Dune

Reducing query cost and query runtimes of Trino powered analytics platforms

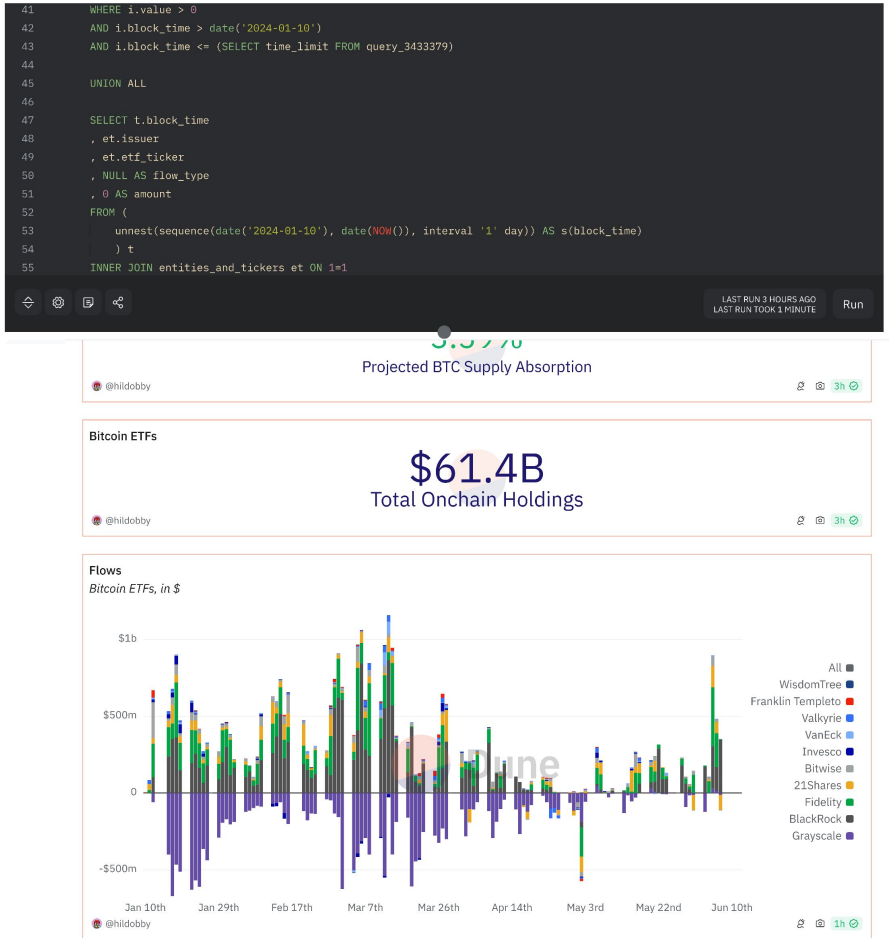
2024-06-13

Jonas Irgens Kylling, Dune Analytics

Dune

Dune Analytics

- Product
  - Data platform for blockchain data
  - SQL interface, web page and API
  - Free to use
  - Built on Trino fork and Delta Lake
- Problems
  - Keep cost under control
  - Multi-tenancy
  - Resource utilization



