Approaches for Data Loading

HP Vertica Analytic Database

HP Big Data Foundations
Document Release Date: July, 2015

Contents

Bulk Loading with the COPY Statement

The COPY statement is the most efficient way to load large amounts of data into an HP Vertica database. You can copy one or more files onto a cluster host using the COPY command. For bulk loading, the most useful COPY commands are:

- **COPY LOCAL:** Loads a data file or all specified files from a local client system to the HP Vertica host, where the server processes the files.
- **COPY with source data in an HP cluster:** Loads a data file or all specified files from different sources like JSON and CSV to HP Vertica internal format in an HP cluster.
- **COPY using User Defined Load (UDL) functions with custom sources, filters, or parsers:** Loads a data file or specified files from custom user defined source, parser, and filters by controlling data load settings.

All types of COPY statements share the same methodologies and processes, but they all have different limitations. Regardless of the differences, the COPY statement always has two phases:

- Phase I (initiator) loads and parses the file and distributes the file to other nodes.
- Phase II (executor) processes the data on all the nodes.

With COPY, you can use many Execution Engine operators for bulk loading. Some of the Execution Engine operators for loading one or more files are Load, Parse, Load Union, Segment, Sort/Merge, and DataTarget.

The COPY statement creates segments for each projection. *Segmentation* defines how the data is spread among the cluster nodes for query performance and fast data purges.

How the COPY Statement Loads Data

The COPY statement workflow for loading one or more files occurs in two phases:

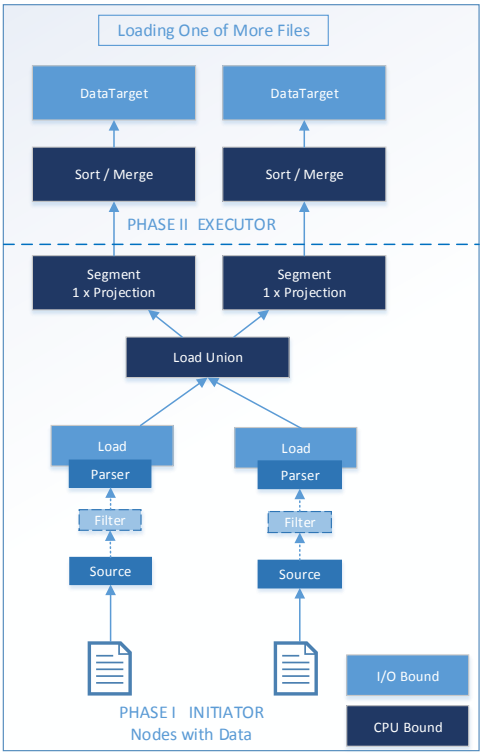
Phase I

1. The *Load* operator loads the source file with data into the database. The *Parse* operator parses the loaded data in the database.
2. The *Load Union* operator merges the parsed data into one container before segmenting the data. The operator is active when loading multiple files and is inactive when loading one file.
3. The *Segment* operator segments the parsed data into one or more projections depending on the size of the data. In addition, table partitioning segregates the data on each node to distribute the data evenly across multiple database nodes. Doing so ensures that all nodes participate in executing the query.

Phase II

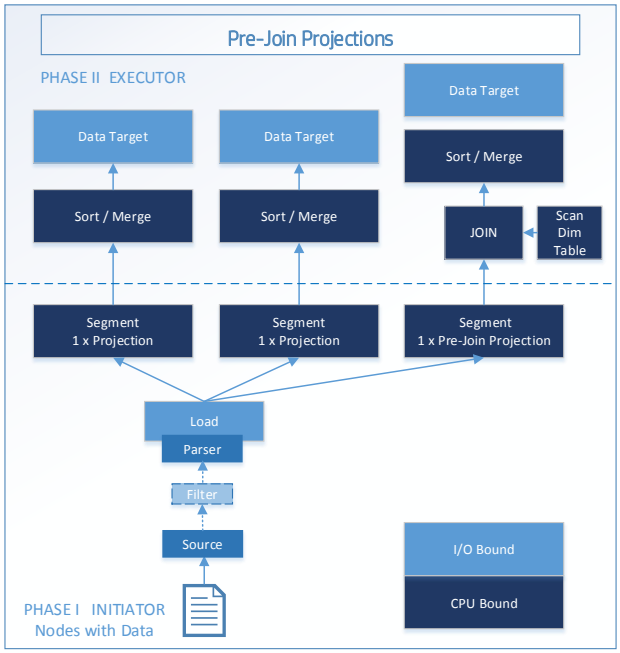
1. The *Sort* operator sorts the segmented data and projections. The *Merge* operator merges the sorted data appropriately. The Sort and Merge operators work on aggregated data.
2. The *DataTarget* operator copies the data on the disk.

