对 Postgresql 和 OpenGauss 的性能比较

摘要

在企业工作生产中，可能面临对于数据库产品的选择。一款优秀的数据库产品应当是稳定的，安全的，高性能的。具有这些特性，才能让数据不泄露，运行少出现问题，且能高效率地处理问题。Postgres 是一款强大的、开源的关系型数据库管理系统（RDBMS），以其稳定性、功能丰富性以及可扩展性而广泛应用。它支持 SQL（结构化查询语言）标准，并且有许多先进的特性，使得 PostgreSQL 成为开发者和企业的首选数据库之一。而 OpenGauss 是一款由华为研发的数据库语言，基于 PostgreSQL 的开源版本，进行了定制和优化，加入了多项自有技术与特性，适用于企业级应用、云计算、大数据处理、人工智能等领域。它强调高性能、高可用性和大规模分布式处理能力，尤其适用于大规模数据分析和海量数据存储的应用场景。

要比较两个数据库的优劣，可以从很多方面进行，例如性能、拓展性、数据一致性和完整性、兼容性、安全性等。其中性能是最重要的比较准则，最关键地影响到企业对于数据库产品的选择。因此，本文将从主要从性能方面通过实验对 postgresql 和华为 opengauss 进行对比。

数据库的性能差距主要体现在以下几个方面：连接数据库的速度、单次查询速度、全表扫描速度、处理并发请求能力、写入与输出性能。本实验将基于云服务器和 JDBC 对 OpenGauss 和 Postgres 进行比较，并得出二者各自的优劣之处。

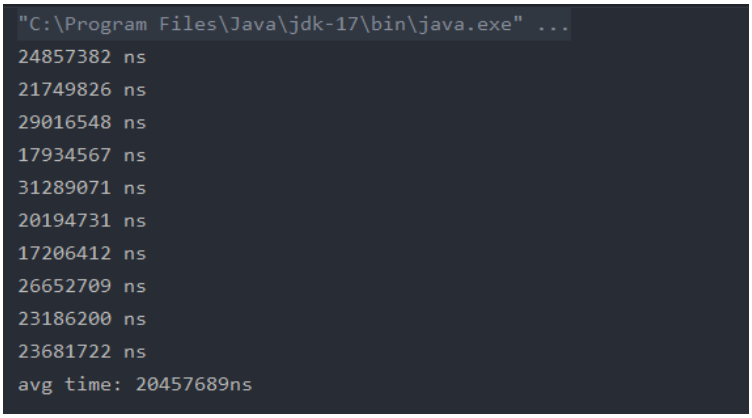
关键词　　数据库性能、JDBC、OpenGauss、Postgresql

为保证公平的性能测试，在华为云服务器上分别安装了 Postgresql 和 OpenGauss 数据库。

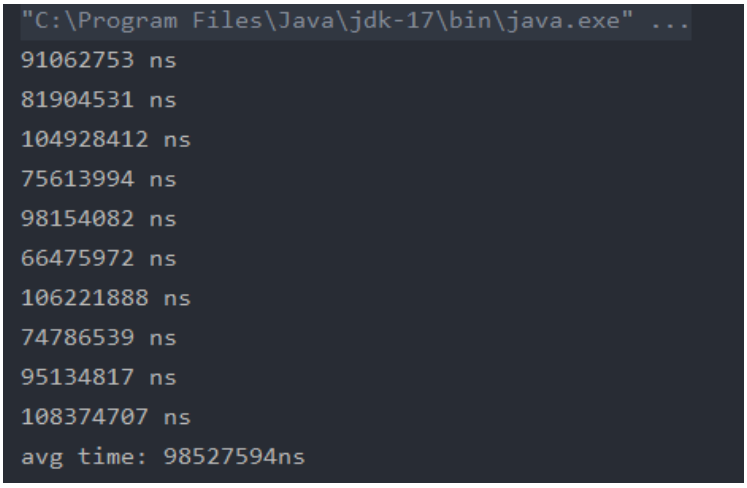
连接到 OpenGauss 数据库

1. 连接数据库的性能对比

测试流程：写了一个利用 jar 包连接到数据库的程序（DataBase.java），比较连接到 Postgres 和 OpenGauss 的需要时间，重复 10 次取平均值。结果如下：



