DBMS 性能评估项目报告

江博成12312505

项目内容

本项目通过列出衡量数据库优劣的关键指标，进而设计实验，对比 openGauss 和 PostgreSQL 的各个关键指标，以全面评价 openGauss 的优势与缺点。

项目的Github链接：https://github.com/Baiyu0123/project3

测试配置与环境

测试使用个人笔记本，通过 docker 提供 linux 环境。

使用的处理器是 11th Gen Intel(R) Core(TM) i7-1165G7 @ 2.80GHz 2.80GHz，内存32GB。

第一节：衡量数据库优劣的指标

衡量数据库最核心的指标是性能。

单个用户使用时，查询响应时间和数据处理速度都是很关键的，这将在第二节具体设计实验进行对比。

站在企业视角上，数据库对高并发的支持程度同样关键。在第三节的实验对比中，我将着重选取查询响应时间和每秒事务数（TPS）作为衡量指标。

性能之外，安全性和易用性也是用户在意的指标，这将在第四节做一些讨论。

除了这些以外，在企业视角上，数据库是否可靠（包括确保数据随时间保持准确和一致，在故障后能否快速恢复操作），是否具有可扩展性（细分为垂直扩展性和水平扩展性，垂直扩展性指通过添加更强大的硬件资源来增加数据库容量的能力，水平扩展性指跨多个服务器或节点分布数据库以增加容量和性能的能力）等指标同样十分关键。但限于笔者水平和硬件配置不足，这部分不在本 project 中比较。

第二节：基础性能测试

2.1 插入数据

第2节中，我使用c++程序 sql_ger.cpp 生成了 insert.sql。insert.sql 脚本创建了表 text_table，接着插入 10^6 条数据。text_table 的参数和生成方式见下文代码及注释。

```
CREATE TABLE test_table(
    id integer, --id是从1到100000
    x integer, --x在0到50间随机生成
    y integer, --y在0到998244353间随机生成
    s character(16) --s包含16个小写字母，每个位置的小写字母随机生成
);
```

在 openGauss 和 PostgreSQL 中分别运行 insert.sql，运行时间如下：

	运行时间（秒）
openGauss	123.869

	运行时间（秒）
PostgreSQL	71.842

可以发现，openGauss 执行这 10^6 条 INSERT 指令的用时是 PostgreSQL 的 1.72 倍。

2.2 查询数据

尝试以下查询（select.sql）：

```
--1 查询x小于30的行数
select count(*) from test_table where x <=30;

--2 查询s包含子串ab的行数
select count(*) from test_table where s like '%ab%';

--3 复合查询
select avg(y/id) from test_table where x <=40 and y%10!=id%10 and s not like
'abc';

--4 复杂查询
select tmp_table.mx
from (select max(test_table.y) mx
      from test_table
      group by x) tmp_table;

--5 多表联合查询
select count(*)
from (select max(test_table.y) mx
      from test_table
      group by x) tmp_table,test_table where tmp_table.mx%100>test_table.x;
```

在 openGauss 和 PostgreSQL 中分别运行，运行时间（毫秒）如下：

	1	2	3	4	5
openGauss	156	170	425	283	7582
PostgreSQL	79	91	114	140	2887

可以发现，在五项简单查询中 openGauss 的运行时间都长于 PostgreSQL。

2.3 修改数据

尝试以下修改（update.sql）：

```
--1 数字修改
UPDATE test_table SET y=y+1 WHERE x<=40;

--2 字符替换
UPDATE test_table SET s = REPLACE(s, 'a', '1') WHERE s LIKE '%a%';

--3 大小写转换
UPDATE test_table SET s=upper(s);
```

在 openGauss 和 PostgreSQL 中分别运行，运行时间（秒）如下：

	1	2	3
openGauss	8.744	12.983	18.189
PostgreSQL	2.108	1.318	2.431

可以发现，在三项简单修改中 openGauss 的运行时间都远长于 PostgreSQL。

补充测试

在 2.1.3 节的修改测试中，发现 openGauss 修改的运行时间波动很大，以第三组测试（大小写替换）为例多次测试，运行时间（秒）如下：

	1	2	3	4	5	6	7	8
openGauss	22.298	21.339	20.572	10.19	19.03	12.962	8.895	16.497

据计算，八次测试的平均值是 15.223 秒，标准差是 5.329 秒。这说明 openGauss 的运行稳定性不佳，同样的测试数据，在 PostgreSQL 的运行时间的极差小于 0.5 秒。

