openGauss 与 PostgreSQL 的对标：比较分析

目录

1. 前言

1.1 数据库

数据库是一种系统化的数据存储解决方案，它允许用户以结构化的方式存储、检索和管理数据，通过提供数据一致性、完整性和安全性，支持各种应用程序和服务的数据需求。

当下，随着对于数据处理的要求越来越大，数据库正在蓬勃发展，不仅有经典的 PostgreSQL、MySQL 等数据库，还有在国内兴起的诸多数据库，例如 GaussDB、OceanBase、PolarDB 等。

市面上的数据库越来越多，我们势必要对数据库的性能做出测试和比较。本文就会对 openGauss 和 PostgreSQL 的性能做比较分析，并对 openGauss 做出评估。

1.2 openGauss

openGauss 数据库源于 PostgreSQL-XC 项目，内核源于 Postgres 9.2.4，总代码量约 120W 行，其中内核代码约 95W 行。华为结合企业级场景需求，深度融合其在数据库领域多年的经验，新增或修改了内核代码约 70W 行，内核代码修改比例约占总内核代码量的 74%。openGauss 保留了原先 PostgreSQL 的接口和公共函数代码(约 25W 行)，仅对这些代码做了适当优化，与现有的 PG 生态兼容性较好。

openGauss 是一款高性能，高安全，高可靠的企业级开源关系数据库。openGauss 特性是一款开源关系型数据库管理系统，采用木兰宽松许可证 v2 发行。openGauss 内核深度融合华为在数据库领域多年的经验，结合企业级场景需求，持续构建竞争力特性。

在墨天轮发布的 2024 年 12 月中国数据库流行度排行榜中，openGauss 位列第四。

排行	上月	半年前	名称	模型	数据处理	部署方式	商业模式	专利	论文	案例	资质	书籍	得分
1	1	1	OceanBase	关系型	HP			151	19	43	16	2	710.49
2	2	2	PolarDB	关系型	HP			592	78	85	14	2	688.84
3	↑↑ 5	↑↑↑ 9	GoldenDB	关系型	HP			806	54	104	12	2	668.51
4	4	↑↑↑ 7	GaussDB	关系型	TP HP			933	63	34	17	5	648.30
5	↓↓ 3	5	TiDB	关系型	HP			40	25	19	10	1	588.94

1.3 PostgreSQL

PostgreSQL 是一个功能丰富、开源的对象关系型数据库管理系统，以其对 SQL 标准的严格遵循、强大的数据类型支持、高级索引选项、窗口函数、表继承以及优秀的并发控制而闻名，使其成为从小型项目到大型企业应用的理想选择。

由于其的开源属性，openGuass 就是在其基础上编写的。

1.4 比较分析

因为 openGuass 是在 PostgreSQL 的基础上编写，两者目前都是的曾经 PostgreSQL 版本的改进版。比较的结果将不仅体现数据库的性能情况，也体现数据库的改进成效。

2. 评估数据库性能的指标

2.1 考察维度

数据库的性能要求要面对多种多样的现实情形，有单个用户在处理事务时的性能，有不同用户在同时处理事务时的性能，有面对大量事务处理时对硬件的需求等等。

本次的实验尽可能的覆盖数据库在现实中可能会遇到的情况。

2.2 单一用户的事务处理

数据库作为数据处理的工具，目的就是帮助人们简单、方便而又高效的处理数据。很多时候，人们会利用数据库处理私人的信息。在这种情况下，数据库往往面对的是一个用户的数据库操作指令。所以，数据库处理单端口信息的能力和性能十分重要。该项能力会作为评估的一个指标。

在该部分，将会比较两个数据库，插入、更新、查询、删除等多种操作的所需时间。

2.3 多用户的事务处理

数据库作为在数据量蓬勃发展下的产物，用于私人处理数据仅仅是衍生的用法，更重要的是能对一定规模的数据进行处理。在现实的场景中，一个数据库往往连接着一定数量的用户，而这些用户可能会并行地发出事务处理的请求。如果数据库不能很好的解决这一问题，会导致反馈结果所需的时间大大增加，影响用户的使用体验。所以，数据库处理并发请求的能力与性能十分重要。这项能力也会作为评估的一个指标。

