DBMS Performance Evaluation

Author: 项楚涵
Student ID: 12413047
Course: Database Systems Principles

The report can be found at https://github.com/xch1234/DBMS-Performance-Evaluation.

Abstract

This report compares the performance of DBMS-based and file-based data operations and compares PostgreSQL and openGauss performance. By using a large dataset of over 10 million records, we measured and analyzed the retrieval and update execution times under identical hardware and software environments. The results show that DBMS havs unique advantages in efficiency, convenience and readability compared with data operations in files. Between PostgreSQL and openGauss, PostgreSQL demonstrated better performance under single-threaded conditions and higher compatibility across systems, while openGauss exhibited stronger security features. These findings highlight the advantages of DBMS usage for large-scale data processing and provide insights into performance trade-offs between PostgreSQL and openGauss.

Table of Contents

1. Introduction

1.1 Background

A **Database Management System (DBMS)** provides an efficient and convenient way to manage data, significantly improving the speed and reliability of data retrieval. **PostgreSQL** is a widely used open-source relational DBMS known for its robustness, advanced functionality, and strong compliance with SQL standards. **openGauss**, developed by Huawei, is an enterprise-grade open-source RDBMS designed for high performance, security, and scalability in demanding business environments.

In contrast to DBMS-based management, data can also be handled through **file-based operations**, a method commonly used by beginners when first learning about database.

1.2 Objectives

- To compare the performance of DBMS-based and file-based data operations in order to help beginners analyze the unique advantages of DBMS over file-based data operations.
- To compare PostgreSQL and openGauss performance using different kinds of benchmarks and analyze the better in different standard.

1.3 Report Structure

---

2. Experimental Design

2.1 Experimental Environment

We will compare the performance of file-based data operations, PostgreSQL and openGauss. Because the operating system is Windows and openGauss does not support a Windows version, we used Docker to install openGauss. And Because we want to compare the performance of PostgreSQL and openGauss more fairly, so we not only run PostgreSQL in local, but also use Docker to install PostgreSQL.

2.1.1 Hardware

Component	Specification
CPU	12th Gen Intel(R) Core(TM) i5-12500H (2.50 GHz, 12 cores, 16 threads)
RAM	16 GB DDR5 @ 4800 MT/s
Storage	SSD (477 GB, NVMe)
Network	1.1 Gbps Ethernet (wired, Intel Gigabit LAN) / Wi-Fi connection (200 Mbps)

2.1.2 Software

Component	Specification
Operating System	Windows 11 Home, Version 24H2 (Build 26100.6899)
Programming Language	C++17
PostgreSQL Version (local)	17
PostgreSQL Version (docker)	17
openGauss Version (docker)	7.0.0

Unless otherwise specified, all parameters are set to their default value.

2.2 Dataset

- **Type:** Orders database
- **Size:** 10235194 rows (>10M), 6 columns, 684MB
- **Format:** CSV
- **Source:** Order information from 3A Superstore, a fake retail chain located in Turkiye.
- **URL:** https://www.kaggle.com/datasets/cemeraan/3a-superstore?select=Orders.csv

If the dataset is too small, the performance difference (e.g., execution time) will not be significant. So we find a dataset from *kaggle* that the rows are over 10M.

2.3 Experimental Tasks

Performance tests focused on the **the execution time**, so we do these comparison on the following:

- **Retrieval Comparison:** Find orders with the word "Ali" in NAMESURNAME and record the execution time.
- **Update Comparison:** Change all "a" in NAMESURNAME to "A" and record the execution time.

3. Implementation

3.1 Data Import

- **PostgreSQL and openGauss:** We run the following SQL statements to create table and import CSV dataset:

```
create table orders (id integer not null,
                     title varchar not null,
                     DATE_ date,
                     USERID int,
                     NAMESURNAME varchar(100),
                     TOTALBASKET varchar(30));
\copy orders FROM 'path to Orders.csv'
             WITH (FORMAT csv, HEADER true, ENCODING 'UTF8');
```

- **file-based data operations (C++):** In cpp document, we run the following statements to read CSV dataset.

```
string filename = "Orders.csv";
ifstream file(filename);
while (getline(file, line)) ...;//read every lines of CSV document.
```

3.2 Queries

- **PostgreSQL and openGauss:**
  - **Retrieval Comparison:** Use SELECT in SQL to find orders with the word "Ali" in NAMESURNAME.

    ```
    \timing
    select * from orders where upper(NAMESURNAME) like 'ALI %'
                            or upper(NAMESURNAME) like '% ALI'
                            or upper(NAMESURNAME) like '% ALI %'
                            or upper(NAMESURNAME) like 'ALI';
    ```

  - **Update Comparison:** Use UPDATE in SQL to change all "a" in NAMESURNAME to "A".

    ```
    \timing
    update orders set NAMESURNAME=replace(NAMESURNAME,'a','A');
    ```

- **file-based data operations (C++):**
  - **Retrieval Comparison:** See in the `select.cpp`.
  - **Update Comparison:** See in the `update.cpp`.

3.3 Time Record

- **PostgreSQL and openGauss:** We run the following statement before queries:

  ```
  \timing
  ```

- **file-based data operations (C++):** We use `chrono` to record time:

  ```
  auto start = chrono::high_resolution_clock::now();

  //... queries

  auto end = chrono::high_resolution_clock::now();
  chrono::duration<double> elapsed = end - start;
  cerr << fixed << setprecision(3) << "\ntime: " << 1000.0 * elapsed.count() << " ms" <<
  endl;
  ```

---

4. Results and Analysis

Each query was executed five times, and the median execution time was taken as the final result to minimize deviations caused by system load fluctuations, caching effects, and other runtime factors.