Postgresql 的连接时间



OpenGauss 的连接时间

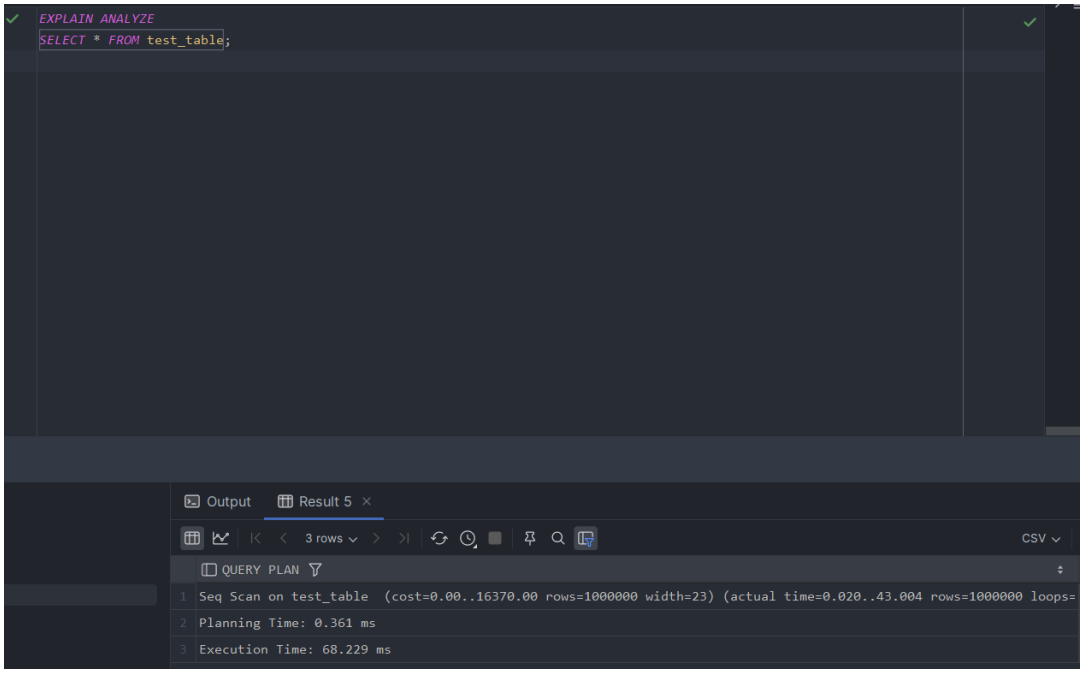
可以看到，Postgres 的连接速度明显快于 OpenGauss,差异大概在 4.8 倍左右。不过，连接数据库毕竟不是需要一直执行的任务，而 OpenGauss 的连接速度虽然不如 Postgres，但也在毫秒级别，可以说也比较快。

2. 全表扫描（select *）和写入（Insert）性能(IO)：

在现代数据库中，决定整体操作快慢的往往不是计算速度，而是 IO 处理的速度，即写入性能和输出性能。全表查询的能力体现了读取的性能，而执行大量 Insert 语句的速度决定了数据库写入的能力。

测试数据库语言的全表扫描性能可以通过生成具有大量数据的表格，然后使用 select 命令选取全部行观察所需时间。对于 Postgres,在 Datagrip 中生成一个有 1000000 行 VARCAHR 格式数据的 test_table，并进行全表扫描。