在该部分，将会采用多种方式，模拟现实中多用户进行事务处理的情形。

2.4 硬件资源的使用情况

数据库在部署初期，可能只是面对一个小规模的数据处理要求。随着进一步的发展，对数据处理的要求会不断提高。同时，用户对于数据库的访问可能由于现实中的某些情况而突发性的增长。部署在相同硬件设施上的数据库如果有对硬件资源更好的利用，那么在面对风险和挑战时就能更好的应对。

3. 实验测量

3.1 实验环境

3.1.1 云服务器性能

操作系统

```
NAME="openEuler"
VERSION="20.03 (LTS)"
ID="openEuler"
VERSION_ID="20.03"
PRETTY_NAME="openEuler 20.03 (LTS)"
ANSI_COLOR="0;31"
```

内存

```
              total        used        free      shared  buff/cache   available
Mem:          2.9Gi       740Mi       264Mi       217Mi       1.9Gi       1.6Gi
Swap:            0B          0B          0B
```

CPU

```
Architecture:                    aarch64
CPU op-mode(s):                  64-bit
Byte Order:                      Little Endian
CPU(s):                          2
On-line CPU(s) list:             0,1
Thread(s) per core:              1
Core(s) per socket:              2
Socket(s):                       1
NUMA node(s):                    1
Vendor ID:                       HiSilicon
Model:                           0
Model name:                      Kunpeng-920
Stepping:                        0x1
CPU max MHz:                     2600.0000
CPU min MHz:                     2600.0000
BogoMIPS:                        200.00
L1d cache:                       128 KiB
L1i cache:                       128 KiB
L2 cache:                        1 MiB
L3 cache:                        32 MiB
NUMA node0 CPU(s):               0,1
Vulnerability Itlb multihit:     Not affected
Vulnerability L1tf:              Not affected
Vulnerability Mds:               Not affected
Vulnerability Meltdown:          Not affected
Vulnerability Spec store bypass: Vulnerable
Vulnerability Spectre v1:        Mitigation; __user pointer sanitization
Vulnerability Spectre v2:        Not affected
Vulnerability Srbds:             Not affected
Vulnerability Tsx async abort:   Not affected
Flags:                           fp asimd evtstrm aes pmull sha1 sha2 crc32 atomics fphp asimdhp cpuid asimdrdm jscvt fcma dcpop asimddp asimdfhm
```

值得一提的是，由于华为云服务器为了降低用户对于服务器传输文件性能的高要求，在文件上传上，有 500MB 的限制，使得数据库中表格的大小受到一定限制。

3.1.2 openGuass

版本 openGauss 5.0.1

```
                                                              version
------------------------------------------------------------------------------------------------------------------------------------------------
 (openGauss 5.0.1 build 33b035fd) compiled at 2023-12-15 20:28:19 commit 0 last mr   on aarch64-unknown-linux-gnu, compiled by g++ (GCC) 7.3.0, 64-bit
(1 row)
```

3.1.3 PostgreSQL

版本 PostgreSQL 14.11

```
[postgres@firebolt ~]$ pg_config --version
PostgreSQL 14.11
```

3.1.4 数据集

movies_metadata 32.8MB 共 45460 行 24 列。

3.2 对单一用户处理事务性能的评估

3.2.1 测试方式

在该部分，主要通过使用命令行和运行.sql 文件的方式对数据的性能进行考察。

3.2.2 插入操作

运行含有插入操作的 4.sql 文件，查看运行时间(由于.sql 文件与 openGuass 的兼容性存在一定问题，在输入表格后缺失一行，但相较于庞大的数据，此处可忽略)

openGuass 插入时间 65.291s

```
id |         event_time
----+----------------------------
  4 | 2024-12-22 15:11:53.577686
  5 | 2024-12-22 15:12:58.868911
```

