PG benchmark

如何压测、瞬间构造大量测试数据

阿里云

digoal

目录

- 压测工具
- 瞬间构造大量测试数据
- 压测case

压测工具

- pgbench
  - https://www.postgresql.org/docs/current/pgbench.html
- 参数
  - 连接数、线程数、报告、绑定变量、重置连接等选项
- 内置模型
  - 初始化、sql
- 自定义模型方法
  - 变量
  - 变量传递
  - 随机数
  - 睡眠
  - 多脚本调用和权重

尽快生成1000万行数据 – 方法1

```
create table test (id int primary key , c1 int, c2 int, c3 int, info text, crt_time timestamp);
create sequence seq cache 1000;

vi test.sql
\set c1 random(1,10000)
\set c2 random(1,1000)
\set c3 random(1,100)
insert into test values (nextval('seq'::regclass), :c1, :c2, :c3, md5(random()::text), clock_timestamp());

pgbench -M prepared -n -r -P 1 -f ./test.sql -c 24 -j 24 -T 120
```

尽快生成1000万行数据 – 方法2

create table test (id int primary key , c1 int, c2 int, c3 int, info text, crt_time timestamp);

create sequence seq cache 1000;

vi test.sql

insert into test select nextval('seq'::regclass), random()*10000, random()*1000, random()*100, md5(random()::text), clock_timestamp() from generate_series(1,100);

. ./env.sh

pgbench -M prepared -n -r -P 1 -f ./test.sql -c 8 -j 8 -T 120

尽快生成1000万行数据 – 方法3

```
create table test (id int primary key , c1 int, c2 int, c3 int, info text, crt_time timestamp);

create sequence seq cache 1000;


insert into test select nextval('seq'::regclass), random()*10000, random()*1000, random()*100, md5(random()::text), clock_timestamp() from generate_series(1,10000000);
```

瞬间构造大量测试数据

- https://github.com/digoal/blog/blob/master/201711/20171121_01.md

压测case

- tpcb
- 简单key查询
- upsert, update
- 随机数变量
- 模糊查询(模拟交互)
  - 模拟交互(gset)
- 查询轨迹历史（包括优化方法）
  - 模拟睡眠(\sleep x ms|us|s)
- 数值相近排序
- 空间距离相近排序
- 为什么高并发要用长连接、绑定变量？

```
create table test (id int primary key , c1 int, c2 int, c3 int, info text, crt_time timestamp);
create sequence seq cache 1000;

vi test.sql
\set c1 random(1,10000)
\set c2 random(1,1000)
\set c3 random(1,100)
insert into test values (nextval('seq'::regclass), :c1, :c2, :c3, md5(random()::text), clock_timestamp());


pgbench -M prepared -n -r -P 1 -f ./test.sql -c 52 -j 52 -T 120
```

```
pgbench -i -s 5000
pgbench -M prepared -n -r -P 1 -c 104 -j 104 -T 120 -S
pgbench -M prepared -n -r -P 1 -c 52 -j 52 -T 120 -b simple-update
```

```
create table a(id int primary key, info text, crt_time timestamp);


vi test.sql

\set id random(1,2000000000)

insert into a(id,info,crt_time) values (:id, md5(random()::text), now()) on conflict(id) do update set info=excluded.info,crt_time=excluded.crt_time;


pgbench -M prepared -n -r -P 1 -f ./test.sql -c 52 -j 52 -T 120
```

i 取值范围 range[x,y]

random(x,y)

i概率随机

random_exponential(min,max,p)

f(x) = exp(-parameter * (x - min) / (max - min + 1)) / (1 - exp(-parameter))

x的概率：f(x) - f(x + 1)

random_gaussian(x,y,p)

f(x) = PHI(2.0 * parameter * (x - mu) / (max - min + 1)) /

(2.0 * PHI(parameter) - 1)

i的概率：f(i + 0.5) - f(i - 0.5)

random_zipfian(x,y,p)

