Technical white paper

Efficient analytics for data lakehouse with HPE and CelerData

HPE Alletra Storage MP X10000 and CelerData Enterprise deliver the speed, scale, and confidence needed for modern decision-making

Contents

Executive Summary 3
Solution Overview 3
Solution Components 4
HPE Alletra Storage MP X10000.................................... 4
CelerData Enterprise....................................5
HPE ProLiant Compute 5
HPE Morpheus Enterprise software....................................5
Solution validation....................................7
Analysis 13
Summary 13
Appendix A 15
References....................................21

Executive Summary

Many organizations adopt data lakehouses to unify analytics, but they often face issues with disorganized data integration, inconsistent performance at scale, and challenges in establishing governance and security controls. Along with steep learning curves and operational difficulties, the potential for quick, unified analytics is hampered by complexity and inefficiency.

HPE Alletra Storage MP X10000 delivers the high performance and large-scale capacity that modern data lakehouses require, featuring an all-flash, cloud-native design optimized for real-time analytics at any scale. Its S3-compatible architecture easily integrates with open data ecosystems, while enterprise-grade reliability and efficiency reduce costs and simplify operations. This results in faster decision-making, greater confidence, and a strong foundation for future growth.

CelerData Enterprise is built on StarRocks, a high-performance open-source SQL engine and MPP database. StarRocks benefits from a large and growing community of contributors, which ensures ongoing innovation and reliability. Featuring native Apache Iceberg integration, elastic scalability, and ANSI SQL compatibility, CelerData Enterprise provides an efficient, enterprise-grade platform for running complex queries directly on large datasets, removing the need for additional pipelines or overhead. This results in faster insights, lower costs, and better decisions with less complexity.

CelerData and HPE collaborate to turn the potential of the data lakehouse into reality. With blazing-fast queries, highly scalable storage, and integrated governance, they simplify unifying all data and obtaining real-time insights without the typical complexity. Customers gain the speed, efficiency, and confidence to make better decisions at scale.

Target audience: This technical paper is intended for solution architects, field consultants, IT specialists, and data professionals who work with data lakehouse solutions, particularly those integrating HPE Alletra Storage MP X10000 with CelerData Enterprise. Readers should have a basic understanding of object storage and data lakehouse architecture.

Document purpose: This document summarizes the integration between HPE Alletra Storage MP X10000 and CelerData Enterprise. It provides guidance on managing and analyzing large datasets, including how the integration enhances performance and scalability to deliver faster, more efficient insights.

Solution Overview

A data lakehouse combines the benefits of both systems—providing the unlimited scalability and flexibility of a data lake along with the speed and reliability of a data warehouse. By addressing the limitations of isolated architectures, modern lakehouses offer a unified platform for storing, processing, and analyzing all types of data.

Figure 1 presents an overview of the validated data lakehouse solution.

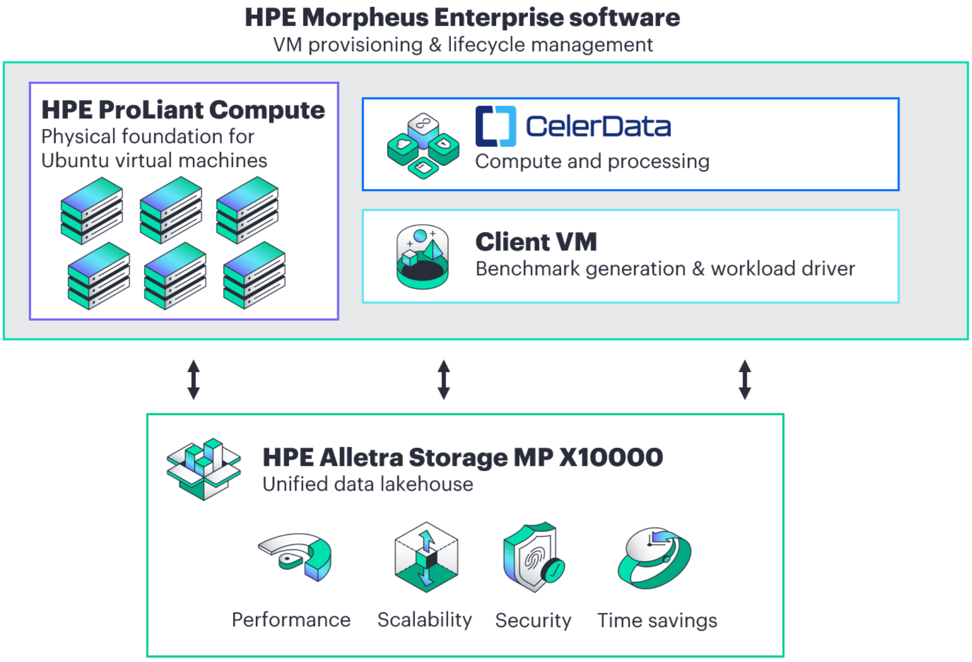

Figure 1. The components of a data lakehouse powered by CelerData and HPE Alletra Storage MP X10000

This solution delivers real-time analytics at scale by integrating HPE Alletra Storage MP X10000 with CelerData Enterprise, deployed in a virtualized environment managed by HPE Morpheus Enterprise software and built on HPE ProLiant Compute. HPE ProLiant servers provide a secure and reliable computing foundation, while HPE Morpheus Enterprise software provisions and manages the Ubuntu VMs used by both the CelerData cluster and the client system. Running on this virtualized infrastructure, CelerData performs complex SQL queries directly on data stored in HPE Alletra Storage MP X10000, which provides high-performance, S3-compatible object storage with simplified management via HPE GreenLake. By utilizing S3 APIs, CelerData removes the need for data copies and pipelines, enabling analytics to be performed directly on large datasets. Together, these technologies create a flexible, efficient platform for accelerating data-driven insights.

Want to better understand data lakehouses? This overview explains the key differences between a data lake, a data warehouse, and a data lakehouse, all in one place.

Solution Components

This section explains how each component adds to the overall solution.

HPE Alletra Storage MP X10000

HPE Alletra Storage MP X10000 provides a high-performance, scalable storage layer for this solution, serving as the foundation of the data lakehouse for CelerData Enterprise. Its S3-compatible object storage architecture supports structured, semi-structured, and unstructured data, enabling unified management of diverse workloads. With GPU-accelerated performance and Kubernetes-based elasticity, the X10000 is optimized for analytics and AI applications, ensuring consistent and rapid data access across the environment.

By delivering secure, reliable, and efficient data operations, the X10000 enables analytical and benchmarking workloads to run at scale without bottlenecks. Integration with HPE GreenLake further simplifies management and reduces operational overhead, allowing teams to focus on analysis and model development rather than infrastructure management. In this deployment, the X10000 ensures that CelerData queries have direct, low-latency access to large datasets through its native S3 APIs, making it a key enabler of real-time analytics at scale. **Figure 2** illustrates the X10000 composable building blocks—compute (nodes), capacity (JBoF), and switches.

