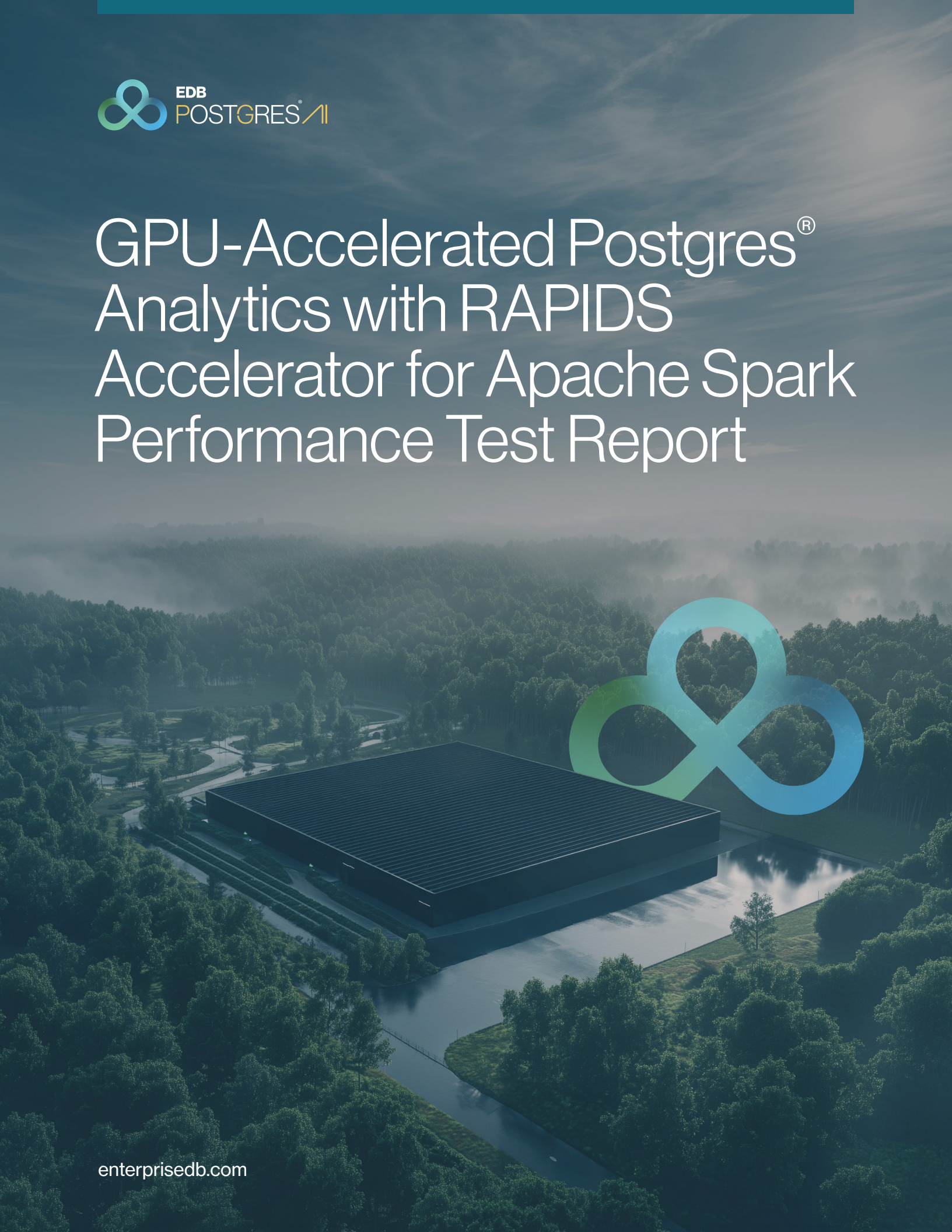
EDB POSTGRES AI

GPU-Accelerated Postgres® Analytics with RAPIDS Accelerator for Apache Spark Performance Test Report



enterprisedb.com

Contents

1. Overview

This document represents a comprehensive comparative performance benchmark of two query processing systems, PostgreSQL and, from EnterpriseDB (EDB), Postgres Analytics Accelerator (PGAA). We analyze their respective abilities to execute a standard analytical processing/decision support benchmark as defined by the TPC Benchmark™ DS (TPC-DS) specification's Power Test. Our testing measures the ability to execute queries at reasonable rates and latency with the respective systems across various scale factors.

