dremio

Dremio Software

Evaluating Coordinator Scaling

Introduction

This guide concentrates on the scale-out strategies in Dremio. It explains the significance of Dremio's architecture, especially the roles of the master and scale-out coordinators.

The document primarily focuses on the horizontal scaling (scale-out) of coordinators in Dremio, outlining the scenarios that necessitate the addition of scale-out coordinators to handle increased system demands. It discusses critical factors such as disk performance, garbage collection optimization, and memory management that influence the decision to scale out.

Practical guidance is provided for implementing scale-out coordinators in non-Kubernetes and Kubernetes environments, making this guide valuable for enhancing Dremio's performance and capacity through effective scaling strategies.

Monitoring Prerequisites

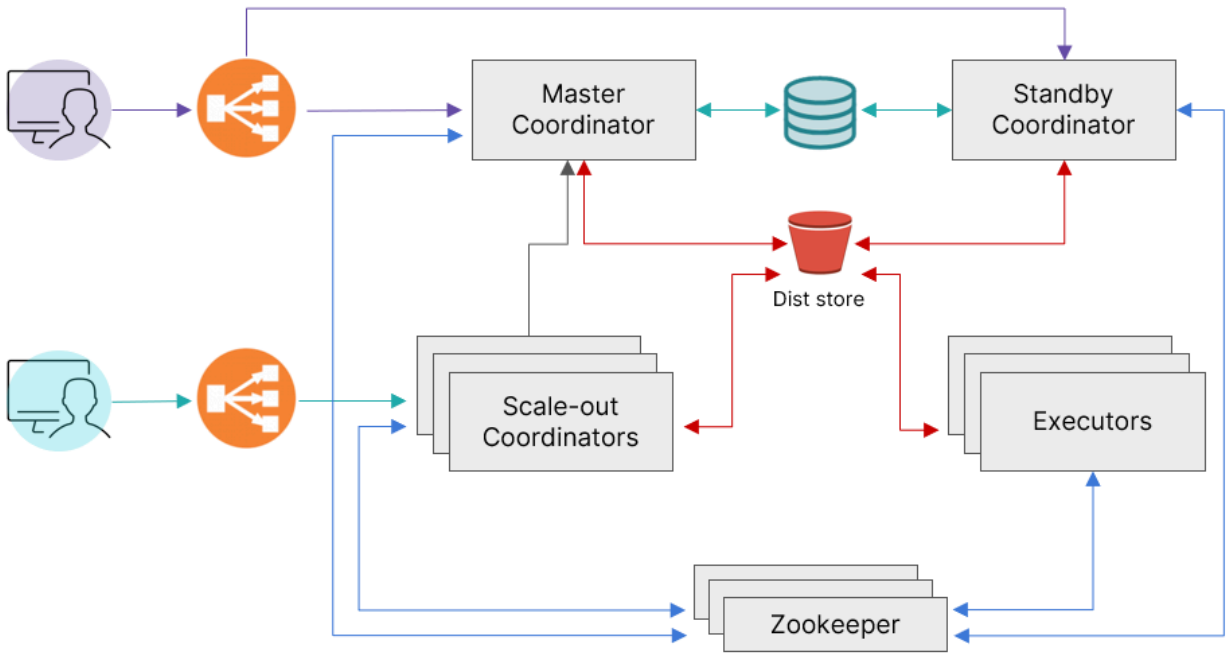
Monitoring is an essential part of understanding the system and capacity limits and includes:

- CPU utilization
- Memory utilization
- Network utilization and saturation
- Disk utilization and saturation
- Dremio Java heap utilization
- Dremio direct memory utilization
- Dremio's command pool queue size

There are many monitoring tools available. If you already have one, use the one you have to avoid additional efforts.
If you have no monitoring solution, Dremio recommends setting one up. For Kubernetes, you can follow this guide.

Dremio Components & Architecture

Dremio is an open data lakehouse that provides self-service analytics, data warehouse performance and functionality, and data lake flexibility across all your data. Below, you will see a typical architecture for Dremio.



The key components of this architecture are described below:

- **Master Coordinator:** The master coordinator in Dremio serves as a central component that manages and coordinates tasks and operations within the Dremio cluster. This crucial role involves overseeing cluster management, which includes the coordination of Dremio executors and nodes. It is also responsible for query parsing and planning, determining the most efficient execution strategy across the available resources. Additionally, the master coordinator handles resource allocation, ensuring efficient distribution of tasks among executors. Another critical function of the master coordinator is metadata management, encompassing information about data sources, datasets, and schema.
- **Scale-out Coordinator:** In Dremio, the scale-out coordinator is used explicitly for load balancing during the planning phase of query execution. Its primary responsibility is ensuring the planning workload is evenly distributed across the cluster. By balancing the planning tasks, the scale-out coordinator prevents any single node from becoming overloaded during this critical stage, thereby optimizing the overall efficiency and performance of the cluster.
- **Standby Coordinator (Failover Coordinator):** In Dremio, a standby coordinator is a component designed to enhance the resilience and high availability of the system. A failover coordinator's primary function is to ensure operations continuity in case the master coordinator encounters issues or fails. By having a standby coordinator, Dremio enhances its system's robustness, ensuring that critical functions like query planning, resource management, and cluster coordination continue uninterrupted, providing a more reliable and stable data analytics and processing environment.
- **Executor:** In Dremio, an executor is a critical component that performs the actual data processing for query execution. Executors handle tasks such as reading data, performing calculations, aggregations, and joins, enabling Dremio to manage large-scale data analytics. They operate within Dremio's distributed architecture, allowing for parallel processing of queries across multiple executors, significantly enhancing query performance, especially for large datasets. Additionally, executors contribute to the scalability of a Dremio cluster.

This document focuses on scale-out coordinators.

Horizontal and Vertical Scaling

What is Vertical Scaling?

