White paper

Born for the Enterprise, Stronger in the Cloud

Carrie Ballinger
Cloud Architecture Performance Solutions

teradata.

Table of Contents

Introduction

During the Great Depression of the 1930s, my grandmother had to become frugal to survive. This meant making the most of what she had and not wasting anything. Her family never threw out food, always turned off lights when leaving a room, and saved every penny. My grandmother's entire life was oriented around efficient use of resources learned during a time of scarcity. While she adopted this approach out of necessity during the Depression, she passed those traits down to her descendants. Through the decades, these thrifty habits have served myself and my family well, even in times when resources are more plentiful.

Teradata was born and grew up centered on complex enterprise platforms. Like families scraping by in the Great Depression, enterprise users had to learn how to live in a world of limited and inflexible assets. During its initial years, Teradata responded to the needs of customers running on fixed-size systems by making widespread economies inside the database not just a nicety, but a "must-have."

Teradata's architecture has been designed with efficiency and cost at top of mind. A strong focus on fine-tuning in the enterprise world for over 30 years has led to high performance with the least possible effort. Just like turning off the lights when you leave a room, weeding out wasted resources or streamlining algorithms saves money.

As Teradata has advanced into the cloud, its past has not become a liability. Instead, having been born for, and reaching adulthood in, the enterprise world gives Teradata an edge in maturity, robustness, cost-effectiveness, depth of parallel capabilities, and the variety of optimizations. Although resources in the cloud may be infinite, budgets are not. Because of the craftsmanship embraced during its enterprise-focused years, Teradata has established itself in the cloud having already solved problems that other cloud vendors do not even know exist.

The focus of this paper begins with some of the key "born with" characteristics that were architected into the Teradata database. Next, it will cover some of the notable evolutionary steps that enriched the original capabilities. Finally, the paper will discuss the impact of these existing and evolving competencies on Teradata's cloud-native offering, VantageCloud Lake.

Born with:
Multidimensional parallel capabilities

Everything in the Teradata database was designed for parallelism—from the entry of SQL statements to the smallest detail of their execution—to weed out any possible single point of control and to effectively eliminate the conditions that can breed gridlock in a system. This section describes three basic dimensions of parallelism that were architected into the database.

1. Parallel execution across all units of parallelism

Everything that happens inside Teradata's database is distributed across a predefined number of parallel units called AMPs. Each AMP acts like a microcosm of the database, supporting such things as data loading, reading, writing, journaling, and recovery for all the data that it owns.

While the AMP is the fundamental unit of parallelism, there are two additional parallel dimensions woven into the database, specifically for query performance. These are referred to here as "within-a-step" parallelism and "multistep" parallelism. The following sections describe these two capabilities.

2. Within-a-step parallelism

When the Teradata optimizer builds a query plan, it breaks work into one or more execution chunks known as "steps". Each of these steps can include multiple distinct operations. Within-a-step parallelism is when multiple operations within a step flow into each other so that several operations can overlap and execute in parallel (see Figure 2).

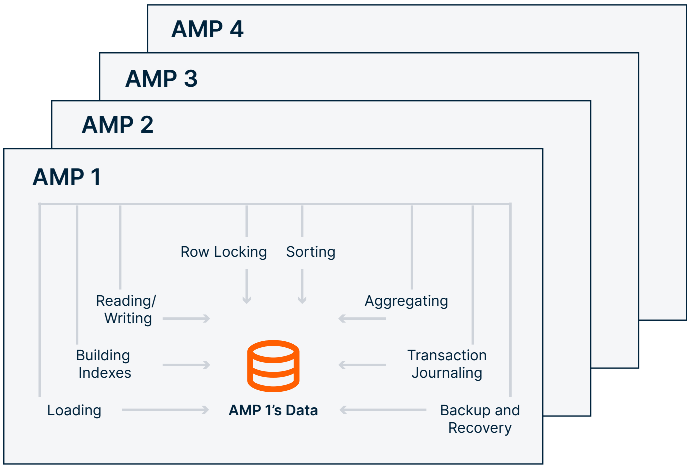
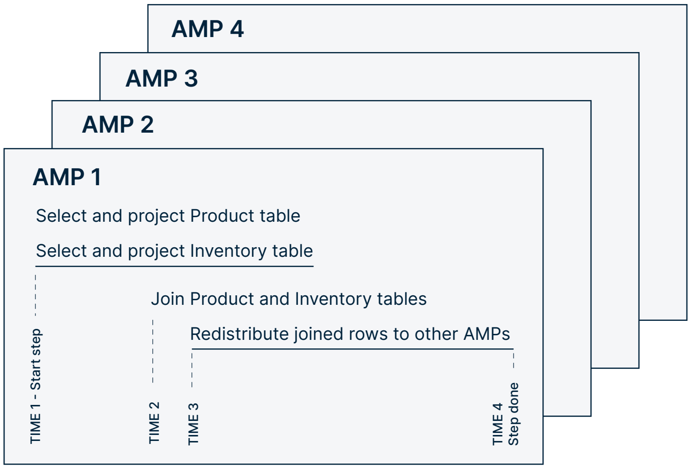


Figure 1. Inside of a single unit of parallelism



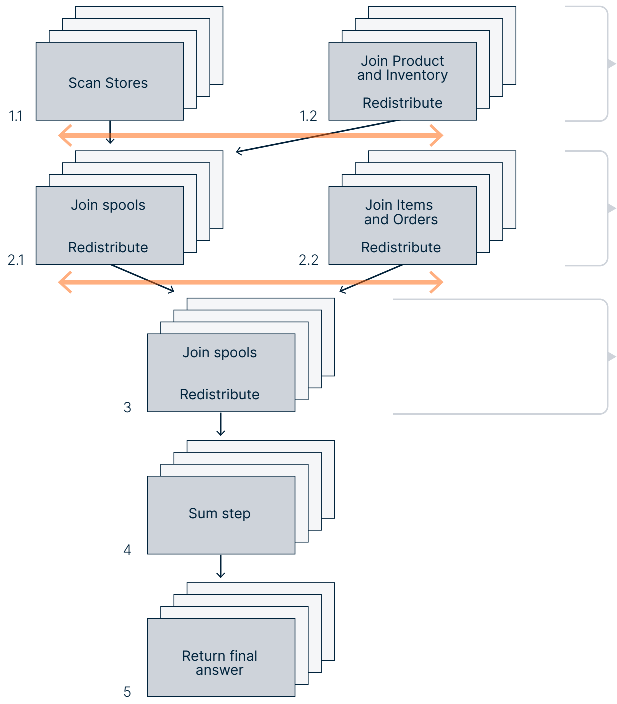
AMP 4
AMP 3
AMP 2
AMP 1
Select and project Product table
Select and project Inventory table
Join Product and Inventory tables
Redistribute joined rows to other AMPs
TIME 1 - Start step
TIME 2
TIME 3
TIME 4 Step done

Figure 2. Multiple operations executing in parallel within a single query step

3. Multistep parallelism

Multistep parallelism is enabled when the optimizer chooses to execute multiple steps of a query simultaneously, across all the participating units of parallelism (see Figure 3).

Figure 3 shows four AMPs supporting a single query's execution, and the query has been optimized into seven steps. Step 1.2 demonstrates within-a-step parallelism as described in the section above. Steps 1.1 and 1.2 execute at the same time, illustrating multistep parallelism, as do steps 2.1 and 2.2.



Within-a-step Parallelism
Multiple operations are pipelined

1. Scan Product
2. Scan Inventory
3. Join Product and Inventory
4. Redistribute joined rows

Multi-step Parallelism
Do step 1.1 and 1.2 and also steps 2.1 and 2.2 simultaneously

Query Execution Parallelism
Four AMPs perform each step on their data blocks at the same time

Figure 3. Multiple dimensions of parallelism

Born with: Optimizer with a 360-degree perspective

Having an array of parallel techniques can turn into a disadvantage and may lead to congestion if they are not carefully applied around the needs of each request. Orchestration of the different parallelization techniques is driven by the query optimizer, which lives within a component called the "parsing engine" (PE).

