Starburst 性能白皮书二 - Presto 基于 Connecter 的性能提升

Starburst provides connectors to the most popular data sources included in many of these connectors are a number of exclusive enhancements. Many of Starburst's connectors when compared with open source Trino have enhanced extensions such as parallelism, pushdown and table statistics, that drastically improve the overall performance. Parallelism distributes query processing across workers, and uses many connections to the data source at the same time for increased overall performance. Pushdown and Starburst Cached Views are discussed in the section below.

Pushdown

Introduction

In addition to the performance improvement associated with the Cost-Based Optimizer, Trino can push down the processing of queries, or parts of queries, into the connected data source. This means that a specific predicate, aggregation function, or other operation, is passed through to the underlying database or storage system for processing.

The results of this pushdown can include the following benefits:

- Improved overall query performance
- Reduced network traffic between Trino and the data source
- Reduced load on the remote data source

Predicate and Column Projection Pushdown

In addition to the computation aspects discussed in the CBO section above, the amount of data that is brought back to the cluster can also be an important factor affecting performance. For example, I/O costs and network factors can mean that the transfer of data from source systems to the Starburst cluster can either be rate limiting or represent a significant chunk of time. As a consequence, there are several optimizations built in to ensure that only a minimal amount of data is brought back to the cluster for processing. After all, you don't want to transfer unnecessary amounts of data to the cluster just to filter it out after it gets here.

The majority of connectors will support some form of predicate pushdown to reduce the number of rows fetched from the source. Depending on the specific data source, connectors may leverage advanced capabilities, such as partition pruning, index scans, block-level min/max indices, hash-based bucketing, data clustering, to skip reading unnecessary data.

Additionally, only columns necessary for further query stages will be requested. That allows a connector to utilize inherent data source capabilities to fetch needed columns efficiently. For example, data may be organized in a columnar fashion at the source to facilitate fast access.

It is worth noting that the above mentioned pushdown optimizations can be applied very broadly when reading data from RDBMS, HDFS, Object Storage, and NoSQL systems. The specific features and implementation will naturally differ between the connectors and file formats. As an example, unpartitioned Hive tables with text files will have quite limited benefit from pushdown, while the same data organized in partitions and stored in a columnar file format such as ORC or Parquet, will benefit significantly.

As an example, we loaded the TPCH data set into a PostgreSQL database and then queried it using the PostgreSQL connector:

```
SELECT nationkey, lower(name)
FROM nation
WHERE regionkey = 1;
```

In this case, PostgreSQL will run the following query:

```
SELECT nationkey, name
FROM nation
WHERE regionkey = 1;
```

and then Starburst will apply the lower()function before returning the result to the client.

Aggregation Pushdown

Aggregation pushdown means that, when possible, aggregation operations are executed in source systems. That has the effect of reducing the amount of data transferred over the network to the cluster. Starburst's optimizer will detect when aggregation pushdown is possible and execute it on the fly for analysts.
Aggregation pushdown can take place provided the following conditions are satisfied:

- If aggregation pushdown is generally supported by the connector.
- If pushdown of the specific function or functions is supported by the connector.
- If the query structure allows pushdown to take place.

You can check if pushdown for a specific query is performed by looking at the EXPLAIN plan of the query. If an aggregate function is successfully pushed down to the connector, the explain plan does not show that Aggregate operator. The explain plan only shows the operations that are performed by Trino.

Let's consider another example in which the following SQL statement is executed in Starburst

against data in a PostgreSQL database:

```
SELECT regionkey, count(*)
FROM nation
GROUP BY regionkey;
```

In this case, PostgreSQL will run the following query:

```
SELECT regionkey, count(*)
FROM nation
GROUP BY regionkey;
```

Basically, the entire query was eligible for pushdown which can be confirmed by inspecting the output of EXPLAIN.

A number of factors can prevent a push down:

- Adding a condition to the query
- Using a different aggregate function that cannot be pushed down into the connector
- Using a connector without pushdown support for the specific function

As a result, the explain plan shows the Aggregate operation being performed by Trino. This is a clear sign that now pushdown to the remote data source is not performed, and instead Trino performs the aggregate processing.