In the configuration under test, EDB PGAA offloads analytical PostgreSQL queries to two variants of Apache Spark: vanilla Spark and Apache Spark accelerated by NVIDIA cuDF.

2. Executive summary and key findings

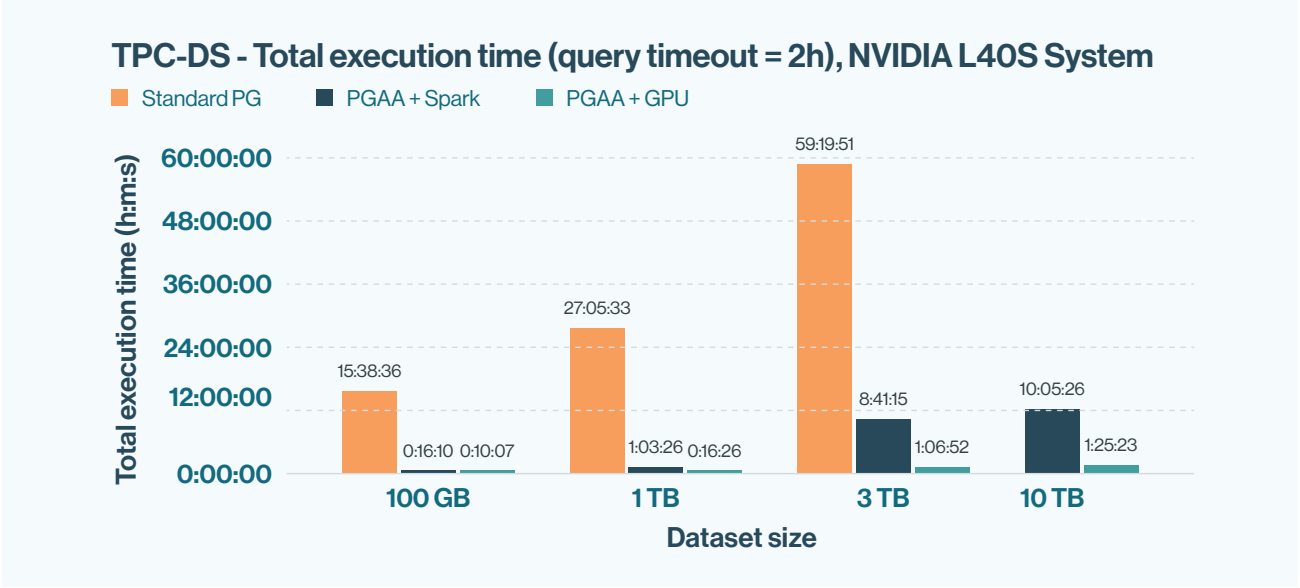
On TPC-DS with dataset sizes of 100GB, 1TB and 3TB (scale factors 100, 1000 and 3000, respectively), EDB PGAA with spark-rapids outperforms PostgreSQL transactional tables, completing the full workload about 50-100 times faster.

On dataset sizes below 3TB, most of the performance improvement comes from using PGAA to offload queries to Spark, with RAPIDS GPU acceleration providing a minor contribution on top of Spark to the query performance. For larger datasets, we attribute the majority of performance improvement to spark-rapids.