Reducing query cost and query runtimes of Trino powered analytics platforms

Agenda

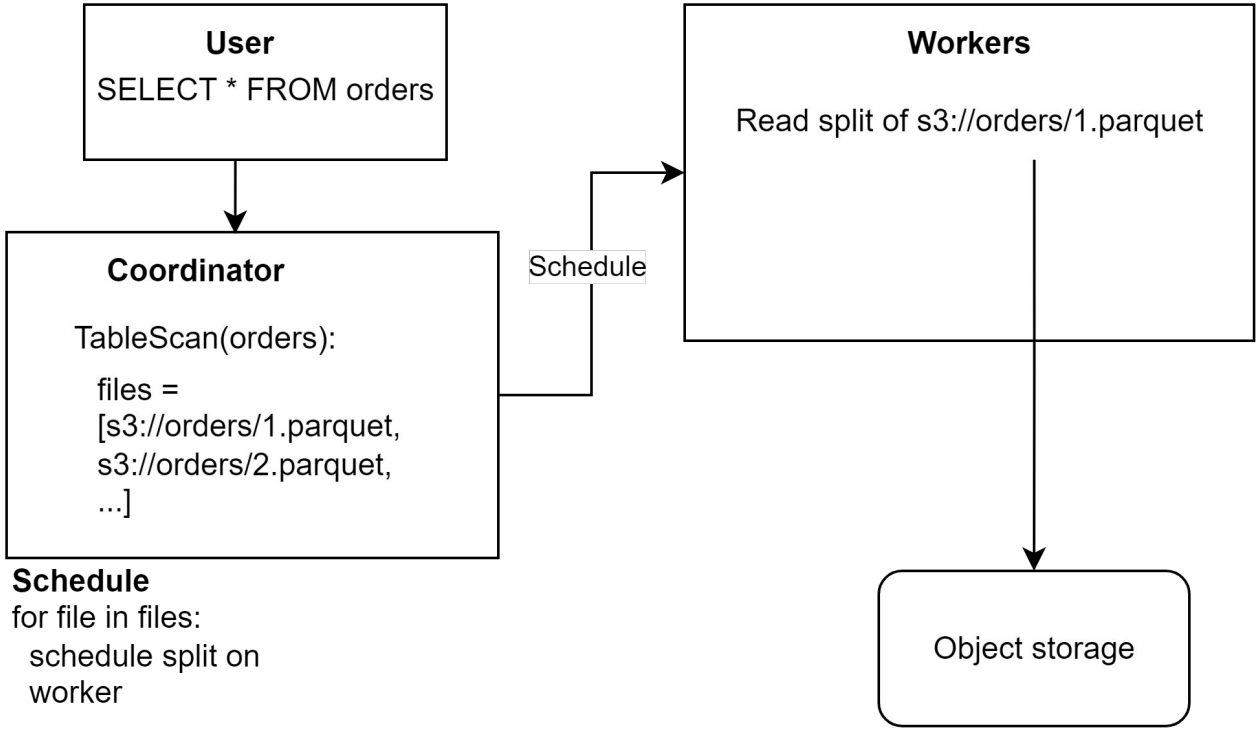
01 File system caching with Alluxio

02 Emulating multiple Trino clusters sizes

File system caching

File system caching

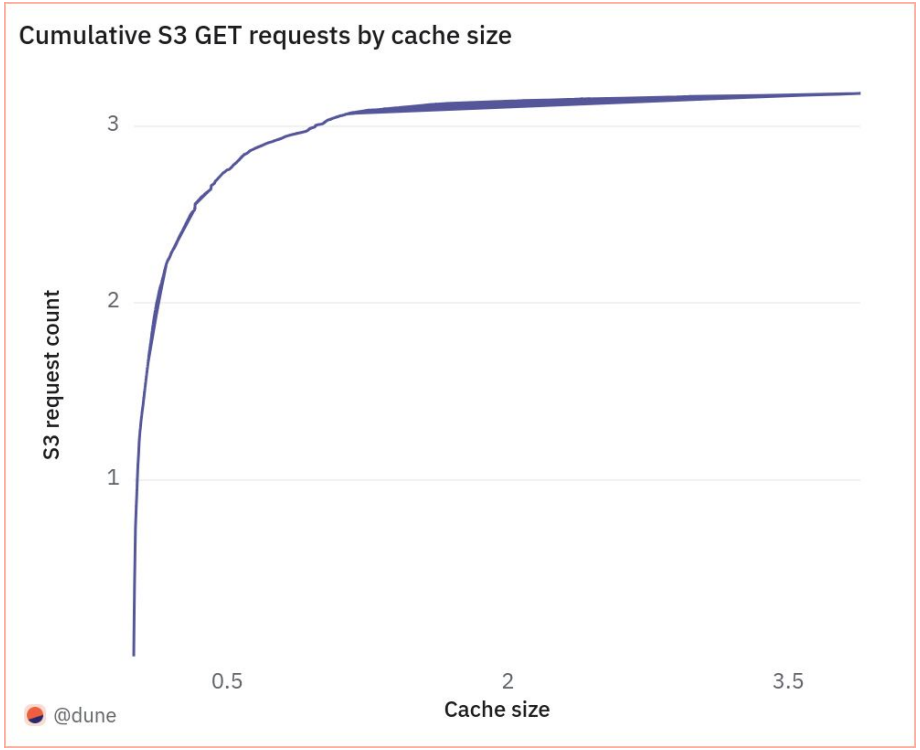
Life of a table scan



File system caching

Motivation

- Trino is very efficient at reading from object storage (`parquet_max_read_block_size = 16 MB`)
- Cloud object storage API operations are expensive
- ~2.6 cent per TB
- Most queries read the same small subset of the entire data lake



