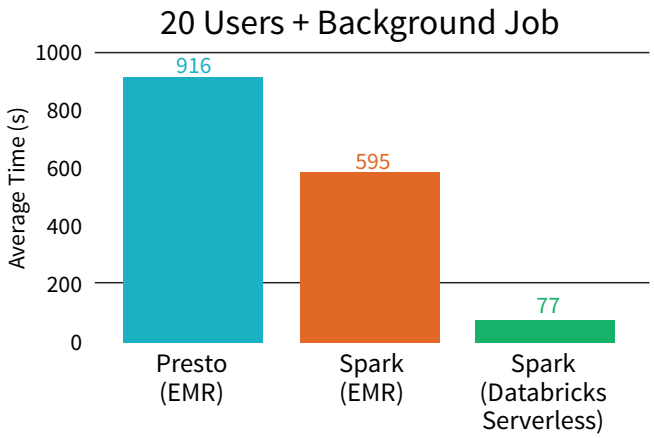
SCENARIO 1: 5 USERS ON A CLUSTER

The results in the graph below shows that with five users each running a TPC-DS workload concurrently on the cluster, the average query latencies for Databricks Serverless pools were 75% less than Spark on AWS and an order of magnitude lower than Presto on AWS.



SCENARIO 2: 20 USERS ON A CLUSTER WITH A BACKGROUND JOB

When 20 users were on the cluster and and a background ETL job was also running Databricks Serverless performed even better than in Scenario 1 which is shown in the graph below. In this performance test on the cluster Databricks is 12 times faster than Presto on AWS, and 7 times faster than Spark on AWS.



As the performance evaluation shows Databricks serverless is highly performant. It takes away the guesswork of cluster management, which is time consuming and defocuses teams from their core tasks. With serverless a team can just set the minimum and maximum size of a pool and it will automatically scale within those bounds to adapt to the load being placed on it.

CONCLUSION

In this ebook we've looked at different independent performance benchmarks with The Databricks Unified Analytics Platform. Each of the tests showed how different aspects of the platform perform better than older versions and other technologies and implementations that are available. We've also looked at how functionality and performance can be applied in real world use cases across a variety of different verticals.

BENCHMARKS COVERED IN THIS EBOOK:

- Runtime performance - a performance standard from the Transaction Processing Performance Council (TPC) the TPC-DS benchmark is used to analyze the performance of decision support systems.
- R workflows - benchmark that shows accelerated common R workflows with improvements included in Databricks Runtime 3.0.
- Streaming - to test throughput, latency and completeness of streaming big data platforms.
- Serverless pools - benchmark test to demonstrate how the serverless pools fare when there is a concurrent and heterogeneous load.

ADDITIONAL RESOURCES

- Databricks Serverless Keynote and Demo - a live demo of Databricks Serverless performed at Spark Summit.
- Benchmarking Big Data SQL Platforms Webinar - a webinar featuring Databricks Co-founder and Chief Architect Reynold Xin.

Market Insights: Hotels.com

Hotels.com is a premier website for booking accommodations online with 85 websites in 34 languages, listing over 325,000 hotels in approximately 19,000 locations. To reach their goal of becoming more 'algorithm-focused' to deliver real-time customer personalization, Hotels.com required massive compute and analytics capabilities to ensure a targeted and satisfying customer experience when booking travel

THE CHALLENGES

- **Build a more robust and faster data pipeline:** On-premises Hadoop cluster using SQL and SAS to do data science at scale was slow and limiting – taking 2 hours to process the data pipeline on only 10% of the data.
- **Leverage machine learning to drive consumer experience:** Massive volume of image files corresponding to each property listing included duplicates and lacked organization for ranking and classification. Needed to build in real-time scoring and become more efficient at deploying machine learning models into production.
- **Increase customer conversions:** Being able to understand customer trends in real-time to develop strategies to drive conversion and lifetime value.