2.4 排序数据

尝试运行以下排序代码（orderby.sql）：

```
--1大值域整数排序
select * from test_table order by y;

--2长为16的定长随机字符串排序
select * from test_table order by s;

--3小值域整数排序
select * from test_table order by x;
```

在 openGauss 和 PostgreSQL 中分别运行，运行时间（秒）如下：

	1	2	3
openGauss	0.864	1.468	0.788
PostgreSQL	0.559	4.877	0.332

可以发现，1,3 两组以整数为关键字的排序 openGauss 的运行时间都比 PostgreSQL 长，但第 2 组以字符串为关键字的排序 openGauss 的运行时间只有 PostgreSQL 的三分之一。猜测原因是 PostgreSQL 是直接比较字符串进行复杂度为 $O(n \log n)$ 的排序，而 openGauss 可能通过代价分析，执行了基于小字符集的复杂度更低的字符串排序算法。

总结

可以发现，在绝大部分测试中，PostgreSQL 都有更好的运行效率，以及更好的稳定性。

特别地，在以字符串为关键字的排序中，openGauss 的运行效率是 PostgreSQL 的三倍。结合 openGauss 的项目文档，可认为是 openGauss 的 AI 引擎做了查询路径上的优化。

总的而言，以轻量使用 SQL 进行查询的视角而言，PostgreSQL 具有更快的运行效率和稳定性。openGauss 虽然在一些局部能体现出 AI 引擎的效率提升，但整体效率不如 PostgreSQL，劣势十分明显。

在下一节中，我会基于 pgbench，测试 openGauss 和 PostgreSQL 在高并发情况下的表现。

第三节：高并发条件下的测试

在第三节中，我将使用 pgbench，对 openGauss 和 PostgreSQL 进行高并发条件下的压力测试。

测试基本信息

pgbench默认的测试脚本使用 tid 作为列名，同时，openGauss 也使用了 tid 作为关键字。这导致了 pgbench 的默认测试无法在 openGauss 中运行，笔者改写了 pgbench 的默认测试，使之能正常运行。

初始化

初始化阶段运行pg_init.sql，创建四个表：支行表（pgbench_branches），出纳表（pgbench_tellers），账户表（pgbench_accounts）,历史信息表（pgbench_history），并填充初始数据（账户表中 10^6 条数据，出纳表 100 条数据，支行表 10 条数据）。各表包含信息如下：

支行表：

```
CREATE TABLE pgbench_branches (
    bid integer NOT NULL,
        --支行的唯一标识符（主键）。
    bbalance bigint,
        --支行的余额。
    filler character(88)
        --用于填充数据，以便在测试中模拟更多的磁盘I/O操作。
);
```

出纳表：

```
CREATE TABLE pgbench_tellers (
    tid0 integer NOT NULL,
        --出纳员的唯一标识符（主键）。
    bid integer,
        --出纳员所属的支行标识符。
    tbalance bigint,
        --出纳员的余额。
    filler character(84)
        --用于填充数据，以便在测试中模拟更多的磁盘I/O操作。
);
```

账户表：

```
CREATE TABLE pgbench_accounts (
    aid integer NOT NULL,
        --账户的唯一标识符（主键）。
    bid integer,
        --账户所属的支行标识符。
    abalance bigint,
        --账户的余额。
    filler character(84)
        --用于填充数据，以便在测试中模拟更多的磁盘I/O操作。
);
```

历史信息表：

```
CREATE TABLE pgbench_history (
    tid0 integer,
        --出纳员的唯一标识符。
    bid integer,
        --支行的唯一标识符。
    aid integer,
        --账户的唯一标识符。
    delta integer,
        --交易金额的变化。
    mtime timestamp without time zone default CURRENT_TIMESTAMP
        --交易的时间戳。
);
```

压力测试脚本

压力测试阶段通过 pgbench 多次运行 pg_test.sql，每次执行 pg_test.sql，都会随机选择一个支行（bid），一个出纳员（tid0），一个账户（aid），和金额变化量（delta）。然后更新被选择的支行、出纳员、账户的余额，将这次交易的信息存入历史信息表（pgbench_history）中。

压力测试参数

压力测试阶段通过 pgbench 多次运行 pg_test.sql，固定每次测试时间为 60 秒，改变并发客户端数量，分配线程数等于客户端数量，多次实验，记录平均响应时间和每秒事务数（TPS）。

测试结果

openGauss：

并发客户端数量	1	2	3	4	5	6	7	8	9	10
TPS	3.87	4.78	8.35	9.83	11.88	12.54	13.83	13.61	14.89	14.84
平均响应时间（毫秒）	258	418	359	407	421	479	506	587	603	671
连接时间（毫秒）	4.6	25.4	27.6	26.5	27.6	8.5	26.5	27.9	68.1	28.8