File system caching

Options for reducing S3 costs (Summer 2023)

- Out-of-cluster caching
  - Distributed Alluxio
  - MinIO

- Separate system
- Beware of cross AZ egress costs

File system caching

Options for reducing S3 costs (Summer 2023)

- Out-of-cluster caching
  - Distributed Alluxio
  - MinIO

- Separate system
- Beware of cross AZ egress costs

- Use cheaper cloud storage
  - Backblaze, Cloudflare

- Egress costs
- Requires modifying writers

File system caching

Options for reducing S3 costs (Summer 2023)

- Out-of-cluster caching
  - Distributed Alluxio
  - MinIO
- Use cheaper cloud storage
  - Backblaze, Cloudflare
- Intra-cluster caching
  - Rubix
  - Alluxio (Trino Fest talk 2023)

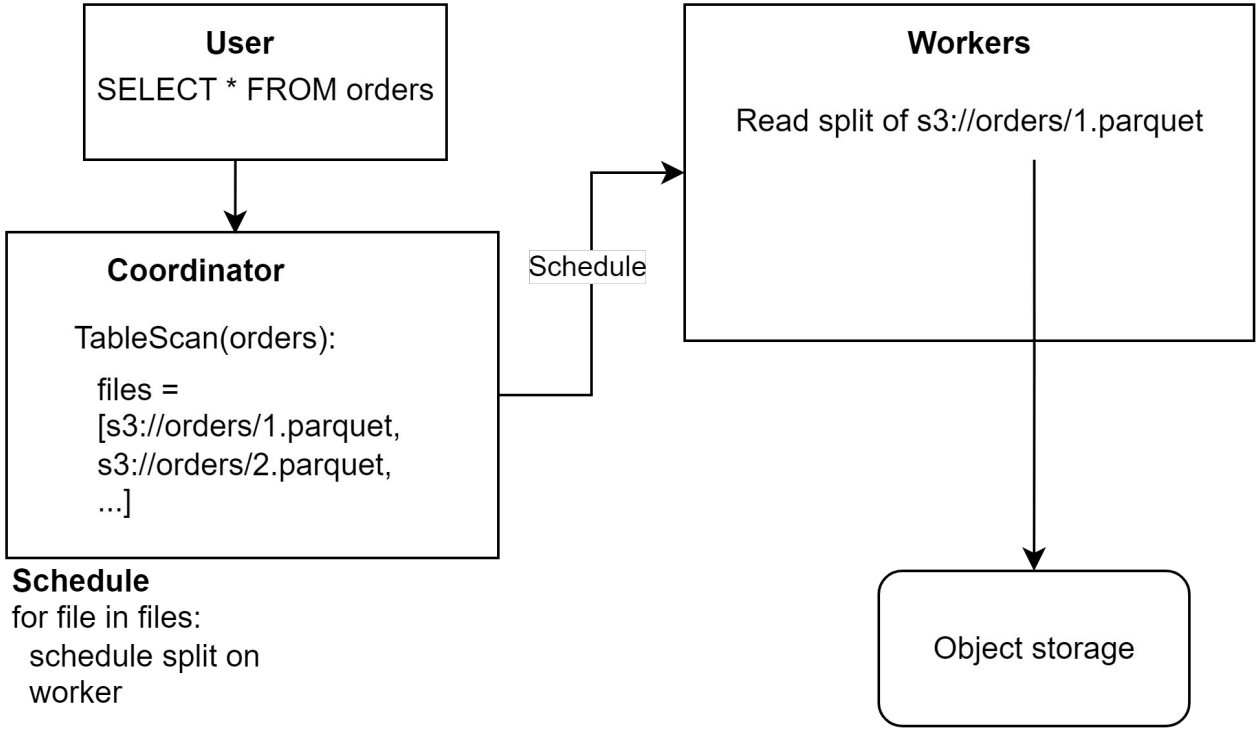
- Separate system
- Beware of cross AZ egress costs