4.1 Retrieval Performance

	file-based data operations(c++)	PostgreSQL(local)	PostgreSQL(docker)	openGauss(docker)
1	29339.600 ms	14705.206 ms	8399.244 ms	29393.966 ms
2	33620.630 ms	15289.207 ms	7417.053 ms	26115.121 ms
3	32828.762 ms	15418.504 ms	7536.076 ms	25909.257 ms
4	29978.595 ms	15145.231 ms	7389.660 ms	25958.853 ms
5	30303.700 ms	15154.551 ms	7538.672 ms	25804.604 ms
final	30303.700 ms	15154.551 ms	7536.075 ms	25958.853 ms

Analysis:

- File-based retrieval operation have longer execution time compared with all DBMS-based retrieval operations.
- PostgreSQL retrieval operation whether in local or in docker have shorter execution time compared with openGauss retrieval operations.

4.2 Update Performance

	file-based data operations(c++)	PostgreSQL(local)	PostgreSQL(docker)	openGauss(docker)
1	384826.994 ms	135854.258 ms	36953.798 ms	56651.535 ms
2	388355.940 ms	125680.891 ms	35028.346 ms	73746.298 ms
3	354737.900 ms	76495.985 ms	40087.316 ms	94564.363 ms
4	370378.339 ms	92355.403 ms	37577.887 ms	94122.161 ms
5	418492.267 ms	116344.491 ms	40288.629 ms	93638.738 ms
final	384826.994 ms	116344.491 ms	37577.887 ms	93638.738 ms

Analysis:

- File-based update operation have longer execution time compared with all DBMS-based update operations.
- PostgreSQL update operation in local have longer execution time compared with openGauss update operations. But in the similar environment (both in docker), PostgreSQL have shorter execution time compared with openGauss.

4.3 Overall Performance Summary

- DBMS-based operations always have better performance compared with file-based operations.
- The better in PostgreSQL and openGauss depends in their environment and query kinds. In the similar environment, PostgreSQL are always better in single thread query.

---

5. Discussion

5.1 File-based and DBMS-based

In this subsection, we discuss the unique advantages of a DBMS compared with data operations in files.

- **Efficiency:** In **4. Results and Analysis**, we can find that DBMSs show distinctly better performance compared with file-based operations.
- **Convenience:** In **3. Implementation**, compared SQL statements in DBMS-based operation with cpp documents in file-based operation, we can find DBMS is more convenient to write different kings of operation. If there are some changes in queries, file-based operations need to change cpp/java documents a lot and give a new document.
- **Readability:** In **3. Implementation**, we can also find it is easier to read SQL statements and understand what they will do. But file-based operations need people to read the total document which is always long.

5.2 PostgreSQL and openGauss

In this subsection, we discuss by different standards and in different situation, which one is better, PostgreSQL or openGauss.

- **Performance:** In the situation that both are in the **similar environment** and run **single thread** queries, **PostgreSQL** is always better for the beginner and the learner because of better performance In **4. Results and Analysis**.
- **Compatibility:** PostgreSQL can be installed in almost every operating system. But openGauss is not well-supported by many operating systems, like Windows. And PostgreSQL also has SQL standard compliance, broad API availability, rich ecosystem, and universal cloud integration. Therefore, in terms of compatibility, **PostgreSQL** performs better than openGauss.
- **Security:** openGauss provides stronger security than PostgreSQL through its multi-layer protection architecture, unified authentication, encryption and masking features, detailed auditing, and trusted execution environment, making it particularly suitable for high-security enterprise and government use cases. So in security, **openGauss** is better than PostgreSQL.

---

6. Conclusion and Future Work

6.1 Conclusion

- The unique advantages of a DBMS compared with data operations in files:
    - **Efficient**
    - **Convenient**
    - **Readable**
- By different standards, different DBMS are better:
    - **Performance (similar environment, single thread):** PostgreSQL.

  - **Compatibility:** PostgreSQL.
  - **Security:** openGauss.

6.2 Future Work

While this report contains some typical performance comparison works, there are several areas where can be studied in the future work:

- **Concurrency tests:** We only compare the performance of PostgreSQL and openGauss in single thread queries. In multi-threaded queries, there are maybe some difference in their performance.
- **Different size dataset tests:** Although in the smaller size dataset there are not distinct difference, there be able to have a bit better performance. And in the larger size dataset, there are maybe also different with now.

---

7. References

[1] PostgreSQL Global Development Group. (2024). *PostgreSQL 17 Documentation*. Retrieved from https://www.postgresql.org/docs/.

[2] openGauss Community. (2024). *openGauss Database User Guide*.
Retrieved from https://docs.opengauss.org/zh/.

---

8. Appendix

Code Snippets

- SQL document in the entire study for PostgreSQL and openGauss:

```
create table orders (id integer not null,
                     title varchar not null,
                     DATE_ date,
                     USERID int,
                     NAMESURNAME varchar(100),
                     TOTALBASKET varchar(30));
\copy orders FROM 'path to Orders.csv'
             WITH (FORMAT csv, HEADER true, ENCODING 'UTF8');
\timing
select * from orders where upper(NAMESURNAME) like 'ALI %'
                        or upper(NAMESURNAME) like '% ALI'
                        or upper(NAMESURNAME) like '% ALI %'
                        or upper(NAMESURNAME) like 'ALI';
update orders set NAMESURNAME=replace(NAMESURNAME,'a','A');
```

- cpp documents in the entire study for file-based operation: `select.cpp` and `update.cpp`.
