Cloud Engineer Assignment - Benchmarking

High level goal: To implement a command line tool that can be used to benchmark SELECT query performance across multiple workers/clients against a TimescaleDB instance. The tool should take as its input either a CSV-formatted file or standard input (the CSV format is specified below) and a flag to specify the number of concurrent workers. The tool should gracefully handle files that are larger than the one given, and should not wait to start processing queries until all input is consumed. After processing all the queries specified by the parameters in the CSV file, the tool should output a summary with the following stats:

- # of queries processed,
- total processing time across all queries,
- the minimum query time (for a single query),
- the median query time,
- the average query time,
- and the maximum query time.

Task 1: Set up a TimescaleDB instance with sample data

We cover how to set up TimescaleDB on a variety of platforms on our docs site: https://docs.timescale.com/latest/getting-started

Of course if you find anything missing or confusing in the instructions, please let us know!

You will be provided with two files to set up the database with sample data: cpu_usage.sql and cpu_usage.csv. Here are the files.

cpu_usage.sql will setup a hypertable with the following schema in a database named “homework”:

```
CREATE TABLE cpu_usage(ts TIMESTAMPTZ, host TEXT, usage DOUBLE PRECISION);
```

Records in this table will be instantaneous readings of CPU usage where ts specifies the time the measurement was recorded, host specifies the host who reported it, and usage is the value of the cpu at that time. To run cpu_usage.sql, run the following from the command line:

```
psql -U postgres < cpu_usage.sql
```

Now, to populate the table with data:

```
psql -U postgres -d homework -c "\COPY cpu_usage FROM cpu_usage.csv CSV
HEADER"
```

Your hypertable should now be ready to go!

Task 2: Create the query tool

For this task you will be provided with a CSV named query_params.csv. The file contains query specifications with the following form:

hostname,start_time,end_time host_000008,2017-01-01 08:59:22,2017-01-01 09:59:22 host_000001,2017-01-02 13:02:02,2017-01-02 14:02:02 host_000008,2017-01-02 18:50:28,2017-01-02 19:50:28 …

Your tool should take the CSV row values hostname, start time, end time and use them to generate a SQL query for each row that returns the max cpu usage and min cpu usage of the given hostname for every minute in the time range specified by the start time and end time. Each query should then be executed by one of the concurrent workers your tool creates, with the constraint that queries for the same hostname be executed by the same worker each time. Note that the constraint does not mean that the worker _only_ executes for that hostname (i.e., it can execute for multiple hostnames).

Your tool should measure the processing time of each query and output benchmark stats once all queries have been run. In particular, we are looking for the # of queries run, the total processing time across all queries, the minimum query time (for a single query), the median query time, the average query time, and the maximum query time.

Additional notes

Interaction with TimescaleDB team members is *encouraged* for discussing the design of your solution. Imagine that you are implementing a tool that will eventually be used by the team. The point of this assignment is, apart from assessing coding skills, to give us a better idea of how you would solve tasks within a team. So don't hesitate to ask questions if something is unclear, and if you believe that several different approaches to solving the problem are possible, you are *encouraged* to reach out to discuss your proposed approach before implementing it.

There is no set deadline for completing the assignment, but, obviously, delivering a solution after several weeks may not be judged as favorably. That said, we prefer a correct and clean solution over a quick and dirty one, and if other deadlines would conflict with this work, please discuss an expected timeline with us. After submission, we will review and discuss your solution with suggestions for improvements, akin to what our normal PR review process looks like.

Please make sure that the submitted assignment:

- Is written in a language that the team already employs (such as Golang, C, or Rust), as opposed to something like Java, Haskell, JS, or Python.
- Can be built and executed via Docker or Docker Compose. This should include database

setup, data migrations, compilation, and running the program. **Note: If we are unable to run the program according to the instructions provided in your README, we will not move forward with reviewing the assignment.**
- Handles any invalid input appropriately.
- Is submitted as a link to a git repository containing your solution (please remove after the interview process concludes).

Optional functionality:

- Provides additional benchmark statistics that you think are interesting (be prepared to explain why, don't just dump a bunch of numbers on the user).
- Handle CSV as either STDIN or via a flag with the filename.
- Unit / functional tests.

The assignment will be evaluated based on the following:

- Correctness
- Robustness (does the code work in all cases or does it fail for some?)
- Efficiency (i.e., algorithm/approach used, memory usage, etc.)
- Cleanliness (is the code easy to understand? Here, less really is more)
- Interaction with team members

Have fun, and we look forward to working with you!

Resources: https://github.com/timescale/timescaledb
https://docs.timescale.com/latest/getting-started