新手机 新应用 新娱乐

PostgreSQL 9.3 培训
Day 3

digoal.zhou

2013/12/11

SKY mobi
新手机 新应用 新娱乐

课程内容

- Day - 3
- 数据分区
- 目标:
- 了解本地表分区的方法和具体实现, 了解分区表的DML操作
- 了解异地分区的方法, 挑选PL/Proxy分区为例进行介绍
- 了解异地数据合并和数据拆分的方法

- 事务处理与并发控制
- 目标:
- 事务, 并发控制, 锁的介绍, 死锁的发现和处理, 实际应用中如何避免死锁

- 监控数据库活动
- 目标:
- 了解各种维度的统计信息的解读(table, index, database, replication, sql...), 锁, 磁盘使用, 活动信息, 历史数据库活动统计报告

- PostgreSQL日常维护和检查
- 目标:
- 日志记录类型, 日志的处理, 表的维护, 数据的维护.

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 数据分区

- 了解本地表分区的方法和具体实现, 了解分区表的DML操作
- 了解异地分区的方法, 挑选PL/Proxy分区为例进行介绍
- 了解异地数据合并和数据拆分的方法

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 本地分区

- PostgreSQL通过继承表的概念来实现数据分区表的查询,更新,删除,插入数据的逻辑.
- 继承表和主表以及继承表之间是完全独立的存储结构, 可以单独指定表空间, 甚至允许字段不完全一致.

- 数据分区的好处

- 分区后, 单个分区表的索引和表都变小了, 可以保持在内存里面, 适合把热数据从大表拆分出来的场景.
- 对于大范围的查询, 大表可以通过索引来避免全表扫描. 但是如果分区了的话, 可以使用分区的全表扫描. 适合经常要做大范围扫描的场景, 按照范围分区(分区后采用全表扫描), 减少索引带来的随机BLOCK扫描.
- 大批量的数据导入或删除, 对于大表来说, 删除大量的数据使用DELETE的话会带来大量的VACUUM操作负担.而使用分区表的话可以直接DROP分区, 或者脱离子表和父表的继承关系.
- 使用分区表, 还有一个好处是, 可以把不常用的分区放到便宜的存储上.
- 因为每个表只能放在一个表空间上, 表空间和目录对应, 表的大小受到表空间大小的限制. 使用分区表则更加灵活.

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 本地分区

- 数据库分区表举例
- 范围分区
- 根据字段存储的值取值范围进行分区, 例如日志表的时间字段, 用户表的ID范围等等.

- 哈希分区
- 根据字段存储值HASH再做和分区数做比特运算得到一个唯一的分区ID.
- 或者取模也行.
- 例如mod(hashtext(name),16), 对16个分区的场景.

- list分区
- 与哈希分区类似, 但是直接使用字段值作为分区条件. 适合KEY值比较少并且比较均匀的场景.
- 例如按性别字段作为分区字段. 那么就分成了2个区.

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 本地分区

- 分区和表继承的概念
- 继承表自动继承父表的约束, 非空约束. 但是不自动继承的是(uk,pk,fk,索引,存储参数等).
- 例如 :
- digoal=# create table p1(id int primary key, info text unique, c1 int check(c1>0), c2 int not null, c3 int unique);
- CREATE TABLE
- digoal=# create table c1(like p1) inherits(p1);
- NOTICE: merging column "id" with inherited definition
- NOTICE: merging column "info" with inherited definition
- NOTICE: merging column "c1" with inherited definition
- NOTICE: merging column "c2" with inherited definition
- NOTICE: merging column "c3" with inherited definition
- CREATE TABLE

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 本地分区

```
digoal=# \d+ p1
                    Table "postgres.p1"
 Column |  Type   | Modifiers | Storage  | Stats target | Description
--------+---------+-----------+----------+--------------+-------------
 id     | integer | not null  | plain    |              |
 info   | text    |           | extended |              |
 c1     | integer |           | plain    |              |
 c2     | integer | not null  | plain    |              |
 c3     | integer |           | plain    |              |
Indexes:
    "p1_pkey" PRIMARY KEY, btree (id)
    "p1_c3_key" UNIQUE CONSTRAINT, btree (c3)
    "p1_info_key" UNIQUE CONSTRAINT, btree (info)
Check constraints:
    "p1_c1_check" CHECK (c1 > 0)
Child tables: c1
Has OIDs: no
```

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 本地分区

- 子表自动继承了父表的非空约束, 自定义约束. 未自动继承PK和UK.
- digoal=# \d+ c1
- Table "postgres.c1"
- Column | Type | Modifiers | Storage | Stats target | Description
- --------+---------+-----------+----------+--------------+-------------
- id | integer | not null | plain | |
- info | text | | extended | |
- c1 | integer | | plain | |
- c2 | integer | not null | plain | |
- c3 | integer | | plain | |
- Check constraints:
- "p1_c1_check" CHECK (c1 > 0)
- Inherits: p1
- Has OIDs: no

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 本地分区

- 如果要继承UK,PK,索引,存储结构等, 创建时加including all;
- digoal=# drop table c1;
- digoal=# create table c1(like p1 including all) inherits(p1);
- digoal=# \d c1
- Table "postgres.c1"
- Column | Type | Modifiers
- --------+---------+-----------
- id | integer | not null
- info | text |
- c1 | integer |
- c2 | integer | not null
- c3 | integer |
- Indexes:
- "c1_pkey" PRIMARY KEY, btree (id)
- "c1_c3_key" UNIQUE CONSTRAINT, btree (c3)
- "c1_info_key" UNIQUE CONSTRAINT, btree (info)
- Check constraints:
- "p1_c1_check" CHECK (c1 > 0)
- Inherits: p1

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 本地分区

- 可以使用alter table解除和加入继承关系.
- digoal=# alter table c1 no inherit p1;
- ALTER TABLE
- digoal=# alter table c1 drop constraint p1_c1_check;
- ALTER TABLE

- 当主表和继承表的字段个数, 顺序, 名字, 以及类型或者约束条件有任何不一致时, 就无法自动添加继承.
- 这里指的是默认继承的约束, 非空约束等. 不包含不会默认继承的约束UK,PK,FK等.
- digoal=# alter table c1 inherit p1;
- ERROR: child table is missing constraint "p1_c1_check"

- 加上约束后就可以了
- digoal=# alter table c1 add constraint p1_c1_check check(c1>0);
- ALTER TABLE
- digoal=# alter table c1 inherit p1;
- ALTER TABLE

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 本地分区

- 一个表可以同时继承多个父表, 一个父表可以被多个子表继承.
- 但是必须注意, 一个表继承了多个主表的情况, 共有字段上, 所有的父表的约束包括not null的定义都必须继承过来. (同样不包扩pk, uk, fk等)

- 查主表默认情况下是会连带查询所有的子表的, 包括更深的子表(子表的子表).
- 例如select * from p1; 默认会查询所有子表和自身.
- digoal=# explain select * from p1;
- QUERY PLAN
- ---
- Append (cost=0.00..20.40 rows=1041 width=48)
- -> Seq Scan on p1 (cost=0.00..0.00 rows=1 width=48)
- -> Seq Scan on c1 (cost=0.00..20.40 rows=1040 width=48)
- (3 rows)

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 本地分区

- 除非使用only或者修改sql_inheritance
- digoal=# set sql_inheritance=false; -- false时不会自动在表后面加*
- SET
- digoal=# explain select * from p1;
- QUERY PLAN
- --
- Seq Scan on p1 (cost=0.00..0.00 rows=1 width=48)
- (1 row)

- digoal=# set sql_inheritance=true; -- true时自动在表后面加*, 使用only可以忽略*.
- SET
- digoal=# explain select * from only p1;
- QUERY PLAN
- --
- Seq Scan on p1 (cost=0.00..0.00 rows=1 width=48)
- (1 row)

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 本地分区

- set sql_inheritance=true;的目的是在表名后自动添加星号.
- Select * from p1*;
- 这样就会自动查所有的子表.
- 所以set sql_inheritance=false, 使用星号suffix就和set sql_inheritance=true的目的意义.

- digoal=# set sql_inheritance=false;
- SET
- digoal=# explain select * from p1*;
- QUERY PLAN
- ---
- Append (cost=0.00..20.40 rows=1041 width=48)
- -> Seq Scan on p1 (cost=0.00..0.00 rows=1 width=48)
- -> Seq Scan on c1 (cost=0.00..20.40 rows=1040 width=48)
- (3 rows)

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 本地分区

- SELECT, UPDATE, DELETE, TRUNCATE, DROP命令在主表上操作默认都会影响到子表.
- 这些操作必须注意了, 小心谨慎.
- INSERT和COPY命令, 这两条命令是只对当前表操作的, 不会扩展到子表.

- 结合分区规则优化DML
- 查询优化, 主要目标缩小表的扫描范围. 只需要扫描有效子表.
  - 通过约束和参数constraint_exclusion来优化.
  - The allowed values of constraint_exclusion are on (examine constraints for all tables), off (never examine constraints), and partition (examine constraints only for inheritance child tables and UNION ALL subqueries). partition is the default setting.
- 例如 :
- digoal=# show constraint_exclusion;
- ----------------------
- partition
- digoal=# create table p(id int, info text, crt_time timestamp);
- digoal=# create table c1(like p) inherits(p);
- digoal=# create table c2(like p) inherits(p);
- digoal=# create table c3(like p) inherits(p);
- digoal=# create table c4(like p) inherits(p);

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 本地分区

- digoal=# alter table c1 add constraint ck check (crt_time>='2013-01-01' and crt_time<'2013-02-01');
- digoal=# alter table c2 add constraint ck check (crt_time>='2013-02-01' and crt_time<'2013-03-01');
- digoal=# alter table c3 add constraint ck check (crt_time>='2013-03-01' and crt_time<'2013-04-01');
- digoal=# alter table c4 add constraint ck check (crt_time>='2013-04-01' and crt_time<'2013-05-01');
- digoal=# explain select * from p where crt_time='2013-01-01';
- QUERY PLAN
- ---
- Append (cost=0.00..23.75 rows=7 width=44)
- -> Seq Scan on p (cost=0.00..0.00 rows=1 width=44)
- Filter: (crt_time = '2013-01-01 00:00:00'::timestamp without time zone)
- -> Seq Scan on c1 (cost=0.00..23.75 rows=6 width=44)
- Filter: (crt_time = '2013-01-01 00:00:00'::timestamp without time zone)
- (5 rows)

- 这种方法同样适用UPDATE, DELETE.