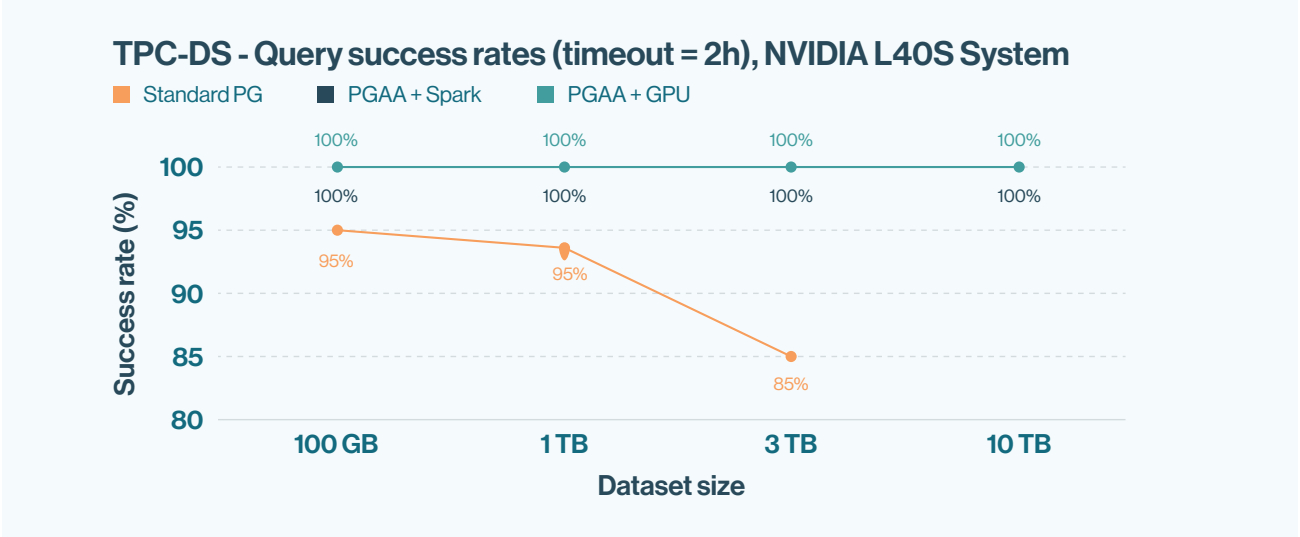
This holds even after more in-depth index tuning of the PostgreSQL system. With indexes, we managed to improve the performance of PostgreSQL on the 1TB dataset by 10%, however this came at a cost of slowing some queries down by as much as 29x. In comparison, EDB PGAA with spark-rapids required minimal configuration out of the box and no need for creating indexes.

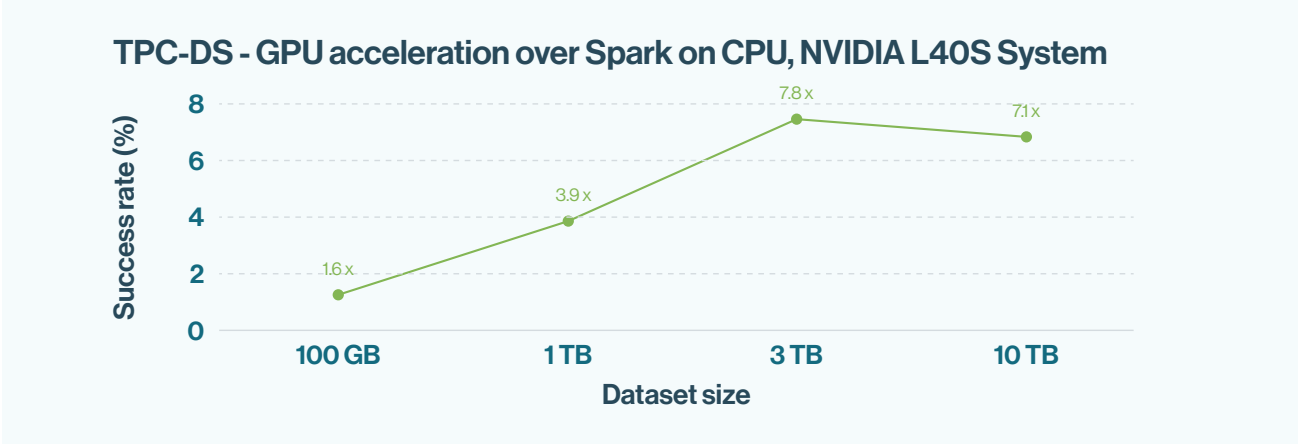
This showcases that EDB PGAA is vital for making PostgreSQL ready for the era of conversational and agentic analytics, where the exact query that a user or an agent will run is hard to predict in advance and optimize the database system for. EDB PGAA can add value not only in terms of improving raw analytical query performance but also in reducing the maintenance overhead of tuning the database system.

2.1. L40S system and PostgreSQL timings



On PostgreSQL, several queries do not complete within the allotted two-hour timeout, and that number grows with the scale factor. The times for these were clipped at 7200s (2h).





We observe that the performance gap between the GPU and CPU on the L40S system narrows at dataset sizes over 10 TB. We estimate that this is due to our reaching the I/O limits of a single machine.