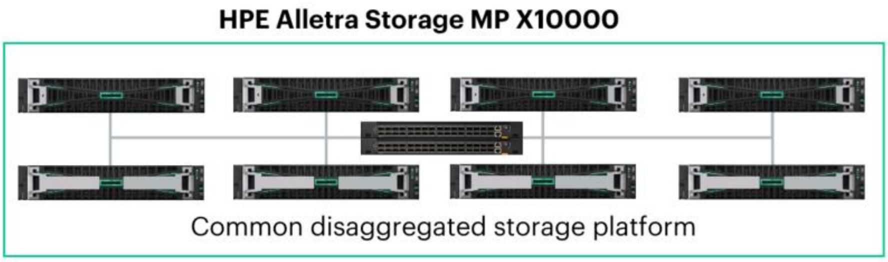

Figure 2. The modular, disaggregated hardware infrastructure of HPE Alletra Storage MP X10000

CelerData Enterprise

CelerData Enterprise, built on StarRocks—a high-performance, open-source SQL engine and MPP database supported by a growing community of contributors—provides the compute and query processing layer for this solution. It runs real-time analytics directly on data in HPE Alletra Storage MP X10000 through its native S3 support, reducing reliance on traditional data warehouses and minimizing pipeline complexity. While this testing environment uses a shared-data deployment model on virtualized infrastructure, CelerData also fully supports bare-metal deployments, offering flexibility to meet diverse production needs.

By removing the need for intermediate data pipelines and decreasing data movement costs, CelerData simplifies the architecture and reduces infrastructure expenses. With native Iceberg integration, elastic scalability, and SQL compatibility, it integrates smoothly into existing data ecosystems. Its ability to deliver sub-second query speeds while efficiently utilizing compute resources ensures support for even the most demanding analytical workloads. This combination of performance, scalability, and efficiency makes CelerData an essential part of the overall solution, enabling organizations to gain insights quickly and reliably from large datasets.

HPE ProLiant Compute

HPE ProLiant Compute provides the hardware foundation for this solution, delivering the performance and reliability needed to support the virtualized CelerData cluster and client system. Its enterprise-grade hardware ensures consistent throughput for demanding analytical and benchmarking workloads, while also enabling efficient resource use through advanced energy management.

With integrated security features and support for AI-optimized hardware configurations, ProLiant servers enable the environment to run complex queries and large-scale benchmarks securely and efficiently. Paired with HPE Morpheus Enterprise software, they serve as the processing core for deploying and managing Ubuntu VMs, ensuring that CelerData can operate effectively alongside HPE Alletra Storage MP X10000 to deliver real-time analytics at scale.

HPE Morpheus Enterprise software

HPE Morpheus Enterprise software was used to deploy and manage the virtual infrastructure in this solution. Built on Morpheus Data's technology, a leading provider of private and hybrid cloud orchestration, it offers a unified approach to virtualization through simplified and flexible management, providing a single platform for provisioning and managing both HPE (KVM-based) and VMware ESXi™ virtual machines.

For this deployment, HPE Morpheus Enterprise software was used to provision and manage the Ubuntu VMs running the CelerData cluster, as well as the client VM responsible for generating the dataset and benchmarking workload. Morpheus Enterprise offers advanced features, including profiling, cost calculators, and broader multi-cloud orchestration capabilities, making it well-suited for complex or large-scale environments. However, HPE Morpheus VM Essentials software can also be used in similar deployments. VM Essentials is tailored for smaller environments and supports managing local KVM clusters and VMware® clusters through vCenter. It offers a much lower entry point compared to Enterprise, and its per-socket licensing avoids the per-core costs often associated with Broadcom licensing. This makes VM Essentials particularly attractive for organizations running high-core-count systems that want predictable costs while still benefiting from streamlined VM lifecycle management. This enables customers to provision workloads on demand in the right environment, thereby preventing vendor lock-in. There's also a bulk migration feature to streamline the transition of VMs from vCenter environments to HVM clusters. Overall, customers can optimize their virtualized infrastructure costs by using HPE Morpheus VM Essentials:

- Reduce VM licenses by 27% by upgrading to HPE ProLiant Gen12 servers. Upgrading from a HPE Gen10 ProLiant server to the latest generation (e.g., a 48-core Gen 12) reduces the number of VM core-based licenses you need to get the same amount of work done by up to 27%.
- Up to 90% savings for migrated workloads. Based on internal analysis of the cost of HPE Morpheus VM Essentials licensing versus available market data on the cost of existing virtualization licensing (VMware vSphere® Foundation).
- Achieve 7:1 consolidation and save up to 65% in power savings per year. The results are published as of 01-01-2025 on spec.org, and compared against a 48-core estimated Gen12 system.

Figure 3 shows the VM instances in Morpheus Enterprise software.

