Proposal for PostgreSQL

DBT-5 Stored Procedure Development (2022)

Contact Detail

Name: Mahesh Varun Gouru
Email: mahesh.gouru@gmail.com
GitHub: https://github.com/MaheshGouru
LinkedIn: https://www.linkedin.com/in/maheshgouru/
Location: Delaware, United States | Tennesse, United States

Project Description

This project involves updating DBT-5, a fair use benchmarking kit based on the TPC Benchmark(TM) E. The server-side business logic was developed at a time before stored procedures and sub-transaction support was available for PostgreSQL.

The goal is to update the current server-side business logic, up to 12 transactions, to follow the benchmark specification to use stored procedures and sub-transaction as described by the specification.

Scripts need to be developed to:

1. Create a database (including schema for tables and indexes)
2. Load data to a database
3. Create transactional load with read and write queries (INSERTs, UPDATEs, and DELETEs)
4. Perform additional maintenance operations such as creating indexes and vacuuming

Motivation

As a current graduate engineering student at Vanderbilt University in Nashville, TN (USA). I am pursuing my MS in Computer Science and I am interested in the project DBT-5 Stored Procedure Development (2022). I have previous experience with evaluating various message buffers and data stores in 2 technical papers respectively for the Institute for Software Integrated Systems.

Data is an integral part of large distributed systems for which scalable storage, reliability, consistency, and fast retrieval are important desired features. The existing solution space of modern datastores is quite broad and demonstrates significant heterogeneity in data layout, SQL

vs NoSQL architectures, replication and consistency solutions, performance traits for different workloads, configuration space, and deployment environment, among many other criteria. This heterogeneity makes it substantially challenging for system designers to decide on the most appropriate solution to adopt for their use cases. To address this significant but mostly unresolved problem, this proposal presents insights gained from a comprehensive evaluation of a broad range of representative and contemporary datastores.

As TPC-E is more complex than previous OLTP benchmarks such as TPC-C because of its diverse transaction types, more complex database and overall execution structure. TPC-E involves a mix of twelve concurrent transactions of different types and complexity, either executed on-line or triggered by price or time criteria. The database is comprised of thirty-three tables with a wide range of columns, cardinality, and scaling properties. TPC-E is measured in transactions per second (tpsE). While the benchmark portrays the activity of a stock brokerage firm, TPC-E is not limited to the activity of any particular business segment, but rather represents any industry that must report upon and execute transactions of a financial nature.

Approach

The first step would be creating a table, this was done using a .psql file as were all the writers and readers. We would follow the base experimental design for the Performance Experiment measuring the latency with 1 writer (no reader), 1 reader (no writer), 10 writers and 1 reader, 10 readers and 1 writer, and finally 25 writers and 25 readers. Each of the 5 sample texts referenced in section C (Performance Testing) were written 10,000 times. For the CAP analysis experiment here were 3 writers, the first writer being the writer from which timestamp data was collected and the remaining two writers were utilized as sources of extra ’stress’ on the system. We were not able to set up a continuous loop as it did not write any data since it only ended on manual cancel. This was alleviated by setting the loop to one million so that it would run longer than the readers. It is important to note that this was done on a local machine, and we didn't scale to utilize cloud technology.

From there we took the data collected surrounding the latency numbers and used excel to provide us with the averages and standard deviation across the different experimental setups in a .csv format. The objectives will be to

1. Review of the 12 transactions to determine which should be converted to procedures and updated with appropriate sub-transaction handling
2. Updated C code for all transactions that need to be updated.
3. Updated pl/PGSQL code for all transactions that need to be updated.
4. The C++ code may need to be updated in conjunction with updates to the server-side business logic.

This benchmark report summarizing the test details and different statistics would align with the TPC-E is an On-Line Transaction Processing Benchmark.

This comprehensive project will allow me to work closely with my mentor Mark Wong and other potential mentors in the PostgreSQL community.

Timeline

By factoring in time to summarize and document my work with feedback, automated testing, and documentation in the proposed timeline, this will be conducive to a successful project.

Week	Plan of Action
May 20 - June 12 (Community Bonding Period)	- Research about techniques used in industry for stress benchmarking - The C++ code may need to be updated in conjunction with updates to the server-side business logic. - Discussion with my mentor (Mark Wong) about all the details needed to implement this project
June 13 - June 19	- Review of the 12 transactions to determine which should be converted to procedures and updated with appropriate sub-transaction handling - Finalize (initial) implementation plan

June 20 - July 10	- Implement a generic pipeline for generating random data
	  - Different data types[1] (numeric types, character types, date/time types)
	  - Different data sizes or scales (ranging from 1GB to 100 GB or more; needed to be discussed with mentor for more detail)
	- Update C code for all transactions that need to be updated.
	- Discuss detail with mentor; and get input/feedback about what and how to implement

July 11 - July 24	- Implement the required scripts (mentioned above) to manage the overall flow - Get some input/feedback from Mark Wong and other mentors about the implementation details - Update pl/PGSQL code for all transactions that need to be updated.
July 25 - July 31	- Review, test fix bug - Phase 1 evaluation (expect some discussion and paperwork) - Automated test cases for the industry - Finalize plan for next phase

[1] https://www.postgresql.org/docs/current/datatype.html

Aug 01- Aug 21	- For the CAP analysis experiment here were 3 writers, the first writer being the writer from which timestamp data was collected and the remaining two writers were utilized as sources of extra 'stress' on the system. - Pipeline for generating benchmark report - With test result and statistics (needed to be discuss with mentor about which metrics to display) - Discuss with mentors about the implementation details and how to improve
Aug 22 - Aug 28	- Extensive testing for testing scenarios implementation - Review, test and bug fixing
Aug 29 - Sep 04	- Write documentation and any other paperwork - Get feedback on things to improve and do refactoring (if needed)
Sep 05 - Sep 12	- Buffer zone for any slack in the project

Post-GSoC

After GSoC, I plan to continually contribute to the Postgres community and serve as a data engineer for quantitive hedge funds to upgrade to the service.

Availability

No conflicts noted at this time

Communication

I have joined slack and subscribed to the mailing list. Additional contact information will be provided.