3. Test description

3.1. Tested software

Apache Spark is an engine for executing massively parallel data engineering and data science jobs on single-node machines or clusters.

EDB PostgreSQL Analytics Accelerator (PGAA) is a proprietary extension for PostgreSQL developed by EDB. It offloads queries on datasets in popular data lake formats (Delta, Apache Iceberg, Apache Parquet) to an external analytics query engine such as Seafowl (based on Apache DataFusion) or Apache Spark (using Spark Connect). When combined with EDB Postgres Distributed (EDB's distributed solution for highly available Postgres), it can replicate PostgreSQL tables to Apache Iceberg, enabling near-real-time analytics on PostgreSQL data.

NVIDIA RAPIDS Accelerator for Apache Spark is a Spark plug-in maintained by NVIDIA that accelerates Spark query processing using GPUs.

For this test, EDB PGAA was configured to send queries to a Spark Connect endpoint (Spark 3.5.6) with NVIDIA spark-rapids 2025.10 installed.

3.2. System under test configuration

System name	Logical processors	CPU type	Memory (GB)	GPU	Storage
L40S System	240	2x Xeon Platinum 8580 (60 physical cores, 2.0 GHz base clock, and 4.0 GHz max, 300 MB L3 cache)	2048	8xL40S	56 TB locally attached NVMe

PostgreSQL settings (postgresql.conf) are as follows:

```
checkpoint_timeout = 30min
default_statistics_target = 10000
effective_cache_size = 200GB
effective_io_concurrency = 200
maintenance_work_mem = 1GB
max_parallel_maintenance_workers = 128
max_parallel_workers = 240
max_parallel_workers_per_gather = 128
max_wal_size = 500GB
max_worker_processes = 240
random_page_cost = 1.1
shared_buffers = 32GB
work_mem = 32GB
```

3.3. Test scenarios

The test scenarios aim to analyze running the TPC-DS benchmark suite against several systems:

1. **Vanilla PG18 (tuned):** This setup is running queries against PostgreSQL heap tables, with PostgreSQL configured as above and extra indexes created to optimize the system for TPC-DS queries.
2. **PG18 with EDB PGAA + Spark:** This setup is running queries against PostgreSQL 18, which then leverages PGAA with a Docker Compose Spark cluster that is configured to only use CPUs.
3. **PG18 with EDB PGAA + spark-rapids:** This setup is running queries against PostgreSQL 18 which then leverages PGAA with a Docker Compose Spark cluster with the spark-rapidsplugin v25.10 and specific Spark configuration enabled.

##	Description
T1	Vanilla PG18 with indexes and GUCs using Scale Factor 100 (100 GB)
T2	PG18 with EDB PGAA and Spark using Scale Factor 100 (100 GB)
T3	PG18 with EDB PGAA and **spark-rapids** v25.10 using Scale Factor 100 (100 GB)
T4	Vanilla PG18 with indexes and GUCs using Scale Factor 1000 (1 TB)
T5	PG18 with EDB PGAA and Spark using Scale Factor 1000 (1 TB)
T6	PG18 with EDB PGAA and Spark with NVIDIA spark-rapids v25.10 using Scale Factor 1000 (1 TB)
T7	Vanilla PG18 with indexes and GUCs using Scale Factor 3000 (3 TB)
T8	PG18 with EDB PGAA and Spark using Scale Factor 3000 (3 TB)
T9	PG18 with EDB PGAA and Spark with NVIDIA spark-rapids v25.10 using Scale Factor 3000 (3 TB)
T10	PG18 with EDB PGAA and Spark using Scale Factor 10000 (10 TB)
T11	PG18 with EDB PGAA and Spark with NVIDIA spark-rapids v25.10 using Scale Factor 10000 (10 TB)

3.4. Test workload

To execute performance tests against PostgreSQL, EDB built scripts that automates the various steps involved in the benchmark: load dataset, setup and run Spark cluster in Docker Compose, build indexes for PG, and run the queries. The dataset used for the tests is TPC-DS, an industry standard benchmark for evaluating the performance of complex data warehousing and analytics systems.

The indexes were made by an EDB performance expert. All tests were run on an NVIDIA system in their Launchpad environment.

We used the dsdgen utility from the TPC-DS Tools suite to generate the TPC-DS data in a textual format. For PostgreSQL, we loaded the data into the tables directly. For Spark tests, we exported the flat files into Parquet and configured external tables with EDB PGAA to query those Parquet files.