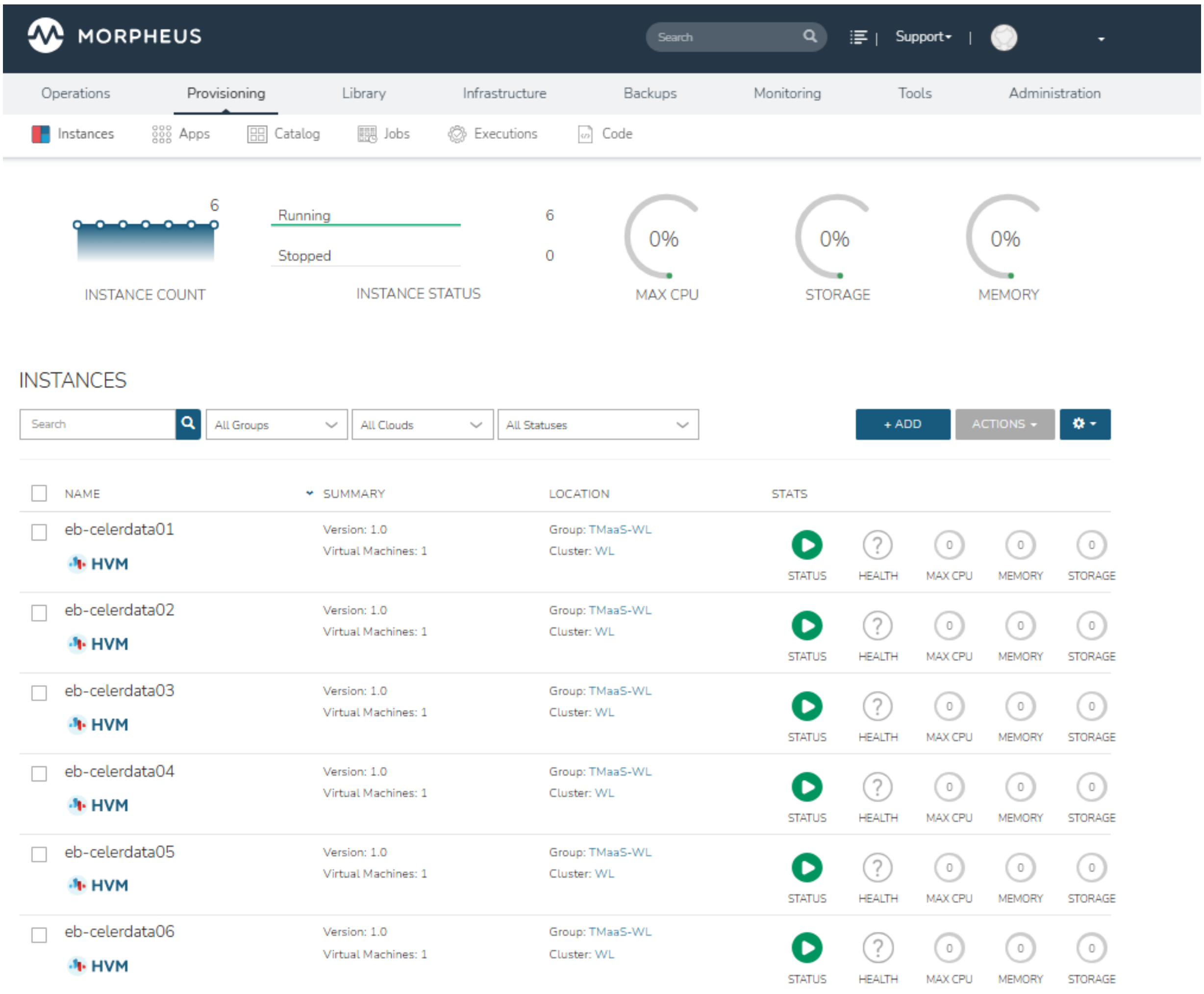

Figure 3. VM management using HPE Morpheus Enterprise

In this solution, Morpheus Enterprise software provides the agility and lifecycle management needed to run CelerData Enterprise workloads in a VM-based architecture, closely integrated with HPE ProLiant compute and HPE Alletra Storage MP X10000 for unified data lakehouse analytics. The table below summarizes the main benefits of using HPE Morpheus Enterprise software in this solution.

Benefit	Description
Elastic scaling	Instantly grow or shrink CelerData Enterprise clusters to meet changing workload demands.
Fast provisioning	Deploy new Ubuntu VMs and environments in minutes, not days.
High availability	Enterprise-grade resilience with built-in recovery, live migration, and workload protection.
Simplified lifecycle	Automates provisioning, patching, and decommissioning of CelerData VMs, reducing administrative overhead.
Optimized resource use	Distributes workloads efficiently across HPE ProLiant Compute for better performance and cost control.
Hybrid flexibility	Unified management for both KVM-based and VMware-based VMs, ensuring adaptability and avoiding vendor lock-in.

Solution validation

To validate the solution, CelerData Enterprise was deployed and connected to an HPE Alletra Storage MP X10000. The process followed CelerData's documentation and included preparing hosts, configuring storage, and loading a TPC-DS dataset for benchmarking:

1. Each host was configured with 16 vCPUs, 64 GB of memory, and a dedicated drive for CelerData.
    a. The CelerData cluster has five Ubuntu VMs:
        i. one Front End (FE) node
        ii. four Compute Nodes (CNs)
2. AWS CLI and MySQL 8.4 were installed on the FE node following the documentation. The AWS CLI, which is simple to install and configure, was used to verify—via S3 commands—that all hosts requiring access to the HPE Alletra Storage MP X10000 had proper connectivity. MySQL provides the necessary client tools to interact with the CelerData cluster.
3. CelerData was installed following the documented process, including the free trial workflow.
4. During installation, the X10000 bucket was configured for use with the FE node, as shown in **Figure 4**.

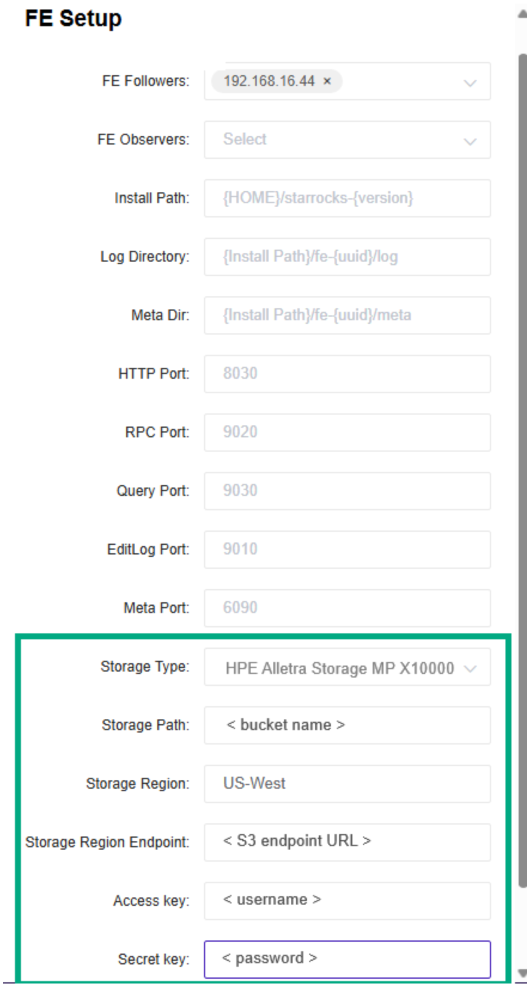

Figure 4. Configuring the FE node for use with the X10000

5. A valid license was applied, enabling access to the CelerData web UI for configuration (**Figure 5** and **Figure 6**).