Vertical scaling, also known as "scaling up," refers to increasing the capacity of a single server or system by adding more resources. This typically involves upgrading the physical hardware, such as adding more CPU cores, increasing memory (RAM), or expanding storage capacity.

Vertical scaling is commonly used to improve the performance of existing servers or systems when more processing power, memory, or storage is needed. It's straightforward because it doesn't involve adding more machines or instances. Instead, it focuses on making a single unit more powerful.

However, there are limits to vertical scaling. Physical servers can only accommodate a certain amount of hardware upgrades, and at some point, it becomes more cost-effective or technically feasible to consider horizontal scaling. Additionally, vertical scaling often requires downtime for installing new hardware, and it can represent a single point of failure since all resources are concentrated in one machine.

What is Horizontal Scaling?

Due to the limitations of vertical scaling, horizontal scaling may be required. Horizontal scaling, often referred to as "scaling out," is the process of increasing the capacity of a system by adding more nodes or instances rather than upgrading the existing ones. Scaling out for the Dremio coordinator involves adding more coordinator nodes to the Dremio cluster to distribute the workload and improve the performance and reliability of the coordinator.

When the demand exceeds the capacity of the current setup, more servers or instances are added to distribute the load. This can be done in data centers with physical servers or cloud computing environments where virtual instances can be added dynamically.

However, horizontal scaling also has its challenges. It requires a system architecture that supports the distribution of tasks across multiple nodes. This often involves load balancing and efficiently partitioning tasks or data. Additionally, managing a larger number of nodes can be more complex regarding maintenance and configuration.

When to Scale Vertically vs. When to Scale Horizontally?

A vertical scale-out should be done before considering a horizontal scale-out for the coordinator. On the Dremio master coordinator node, the CPU can be increased to 96 cores, and the memory can go up to 256 GB.

After reaching the capacity limits of the master coordinator, a horizontal scale-out can be considered. The master coordinator derives its name from being the single-access point to Dremio's central metadata KV store, which is located on a persistent disk. Therefore, before adding scale-outs, it needs to be ensured that the disk capacity in terms of throughput and IOPS is not reaching its limits. This can be verified using a monitoring solution (more details in "When a Scale-Out Coordinator does not help"). If the disk is already saturated and the main factor that slows down the query planning, a scale-out coordinator would not improve the performance significantly because the scale-out coordinators need to communicate to the master coordinator via the network, and the master coordinator provides the disk and stores all the metadata which is required for the planning. The scale-out coordinators do not store any metadata and state.
Furthermore, scale-out coordinators come with some caveats which should also be considered:

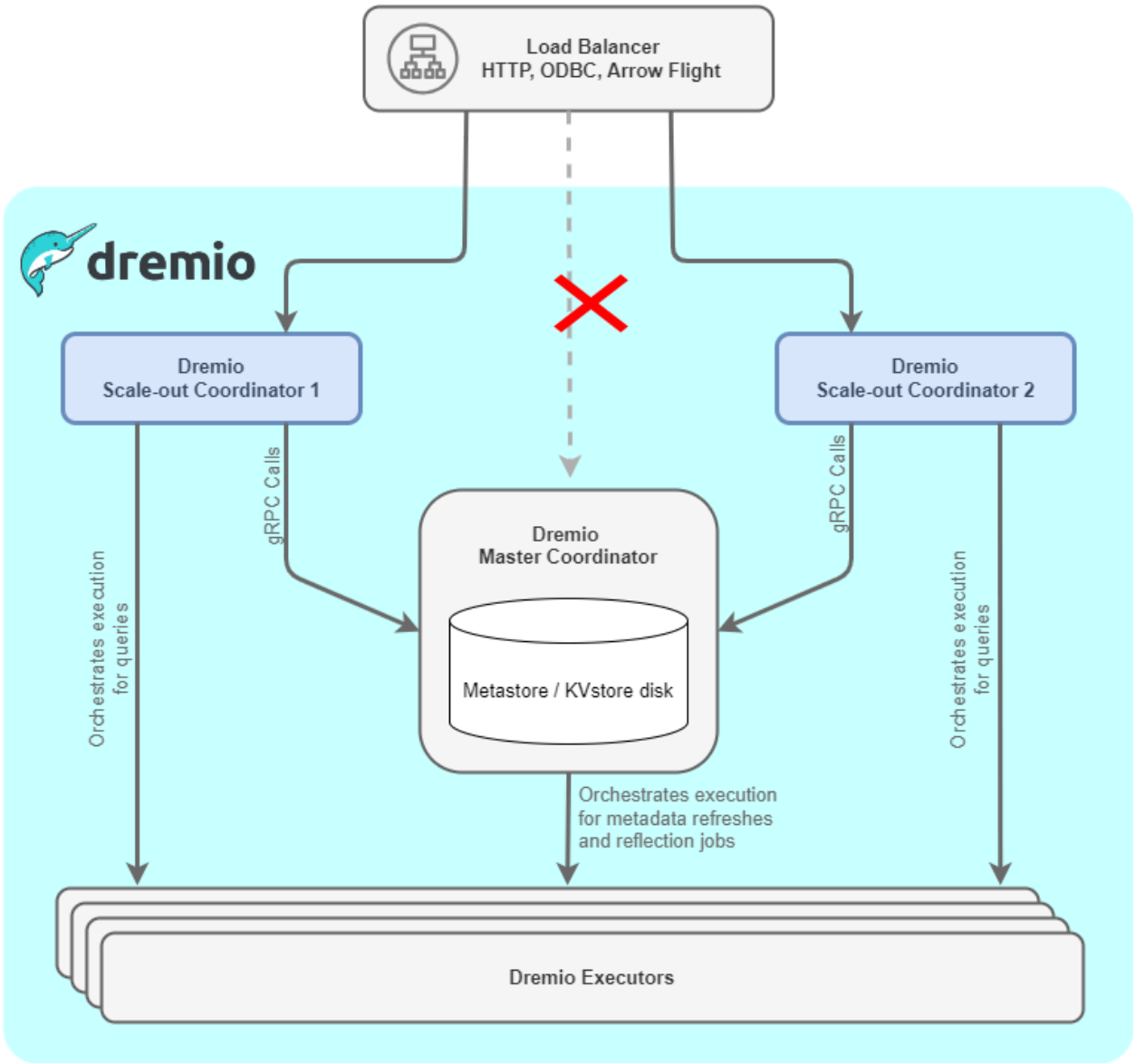
- There will be a significant increase in network traffic between the master and scale-out coordinators since all the metadata is stored on the master coordinator. This metadata must be sent to the scale-out coordinator to plan the queries.
- Since all the additional metadata is exchanged via network gRPC calls, there will also be a serialization and deserialization overhead, requiring additional CPU capacity.