i的概率：((i+1)/i)**parameter

Gin倒排索引（模糊查询）

```
create table t1 (id int, info text, crt_time timestamp);
create table t2 (id int primary key, info text);

insert into t1 select generate_series(1,10000000), md5(random()::text), clock_timestamp();
insert into t2 select generate_series(1,100000), md5(random()::text);
insert into t2 select id,substring(info,5,10) from t1 where id>=100001 and id<=1000000;
create extension pg_trgm;
create index idx_t1_info on t1 using gin (info gin_trgm_ops);

vi test.sql
\set id random(1,1000000)
select info as v_info from t2 where id=:id \gset
select * from t1 where info like '%'||:v_info||'%';

pgbench -M prepared -n -r -P 1 -f ./test.sql -c 104 -j 104 -T 120

select * from t1 where info like '%'||'abcdef'||'%';
select * from t1 where info like '%abcdef%';
```

```
create table t_pos (ordid int, pos point, crt_time timestamp, c1 int, c2 int, c3 int);

insert into t_pos select random()*10000, point(random()*180, random()*90), clock_timestamp(), random()*100, random()*1000, random()*10000 from generate_series(1,10000000);


create index idx_t_pos on t_pos (ordid);


vi test.sql

\set ordid random(1,10000)

select * from t_pos where ordid=:ordid;


pgbench -M prepared -n -r -P 1 -f ./test.sql -c 104 -j 104 -T 120


drop index idx_t_pos;

create index idx_t_pos on t_pos (ordid) include (pos,crt_time,c1,c2,c3);

vacuum analyze t_pos;

explain select * from t_pos where ordid=1;


pgbench -M prepared -n -r -P 1 -f ./test.sql -c 104 -j 104 -T 120
```

```
create table t_num (id int, age float8, info text, crt_time timestamp);
insert into t_num select generate_series(1,10000000), random()*120, md5(random()::text), clock_timestamp();
create extension btree_gist;
create index idx_t_num on t_num using gist (age);
select * from t_num order by age <-> 24 limit 10;

vi test.sql
\set age random(1,120)
select * from t_num order by age <-> :age limit 1;

pgbench -M prepared -n -r -P 1 -f ./test.sql -c 52 -j 52 -T 120
```

```
create index idx_t_pos2 on t_pos using gist (pos);

vi test.sql
\set x random(1,180)
\set y random(1,90)
select * from t_pos order by pos <-> point(:x, :y) limit 1;

pgbench -M prepared -n -r -P 1 -f ./test.sql -c 104 -j 104 -T 120
```

每次新建连接

pgbench -M simple -n -r -P 1 -c 104 -j 104 -T 120 -S -C

长连接、绑定变量

pgbench -M prepared -n -r -P 1 -c 104 -j 104 -T 120 -S

长连接、simple query

pgbench -M simple -n -r -P 1 -c 104 -j 104 -T 120 -S

一期开课计划(PG+MySQL联合方案)

- - 2019.12.30 19:30 RDS PG产品概览，如何与MySQL结合使用
- - 2019.12.31 19:30 如何连接PG，GUI，CLI的使用
- - 2020.1.3 19:30 如何压测PG数据库、如何瞬间构造海量测试数据
- - 2020.1.6 19:30 MySQL与PG类型、语法、函数等对应关系
- - 2020.1.7 19:30 如何将MySQL数据同步到PG（dts）
- - 2020.1.8 19:30 PG外部表妙用 - mysql_fdw, oss_fdw（直接读写MySQL数据、冷热分离）
- - 2020.1.9 19:30 PG应用场景介绍 - 并行计算，实时分析
- - 2020.1.10 19:30 PG应用场景介绍 - GIS
- - 2020.1.13 19:30 PG应用场景介绍 - 用户画像、实时营销系统
- - 2020.1.14 19:30 PG应用场景介绍 - 多维搜索
- - 2020.1.15 19:30 PG应用场景介绍 - 向量计算、图像搜索
- - 2020.1.16 19:30 PG应用场景介绍 - 全文检索、模糊查询
- - 2020.1.17 19:30 pg 数据分析语法介绍
- - 2020.1.18 19:30 pg 更多功能了解：扩展语法、索引、类型、存储过程与函数。如何加入PG技术社群

技术社群

PG技术交流钉钉群(3500+人)