Proposal for “Develop Performance Farm Benchmarks and Website”

10.04.2022

Yedil Serzhan
Uni Freiburg, Germany
edilserjan@gmail.com
Linkedin

1.Overview

The PostgreSQL Performance Farm (short for Pgperffarm) is a project to measure and visualize the benchmark performance of each version of PostgreSQL on Linux and macOS. Currently, Pgperffarm is using Pgbench of the same dataset & queries and different configurations. The main problem with Pgperffarm is that the variety of benchmark tests supported now is too limited. To enrich the functionalities of Pgperffarm, implementing **custom queries** for Pgbench is needed. So are **additional benchmarks** like TPC-h and YCSB. Apart from benchmark clients, the server also needs new functionalities like **email notifications** for various purposes.

2.My motivation and experience

My name is Yedil Serzhan and I am studying for a Master's degree in Computer Science at the University of Freiburg, Germany. I'm passionate about programming.

In the past, I was mainly interested in algorithms and I participated in many algorithm competitions. Recently I had the opportunity to get in touch with the usegalaxy.eu project, supported by the Bioinformatics Lab at the University of Freiburg, which is used by tens of thousands of bioinformaticians all over the world, and that's when I got a taste of open source projects.

I had a lot of experience in full-stack development before, and almost all of them used PostgreSQL, so when I heard about Google's summer of code and saw PostgreSQL in it, I was determined to get involved. So Here I am! To contribute and to learn.

3.List of deliverables

3.1 A python benchmark client

with the integration of TPC-h, YCSB and Pgbench with customizable queries.

3.2 A Django backend

with the integration of TPC-h, YCSB, Pgbench with customizable queries, respective databases and email notification.

3.3 A vanilla Js web front-end

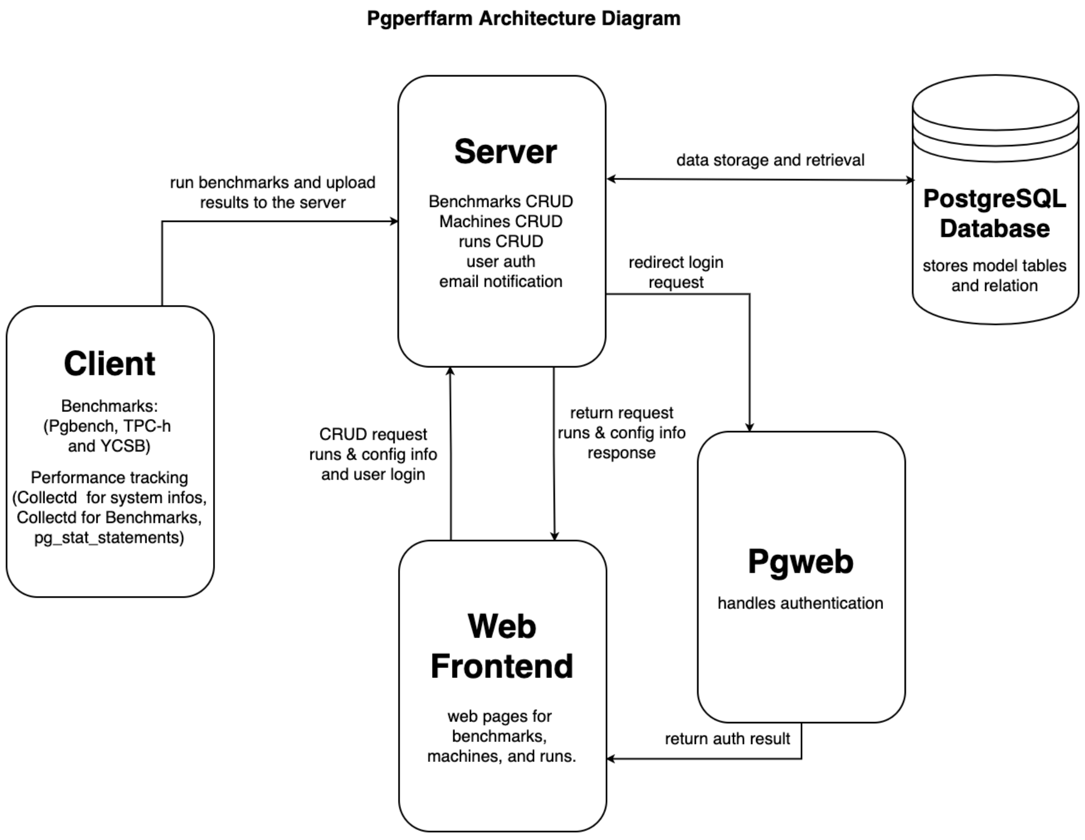
with improvements based on client-side functionality changes.
(more stylized and visualized)

3.4 Documentation and a report

with details about code usage and the process of the implementation.