This document's Coordinator Scale-Out Scenarios section covers more details about horizontally scaling.

Dremio Scale-Out Architecture

When considering scaling out, we assume that the master coordinator has already been scaled up to the limits described in the previous chapter. Dremio recommends starting with at least two scale-out coordinators; in the case where scaling out becomes necessary, then we should avoid any user traffic like HTTP or ODBC workloads hitting the master coordinator anymore. The scale-out coordinators become fully responsible for managing end-user and application connections. The master coordinator is responsible for serving requests from the scale-out coordinators, and it also does metadata refreshes and manages reflection refreshes.

Therefore, the minimum number of scale-out coordinators should be two to achieve at least doubling coordinator request capacity. The graphic below shows that the load balancer should only forward the traffic to the scale-out coordinators. It should not forward any traffic to the master coordinator. The disk of the master coordinator node needs to be highly performant. If the master coordinator disk gets saturated, the performance will drop across all scale-out coordinators. This architecture ensures the best performance when using scale-out coordinators.



Verify Before Adding Scale-Out Coordinators

A couple of things should be verified and optimized before considering a scale-out of the coordinators.

Slow Disks

We must ensure the master coordinator disk is performant enough to serve all requests. A scale-out coordinator with a slow disk will also not materially improve the query planning performance. Coordinator disk metrics that need to be monitored are:

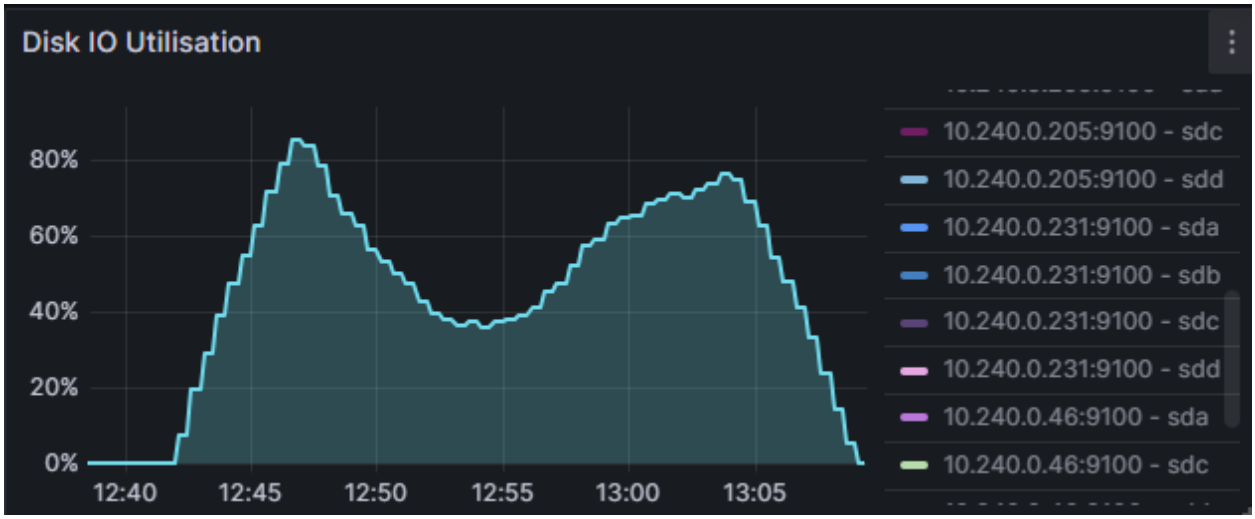
- Disk utilization: Increase the capacity if the disk utilization exceeds 70%.

- Disk saturation / Queue Depth: Disk queue depth refers to the number of input/output operations waiting to be processed by the disk at any given time. A value up to 4 is acceptable.
- I/O Wait: I/O wait is a metric representing the time a system's CPU is idle while waiting for input/output operations to complete, typically indicating a bottleneck in data access or disk performance. The desired value is close to zero, but improving the disk performance can be considered when the disk regularly hits a value above 20 to 40%.