- Egress costs
- Requires modifying writers

- Rubix is Hive only, unmaintained
- Need disks on all nodes
- Cache not shared between nodes
- Alluxio PR, maintained and well tested

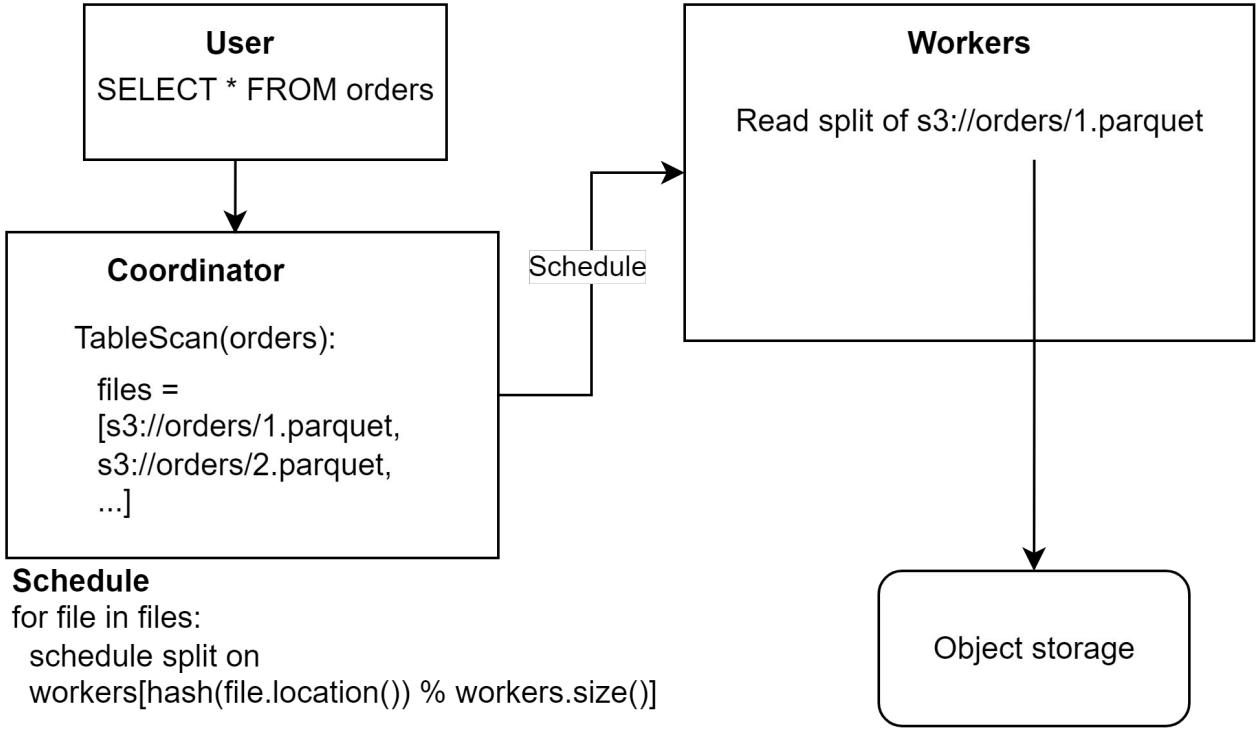
File system caching

Implementation



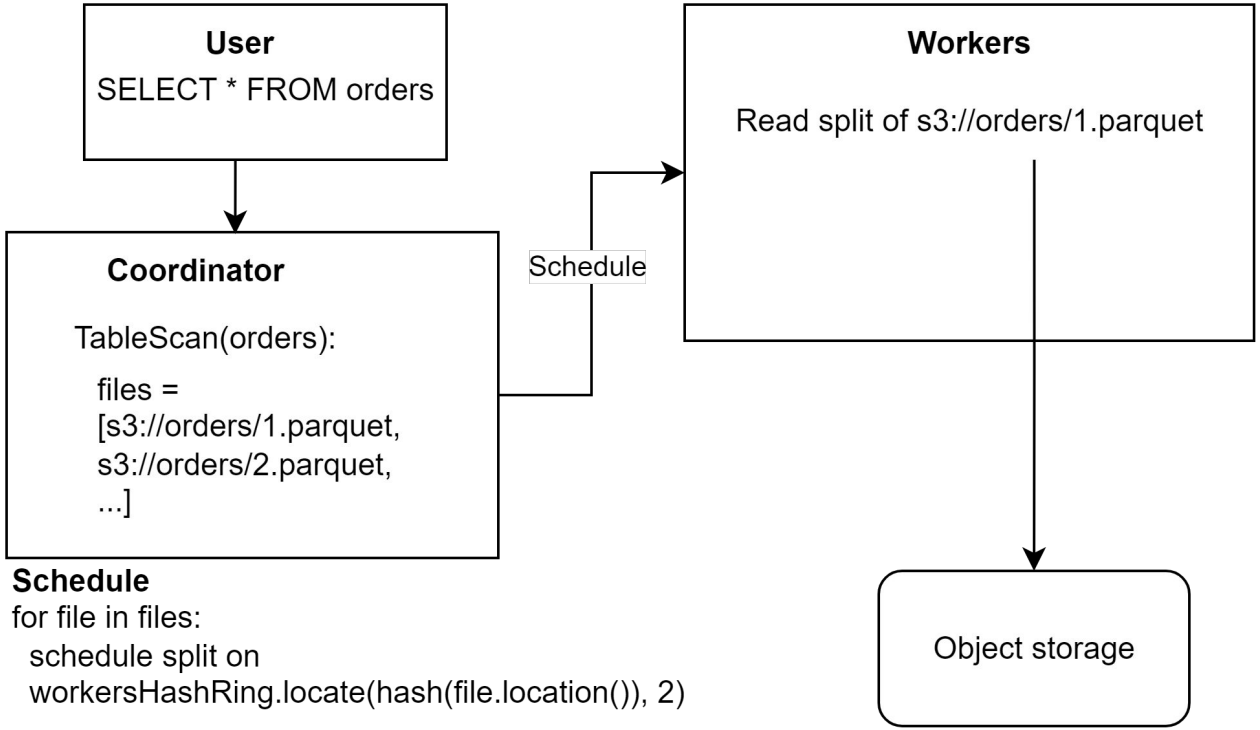
