

CS213 project3 报告

12313202 曾子华

github仓库地址: [sgweo8ys/CS213project3](https://github.com/sgweo8ys/CS213project3)

CS213 project3 报告

数据库的评估标准

openGauss 与 PostgreSQL 的比较

环境准备

安装 opengauss

安装 postgresql

少量数据下的性能比较

数据生成

性能比较

大量数据下的性能比较

数据生成

性能比较

使用 Benchmarksql 进行 TPC-C 测试

参数设置

测试结果

测试两种数据库的硬件使用情况

实验结果

总结

数据库的评估标准

用于评估数据库的标准有很多, 如性能, 可拓展性, 安全性, 易用性和用户社区等。

其中最重要的是数据库的性能, 本次 project 重点比较了两种数据库的性能差异。

首先使用对数据库命令进行记时统计的方法, 得到两种数据库在运行数据库命令时的速度比值。

之后使用 benchmarksql 对两个数据库进行 TPC-C 压力测试, 测试两种数据库在处理复杂事务时的效率差距。

最后使用华为云监测功能, 测试两种数据库在运行复杂命令时的硬件使用情况。

openGauss 与 PostgreSQL 的比较

环境准备

使用华为云服务器完成本次 project, 型号与下发文档一致。

安装 opengauss

按照下发文档的过程, 安装 OpenGauss 5.0.1 版本至服务器目录 `/opt/software/openGauss`

安装 postgresql

1. 安装依赖包

```
yum install -y perl-ExtUtils-Embed readline-devel zlib-devel pam-devel  
libxml2-devel libxslt-devel openldap-devel python-devel gcc-c++ openssl-devel  
cmake
```

2. 类似安装 OpenGauss 的过程，在 /opt/software 目录下新建目录 pgsql，并从官网下载压缩包并解压

```
wget https://ftp.postgresql.org/pub/source/v12.4/postgresql-12.4.tar.gz  
tar -xvf postgresql-12.4.tar.gz
```

3. 进入解压后的文件夹，编译源码，安装

```
cd postgresql-12.4  
./configure --prefix=/opt/software/pgsql/postgresql  
make && make install
```

4. 创建 postgres 用户操作 postgresql

```
groupadd postgres  
useradd -g postgres postgres  
mkdir /opt/software/pgsql/postgresql/data  
chown postgres:postgres /opt/software/pgsql/postgresql/data
```

5. 将 postgresql 放入环境变量

```
vim .bash_profile
```

加入

```
export PGHOME=/opt/software/pgsql/postgresql  
export PGDATA=/opt/software/pgsql/postgresql/data  
PATH=$PATH:$HOME/bin:$PGHOME/bin
```

```
source .bash_profile
```

少量数据下的性能比较

使用一张大小为 10^5 的表，对比 postgresql 和 opengauss 运行数据库增删改查命令的时间。

数据生成

在服务器上使用 C++ 程序随机生成一张大小为 10^5 的表，包含以下属性：

| 名称 | 类型 |
|----|---------|
| id | integer |

| 名称 | 类型 |
|-------|--------|
| name | string |
| value | int |

并将其输出到 `/opt/test/tu1.csv` 中:

```
#include <stdio>
#include <ctime>
#include <stdlib>
int main()
{
    srand((int)time(0));
    freopen("tu1.csv", "w", stdout);
    puts("id,name,value");
    int n = 100000;
    for(int i = 1; i <= n; i++){
        char s[30];
        for(int j = 0; j < 20; j++) s[j] = rand() % 26 + 'a';
        printf("%d,", i);
        for(int j = 0; j < 20; j++) printf("%c",s[j]);
        printf(",%d\n",rand() % 300);
    }
    return 0;
}
```

性能比较

将数据分别导入 postgresql 和 opengauss 后，运行数据库命令。

```
create table test(id int, name varchar(30), value int);
copy test(id, name, value) from '/opt/test/tu1.csv' DELIMITER ',' CSV HEADER;
```

使用 `explain analyse` 命令统计在两种不同数据库下命令的运行时间，单位均为毫秒。

1. 查询命令/精确查询

测试命令1:

```
explain analyse select * from test where id = 1;
```

| 数据库/运行次数编号 | 1 | 2 | 3 | 平均用时 |
|------------|--------|--------|--------|--------|
| postgresql | 7.69 | 7.387 | 7.389 | 7.489 |
| opengauss | 26.211 | 16.607 | 16.561 | 19.793 |