After the database build is complete, the 99 queries are run against the three systems (PG18, EDB PGAA + Spark, EDB PGAA + spark-rapids).

For the Spark cluster configurations, Spark 3.5.6 was used with 16 executors for CPU only configuration and 8 executors for GPU spark-rapids configuration. Both configurations had one Spark worker with 16 CPU cores and 64GB for each executor. The cluster was built and run in a Docker Compose environment.

For the Vanilla PG18 configuration, all 240 processor threads and roughly 2TB of RAM from the instance were available.

Running the tests required:

- A directory with the 99 TPC-DS queries, templated such that the same queries ran on PostgreSQL and on EDB PGAA.
- A benchmark script that can be configured to run the queries against the Spark cluster or only against Postgres. The scale factor and table schema also needed to be defined.

In the interest of completing the test expediently, we set a statement timeout on all workloads to 2h. As the report will show later, PostgreSQL cannot plan certain DS queries correctly or leverage CPU-level parallelism. Without the timeout, this would have resulted in execution times of hours to days at larger dataset sizes for those queries.

3.4.1. PostgreSQL

DBT-7, an open source, fair-use implementation of the TPC-DS, was used to execute the load and Power Test.

Instructions for getting and using the test kit are available online at osdldbt.github.io/dbt7.

Example executing the Load Test and Power Test at scale factor 1000 that collects EXPLAIN ANALYZE output:

```
dbt7 run -d postgresqlea -s 1000 --power pgsql /tmp/results
```

4. Test results: L40S GPU

The results in this section are structured as follows. For each dataset size, we first discuss and explain the speedup between vanilla PostgreSQL and PostgreSQL with PGAA and spark-rapids. Then, to measure and isolate the effect of GPU acceleration, we compare the performance of PGAA with spark-rapids to that of PGAA with standard Spark.

4.1. 100 GB

spark-rapids GPU speedup	vs. tuned PG	vs. Spark CPU
Avg	92.79x	1.60x
Median	5.79x	1.46x
Min	-16.23x	-1.85x
Max	2400.00x	6.77x
Stddev	389.79x	1.02x
Geometric mean	7.20x	1.53x
Num queries	99	99
Num speedups	82	95
% of queries with speedups	82.83%	95.96%
	Number of queries that failed	**Slowest query (seconds)**
Tuned PG	5	7200
Spark GPU	0	35
Spark CPU	0	50
	Average speedup	
Spark GPU vs. Tuned PG	93x	
Spark GPU vs. Spark CPU	2x	

4.1.1. EDB PGAA + spark-rapids vs PostgreSQL 18

While spark-rapids maintained a significant lead overall, the smaller data scale allowed PostgreSQL to perform competitively on specific short-running queries.

spark-rapids achieved an overall speedup of 93x over Postgres, heavily skewed by queries where PostgreSQL timed out or took exceptionally long. In 82/99 queries, spark-rapids was faster than PostgreSQL.

PostgreSQL struggled with the most complex analytical queries even at this smaller scale. PostgreSQL recorded “DNF” (did not finish) or extreme runtimes for queries like Q1 and Q14 (7200s timeouts), whereas spark-rapids finished them in seconds (6.5s and 4.4s respectively).

4.1.2. EDB PGAA + spark-rapids vs EDB PGAA + Spark

For smaller datasets like the 100GB, the ROI of GPU acceleration is not as impressive compared to 1TB and 3TB. While it still provides a 50% performance boost on average and speeds up 96% of queries, the “fixed costs” of distributed GPU computing dampen the gains.

4.2. 1 TB

spark-rapids GPU speedup	vs. Tuned PG	vs. Spark CPU
Avg	98.94x	3.86x
Median	14.26x	2.74x
Min	-143.30x	1.11x
Max	1100.52x	12.66x
Stddev	227.73x	1.88x
Geometric mean	15.78x	2.92x
Num queries	99	99
Num speedups	91	99
% of queries with speedups	91.92%	100.00%
	Number of queries that failed	**Slowest query (seconds)**
Tuned PG	7	7200
Spark GPU	0	55
Spark CPU	0	726
	Average speedup	
Spark GPU vs. Tuned PG	99x	
Spark GPU vs. Spark CPU	4x	

4.2.1. EDB PGAA + spark-rapids vs PostgreSQL 18

At this dataset size, spark-rapids begins to demonstrate an overwhelming performance advantage over PostgreSQL 18, consistently outperforming it by an order of magnitude on the majority of queries.

spark-rapids achieved a total speedup of 99x over Postgres, driven by several extreme outliers where PostgreSQL struggled significantly. It was faster than PostgreSQL in 91/99 of the executed queries.