Join planning

Joining tables in a linear fashion (join table 1 to table 2, then join their result to table 3, and so on) can increase query time, particularly for queries joining hundreds of tables. Instead, the Teradata optimizer can choose to join multiple tables simultaneously, leveraging different types of joins (e.g., hash join, merge join, product join) to build a more condensed query plan.

The figure below illustrates the differences when optimizing a six-table join between a plan that is restricted to linear joins, and one that has the option of performing some of the join steps in parallel.

Sizing up the environment

In addition to the parallelism methods described above, the optimizer considers numerous other factors, including the characteristics of the data itself, the number of AMPs on each of the nodes, and the processing power of the underlying hardware. Putting all this information together, the optimizer comes up with an estimated cost in terms of resources expected to be used for each of several candidate query plans, then picks the least costly candidate.

Synchronizing table scans from multiple queries

Another cost-saving technique in Teradata's platform is the synchronized scanning of large tables. This Teradata optimizer option permits a new full-table scan to begin at the current point of an ongoing scan of the same table that is part of another session. Piggybacking table scans reduces the input/output (I/O) load and supports higher concurrency.

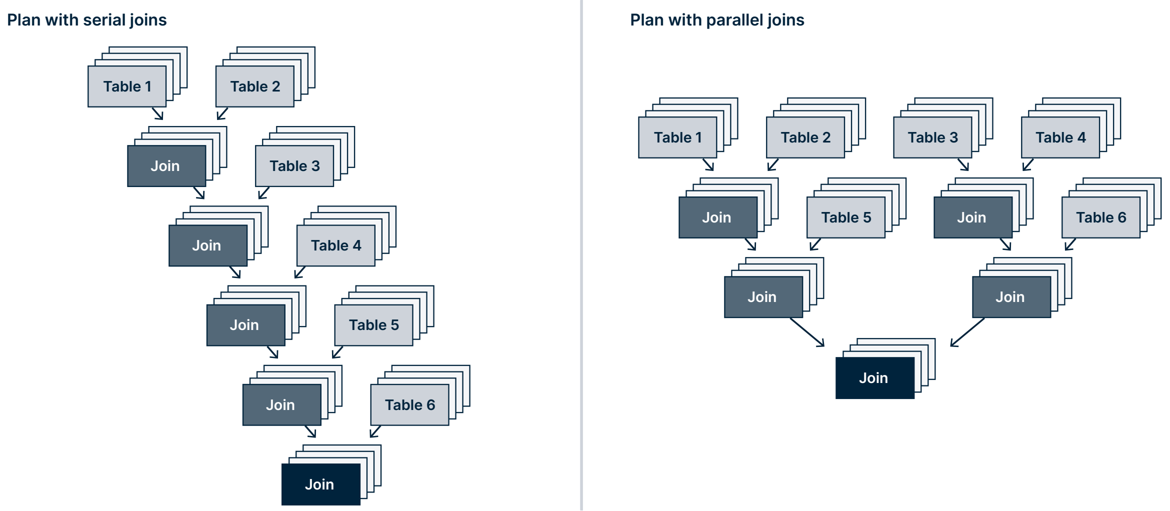


Figure 4. A serial query plan vs. a bushy query plan

Optimizer evolution

Although the fundamentals have remained the same, the Teradata optimizer has continued to evolve over time to meet changing customer needs.

Adaptive optimization

Typically, the optimizer will build a static query plan that is then executed by the system. However, the optimizer also can interleave optimization with execution within a single query. This is called adaptive optimization.

Adaptive optimization is not wasted on short or simple queries. The optimizer picks and chooses when to apply this technique and determines eligible requests based on certain thresholds—for example, expected execution time.

Adaptive optimization builds what is called a "dynamic query plan." It breaks a query into blocks of steps called "fragments." A plan is built for the first fragment, it executes on the AMPs, and intermediate results are returned to the optimizer. Then, input from the previous fragment is considered and the next fragment is optimized, then executed.

Query rewrite

Another advanced optimizer technique is to rewrite queries to eliminate redundant logic. One example is producing a temporary dataset within a query to use as input to multiple subqueries later in the plan, rather than rebuilding the data set multiple times.

Other examples of query rewrite are moving blocks of SQL code in the query around to achieve predicate simplification, view folding for more optimal join plans, and "projection pushdown," which eliminates unnecessary column projections.

Joining relational data to external object stores

The Teradata optimizer can incorporate several different open file formats (OFF), such as Parquet, JSON, and CSV, as well as open table formats (OTFs), such as Apache Iceberg, into a single query execution. It can then join that external data to relational tables inside the database.

Key to optimizing queries with external object store data is the distribution of objects evenly across all AMPs in the configuration to leverage Teradata's parallel architecture. In a later chapter ("Evolution: Being parallel in the ecosystem"), additional information about how this processing balance has been achieved is explained for the initial external object tier, OFF.

While the optimizer may not hold the purse strings, it certainly has a major influence in how and where money is spent. The goal in all these ever-evolving optimizer capabilities is the same: Ensuring that Teradata customers enjoy the best performance and lowest possible cost per query.

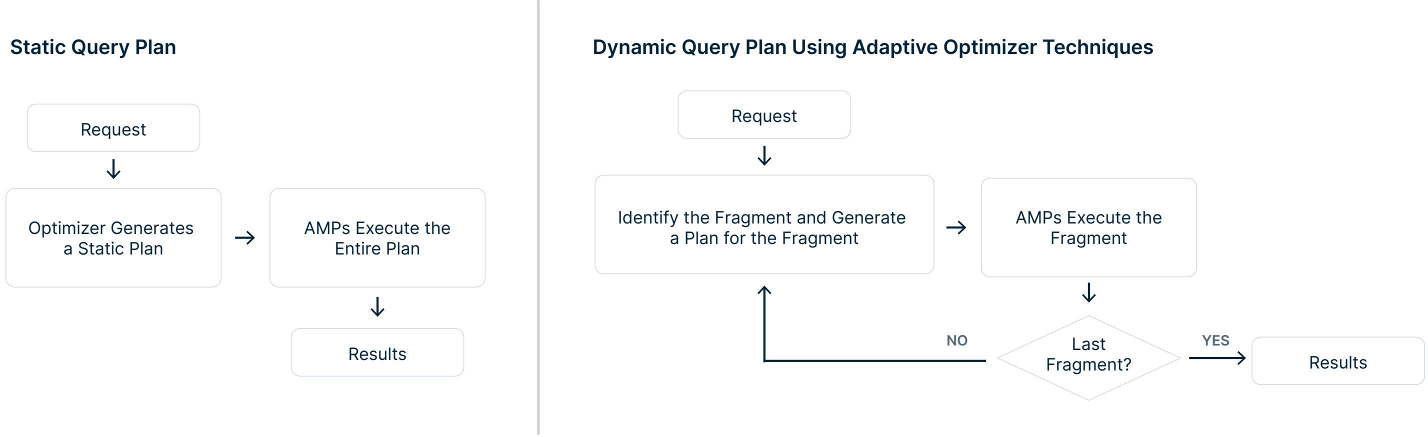

Static Query Plan
Request
Optimizer Generates a Static Plan
AMPs Execute the Entire Plan
Results
Dynamic Query Plan Using Adaptive Optimizer Techniques
Request
Identify the Fragment and Generate a Plan for the Fragment
AMPs Execute the Fragment
Last Fragment?
NO
YES
Results

Figure 5. Adaptive optimizer generates a dynamic query plan

Born with:
BYNET's considerable contribution

Another important component of the Teradata architecture is referred to as the "BYNET." This acts as the interconnection between all the independent parallel components. Originally implemented within the hardware of traditional on-premises Teradata systems, this functionality is now implemented as software.

Beyond just passing messages, the BYNET is a bundle of intelligence and low-level functions that aid in efficient processing at practically each point in a query's life. It offers coordination as well as oversight and control to every optimized query step.

In short, the BYNET acts as an air traffic controller, ensuring that the entire system is working in concert and managing unusual situations as they arise. This can include adjusting to hardware failures, monitoring for points of congestion, or even sequencing results from across parallel units.