4. Approaches

4.1 System architecture overview


Pgperffarm Architecture Diagram
Server
Benchmarks CRUD
Machines CRUD
runs CRUD
user auth
email notification
PostgreSQL Database
stores model tables and relation
data storage and retrieval
run benchmarks and upload results to the server
redirect login request
Client
Benchmarks: (Pgbench, TPC-h and YCSB)
Performance tracking (Collectd for system infos, Collectd for Benchmarks, pg_stat_statements)
CRUD request runs & config info and user login
return request runs & config info response
Pgweb
handles authentication
Web Frontend
web pages for benchmarks, machines, and runs.
return auth result

Above is a rough system architecture of Pgperffarm. Five components are composed of Pgperffarm. They are Python client, Django server, Web frontend, PostgreSQL and Pgweb. The respective relations are illustrated above.

The major parts that I mainly need to put effort into are client, server and frontend. Since Pgweb is for authentication usage only, the database can be generated by Django models.

Below are the more detailed approach descriptions for the 3 deliverables:

4.2 Approach for benchmark client

- Refactor the code to make the client scalable for additional benchmarks
- Add custom queries for Pgbench benchmarking.
- Add TPC-h benchmark (data, queries and performance metrics “QphH” i.e.)
- Add YCSB benchmark (data, queries and performance metrics “ops/sec” i.e.)
- Run Integration test to benchmarks.

Due to the need for extensions of Pgbench with extra benchmarks like TPC-h and YCSB. The first step I’m going to do is to refactor the client codebase to make it scalable for adding additional benchmarks in the future. Then we can add the benchmarks step by step.

4.3 Approach for web frontend

- Update Navbar with dropdown options for different benchmarks.
- Add TPC-h and YCSB benchmark web pages.
- Add support for querying certain attributes, such as date, machine, etc.
- Refine the web design and make the result more insightful and visualized.

Since we added more benchmarks, we need to update the web pages according to the changes. Like different benchmarks might focus on different performance metrics, so we need to design new pages and new tables for different benchmarks.

4.4 Approach for Django Server

- Add models for HTP-h, YCSB and email notification.
- Add related views
- Add related urls

In order to retrieve benchmarking results from the client and then send them to the web frontend, we need to add models to the server and generate the new database as well. And then implement the related views and urls to support user' request.

5. Schedule

- Intended workload : around 20 hours per week
- No other major commitments during the meantime.

5.1 Community bonding period: May 20 - June 12

- Read through the codebase carefully to make a good foundation for the coding phase.
- Understanding the details of TPC-h and YCSB implementation.
- Communicate and discuss with the mentor and the community about my doubts and questions, and learn from them.

5.2 Coding & work period: June 12 - Sep 4

Week 1 & 2 | June 13 - 27 | Codebase refactor

Refactor the codebase including client server and frontend to make it scalable for more benchmarks. At the same time, test it to make sure the changes won't change the intended behaviors of the project.

Week 3 & 4 | June 27 - July 11 | Add TPC-h benchmarking

Implement TPC-h benchmarking for Pgperffarm to achieve a workflow from benchmarking on the client side to seeing the results on the web frontend.

Week 5 & 6 | July 11 - 25 | Add YCSB benchmarking

Implement YCSB benchmarking for Pgperffarm to achieve a workflow from benchmarking on the client side to seeing the results on the web frontend.

July 25 - 29 | First evaluation

Submit for the evaluation, learn from the feedback from my mentor.

Week 7 & 8 | July 29 - Aug 12 | Add custom queries for Pgbench

Add custom queries for Pgbench benchmarking and it's desirable to be able to show the custom queries executed during the run on the frontend.

Week 9 | Aug 12 - 19 | Add email notification functionalities

Implement the feature that if the performance dropped by an undesired threshold, send an email to notify the user.

Week 10 | Aug 19 - 26 | Integration test & debugging

All features should be done by now. Run an integration test to find any bugs to correct them.

Week 10 | Aug 26 - Sep 2 | Documentation

Complete the documentation of Pgperffarm and publish it on github. Write a report of what I've experienced during the 10 weeks' time.

5.3 Post GSoC

After the GSoC is over, I still wish to use what I have learned along the way to continue to give back to and contribute to the community. And pass the experience of GSoC and the idea of contributing to open source projects to others!