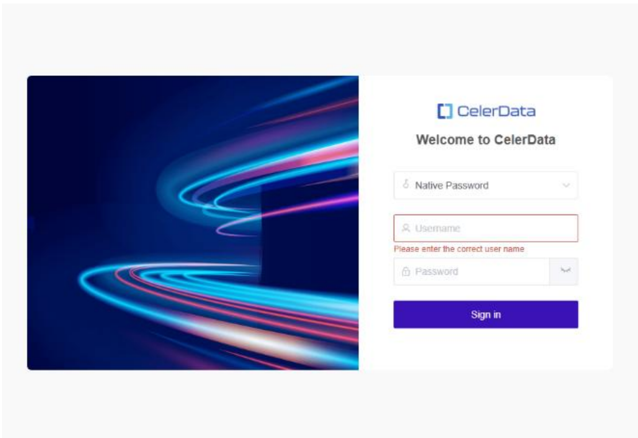

Figure 5. Initial login to the CelerData web UI

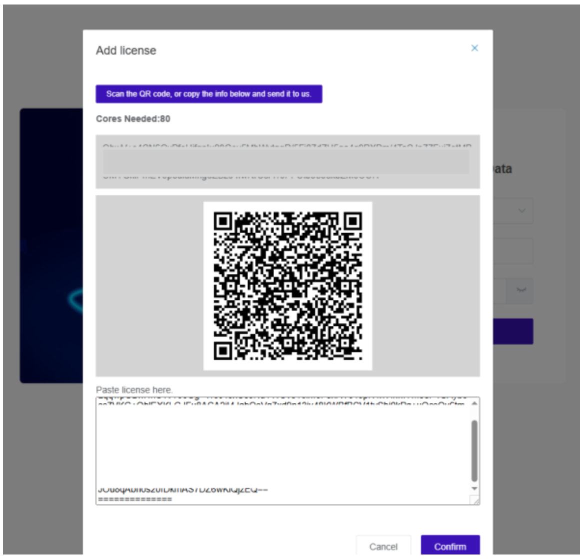

Figure 6. Adding a license using the CelerData web UI

6. Using the CelerData Editor, an external catalog—which uses the S3 bucket—and database were created and defined in Apache Iceberg format to ensure compatibility and performance for the TPC-DS dataset.

7. TPC-DS was installed on the client system and used to generate a 1 TB dataset.

Note

Generating the TPC-DS dataset and converting it to Parquet requires approximately 2 TB of free disk space. If one drive isn't large enough, you may need to distribute the workload across multiple drives.

---

8. The 24 generated .dat files were converted to Parquet format using Spark. For reference, view the beginning and end of the script—and add the other 23 schemas.

9. Parquet files were then copied to the X10000 bucket for use with CelerData.

10. Tables with full schemas were created using the CelerData Editor (snippet is only for "`catalog_page`").

11. The Parquet data was inserted into the Iceberg tables.

12. Verified that the tables contained data using the CelerData Editor.
Figure 7 shows the CelerData Editor and the catalog, database, and commands run against the X10000 bucket are visible:

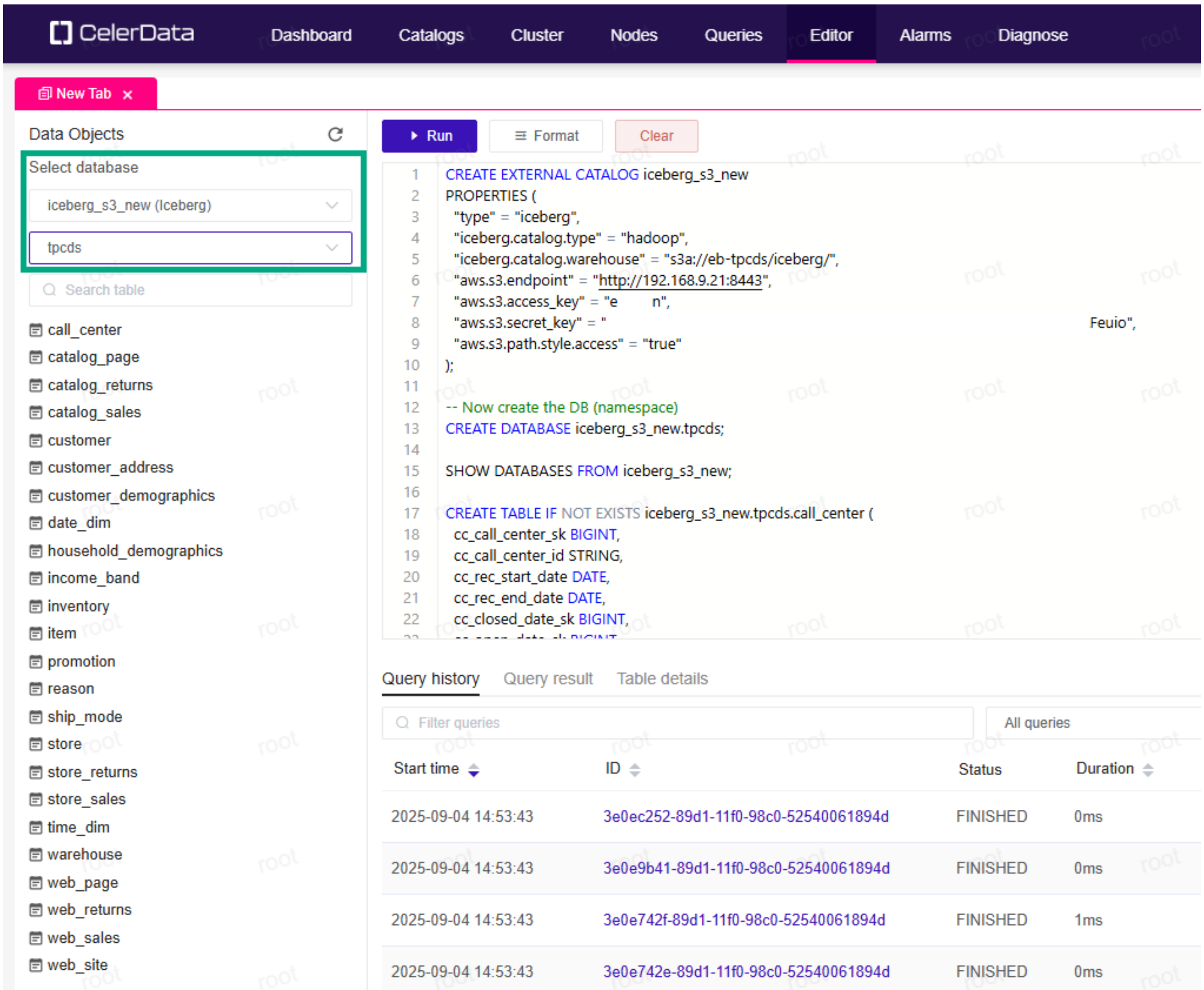

Figure 7. Using the CelerData Editor with the catalog and database for the X10000 bucket

13. After verifying that the tables were fully populated, Apache JMeter was installed on the client system and used to run TPC-DS benchmark tests, "no partition" and "gold" with CelerData. These tests demonstrate both the raw capabilities of CelerData and the performance improvements gained through optimization.
    - The no partition benchmark provides a baseline by running SQL queries against unoptimized tables using Apache JMeter with CelerData, showing how the system performs "out of the box."
    - The Gold benchmark represents a tuned, production-ready environment where JMeter runs queries against partitioned tables with collected statistics, highlighting the system's peak performance under real-world best practices.
14. Highlights for the benchmark tests:
    - Queries ran = 99/99
    - Error rate = 0%
    - Average response time = ~4.1 seconds
    - Maximum observed latency = ~53 seconds
    - APDEX score = ~0.38
15. **Figure 8** and **Figure 9** show the benchmark testing results summarized in the Apache JMeter dashboard:

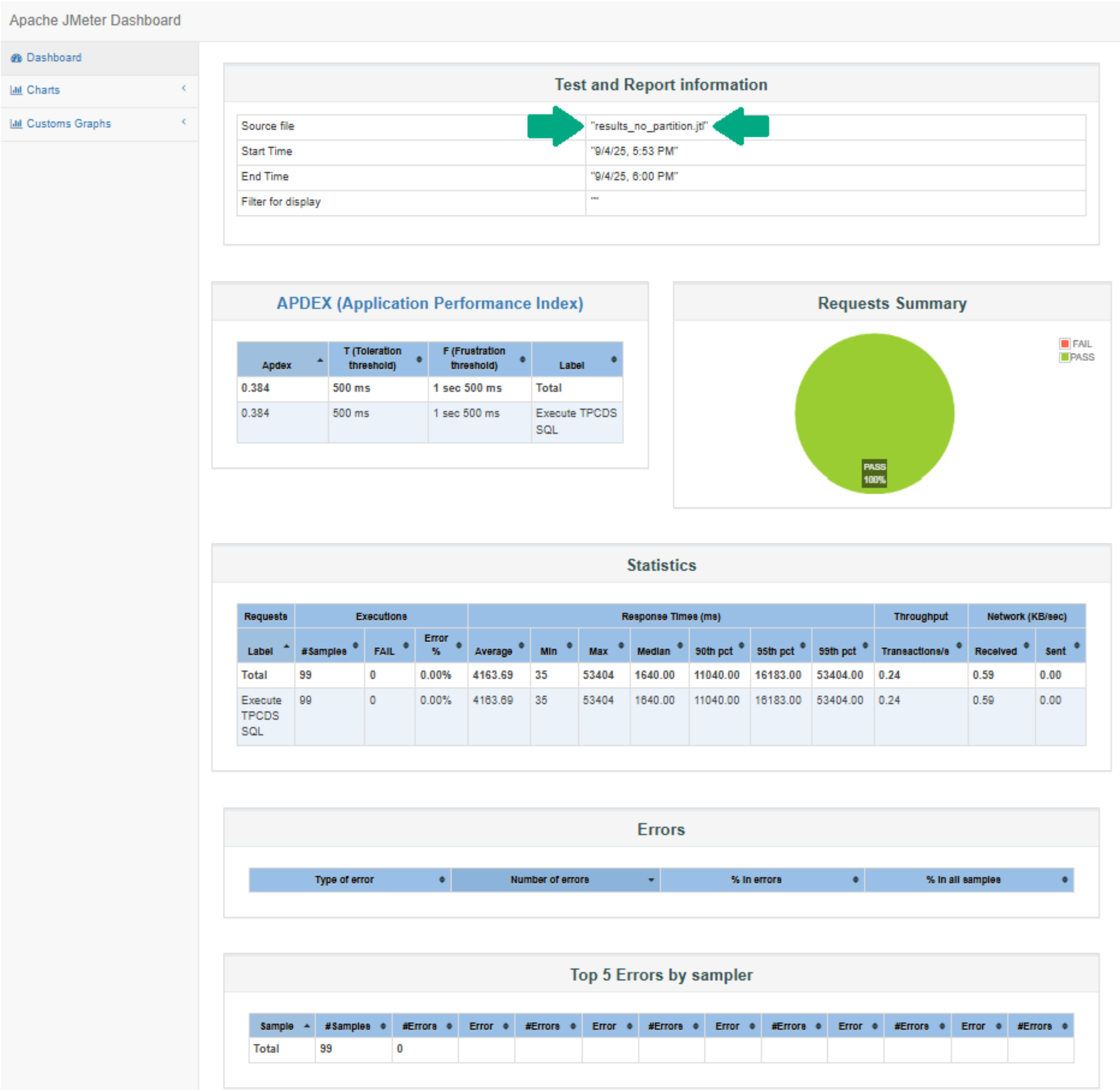

Figure 8. Viewing the results for the "no partition" benchmark test using the Apache JMeter dashboard