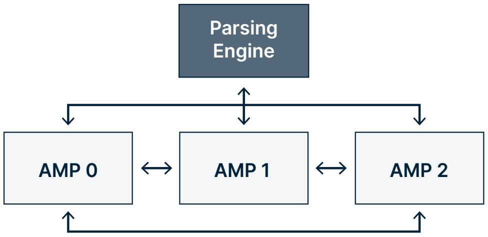


Figure 6. AMPs and parsing engines communicate using messages

Messaging

A key role of the BYNET is to support communication between the PE and the AMPs and from AMPs to other AMPs. These simple message-passing requirements are fulfilled using a low-level messaging approach, bypassing more heavyweight protocols for communication to:

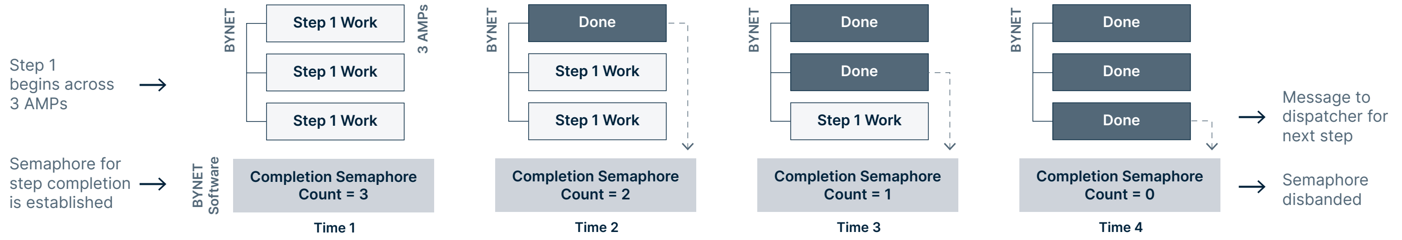
- Send a step from the PE to AMPs to initiate a query step
- Redistribute rows from one AMP to another to support different join geographies
- Sort and merge a (potentially very large) answer set across multiple AMPs

The BYNET illustrates how cost savings inside the engine have been instilled into Teradata from the very beginning. Even though message protocols are low cost, Teradata goes further by minimizing interconnect traffic. Localized activity that can be performed without moving data outside of an AMP is encouraged whenever possible.

BYNET communication between AMPs

Without the BYNET's ability to combine and consolidate information acrosts all units of parallelism, each AMP would have to independently connect to each other AMP in the system to get on the same page about each query step that is underway. As the configuration grew, such a distributed approach to coordinating query work would quickly become a bottleneck.

Instead, the BYNET creates a dynamic relationship between AMPs that are working on the same query step. This loose association of AMPs at run time is used to communicate things between AMPs, like the completion or the success/failure of a step on each participating AMP. This is accomplished by signaling across semaphores, rather than more expensive messaging.



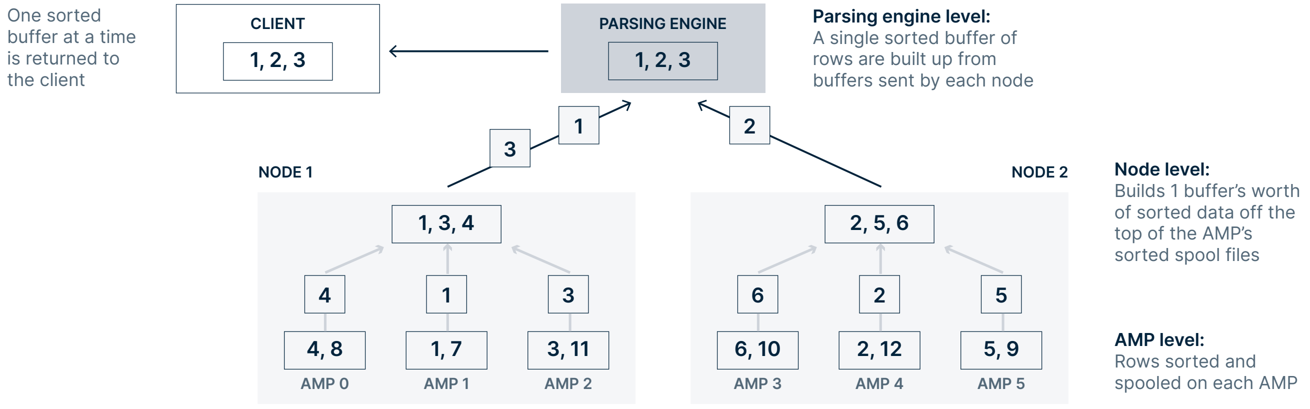
Step 1 begins across 3 AMPs
Semaphore for step completion is established
BYNET
3 AMPs
BYNET Software
Step 1 Work
Step 1 Work
Step 1 Work
Completion Semaphore Count = 3
Time 1
Done
Step 1 Work
Step 1 Work
Completion Semaphore Count = 2
Time 2
Done
Done
Step 1 Work
Completion Semaphore Count = 1
Time 3
Done
Done
Done
Completion Semaphore Count = 0
Time 4
Message to dispatcher for next step
Semaphore disbanded

Figure 7. Communication of AMPs using completion semaphores

Final answer set sort/merge

Never needing to materialize a query's final answer set inside the database has long been a Teradata differentiator. The final sort/merge of a query takes place within the BYNET as the answer set rows are being funneled up to the client. This dynamic no-I/O merge is taking place simultaneously at the AMP, the node, and, finally, the PE level, with the highest values in the sort order being elevated into the answer set in the client buffer first, until the client is ready to receive more. The final answer set never has to be brought together, saving considerable resources. A potential "big sort" penalty has been eliminated—or never existed.

> The goal in all these ever-evolving optimizer capabilities is the same: Ensuring that Teradata customers enjoy the best performance and lowest possible cost.



One sorted buffer at a time is returned to the client
CLIENT
1, 2, 3
PARSING ENGINE
1, 2, 3
Parsing engine level: A single sorted buffer of rows are built up from buffers sent by each node
1
3
2
NODE 1
NODE 2
1, 3, 4
4
1
3
4, 8
1, 7
3, 11
AMP 0
AMP 1
AMP 2
2, 5, 6
6
2
5
6, 10
2, 12
5, 9
AMP 3
AMP 4
AMP 5
Node level: Builds 1 buffer's worth of sorted data off the top of the AMP's sorted spool files
AMP level: Rows sorted and spooled on each AMP

Figure 8. Sorting the final answer aset inside the BYNET

Born with:
Workflow self-regulation

A shared-nothing parallel database has a special challenge when it comes to knowing how much new work it can accept and how to identify congestion that may start to build up inside one or more of the parallel units. Inherent in its design, the optimizer aggressively applies multiple dimensions of parallelism to each query that it sees. This approach maximizes resource utilization for each query to deliver performance and throughput. But it also means that it's easy to exhaust overall system resources. Fortunately, Teradata's platform has numerous techniques to manage workflow to ensure optimal use of all system resources.

AMP-level control

The Teradata database manages the flow of work that enters the system in a highly decentralized manner, in keeping with its shared-nothing architecture There are no messages sent between AMPs to determine if it's time to hold back new requests. Each AMP evaluates its own ability to take on more work and temporarily pushes back when it experiences a heavier load than it can efficiently process. And when an AMP does pause to catch its breath, it does so for the briefest moments of time, often measured in milliseconds.

This bottom-up control over the flow of work inside the engine is the cornerstone of the database's ability to accept impromptu swings of very high and very low demand and gracefully and unobtrusively accommodate whatever comes its way.

AMP worker tasks

AMP worker tasks (AWTs) are the tasks inside each AMP that get the database work done. Pre-allocated AWTs are assigned to each AMP at database startup time, and, like a valet parking attendant, they wait for work to arrive, do the work, and come back for more work.

Because of their stateless condition, AWTs respond quickly to a variety of database execution needs. There are a fixed number of AWTs in each AMP. For a task to start running, it must acquire an available AWT. Having an upper limit on the number of AWTs per AMP keeps the number of activities performing database work within each AMP at a reasonable level. AWTs play the role of both expeditor and governor.