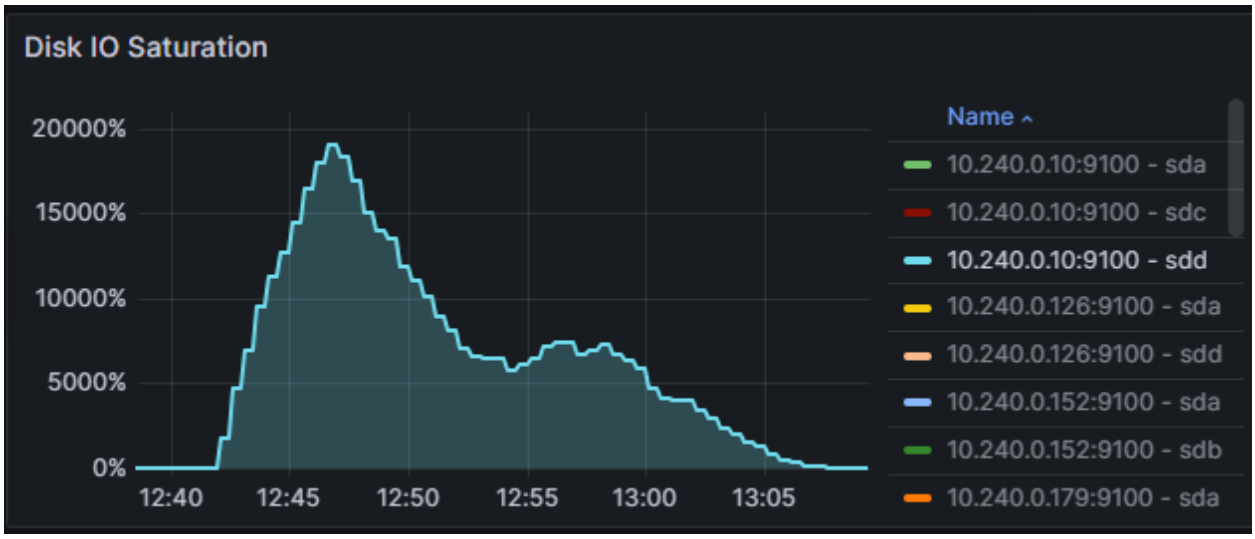
If you hit any of the abovementioned thresholds, we should add more IOPS and bandwidth to the provisioned disk. Often, all three issues appear together. For some cloud vendors, both the type of disk and its size determine the available amount of IOPS (e.g. for Azure “managed-premium” disks between 32 GiB and 1 TiB, IOPS scale linearly with the disk size).

Below are some examples of the Dremio Grafana dashboard (see Monitoring Prerequisites).

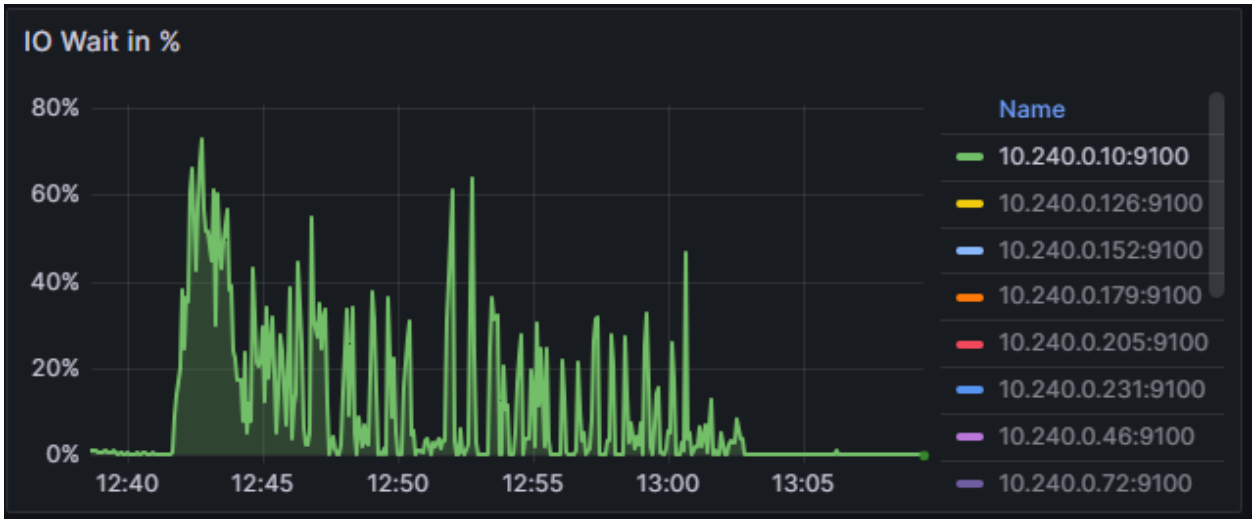
This is an example of high disk utilization above 70%:



The disk saturation is at 20000%, which represents a value of 200 and much higher than the expected maximum of 4:



Furthermore, we can see a lot of IO wait, higher than 20% to 40%. The desired goal is that all CPU capacity can be used for processing instead of needing to wait on the disk:



Optimize Garbage Collection Settings

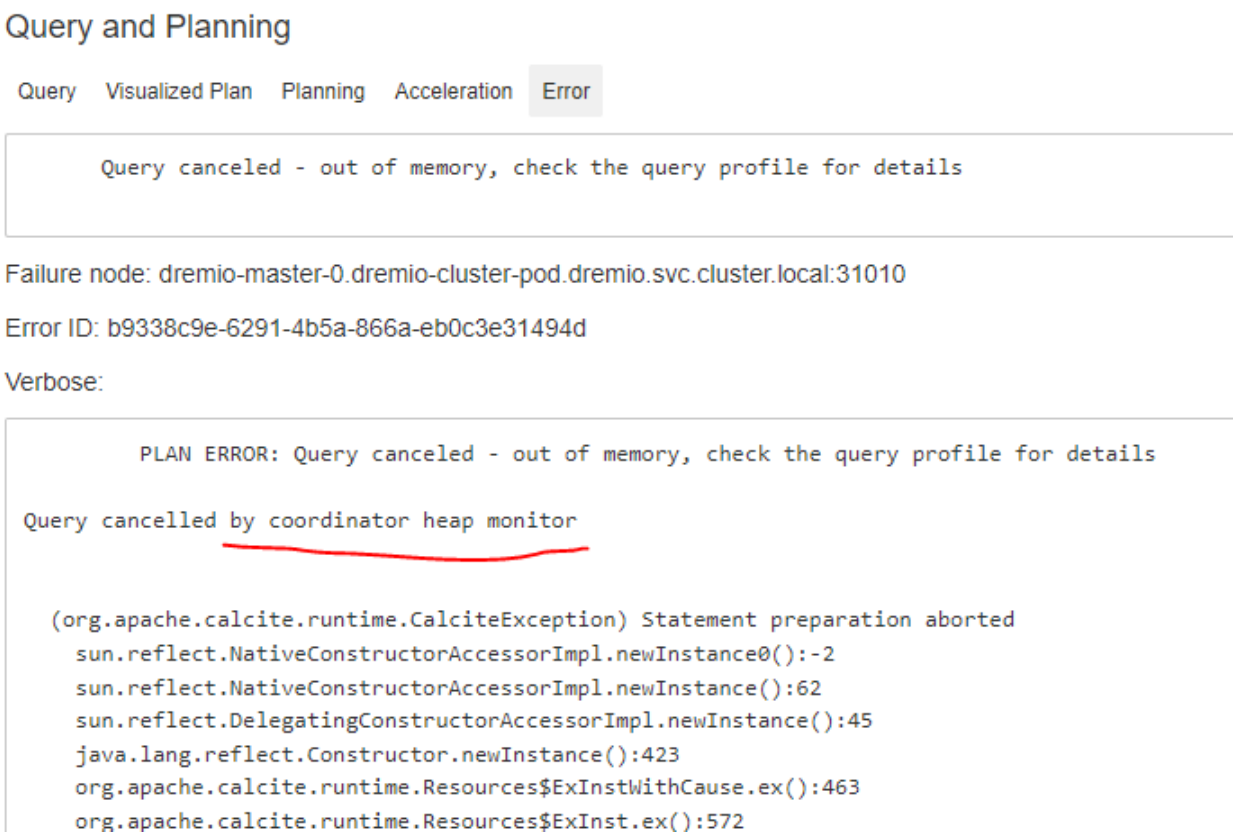
Ensure that you already implemented these optimal garbage collection settings.

An optimal garbage collection is essential to reduce garbage collection pauses and to avoid reaching the maximum heap capacity.

Sources of Out-Of-Memory Exceptions

If you plan to scale out because of seeing out-of-memory exceptions in your job profiles, please ensure that the errors occurred on the coordinator, typically during the query planning

stage. A scale-out does not help with out-of-memory errors that originated during query execution on the executor nodes.
This is an example of a job profile error when heap memory was too low on the coordinator node:



Coordinator Scale-Out Scenarios

The following scenarios describe when scaling out should be considered. Please carefully review the use cases and keep in mind that scale-out coordinators are not free; they always come with associated costs:

- Additional nodes and capacity are required
- Increased network utilization and latencies
- Serialization and deserialization overhead on the master and scale-out coordinators to exchange data

Increase the Capacity for Query Planning

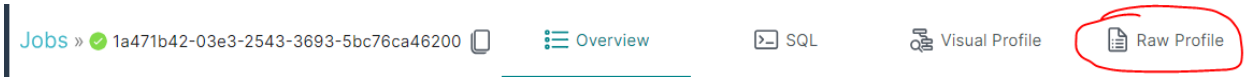
After scaling vertically to the maximum master coordinator node size (96 cores, 256 GB memory), it might be possible that you still hit the node capacity limits for the query planning. Symptoms that you hit the node capacity limits are:

- High planning times (regularly > 10 seconds or any planner timeouts can be observed in job profiles and queries.json logs)
- High command pool wait times for queries (> 1 second, can be observed in job profiles and queries.json logs)
- Command pool queue on the coordinator fills up (Prometheus metric)
- Query planner out-of-memory errors causing coordinator restarts or failures
- Long garbage collection pauses (> 1 second, can be observed in GC logs)
- Consistently high CPU utilization (Regularly > 70%, can be observed via monitoring)
- High Java heap memory utilization (Regularly > 70%, can be observed via monitoring)

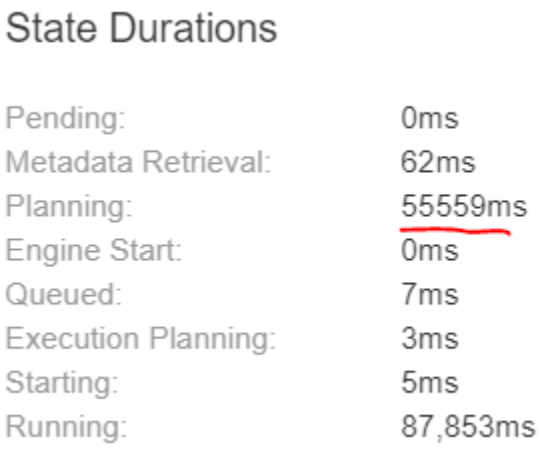
For more details, see the explanations below.

Increased Planning Times

Go to the jobs overview and select a jobs profile. Then click on “Raw Profile”:



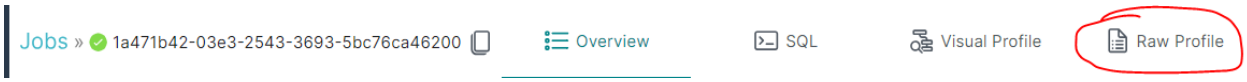
If you see regularly that “State Durations → Planning” takes many seconds or even times out, you should consider scaling out.