速度比值: 2.64

测试命令2:

```
explain analyse select * from test where id = 67364;
```

| 数据库/运行次数编号 | 1 | 2 | 3 | 平均用时 |
|------------|--------|--------|--------|--------|
| postgresql | 7.374 | 7.436 | 7.377 | 7.396 |
| opengauss | 16.721 | 16.518 | 16.428 | 16.556 |

速度比值: 2.24

测试命令3:

```
explain analyse select * from test where name = 'okvqibkayswynkfbkprk';
```

| 数据库/运行次数编号 | 1 | 2 | 3 | 平均用时 |
|------------|--------|--------|--------|--------|
| postgresql | 11.553 | 11.438 | 11.441 | 11.477 |
| opengauss | 20.663 | 20.75 | 20.566 | 20.660 |

速度比值: 1.80

2. 查询命令/其他查询

测试命令1:

```
explain analyse select * from test where id > 46736 and id < 46759;
```

| 数据库/运行次数编号 | 1 | 2 | 3 | 平均用时 |
|------------|--------|--------|--------|--------|
| postgresql | 8.125 | 8.117 | 8.088 | 8.110 |
| opengauss | 19.166 | 19.213 | 19.427 | 19.269 |

速度比值: 2.38

测试命令2:

```
explain analyse select * from test where name like '%abc%';
```

| 数据库/运行次数编号 | 1 | 2 | 3 | 平均用时 |
|------------|--------|--------|--------|--------|
| postgresql | 19.156 | 19.198 | 19.078 | 19.144 |
| opengauss | 27.486 | 27.457 | 27.603 | 27.515 |

速度比值: 1.44

3. 更新命令

```
begin;  
explain analyse update test set name = 'hello' where id = 4762;  
rollback;
```

| 数据库/运行次数编号 | 1 | 2 | 3 | 平均用时 |
|------------|--------|-------|--------|--------|
| postgresql | 7.392 | 7.387 | 7.358 | 7.379 |
| opengauss | 17.597 | 16.46 | 16.581 | 16.879 |

速度比值: 2.29

4. 单行插入命令

```
begin;  
explain analyse insert into test values(100001, 'hello world', 176);  
rollback;
```

| 数据库/运行次数编号 | 1 | 2 | 3 | 平均用时 |
|------------|-------|-------|-------|-------|
| postgresql | 0.057 | 0.057 | 0.054 | 0.056 |
| opengauss | 0.126 | 0.116 | 0.133 | 0.125 |

速度比值: 2.23

5. 删除命令

```
begin;  
explain analyse delete from test where value < 100;  
rollback;
```

| 数据库/运行次数编号 | 1 | 2 | 3 | 平均用时 |
|------------|--------|--------|--------|--------|
| postgresql | 38.735 | 33.464 | 34.043 | 35.414 |
| opengauss | 77.596 | 78.14 | 84.287 | 80.008 |

速度比值: 2.26

结论: 少量数据下的各种命令, postgresql 均明显快于 opengauss, 二者的速度比值在 1.44 到 2.64 之间。

大量数据下的性能比较

使用一张大小为 10^7 的表, 对比 postgresql 和 opengauss 运行数据库增删改查命令的时间。

使用三张大小为 10^5 的表, 使用 join 合并得到一张大小约为 3×10^6 的表, 对比 postgresql 和 opengauss 运行数据库查询命令的时间。

数据生成

使用与少量数据类似的方法，生成列一样但行数为 10^7 的表，使用类似的方法导入到两个数据库的表 test 中。

用 C++ 程序生成用于测试 join 的数据，并将其输出到 csv 文件中，下面展示第一个数据的生成代码：

```
// joindata1.cpp
#include <stdio>
#include <ctime>
#include <stdlib>
int main()
{
    srand((int)time(0));
    freopen("jointest1.csv", "w", stdout);
    int n = 100000;
    puts("id1,name,grp");
    for(int i = 1; i <= n; i++){
        char s[30];
        for(int j = 0; j < 20; j++) s[j] = rand() % 26 + 'a';
        printf("%d,", i);
        for(int j = 0; j < 20; j++) printf("%c",s[j]);
        printf(",%d\n", i / 3);
    }
    return 0;
}
```