Join Pushdown

Join pushdown provides similar performance benefits to the discussion above on aggregation pushdown in that join operations can be performed in source systems prior to transferring the data to the cluster. Join pushdown allows the connector to delegate the table join operation to the underlying data source. For selective join operations that has the impact of reducing the amount of data transferred over the network to the cluster. This can result in performance gains, and allows Trino to perform the remaining query processing on a smaller amount of data.
The specifics for the supported pushdown of table joins varies for each data source, and therefore for each connector.

Limited Pushdown

A LIMIT or FETCH FIRST clause reduces the number of returned records for a statement. Limit pushdown enables a connector to push processing of such queries of unsorted record to the underlying data source.

A pushdown of this clause can improve the performance of the query and significantly reduce the amount of data transferred from the data source to Trino.

Queries include sections such as LIMIT N or FETCH FIRST N ROWS.

Implementation and support is connector-specific since different data sources have varying capabilities.

Top-N Pushdown

The combination of a LIMIT or FETCH FIRST clause with an ORDER BY clause creates a small set of records to return out of a large sorted dataset. It relies on the order to determine which records need to be returned, and is therefore quite different to optimize compared to a Limit pushdown.

The pushdown for such a query is called a Top-N pushdown, since the operation is returning the top N rows. It enables a connector to push processing of such queries to the underlying data source, and therefore significantly reduces the amount of data transferred to and processed by Trino.

Queries include sections such as ORDER BY ... LIMIT N or ORDER BY ... FETCH FIRST N ROWS. Implementation and support is connector-specific since different data sources support different SQL syntax and processing.

Starburst Cached Views

Starburst Cached Views is a collection of performance features. Starburst Cached Views includes table scan redirection which allows read operations in a source catalog to be transparently replaced by reading the data from another catalog, the target catalog. The cache service which manages the configuration and synchronization of the source and target catalogs. And the cache service command line interface (CLI) which provides a terminal-based interface for listing and configuring redirections managed by the cache service.

Table Scan Redirection

For those use cases where the data needing to be access-optimized is in a relational or NoSQL database, Starburst has a platform feature that allows for some, or all, of the data to be mirrored into a faster, closer or even more cost-effective location. Table scan redirection enables Starburst to offload data from tables accessed in one catalog to equivalent tables accessed in another catalog. This can improve performance by shifting data access to a more performant system. It can also reduce load on a data source.

Once enabled, all queries will automatically be directed to the mirrored data location by the Starburst platform. This allows for a simple and efficient deployment that does not affect users or applications. Schedules can be set up to refresh all or incremental portions of the data as desired.

Redirection is transparent to the user, and therefore provides performance improvements without the need to modify queries.

A typical use case is the redirection from a catalog configuring a relational database to a catalog using the Hive connector to access a data lake. That catalog can also take advantage of Hive connector storage caching.

An additional use case is when queries need to access data spread over complex environments such as multi-cloud or hybrid on-prem/cloud. Under those circumstances performance can take a hit if you need to pull large amounts of data over the network. Table scan redirections allow you to materialize the data from remote locations in a center of data gravity. That access pattern allows us to get good performance when federating between geographically distinct locations.

Redirection of table scans is performed by Starburst after applying authentication and permission checks from the source catalog.

Most Starburst connectors support table scan redirection. A comprehensive list can be found in the documentation.

The target catalog can use an identical connector for maximum compatibility, or any other connector. Data types are translated based on the type mapping of the connector used for the source and target catalog. This type mapping can be customized to work around unsupported types by setting an explicit type mapping for the target catalog.
If table properties like partitioning, bucketing, sorting are used, then the target can only be Hive as other connectors don’t support these table properties.

Cache Service

Starburst’s cache service provides the ability to configure and automate the management of table scan redirections. The service connects to an existing Starburst installation to run queries for copying data from the source catalog to the target catalog. The target catalog is regularly synchronized with the source and used as a cache.

The cache service can be run as a standalone service or within the coordinator process. You can interact with it using its REST API, or the cache service CLI.