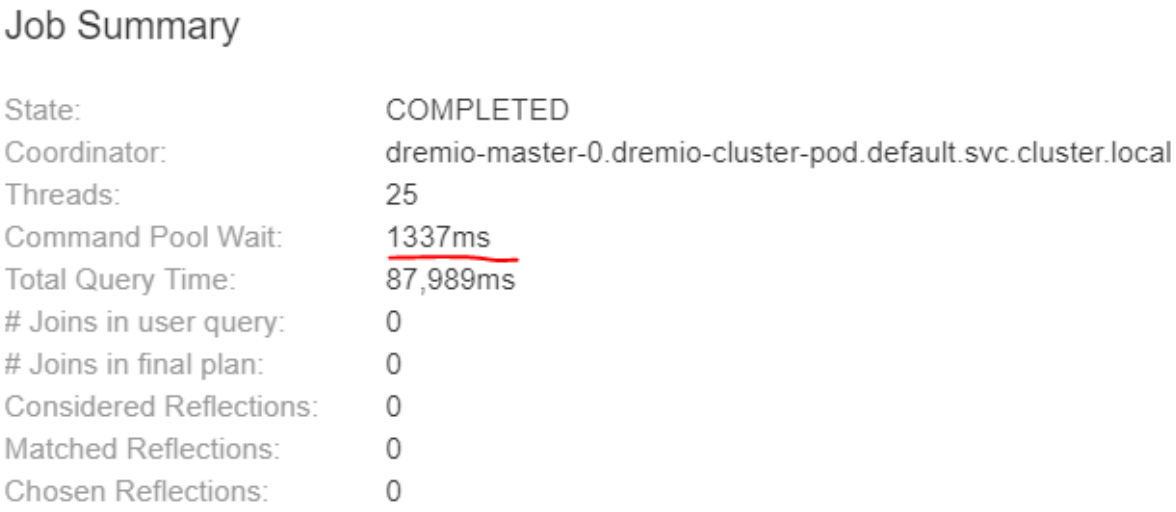
Alternatively, planning times for all queries can be viewed and analyzed in Dremio’s `queries.json` logs in the “`planningTime`” field. See here for further details.

Command Pool Wait Time for Queries increases

Go to the jobs overview and select a jobs profile. Then click on “Raw Profile”:



You should consider scaling out if you regularly see “Command Pool Wait” exceeding one second.



Alternatively, command pool wait times for all queries can also be viewed and analyzed in Dremio’s `queries.json` logs in the “`poolWaitTime`” field. See here for further details.

Command Pool Queue on the Coordinator Fills Up

Reviewing the Command Pool metrics requires a working monitoring solution consuming Dremio’s JMX or Prometheus metrics.
The screenshot below is from a Grafana dashboard (link in Monitoring Prerequisites). The desired value should always be close to zero:

Regular Out-Of-Memory Errors Causing Coordinator Restarts or Failures

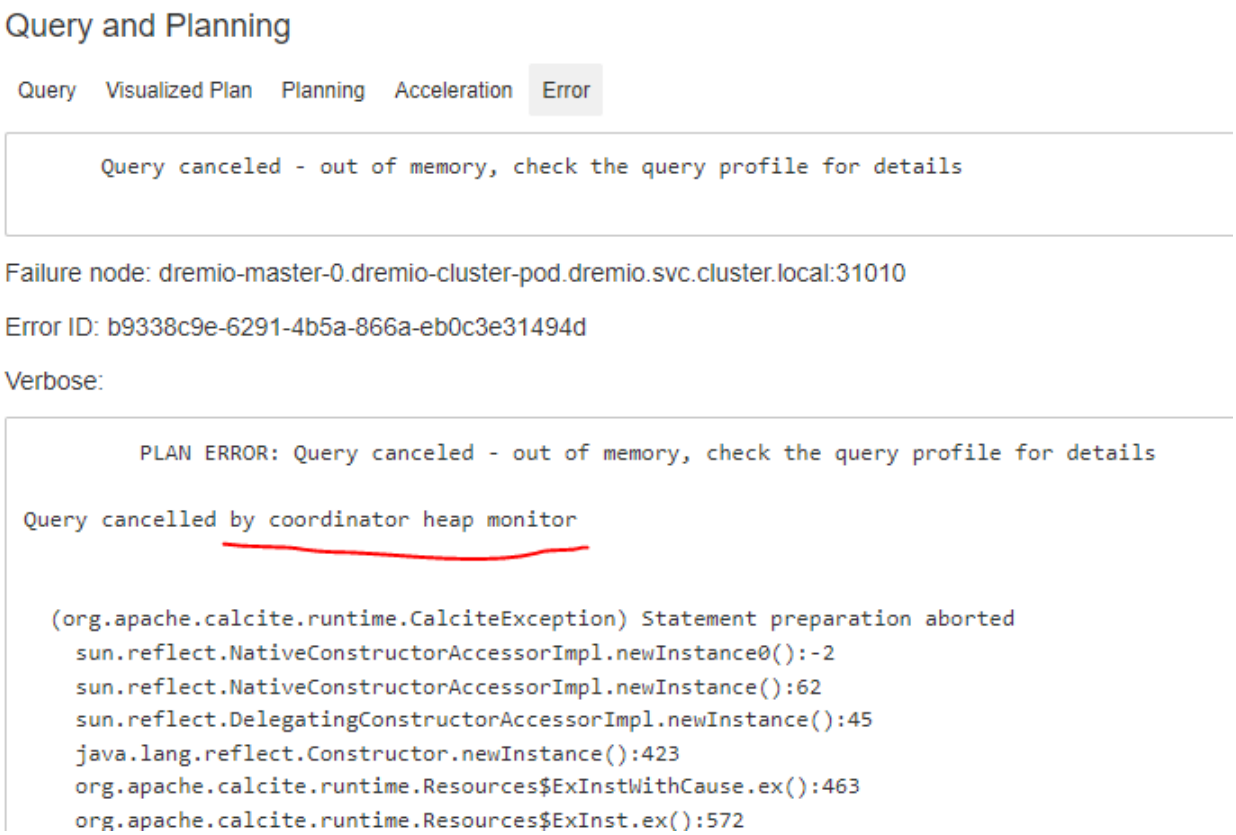
These errors require a review of the server.log files on the master coordinator node. If you see OutOfMemoryExceptions, the memory is likely too low, and its upper threshold has been exceeded:

```
Exception in thread "main" java.lang.OutOfMemoryError: unable to create native thread:
possibly out of memory or process/resource limits reached
  at java.base/java.lang.Thread.start0(Native Method)
  at java.base/java.lang.Thread.start(Thread.java:802)
  at memory.ThreadsLimits.main(ThreadsLimits.java:15)
```