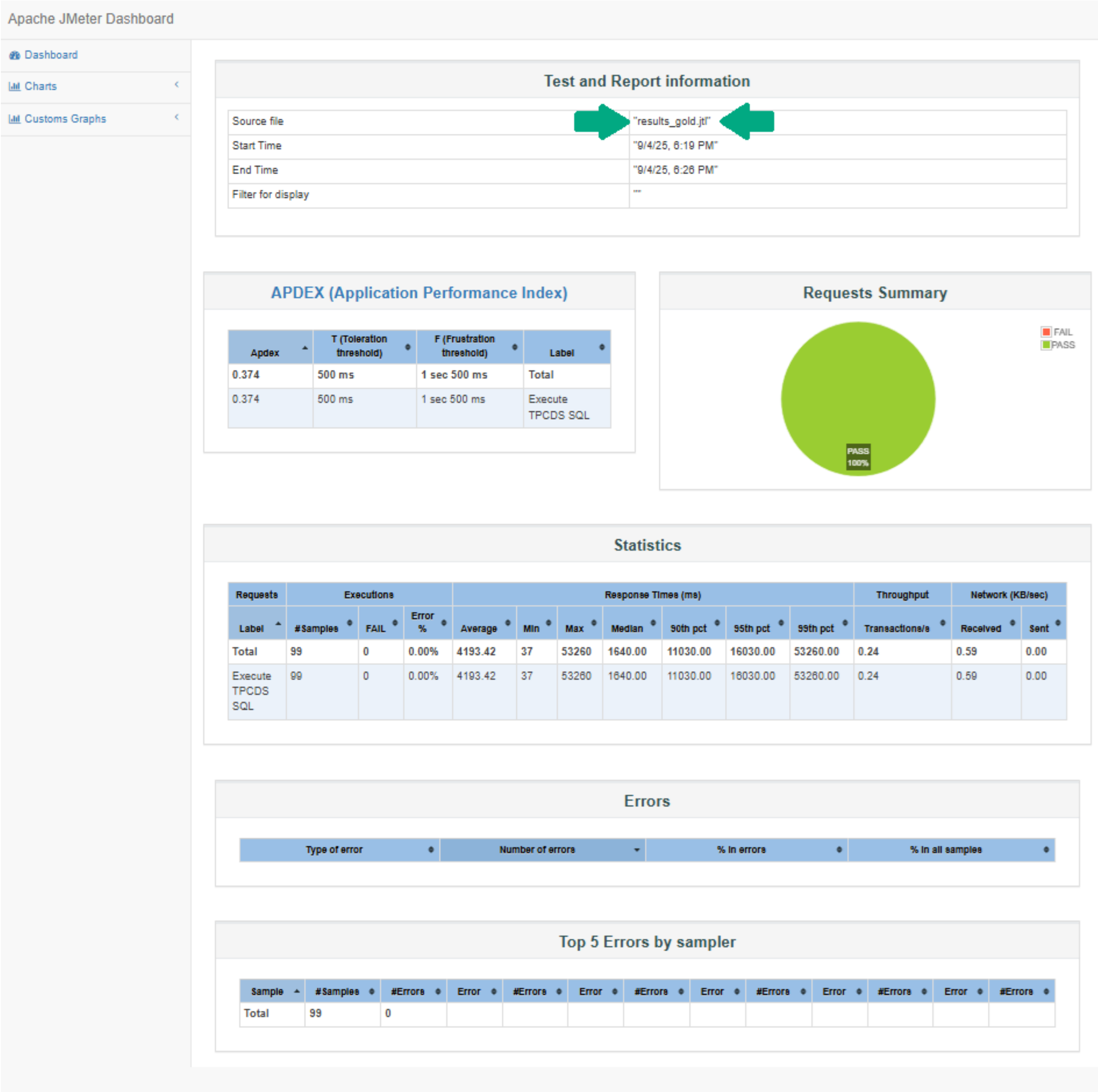

Figure 9. Viewing the results for the "gold" benchmark test using the Apache JMeter dashboard

16. Managing and viewing the metrics for the X10000 bucket through HPE GreenLake was straightforward, as shown in **Figure 10**:

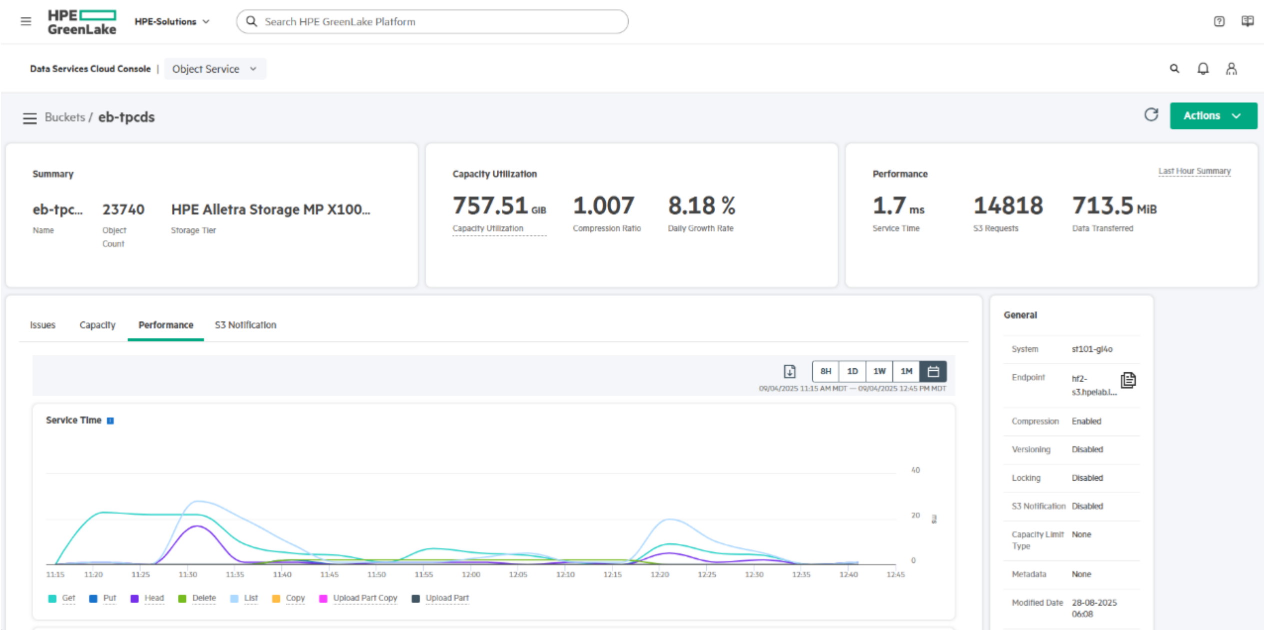

Figure 10. Using HPE GreenLake to view the X10000 bucket metrics

Analysis

Testing focused on customer-centric scenarios, emphasizing querying, processing, and analytics at scale. Using industry-standard tools from the Transaction Processing Performance Council (TPC), a one-terabyte dataset was created, converted to Parquet format, and stored in an HPE Alletra Storage MP X10000 bucket. This dataset simulated real-world workloads with complex, columnar data to provide an accurate performance benchmark.

An external catalog was created using the X10000 bucket and database, defined in Apache Iceberg format to ensure compatibility and optimal performance for the TPC-DS dataset. Tables with full schemas were added, and then data was inserted into them. After verifying that all tables contained data, the benchmark tests were run.

The results showed that combining HPE Alletra Storage MP X10000 and CelerData Enterprise, along with HPE ProLiant Compute and HPE Morpheus Enterprise software, allows for fast, reliable, and scalable analytics on large, diverse datasets. HPE ProLiant servers offer a secure, high-performance foundation, while HPE Morpheus Enterprise provisions and manages Ubuntu virtual machines hosting the CelerData cluster and client systems. Running on this infrastructure, CelerData runs complex queries directly on the data stored in the X10000, providing consistent, high-performance insights in real-world scenarios. Together, these technologies enable organizations to easily transform extensive, distributed data into actionable intelligence.

Summary

Organizations are increasingly recognizing the potential of artificial intelligence (AI) and data analytics to extract insights from their extensive and diverse datasets. However, many struggle to fully utilize these technologies, especially when it comes to accessing and analyzing data stored on traditional storage systems. These legacy systems hinder organizations from integrating information and gaining a comprehensive view of their data.

A significant challenge is the vast volume of data, which makes up most of today's information. The raw data—often riddled with errors, inconsistencies, and biases—impedes processing and understanding. Traditional data systems, not designed to handle such complexity, often struggle to effectively combine and analyze this diverse information.

As organizations attempt to leverage AI and data analytics, they face increasing pressure from regulations and rising security threats associated with large, dispersed data stores that are more susceptible to cyberattacks.

Consequently, organizations struggle to fully realize the potential of their datasets, even with the right tools at their disposal. The primary issue is not the lack of technology, but rather the difficulty of managing and extracting insights from data within legacy systems.

HPE Alletra Storage MP X10000, CelerData Enterprise, HPE ProLiant Compute, and HPE Morpheus Enterprise software work together to deliver high-performance analytics at scale. They simplify data operations, transforming large, complex datasets into clear insights. These technologies help optimize data workflows, reduce costs, and accelerate AI and analytics processes. They provide organizations with a flexible, future-ready platform for making data-driven decisions.