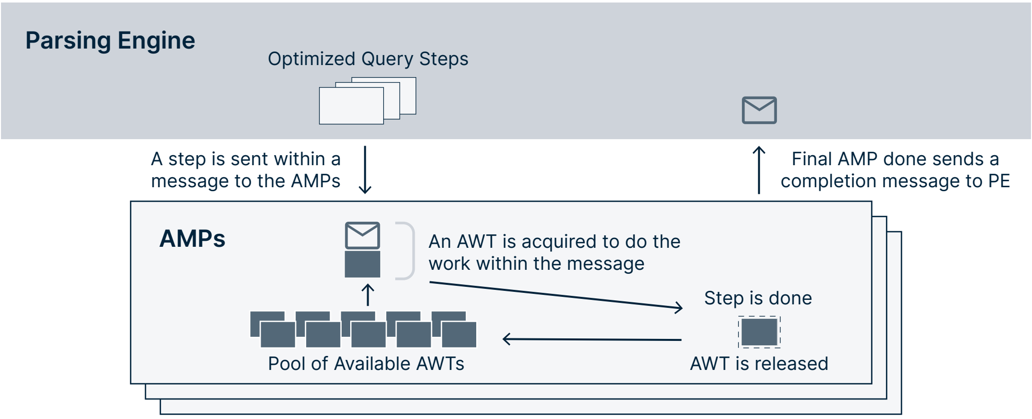

Parsing Engine
Optimized Query Steps
A step is sent within a message to the AMPs
Final AMP done sends a completion message to PE
AMPs
An AWT is acquired to do the work within the message
Step is done
Pool of Available AWTs
AWT is released

Figure 9. AWTs process query steps on the AMPs

Queueing up and turning away new messages

When all the AMP worker tasks on an AMP are busy servicing other query steps, arriving work messages are placed in a message queue that resides in the AMP's memory. When a message is sent to multiple AMPs, some AMPs may provide an AWT immediately, while other AMPs may have to queue the message. This is typical behavior on a busy system where each AMP is managing its own flow of work.

If the message queue gets too long, arriving messages will be rejected and returned to the sender to be re-sent later. The impact of turning on and turning off the flow of messages is kept local—only the AMP hit by an overabundance of messages for that brief period throttles back temporarily.

Riding the wave of full usage

Teradata's platform was designed as a throughput engine, able to exploit parallelism to maximize resource usage of each request when only a few queries are active, while at the same time able to continue churning out answer sets in high demand situations. To protect overall system health under extreme usage conditions, highly decentralized internal controls were put into the foundation, as has been discussed in this section.

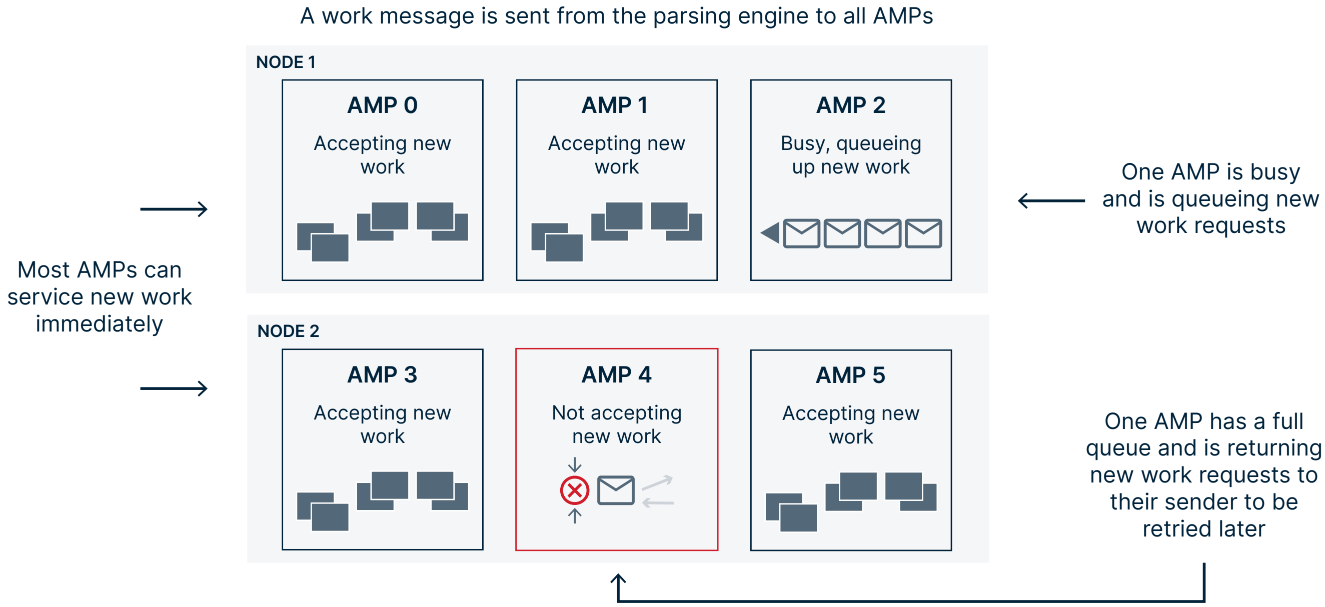

A work message is sent from the parsing engine to all AMPs
NODE 1
AMP 0
Accepting new work
AMP 1
Accepting new work
AMP 2
Busy, queueing up new work
One AMP is busy and is queueing new work requests
Most AMPs can service new work immediately
NODE 2
AMP 3
Accepting new work
AMP 4
Not accepting new work
AMP 5
Accepting new work
One AMP has a full queue and is returning new work requests to their sender to be retried later

Figure 10. Individual AMPs push back temporarily when tehy have too much work

Born with:
Workload management

An earlier section in this white paper called attention to the multifaceted parallelism available for queries on the Teradata database. A subsequent section discussed how the optimizer uses those parallel opportunities in smart ways to improve performance on a query-by-query basis. And the previous section illustrated internal AMP-level controls to keep high levels of user demand and an overabundance of parallelism manageable.

In addition to those automatic controls at the PE and AMP levels, Teradata has always had some type of system-level workload management, mainly priority differences, that are used by the internal database routines.

The original four priorities

One of the challenges faced by the original architects of the Teradata database was how to support maximum levels of user requests on the platform and still get critical pieces of internal database code to run quickly when needed. For example, if there is a rollback taking place due to an aborted transaction, it benefits the entire system if the reversal of updates to clean up the failure can happen quickly.

It was also important to ensure that background tasks running inside the database didn't lag too far behind. If city streets are so congested with automobile traffic that the weekly garbage truck can't get through and is delayed for weeks at a time, a health crisis could arise.

The solution the architects found was a simple priority scheme that assigned priorities to all tasks running on the system, either user initiated or internal. This rudimentary approach offered four priority buckets: Rush, High, Medium, and Low, with a default of Medium.

Internal database routines and parts of query code could be assigned to one of the other priorities, based on the importance of the work. For example, all "end transaction" steps are assigned the Rush priority, because finishing almost-completed work at an accelerated speed frees up valuable resources for new work sooner and was seen as critical to prevent congestion within the database. In addition, if the administrator wanted to give a favored user a higher priority, all that was involved was manually adding one of the priority identifiers into the user's account string.

Impact of mixed workloads

Customer requirements shifted over time as Teradata users began to supplement their traditional decision support queries with new types of more varied workloads.

In the late 1990s, a few Teradata sites began to issue direct lookup queries against entities like their inventory tables or their customer databases at the same time as their standard decision support queries were running. Call centers started using data in their Teradata database to validate customer accounts and recent interactions. Online applications blossomed at the same time as more sites turned to continuous loading to supplement their batch windows, giving their end users more timely access to recent activity. Service level goals became more important. Today it is typical for 80% to 90% of the queries on a Teradata system to execute in less than a second.

Stronger, more flexible workload management was required.