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 本地分区

```
digoal=# set constraint_exclusion=off;
SET
digoal=# explain select * from p where crt_time='2013-01-01';
                                QUERY PLAN
-------------------------------------------------------------------------------
 Append  (cost=0.00..95.00 rows=25 width=44)
   ->  Seq Scan on p  (cost=0.00..0.00 rows=1 width=44)
         Filter: (crt_time = '2013-01-01 00:00:00'::timestamp without time zone)
   ->  Seq Scan on c1  (cost=0.00..23.75 rows=6 width=44)
         Filter: (crt_time = '2013-01-01 00:00:00'::timestamp without time zone)
   ->  Seq Scan on c2  (cost=0.00..23.75 rows=6 width=44)
         Filter: (crt_time = '2013-01-01 00:00:00'::timestamp without time zone)
   ->  Seq Scan on c3  (cost=0.00..23.75 rows=6 width=44)
         Filter: (crt_time = '2013-01-01 00:00:00'::timestamp without time zone)
   ->  Seq Scan on c4  (cost=0.00..23.75 rows=6 width=44)
         Filter: (crt_time = '2013-01-01 00:00:00'::timestamp without time zone)
(11 rows)
```

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 本地分区

- 更新, 删除优化, 优化目标同样是缩小扫描范围. 除了使用约束的方法, 还可以使用触发器或规则.
- 插入到指定分区的实现, 使用触发器.

```
create or replace function sel_tg() returns trigger as $$
declare
begin
  if NEW.crt_time >= '2013-01-01' and NEW.crt_time < '2013-02-01' then
    insert into c1(id,info,crt_time) values (NEW.*);
  elsif NEW.crt_time >= '2013-02-01' and NEW.crt_time < '2013-03-01' then
    insert into c2(id,info,crt_time) values (NEW.*);
  elsif NEW.crt_time >= '2013-03-01' and NEW.crt_time < '2013-04-01' then
    insert into c3(id,info,crt_time) values (NEW.*);
  elsif NEW.crt_time >= '2013-04-01' and NEW.crt_time < '2013-05-01' then
    insert into c4(id,info,crt_time) values (NEW.*);
  else
    raise exception 'crt_time overflow.';
  end if;
  return null;
end;
$$ language plpgsql strict;
```

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 本地分区

- digoal=# create trigger tg1 before insert on p for each row execute procedure sel_tg();
- CREATE TRIGGER
- digoal=# insert into p values (1,'test',now());
- ERROR: crt_time overflow.

- digoal=# insert into p values (1,'test','2013-03-01 10:00:00');
- INSERT 0 0
- digoal=# insert into p values (1,'test','2013-03-01 10:00:00');
- INSERT 0 0

- digoal=# select * from only p;
- id | info | crt_time
- ----+------+----------
- (0 rows)

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 本地分区

- digoal=# select * from p;
- id | info | crt_time
- ----+------+---------------------
- 1 | test | 2013-03-01 10:00:00
- 1 | test | 2013-03-01 10:00:00
- (2 rows)
- 已插入子表
- digoal=# select * from c3;
- id | info | crt_time
- ----+------+---------------------
- 1 | test | 2013-03-01 10:00:00
- 1 | test | 2013-03-01 10:00:00
- (2 rows)

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 本地分区

- 继承表极大的方便了分区表的管理 (如字段的维护, 分区的隔离等.)
- 在对主表增加字段时, 子表自动增加字段. 通过这种方式增加的子表字段是继承字段. (pg_attribute.attislocal = false)

- digoal=# alter table p add column username name;
- ALTER TABLE
- digoal=# \d p
- Table "public.p"
- Column | Type | Modifiers
- ----------+-----------------------------+-----------
- id | integer |
- info | text |
- crt_time | timestamp without time zone |
- username | name |
- Triggers:
- tg1 BEFORE INSERT ON p FOR EACH ROW EXECUTE PROCEDURE sel_tg()
- Number of child tables: 4 (Use \d+ to list them.)

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 本地分区

- digoal=# \d c1
- Table "public.c1"
- Column | Type | Modifiers
- ----------+----------+----------
- id | integer |
- info | text |
- crt_time | timestamp without time zone |
- username | name |
- Check constraints:
- "ck" CHECK (crt_time >= '2013-01-01 00:00:00'::timestamp without time zone AND crt_time < '2013-02-01 00:00:00'::timestamp without time zone)
- Inherits: p

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 本地分区

- 当主表修改字段时, 继承表自动修改带继承属性的字段. 注意pg_attribute.attislocal
- digoal=# select attname,attislocal from pg_attribute where attrelid='p'::regclass;
- attname | attislocal
- ----------+------------
- tableoid | t
- cmax | t
- xmax | t
- cmin | t
- xmin | t
- ctid | t
- id | t
- info | t
- crt_time | t
- username | t
- (10 rows)

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 本地分区

- 新增的username字段是继承字段, 所以attislocal=false.
- digoal=# select attname,attislocal from pg_attribute where attrelid='c1'::regclass;
- attname | attislocal
- ----------+------------
- tableoid | t
- cmax | t
- xmax | t
- cmin | t
- xmin | t
- ctid | t
- id | t
- info | t
- crt_time | t
- username | f
- (10 rows)

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 本地分区

- 通过这种方式创建的继承表, 继承表的字段是本地字段.
- digoal=# create table c5(like p) inherits(p);
- NOTICE: merging column "id" with inherited definition
- NOTICE: merging column "info" with inherited definition
- NOTICE: merging column "crt_time" with inherited definition
- NOTICE: merging column "username" with inherited definition
- digoal=# select attname,attislocal from pg_attribute where attrelid='c5'::regclass;
- attname | attislocal
- ----------+------------
- tableoid | t
- cmax | t
- xmax | t
- cmin | t
- xmin | t
- ctid | t
- id | t
- info | t
- crt_time | t
- username | t

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 本地分区

- 如果要建成所有字段都是继承字段的子表, 请使用这种方式.
- digoal=# create table c7() inherits(p);
- CREATE TABLE
- digoal=# select attname,attislocal from pg_attribute where attrelid='c7'::regclass;
- attname | attislocal
- ----------+------------
- tableoid | t
- cmax | t
- xmax | t
- cmin | t
- xmin | t
- ctid | t
- id | f
- info | f
- crt_time | f
- (9 rows)
- 建完后, 所有的业务字段attislocal都是false的.

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 本地分区

- 注意, 本地字段不能从主表删除, 但是继承字段随着主表删除该字段而自动删除.
- digoal=# alter table p drop column username;
- c1, c2, c3, c4, c5的username字段能不能自动删除, 完全取决于它对应的系统表pg_attribute.attislocal的值. False则可以字段删除. 否则不会自动删除.

- 这些特点维护时必须加以注意.
- 正规的创建继承表应该使用这种方式
- digoal=# create table c7() inherits(p); -- 这种方式不会自动创建与主表类似的索引
- 或者用这种方式
- digoal=# create table c7(like p including all) inherits(p); -- 现在的子表字段是被创建为本地字段的.
- digoal=# update pg_attribute set attislocal=false where attrelid='c7'::regclass and attnum>=0; -- 使用这个方法可以把他们更新为继承字段.
- digoal=# select attname,attislocal from pg_attribute where attrelid='c7'::regclass;
- attname | attislocal
- ……
- ctid | t
- id | f
- info | f
- crt_time | f
- (9 rows)

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 本地分区

- 目前PostgreSQL分区表实施方法的存在的缺陷
- 性能缺陷
  - 目前PostgreSQL不管是使用约束还是触发器来实现范围缩小到分区表的情况, 性能都不是非常的理想, 会损耗一些CPU运算.
- 全局约束缺陷
  - 数据分布到多个子表后, 就无法简单的实现全局唯一了. 除非约束字段是是分布列.
- 全局外键关联缺陷
  - 全局外键也没有办法实现. 例如create table test (id int references p (id)), 如果p是主表的话, 是只能外键约束到主表本身的, 无法延续到子表.

SKY mobi
新手机 新应用 新娱乐

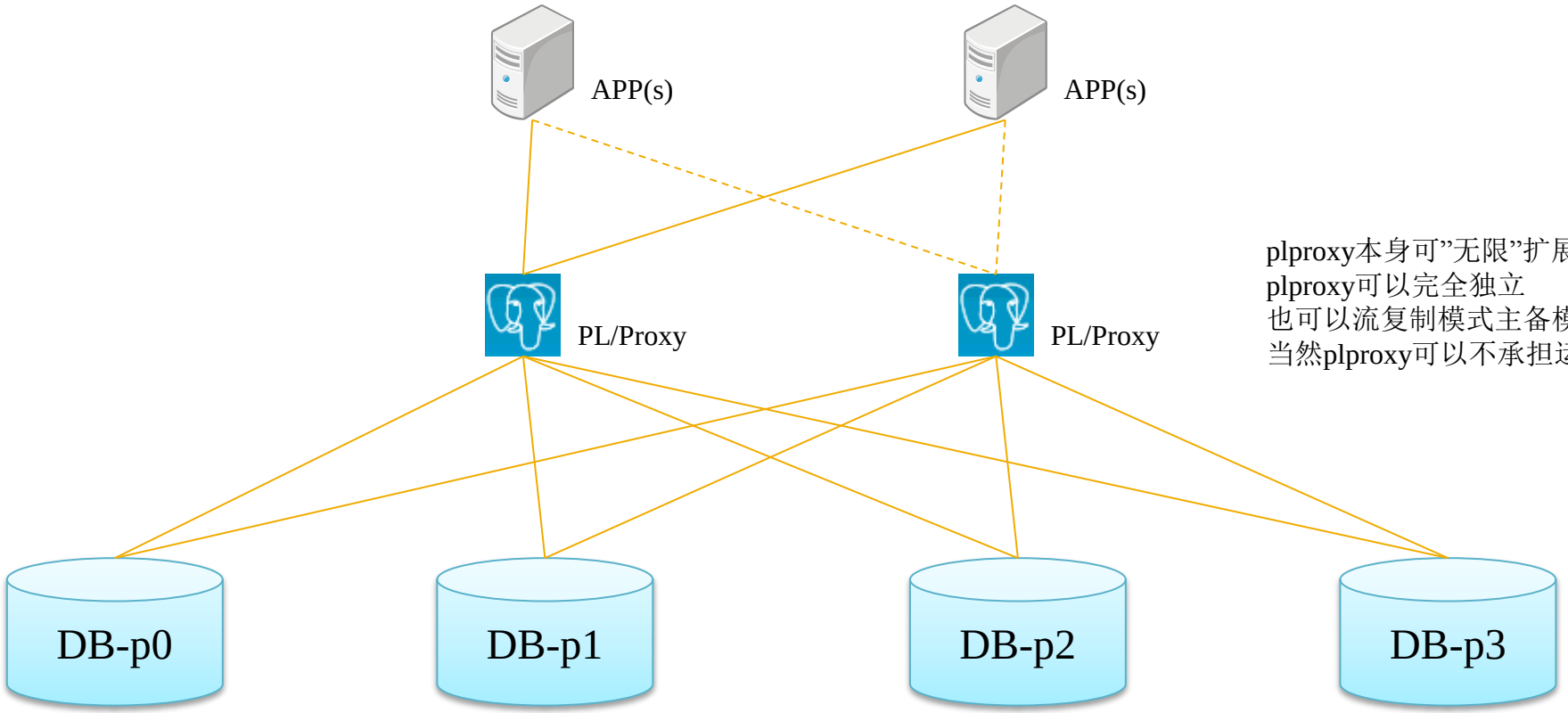
PostgreSQL 异地分区

- 异地shared nothing数据分区.
- 以PL/Proxy为例, 讲解异地shared nothing数据分区的部署.
- 应用场景设计
- PL/Proxy的安装
- 代理库的配置
- 编写代理函数
- 编写实体函数(负责实际的操作)

SKY mobi
新手机 新应用 新娱乐

PL/Proxy 应用场景举例

- Shared nothing(run on any|nr|hash) + replication(语句级复制,run on all) + Load Balance(run on ANY|NR|HASH)