这样生成的表具有 id, name, grp 三个属性，其中 grp 每相邻三行相同，以保证 join 得到的表不会太大。

性能比较

同样的，将数据分别导入 postgresql 和 opengauss 后，运行数据库命令。

```
delete from test;
copy test(id, name, value) from '/opt/test/tu1.csv' DELIMITER ',' CSV HEADER;
```

使用 explain analyse 命令统计在两种不同数据库下命令的运行时间，单位均为毫秒。

1. 精确查询命令

```
explain analyse select * from test where id = 3987654;
```

| 数据库/运行次数编号 | 1 | 2 | 3 | 平均用时 |
|------------|----------|----------|---------|----------|
| postgresql | 424.025 | 427.914 | 424.186 | 425.375 |
| opengauss | 2088.824 | 2085.112 | 2082.3 | 2085.412 |

速度比值：4.90

2. 更新命令

```
begin;
explain analyse update test set name = 'hello' where value = 101;
rollback;
```

| 数据库/运行次数编号 | 1 | 2 | 3 | 平均用时 |
|------------|----------|----------|---------|----------|
| postgresql | 424.025 | 427.914 | 424.186 | 425.375 |
| opengauss | 2088.824 | 2085.112 | 2082.3 | 2085.412 |

速度比值: 15.35

3. 删除命令

```
begin;
explain analyse delete from test where value < 15;
rollback;
```

| 数据库/运行次数编号 | 1 | 2 | 3 | 平均用时 |
|------------|----------|----------|----------|----------|
| postgresql | 7277.684 | 3678.181 | 4053.929 | 5003.265 |
| opengauss | 41663.23 | 41112.84 | 42309.1 | 41695.06 |

速度比值: 8.33

4. 单条多行插入命令

使用 C++ 程序生成一条进行 1000 行插入的命令:

```
#include <cstdio>
int main()
{
    freopen("insertsql.out", "w", stdout);
    int n = 1000;
    printf("begin;\n"
           "explain analyse insert into test values\n");
    for(int i = 1; i <= n; i++){
        printf("(%d,'cstowonethree',%d)", i, i % 300);
        if(i < n) printf(",\n");
        else printf(";\n");
    }
    printf("rollback;\n");
}
```

比较二者的运行时间:

| 数据库/运行次数编号 | 1 | 2 | 3 | 平均用时 |
|------------|-------|-------|-------|-------|
| postgresql | 3.291 | 2.632 | 2.565 | 2.829 |
| opengauss | 2.786 | 2.873 | 2.87 | 2.843 |

速度比值: 1.00

5. 循环插入命令

由于循环无法使用 `explain analyse` 命令, 故使用 `\timing` 进行计时。

```
begin;
\timing on
do $$
    begin
        for i in 1..100000 loop
            insert into test values(i, 'hello', i % 300);
        end loop;
    end;
$$;
\timing off
rollback;
```

| 数据库/运行次数编号 | 1 | 2 | 3 | 平均用时 |
|------------|----------|----------|----------|----------|
| postgresql | 396.13 | 396.301 | 396.309 | 396.2467 |
| opengauss | 2782.759 | 2846.032 | 2730.429 | 2786.407 |

速度比值: 7.03

6. 带索引的精确查询:

```
create index ind1 on test(id);
```

```
explain analyse select * from test where id = 3987655;
```

| 数据库/运行次数编号 | 1 | 2 | 3 | 平均用时 |
|------------|-------|-------|-------|-------|
| postgresql | 0.09 | 0.092 | 0.104 | 0.095 |
| opengauss | 0.074 | 0.071 | 0.071 | 0.072 |

速度比值: 0.76

7. join 查询:


```
create table jointable1(id1 int, name varchar(30), grp int);
copy jointable1(id1, name, grp) from '/opt/test/jointest1.csv' DELIMITER ','
CSV HEADER;

create table jointable2(id2 int, name varchar(30), grp int);
copy jointable2(id2, name, grp) from '/opt/test/jointest2.csv' DELIMITER ','
CSV HEADER;

create table jointable3(id3 int, name varchar(30), grp int);
copy jointable3(id3, name, grp) from '/opt/test/jointest3.csv' DELIMITER ','
CSV HEADER;
```

```
explain analyse select * from jointable1
  join jointable2 on jointable2.grp = jointable1.grp
  join jointable3 on jointable3.grp = jointable2.grp
where id1 + id2 + id3 = 8297;
```

| 数据库/运行次数编号 | 1 | 2 | 3 | 平均用时 |
|------------|---------|---------|---------|----------|
| postgresql | 341.229 | 342.213 | 339.905 | 341.1157 |
| opengauss | 540.417 | 503.93 | 501.569 | 515.3053 |