PostgreSQL 插入时间 52.193s

```
id |         event_time
----+----------------------------
  2 | 2024-12-22 07:45:16.766644
  3 | 2024-12-22 07:46:08.959639
```

3.2.3 查询操作

使用 EXPLAIN ANALYZE 对查询操作的时间做分析

操作一：查询所有内容

```
EXPLAIN ANALYZE select * from movies_metadata;
```

openGuass 查询时间 1.727s

```
                                                     QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------
 Seq Scan on movies_metadata  (cost=0.00..4917.08 rows=38408 width=934) (actual time=0.046..1723.185 rows=45459 loops=1)
 Total runtime: 1727.385 ms
(2 rows)
```

PostgreSQL 查询总时间 0.022s

```
                                                     QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------
 Seq Scan on movies_metadata  (cost=0.00..9416.60 rows=45460 width=934) (actual time=0.016..20.282 rows=45460 loops=1)
 Planning Time: 0.370 ms
 Execution Time: 22.083 ms
(3 rows)
```

操作二：查询符合要求的行的数量

```
EXPLAIN ANALYZE select count(*) from movies_metadata where original_language = 'cn';
```

openGuass 查询时间 0.373s

```
                                                     QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=51039.47..51039.48 rows=1 width=8) (actual time=373.571..373.571 rows=1 loops=1)
   ->  Seq Scan on movies_metadata  (cost=0.00..51038.03 rows=578 width=0) (actual time=334.544..373.489 rows=313 loops=1)
         Filter: (original_language = 'cn'::text)
         Rows Removed by Filter: 45146
 Total runtime: 373.691 ms
(5 rows)
```

PostgreSQL 查询时间 0.026s

```
                                                     QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------
 Aggregate  (cost=9531.02..9531.03 rows=1 width=8) (actual time=26.298..26.299 rows=1 loops=1)
   ->  Seq Scan on movies_metadata  (cost=0.00..9530.25 rows=308 width=0) (actual time=0.215..26.261 rows=313 loops=1)
         Filter: (original_language = 'cn'::text)
         Rows Removed by Filter: 45147
 Planning Time: 0.125 ms
 Execution Time: 26.325 ms
(6 rows)
```

3.2.4 更新操作

使用 EXPLAIN ANALYZE 对更新操作的时间做分析

```
EXPLAIN ANALYZE UPDATE movies_metadata SET adult = 'True' WHERE adult = 'False';
```

openGuass 更新时间 3.214s

```
                                                     QUERY PLAN
---------------------------------------------------------------------------------------------------------------------------------
 Update on movies_metadata  (cost=0.00..45926.82 rows=45654 width=935) (actual time=0.322..3214.000 rows=45450 loops=1)
   ->  Seq Scan on movies_metadata  (cost=0.00..45926.82 rows=45654 width=935) (actual time=0.297..2928.305 rows=45450 loops=1)
         Filter: (adult = 'False'::text)
         Rows Removed by Filter: 9
 Total runtime: 3214.141 ms
```

PostgreSQL 更新时间 0.364s

```
                                                                QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------
 Update on movies_metadata  (cost=0.00..9530.25 rows=0 width=0) (actual time=364.385..364.385 rows=0 loops=1)
   ->  Seq Scan on movies_metadata  (cost=0.00..9530.25 rows=45455 width=38) (actual time=0.062..109.039 rows=45451 loops=1)
         Filter: (adult = 'False'::text)
         Rows Removed by Filter: 9
 Planning Time: 0.053 ms
 Execution Time: 364.430 ms
(6 rows)
```

3.2.5 删除操作

使用 EXPLAIN ANALYZE 对删除操作的时间做分析

```
EXPLAIN ANALYZE DELETE FROM movies_metadata;
```

openGuass 删除时间 4.379s

```
                                                                QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------
 Delete on movies_metadata  (cost=0.00..45812.66 rows=45666 width=6) (actual time=0.053..4379.738 rows=45459 loops=1)
   ->  Seq Scan on movies_metadata  (cost=0.00..45812.66 rows=45666 width=6) (actual time=0.021..4297.939 rows=45459 loops=1)
 Total runtime: 4379.861 ms
(3 rows)
```

