Dremio Software

Rebalance Workload Management Rules

Introduction

Dremio services many varieties of query complexity. Observations on Dremio usage suggest that 80% of all queries are low or medium complexity, while the rest are complicated. Dremio uses query cost to evaluate this complexity and then directs the queries to appropriate queues based on query cost.

A newly installed Dremio defaults to using a query cost of 30000000 as a rule to govern routing queries to queues. So anything below this value goes to low cost queues, and anything higher goes to the high cost queue. That amount is somewhat arbitrary and should be evaluated against actual queries as soon as practical. Additionally, different environments and versions can affect the query cost. Therefore, Workload Management Rules should be re-evaluated after every upgrade, migration to new environments, and change to workload composition.

Down the line, Business users start using Dremio and execute various queries with different query costs. So, there comes a need to re-evaluate the distribution of query cost based on actual query data.

This document discusses the steps to

- Re-evaluate the threshold query cost based on usage
- Update the queues with the newly evaluated threshold query cost in an automated way.

Rebalance Workload

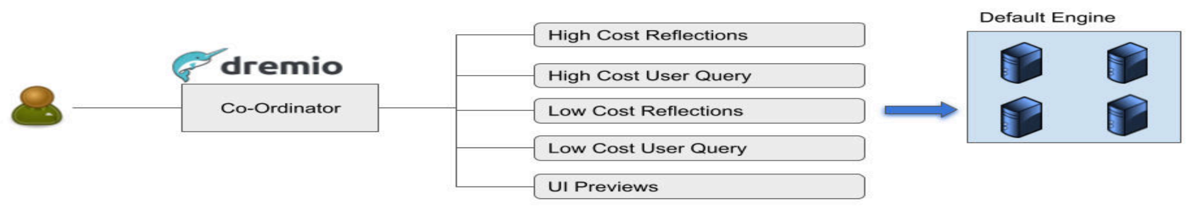
Dremio stores all time statistics and cost inside the "Queries.json" file for every query executed. The default location of "Queries.json" is inside the $DREMIO_HOME/log folder on each coordinator node. This file is automatically moved to the archive folder at the end of the day or once the file size crosses a threshold, whichever happens earlier. For optimal analysis, all historical queries.json files should be saved and archived.

For a 2-queue set-up, queries having queryCost below the 80th percentile should go to the low-cost user queries queue. The remaining ones go to the high-cost user queries queue. For a 3-queue set-up, base the distribution on the 50th and 80th percentiles, where queries having a queryCost below the 50th percentile go to the low-cost user queries queue, queries having a queryCost greater than the 50th percentile and below the 80th percentile go into the medium-cost user queries queue and queries having a queryCost greater than the 80th percentile go to the high-cost user queries queue.

The below sections discuss rebalancing the workload based on a single-engine or a multi-engine setup.