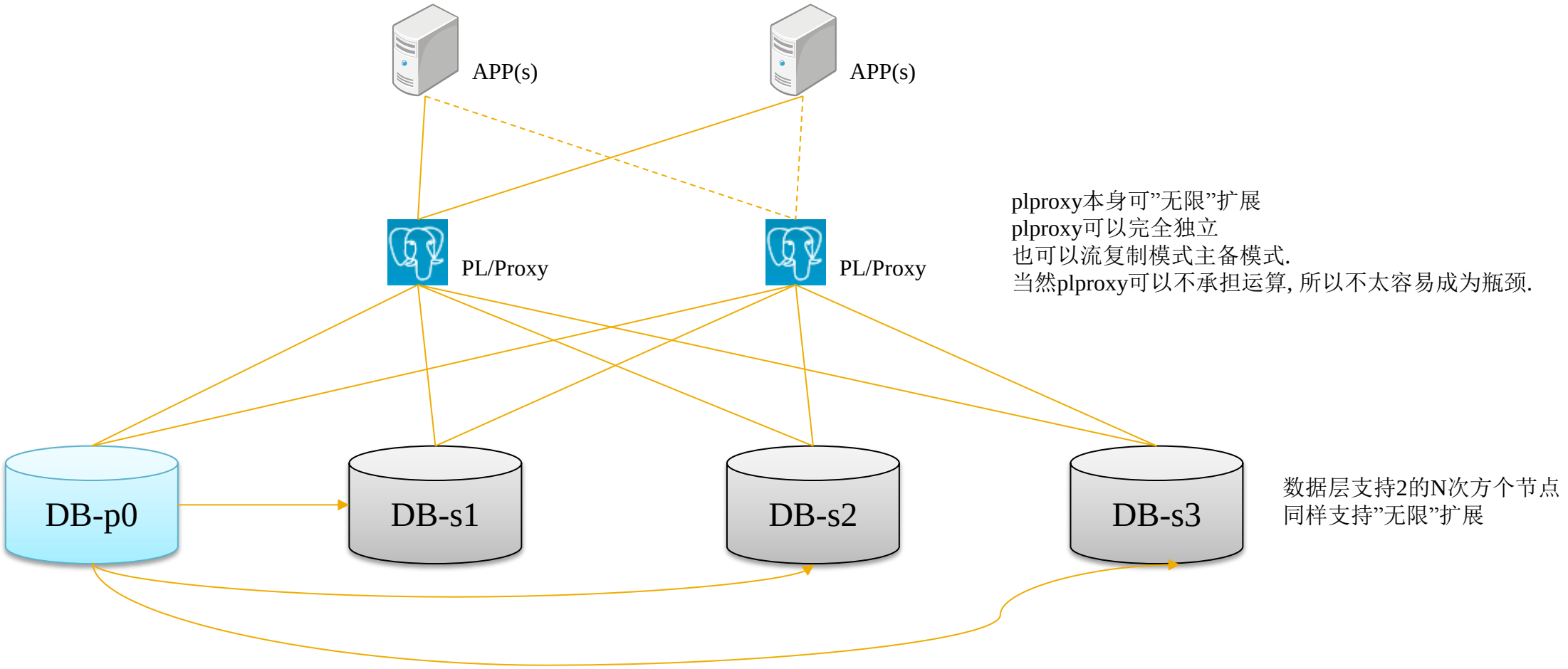
plproxy本身可”无限”扩展
plproxy可以完全独立
也可以流复制模式主备模式.
当然plproxy可以不承担运算, 所以不太容易成为瓶颈.

数据层支持2的N次方个节点
同样支持”无限”扩展

SKY mobi
新手机 新应用 新娱乐

PL/Proxy 应用场景举例

- Replication(流复制) + Load balance(读写分离, 读run on any|nr|hash, 写run on 0)



plproxy本身可”无限”扩展
plproxy可以完全独立
也可以流复制模式主备模式.
当然plproxy可以不承担运算, 所以不太容易成为瓶颈.

数据层支持2的N次方个节点
同样支持”无限”扩展

SKY mobi
新手机 新应用 新娱乐

PL/Proxy 使用背景

- PostgreSQL PL/Proxy解决方案
  - 水平扩展，对硬件要求不高，投入产出比非常容易计算。
  - PL/Proxy通过调用PostgreSQL函数支持事务，在PG中函数操作具有原子性，因此需要用到事务的操作可以封装在一个PG函数中。
  - PL/Proxy路由选择非常灵活，非常容易做到读写的负载均衡。
  - PL/Proxy与PostgreSQL免费。
  - PL/Proxy耦合度不高，如果应用不想通过PL/Proxy而想直连数据库可以不需要修改代码。因为代理函数与实体函数的输入输出参数是一致的。
  - 调用函数的好处：
    - 安全性提高, 应用连接的为代理数据库，要操作实体数据必须在数据节点有对应的函数，或者只有SELECT权限(甚至编译时可以选择不支持SELECT)。
    - 业务逻辑的代码放在数据端使处理效率更高。

SKY mobi
新手机 新应用 新娱乐

PL/Proxy 原理

- 层次 : PL/Proxy节点(s), 连接池(s), 数据节点(s)
- PL/Proxy
  - 接收应用程序发起的SQL请求(调用plproxy函数),
  - 解析为提交给数据节点的SQL,
    - PL/Proxy函数, CREATE FUNCTION get_data(IN first_name text, IN last_name text, OUT bdate date, OUT balance numeric(20,10))
    - 调用该函数被解析为, SELECT bdate::date, balance::numeric(20,10) FROM public.get_data($1::text, $2::text);
    - Explicite 类型转换, 指定输出顺序.
  - 旁路(CONNECT模式)或者选择数据节点(CLUSTER模式),
  - (CLUSTER模式1)查询SQL/MED配置的集群信息,选择数据节点通过libpq async API发送解析的SQL给数据节点(多个则并行),等待所有数据节点返回结果,返回结果给应用程序.
  - (CLUSTER模式2)查询集群配置版本,是否更新集群配置缓存,选择数据节点通过libpq async API发送解析的SQL给数据节点(多个则并行),等待所有数据节点返回结果,返回结果给应用程序.

SKY mobi
新手机 新应用 新娱乐

PL/Proxy 原理

- 连接池
  - 提高连接效率,复用连接. Plproxy到数据节点为长连接, 所以plproxy和数据节点之间一般可以不需要连接池.
  - 推荐短连接的应用在plproxy和应用程序之间架设连接池.
  - PL/Proxy 2以后代理和连接池的模块拆分了, 因此PL/Proxy不依赖连接池或连接池的类型.
- 数据节点
  - 存放实体数据,实体函数(plpgsql),接收并执行plproxy发送的SQL请求,将执行结果返回给plproxy.

SKY mobi
新手机 新应用 新娱乐

PL/Proxy 原理

- plproxy函数的用法。
  - 代理函数用法：应用程序传递参数，选择远程数据库节点，参数传递给远程数据库相同名字及参数类型的函数执行(或者target指定的数据节点函数)，收集数据节点返回结果发送给应用程序。
  - 不代理函数用法：在PL/Proxy函数中直接写SELECT查询，收集返回结果发送给应用程序。使用SELECT需要考虑转义字符的问题。
- PL/Proxy 函数支持的命令.
- CONNECT
  - CONNECT 'libpq connstr' ; | connect_func(...) | argname
- CLUSTER, [RUN ON ALL|ANY|int2,4,8|NR]
  - CLUSTER 'cluster_name'; | cluster_func(..)
- SELECT (CONNECT+SELECT 旁路模式)
- SPLIT, 传入参数为数组时，按元素分组拆分，可以减少plproxy和程序端的交互次数.
- TARGET, 当代理函数和节点函数不同名或者schema不同时可以使用target指定.

SKY mobi
新手机 新应用 新娱乐

PL/Proxy 原理

```
tag_run_on_partitions(ProxyFunction *func, FunctionCallInfo fcinfo, int tag,
                                        DatumArray **array_params, int array_row)
{
        ProxyCluster    *cluster = func->cur_cluster;
        int                             i;

        switch (func->run_type)
        {
                case R_HASH:
                        tag_hash_partitions(func, fcinfo, tag, array_params, array_row);
                        break;
                case R_ALL:
                        for (i = 0; i < cluster->part_count; i++)
                                cluster->part_map[i]->run_tag = tag;
                        break;
                case R_EXACT:
                        i = func->exact_nr;
                        if (i < 0 || i >= cluster->part_count)
                                plproxy_error(func, "part number out of range");
                        cluster->part_map[i]->run_tag = tag;
                        break;
                case R_ANY:
                        i = random() & cluster->part_mask;
                        cluster->part_map[i]->run_tag = tag;
                        break;
                default:
                        plproxy_error(func, "uninitialized run_type");
        }
}
```

SKY mobi
新手机 新应用 新娱乐

安装和使用PL/Proxy

- http://blog.163.com/digoal@126/blog/static/16387704020131022425437 65/
- 环境介绍
- PostgreSQL 9.3.2
- plproxy 2.x

- plrpoxy库：
- hostaddr 172.16.3.150
- port 1921
- user proxy
- password proxy
- dbname proxy
- schema digoal // 这个schema名和数据节点一致, 可以省去写target的步骤.

SKY mobi
新手机 新应用 新娱乐

安装和使用PL/Proxy

- 数据节点：
- hostaddr 172.16.3.150
- port 1921
- user digoal // plproxy将使用digoal用户连接数据节点.
- password digoal

- dbname db0
- schema digoal
- dbname db1
- schema digoal
- dbname db2
- schema digoal
- dbname db3
- schema digoal

SKY mobi
新手机 新应用 新娱乐

安装和使用PL/Proxy

- 安装, 配置, 演示
- http://blog.163.com/digoal@126/blog/static/16387704020131022425437 65/

SKY mobi
新手机 新应用 新娱乐

PL/Proxy 使用注意事项

- 1. 设计时需要注意
- plproxy函数所在的schema尽量和数据节点上实际函数的schema一致.
- 否则需要在plproxy函数中使用target指定 schema.functionname;
- 2. 数据节点的个数请保持2^n
- 3. 如果业务为短连接的形式, 那么需要1层连接池, 在应用程序和plproxy数据库之间. 而不是plproxy和数据节点之间.
- 在应用程序和plproxy之间加连接池后, 其实对于plproxy来说就是长连接了, 所以在plproxy和数据节点之间也就不需要连接池了.
- 4. 长连接不需要连接池, 因为plproxy和数据节点之间的连接是长连接.
- 5. plproxy语法 :
- connect, cluster, run, select, split, target.