Evolution:
Enhancements to workload management

While the internal management of the flow of work has changed little, the capabilities within system-level workload management have expanded dramatically over the last 25 years. As the first step beyond the original four priorities, Teradata engineering developed a more extensive priority scheduler composed of multiple resource partitions and performance groups, and provided the flexibility of assigning your own customized weighting values. These custom weightings and additional enhancements made it easier to match controls to business workloads and priorities than the original capabilities designed more for controlling internal system work. This became particularly important for fixed-size enterprise platforms.

Additional workload management features and options that have evolved over the years include:

- Ability to assign priorities and other controls by multiple classifications, such as server name, application, database objects referenced, or the optimizer's estimated statistics
- Concurrency control mechanisms that can be placed at multiple levels and tailored to specific types of queries
- Rules to reject queries that are poorly written or that are inappropriate to run at certain times of the day
- Ability to automatically reduce the priority of a running query that exceeds the threshold of resources consumed for its current priority

Workload management in Teradata has proven to be a rapidly expanding area, indispensable to customers who are combining a wide variety of work on their Teradata platforms.

Evolution: Database extensibility

Teradata has a rich tradition of continually extending its capabilities, both internally as well as externally. Column-based tables, time series databases, and features like geospatial and temporal data types are all part of the Teradata platform today. The solid MPP foundation has eased the way for these innovations.

The growth of special functions

Teradata allows users to create user-defined functions (UDFs) to extend SQL functionality, including scalar, aggregate, table, table operators, and external stored procedures. This portfolio of special functions allows end users and Teradata partners to create custom database objects in Java or C++ to provide capabilities that were not previously a part of the Teradata database. Some of these functions even extend their activity outside the database.

In-database analytics

Teradata was the first company in the early 2000s to pioneer parallel, scalable in-database analytics leveraging extensibility features as described above. Over time, as machine learning and artificial intelligence (AI/ML) became ubiquitous within Teradata's customer base and data volumes grew to unprecedented sizes, it became necessary to build low-level internal interfaces for these types of analytics.

The implementation of these internal interfaces allowed Teradata to scale beyond what was previously thought possible for use cases requiring supervised and unsupervised learning. This includes classification, regression, segmentation, time series, and digital signal processing. These low-level frameworks are unbounded in terms of the number of variables and the number of series or signals they can process simultaneously.

These extensibility features also opened the possibility of processing languages other than traditional SQL. This has led to a whole new set of user personas and use cases built upon the power of the Teradata platform.

> Teradata's ability to grow in new directions and continue to sustain its core competencies is a direct result of its strong, tried-and-true foundation.

Bring Your Own Analytics

It has always been critical for Teradata to be a team player in an overall open analytical ecosystem. For example, Teradata's shared-nothing MPP architecture is the perfect vehicle for model inferencing or scoring—it's tactical in nature with all the data needed for inferencing available on a single unit of parallelism.

This allows Teradata to be open to any analytical modeling tool that can produce a consumable model format. And, for certain strategic partners, such as H20.ai, Dataiku, DataRobot, and SAS, Teradata's platform can consume their models by binding to their run-time environments. Using either of these techniques, Teradata's platform has become a massively parallel scoring engine in many customers' open analytic ecosystem.

Further, as the popularity of the Python and R open-source languages and packages increased in recent years, many Python and R scripts have been put into production environments as analytic pipelines. Building on Teradata's ability to process these languages through table operators, these pipelines can be optimized and run in parallel—even Spark-based Python scripts.

Evolution:
Being parallel in the ecosystem

This chapter covers additional evolutionary improvements that were made to extend Teradata's parallelism to the world outside of the database.

The Native Object Store feature

In today's environment, data of interest may reside in other file systems or data management platforms. Teradata's Native Object Store feature allows users to directly read or write data stored in cloud vendor open file format (OFF) object stores using Teradata SQL. The optimizer will assign the task of reading and transforming CSV, Parquet, or JSON files across all AMPs, so as to take full advantage of the inherent parallelism of the system.

This means OFF data is never out of reach, even though it is stored outside of the database.

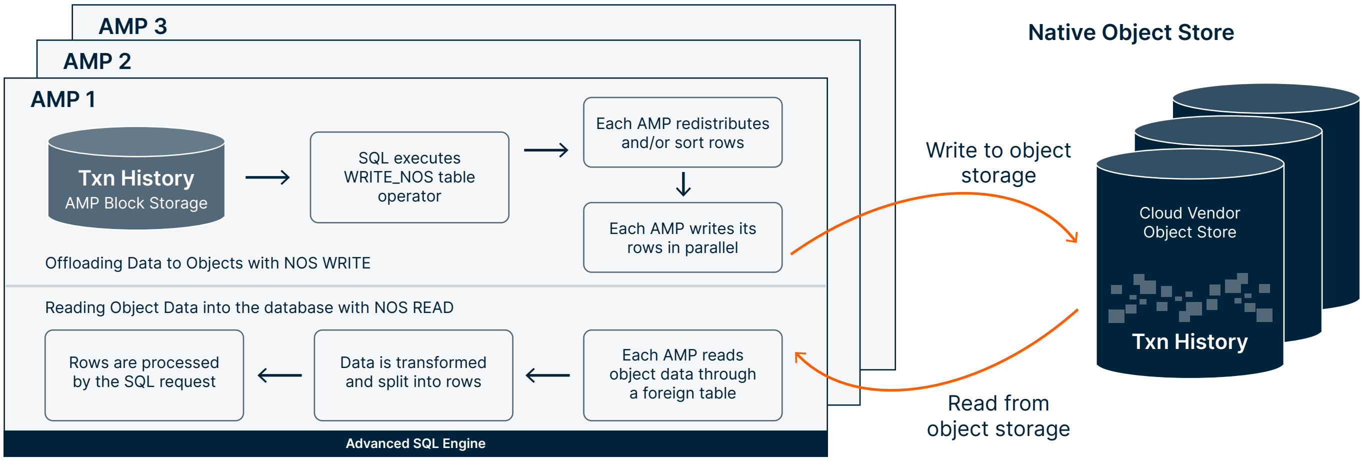
QueryGrid™

QueryGrid is a data analytics fabric that provides data access, processing, and table-level data movement across one or more data sources to enable federated queries. The data sources can be of the same type, such as one Teradata platform connecting to another Teradata platform, or different types, such as a Teradata platform connecting to a remote server, such as a Google BigQuery instance.

One of the things that makes QueryGrid unique is the ability to push predicate filtering and processing closer to the data to minimize the amount of data that must move between the two data sources. QueryGrid enables customers to:

- Minimize data movement and process data where it resides
- Reduce data duplication
- Transparently automate analytic processing and table-level data movement between data sources

QueryGrid was architected to take advantage of Teradata's built-in parallelism, enhancing its performance capabilities. Each AMP connects to and processes a subset of the data from a remote server in parallel.



AMP 3
AMP 2
AMP 1
Txn History
AMP Block Storage
SQL executes WRITE_NOS table operator
Each AMP redistributes and/or sort rows
Each AMP writes its rows in parallel
Offloading Data to Objects with NOS WRITE
Reading Object Data into the database with NOS READ
Rows are processed by the SQL request
Data is transformed and split into rows
Each AMP reads object data through a foreign table
Advanced SQL Engine
Native Object Store
Write to object storage
Read from object storage
Cloud Vendor Object Store
Txn History

Figure 11. Native Object Store parallelizes access to object file format tables

Today:
Cloud-native Teradata VantageCloud Lake

VantageCloud Lake, Teradata's complete cloud analytics and data platform for AI, builds upon the foundation established in the enterprise world. This section provides a high-level description of VantageCloud Lake, specifically how its architecture leverages the best features of the cloud while still preserving the key benefits of the Teradata database.

Primary and compute clusters

The VantageCloud Lake architecture moves Teradata from being a single large system on a fixed set of BYNET-connected nodes into a logical collection of smaller clusters that become the building blocks making up the VantageCloud Lake environment.

Each VantageCloud Lake cluster is a discrete set of processing units or nodes that are BYNET-connected, with each node containing the familiar AMPs and parsing engines just as does any other traditional Teradata node. Each VantageCloud Lake cluster delivers parallelism across its AMPs independent from other clusters.