PostgreSQL 删除时间 0.100s

```
                                                                QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------
 Delete on movies_metadata  (cost=0.00..9502.76 rows=0 width=0) (actual time=99.963..99.964 rows=0 loops=1)
   ->  Seq Scan on movies_metadata  (cost=0.00..9502.76 rows=45876 width=6) (actual time=0.056..76.702 rows=45460 loops=1)
 Planning Time: 0.036 ms
 Execution Time: 99.985 ms
(4 rows)
```

3.2.6 小结

对实验数据进行统计，并制作表格

数据库	插入时间	查询时间		更新时间	删除时间
		操作一	操作二		
openGuass	65.291s	1.727s	0.373s	3.214s	4.379s
PostgreSQL	52.193s	0.022s	0.026s	0.364s	0.100s
比例	1.251	78.50	14.35	8.830	43.79

总体上将，PostgreSQL 作为经典的数据库，在基础的功能上还是十分扎实的，在各种操作上都优于 openGuass。

3.3 对多个用户同时处理事务性能的评估

3.3.1 测试方法

在这一部分，主要利用.sql 文件和多个客户端连接来进行测试。

一个用户通过运行.sql 文件进行一段时间连续的文件输入，其余用户不定时利用剪切板快速输入查询、更新、删除等多条指令等多种方式，通过 EXPLAIN ANALYZE 了解运行时间。

3.3.2 双用户情形

操作一：考察多用户情况下插入操作的情况

一个用户通过运行 4.sql 文件进行一段时间连续的文件输入，另一用户不定时利用剪切板快速输入查询指令。

```
EXPLAIN ANALYZE select * from movies_metadata;
```

openGuass

数据量	4487	11552	23405	37118	45459
运行时间	1.277s	0.188s	0.180s	0.182s	0.183s

```
                                                    QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------
 Seq Scan on movies_metadata  (cost=0.00..24106.46 rows=45146 width=933) (actual time=30.316..1276.842 rows=4487 loops=1)
 Total runtime: 1277.300 ms
(2 rows)
```

```
                                                    QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------
 Seq Scan on movies_metadata  (cost=0.00..24106.46 rows=45146 width=933) (actual time=0.028..187.291 rows=11552 loops=1)
 Total runtime: 188.272 ms
(2 rows)
```

```
                                                    QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------
 Seq Scan on movies_metadata  (cost=0.00..24106.46 rows=45146 width=933) (actual time=0.021..177.874 rows=23405 loops=1)
 Total runtime: 179.848 ms
(2 rows)
```

```
                                                    QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------
 Seq Scan on movies_metadata  (cost=0.00..24106.46 rows=45146 width=933) (actual time=0.024..179.189 rows=37118 loops=1)
 Total runtime: 182.281 ms
(2 rows)
```

```
                                                    QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------
 Seq Scan on movies_metadata  (cost=0.00..24106.46 rows=45146 width=933) (actual time=0.037..179.806 rows=45459 loops=1)
 Total runtime: 183.454 ms
(2 rows)

db_tpcc=> EXPLAIN ANALYZE select * from movies_metadata;
```

PostgreSQL

数据量	660	17001	26052	34497	42623
运行时间	0.020s	0.021s	0.026s	0.025s	0.035s

```
                                                    QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------
 Seq Scan on movies_metadata  (cost=0.00..9380.13 rows=2913 width=982) (actual time=0.006..19.534 rows=6607 loops=1)
 Planning Time: 0.403 ms
 Execution Time: 19.804 ms
(3 rows)
```

```
                                                    QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------
 Seq Scan on movies_metadata  (cost=0.00..9380.13 rows=2913 width=982) (actual time=0.014..20.032 rows=17001 loops=1)
 Planning Time: 0.092 ms
 Execution Time: 20.722 ms
(3 rows)
```

```
                                                    QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------
 Seq Scan on movies_metadata  (cost=0.00..9380.13 rows=2913 width=982) (actual time=0.015..24.546 rows=26052 loops=1)
 Planning Time: 0.038 ms
 Execution Time: 26.020 ms
(3 rows)
```

```
                                                    QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------
 Seq Scan on movies_metadata  (cost=0.00..9380.13 rows=2913 width=982) (actual time=0.012..23.922 rows=34497 loops=1)
 Planning Time: 0.046 ms
 Execution Time: 25.288 ms
(3 rows)
```

```
                                                    QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------
 Seq Scan on movies_metadata  (cost=0.00..9380.13 rows=2913 width=982) (actual time=0.013..33.349 rows=42623 loops=1)
 Planning Time: 0.038 ms
 Execution Time: 35.042 ms
(3 rows)
```