The following figure shows the workload of loading one or more files in two phases. The light-blue and dark-blue boxes represent Execution Engine operators.

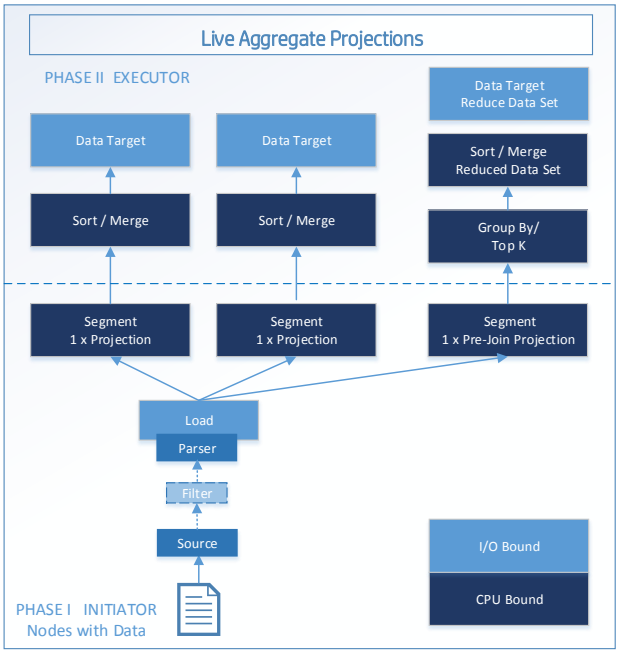


In HP Vertica 7.1.x with apportioned load, if all the nodes have access to the source data, Phase I occurs on several nodes. An apportioned load is a divisible load, such that you can load a single data file on more than one node. If the apportioned load is not available, Phase I occurs only on the nodes that read the file.

Phase II uses additional Execution Engine operators with pre-join projections and live aggregate projections. The following figure on the left (Pre-Join Projections) shows the additional Execution Engine operators, JOIN and SCAN, for a dimension table. The following figure on the right (Live Aggregate Projections) shows the additional GROUP BY/Top-K Execution Engine operator.



Pre-join projections add additional Execution Engine operators JOIN and SCAN for the dimension table.



Live aggregate projections add GROUP BY/Top-K Execution Engine operators.

Memory Requirements

The COPY statement uses multiple Execution Engine operators with minimum memory requirements.

Execution Engines operators of COPY Statement	Memory Requirements
SORT	4 GB memory per projection, with two 2 GB buffers • The first 2 GB is loaded with unsorted data and sorted data. • The second 2 GB is loaded to the temporary space.
MERGE	4 GB memory per projection, with eight 512 MB buffers
JOIN (pre-join projections)	2 GB memory per projection
SCAN (pre-join projections)	2 GB memory per projection
GROUP BY/Top-K (live aggregate projections)	2 GB memory per projection
Load Union	No minimum requirement

These examples show memory requirements for the COPY statement:

- Two projections require 8 GB memory to execute the COPY statement, 4 GB per projection.
- Pre-join projections require 6 GB memory to execute the COPY statement, 4 GB for the SORT operator and 2 GB for the JOIN operator.
- Live aggregate projections require 6 GB memory to execute the COPY statement, 4 GB for the SORT operator and 2 GB for the GROUP BY/Top-K operator.

Temp Space

The memory available in the system limits the number of COPY statements you can execute. You must have sufficient temp space to accommodate the size of the files you can load in one COPY command. To avoid file-size limitations from a lack of temp space, you can create temp space on another disk to improve performance. Temp space is disk space temporarily occupied by temporary files created by certain query execution operations, such as hash joins and sorts, in the case when they have to spill to disk. You might also encounter such operations during queries, recovery, refreshing projections, and so on. You can isolate Execution Engine temp files from data files by creating a separate storage location for temp space. If a temporary location is provided to the database, HP Vertica uses it as temp space.