PostgreSQL failed to complete several queries within the 2-hour timeout (7200 seconds), resulting in “DNF” (did not finish) statuses. For example, spark-rapids completed Query 1 in 6s and PostgreSQL timed out after 2h, resulting in an observed speedup of 1100x (which is an underestimation of the real speedup given the timeout settings).

In addition, to estimate the hidden total cost of configuring a PostgreSQL system to support analytics queries, we performed a round of tuning after looking at the initial performance numbers, recording the time it took us and the difference in performance.

We spent 1 person-days inspecting the query plans for the slowest queries on PostgreSQL, adding suitable indexes and rerunning the queries, which resulted in an overall 10% speedup (before tuning: 11 queries timed out after 2h, total time 31 hrs; after tuning: 9 queries timed out after 2h; total time ~27 hrs). However, after index tuning, some queries ran slower on PostgreSQL. For example, the runtime for Query 17 went up 13x from 85s to 1141s, and the runtime for Query 72 went up 29x from 122s to nearly an hour.

4.2.2. EDB PGAA + spark-rapids vs EDB PGAA + Spark

At this scale factor, considering the speedup of spark-rapids over vanilla Spark, we also begin to observe a larger contribution from GPUs towards the performance of the system as the impact of fixed overheads decreases.

Overall, the GPU-accelerated Spark cluster was 3.8x faster than the CPU-only cluster (up from 1.6x on 100 GB). The acceleration was universal, with 100% of queries showing performance improvements when running on GPUs.

50% of all queries were accelerated by 2.74x or more with spark-rapids (compared to 1.46x on the 100 GB dataset).

4.3 3 TB

spark-rapids GPU speedup	vs. Tuned PG	vs. Spark CPU
Avg	53.24x	7.80x
Median	33.26x	7.46x
Min	-2.07x	-103.x
Max	382.73x	24.11x
Stddev	89.39x	5.63x
Geometric mean	35.30x	6.65x
Num queries	99	99
Num speedups	98	98
% of queries with speedups	98.99%	98.99%
	Number of queries that failed	**Slowest query (seconds)**
Tuned PG	15	7200
Spark GPU	0	311.57
Spark CPU	0	2651
	Average speedup	
Spark GPU vs. Tuned PG	53x	
Spark GPU vs. Spark CPU	8x	

4.3.1. EDB PGAA + spark-rapids vs PostgreSQL 18

At 3 TB, the effect of “clipping” PostgreSQL times to 2h becomes very pronounced and impacts the overall speedup negatively (53x on 3TB vs 99x on 1TB). The issue is that at larger dataset sizes, more PostgreSQL query times become bounded at the 2h ceiling whereas the total runtime of queries on Spark continues to grow.

This means that the overall speedup figure is not informative for representing how long the queries on PostgreSQL would run if we didn’t terminate them at 2h.

To gauge the scalability of spark-rapids and the true performance improvement, it is more informative to look at the speedup of a median query: 50% of queries were accelerated by 33x or more (at 1 TB: 14x, at 100 GB: 5.8x). Almost all queries were accelerated by spark-rapids, with the slowest query completing in 5 minutes, whereas 15/99 queries on PostgreSQL were timed out after 2 hours.

This suggests that the ROI of using spark-rapids increases at larger dataset sizes.

4.3.2. EDB PGAA + spark-rapids vs EDB PGAA + Spark

spark-rapids accelerates the median query by 7.5x, compared to vanilla Spark (1TB: 2.7x; 100GB: 1.5x).

As opposed to smaller dataset sizes, where the majority of the performance improvement over PostgreSQL came from using EDB PGAA to run analytical queries on Spark, at this dataset size, we can establish that the majority of the performance improvement comes from GPU acceleration and not from Spark itself.