SKY mobi
新手机 新应用 新娱乐

PL/Proxy 使用注意事项

- 6. 关于连接密码
- 出于安全考虑, 建议在任何配置中不要出现明文密码, 所以最好是plproxy服务器到数据节点是trust验证, 保护好plproxy即可.
- 假设plproxy在172.16.3.2上. 数据节点有4个, 库名和用户名都为digoal. 那么在4个节点上配置pg_hba.conf如下.
- node0
- host digoal digoal 172.16.3.2/32 trust
- node1
- host digoal digoal 172.16.3.2/32 trust
- node2
- host digoal digoal 172.16.3.2/32 trust
- node3
- host digoal digoal 172.16.3.2/32 trust

SKY mobi
新手机 新应用 新娱乐

PL/Proxy 使用注意事项

- 7. run 详解:
- run on <NR>, <NR>是数字常量, 范围是0 到 nodes-1; 例如有4个节点 run on 0; (run on 4则报错).
- run on ANY,
- run on function(...), 这里用到的函数返回结果必须是int2, int4 或 int8.
- run on ALL, 这种的plproxy函数必须是returns setof..., 实体函数没有setof的要求.
- 8. 一个plproxy中只能出现一条connect语句, 否则报错.
- digoal=# create or replace function f_test3() returns setof int8 as $$
- connect 'hostaddr=172.16.3.150 dbname=db0 user=digoal port=1921';
- connect 'hostaddr=172.16.3.150 dbname=db1 user=digoal port=1921';
- select count(*) from pg_class;
- $$ language plproxy strict;
- ERROR: PL/Proxy function postgres.f_test3(0): Compile error at line 2: Only one CONNECT statement allowed

SKY mobi
新手机 新应用 新娱乐

PL/Proxy 使用注意事项

- 9. 不要把plproxy语言的权限赋予给普通用户, 因为开放了trust认证, 如果再开放plproxy语言的权限是非常危险的.
- 正确的做法是使用超级用户创建plproxy函数, 然后把函数的执行权限赋予给普通用户.
- 千万不要这样省事 :
- update pg_language set lanpltrusted='t' where lanname='plproxy';
- 10. 如果有全局唯一的序列需求, 可以将序列的步调调整一下, 每个数据节点使用不同的初始值.
- 例如
- db0=# create sequence seq1 increment by 4 start with 1;
- CREATE SEQUENCE
- db1=# create sequence seq1 increment by 4 start with 2;
- ...

SKY mobi
新手机 新应用 新娱乐

练习

- 触发器实现本地分区
- 规则实现本地分区
- Pl/Proxy实现异地分区
- Londiste3进行数据合并和拆分(Day-2已经包含这部分内容)
- http://blog.163.com/digoal@126/blog/static/16387704020124294563291 2/
- http://blog.163.com/digoal@126/blog/static/16387704020124305133813 7/
- http://blog.163.com/digoal@126/blog/static/16387704020124311024489 51/
- http://blog.163.com/digoal@126/blog/static/1638770402012544131432 4/

SKY mobi
新手机 新应用 新娱乐

事务处理与并发控制

- 事务隔离级别,
- 并发控制,
- 锁的介绍,
- 死锁的发现和处理, 实际应用中如何避免死锁.

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 多版本并发控制

- PostgreSQL的多版本并发控制
- 版本识别演示.(INSERT, UPDATE, DELETE, 使用ctid定位, 并查看该TUPLE xmin, xmax的变化)

- 关键词
- XID -- 数据库的事务ID
- Tuple head: xmin, xmax, 行头部的XID信息, xmin表示插入这条记录的事务XID, xmax表示删除这条记录的事务XID
- Xid_snapshot : 当前集群中的未结束事务.
- Clog : 事务提交状态日志.

- 数据可见性条件 :
- 1. 记录的头部XID信息比当前事务更早. （repeatable read或ssi有这个要求, read committed没有这个要求）
- 2. 记录的头部XID信息不在当前的XID_snapshot中. (即记录上的事务状态不是未提交的状态.)
- 3. 记录头部的XID信息在CLOG中应该显示为已提交.

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 多版本并发控制

- 更新和删除数据时, 并不是直接删除行的数据, 而是更新行的头部信息中的xmax和infomask掩码.
- 事务提交后更新当前数据库集群的事务状态和pg_clog中的事务提交状态.
- Infomask和infomask2参看
- src/include/access/htup_details.h

- 例子 :
- 会话1 :
- digoal=# truncate iso_test ;
- TRUNCATE TABLE
- digoal=# insert into iso_test values (1,'test');
- INSERT 0 1
- digoal=# begin;
- BEGIN
- digoal=# update iso_test set info='new' where id=1;
- UPDATE 1

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 多版本并发控制

- 会话2 :
- digoal=# select ctid,xmin,xmax,* from iso_test where id=1;
- ctid | xmin | xmax | id | info
- -------+-----------+-----------+----+------
- (0,1) | 316732572 | 316732573 | 1 | test
- (1 row)

- PostgreSQL多版本并发控制不需要UNDO表空间.

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 多版本并发控制

- RR1 tuple-v1 IDLE IN TRANSACTION;
- RC1 tuple-v1 IDLE IN TRANSACTION;
- RC2 tuple-v1 UPDATE -> tuple-v2 COMMIT;
- RR1 tuple-v1 IDLE IN TRANSACTION;
- RC1 tuple-v2 IDLE IN TRANSACTION;
- RR2 tuple-v2 IDLE IN TRANSACTION;
- RC3 tuple-v2 UPDATE -> tuple-v3 COMMIT;
- RR1 tuple-v1 IDLE IN TRANSACTION;
- RR2 tuple-v2 IDLE IN TRANSACTION;
- RC1 tuple-v3 IDLE IN TRANSACTION;

SKY mobi
新手机 新应用 新娱乐

事务隔离级别

- 脏读
  - 在一个事务中可以读到其他未提交的事务产生或变更的数据.
  - PostgreSQL不支持read uncommitted事务隔离级别, 无法测试.
- 不可重复读
  - 在一个事务中, 再次读取前面SQL读过的数据时, 可能出现读取到的数据和前面读取到的不一致的现象. (例如其他事务在此期间已提交的数据)
  - 使用read committed事务隔离级别测试
- 幻像读
  - 在一个事务中, 再次执行同样的SQL, 得到的结果可能不一致.

- 标准SQL事务隔离级别, (PostgreSQL的repeatable read隔离级别不会产生幻像读)
- PostgreSQL不支持read uncommitted隔离级别.

Table 13-1. Standard SQL Transaction Isolation Levels

Isolation Level	Dirty Read	Nonrepeatable Read	Phantom Read
Read uncommitted	Possible	Possible	Possible
Read committed	Not possible	Possible	Possible
Repeatable read	Not possible	Not possible	Possible
Serializable	Not possible	Not possible	Not possible

SKY mobi
新手机 新应用 新娱乐

事务隔离级别测试1

- 不可重复读测试
- digoal=# create table iso_test(id int, info text);
- digoal=# insert into iso_test values (1, 'test');
- digoal=# begin isolation level read committed;
- BEGIN
- digoal=# select * from iso_test where id=1;
- id | info
- ----+------
- 1 | test
- (1 row)
- -- 其他会话更新这份数据, 并提交.
- digoal=# update iso_test set info='new' where id=1;
- -- 不可重复读出现.
- digoal=# select * from iso_test where id=1;
- id | info
- ----+------
- 1 | new
- (1 row)

SKY mobi
新手机 新应用 新娱乐

事务隔离级别测试2

- 幻象读测试
- digoal=# begin isolation level read committed;
- digoal=# select * from iso_test;
- id | info
- ----+------
- 1 | new
- (1 row)
- -- 其他会话新增数据
- digoal=# insert into iso_test values (2, 'test');
- -- 幻象读出现
- digoal=# select * from iso_test;
- id | info
- ----+------
- 1 | new
- 2 | test
- (2 rows)

SKY mobi
新手机 新应用 新娱乐

事务隔离级别测试3

- 使用repeatable read可避免不可重复读和幻象读.
- digoal=# delete from iso_test;
- digoal=# insert into iso_test values (1, 'test');
- digoal=# begin isolation level repeatable read;
- digoal=# select * from iso_test where id=1;
- id | info
- ----+------
- 1 | test
- (1 row)
- -- 其他会话修改数据, 并提交
- digoal=# update iso_test set info='new' where id=1;
- -- 未出现不可重复读现象.
- digoal=# select * from iso_test where id=1;
- id | info
- ----+------
- 1 | test
- (1 row)

SKY mobi
新手机 新应用 新娱乐

事务隔离级别测试3

- -- 其他会话新增数据.
- digoal=# insert into iso_test values (2, 'test');
- INSERT 0 1

- -- 未出现幻象读
- digoal=# select * from iso_test ;
- id | info
- ----+------
- 1 | test
- (1 row)

SKY mobi
新手机 新应用 新娱乐

事务隔离级别测试4

- PostgreSQL repeatable read情景案例
- 当repeatable read的事务去更新或删除在事务过程中被其他事务已经变更过的数据时, 将报错等待回滚.
- digoal=# truncate iso_test ;
- digoal=# insert into iso_test values (1,'test');
- digoal=# begin isolation level repeatable read;
- digoal=# select * from iso_test ;
- id | info
- ----+------
- 1 | test
- (1 row)
- -- 其他事务更新或者删除这条记录, 并提交.
- digoal=# update iso_test set info='new' where id=1;
- UPDATE 1
- -- 在repeatable read的事务中更新或者删除这条记录. 会报错回滚
- digoal=# update iso_test set info='tt' where id=1;
- ERROR: could not serialize access due to concurrent update
- digoal=# rollback;
- ROLLBACK

SKY mobi
新手机 新应用 新娱乐

事务隔离级别测试4

- 先获取锁, 再处理行上的数据(例如做条件判断.)
- 所以会有这种现象.
- -- 会话1
- digoal=# truncate iso_test ;
- TRUNCATE TABLE
- digoal=# insert into iso_test values (1,'test');
- INSERT 0 1
- digoal=# begin;
- BEGIN
- digoal=# update iso_test set id=id+1 returning id;
- id
- ----
- 2
- (1 row)
- UPDATE 1

SKY mobi
新手机 新应用 新娱乐

事务隔离级别测试4

- -- 会话2
- digoal=# select * from iso_test ;
- id | info
- ----+------
- 1 | test
- (1 row)
- digoal=# delete from iso_test where id=1; -- 等待ctid=(0,1)的行exclusive锁

- -- 会话1, 提交事务
- digoal=# end;
- COMMIT

- -- 会话2, 此时会话2等待的这条ctid(0,1)已经被会话1删除了(如果会话2是repeatable read模式的话这里会报错).
- DELETE 0
- digoal=# select * from iso_test;
- id | info
- ----+------
- 2 | test
- (1 row)

SKY mobi
新手机 新应用 新娱乐

事务隔离级别测试5

- **Serializable** 隔离级别
- 目标是模拟serializable的隔离级别事务的提交顺序转换为串行的执行顺序.
- 例如 :
- Start session a serializable
- Start session b serializable
- Session a SQL …
- Session b SQL …
- Session a|b SQL …
- …. Session a|b SQL …
- Commit b
- Commit a
- 这个场景模拟成 ：
- Start session b
- Sql …
- Commit b
- Start session a
- Sql … 如果会话a读过的数据在B中被变更, 那么a会话将提交失败.
- Commit a

SKY mobi
新手机 新应用 新娱乐

事务隔离级别测试5

- PostgreSQL 串行事务隔离级别的实现, 通过对扫描过的数据加载预锁来实现(内存中的一种弱冲突锁, 只在事务结束时判断是否有数据依赖性的冲突)
- 因为涉及到扫描的数据, 所以这种锁和执行计划有关.