File system caching

Implementation



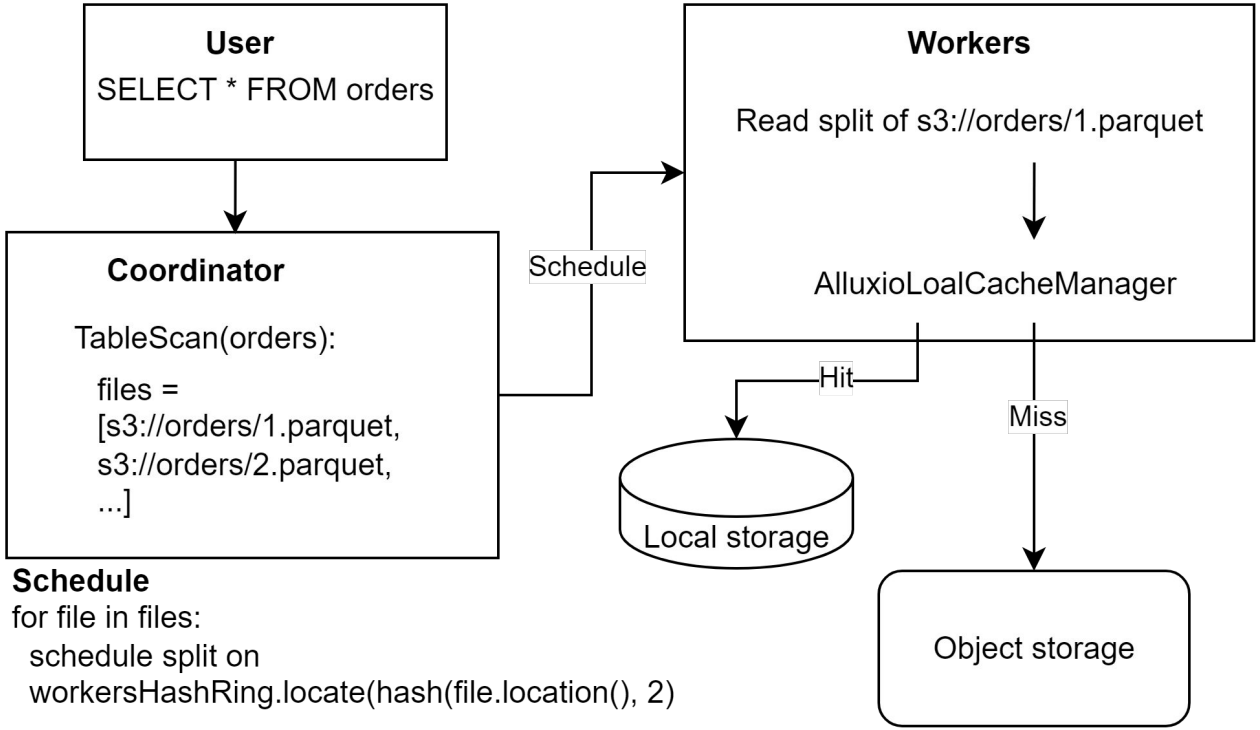
File system caching

Implementation



File system caching

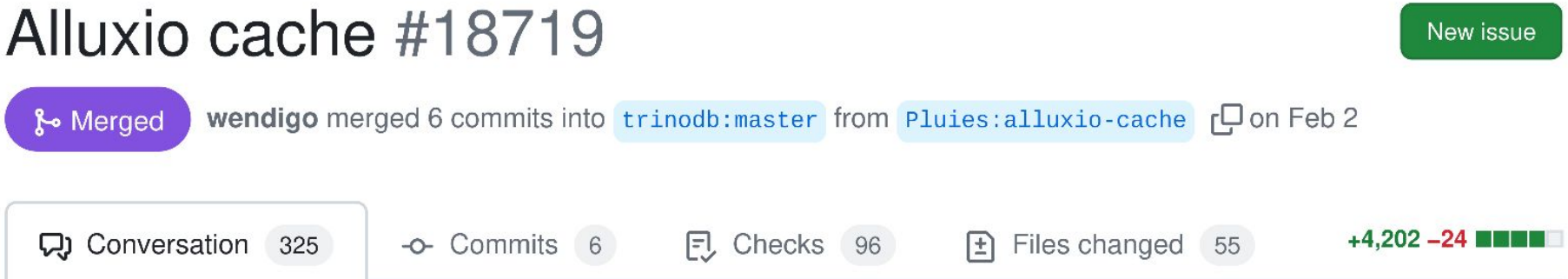
Implementation



File system caching

The journey

- Collaborative effort to go from prototype to merged PR and support in multiple connectors
- Available for Hive, Iceberg and Delta Lake connectors since (see #20550 for full credits)



File system caching

Usage

manifests.yaml

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: cache-volume
spec:
  resources:
    requests:
      storage: 2000Gi
  storageClassName: s3cache
```