4.4. 10 TB: EDB PGAA + spark-rapids vs EDB PGAA + Spark

We did not think that it was worthwhile to test PostgreSQL with dataset sizes this large. Extrapolating from our results for the 3 TB dataset, we estimate the benchmark would take 4 days to run, with 29/99 queries timing out after 2h.

The total time for Spark CPU to complete all queries was 36,326s. For spark-rapids, it was 5,123s, 7x faster than vanilla Spark. Interestingly, this is a 1.2x and a 1.3x increase in the total runtime, respectively, despite scaling the dataset size by 3x.

At this scale factor, we suspect we are reaching a saturation point: the performance gap between CPU and GPU execution becomes 7.1x, lower than at the 3 TB dataset size.

5. Conclusion

EDB PGAA is an addition to PostgreSQL that is vital to make it suitable for analytics workloads at any scale.

At lower dataset sizes (100 GB-1 TB), most of the acceleration comes from PGAA being able to offload analytical queries to Spark, with RAPIDS contributing a 2x-4x improvement over vanilla Spark on top of that. However, at 3TB and above, GPU acceleration with spark-rapids contributes the majority of the overall performance improvement (7x-14x over vanilla Spark).

At low dataset sizes (100 GB), EDB PGAA with spark-rapids dramatically outperforms vanilla PostgreSQL on most (82/99) queries from the TPC-DS benchmark. At 1 TB, spark-rapids is faster than PostgreSQL on 91/99 queries, with the median query being 14.2x faster and the slowest query taking close to a minute. In comparison, 9% of queries fail to complete in 2h on PostgreSQL and only 45 queries take less than a minute.

This performance gap grows further with the dataset size: at 3 TB, the median query is accelerated 33x by spark-rapids over PostgreSQL, with 15 queries failing to complete in 2h on PostgreSQL and the slowest query completing in 5 minutes on spark-rapids.

Due to timing out PostgreSQL queries at 2 hours, we were not able to measure the overall performance improvement on the full query set at 3 TB: the reported 53x is an understatement. We estimate that the real speedup that we would measure if we let PostgreSQL run the timed out queries to completion would be beyond the 90-100x observed at lower dataset sizes.

It's also worth noting the costs of tuning PostgreSQL for analytics workloads: this process is time consuming and, as evidenced by some queries being made slower by indexes, it is very challenging to optimize PostgreSQL for a general-purpose analytics use case. In contrast, EDB PGAA requires minimal configuration out of the box and no index tuning.

This advantage stops growing (or even decays on the RTX Pro 6000 system) at 10 TB which we hypothesize is the I/O limit of a single node.

6. Next steps

6.1. Iceberg support

The Spark and spark-rapids tests above were performed with Apache Parquet as the data format. EDB PGAA also supports replication of PostgreSQL tables to Apache Iceberg tables managed by an Iceberg REST catalog. While spark-rapids supports Apache Iceberg with REST catalogs as of 25.10, we were not able to benchmark this due to spark-rapids requiring an older Apache Iceberg version (1.6) that had issues with PGAA-written data.

spark-rapids support for Apache Iceberg 1.9 is coming in February 2026. Using this version, we would like to benchmark the full proposed system for GPU-accelerated Postgres, in particular, including spark-rapids reading merge-on-read equality delete files that PGAA uses to provide near-realtime analytics on PostgreSQL data.

6.2. Multi-node Spark cluster testing

We performed these tests on a single high-spec system, which doesn't exercise Spark's distributed query execution and is unrealistic for the kinds of systems EDB PostgreSQL will run on in real-life use cases.

In addition, we can see the acceleration from spark-rapids stalling beyond 3 TB in the case of the L40S system and going back from 14x to 8x on the RTX 6000 Pro system. We suspect that this is due to I/O saturation.

We would like to perform these benchmarks on a multi-node Spark cluster, perhaps with the same overall number of GPUs but with other I/O bottlenecks eliminated. This will also help us gauge the optimal GPUs-per-machine configuration for analytics use cases.

6.3. RTX 6000 Pro testing

We have seen some promising results on the next generation of NVIDIA GPUs such as RTX 6000: early tests with an 8x RTX 6000 Pro system show superior performance to the L40S-based system at a lower RRP. Further benchmarking is required to investigate this, as the RTX 6000 Pro system and the L40S system we tested had different CPU and RAM configurations due to system availability issues.