A primary cluster is required in all VantageCloud Lake environments. A primary cluster includes an always-on, fixed set of nodes whose AMPs own data in persistent block storage managed by the block file system (BFS). All queries begin and end their execution in the primary cluster. Query parsing and optimizing, as well as final answer set prep, takes place on the primary cluster.

A second type of cluster is called a compute cluster. A compute cluster is a collection of ephemeral compute-only nodes where query work can be sent from the primary cluster for execution. Compute clusters contain AMPs and parsing engines, but, unlike primary clusters, they don't have any persistent block storage for user data and are focused on accessing data from object storage.

Automatic addition or removal of processing power, called autoscaling, can take place among these compute clusters. Internal thresholds based on resource demand and usage trigger these changes in compute power.

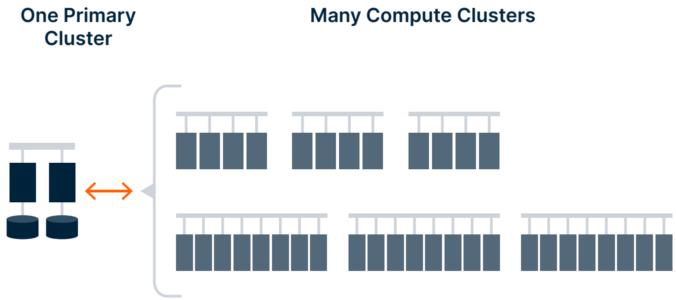


Figure 12. VantageCloud Lake architecture

Cloud object storage

VantageCloud Lake supports the separation of compute and storage. All clusters in a VantageCloud Lake environment can access the same public cloud vendor object storage. And because cloud object storage is less expensive than block storage, it can be used for massive amounts of business data to be stored and analyzed cost effectively. However, there is still a place for BFS because of its unique capabilities, and the primary cluster can take advantage of BFS when appropriate.

Several tiers of object storage are accessible by VantageCloud Lake:

- Object file system (OFS) storage: This is proprietary object storage managed by the VantageCloud Lake environment that is dedicated to user tables. Tables stored in OFS support inserts, updates, and deletes, as well as time travel. OFS has been optimized with special indexing techniques to support efficient access.

- Open file format (OFF): OFF object storage is external to the VantageCloud Lake environment, and, depending on security conventions, may be shared across multiple other platforms or with trusted third parties. The definition of the data is held within a "foreign table" stored within the Teradata data dictionary. The Native Object Store (NOS) feature is the method of reading and writing OFF data.

- Open table format (OTF) storage: Tables stored in OTF are also external to VantageCloud Lake. OTF is an industry-standard "open," ACID-compliant table format that supports open file formats. With OTF, none of the table's metadata is stored in the Teradata data dictionary. Instead, all the metadata is found within the object store itself.

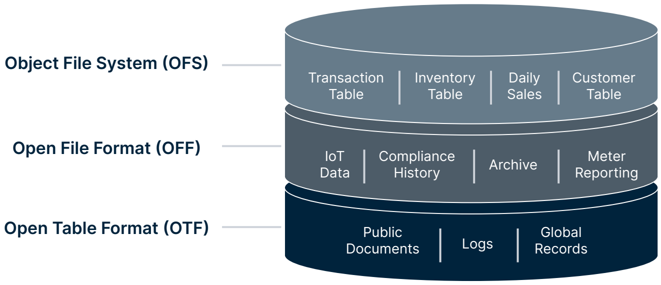


Figure 13. Multiple object storage tiers accessible by VantageCloud Lake

Optimizer enhancements for VantageCloud Lake

The optimizer as it exists in VantageCloud Lake has all the special techniques and maturity that the enterprise optimizer has perfected over the years. However, the optimizer also includes some new enhancements that recognize and take advantage of the new VantageCloud Lake architecture.

Global planner

A new component in the primary cluster optimizer, called the "global planner," determines where each step in a query plan will execute, on primary or compute cluster, if compute clusters are available. Every effort is made to push resource-intensive work, such as reading and processing data from object storage or doing advanced analytics, onto the compute clusters. When the VantageCloud Lake optimizer builds a query plan, it recognizes and considers the level of processing power within each cluster.

Pipelining between query steps

Pipelining is a query optimization extension available on VantageCloud Lake environments. Pipelining can improve the elapsed time of a query by eliminating intermediate files between steps. Traditionally, a producer step in a query will write all qualifying rows to a temporary file to disk, and subsequent consumer step will read the file. With pipelining, a producer step generates rows and passes them on to a consumer step in memory without having to write a temporary dataset to disk. With pipelining, data is consumed as soon as it is produced.

Pipelining in VantageCloud Lake is an option the optimizer can apply. Pipelining in VantageCloud Lake is applied selectively during query optimization, based on a variety of query characteristics. Because it is one of many optimizer options, pipelining in VantageCloud Lake preserves the existing nuances built into the original optimized query plan. A key goal of pipelining in VantageCloud Lake is to apply it where it is practical and beneficial, while at the same time honoring the query plan as it was optimized.

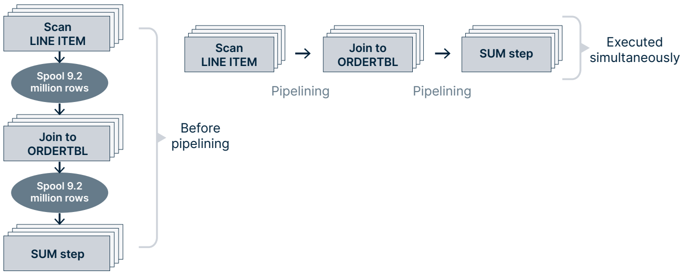


Figure 14. VantageCloud Lake offers a hybrid approach to pipelining between query steps

A multicloud-aware optimizer (Amazon Web Services, Microsoft Azure, Google Cloud)

In the Teradata platform, cost profiles have evolved to help provide information to the optimizer about underlying hardware capabilities of the specific configuration, influencing the development of query plans. This capability has been expanded in VantageCloud Lake to include cost profiles associated with different platforms and their instance types, including Amazon Web Services (AWS), Azure, and Google Cloud.

For example, cost profiles might include CPU MIPS ratings of a particular node type or I/O throughput ratings. Using cost profiles, the optimizer can distinguish different vendor configurations and create optimal plans around each.

Multi-storage connectivity in a single query

There may be a valid business need to pull IoT data from an OTF table, join it to a highly curated customer table in BFS storage, and then join the results to an OFF history table. This can be accomplished in VantageCloud Lake within a single query because of the open and connected architectures of VantageCloud Lake and the intelligence of the optimizer.

The query optimizer in VantageCloud Lake is aware of the location of data and performance characteristics specific to each storage tier (BFS, OFS, OFF, or OTF). And, as mentioned in the section above, there is awareness of the power of the different vendors' underlying hardware. This attention to detail makes it possible to join disparate data sources in a single query in an optimal manner.

Automated workload management in VantageCloud Lake

A VantageCloud Lake environment can support multiple clusters. Each cluster performs work independently of other clusters. Because individual clusters are isolated from all other clusters, there is less requirement for the complex workload management that has evolved for fixed-system enterprise platforms.

Consequently, Lake comes with a default workload management rule set with a simplified set of priorities that are automatically set up for each cluster. The administrator can assign specific users to one of these default workloads if they wish to have "explicit" priority differentiation among the work active within the cluster.

On the other hand, queries entering a VantageCloud Lake environment from users who are not explicitly prioritized will benefit from automated prioritization. One of five "implicit" priorities will be selected, based on expected execution time of the query.

Once they begin to execute, queries that have been implicitly prioritized will automatically be demoted to a lower priority if their accumulated resource consumption exceeds a specified threshold.

This simple, built-in workload management means users new to Teradata do not have to be concerned about applying complex rules or conditions on their workloads to use VantageCloud Lake successfully. And experienced users can free themselves from having to setup and monitor their workload management decisions.