```
apiVersion: v1
kind: Pod
metadata:
  name: trino-worker
spec:
  containers:
  - name: trino
    ...
    volumeMounts:
    - mountPath: /cache
      name: cache
  volumes:
  - name: cache
    persistentVolumeClaim:
      claimName: cache-volume
  ...
```

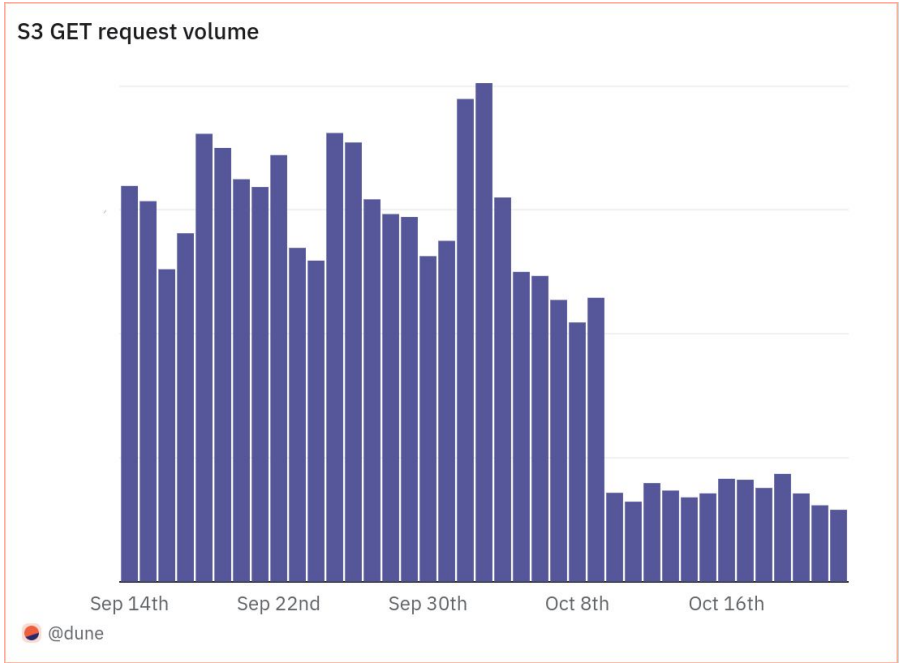
/etc/trino/catalog/delta_lake.properties

```
connector.name=delta_lake
fs.cache.enabled=true
fs.cache.directories=/cache/delta_lake
fs.cache.max-size=1900GB
# smaller than disk because of fs
overhead
```

File system caching

Results

- ~20% speed up of TPC query execution
- ~30% speeds up of analysis phase of TPC query execution for Iceberg tables
- ~70% reduction in S3 GET requests

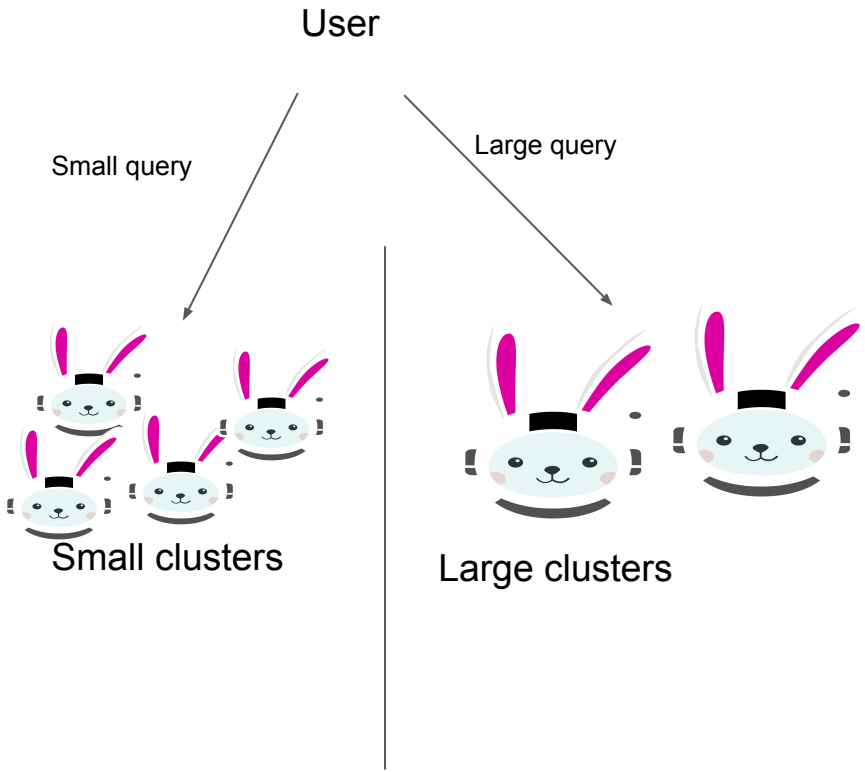


Emulating multiple Trino cluster sizes

Emulating multiple cluster sizes

Motivation

- We want to provide differentiated compute in multi-tenant Trino clusters
- Instant query execution start
- Low cost per query



Emulating multiple cluster sizes

Options

- Standby clusters
  - Idle compute

Emulating multiple cluster sizes

Options

- Standby clusters
  - Idle compute
- Resource groups
  - Only enforced at start of query execution
  - Does not affect running queries

Emulating multiple cluster sizes

Options

- Standby clusters
  - Idle compute
- Resource groups
  - Only enforced at start of query execution
  - Does not affect running queries
- Session properties
  - Hard limits
    - query_max_cpu_time, query_max_total_memory
  - Limits on some resources
    - task_concurrency

Emulating multiple cluster sizes

Options

- Standby clusters
  - Idle compute
- Resource groups
  - Only enforced at start of query execution
  - Does not affect running queries
- Session properties
  - Hard limits
    - query_max_cpu_time, query_max_total_memory
  - Limits on some resources
    - task_concurrency
- Alternative: Limit the number of nodes available to a query

Emulating multiple cluster sizes

Limiting the number of nodes

- NodeSelectorFactory
  - Creates NodeSelector, responsible for assigning splits to nodes
  - Which nodes are part of the cluster?
  - Which catalogs are available on each node (dynamic catalogs)?