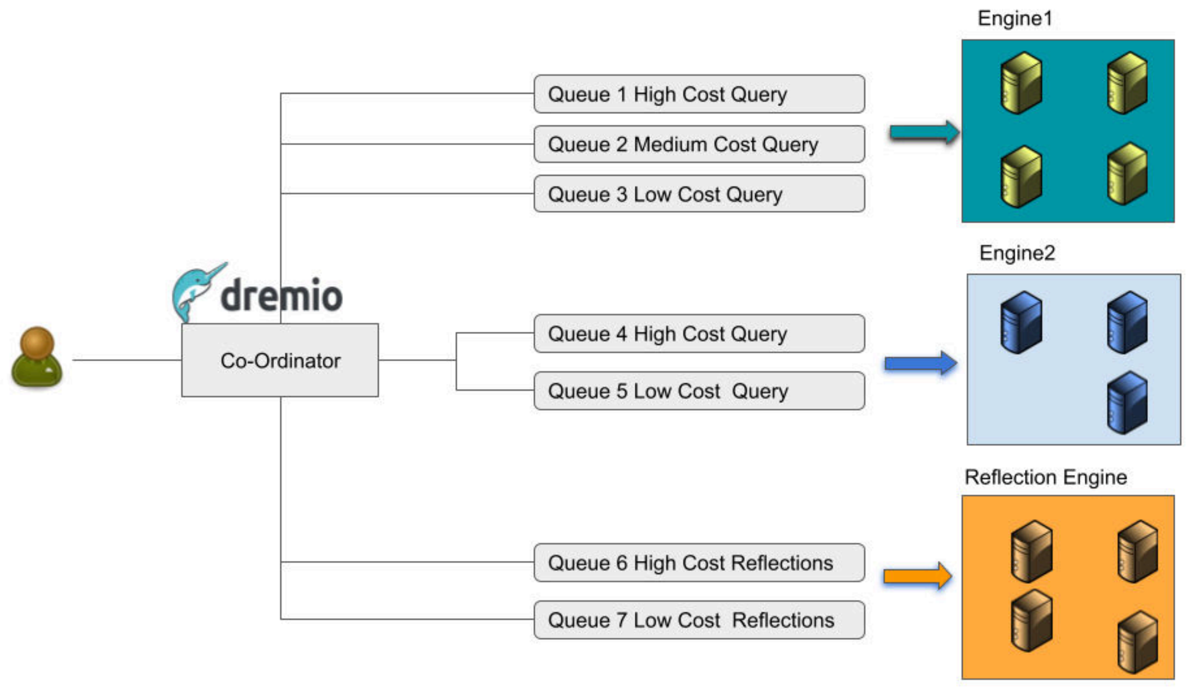
Single Engine

The diagram below shows the set-up of the default queues for a single engine installation.



- Copy "Queries.json" for at least 30 days from the Dremio server to a local folder
- Get the query cost below which 80% of the queries selected in the above step reside. Technically, it's 80th percentile
- Use the threshold value collected in the above step to tune your WLM queue rules. This can also be done programmatically by calling the API `PUT /api/v3/wlm/rule`. More details here: https://docs.dremio.com/current/reference/api/wlm/rule/#creating-or-updating-a-rule

Multiple Engines



In many cases, Dremio is configured to have multiple engines, with each engine having a different set of executors. A widespread use case is to have engines created to handle queries by User Groups. So all queries from the "Sales" user group go to Engine 1, and those of the "Production" user group go to Engine 2.

Each engine can be associated with a different set of queues. As in the example shown above, Queues 1,2 & 3 are associated with Engine1. While queues 4 & 5 are associated with Engine 2 and so on.

So, for the above configuration, sample queue rules at the initial stage, would be:

Query 1 High Cost Query	is_member('Sales') AND query_cost() > 30000000
Query 2 Medium Cost Query	is_member('Sales') AND query_cost() > 20000000 AND query_cost() <= 30000000
Query 3 Low Cost Query	is_member('Sales') AND query_cost() <= 20000000
Query 4 High Cost Query	is_member('Production'') AND query_cost() > 30000000
Query 5 Low Cost Query	is_member('Production') AND query_cost() <= 30000000
Query 7 High Cost Reflections	query_type() = 'Reflections' AND query_cost() > 30000000
Query 8 Low Cost Reflections	query_type() = 'Reflections' AND query_cost() <= 30000000

For this scenario, there is a need to tune the queues per engine. For example, to tune the queues for Engine 1, one must find all queries that went to Engine 1. Then, we need to find the 50th and 80th percentile query costs for those queries and update the queue rules for these queues accordingly.

Similarly, one needs to find the 80th percentile query cost for all queries executed in Engine 2. One must update the queue rules for Queues 4 & 5 using that percentile value.

The same approach to be taken to tune queues 6 & 7.

Automated Load Balancing

Taking Engine 1 as example, below steps can be used for automated tuning:

- Copy "Queries.json" for at least 30 days from the Dremio server to a local folder. The queue associated with a query is mentioned in this file. However, the engine name is not available in this data set

- To get the engine name associated with a queue, call the API GET /api/v3/wlm/queue
  More details on
  https://docs.dremio.com/current/reference/api/wlm/queue/#retrieving-all-queues
  Store all the queues associated with each engine from this data set

- Get all the queries that got executed on a particular engine by joining data gathered from the above 2 steps using the queue name as join condition. For our example, all queries that ran on Engine 1 will be selected for further processing.

- Select only those queries collected from the above step that meets the criteria below:
```
'requestType' belongs to ["RUN_SQL","EXECUTE_PREPARE"])  AND
'outcome' == "COMPLETED")                                AND
'queueName' != "UI Previews")                            AND
'queryCost' > 10
```

- Now get the 50th and 80th percentile query costs using the `NTILE` function against the `queryCost` column. Use these percentiles to update the queue rules using API `PUT /api/v3/wlm/rule`. More details here:
  https://docs.dremio.com/current/reference/api/wlm/rule/#creating-or-updating-a-rule

The Same approach should be followed for Engine 2 & 3 as well.

Conclusion

As users execute queries on the Dremio platform, the rules defined for queue routing may need to be updated. To enable Dremio to continue to efficiently handle the workload, the query cost threshold has to be evaluated at periodic intervals based on queries serviced and potentially updated as necessary. This document provides Dremio users with all the necessary steps to automate re-computation and update the queue rules based on usage patterns.