Appendix A

1. **Creating an external catalog and a database:**

```
CREATE EXTERNAL CATALOG iceberg_s3_new
PROPERTIES (
  "type" = "iceberg",
  "iceberg.catalog.type" = "hadoop",
  "iceberg.catalog.warehouse" = "s3a://eb-tpcds/iceberg/",
  "aws.s3.endpoint" = http://192.168.9.21:8443
  "aws.s3.access_key" = "<username>",
  "aws.s3.secret_key" = "<password>",
  "aws.s3.path.style.access" = "true"
);

CREATE DATABASE iceberg_s3_new.tpcds;
```

2. **TPC-DS command for generating a 1TB dataset:**

```
./dsdgen -DIR=/<local_path_exceeding_1TB> -SCALE=10000 -PARALLEL=8 -CHILD=1
```

3. **Converting the .dat files that TPC-DS generated to Parquet format:**

```
/opt/spark/bin/spark-submit
  --driver-memory 24g
  /home/<user>/convert_tpcds_to_parquet.py
```

4. **The beginning and end of the Spark script for converting the files to parquet format:**

```
from pyspark.sql import SparkSession
from pyspark.sql.types import *
import os

    spark = SparkSession.builder \
        .appName("TPC-DS .dat to Parquet") \
        .getOrCreate()

    schemas = {
    "catalog_page": StructType([
            StructField("cp_catalog_page_sk", LongType(), True),
            StructField("cp_catalog_page_id", StringType(), True),
            StructField("cp_start_date_sk", LongType(), True),
            StructField("cp_end_date_sk", LongType(), True),
            StructField("cp_department", StringType(), True),
            StructField("cp_catalog_number", IntegerType(), True),
            StructField("cp_catalog_page_number", IntegerType(), True),
            StructField("cp_description", StringType(), True),
            StructField("cp_type", StringType(), True),
        ]),

    })

    # Define input and output directories
    input_dir = "/<path_to_.dat_files>"
    output_dir = "/<path_to_converted_files>"

    # Loop through each table and convert .dat to Parquet
    for table_name, schema in schemas.items():
        dat_path = f"{input_dir}/{table_name}_1_8.dat"
        parquet_path = f"{output_dir}/{table_name}"

        if not os.path.exists(dat_path):
            print(f"Skipping {table_name}: File not found at {dat_path}")
            continue

        print(f"Converting {table_name} to Parquet...")

        df = spark.read \
```

```
            .option("delimiter", "|") \
            .schema(schema) \
            .csv(dat_path)

        # Drop the final empty column caused by trailing pipe
        df = df.drop(f"_c{len(df.columns)}")
        df.write.mode("overwrite").parquet(parquet_path)

    print("All available .dat files have been converted to Parquet.")
    spark.stop()
```

5. **Copying the Parquet files to the S3 bucket:**

```
aws s3 cp <local_path_to_files>/ s3://eb-tpcds/staging --recursive \
    --endpoint-url http://192.168.9.21:8443
```

6. **Creating tables for the parquet files—repeat for all remaining tables:**

```
CREATE TABLE IF NOT EXISTS iceberg_s3_new.tpcds.catalog_page (
  cp_catalog_page_sk BIGINT,
  cp_catalog_page_id STRING,
  cp_start_date_sk BIGINT,
  cp_end_date_sk BIGINT,
  cp_department STRING,
  cp_catalog_number INT,
  cp_catalog_page_number INT,
  cp_description STRING,
  cp_type STRING
);

USE iceberg_s3_new.tpcds;
SHOW TABLES;
```

7. **Inserting data into the tables:**

```
/opt/spark/bin/spark-submit \
  --master local[*] \
  --driver-memory 24g \
  --executor-memory 24g \
  --conf spark.network.timeout=800s \
  --conf spark.executor.heartbeatInterval=60s \
  --conf spark.sql.shuffle.partitions=24 \
  --packages org.apache.hadoop:hadoop-aws:3.3.2 \
  load_parquet_to_iceberg.py
```

8. **Spark script for inserting data into the tables:**

```
#!/usr/bin/env python3
# load_parquet_to_iceberg.py

from pyspark.sql import SparkSession
import os
import sys

# -----------------------------
# CONFIG
# -----------------------------
CATALOG_NAME = "iceberg_s3_new"
ICEBERG_DB   = f"{CATALOG_NAME}.tpcds"

# Read source Parquet from local disk by default.
# To read from S3 instead, set READ_SOURCE = "s3".
READ_SOURCE  = os.environ.get("READ_SOURCE", "local").lower()  # "local" or "s3"

LOCAL_BASE   = "/tpcdsconversion"
S3_BASE      = "s3a://eb-tpcds/staging"

# S3 / Warehouse configuration (Iceberg HadoopCatalog)
S3_ENDPOINT  = "http://192.168.9.21:8443"   # your active endpoint
```

```
S3_ACCESS    = "<username>"
S3_SECRET    = "<password>"
WAREHOUSE    = "s3a://eb-tpcds/iceberg/"

TABLES = [
    "call_center", "catalog_page", "catalog_returns", "catalog_sales", "customer",
    "customer_address", "customer_demographics", "date_dim", "household_demographics",
    "income_band", "inventory", "item", "promotion", "reason", "ship_mode", "store",
    "store_returns", "store_sales", "time_dim", "warehouse", "web_page",
    "web_returns", "web_sales", "web_site"
]

# ----------------------------
# SPARK SESSION
# ----------------------------
spark = (
    SparkSession.builder
    .appName("Load TPC-DS Parquet to Iceberg (HadoopCatalog)")
    # Iceberg catalog wired to your warehouse
    .config(f"spark.sql.catalog.{CATALOG_NAME}", "org.apache.iceberg.spark.SparkCatalog")
    .config(f"spark.sql.catalog.{CATALOG_NAME}.catalog-impl",
"org.apache.iceberg.hadoop.HadoopCatalog")
    .config(f"spark.sql.catalog.{CATALOG_NAME}.warehouse", WAREHOUSE)
    .config(f"spark.sql.catalog.{CATALOG_NAME}.io-impl",
"org.apache.iceberg.hadoop.HadoopFileIO")

.config("spark.sql.extensions","org.apache.iceberg.spark.extensions.IcebergSparkSessionExtensi
ons")
    # S3A client settings for your object store
    .config("spark.hadoop.fs.s3a.endpoint", S3_ENDPOINT)
    .config("spark.hadoop.fs.s3a.access.key", S3_ACCESS)
    .config("spark.hadoop.fs.s3a.secret.key", S3_SECRET)
    .config("spark.hadoop.fs.s3a.path.style.access", "true")
    .config("spark.hadoop.fs.s3a.impl", "org.apache.hadoop.fs.s3a.S3AFileSystem")
    # Helpful stability defaults (tune as desired)
    .config("spark.network.timeout", "800s")
    .config("spark.executor.heartbeatInterval", "60s")
    .config("spark.sql.shuffle.partitions", "24")
    .getOrCreate()
)

def src_path(table: str) -> str:
    if READ_SOURCE == "s3":
        return f"{S3_BASE}/{table}"
    return os.path.join(LOCAL_BASE, table)

def main():
    print(f"\nSource mode: {READ_SOURCE.upper()}  ->  {'S3' if READ_SOURCE=='s3' else 'Local
disk'}")
    print(f"Warehouse : {WAREHOUSE}")
    print(f"Catalog   : {CATALOG_NAME}")
    print(f"Database  : {ICEBERG_DB}\n")

    failures = []
    total_rows = 0

    for table in TABLES:
        input_path   = src_path(table)
        target_table = f"{ICEBERG_DB}.{table}"

        print(f"\n=== Loading {table} ===")
        print(f"Source: {input_path}")
```

```
        print(f"Target: {target_table}")

        try:
            df = spark.read.parquet(input_path)
            cnt = df.count()
            print(f"Row count (source): {cnt}")

            if cnt == 0:
                print(f"Skipping {table} (no rows).")
                continue

            # Use append for safe re-runs. Switch to overwrite if you need a full replace.
            df.writeTo(target_table).append()
            total_rows += cnt
            print(f"Appended {cnt} rows into {target_table}")

        except Exception as e:
            print(f"Failed to load {table}: {e}")
            failures.append((table, str(e)))

    print("\n==================== SUMMARY ====================")
    print(f"Total tables attempted : {len(TABLES)}")
    print(f"Total rows appended    : {total_rows}")
    if failures:
        print("Failures:")
        for t, err in failures:
            print(f"  - {t}: {err}")
    else:
        print("All tables loaded successfully")

if __name__ == "__main__":
    try:
        main()
    finally:
        spark.stop()
```