Monitoring Resource Consumption

You must monitor resource consumption to adjust the number of load streams. The LOAD_STREAMS system table contains useful statistics about how many records get loaded and rejected from the previous load. HP Vertica maintains system table metrics to monitor load metrics for each load stream on each node. The resources to load data depends on many factors, and one size does not fit every use case. You must adjust the load according to the resource consumption, concurrency, and workload requirements.

Load Methods

Depending on the data you are loading, the COPY statement has several load methods. You can choose from three load methods:

- COPY AUTO
- COPY DIRECT
- COPY TRICKLE

When to Use COPY AUTO

COPY uses the AUTO method to load data into WOS. Use this default AUTO load method for smaller bulk loads. The AUTO option is most useful when you cannot determine the size of the file. Once the WOS is full, COPY continues loading directly to ROS containers on disk. ROS data is sorted and encoded.

When to Use COPY DIRECT

COPY uses the DIRECT method to load data directly into ROS containers. Use the DIRECT load method for large bulk loads (100 MB or more). The DIRECT method improves performance for large files by avoiding the WOS and loading data into ROS containers. Using DIRECT to load many smaller data sets results in many ROS containers, which have to be combined later.

When to Use COPY TRICKLE

COPY uses the TRICKLE method to load data directly into WOS. Use the TRICKLE load method to load data incrementally after you complete your initial bulk load. If the WOS becomes full, an error occurs and the entire data load is rolled back. Use this method only when you have a finely tuned load and moveout process at your site, and you are confident that the WOS can hold the data you are loading. This option is more efficient than AUTO when loading data into partitioned tables.

HP Vertica Load System Tables

HP Vertica provides system tables that allow you to monitor your database loads:

- LOAD_STREAMS: Monitors active and historical load metrics for load streams on each node and provides statistics about loaded and rejected records.
- DC_LOAD_EVENTS: Stores information about important system events during load parsing.
    - Batchbegin
    - Sourcebegin
    - Parsebegin
    - Parsedone
    - Sourcedone
    - Batchdone

Tuning the Data Load

Resource pool parameters and configuration parameters affect the performance of data load.

Resource Pool Parameters

The following parameters specify characteristics of resource pools that help the database administrator manage resources for loading data.

Parameter	Description
PLANNEDCONCURRENCY	Defines the amount of memory allocated per COPY command. Represents the preferred number of concurrently executing queries in the resource pool.
MAXCONCURRENCY	Limits the number of concurrent COPY jobs and represents the maximum number of concurrent execution slots available to the resource pool.
EXECUTIONPARALLELISM	Limits the number of threads used to process any single query issued in this resource pool and assigned to the load. HP Vertica sets this value based on the number of cores, available memory, and amount of data in the system. Unless memory is limited, or the amount of data is very small, HP Vertica sets this value to the number of cores on the node.

Query Budget

The `query_budget_kb` column in the RESOURCE_POOL_STATUS system table displays the target memory for queries executed on the associated pool.

To check the `query budget_kb`, use the following command:

```
=> SELECT pool_name, query_budget_kb FROM resource_pool_status;
```

Before you modify the `query budget_kb`, be aware of the following memory considerations:

- If MEMORYSIZE > 0 and MAXMEMORYSIZE is empty or equal to MEMORYSIZE, the query budget = MEMORYSIZE / Planned Concurrency
- If MEMORYSIZE = 0 and MAXMEMORYSIZE > 0, the query budget = (MAXMEMORYSIZE * 0.95) / Planned Concurrency
- If MEMORYSIZE = 0 and MAXMEMORYSIZE is empty, the query budget = [(General Pool * 0.95) – (sum(MEMORYSIZE of other pools)] / Planned Concurrency

How to Change Resource Pool Parameters

To change resource pool parameters, use the following command:

```
=> ALTER RESOURCE POOL <pool_name> <parameter> <new_value>;
```

Data Loading Configuration Parameters

The following configuration parameters can help you improve the performance of data load.