并发客户端数量	5	10	15	20	25	30	35	40
TPS	11.88	14.84	14.58	13.81	13.47	13.61	13.46	13.60
平均响应时间（毫秒）	421	671	1025	1441	1842	2190	2579	2914
连接时间（毫秒）	27.6	28.8	22.8	28.2	28.6	69.7	36.5	46.7

PostgreSQL：

并发客户端数量	1	2	3	4	5	6	7	8	9	10
TPS	6.43	10.02	12.3	14.3	17.48	20.37	21.73	24.23	25.07	25.27
平均响应时间（毫秒）	155	199	244	279	286	294.	322	330	358	395
连接时间（毫秒）	56.5	188	106	157	266	183	290	237	337	272

并发客户端数量	5	10	15	20	25	30	35	40
TPS	17.48	25.27	28.07	31.05	30.05	30.94	30.63	31.67
平均响应时间（毫秒）	286	395	534	643	829	966	1139	1258
连接时间（毫秒）	266	272	472	573	742	829	982	1064

结果分析

吞吐量（TPS）

在不同数量的并发客户端的情况下，PostgreSQL 的 TPS 都高于 openGauss。无论在哪个数据库中，随着并发客户端数量的增长，TPS 的变化都呈现先增长，再平稳波动的的趋势。对于 openGauss，增长与平稳的转折点在 10 左右；对于 PostgreSQL，增长与平稳的转折点在 20 左右，可见 PostgreSQL 在高并发的情况下表现更好。

平均连接时间

在不同数量的并发客户端的情况下，PostgreSQL 的平均连接时间都短于 openGauss。无论在哪个数据库中，随着并发客户端数量的增长，平均连接时间的变化都呈现持续增长趋势。相较而言，PostgreSQL 的增长速度也慢于openGauss，可见 PostgreSQL 在高并发的情况下表现更好。

初始连接时间

在不同数量的并发客户端的情况下，PostgreSQL 的初始连接时间都长于 openGauss。无论在哪个数据库中，随着并发客户端数量的增长，初始连接时间的变化都呈现持续增长趋势。相较而言，openGauss 的增长速度也慢于PostgreSQL。在这一点上，openGauss 的表现要更好。

第四节：安全性和易用性的简单对比

受限于本地的测试环境，无法模拟对云端数据信息的窃取与攻击。同时，安全性和易用性缺乏量化的标准，故本节仅以笔者在完成项目过程中对二者的使用体验，主观评价 openGauss 的安全性和易用性。

安全性

在 openGauss 创建数据库的过程中，会发现 openGauss 要求设定的密码至少要八位，并强制要求包含数字，大小写字母和特殊字符。在设定密码方面，PostgreSQL 则没有这些附加要求。在这一细节上，可以发现 openGauss 对安全性有更多的保护。

易用性

openGauss 是基于 PostgreSQL 开发的，支持 PostgreSQL 支持的全套语法。这在使用中带了极大方便。编写完成的 sql 脚本，可以不加修改的在 openGauss 上运行。

与此同时，openGauss 在基于 PostgreSQL 做其他优化的过程中，也引入了一些破坏易用性的修改。比如，将 tid 这种常用名占用为关键字，在使用中造成不便。此外，强制密码中包含特殊字符也造成了一些不便：并非所有特殊字符都能充当密码而不被解析成其他含义。

第五部分：总结与改进

在本项目中，主要比较了 openGauss 和 PostgreSQL 在性能指标下的差异，具体分成了简单SQL查询，和高并发条件下的压力测试。简单SQL查询中，插入，修改，大部分查询都是PostgreSQL更快，只在以字符串为关键字的排序中openGauss有更好的表现。高并发条件下的压力测试中，每秒事务数和平均响应时间都是PostgreSQL更优，在初始连接时间上openGauss有更好的表现。

就测试结果而言，openGauss 在大部分场景下的表现都不如 PostgreSQL 好，其声称的高并发场景下的优化并未被实验数据观测到。在少部分测试中，能看到 openGauss 做出的优化。整体而言，openGauss 对比 PostgreSQL 几乎没有优势，在个别小点上的优化无法掩盖整体性能上的降低。

需要指出的是，该结果只适用于笔者个人计算机进行的若干测试。实际上，根据 openGauss 的技术架构说明，openGauss 的主要优化方向是面向更大规模数据、拥有更高并发的服务器群、需要更长时间维护的企业级数据库。也就是说，笔者在个人计算机上模拟的情况并非 openGauss 的优化方向，这次的对比是对 openGauss 不够公平的。