- 例如
- Select * from tbl where a=1;
- 如果没有索引, 那么是全表扫描, 需要扫描所有的数据块.
- 加载的预锁是表级别的预锁. (那么期间如果其他串行事务对这个表有任何变更, 包括插入,删除,更新等. 并且先提交的话.)
- 这个会话结束的时候会发现预加锁的数据被其他串行事务变更了, 所以会提交失败.

- 如果a上有索引的话, 执行计划走索引的情况下, 扫描的数据包括行和索引页.
- 那么加载的预锁包含行和索引页.
- 这种情况仅当其他串行事务在此期间变更了相对应的行或者是索引页才会在结束时发生冲突.

SKY mobi
新手机 新应用 新娱乐

事务隔离级别测试5

- 例子 :
- 会话A :
- digoal=# select pg_backend_pid();
- -[RECORD 1]--+------
- pg_backend_pid | 12186
- 会话B :
- digoal=# select pg_backend_pid();
- -[RECORD 1]--+------
- pg_backend_pid | 12222
- 会话A :
- digoal=# truncate iso_test ;
- TRUNCATE TABLE
- digoal=# insert into iso_test select generate_series(1,100000);
- INSERT 0 100000
- digoal=# begin ISOLATION LEVEL SERIALIZABLE;
- BEGIN
- digoal=# select sum(id) from iso_test where id=100;
- -[RECORD 1]
- sum | 100

SKY mobi
新手机 新应用 新娱乐

事务隔离级别测试5

- 会话 C :
- digoal=# select relation::regclass,* from pg_locks where pid in (12186,12222);
- relation | locktype | database | relation | page | tuple | virtualxid | transactionid | classid | objid | objsubid | virtualtrans
- action | pid | mode | granted | fastpath
- ----------+------------+----------+----------+------+-------+------------+---------------+---------+-------+----------+-------------
- -------+-------+----------------+---------+----------
- iso_test | relation | 16384 | 92992 | | | | | | | | 1/157993
- | 12186 | AccessShareLock | t | t
- | virtualxid | | | | | 1/157993 | | | | | 1/157993
- | 12186 | ExclusiveLock | t | t
- iso_test | relation | 16384 | 92992 | | | | | | | | 1/157993
- | 12186 | SIReadLock | t | f
- (3 rows)
- 会话 B :
- digoal=# begin ISOLATION LEVEL SERIALIZABLE;
- BEGIN
- digoal=# select sum(id) from iso_test where id=10;
- -[RECORD 1]
- sum | 10

SKY mobi
新手机 新应用 新娱乐

事务隔离级别测试5

- 会话 C :
- digoal=# select relation::regclass,* from pg_locks where pid in (12186,12222);
- relation | locktype | database | relation | page | tuple | virtualxid | transactionid | classid | objid | objsubid | virtualtrans
- action | pid | mode | granted | fastpath
- ----------+------------+----------+----------+------+-------+------------+---------------+---------+-------+----------+-------------
- -------+-------+----------------+---------+----------
- iso_test | relation | 16384 | 92992 | | | | | | | | 1/157993
- | 12186 | AccessShareLock | t | t
- | virtualxid | | | | | 1/157993 | | | | | 1/157993
- | 12186 | ExclusiveLock | t | t
- iso_test | relation | 16384 | 92992 | | | | | | | | 2/6433312
- | 12222 | AccessShareLock | t | t
- | virtualxid | | | | | 2/6433312 | | | | | 2/6433312
- | 12222 | ExclusiveLock | t | t
- iso_test | relation | 16384 | 92992 | | | | | | | | 1/157993
- | 12186 | SIReadLock | t | f
- iso_test | relation | 16384 | 92992 | | | | | | | | 2/6433312
- | 12222 | SIReadLock | t | f
- (6 rows)

SKY mobi
新手机 新应用 新娱乐

事务隔离级别测试5

- 会话 A :
- digoal=# insert into iso_test values (1,'test');
- INSERT 0 1
- 会话 B :
- digoal=# insert into iso_test values (2,'test');
- INSERT 0 1

SKY mobi
新手机 新应用 新娱乐

事务隔离级别测试5

- 会话 C :
- digoal=# select relation::regclass,* from pg_locks where pid in (12186,12222);
- relation | locktype | database | relation | page | tuple | virtualxid | transactionid | classid | objid | objsubid | virtualtr
- ansaction | pid | mode | granted | fastpath
- ----------+---------------+----------+----------+------+-------+------------+---------------+---------+-------+----------+----------
- ----------+-------+-----------------+---------+----------
- iso_test | relation | 16384 | 92992 | | | | | | | 1/157993
- | 12186 | AccessShareLock | t | t
- iso_test | relation | 16384 | 92992 | | | | | | | 1/157993
- | 12186 | RowExclusiveLock | t | t
- | virtualxid | | | | | 1/157993 | | | | 1/157993
- | 12186 | ExclusiveLock | t | t
- iso_test | relation | 16384 | 92992 | | | | | | | 2/6433312
- | 12222 | AccessShareLock | t | t

SKY mobi
新手机 新应用 新娱乐

事务隔离级别测试5

- iso_test | relation | 16384 | 92992 | | | | | | | 2/6433312
- | 12222 | RowExclusiveLock | t | t
- | virtualxid | | | | | 2/6433312 | | | | | 2/6433312
- | 12222 | ExclusiveLock | t | t
- | transactionid | | | | | | 316732564 | | | | 1/157993
- | 12186 | ExclusiveLock | t | f
- | transactionid | | | | | | 316732565 | | | | 2/6433312
- | 12222 | ExclusiveLock | t | f
- iso_test | relation | 16384 | 92992 | | | | | | | 1/157993
- | 12186 | SIReadLock | t | f
- iso_test | relation | 16384 | 92992 | | | | | | | 2/6433312
- | 12222 | SIReadLock | t | f
- (10 rows)

SKY mobi
新手机 新应用 新娱乐

事务隔离级别测试5

- 会话 A :
- digoal=# commit;
- COMMIT
- 会话 C :
- digoal=# select relation::regclass,* from pg_locks where pid in (12186,12222);
- relation | locktype | database | relation | page | tuple | virtualxid | transactionid | classid | objid | objsubid | virtualtr
- ansaction | pid | mode | granted | fastpath
- ----------+---------------+----------+----------+------+-------+------------+---------------+---------+-------+----------+----------
- ----------+-------+-----------------+---------+----------
- iso_test | relation | 16384 | 92992 | | | | | | | | 2/6433312
- | 12222 | AccessShareLock | t | t
- iso_test | relation | 16384 | 92992 | | | | | | | | 2/6433312
- | 12222 | RowExclusiveLock | t | t
- | virtualxid | | | | | 2/6433312 | | | | | 2/6433312
- | 12222 | ExclusiveLock | t | t

SKY mobi
新手机 新应用 新娱乐

事务隔离级别5

- | transactionid | | | | | | 316732565 | | | | 2/6433312
- | 12222 | ExclusiveLock | t | f
- iso_test | relation | 16384 | 92992 | | | | | | | 1/157993
- | 12186 | SIReadLock | t | f
- iso_test | relation | 16384 | 92992 | | | | | | | 2/6433312
- | 12222 | SIReadLock | t | f
- (6 rows)

SKY mobi
新手机 新应用 新娱乐

事务隔离级别5

- 会话 B :
- digoal=# commit;
- ERROR: could not serialize access due to read/write dependencies among transactions
- DETAIL: Reason code: Canceled on identification as a pivot, during commit attempt.
- HINT: The transaction might succeed if retried.

- 会话 C :
- digoal=# select relation::regclass,* from pg_locks where pid in (12186,12222);
- relation | locktype | database | relation | page | tuple | virtualxid | transactionid | classid | objid | objsubid | virtualtransac
- tion | pid | mode | granted | fastpath
- ----------+----------+----------+----------+------+-------+------------+---------------+---------+-------+----------+---------------
- -----+-----+------+---------+----------
- (0 rows)

SKY mobi
新手机 新应用 新娱乐

事务隔离级别测试6

- 同样的场景, 加索引测试 :
- digoal=# create index idx_iso_test_1 on iso_test (id);
- CREATE INDEX

- digoal=# begin ISOLATION LEVEL SERIALIZABLE;
- BEGIN
- digoal=# select sum(id) from iso_test where id=100;
- -[RECORD 1]
- sum | 100

SKY mobi
新手机 新应用 新娱乐

事务隔离级别测试6

- digoal=# select relation::regclass,* from pg_locks where pid in (12186,12222);
- relation | locktype | database | relation | page | tuple | virtualxid | transactionid | classid | objid | objsubid | virtua
- ltransaction | pid | mode | granted | fastpath
- ----------+----------+---------+---------+------+-------+----------+--------------+---------+-------+---------+-------
- ----------+-------+----------------+---------+----------
- idx_iso_test_1 | relation | 16384 | 93017 | | | | | | | | 1/1579
- 96 | 12186 | AccessShareLock | t | t
- iso_test | relation | 16384 | 92992 | | | | | | | | 1/1579
- 96 | 12186 | AccessShareLock | t | t
- | virtualxid | | | | 1/157996 | | | | | 1/1579
- 96 | 12186 | ExclusiveLock | t | t
- iso_test | tuple | 16384 | 92992 | 0 | 100 | | | | | | 1/1579
- 96 | 12186 | SIReadLock | t | f
- idx_iso_test_1 | page | 16384 | 93017 | 1 | | | | | | | 1/1579
- 96 | 12186 | SIReadLock | t | f
- (5 rows)

- 这里变成了行锁和页锁

SKY mobi
新手机 新应用 新娱乐

事务隔离级别测试6

- digoal=# select relation::regclass,* from pg_locks where pid in (12186,12222);
- relation | locktype | database | relation | page | tuple | virtualxid | transactionid | classid | objid | objsubid | virtua
- ltransaction | pid | mode | granted | fastpath
- ----------+----------+---------+---------+------+-------+----------+--------------+---------+-------+---------+-------
- ----------+-------+----------------+---------+----------
- idx_iso_test_1 | relation | 16384 | 93017 | | | | | | | | 1/1579
- 96 | 12186 | AccessShareLock | t | t
- iso_test | relation | 16384 | 92992 | | | | | | | | 1/1579
- 96 | 12186 | AccessShareLock | t | t
- | virtualxid | | | | 1/157996 | | | | | 1/1579
- 96 | 12186 | ExclusiveLock | t | t
- iso_test | tuple | 16384 | 92992 | 0 | 100 | | | | | | 1/1579
- 96 | 12186 | SIReadLock | t | f
- idx_iso_test_1 | page | 16384 | 93017 | 1 | | | | | | | 1/1579
- 96 | 12186 | SIReadLock | t | f
- (5 rows)

- 这里变成了行锁和页锁

SKY mobi
新手机 新应用 新娱乐

事务隔离级别测试6

- digoal=# begin ISOLATION LEVEL SERIALIZABLE;
- BEGIN
- digoal=# select sum(id) from iso_test where id=10;
- -[RECORD 1]
- sum | 10