Parameter	Description
EnableCooperativeParse	Implements multi-threaded cooperative parsing capabilities on a node. You can use this parameter for both delimited and fixed-width loads. The cooperative parse parallelization is local to the node of the source data.
SortWorkerThreads	Controls the number of sort worker threads. When set to 0, it disables background threads. Improves the load performance when the bottleneck is at parse/sort phase of load.
ReuseDataConnections	Attempts to reuse TCP connections between query executions.
DataBufferDepth	Governs the buffers to allocate for data connections.
CompressNetworkData	Compresses the data traffic and reduces the data bandwidth.
EnableApportionLoad	Defines the apportionable source/parser for the load and splits the data into appropriate portions. In HP Vertica 7.1.x, the apportioned load works with `FilePortionSource` source function.
MultiLevelNetworkRoutingFactor	Defines the network routing for large clusters and adjusts the count reduction factor.

Troubleshooting Load Scenarios

Hewlett-Packard provides recommendations for different data loading scenarios. If you cannot resolve an issue when loading data into your database, contact HP Vertica Support.

Loading Large Files

Use Case: Large files require a long time to load.

Recommendation: Hewlett-Packard recommends that you make the load parallel on each node in one of two ways:

- Use the `EnableApportionLoad` parameter to make the work load parallel between different nodes in the cluster. For apportioned load, share the files between the nodes loading the file, and install the `FilePortionSource Source UDx` parameter using the following statement:

```
=> COPY copy_test.store_sales_fact with source
FilePortionSource(file='/data/test_copy/source_data5/Store_Sales_Fact.tbl',nodes='v
_vdb_node0001,v_vdb_node0002,v_vdb_node0003') direct;
```

- Split and stage the files in the NFS mount point so that all the nodes have access to the files on any node.

Both options have similar loading performance. However, the second option requires you to manually split the files.

Loading Multiple Small Files

Use Case: Using COPY DIRECT to load multiple small files degrades performance. Multiple statements with small files generate multiple ROS containers. A large number of ROS containers affects the performance of HP Vertica and requires additional work for the Tuple Mover after the load completes.

Recommendation: Hewlett-Packard recommends the following options for loading multiple small files with the COPY statement:

- Control the number of COPY statements to combine the loading files. Fewer COPY statements reduce the number of transactions and load more data in one transaction.
- Use Linux pipes to combine the loading files.

- Combine files in the same COPY statement to give better performance.

Loading Wide Tables

Use Case: Wide tables with large VARCHAR columns are bottlenecks of the workflow in Phase II of the COPY command.

Recommendation: Hewlett-Packard recommends the following options for loading wide tables:

- Change the `LoadMergeChunkSizeK` parameter as an exception for specific loads.
- Use flex tables for wide tables and for multiple small tables. Loading wide tables into flex tables requires loading one field instead of many fields. Thus, it reduces the size of the catalog and improves overall database performance. The initial load is very fast, with data available to the users quickly. However, query performance is lower in comparison to columnar storage.
- Using GROUPED correlated columns to load wide tables. GROUPED clause groups two or more columns into a single disk file. Two columns are correlated if the value of one column is related to the value of the other column.

You cannot resolve this issue by adding more resources, splitting the files, or parallelizing the work between the nodes. You should contact HP Vertica Support and adjust the configuration parameters under their guidance.

Executor Nodes for Load

Use Case: An executor node is reserved for computation purposes only and does not contain any data segments. Dedicated nodes use the total CPU. Two use cases relate to this issue:

- Large files must be loaded within 24 hours.
- The parse operation blocks the workflow.

In such use cases, the following conditions usually exist:

- Wide tables have high kilobytes per rows.
- The GZIP files are compressed to reduce the network transfer time. When the files are placed in HP Vertica local storage, the COPY command uncompresses the data increasing the CPU usage.
- The tables with wide segmentation keys use more CPU.

Recommendation: Hewlett-Packard recommends using executor nodes after consultation with HP Vertica Support. The resource pool configuration results in appropriate usage of resources on executor and data nodes. As resource pool parameters apply across the cluster, HP Vertica Support provides parameters for an efficient model. Instead of using executor nodes for loading, use servers with more CPU and assign exclusive CPUs to the load resource pool, thus limiting resources dedicated to load operations.