pepperdata

Pepperdata Reduces the Cost of Running Spark Workloads on Amazon EKS at Scale by 41.8%

Benchmark Results Using a TPC-DS Model on Amazon EKS at Scale
October 2023

TPC-DS Benchmarking

TPC-DS is the Decision Support framework from the Transaction Processing Performance Council. TPC-DS is an industry-standard big data analytics benchmark. Pepperdata's work is an unofficial benchmark as defined by TPC.

Using a 1 TB dataset on 500 nodes with 275 executors and 99 TPC-DS queries running on Amazon EKS, Pepperdata found that Capacity Optimizer:

- **Improved price-performance**
  Decreased instance hours consumed by 41.8 percent
- **Increased throughput**
  Decreased total workload run time by 45.5 percent

Special, Limited Time Offer

Get started today with a free, 2-day waste assessment of your Amazon EKS environment. Visit pepperdata.com or contact us at sales@pepperdata.com.

Supported Technologies

- Amazon EKS
- Amazon EMR for EKS
- Apache Spark

Introduction

A recent survey conducted by Pepperdata identified "significant or unexpected spend" as the top challenge to Kubernetes adoption. Pepperdata Capacity Optimizer has improved resource utilization and optimized price/performance in some of the most complex and highly-scaled enterprises around the globe. Because each enterprise environment is different, Pepperdata embarked upon third-party benchmarking to provide an objective standard against which to measure the true power of Capacity Optimizer in reducing unexpected spend on Amazon EKS on a large-scale cluster.

Standardized Benchmark Methodology

Pepperdata selected TPC-DS, the decision support benchmark from the Transaction Processing Performance Council, as the standardized workload model and used a GitHub repository recommended by Amazon Web Services (AWS) that closely adheres to the TPC-DS model for benchmarking purposes. Pepperdata ran this benchmark "out of the box" and did not modify or recompile them using any special libraries. (Note: This work is not an official audited benchmark as defined by TPC.)

Benchmark Environment and Execution

Pepperdata set up a Spark benchmarking test environment on Amazon EKS to test Pepperdata Capacity Optimizer for large-scale workloads according to the following parameters:

- Workload: TPC-DS with Spark SQL (all 99 queries) running 10 parallel applications (with 275 executors per application) to simulate a multi-tenant environment
- Data volume: 1 TB
- Static node count: 500
- Amazon EKS version: 1.24
- Apache Spark version: 3.1.2
- Instance type: r5.2xlarge
- Autoscaling: No
- Scheduler: YuniKorn

Pepperdata ran this 1 TB Spark SQL Workload running 10 parallel Spark applications both with and without Capacity Optimizer to capture the "before and after" effect of Capacity Optimizer.

Key Performance Metrics

Pepperdata used the following metrics to assess the performance and effectiveness of Capacity Optimizer:

- Total instance hours consumed
- Total workload run time
- Average run time per application
- Average concurrent container count

All of these metrics were obtained through the Pepperdata dashboard. The Pepperdata dashboard gathers and aggregates hundreds of cluster metrics in near real time to provide visibility into the performance of all applications running in a cloud environment via a single pane of glass.

Summary of Key Findings

Using the standardized workload and benchmark environment described above, Pepperdata measured the following:

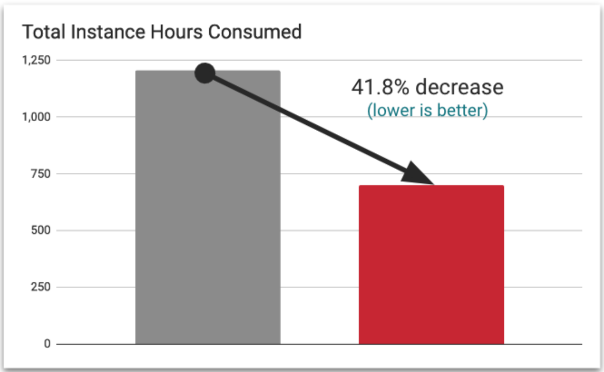

Figure 1: **Total Instance Hours Consumed Decreased by 41.8%.**

The 41.8 percent reduction that Capacity Optimizer achieved translates to reduced cost, since cloud pricing is directly correlated with instance hour utilization.