Another example:

```
2023-09-25 09:15:59,732 [1aeeb031-7007-fd79-1755-172a9adae400/0:foreman-planning] ERROR
c.d.s.commandpool.CommandWrapper - command
1aeeb031-7007-fd79-1755-172a9adae400/0:foreman-planning failed
com.dremio.common.exceptions.UserException: Query canceled - out of memory, check the
query profile for details
        at
com.dremio.common.exceptions.UserException$Builder.build(UserException.java:885)
        at
com.dremio.exec.planner.ExceptionUtils.throwUserException(ExceptionUtils.java:53)
        at
com.dremio.exec.planner.DremioVolcanoPlanner.checkCancel(DremioVolcanoPlanner.java:162)
        at
com.dremio.exec.planner.cost.DremioRelMetadataCache.put(DremioRelMetadataCache.java:109
)
        at GeneratedMetadata_CollationHandler.collations(Unknown Source)
        at
org.apache.calcite.rel.metadata.RelMetadataQuery.collations(RelMetadataQuery.java:567)
```

When checking the query job profile for an error, you might see this error message:



Note: Please review the message carefully because out-of-memory errors can also occur on the executor nodes. In this case, a scale-out would not help.

Long Garbage Collection Pauses

It is required to set up garbage collection logging to validate long garbage collection pauses.

You can use the following command to validate the garbage collection pauses:

```
$ grep 'real=' /opt/dremio/data/log/gc*.log | grep -v 'real=0'
```

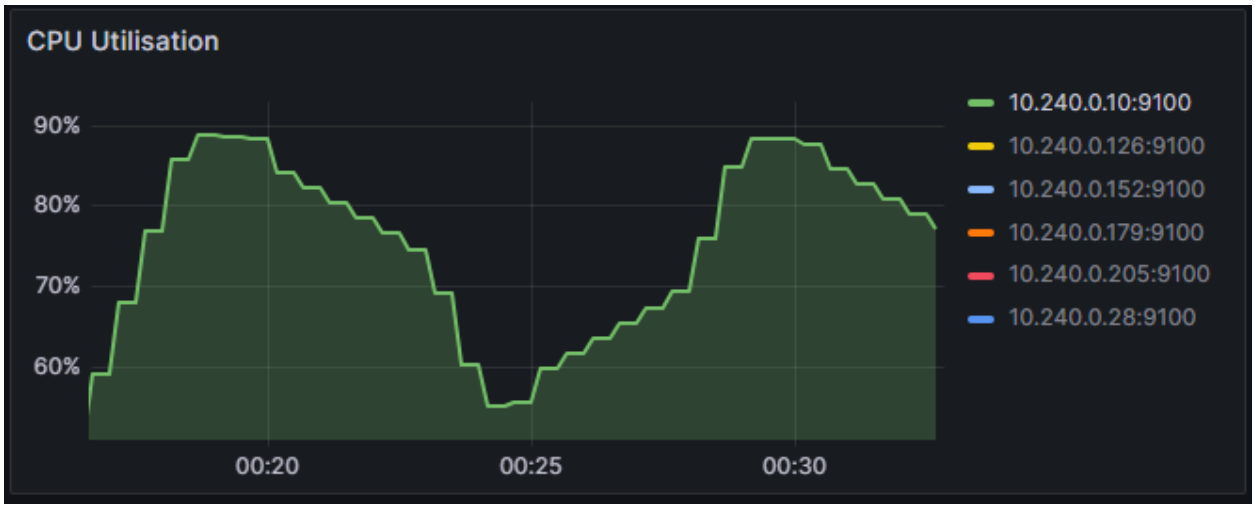
If you see large garbage collection pauses, it might indicate that the Java virtual machine hits its heap limits and needs to run full garbage collections and pause everything:

```
/opt/dremio/data/log/gc-2024-01-03_12-26-19.log: [Times: user=9.03 sys=0.04, real=8.46
secs]
/opt/dremio/data/log/gc-2024-01-03_12-26-19.log: [Times: user=1.41 sys=0.05, real=1.26
secs]
/opt/dremio/data/log/gc-2024-01-03_12-26-19.log: [Times: user=8.45 sys=0.05, real=8.06
secs]
```

```
/opt/dremio/data/log/gc-2024-01-03_12-26-19.log: [Times: user=1.46 sys=0.05, real=1.14
secs]
/opt/dremio/data/log/gc-2024-01-03_12-26-19.log: [Times: user=9.59 sys=0.04, real=8.96
secs]
```

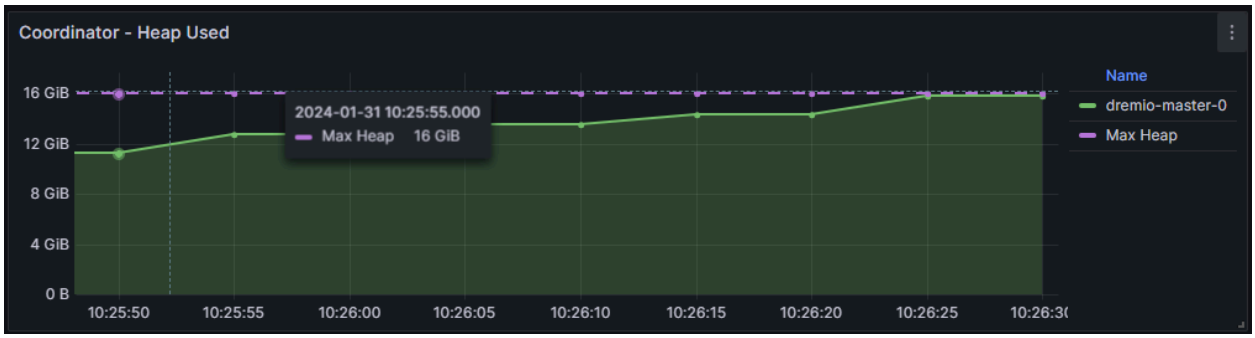
High CPU Utilization

Consider a scale-out if you regularly see that the CPU utilization exceeds 70% on the coordinator node. This means there is a permanent pressure on this node.