```
private NodeMap createNodeMap(Session session,
Optional<CatalogHandle> catalogHandle)
{
    Set<InternalNode> nodes = catalogHandle
            .map(nodeManager::getActiveCatalogNodes)
            .orElseGet(() -> nodeManager.getNodes(ACTIVE));

    ...
}
```

Emulating multiple cluster sizes

Limiting the number of nodes

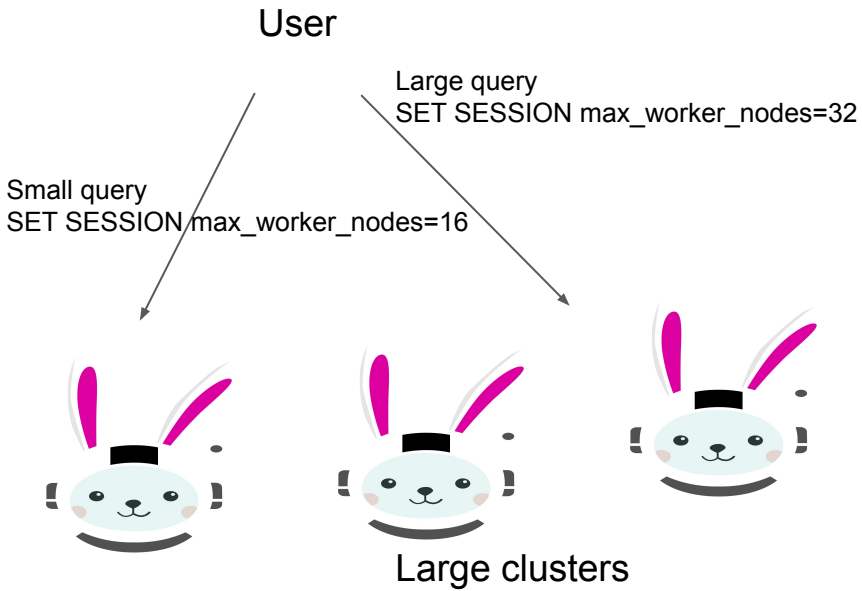
- NodeSelectorFactory
  - Creates NodeSelector, responsible for assigning splits to nodes
  - Which nodes are part of the cluster?
  - Which catalogs are available on each node (dynamic catalogs)?
  - How many workers can be used in this session?

```
private NodeMap createNodeMap(Session session,
Optional<CatalogHandle> catalogHandle)
{
  Set<InternalNode> nodes = catalogHandle
        .map(nodeManager::getActiveCatalogNodes)
        .orElseGet(() -> nodeManager.getNodes(ACTIVE));

  nodes = sample(nodes, getMaxNodesToUse(session));

  ...
}
```

Emulating multiple cluster sizes

Motivation

- ~20% reduction in average query cost
- Better margins on all execution types
- Better resource utilization
- Same query runtimes



Thank You!

✉ jonas@dune.com

dune.com

Questions?

✉ jonas@dune.com

dune.com