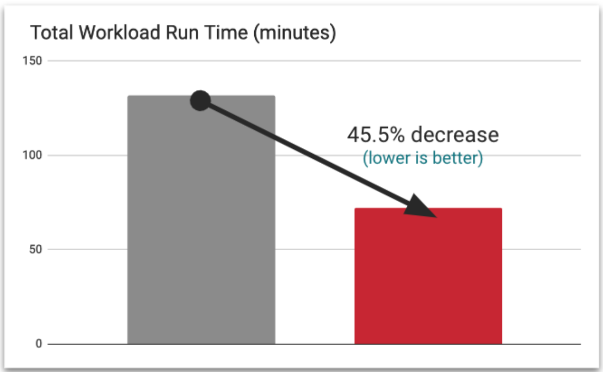

Figure 2: **Total Workload Run Time (minutes) Decreased by 45.5%.**

The 45.5 percent reduction that Capacity Optimizer achieved means that workloads can complete faster.

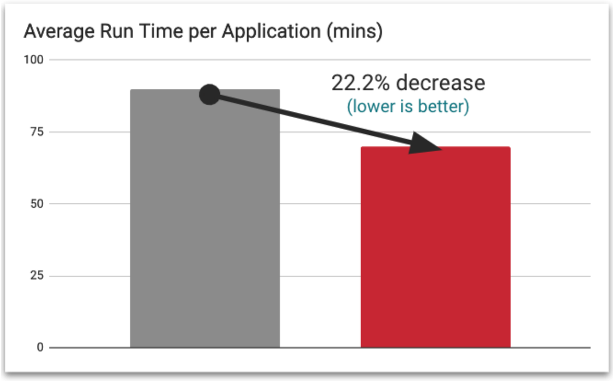

Figure 3: **Average Run Time per Application Decreased by 22.2%.**

Not only did Pepperdata Capacity Optimizer decrease the run time of the entire workload by 45.5 percent, it also reduced the run time of the individual Spark applications by 22.2 percent on average.

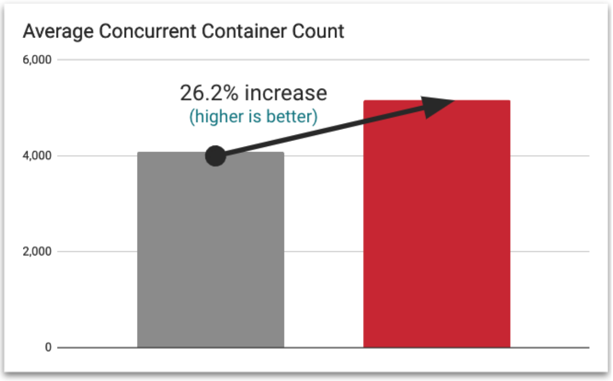

Figure 4: **Average Concurrent Container Count Increased by 26.2%.**

By increasing the average concurrent container count by 26.2 percent, Pepperdata Capacity Optimizer increased the ability of a cloud environment to process applications and thereby improves overall throughput.

Additionally, Pepperdata Capacity Optimizer also improved average memory and core utilization, which enables greater throughput. Average memory utilization increased 6.4 percent while average core utilization increased 87.5 percent, a significant uplift.

Figure 5 below visualizes the core utilization uplift achieved by Capacity Optimizer, which added what was effectively hundreds of unpaid cores to this cluster, allowing the workloads to run faster and at lower cost.