创建数据表并插入一百万条数据



使用 EXPLAIN 察看运行时间

```
openGauss=# create table test_table (name VARCHAR(100));
CREATE TABLE
openGauss=# -- 使用 generate_series 插入大量数据
openGauss=# INSERT INTO test_table (name)
openGauss-# SELECT 'name_' || gs
openGauss-# FROM generate_series(1, 1000000) AS gs;  -- 插入100万个记录
INSERT 0 1000000
openGauss=# -- 执行全表扫描并分析执行计划
openGauss=# EXPLAIN ANALYZE
openGauss=# SELECT * FROM test_table;
                                                        QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------
 Seq Scan on test_table  (cost=0.00..7186.49 rows=178049 width=218) (actual time=0.039..273.650 rows=1000000 loops=1)
 Total runtime: 335.465 ms
(2 rows)
```

使用 EXPLAIN 察看 OpenGauss 全表查询的时间

得到的结果是：使用同样的 Explain 命令查询时间时，在 Datagrip 中的 Postgres 用的总时间为 68.590ms，而 OpenGauss 使用的时间为 335.465ms。考虑到本地服务器和云服务器环境不同，云服务器的查询速度会受到网络速度和带宽的影响，所以使用云服务器上安装的 postgres 得到：

```
Seq Scan on test_table  (cost=0.00..15406.00 rows=1000000 width=11) (actual time=0.020..175.457 rows=1000000 loops=1)
 Total runtime: 162.745 ms
(2 rows)
```

经过多次测试取平均，得到 Postgres 进行大数据量全表查询的速度约是 OpenGauss 的 1.7 倍左右，差距较为明显，在数据数量级更高时会产生更大的差异。这种倍数差异在更换数据类型为 text，varchar，int 等时均保持基本一致。这说明了 Postgres 的读取能力优于 OpenGauss。

对于写入能力的测试，在插入一百万条数据的前后分别记录当前时间，对于

OpenGauss，前后一共用时大约为 9.04s。

```
-------------------------------
 2024-12-21 22:49:47.181074+08
(1 row)

openGauss=#
openGauss=# -- 执行批量插入
openGauss=# INSERT INTO test_data (data)
openGauss-# SELECT generate_series(1, 1000000);
-- 结束记录时间
SELECT clock_timestamp();
INSERT 0 1000000
openGauss=#
openGauss=# -- 结束记录时间
openGauss=# SELECT clock_timestamp();
        clock_timestamp
-------------------------------
 2024-12-21 22:49:56.228767+08
(1 row)
```

对于 Postgres，前后用时大约为 7.20s。

```
-------------------------------
 2024-12-21 22:54:36.729462+08
(1 row)

Postgres=#
Postgres=# -- 执行批量插入
Postgres=# INSERT INTO test_data (data)
Postgres-# SELECT generate_series(1, 1000000);
-- 结束记录时间
SELECT clock_timestamp();
INSERT 0 1000000
Postgres=#
Postgres=# -- 结束记录时间
Postgres=# SELECT clock_timestamp();
        clock_timestamp
-------------------------------
 2024-12-21 22:54:43.925875+08
```

因此，Postgres 的写入能力也优于 OpenGauss。在此项实验中，发现 Postgresql 的写入和读取能力均快于 OpenGauss，这可能导致在处理大量数据时，Postgres 的数据远快于 OpenGauss。

3. 条件查询和更新数据速度性能：当不是全表查询，而是有特定条件筛选时，例如（m==1）时，扫表查询速度的差距。需要提到的是，Postgres 支持并行查询，而 OpenGauss 并不支持。并行处理查询任务会显著地提升查询的性能。