High Java Heap Memory Utilization

You should consider a scale-out coordinator if the heap often reaches the defined maximum (-Xmx option in Java) and you have already scaled vertically. A maximum utilized heap comes with long garbage collection pauses, unresponsiveness of the coordinator, and long planning times.

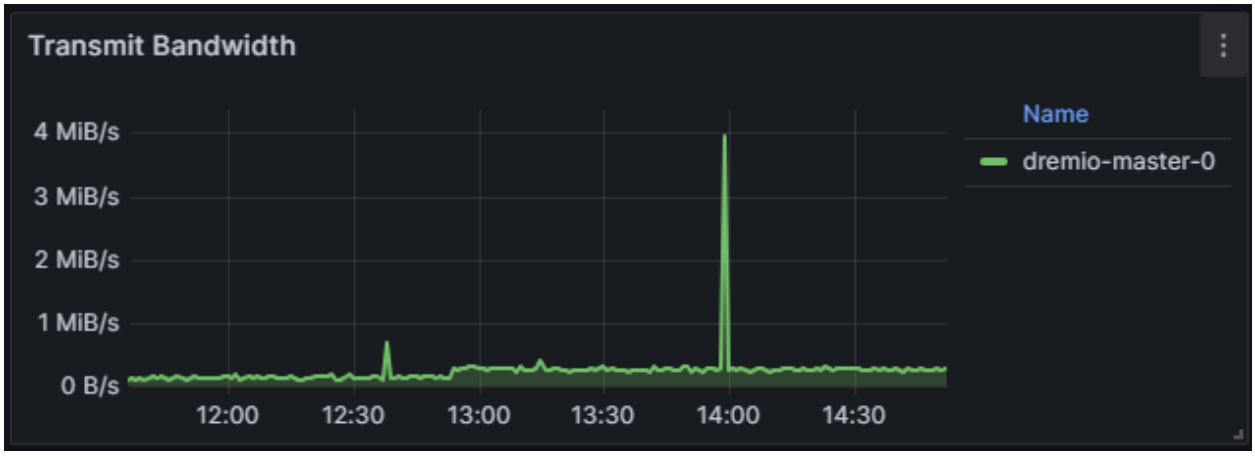


Increase the Network Capacity when having many ODBC/JDBC connections

If you have many ODBC and JDBC connections, e.g. more than 50% of the entire workload coming along with many parallel queries, consider adding a scale-out coordinator. Depending

on the result set size, the result delivery to the client can take a long time, and during this time the coordinator needs to keep the network connection open. The job also maintains many connections to the executors, and the executors send the results to the coordinator. Based on operating system configurations and limitations, there might be a limited number of network and file handles. Additionally, network interfaces have a limited bandwidth (e.g. 10 Gbit/s), which needs to handle the workload of receiving data from the executors and sending data to the querying client.

Below is an example of Prometheus monitoring of the network bandwidth. In this example, the network utilization is low and acceptable. Consider a scale-out when you hit 50% of the available network bandwidth.



Isolate the Planning Workload between the Coordinators to Isolate Load and Failure Domains

Sometimes, there are use cases with queries that always have expensive query planning due to the nature of the queries. This could be due to querying many tables and joining them. Additionally, highly complex queries could be used.

For such queries, the planning can take many seconds, along with high CPU utilization. This might be acceptable for some use cases, but others might have sub-second requirements. If the planning queue (command pool) fills up because of the long planning time, the SLAs for the other use case cannot be guaranteed any more. In this case, use a scale-out coordinator and route all traffic for the queries that result in expensive planning to a dedicated scale-out coordinator.

Scale-out Coordinator Implementation

Non-Kubernetes

In non-Kubernetes software installations, to enable a node to act as a scale-out coordinator set the below configuration in dremio.conf file.

```
services: {
    coordinator.enabled: true,
    coordinator.master.enabled: false,
    coordinator.web.enabled: false,
    executor.enabled: false
}
```

Kubernetes

The Dremio deployment in Kubernetes is usually performed using Dremio's official Helm Charts, which can be found here.

All customizations for deployment are defined in the values.yml, located here.

By default, there will only be a master coordinator deployed. The field "coordinator→count" defines the number of scale-out coordinators. The default of zero means that no scale-out coordinator will be deployed. A value of two would deploy two scale-out coordinators plus the master coordinator.

```
coordinator:
```

```
# CPU & Memory
# Memory allocated to each coordinator, expressed in MB.
# CPU allocated to each coordinator, expressed in CPU cores.
cpu: 70
memory: 229376

# This count is used for secondary coordinators only.
# The total number of coordinators will always be count + 1.
count: 0
```

Limitations

A few limitations exist when considering implementing scale-out coordinators in a Dremio cluster.

- Dremio recommends a maximum of five scale-out coordinators for a deployment
- Dremio deployments on AWS Edition and Azure ARM do not support secondary coordinator nodes.

Summary

In conclusion, this guide presents a thorough understanding of the scale-out process in Dremio, focusing on the pivotal roles of master and scale-out coordinators in the platform's architecture. It emphasizes the need for scale-out coordinators to handle increased system demands and improve performance.

The guide outlines key considerations like disk performance and memory management that influence scale-out decisions. It also provides practical steps for implementing scale-out coordinators in non-Kubernetes and Kubernetes environments, thereby offering a valuable resource for optimizing Dremio's scalability and efficiency.