操作二：用户进行插入操作的相互影响

一个用户通过运行 4.sql 文件进行一段时间连续的文件输入，另一用户也运用 4.sql 文件进行输入，但统计其插入操作的时间。

openGuass 插入时间 90.800s

```
 id |         event_time
----+----------------------------
  1 | 2024-12-22 13:33:55.109847
  2 | 2024-12-22 13:35:25.909759
(2 rows)
```

PostgreSQL 插入时间 92.030s

```
 id |         event_time
----+----------------------------
  1 | 2024-12-22 14:25:40.923192
  2 | 2024-12-22 14:27:12.953371
(2 rows)
```

3.3.3 多用户情形

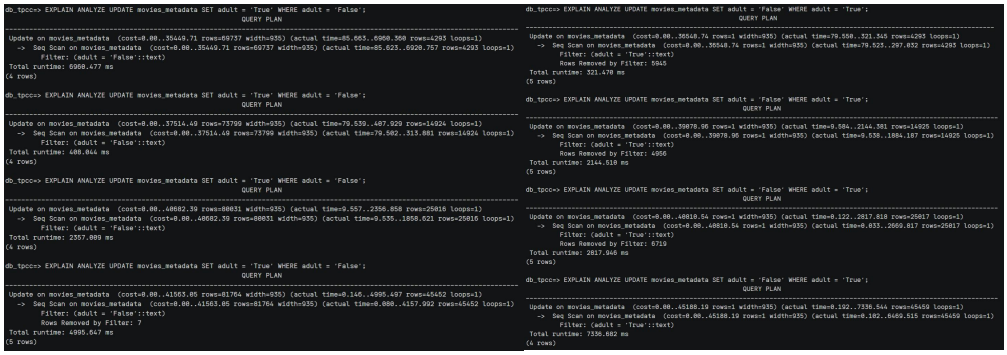
此处以三用户为例，用户一运行 4.sql 文件进行一段时间连续的文件输入，用户二和三则不定时的依次利用剪切板快速更新指令，为提高对数据库的考察，用户二和三所更新的内容会相互干扰以增多数据的更新量。

```
EXPLAIN ANALYZE UPDATE movies_metadata SET adult = 'True' WHERE adult = 'False';--用户二
EXPLAIN ANALYZE UPDATE movies_metadata SET adult = 'False' WHERE adult = 'True';--用户三
```

openGuass

用户编号	更新指令一	更新指令二	更新指令三	更新指令四
用户二	6.960s	0.408s	2.357s	4.996s
用户三	0.321s	2.145s	2.818s	7.337s