THE SOLUTION

Databricks has helped Hotels.com to realize its goal of becoming "data science focused" so that they can anticipate customer behavior and provide a more optimized user experience

- **Databricks Runtime:** Increase processing performance of streaming data even at scale.
- **Cluster Management:** Able to scale volume of data significantly without adding infrastructure complexity.
- **Interactive Workspace:** Foster a culture of collaboration among data science teams within Hotels.com as well as other business units within Expedia.

THE RESULTS

- **Accelerate ETL at scale:** Able to increase the volume of data processed by 20x without impacting performance.
- **Optimized user experience:** Highly accurate and effective display of images within the context of property searches by customers.
- **Increased sales efficiency:** Providing the right hotel with the right images based on searches has resulted in higher conversions.

"Agility and flexibility were critical for us to successfully support our data science and engineering goals. Moving to Databricks' Unified Analytics Platform to run 100% of our workflows has been a huge boost for our business and our customers."

MATT FRYER
VP, CHIEF DATA SCIENCE OFFICER, HOTELS.COM

Market Insights: 

Shell is a recognized pioneer in oil and gas exploration and production technology and one of America's leading oil and natural gas producers, gasoline and natural gas marketers and petrochemical manufacturers.

THE CHALLENGES

- **Disjointed Inventory Distribution:** Stocking practices are often driven by a combination of vendor recommendations, prior operational experience and "gut feeling".
- **Limited DSS Availability:** There has been limited focus directed towards incorporating historical data and doing advanced analysis to come up with decisions.
- **Lost Business Agility:** This can lead to excessive or insufficient stock being held at Shell's locations, like oil rigs which has significant business implications.

THE SOLUTION

Databricks provides Shell with a cloud-native unified analytics platform that helps with improved inventory and supply chain management:

- **Databricks Runtime:** The team to dramatically improved the performance of the simulations.
- **Interactive Workspace:** The data science team is able to collaborate on the data and models via the interactive workspace.
- **Cluster Management:** Significant reduction in total cost of ownership by moving to the Databricks cloud solution and gains in operational efficiency.
- **Automated Workflows:** Using analytic workflow automation, Shell is easily able to build reliable and fast data pipelines that allow them to predictive when to purchase parts, how long to keep them, and where to place inventory items.

THE RESULTS

- **Predictive Modeling:** Scalable predictive model is developed and deployed across more than 3,000 types of materials at 50+ locations.
- **Massive Performance Gains:** With a focus on improving performance the data science team reduced the inventory analysis and prediction time to 45 minutes from 48 hours on a 50 node Spark cluster on Databricks — a 32X performance gain.
- **Reduced Expenditures:** Cost savings equivalent to millions of dollars per year.

"We were able to take a tool that previously would have been fairly localised to a single region and turn that into a global product which actually is now becoming the foundation for the way our inventory analysts will now do their work."

DANIEL JEAVONS
GENERAL MANAGER - ADVANCED ANALYTICS COE, SHELL

Market Insights: nielsen

Nielsen is a leading global, independent measurement and data company for fast-moving consumer goods, consumer behavior, and media. With a presence in more than 100 countries and services covering more than 90% of the globe's GDP and population. Nielsen's Marketing ROI information services links media consumption data with purchase behavior to provide clients with the insights they need to make smarter marketing investments.

THE CHALLENGES

- **Flexible and efficient use of resources:** Working on-premise, under a fixed cluster architecture restriction required the use of multiple technologies that needed to integrate well to perform optimally.
- **Ensure security and integrity:** Data sensitivity and restrictions required the creation of an on-premises data analytics environment to support their consumer facing customers that are cloud averse.
- **Consistency and performance:** Use of existing analytics system was not compatible with the restrictions of the required on-premises environment which caused massive slowdowns.

THE SOLUTION

Databricks has empowered Nielsen to build a recommendation engine that helps their customers make insights-driven decisions.