- digoal=# select relation::regclass,* from pg_locks where pid in (12186,12222);
- relation | locktype | database | relation | page | tuple | virtualxid | transactionid | classid | objid | objsubid | virtua
- ltransaction | pid | mode | granted | fastpath
- ----------+----------+----------+----------+------+-------+------------+---------------+---------+-------+----------+-------
- ----------+-------+----------------+---------+----------
- idx_iso_test_1 | relation | 16384 | 93017 | | | | | | | | 1/1579
- 96 | 12186 | AccessShareLock | t | t
- iso_test | relation | 16384 | 92992 | | | | | | | | 1/1579
- 96 | 12186 | AccessShareLock | t | t
- | virtualxid | | | | | 1/157996 | | | | | 1/1579
- 96 | 12186 | ExclusiveLock | t | t

SKY mobi
新手机 新应用 新娱乐

事务隔离级别测试6

- idx_iso_test_1 | relation | 16384 | 93017 | | | | | | | | 2/6433
- 314 | 12222 | AccessShareLock | t | t
- iso_test | relation | 16384 | 92992 | | | | | | | | 2/6433
- 314 | 12222 | AccessShareLock | t | t
- | virtualxid | | | | | 2/6433314 | | | | | 2/6433
- 314 | 12222 | ExclusiveLock | t | t
- iso_test | tuple | 16384 | 92992 | 0 | 100 | | | | | | 1/1579
- 96 | 12186 | SIReadLock | t | f
- iso_test | tuple | 16384 | 92992 | 0 | 10 | | | | | | 2/6433
- 314 | 12222 | SIReadLock | t | f
- idx_iso_test_1 | page | 16384 | 93017 | 1 | | | | | | | 1/1579
- 96 | 12186 | SIReadLock | t | f
- idx_iso_test_1 | page | 16384 | 93017 | 1 | | | | | | | 2/6433
- 314 | 12222 | SIReadLock | t | f
- (10 rows)

SKY mobi
新手机 新应用 新娱乐

事务隔离级别测试6

- digoal=# insert into iso_test values (1,'test');
- INSERT 0 1

- digoal=# select relation::regclass,* from pg_locks where pid in (12186,12222);
- relation | locktype | database | relation | page | tuple | virtualxid | transactionid | classid | objid | objsubid | vir
- tualtransaction | pid | mode | granted | fastpath
- ----------+----------+----------+----------+------+-------+------------+---------------+---------+-------+----------+----
- ----------+-------+----------+---------+----------
- idx_iso_test_1 | relation | 16384 | 93017 | | | | | | | | 1/1
- 57996 | 12186 | AccessShareLock | t | t
- iso_test | relation | 16384 | 92992 | | | | | | | | 1/1
- 57996 | 12186 | AccessShareLock | t | t
- iso_test | relation | 16384 | 92992 | | | | | | | | 1/1
- 57996 | 12186 | RowExclusiveLock | t | t
- | virtualxid | | | | | 1/157996 | | | | | 1/1

SKY mobi
新手机 新应用 新娱乐

事务隔离级别测试6

- 57996 | 12186 | ExclusiveLock | t | t
- idx_iso_test_1 | relation | 16384 | 93017 | | | | | | | | 2/6
- 433314 | 12222 | AccessShareLock | t | t
- iso_test | relation | 16384 | 92992 | | | | | | | | 2/6
- 433314 | 12222 | AccessShareLock | t | t
- | virtualxid | | | | | 2/6433314 | | | | | 2/6
- 433314 | 12222 | ExclusiveLock | t | t
- | transactionid | | | | | | 316732567 | | | | 1/1
- 57996 | 12186 | ExclusiveLock | t | f
- iso_test | tuple | 16384 | 92992 | 0 | 100 | | | | | | 1/1
- 57996 | 12186 | SIReadLock | t | f
- iso_test | tuple | 16384 | 92992 | 0 | 10 | | | | | | 2/6
- 433314 | 12222 | SIReadLock | t | f
- idx_iso_test_1 | page | 16384 | 93017 | 1 | | | | | | | 1/1
- 57996 | 12186 | SIReadLock | t | f
- idx_iso_test_1 | page | 16384 | 93017 | 1 | | | | | | | 2/6
- 433314 | 12222 | SIReadLock | t | f
- (12 rows)

SKY mobi
新手机 新应用 新娱乐

事务隔离级别测试6

- digoal=# insert into iso_test values (2,'test');
- INSERT 0 1

- digoal=# select relation::regclass,* from pg_locks where pid in (12186,12222);
- relation | locktype | database | relation | page | tuple | virtualxid | transactionid | classid | objid | objsubid | vir
- tualtransaction | pid | mode | granted | fastpath
- ----------+----------+----------+----------+------+-------+------------+---------------+---------+-------+----------+----
- ----------+-------+----------+---------+----------
- idx_iso_test_1 | relation | 16384 | 93017 | | | | | | | | 1/1
- 57996 | 12186 | AccessShareLock | t | t
- iso_test | relation | 16384 | 92992 | | | | | | | | 1/1
- 57996 | 12186 | AccessShareLock | t | t
- iso_test | relation | 16384 | 92992 | | | | | | | | 1/1
- 57996 | 12186 | RowExclusiveLock | t | t
- | virtualxid | | | | | 1/157996 | | | | | 1/1
- 57996 | 12186 | ExclusiveLock | t | t
- idx_iso_test_1 | relation | 16384 | 93017 | | | | | | | | 2/6
- 433314 | 12222 | AccessShareLock | t | t

SKY mobi
新手机 新应用 新娱乐

事务隔离级别测试6

- iso_test | relation | 16384 | 92992 | | | | | | | | 2/6
- 433314 | 12222 | AccessShareLock | t | t
- iso_test | relation | 16384 | 92992 | | | | | | | | 2/6
- 433314 | 12222 | RowExclusiveLock | t | t
- | virtualxid | | | | | 2/6433314 | | | | | 2/6
- 433314 | 12222 | ExclusiveLock | t | t
- | transactionid | | | | | | 316732567 | | | | 1/1
- 57996 | 12186 | ExclusiveLock | t | f
- | transactionid | | | | | | 316732568 | | | | 2/6
- 433314 | 12222 | ExclusiveLock | t | f
- iso_test | tuple | 16384 | 92992 | 0 | 100 | | | | | | 1/1
- 57996 | 12186 | SIReadLock | t | f
- iso_test | tuple | 16384 | 92992 | 0 | 10 | | | | | | 2/6
- 433314 | 12222 | SIReadLock | t | f
- idx_iso_test_1 | page | 16384 | 93017 | 1 | | | | | | | 1/1
- 57996 | 12186 | SIReadLock | t | f
- idx_iso_test_1 | page | 16384 | 93017 | 1 | | | | | | | 2/6
- 433314 | 12222 | SIReadLock | t | f
- (14 rows)

SKY mobi
新手机 新应用 新娱乐

事务隔离级别测试6

- digoal=# commit;
- COMMIT

- digoal=# commit;
- ERROR: could not serialize access due to read/write dependencies among transactions
- DETAIL: Reason code: Canceled on identification as a pivot, during commit attempt.
- HINT: The transaction might succeed if retried.
- 索引页用了同一个, 并且被插入语句更新了. 所以发生了冲突

- 如果其中一个插入的值不在1号索引页则没有问题, 例如
- digoal=# begin ISOLATION LEVEL SERIALIZABLE;
- BEGIN
- digoal=# select sum(id) from iso_test where id=100;
- -[RECORD 1]
- sum | 100

SKY mobi
新手机 新应用 新娱乐

事务隔离级别测试6

- digoal=# insert into iso_test values (1,'test');
- INSERT 0 1
- digoal=# commit;
- COMMIT

- digoal=# begin ISOLATION LEVEL SERIALIZABLE;
- BEGIN
- digoal=# select sum(id) from iso_test where id=10;
- -[RECORD 1]
- sum | 10

- digoal=# insert into iso_test values (200000,'test');
- INSERT 0 1
- digoal=# commit;
- COMMIT

- (200000,'test') 这个索引页不在1号
- idx_iso_test_1 | page | 16384 | 93017 | 275 | | | | | | | 2/6433
- 316 | 12222 | SIReadLock | t | f

SKY mobi
新手机 新应用 新娱乐

事务隔离级别测试6

- 注意事项
- PostgreSQL 的 hot_standby节点不支持串行事务隔离级别, 只能支持read committed和repeatable read隔离级别.

SKY mobi
新手机 新应用 新娱乐

PostgreSQL锁的介绍

- 锁对象的类型
- relation, extend, page, tuple, transactionid, virtualxid, object, userlock, or advisory
- src/include/storage/lock.h
- /*
- * LOCKTAG is the key information needed to look up a LOCK item in the
- * lock hashtable. A LOCKTAG value uniquely identifies a lockable object.
- *
- * The LockTagType enum defines the different kinds of objects we can lock.
- * We can handle up to 256 different LockTagTypes.
- */
- typedef enum LockTagType
- {
- LOCKTAG_RELATION, /* whole relation */
- /* ID info for a relation is DB OID + REL OID; DB OID = 0 if shared */
- LOCKTAG_RELATION_EXTEND, /* the right to extend a relation */
- /* same ID info as RELATION */
- LOCKTAG_PAGE, /* one page of a relation */
- /* ID info for a page is RELATION info + BlockNumber */

SKY mobi
新手机 新应用 新娱乐

PostgreSQL锁的介绍

```
        LOCKTAG_TUPLE,                  /* one physical tuple */
        /* ID info for a tuple is PAGE info + OffsetNumber */
        LOCKTAG_TRANSACTION,            /* transaction (for waiting for xact done) */
        /* ID info for a transaction is its TransactionId */
        LOCKTAG_VIRTUALTRANSACTION, /* virtual transaction (ditto) */
        /* ID info for a virtual transaction is its VirtualTransactionId */
        LOCKTAG_OBJECT,                 /* non-relation database object */
        /* ID info for an object is DB OID + CLASS OID + OBJECT OID + SUBID */

        /*
         * Note: object ID has same representation as in pg_depend and
         * pg_description, but notice that we are constraining SUBID to 16 bits.
         * Also, we use DB OID = 0 for shared objects such as tablespaces.
         */
        LOCKTAG_USERLOCK,               /* reserved for old contrib/userlock code */
        LOCKTAG_ADVISORY                /* advisory user locks */
    } LockTagType;
```

SKY mobi
新手机 新应用 新娱乐

PostgreSQL锁的介绍

- 锁模式1(标准锁和用户锁方法支持的模式都在这里)
- /*
- * These are the valid values of type LOCKMODE for all the standard lock
- * methods (both DEFAULT and USER).
- */
- #define NoLock 0 /* NoLock is not a lock mode, but a flag value meaning "don't get a lock" */
- #define AccessShareLock 1 /* SELECT */
- #define RowShareLock 2 /* SELECT FOR UPDATE/FOR SHARE */
- #define RowExclusiveLock 3 /* INSERT, UPDATE, DELETE */
- #define ShareUpdateExclusiveLock 4 /* VACUUM (non-FULL),ANALYZE, CREATE
- * INDEX CONCURRENTLY */
- #define ShareLock 5 /* CREATE INDEX (WITHOUT CONCURRENTLY) */
- #define ShareRowExclusiveLock 6 /* like EXCLUSIVE MODE, but allows ROW
- * SHARE */
- #define ExclusiveLock 7 /* blocks ROW SHARE/SELECT...FOR
- * UPDATE */
- #define AccessExclusiveLock 8 /* ALTER TABLE, DROP TABLE, VACUUM
- * FULL, and unqualified LOCK TABLE */