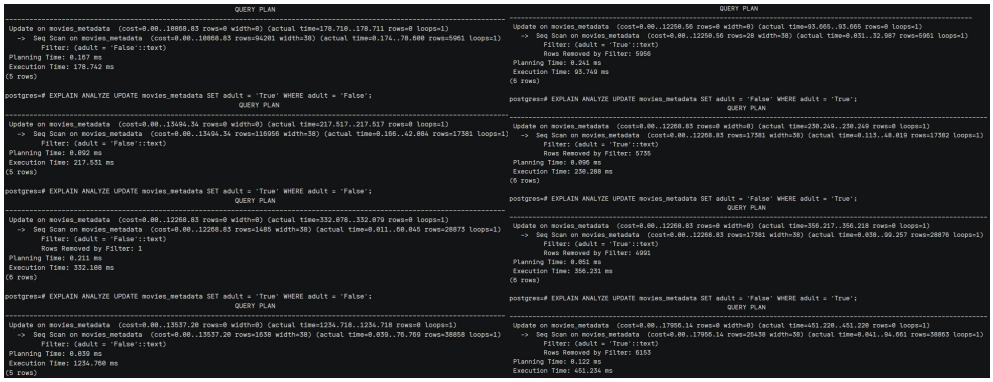
具体情形：



Postgresql

用户编号	更新指令一	更新指令二	更新指令三	更新指令四
用户二	0.179s	0.218s	0.332s	1.235s
用户三	0.094s	0.230s	0.356s	0.451s

具体情形



3.3.4 小结

总体上讲，两个数据库之间还是存在不小的差距。除了在少数情况下 openGuass 与 PostgreSQL 差不多外，在大部分情况下，openGuass 远不如 PostgreSQL，甚至面对某些情况，两者的差距到达了两个数量级。

3.4 硬件使用情况的评估

3.4.1 测试方式

在对应的数据库中运行 4.sql 文件，进行读入操作，在云监控服务中查看各项硬件的使用情况。

3.4.2 执行结果

openGuass

PostgreSQL

3.4.3 小结

总体上讲，openGuass 和 PostgreSQL 在处理高频的读写时，对硬件的使用情况相差不大，但具体比较 PostgreSQL 还是略胜一筹。在占用 CPU 较小的情况下，PostgreSQL 还保持这着更高的命令读取和磁盘写入的速度。

4. openGuass 性能评估

openGuass 作为一款基于 PostgreSQL 开发的具有一定创新性的关系数据库，处理事务能力和性能在可接受的范围内，但相较于当下较新版本的 PostgreSQL 还是有一定的差距。

在单一用户事务处理的性能上，openGuass 的运行时间会慢于 PostgreSQL，在大部分情况下，差距一般在一个数量级。在多个用户事务处理的性能上，openGuass 的运行时间依旧会慢于 PostgreSQL，在大部分情况下，差距是比较明显的，在某些问题的处理上，甚至会出现两个数量级的差距。在硬件资源的利用方面，两者是比较接近的，PostgreSQL 也仅仅比 openGuass 略好一点。

在本实验小规模数据对于两个数据库性能的测试中，openGuass 并没有体现出相较于 PostgreSQL 的明显优势，甚至在很多方面不如当下版本的 PostgreSQL。但是，考虑到 PostgreSQL 本身就是处理小规模数据的利器，在小规模数据上更是强于国内热门数据库 MySQL 的存在，openGuass 本身的性能已经是足够优秀了。

总体上来说，openGuass 是一款比较优秀的数据库，但并没有完全达到开发者所声称的那么强大和恐怖。此外，在基于 PostgreSQL 开发的基础上，openGuass 可能一定程度上增加了企业级应用的能力，但同时也损失了一部分 PostgreSQL 独有的快速处理小规模数据的能力。

5. 总结

本次实验想通过 openGuass 和 PostgreSQL 的对标，来评估 openGuass 数据库的性能，从而来判断 openGuass 是否如开发人员所说的十分强大。在本实验中，利用小规模的数据集，主要从单个用户在处理事务时的性能，不同用户在同时处理事务时的性能和面对大量事务处理时对硬件的需求三个方面来设计方案，对两个数据库的性能进行比较和评估。最后得出结论，openGuass 是一款比较优秀的数据库，但并没有完全达到开发者所声称的那么强大和恐怖。

在本次实验中，受限于华为云服务器对上传文件大小的限制，所用的数据集规模也受到限制。在前进性上，用更大规模的数据集可能更能去反映两个数据库的优劣差别。此外，openGuass 的内核源于 Postgres 9.2.4。在前进性上，可以尝试利用旧版本的 PostgreSQL 来与其进行比较，来更加有效的体现 openGuass 的优化性。

Github 仓库：https://github.com/Nimbus3009/DataBase_Project3