   首先，将 filmdb.sql 导入 OpenGauss 数据库。对于表格 movies 分别进行查询 select * from movies where year_released=1983，得到结果如下：

```
 727 | Reuben, Reuben                | us |  1983 | 101
 730 | The Right Stuff               | us |  1983 | 192
 803 | Tender Mercies                | us |  1983 |  92
1090 | Flashdance                    | us |  1983 |  97
1091 | Never Say Never Again         | gb |  1983 | 134
1101 | Sudden Impact                 | us |  1983 | 117
1102 | Trading Places                | us |  1983 | 116
1103 | Return of the Jedi            | us |  1983 | 132
1104 | The House on Sorority Row     | us |  1983 |  91
1105 | Eddie Macon's Run             | us |  1983 |  95
1441 | Videodrome                    | ca |  1983 |  89
1526 | Le Bal                        | fr |  1983 | 110
1564 | Scarface                      | us |  1983 | 170
2319 | Berlin Alexanderplatz         | de |  1983 | 894
2379 | To Be or Not to Be            | us |  1983 | 107
2657 | Sans Soleil                   | fr |  1983 | 100
2660 | Nostalghia                    | ru |  1983 | 125
2788 | WarGames                      | us |  1983 | 114
3637 | Wavelength                    | us |  1983 |  87
3653 | Zelig                         | us |  1983 |  79
3682 | La Rue Cases-Nègres           | fr |  1983 | 103
penGauss=# EXPLAIN ANALYZE select * from movies where year_released=1983;
                                                  QUERY PLAN
--------------------------------------------------------------------------------------------------------------
Seq Scan on movies  (cost=0.00..194.21 rows=52 width=31) (actual time=0.019..2.216 rows=78 loops=1)
  Filter: (year_released = 1983)
  Rows Removed by Filter: 9459
Total runtime: 2.297 ms
4 rows)
```

OpenGauss 的运行结果

```
 Seq Scan on movies  (cost=0.00..194.21 rows=52 width=31) (actual time=0.019..2.216 rows=78 loops=1)
   Filter: (year_released = 1983)
   Rows Removed by Filter: 9459
 Total runtime: 2.846 ms
(4 rows)
```

Postgresql 的运行结果

可以发现，OpenGauss 略快于 Postgresql,但差距并不明显，因为查询结果的数量非常少。

随后，如果查询结果很多，例如使用命令：select * from movies where year_released>1983 时，得到结果：

```
Seq Scan on movies  (cost=0.00..194.21 rows=5125 width=31) (actual time=0.029..2.655 rows=5098 loops=1)
  Filter: (year_released > 1983)
  Rows Removed by Filter: 4439
Total runtime: 3.028 ms
(4 rows)
```

OpenGauss 的结果：3.028ms

```
Seq Scan on movies  (cost=0.00..194.21 rows=5125 width=31) (actual time=0.029..2.655 rows=5098 loops=1)
   Filter: (year_released > 1983)
   Rows Removed by Filter: 4439
 Total runtime: 2.189 ms
(4 rows)
```

Postgres 的结果：2.189ms

可以发现，在查询数据结果总量较多时，Postgresql 的表现更为优秀，大约是 OpenGauss 的 1.4 倍左右。

随后，对于不同数据类型，扩大（缩小）一定数据量后的数据库进行了同样的测试，得出的结果大致相同。

最后，对 Update 语句进行测试。

```
openGauss=# EXPLAIN ANALYSE UPDATE movies
openGauss-# SET runtime = 70000
openGauss-# WHERE movieid = 5;
                                                                  QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------
 Update on movies  (cost=0.00..8.27 rows=1 width=33) (actual time=0.135..0.137 rows=1 loops=1)
   ->  Index Scan using movies_pkey on movies  (cost=0.00..8.27 rows=1 width=33) (actual time=0.011..0.012 rows=1 loops=1)
         Index Cond: (movieid = 5)
 Total runtime: 0.244 ms
(4 rows)
```

OpenGauss 结果

```
Update on movies  (cost=0.00..8.27 rows=1 width=33) (actual time=0.135..0.137 rows=1 loops=1)
   ->  Index Scan using movies_pkey on movies  (cost=0.00..8.27 rows=1 width=33) (actual time=0.011..0.012 rows=1 loops=1)
         Index Cond: (movieid = 5)
 Total runtime: 0.234 ms
(4 rows)
```