SKY mobi
新手机 新应用 新娱乐

PostgreSQL锁的介绍

- 锁模式冲突表

Table 13-2. Conflicting Lock Modes

Requested Lock Mode	Current Lock Mode							
	ACCESS SHARE	ROW SHARE	ROW EXCLUSIVE	SHARE UPDATE EXCLUSIVE	SHARE	SHARE ROW EXCLUSIVE	EXCLUSIVE	ACCESS EXCLUSIVE
ACCESS SHARE								X
ROW SHARE							X	X
ROW EXCLUSIVE					X	X	X	X
SHARE UPDATE EXCLUSIVE				X	X	X	X	X
SHARE			X	X		X	X	X
SHARE ROW EXCLUSIVE			X	X	X	X	X	X
EXCLUSIVE		X	X	X	X	X	X	X
ACCESS EXCLUSIVE	X	X	X	X	X	X	X	X

SKY mobi
新手机 新应用 新娱乐

PostgreSQL锁的介绍

- 行锁模式
- src/include/access/heapam.h

```
/*
 * Possible lock modes for a tuple.
 */
typedef enum LockTupleMode
{
        /* SELECT FOR KEY SHARE */
        LockTupleKeyShare,
        /* SELECT FOR SHARE */
        LockTupleShare,
        /* SELECT FOR NO KEY UPDATE, and UPDATEs that don't modify key columns */
        LockTupleNoKeyExclusive,
        /* SELECT FOR UPDATE, UPDATEs that modify key columns, and DELETE */
        LockTupleExclusive
} LockTupleMode;
```

SKY mobi
新手机 新应用 新娱乐

PostgreSQL锁的介绍

- 行锁模式冲突表
- src/backend/access/heap/README.tuplock

	KEY UPDATE	UPDATE	SHARE	KEY SHARE
KEY UPDATE	conflict	conflict	conflict	conflict
UPDATE	conflict	conflict	conflict	
SHARE	conflict	conflict		
KEY SHARE	conflict			

- 例子
- http://blog.163.com/digoal@126/blog/static/163877040201303051096 87/

SKY mobi
新手机 新应用 新娱乐

PostgreSQL锁的介绍

- 串行锁模式
- SIReadLock

- src/backend/storage/lmgr/README-SSI
- http://www.postgresql.org/docs/9.3/static/transaction-iso.html#XACT-SERIALIZABLE

SKY mobi
新手机 新应用 新娱乐

PostgreSQL锁的介绍

- PostgreSQL获取锁的方法
- /*
- * These macros define how we map logical IDs of lockable objects into
- * the physical fields of LOCKTAG. Use these to set up LOCKTAG values,
- * rather than accessing the fields directly. Note multiple eval of target!
- */
- #define SET_LOCKTAG_RELATION(locktag,dboid,reloid) \
- ((locktag).locktag_field1 = (dboid), \
- (locktag).locktag_field2 = (reloid), \
- (locktag).locktag_field3 = 0, \
- (locktag).locktag_field4 = 0, \
- (locktag).locktag_type = LOCKTAG_RELATION, \
- (locktag).locktag_lockmethodid = DEFAULT_LOCKMETHOD)

SKY mobi
新手机 新应用 新娱乐

PostgreSQL锁的介绍

```
#define SET_LOCKTAG_RELATION_EXTEND(locktag,dboid,reloid) \
        ((locktag).locktag_field1 = (dboid), \
         (locktag).locktag_field2 = (reloid), \
         (locktag).locktag_field3 = 0, \
         (locktag).locktag_field4 = 0, \
         (locktag).locktag_type = LOCKTAG_RELATION_EXTEND, \
         (locktag).locktag_lockmethodid = DEFAULT_LOCKMETHOD)

#define SET_LOCKTAG_PAGE(locktag,dboid,reloid,blocknum) \
        ((locktag).locktag_field1 = (dboid), \
         (locktag).locktag_field2 = (reloid), \
         (locktag).locktag_field3 = (blocknum), \
         (locktag).locktag_field4 = 0, \
         (locktag).locktag_type = LOCKTAG_PAGE, \
         (locktag).locktag_lockmethodid = DEFAULT_LOCKMETHOD)
```

SKY mobi
新手机 新应用 新娱乐

PostgreSQL锁的介绍

```
#define SET_LOCKTAG_TUPLE(locktag,dboid,reloid,blocknum,offnum) \
        ((locktag).locktag_field1 = (dboid), \
         (locktag).locktag_field2 = (reloid), \
         (locktag).locktag_field3 = (blocknum), \
         (locktag).locktag_field4 = (offnum), \
         (locktag).locktag_type = LOCKTAG_TUPLE, \
         (locktag).locktag_lockmethodid = DEFAULT_LOCKMETHOD)

#define SET_LOCKTAG_TRANSACTION(locktag,xid) \
        ((locktag).locktag_field1 = (xid), \
         (locktag).locktag_field2 = 0, \
         (locktag).locktag_field3 = 0, \
         (locktag).locktag_field4 = 0, \
         (locktag).locktag_type = LOCKTAG_TRANSACTION, \
         (locktag).locktag_lockmethodid = DEFAULT_LOCKMETHOD)
```

SKY mobi
新手机 新应用 新娱乐

PostgreSQL锁的介绍

```
#define SET_LOCKTAG_VIRTUALTRANSACTION(locktag,vxid) \
        ((locktag).locktag_field1 = (vxid).backendId, \
         (locktag).locktag_field2 = (vxid).localTransactionId, \
         (locktag).locktag_field3 = 0, \
         (locktag).locktag_field4 = 0, \
         (locktag).locktag_type = LOCKTAG_VIRTUALTRANSACTION, \
         (locktag).locktag_lockmethodid = DEFAULT_LOCKMETHOD)

#define SET_LOCKTAG_OBJECT(locktag,dboid,classoid,objoid,objsubid) \
        ((locktag).locktag_field1 = (dboid), \
         (locktag).locktag_field2 = (classoid), \
         (locktag).locktag_field3 = (objoid), \
         (locktag).locktag_field4 = (objsubid), \
         (locktag).locktag_type = LOCKTAG_OBJECT, \
         (locktag).locktag_lockmethodid = DEFAULT_LOCKMETHOD)
```

SKY mobi
新手机 新应用 新娱乐

PostgreSQL锁的介绍

```
#define SET_LOCKTAG_ADVISORY(locktag,id1,id2,id3,id4) \
        ((locktag).locktag_field1 = (id1), \
         (locktag).locktag_field2 = (id2), \
         (locktag).locktag_field3 = (id3), \
         (locktag).locktag_field4 = (id4), \
         (locktag).locktag_type = LOCKTAG_ADVISORY, \
         (locktag).locktag_lockmethodid = USER_LOCKMETHOD)
```

SKY mobi
新手机 新应用 新娱乐

PostgreSQL advisory锁介绍

- repeatable read及以上级别长事务带来的问题举例
- 会话A :
- digoal=# begin isolation level repeatable read;
- BEGIN
- digoal=# select 1;
- ?column?
- ----------
- 1
- (1 row)
- 假设这是个长事务.

SKY mobi
新手机 新应用 新娱乐

PostgreSQL advisory锁介绍

- 会话B :
- digoal=# delete from iso_test;
- DELETE 10000
- digoal=# vacuum verbose iso_test ;
- INFO: vacuuming "postgres.iso_test"
- INFO: "iso_test": found 0 removable, 10000 nonremovable row versions in 55 out of 55 pages
- DETAIL: 10000 dead row versions cannot be removed yet.
- There were 0 unused item pointers.
- 0 pages are entirely empty.
- CPU 0.00s/0.00u sec elapsed 0.00 sec.
- INFO: vacuuming "pg_toast.pg_toast_93022"
- INFO: index "pg_toast_93022_index" now contains 0 row versions in 1 pages

SKY mobi
新手机 新应用 新娱乐

PostgreSQL advisory锁介绍

- DETAIL: 0 index row versions were removed.
- 0 index pages have been deleted, 0 are currently reusable.
- CPU 0.00s/0.00u sec elapsed 0.00 sec.
- INFO: "pg_toast_93022": found 0 removable, 0 nonremovable row versions in 0 out of 0 pages
- DETAIL: 0 dead row versions cannot be removed yet.
- There were 0 unused item pointers.
- 0 pages are entirely empty.
- CPU 0.00s/0.00u sec elapsed 0.00 sec.
- VACUUM

- 会话A结束后, 这部分数据才可以被回收掉
- End;

- digoal=# vacuum verbose iso_test ;
- INFO: vacuuming "postgres.iso_test"
- INFO: "iso_test": removed 10000 row versions in 55 pages
- INFO: "iso_test": found 10000 removable, 0 nonremovable row versions in 55 out of 55 pages

SKY mobi
新手机 新应用 新娱乐

PostgreSQL advisory锁介绍

- DETAIL: 0 dead row versions cannot be removed yet.
- There were 0 unused item pointers.
- 0 pages are entirely empty.
- CPU 0.00s/0.00u sec elapsed 0.00 sec.
- INFO: "iso_test": truncated 55 to 0 pages
- DETAIL: CPU 0.00s/0.00u sec elapsed 0.00 sec.
- INFO: vacuuming "pg_toast.pg_toast_93022"
- INFO: index "pg_toast_93022_index" now contains 0 row versions in 1 pages
- DETAIL: 0 index row versions were removed.
- 0 index pages have been deleted, 0 are currently reusable.
- CPU 0.00s/0.00u sec elapsed 0.00 sec.
- INFO: "pg_toast_93022": found 0 removable, 0 nonremovable row versions in 0 out of 0 pages
- DETAIL: 0 dead row versions cannot be removed yet.
- There were 0 unused item pointers.
- 0 pages are entirely empty.
- CPU 0.00s/0.00u sec elapsed 0.00 sec.
- VACUUM

SKY mobi
新手机 新应用 新娱乐

PostgreSQL advisory锁介绍

- advisory会话锁解决的问题
- http://blog.163.com/digoal@126/blog/static/16387704020117249221783O/
- http://blog.163.com/digoal@126/blog/static/16387704020135181110434 63/

- advisory lock的应用场景举例(应用控制的锁):
- 比如数据库里面存储了文件和ID的对应关系，应用程序需要长时间得获得一个锁，然后对文件进行修改，再释放锁。
- 测试数据:
- digoal=> create table tbl_file_info (id int primary key,file_path text);
- NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "tbl_file_info_pkey" for table "tbl_file_info"
- CREATE TABLE
- digoal=> insert into tbl_file_info values (1,'/home/postgres/advisory_lock_1.txt');
- INSERT 0 1
- digoal=> insert into tbl_file_info values (2,'/home/postgres/advisory_lock_2.txt');
- INSERT 0 1
- digoal=> insert into tbl_file_info values (3,'/home/postgres/advisory_lock_3.txt');
- INSERT 0 1

SKY mobi
新手机 新应用 新娱乐

PostgreSQL advisory锁介绍

- SESSION A:
- digoal=> select pg_advisory_lock(id),file_path from tbl_file_info where id=1;
- pg_advisory_lock | file_path
- ----------+----------
- | /home/postgres/advisory_lock_1.txt
- (1 row)
- 应用程序对/home/postgres/advisory_lock_1.txt文件进行编辑之后，再释放这个advisory锁。

- SESSION B:
- 当SESSIONA在编辑/home/postgres/advisory_lock_1.txt这个文件的时候，无法获得这个锁，所以可以确保不会同时编辑这个文件。

SKY mobi
新手机 新应用 新娱乐

PostgreSQL deadlock检测和规避

- SESSION A:
  - Lock tuple 1;
- SESSION B:
  - Lock tuple 2;
- SESSION A:
  - Lock tuple 1 waiting;
- SESSION B:
  - Lock tuple 2 waiting;
- A,B相互等待.