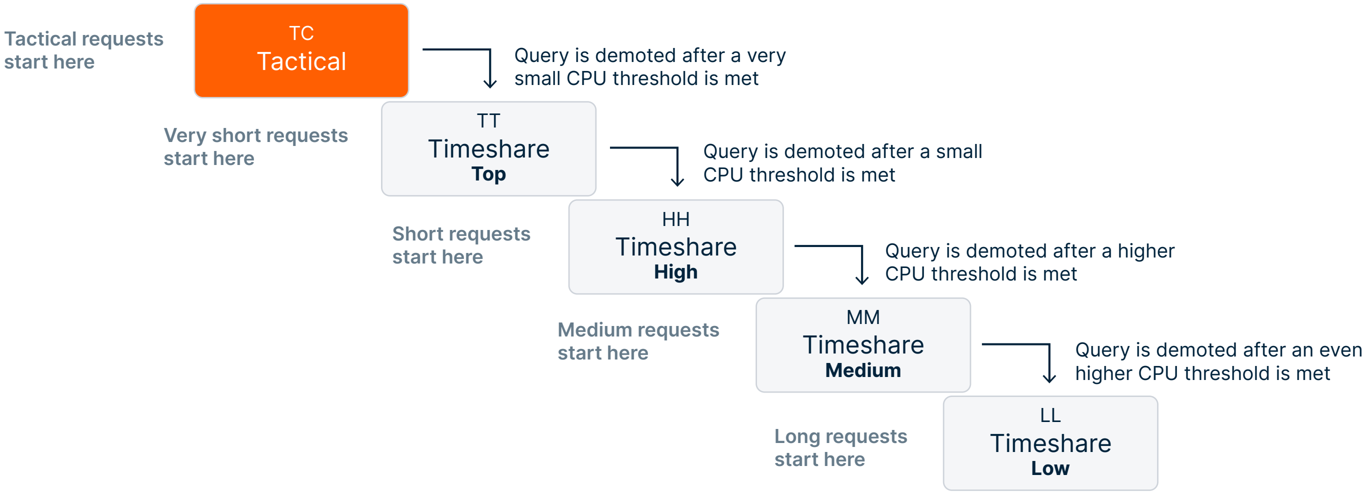

Tactical requests start here
TC
Tactical
Query is demoted after a very small CPU threshold is met
Very short requests start here
TT
Timeshare
Top
Query is demoted after a small CPU threshold is met
Short requests start here
HH
Timeshare
High
Query is demoted after a higher CPU threshold is met
Medium requests start here
MM
Timeshare
Medium
Query is demoted after an even higher CPU threshold is met
Long requests start here
LL
Timeshare
Low

Figure 15. Automated prioritization in VantageCloud Lake

Today:
Preexisting features enrich VantageCloud Lake capabilities

This section discusses some of the Teradata capabilities that emerged during the enterprise years that have energized and strengthened the VantageCloud Lake offering.

Evolution of QueryGrid™

The original QueryGrid feature is provisioned on VantageCloud Lake if requested by a customer. The installation on the nodes is handled automatically as a part of QueryGrid provisioning. The QueryGrid setup on Teradata enterprise platforms or non-Teradata systems requires the customer to step through a self-service guided path, provided through the VantageCloud Lake console.

Hive, Spark, and generic JDBC connectors are currently supported on VantageCloud Lake for both AWS and Azure. Generic JDBC and BigQuery are imminently available, with Hive/Spark planned to follow shortly after. Additional QueryGrid connectors will be available in the future, providing VantageCloud Lake with even more outreach to other data sources. Because each VantageCloud Lake environment can connect to an on-premises platform and other supported platforms within a CSP, expanded QueryGrid enables a true hybrid multi-cloud solution.

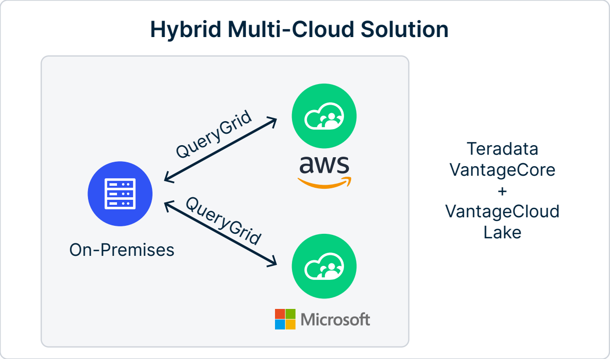

Hybrid Multi-Cloud Solution
QueryGrid
QueryGrid
aws
On-Premises
Microsoft
Teradata VantageCore + VantageCloud Lake

Figure 16. QueryGrid enables a hybrid multi-cloud solution

More importantly, QueryGrid technology has been built into the infrastructure of VantageCloud Lake as a pathway connecting primary and compute clusters in a way that is transparent to users. This internal communication layer supports the passing of data, metadata, and optimized query steps between different clusters and relies upon the well-established QueryGrid functionality that has existed for many years prior to VantageCloud Lake.

Original QueryGrid™ Techniques Benefiting VantageCloud Lake

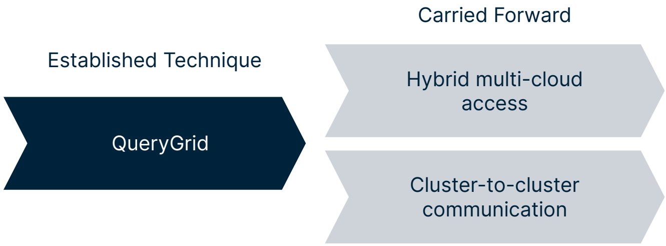


Figure 17. QueryGrid repurposed to strengthen VantageCloud Lake

Evolution of existing techniques to enable OTFs

OTFs let customers store data in an open and standard way that's easily shared across different compute, analytic engines, and tools without needing to manage multiple copies of large datasets. Teradata supports a variety of OTFs and open catalogs in multi-cloud and multi-data-lake environments. Supporting multiple open catalogs, such as AWS Glue, Hive, and Unity, removes the need for metadata replication and promotes interoperability. Iceberg and Delta Lake are the two OTFs initially supported on VantageCloud Lake.

Access of OTF tables from a VantageCloud Lake environment is implemented similarly to accessing OFF tables using Native Object Store. Each AMP in a cluster is given some share of the objects of an OTF table to read. And just as with Native Object Storage access, all manipulations and transformations are performed automatically and in parallel. The optimizer will attempt to send all steps that access data in OTF to compute clusters and builds a plan based on the cluster's available processing power and the parallelism it offers.

One difference with OTF and other storage tiers supported on VantageCloud Lake is that none of an OTF table's metadata is stored in the data dictionary, but rather within the object store itself. The format, makeup, and protocol for interacting with the metadata is defined within the Iceberg and/or Delta Lake specifications, an approach which enables other OTF vendors to simultaneously access and manipulate the OTF dataset. For security purposes, however, all access to OTF data requires privileges to be granted.

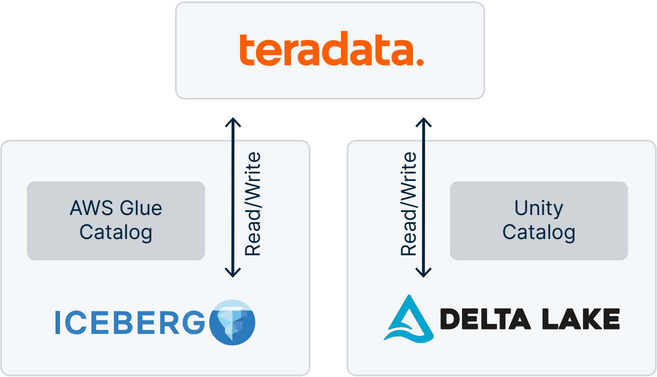


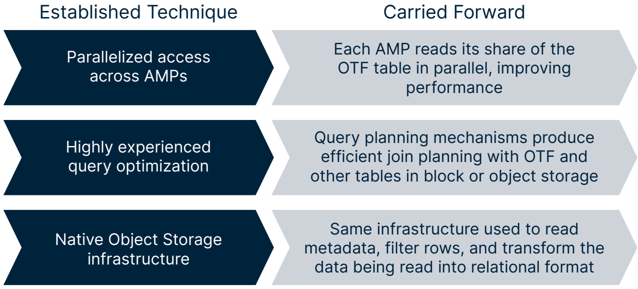
Figure 18. Read/write access to OTF data in both Iceberg and Delta Lake

OTF tables and catalogs reside in customer-owned and managed cloud storage. Teradata OTF integrates with cloud providers' authentication and authorization mechanisms and adheres to access controls to OTF tables and metadata. In addition, the network connections are secured and all data in transit and at rest is encrypted.

