Dremio Software

Dremio Load Testing

Introduction

Dremio is a distributed platform designed to run on clusters of 1000+ nodes. The query performance on a Dremio cluster at any given time mostly depends on the following:

- Size of the Executor Engine (number of nodes, CPU, memory, SSD on each node)
- Number of Engines
- Amount of data scanned by the query
- Query Cost (Multi-table Joins, Aggregations, Ranking, Functions, etc. add to the complexity and cost of the query)
- Number of concurrent user queries and complexity of these queries

This whitepaper discusses concepts and techniques to scope the test, including:

- What to include in a load test?
- How do you perform a load test?
- Analyzing Query Execution results of a load test
- Monitoring the Cluster during a load test

Prerequisites

In a real-world scenario, we will have a variety of workloads executing concurrently and the performance of a query will vary depending on resources held by other queries running concurrently. Often, the need to consider concurrent queries lends to a perception that any kind of load testing is an insurmountable task, which is a myth. Dremio suggests framing the right questions at the start to plan what goes into the load tests.

- Gather SLA requirements for
  - Reporting Dashboard Queries
  - Ad-Hoc Data Scientist / Analyst User Queries
  - Reflection Refreshes
- Realistic number of concurrent users simultaneously executing different types of queries
- Set up monitoring to gather CPU, i/o, and memory statistics from all the coordinator and executor nodes in the cluster at 5-second intervals throughout the duration of the tests.

What to include in a Load Test

Dremio suggests a collection of a sub-set of queries representative of the expected workload. You can get queries from subject matter experts (SME) and/or look at query history within the Dremio instance. The query history is in the file queries.json on each Coordinator node.

For a meaningful test, this should include a broad mix of user queries based on low, medium, and high execution time, query cost, and memory footprint. We also want to limit the number of rows to return to the client since the purpose here is to load test the Dremio cluster, not necessarily the network traffic.

Dremio suggests you simulate the Reflection refresh activity during the load tests. This is all the more important if you have business queries and reflection refreshes happening simultaneously in the real world and if the user quotes and reflection refreshes are configured to use the same execution engine.

This information can be gleaned from queries.json or from Dremio UI

How to perform a Load Test

Baseline Performance - Single Query execution in isolation

Dremio advises that we execute all the collected queries, one after another. This is to document the baseline performance of the query execution time along with information such as query cost, queue name, engine name, and allocated memory.

Starting Queue Concurrency and Memory Limits

If you are a relatively new Dremio deployment, chances are you have the default settings for Workload Manager (Queues and Engine Routing). The default settings break the user queries into two queues (Low Cost and High-Cost User Queries) and the Reflections Queries into two queues (Low Cost and High-Cost Reflections).

For most workloads, there is a need to add at least one more cost-based queue such as a Very High-Cost User Query queue and a Very High-Cost Reflection queue with higher memory limits and lower concurrency limits to avoid resource cannibalization by a single queue. You could use the query cost, and memory utilization of individual queries in the baseline performance test as a guide to setting up these newer queues, alternatively, you could analyze the past queries run on the cluster to set up the new queues. Setting up the cost thresholds, and memory/concurrency limits for queues is discussed in another whitepaper.

Simulate Production Workloads

Next, we want to simulate the expected production workloads. With that in mind, the Load test harness should be able to submit multiple workloads simultaneously. For any meaningful test, we should be able to control the concurrency limits and memory limits of distinct workloads.

A Workload here means different types of queries, this is typically a Reflection workload and User Query Workload and within each of these workloads, it is common to have subsets based on Query Cost or subsets based on user groups such as Marketing, Finance HR, etc or subset base on function such as reporting, ad-hoc analysis etc. It's important to incorporate all or most workloads during the Load test.

Chances are an existing Customer has already identified distinct workloads and set up separate Queues and Engines for these with Dremio Workload Manager (Engine Routing and Queues), the best practices for these are discussed in other whitepapers.

In case Dremio deployments are set up with multiple engines, we will need to run load tests separately for each executor engine.

Analyzing Query Execution Results

Once you have run a Load test, it's obvious you want to analyze the results and compare them with different occurrences of the Load test, whereas each load test execution may be using a different configuration or with different concurrency and memory limits on the Engine.

Out Of Memory or Cluster Stability Errors

With Load tests with higher number of concurrent queries executing at the same time, you are likely to see OOM and stability errors such as {Nodes Lost Connection or Exceeded timeout (100000) due to faulty nodes}. Choices here could be expanding the engine (more executor nodes) or throttling the number of queries running concurrently by managing the concurrency or optimizing the workload.

Execution Time or Enqueued Time

In Load tests with a higher number of Concurrent Queries, you may see execution time for individual queries going up compared to baseline or expected SLA. You may notice queries are waiting in queue for longer times before they start execution. If the total response time on the queries does not satisfy the SLA, we may be at a point where we have to expand the cluster and perhaps add more nodes to the Executor Engines.

PS Tools - Dremio Load Tester and Dremio Query Analyzer

While you could use other Load testers that your team is familiar with, Dremio Professional Services has a Python and jmeter based tool, Dremio Load Tester(DLT, that can be used to simulate Production workloads, allowing you to simulate one or more workloads concurrently.

Similarly, Dremio Professional Services has a Python and SQL VDS-based tool, Dremio Query Analyzer(DQA), that leverages queries.json to observe the usage statistics, identify the most popular datasets, identify queries that have long execution and/or planning times, identify queries that allocate large amounts of memory.

Other white papers describe the best practices and Implementation details of these tools.

Monitoring

Dremio recommends gathering CPU, IO, and memory statistics for all coordinator and executor nodes in the cluster at 5-second intervals throughout the duration of every set of Load Tests. Dremio recommends the same post-production too, this allows us to analyze trends in these statistics over time and enables us to correlate spikes in these statistics with details of which queries were being executed at the same time the spikes occurred.

Best practices and implementation details for monitoring are described in other white papers.

Appendix

Sample Load tests you may want to simulate for Comparison and then use configuration that gives you the best times Query Execution times and no failures due to OOM or Cluster stability

Assumption, Dedicated Engine for User Queries, 3 Queues based on Query Costs (Low, High, Very-High).

Test Name	Num of Execs	vcpu and RAM per Executor	Dremio Heap , Direct memory	Concurrency Limits	Memory Limits per node
LoadTest1	10	16 vcpu 128 GB	12GB Heap, 122GB Direct	Preview - 50 Low - 50 High - 10 Very- High -2	N/A N/A High - 50GB Queue, 10Gb Job Very-High - 60GB Queue, 50Gb Job
LoadTest2	8	16 vcpu 128 GB	14GB Heap, 120GB Direct	Preview - 50 Low - 50 High - 10 Very- High -2	N/A N/A High - 50GB Queue, 10Gb Job Very-High - 60GB Queue, 50Gb Job
LoadTest3	6	16 vcpu 128 GB	12GB Heap, 122GB Direct	Preview - 30 Low - 30 High - 8 Very- High -1	N/A N/A High - 50GB Queue, 10Gb Job Very-High - 60GB Queue, 50Gb Job