- **Big data analytics, big money:** Ability to scale their analytics to support over 4,000 marketing and advertising campaigns and their corresponding data volumes, with $30 billion worth of sales data.
- **Improved online advertising metrics:** 4 billion digital impressions measured with customer-targeted, personalized ads.
- **Effective adoption of DevOps:** Databricks Unified Analytics Platform supported their continuous integration development practice ensuring any modifications to code running in the cluster are seamlessly integrated.

THE RESULTS

- **Reduced infrastructure costs:** Nielsen reduced its SAS footprint by 40% for an annual savings of $2 million.
- **Reduced operational costs:** Lowered compute usage by 45%, greatly reducing operational expenses.
- **Faster performance of models:** Reduced the runtime of machine learning models from 36 days to 4 hours.
- **More accurate models:** Improved the predictive accuracy of their models by 33%.
- **Rapid adoption across teams:** Increased Databricks users from 9 to 156 users due to improved collaboration and productivity.

"Databricks' has allowed us to provide much more reliable attribution to marketing events for a clearer understanding of what drive sales allowing them to refocus investment of their digital media buys."

JOSEPH DE CASTELNAU
SR. VP OF SOFTWARE ENGINEERING, NIELSEN

Market Insights: kik

Kik is a messaging platform, already at 300 million users and is incredibly popular in the youth market, where it is growing fastest among teenagers. Kik doesn't use phone numbers, but instead, usernames for friends to connect and chat with one another. Based in Waterloo, Ontario, this fast startup is valued at over $1B.

THE CHALLENGES

- **Limited Personnel Resource:** Have to compete with giants like Google and Facebook with small team.
- **Poor Choices for Engineering:** Developing fast came at the expense of scalable engineering.
- **Scalability Volumes a Concern:** Scaling to 300 million users and anticipating future fast growth.
- **Disjointed Data Science Efforts:** Multiple systems and tools for data analysis led to complexity and lack of standard.
- **Overload of Data:** Amazon Redshift was starting to break at the seams with 300 data pipelines with 5TB of new data per day.

THE SOLUTION

Databricks has allowed Kik to improve team productivity while significantly reducing data engineering overhead at scale:

- **Support for Multiple Languages:** Familiarity with SQL - finding expertise in SQL is much more cost effective than other languages.
- **Performant and Reliable Data Pipelines:** Ability to handle 5TB new data per day streaming into the data lake.
- **Fully Managed Platform:** Managed Spark takes the burden off of needing Spark expertise and having to understand the internals of Spark.
- **Collaborative Workspace:** Streamlined data analysis and fostered collaboration through support for multiple languages, easy search, and real-time commenting.

THE RESULTS

- **Massive Improvement in Development Demands:** Data engineering efforts reduced by 70%.
- **Performance Gains for Jobs:** Big jobs now run 2x faster.
- **Improved Efficiency of Workflows and Collaboration:** Sharing data is instant via links to notebooks vs putting together a powerpoint.
- **Better Data Analysis:** Ability to combine and ensure clean data to trust for analysis.

"Moving to Databricks was a huge part of easing a lot of the pain of running a 300 node cluster on EMR; we moved those exact same jobs to Databricks and it ran in half the time."

JOEL CUMMING
SENIOR PRINCIPAL DATA SCIENTIST

Try Databricks for free

databricks.com/try-databricks

Contact us for a personalized demo

databricks.com/contact-databricks

About Databricks:

Databricks' mission is to accelerate innovation for its customers by unifying Data Science, Engineering and Business. Founded by the team who created Apache Spark™, Databricks provides a Unified Analytics Platform for data science teams to collaborate with data engineering and lines of business to build data products. Users achieve faster time-to-value with Databricks by creating analytic workflows that go from ETL and interactive exploration to production. The company also makes it easier for its users to focus on their data by providing a fully managed, scalable, and secure cloud infrastructure that reduces operational complexity and total cost of ownership. Databricks, venture-backed by Andreessen Horowitz and NEA, has a global customer base that includes, Salesforce, Viacom, Amgen, Shell and HP. For more information, visit www.databricks.com.