Teradata's single-AMP access optimization provides a highly efficient way to retrieve OTF table information for schema discovery and also for accessing current metadata in a specific catalog. But when it comes to accessing the OTF data itself, each AMP shares in the work of reading and transforming the OTF data. The same optimizer that has been enabling performance of complex queries for decades has made OTF tables a part of the family, so they can be easily joined with other relational and non-relational data from any storage tier.

Everything running on Teradata's platform, including OTF read or write, is tightly integrated with workload management. These ever-present workload management techniques mean that the administrator can assign OTF read or write to execute at any desired priority point and control concurrency effectively.

Techniques Benefiting VantageCloud Lake's Open Table Format Implementation



Established Technique	Carried Forward
Parallelized access across AMPs	Each AMP reads its share of the OTF table in parallel, improving performance
Highly experienced query optimization	Query planning mechanisms produce efficient join planning with OTF and other tables in block or object storage
Native Object Storage infrastructure	Same infrastructure used to read metadata, filter rows, and transform the data being read into relational format

Figure 19. OTF implementation builds on established strengths

Evolution of analytics in VantageCloud Lake

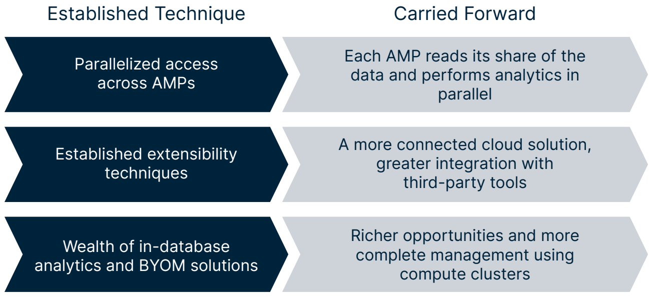
All the extensibility mechanisms discussed earlier (UDFs, external stored procedures, in-database analytics, and support for open-source languages) are highly useful in VantageCloud Lake today. A key advantage of VantageCloud Lake is that these often-intense applications can be isolated into their own set of compute clusters, where they will benefit from dedicated resources and will not interfere with other active work.

But an even more important opportunity relevant to analytics has evolved in VantageCloud Lake: specialized compute clusters. Specialized compute clusters are designed specifically to meet the needs of complex and resource-intensive analytic applications.

For example, the analytic cluster is a specialized cluster for executing user scripts within containers against table data using Teradata's open analytics framework. What makes analytic compute clusters different is that they come with specialized hardware, and statements executing on them have more memory and greater CPU available to them.

A second type of specialized compute clusters are graphics processing unit (GPU) clusters. In a VantageCloud Lake environment, specialized GPU units can be provisioned and leveraged for performing complex mathematical calculations in an efficient and parallel way.

Existing Techniques Benefiting Analytics Processing in VantageCloud Lake



Established Technique	Carried Forward
Parallelized access across AMPs	Each AMP reads its share of the data and performs analytics in parallel
Established extensibility techniques	A more connected cloud solution, greater integration with third-party tools
Wealth of in-database analytics and BYOM solutions	Richer opportunities and more complete management using compute clusters

Figure 20. Analytics in VantageCloud Lake are boosted by established Teradata capabilities

The evolution of AI

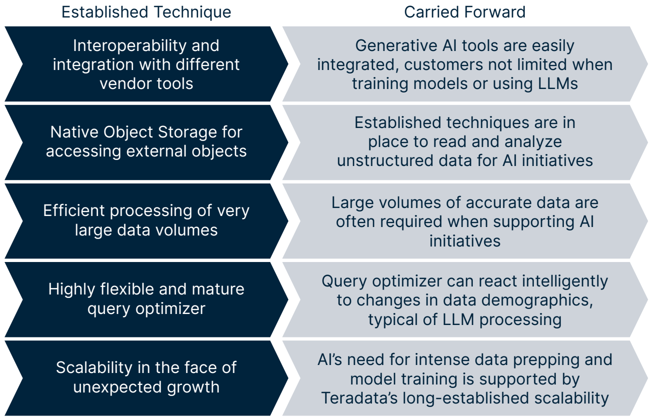
Although generative AI is not new in concept, it is only since late 2022, when OpenAI launched ChatGPT, that the technology began getting a lot of attention.

VantageCloud Lake appeared at about the same time as AI was becoming mainstream. VantageCloud Lake offers an environment that supports large language models (LLMs) and also supports generative AI with the benefit of running in parallel for performance and scalability. Teradata's extensibility features along with the VantageCloud Lake specialized compute clusters makes this possible. Bring Your Own LLM is now realizable for both inferencing and fine-tuning models against a customer's actual data without reaching out to public LLMs where this same data might not be as secure.

VantageCloud Lake can also integrate with other vendors' LLMs, such as Amazon Bedrock and Azure OpenAI Service. This is the approach taken by Teradata's ask.ai product, an intelligent document search capability, which can evolve to do natural query language. For example, ask.ai can provide the correct SQL syntax for a detailed request, such as calculating revenue totals for the current month by region, and then presenting the results in a meaningful.

All these evolutionary threads in Teradata's past in the areas of database extensibility, in-database analytics, and AI have come to fruition inside the VantageCloud Lake offering today. The new cloud architecture strengthens these earlier capabilities, allowing Teradata customers to do things that would have been unbelievable just a year or two ago.

Teradata Strengths Carry Over Into VantageCloud Lake AI Capabilities



Established Technique	Carried Forward
Interoperability and integration with different vendor tools	Generative AI tools are easily integrated, customers not limited when training models or using LLMs
Native Object Storage for accessing external objects	Established techniques are in place to read and analyze unstructured data for AI initiatives
Efficient processing of very large data volumes	Large volumes of accurate data are often required when supporting AI initiatives
Highly flexible and mature query optimizer	Query optimizer can react intelligently to changes in data demographics, typical of LLM processing
Scalability in the face of unexpected growth	AI's need for intense data prepping and model training is supported by Teradata's long-established scalability

Figure 21. VantageCloud Lake AI capabilities are boosted by existing Teradata capabilities

Conclusion

Foundations are important. Teradata's ability to reach out in new directions and continue to sustain its core competencies is a direct result of its strong, tried-and-true foundation. As the Teradata database has grown and matured, the same fundamentals have been adapted into new technology advances. This is particularly true with VantageCloud Lake.

With an understanding of the foundational capabilities that make up important building blocks of the Teradata database, you can appreciate the elegance and durability of the architecture. These features have, in many ways, stayed consistent. At the same time, many have evolved from that original architecture—building on it rather than replacing it.

These foundational components have proven essential each step of the way as Teradata has continued to move forward—whether integrating with mainframe databases, managing mixed workloads, or advancing the world of analytics. Being born in the enterprise does indeed foreshadow strength in the cloud.

About Teradata

At Teradata, we believe that people thrive when empowered with trusted information. We offer the most complete cloud analytics and data platform for AI. By delivering harmonized data and trusted AI, we enable more confident decision-making, unlock faster innovation, and drive the impactful business results organizations need most. See how at **Teradata.com**.

17095 Via Del Campo, San Diego, CA 92127 Teradata.com

The Teradata logo is a trademark, and Teradata is a registered trademark of Teradata Corporation and/or its affiliates in the U.S. and worldwide. Teradata continually improves products as new technologies and components become available. Teradata, therefore, reserves the right to change specifications without prior notice. All features, functions and operations described herein may not be marketed in all parts of the world. Consult your Teradata representative or Teradata.com for more information.

© 2024 Teradata Corporation All Rights Reserved. Produced in U.S.A. 09.24

teradata.