速度比值: 1.51

结论: 大量数据下的情况较为复杂, 整体表现为大部分命令 postgresql 远远快于 opengauss, 少部分命令二者相差不大或者 opengauss 略快于 postgresql。

大量数据下的精确查询, 修改和删除命令, 以及使用循环完成的大量插入命令, postgresql 远远快于 opengauss, 二者的速度比值在 4.9 到 15.35 之间。

多表 join 后查询的命令, postgresql 略快于 opengauss, 单条多行插入命令下二者速度相差无几。速度比值在 1 到 1.44 之间

建立索引后的精确查询命令, opengauss 略快于 potgresql, 二者的速度比值为 0.76。

使用 Benchmarksq1 进行 TPC-C 测试

使用 benchmarksq1 分别对 opengauss 和 postgresql 进行测试。

参数设置

使用的 benchmarksq1 版本为 5.0.1。

在安装后的 run 文件下配置测试 postgresql 的 props.pg 文件。

设置仓库数 warehouse 为 10, 其中每个仓库的大小大约为 76823.04KB。

设置终端数量 terimnals 为 50, 这也是并发客户数量, 可以测试数据库在高并发情况下的表现。

设置运行时间 runMins 为 5, 测试 5 分钟, 统计两种数据库处理的事务数量和处理速度。

测试结果

运行测试，测试完成之后生成报告。

postgresql 测试结果：

| | |
|-------------------|----------|
| Overall tpmC: | 12105.80 |
| Overall tpmTotal: | 26877.20 |

opengauss 测试结果：

| | |
|-------------------|---------|
| Overall tpmC: | 2922.00 |
| Overall tpmTotal: | 6457.00 |

结论：在 TPC-C 测试中，postgresql 的速度显著快于 opengauss，二者的 TPC-C 比值为 4.14。

从图上可以看出，opengauss 处理速度较 postgresql 更加稳定。

测试两种数据库的硬件使用情况

使用华为云监测功能，测试两种数据库的硬件使用情况。

分别用两种数据库运行以下查询命令：

```
explain analyse
select id, name, value, id * value, id % (value + 1) from test
where log(id) * value < 1000;
```

之后使用华为云监控功能，取两种数据库在运行时间内的 CPU 使用率峰值和磁盘读写带宽峰值。

实验结果

postgresql:

| 情况/项目 | CPU使用率 | 磁盘读带宽 | 磁盘写带宽 |
|-------|--------|------------|-------------|
| 对照值 | 3% | 0 Mib/s | 0 Mib/s |
| 运行时峰值 | 3.99% | 2.059Mib/s | 0.056 Mib/s |

opengauss:

| 情况/项目 | CPU使用率 | 磁盘读带宽 | 磁盘写带宽 |
|-------|--------|------------|-------------|
| 对照值 | 3% | 0 Mib/s | 0 Mib/s |
| 运行时峰值 | 8.68% | 0.417Mib/s | 1.479 Mib/s |

postgresql 的 CPU 使用率较 opengauss 更低，磁盘读带宽较 opengauss 更高，写带宽更低。

总结

经过比较，可以得出结论：

1. 少量数据下，postgresql 的各种命令运行速度显著快于 opengauss。
2. 大量数据下，绝大部分命令 postgresql 运行速度远远快于 opengauss。但是在建立索引之后的查询下 opengauss 略快于 postgresql。
3. 在 TPC-C 测试中，postgresql 在高并发操作下处理事务效率远快于 opengauss，但是 opengauss 处理效率更加稳定。
4. 运行查询命令时，postgresql 的 CPU 占用率较 opengauss 更低，更能节约硬件资源。

总的来说，postgresql 相较 opengauss 在性能上仍有较大优势，opengauss 在 TPC-C 测试中表现更加稳定，postgresql 的硬件占用率更低，读带宽更高，写带宽更低。