9. **Commands to use in the CelerData editor for verifying the tables have data—change the table name to verify each table:**

```
SHOW TABLES FROM iceberg_s3_new.tpcds;
USE iceberg_s3_new.tpcds;
SELECT COUNT(*) FROM catalog_sales;
```

10. **Apache JMeter commands for running the "no partition" benchmark, and a snippet from the results:**

```
$ /opt/jmeter/bin/jmeter -n \
  -t  ~/tpcds_benchmark_updated_fixed_patched3.jmx \
  -Jqueries=/home/celerdata/tpcds_queries_no_partition_readonly_nosemi.csv \
  -j  ~/jmeter_no_partition.log \
  -l  ~/results_no_partition.jtl \
  -e -o ~/results_no_partition_report
summary =     54 in 00:02:29 =    0.4/s Avg:  2756 Min:    35 Max: 17667 Err:     0 (0.00%)
summary +     10 in 00:00:58 =    0.2/s Avg:  5770 Min:   188 Max: 33003 Err:     0 (0.00%)
Active: 1 Started: 1 Finished: 0
summary =     64 in 00:03:27 =    0.3/s Avg:  3227 Min:    35 Max: 33003 Err:     0 (0.00%)
summary +     13 in 00:01:02 =    0.2/s Avg:  4783 Min:   519 Max: 16183 Err:     0 (0.00%)
Active: 1 Started: 1 Finished: 0
summary =     77 in 00:04:29 =    0.3/s Avg:  3490 Min:    35 Max: 33003 Err:     0 (0.00%)
summary +      3 in 00:01:03 =    0.0/s Avg: 20983 Min:   172 Max: 53404 Err:     0 (0.00%)
Active: 1 Started: 1 Finished: 0
summary =     80 in 00:05:32 =    0.2/s Avg:  4146 Min:    35 Max: 53404 Err:     0 (0.00%)
summary +      1 in 00:00:30 =    0.0/s Avg: 30325 Min: 30325 Max: 30325 Err:     0 (0.00%)
Active: 1 Started: 1 Finished: 0
summary =     81 in 00:06:02 =    0.2/s Avg:  4469 Min:    35 Max: 53404 Err:     0 (0.00%)
summary +     18 in 00:00:50 =    0.4/s Avg:  2787 Min:   315 Max: 14309 Err:     0 (0.00%)
Active: 0 Started: 1 Finished: 1
```

```
summary =     99 in 00:06:52 =    0.2/s Avg:  4163 Min:    35 Max: 53404 Err:     0 (0.00%)
Tidying up ...  @ 2025 Sep 4 18:00:25 UTC (1757008825820)
```

```
... end of run
```

11. Apache JMeter commands for running the “gold” benchmark, and a snippet from the results:

```
$ /opt/jmeter/bin/jmeter -n \
  -t  ~/tpcds_benchmark_updated_fixed_patched3.jmx \
  -Jqueries=/home/celerdata/tpcds_queries_gold_readonly_nosemi.csv \
  -j  ~/jmeter_gold.log \
  -l  ~/results_gold.jtl \
  -e -o ~/results_gold_report
summary +     6 in 00:00:28 =    0.2/s Avg:  4625 Min:   828 Max: 16030 Err:     0 (0.00%)
Active: 1 Started: 1 Finished: 0
summary =    69 in 00:03:57 =    0.3/s Avg:  3439 Min:    37 Max: 32921 Err:     0 (0.00%)
summary +     4 in 00:00:28 =    0.1/s Avg:  7022 Min:  1653 Max: 12611 Err:     0 (0.00%)
Active: 1 Started: 1 Finished: 0
summary =    73 in 00:04:26 =    0.3/s Avg:  3635 Min:    37 Max: 32921 Err:     0 (0.00%)
summary +     5 in 00:01:00 =    0.1/s Avg: 11972 Min:   446 Max: 53260 Err:     0 (0.00%)
Active: 1 Started: 1 Finished: 0
summary =    78 in 00:05:25 =    0.2/s Avg:  4169 Min:    37 Max: 53260 Err:     0 (0.00%)
summary +    13 in 00:01:00 =    0.2/s Avg:  4579 Min:   154 Max: 30160 Err:     0 (0.00%)
Active: 1 Started: 1 Finished: 0
summary =    91 in 00:06:25 =    0.2/s Avg:  4228 Min:    37 Max: 53260 Err:     0 (0.00%)
summary +     8 in 00:00:30 =    0.3/s Avg:  3796 Min:   266 Max: 14177 Err:     0 (0.00%)
Active: 1 Started: 1 Finished: 0
summary =    99 in 00:06:55 =    0.2/s Avg:  4193 Min:    37 Max: 53260 Err:     0 (0.00%)
summary =    99 in 00:06:55 =    0.2/s Avg:  4193 Min:    37 Max: 53260 Err:     0 (0.00%)
Tidying up ...    @ 2025 Sep 4 18:26:01 UTC (1757010361006)
... end of run
```

References

Drive innovation and accelerate time to value: HPE Alletra Storage MP X10000

Transform data chaos into clarity and opportunity

HPE Morpheus VM Essentials: Reduce costs, eliminate vendor lock-in, and simplify hybrid IT

Compute engineered for your hybrid world: HPE ProLiant Servers

Reduce the time and cost of delivering lakehouse analytics solutions: CelerData

Visit HPE.com

Chat now

a00154356enw

HEWLETT PACKARD ENTERPRISE

hpe.com