Distributed SQL Query Engine

Master-Level Semester Project

Amer Chamseddine and Artyom Stetsenko
{amer.chamseddine, artyom.stetsenko}@epfl.ch

Building and Running

Preparation

Before we can talk about executing queries, we must first generate, import, and partition the data. To be executed on the TPC-H database, the data must first be generated using *dbgen*.

Once generated, the `prep.sh` utility script from the `tools` directory can be used to prepare the PostgreSQL nodes, import the data, and partition it. The basic usage of the script is as follows:

```
./prep.sh [scale]
```

The `scale` argument is the scale factor and is needed to identify the local directory where the `.tbl` files generated by dbgen are stored, as well as to partition the data correctly. The script looks for these files inside the `./datasets/<scale>` directory. If this argument is omitted, the script uses the scale 0.1. The script is configured to load the data on the 8 different databases on the local machine (the setup we used for evaluation). These settings can be changed in the header of the `prep.sh` file.

In addition to the `.tbl` files, the script requires the following files in the same directory as itself:

- `default.pgpass`, containing the login details for logging into the PostgreSQL nodes without being prompted for password;
- `create_schema.sql`, used to create the TPC-H database schema;
- `create_schema_triggers.sql`, used to load the *partsupp* relation properly (*dbgen* does not ensure primary key uniqueness while generating this table, which causes PostgreSQL to complain);
- `create_aggs.sql`, used to load the intermediate and final flavors of the five typical aggregate functions: MIN, MAX, COUNT, SUM, and AVG;
- `helpers.sql`, used to create some helper SQL functions on the nodes; and
- `pg_bloom.sql`, used to create the functions required for bloom filtering.

To load the dataset `./datasets/0.01` onto all nodes, simply execute

```
./prep.sh 0.01
```

Build

The project can be built by running Ant from the root directory of our project. There is a `build.xml` file which makes it possible to build the project this way. The result of this is a file

named `data-distributed-db.jar`. To build the project, simply go to the root of our project (where the `build.xml` file is) and execute

```
ant
```

Execution

A configuration file is required by our program to run, which must be provided as an argument. It specifies the number of nodes where the database is partitioned, and provides JDBC connection strings, usernames, and passwords for accessing these nodes.

We provide a config file called `config.properties` that we used for evaluating the system.

To launch the command-line program for executing queries, simply execute:

```
java -cp data-distributed-db.jar
ch.epfl.data.distribdb.app.CommandLine config.properties
```