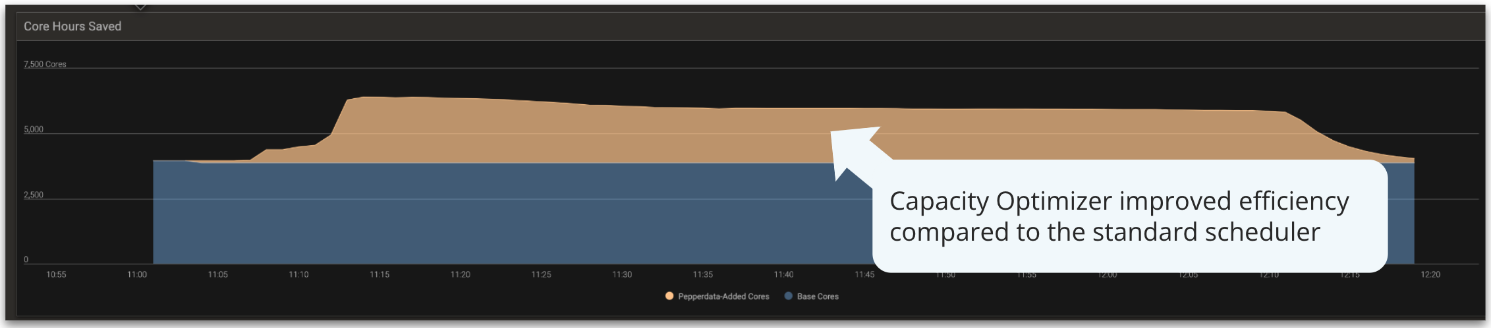

Figure 5: A visualization of the improved cluster performance after enabling Capacity Optimizer.

In summary, running the TPC-DS based benchmark workload in a 500 node Amazon EKS environment both with and without Pepperdata Capacity Optimizer demonstrated the significant cost savings and the efficiency uplift delivered by Capacity Optimizer at scale. Improvements in key performance metrics such as total instance hours consumed, application run times, and concurrent container count all reflect the power of Capacity Optimizer to decrease the cost of running applications in the cloud. For example, by decreasing instance hours consumed by 41.8 percent, Capacity Optimizer directly reduced the cost of the benchmark environment. And by increasing average concurrent container count by 26.2 percent, Capacity Optimizer enhanced resource utilization and therefore enabled greater throughput without requiring additional resources.

Conclusion

Pepperdata's benchmarking work demonstrates that running a simulated TPC-DS industry-standard Spark workload on Amazon EKS at scale with Pepperdata Capacity Optimizer demonstrated significant cost savings and performance improvements. **Capacity Optimizer reduced the total instance hours and related costs by 41.8 percent** (from 1,208 hours to 703 hours) and **enabled the entire workload to run 45.5 percent faster** (as measured by decreased run time). In addition, Capacity Optimizer increased the average concurrent container count by 26.2 percent, meaning that more applications were able to run in a given time period, which increased the overall throughput of the environment.

As more companies migrate workloads to the cloud and Kubernetes, these findings have important implications for FinOps activities and cost and resource management. Capacity Optimizer's ability to autonomously optimize workloads such as Apache Spark on Amazon EKS helps ensure that cloud resources are used efficiently and cost-effectively and makes the cloud an even more attractive and viable option for Spark and other high-performance workloads.

If autonomously reducing the cost of running your Spark applications by 41.8 percent or more and enabling your environment to run more than 45 percent faster sounds compelling, you can learn more here or explore a customized waste assessment to see Capacity Optimizer at work in your environment.

About Pepperdata

Pepperdata is the only cost optimization solution that delivers up to 47% greater cost savings—continuously and in real-time—on Amazon EMR and EKS with no application changes or manual tuning. Our customers include the largest, most complex, and highly-scaled clusters in the world, at top enterprises such as Citibank, T-Mobile, Autodesk, Securonix, Royal Bank of Canada, and those in the Fortune 5. For more information, visit pepperdata.com.

Pepperdata, Inc.
530 Lakeside Drive
Suite 170
Sunnyvale, CA 94085

Start a Free PoV
www.pepperdata.com

Send an Email
eval@pepperdata.com