Postgres 结果

可以得出，在使用 Index Scan 进行的单项扫描与更新中，二者的速度并无明显区别。

4. 并发处理能力：

为了测试并发处理能力，需要模拟多个用户同时对数据库进行访问和操作，探究数据库对于这些并发命令的应对性能如何。本实验采用 JDBC 分别连接 OpenGauss 和 Postgresql，在此基础上进行模拟（Multi_test.java），得到在同时进行 10 线程，每个线程执行 1000 个 insert 操作的条件下，OpenGauss 数据库执行 Insert 语句总耗时为 8237ms,平均每秒吞吐量为 1214.03 次

```
"C:\Program Files\Java\jdk-17\bin\java.exe" ...
Total Time: 8237ms
Total Number of Operations: 10000
Number of Operations per second): 1214.03

Process finished with exit code 0
```

而对于 Postgresql，总耗时为 10482ms,平均每秒吞吐量为 954.02 次。

```
Total Time: 10482ms
Total Number of Operations: 10000
Number of Operations per second): 954.02
```

可以发现 OpenGauss 在并发处理吞吐量上优于 Postgresql,这可能得益于 OpenGauss 增加的分布式架构对于并发处理的优化。

把同时处理的用户数改为 100，再次进行实验。OpenGauss 总共耗费了 278620ms,大约 5 分钟，来处理完这些请求，吞吐量为 358.91。

```
Total Time: 278620ms
Total Number of Operations: 100000
Number of Operations per second): 358.91
```

而 Postgresql 则耗费了 459274ms,大约 8 分钟，吞吐量为 217.73。

```
Total Time: 459274ms
Total Number of Operations: 100000
Number of Operations per second): 217.73
```

随着并发用户数量的提升，处理效率之间的差距也变大。OpenGauss 明显快于 Postgresql。

对于 Update 命令的测试与 Insert 测试结果大致相同,同样由 OpenGauss 领先。可以得出，两者速度之间的差距与 sql 命令种类无关，而是与数据库对于并发请求的处理能力相关性更高。

而在性能以外的方面,由于 Postgres 发展时间较长,所以社区成熟度较高，经常进行更新，而 OpenGauss 由于发展时间较短，还未能形成成熟的讨论环境与社区环境。例如，OpenGauss 目前不能实现并行处理部分子任务，而 Postgres 在 9.6 版本后的更新后可以。

此外，在测试中，OpenGauss 的编译需要很多依赖，而且版本固定，造成跨平台编译难度很大，平台通用性差。相比之下，Postgres 在环境配置方面更容易，并不需要很多依赖，因此能够吸引更多用户。而且，如 DataGrip 等热门数据库管理工具也只支持 Postgres,因此在开发难度和便利度方面来看，更多用户和企业会倾向于选择 Postgres。

综上所述，根据以上实验与分析，初步得出 Postgresql 与 OpenGauss 的优劣点。相比于经典的数据库语言 Postgres,OpenGauss 的优势在于：

1. 支持分布式架构，这使得它相较于传统的单节点式的 Postgres，在大规模数据存储和高并发情况下具有更好的扩展性和性能。在并发处理能力测试中表现更好。
2. OpenGauss 在高并发的情况下也能保持更高程度的稳定。
3. 在对于少量数据的处理上略优于 Postgres。

OpenGauss 的劣势在于：

1. 在单个用户进行大量数据的操作时，进行基本 sql 命令和写入和读取速度不如 Postgres。
2. 连接到数据库的速度相对较慢。
3. 社区成熟度较低，导致用户得到的支持较少，更新迭代也相对较慢。
4. 不能并行处理部分子任务。
5. 编译依赖较多，平台通用性较差，无法积累大量用户。

当然，由于网络延迟、服务器性能差距等问题的存在，本实验的测试结果可能与真实应用场景存在偏差。进一步比较两种数据库的优劣需要使用更专业的工具。通过本实验，发现 OpenGauss 在 Postgres 的框架上进行了优化，在例如分布式架构、稳定性等多个方面取得了更好的效果。同时，它也存在一些不足。期待 OpenGauss 在未来进一步的更新与发展。