- 死锁检测算法介绍
- src/backend/storage/lmgr/README

- 死锁检测的时间间隔配置, deadlock_timeout 默认为1秒.
- 锁等待超过这个配置后, 触发死锁检测算法.
- 因为死锁检测比较耗资源, 所以这个时间视情况而定.
- PostgreSQL和Oracle死锁检测的区别例子.
- http://blog.163.com/digoal@126/blog/static/163877040201138117117 16/
- 规避死锁需要从业务逻辑的角度去规避, 避免发生这种交错持锁和交错等待的情况.

SKY mobi
新手机 新应用 新娱乐

练习

- 通过观察pg_locks了解各种SQL获取的锁对象类型和锁模式对应关系.
- 验证锁冲突

SKY mobi
新手机 新应用 新娱乐

监控数据库活动

- 了解各种维度的统计信息的解读(table, index, database, replication, sql...), 锁信息解读, 磁盘使用, 数据库级别或事务级别或会话级别的活动信息统计, 历史数据库活动统计报告

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 统计信息采集进程

- 了解各种维度的统计信息的解读(table, index, database, replication, sql...), 锁信息解读, 磁盘使用, 数据库级别或事务级别或会话级别的活动信息统计, 历史数据库活动统计报告

- PostgreSQL 用于收集统计信息的进程
  - stats collector process "src/backend/postmaster/pgstat.c"

- PostgreSQL 统计信息的存放地 (启动时, 读入已存在的统计文件, 或初始化0. 数据库运行过程中存储在内存和temp文件; 数据库关闭时保存到非易失存储)
  - 启动时 : shared buffer -> "src/backend/postmaster/pgstat.c" -> PgstatCollectorMain -> pgstat_read_statsfiles
  - postgresql.conf - stats_temp_directory , 推荐配置在高速磁盘或内存文件系统中.
  - 数据库正常关闭时会把统计信息从tmp目录拷贝到$PGDATA/pg_stat目录中, 确保统计信息不会丢失. -> pgstat_write_statsfiles(true, true);

- 统计信息的收集维度配置
  - track_activities (boolean) -- 收集SQL执行开始时间以及SQL语句的内容. 默认打开.
  - track_activity_query_size (integer) -- 指定统计信息中允许存储的SQL长度, 超出长度的SQL被截断. 默认1024. pg_stat_activity.query
  - track_counts (boolean) -- 收集数据库的活动信息(如表新增的行数, 删除的行数等), autovacuum进程需要用到这部分信息.
  - track_io_timing (boolean) -- 收集IO操作的时间开销, 因为需要不断的调用系统当前时间, 所以某些系统中会带来极大的负面影响.
  - 被用于pg_stat_database, pg_stat_statements 显示IO时间. (使用pg_test_timing测试时间统计的影响)
  - track_functions (enum) -- 跟踪函数的调用次数和时间开销. 可配置pl(仅包括plpgsql函数), all(包括SQL,C,plpgsql函数), off
  - update_process_title (boolean) -- 每次服务端process接收到新的SQL时更新command状态. (ps命令可见)

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 统计信息采集进程

- log_statement_stats (boolean) -- 类似unix的getrusage()操作系统函数, 用于收集SQL语句级的资源开销统计. 包含以下3种层面的全部.
- 因此配置了log_statement_stats就不需要配置以下选项.
- log_parser_stats (boolean) -- 同上, 但是只包含SQL parser部分的资源开销统计.
- log_planner_stats (boolean) -- 同上, 但是只包含SQL planner部分的资源开销统计.
- log_executor_stats (boolean) -- 同上, 但是只包含SQL executor部分的资源开销统计.

- 查看统计信息的视图
  - pg_stat和pg_statio开头的系统视图
  - http://www.postgresql.org/docs/9.3/static/monitoring-stats.html
- 查看或重置统计信息的函数

Table 27-13. Additional Statistics Functions

Function	Return Type	Description
pg_backend_pid()	integer	Process ID of the server process handling the current session
pg_stat_get_activity(integer)	setof record	Returns a record of information about the backend with the specified PID, or one record for each active backend in the system if NULL is specified. The fields returned are a subset of those in the pg_stat_activity view.
pg_stat_clear_snapshot()	void	Discard the current statistics snapshot
pg_stat_reset()	void	Reset all statistics counters for the current database to zero (requires superuser privileges)
pg_stat_reset_shared(text)	void	Reset some cluster-wide statistics counters to zero, depending on the argument (requires superuser privileges). Calling pg_stat_reset_shared('bgwriter') will zero all the counters shown in the pg_stat_bgwriter view.
pg_stat_reset_single_table_counters(oid)	void	Reset statistics for a single table or index in the current database to zero (requires superuser privileges)
pg_stat_reset_single_function_counters(oid)	void	Reset statistics for a single function in the current database to zero (requires superuser privileges)

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 统计信息采集进程

- 在事务中获取统计信息时, 仅获取一次, 除非使用pg_stat_clear_snapshot()丢弃这个镜像.
  - SESSION A :
  - digoal=# begin;
  - BEGIN
  - digoal=# select last_analyze from pg_stat_all_tables where relname ='test';
  - last_analyze
  - 2014-02-20 14:04:32.716723+08
  - SESSION B :
  - digoal=# analyze test;
  - SESSION A :
  - digoal=# select last_analyze from pg_stat_all_tables where relname ='test';
  - last_analyze
  - 2014-02-20 14:04:32.716723+08
  - digoal=# select pg_stat_clear_snapshot();
  - digoal=# select last_analyze from pg_stat_all_tables where relname ='test';
  - last_analyze
  - ------------------------------
  - 2014-02-20 15:53:09.52539+08
  - (1 row)

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 统计信息采集进程

- 查看或重置统计信息的函数
- 查看会话 | 服务端进程级别统计信息的函数, 根据进程ID查询统计信息.

Table 27-14. Per-Backend Statistics Functions

Function	Return Type	Description
pg_stat_get_backend_idset()	setof integer	Set of currently active backend ID numbers (from 1 to the number of active backends)
pg_stat_get_backend_activity(integer)	text	Text of this backend's most recent query
pg_stat_get_backend_activity_start(integer)	timestamp with time zone	Time when the most recent query was started
pg_stat_get_backend_client_addr(integer)	inet	IP address of the client connected to this backend
pg_stat_get_backend_client_port(integer)	integer	TCP port number that the client is using for communication
pg_stat_get_backend_dbid(integer)	oid	OID of the database this backend is connected to
pg_stat_get_backend_pid(integer)	integer	Process ID of this backend
pg_stat_get_backend_start(integer)	timestamp with time zone	Time when this process was started
pg_stat_get_backend_userid(integer)	oid	OID of the user logged into this backend
pg_stat_get_backend_waiting(integer)	boolean	True if this backend is currently waiting on a lock
pg_stat_get_backend_xact_start(integer)	timestamp with time zone	Time when the current transaction was started

SKY mobi
新手机 新应用 新娱乐

监控数据库活动

- 使用举例1, 语句级的资源开销统计.　参考 man getrusage, src/backend/tcop/postgres.c
- digoal=# set client_min_messages=log;
- SET
- digoal=# set log_statement_stats=on;
- SET
- digoal=# select count(*) from iso_test;
- LOG: QUERY STATISTICS
- DETAIL: ! system usage stats:
- !　　0.001057 elapsed 0.000000 user 0.001000 system sec
- !　　[0.001999 user 0.001999 sys total]
- !　　0/0 [0/0] filesystem blocks in/out
- !　　0/226 [0/1050] page faults/reclaims, 0 [0] swaps
- !　　0 [0] signals rcvd, 0/0 [0/0] messages rcvd/sent
- !　　0/0 [6/0] voluntary/involuntary context switches
- count
- -------
- 1000
- (1 row)

SKY mobi
新手机 新应用 新娱乐

监控数据库活动

```
struct rusage {
        struct timeval ru_utime; /* user time used */
        struct timeval ru_stime; /* system time used */
        long   ru_maxrss;        /* maximum resident set size */
        long   ru_ixrss;         /* integral shared memory size */
        long   ru_idrss;         /* integral unshared data size */
        long   ru_isrss;         /* integral unshared stack size */
        long   ru_minflt;        /* page reclaims */
        long   ru_majflt;        /* page faults */
        long   ru_nswap;         /* swaps */
        long   ru_inblock;       /* block input operations */
        long   ru_oublock;       /* block output operations */
        long   ru_msgsnd;        /* messages sent */
        long   ru_msgrcv;        /* messages received */
        long   ru_nsignals;      /* signals received */
        long   ru_nvcsw;         /* voluntary context switches */
        long   ru_nivcsw;        /* involuntary context switches */
    };
```

SKY mobi
新手机 新应用 新娱乐

监控数据库活动

- src/backend/tcop/postgres.c
- ShowUsage(const char *title)
- * the only stats we don't show here are for memory usage -- i can't
- * figure out how to interpret the relevant fields in the rusage struct,
- * and they change names across o/s platforms, anyway. if you can figure
- * out what the entries mean, you can somehow extract resident set size,
- * shared text size, and unshared data and stack sizes.
- appendStringInfoString(&str, "! system usage stats:\n");
- appendStringInfo(&str,
- "!\t%ld.%06ld elapsed %ld.%06ld user %ld.%06ld system sec\n",
- (long) (elapse_t.tv_sec - Save_t.tv_sec),
- (long) (elapse_t.tv_usec - Save_t.tv_usec),
- (long) (r.ru_utime.tv_sec - Save_r.ru_utime.tv_sec),
- (long) (r.ru_utime.tv_usec - Save_r.ru_utime.tv_usec),
- (long) (r.ru_stime.tv_sec - Save_r.ru_stime.tv_sec),
- (long) (r.ru_stime.tv_usec - Save_r.ru_stime.tv_usec));
- appendStringInfo(&str,
- "!\t[%ld.%06ld user %ld.%06ld sys total]\n",
-

SKY mobi
新手机 新应用 新娱乐

监控数据库活动

```
                (long) user.tv_sec,
                (long) user.tv_usec,
                (long) sys.tv_sec,
                (long) sys.tv_usec);
#if defined(HAVE_GETRUSAGE)
        appendStringInfo(&str,
                "!\t%ld/%ld [%ld/%ld] filesystem blocks in/out\n",
                r.ru_inblock - Save_r.ru_inblock,
        /* they only drink coffee at dec */
                r.ru_oublock - Save_r.ru_oublock,
                r.ru_inblock, r.ru_oublock);
        appendStringInfo(&str,
                "!\t%ld/%ld [%ld/%ld] page faults/reclaims, %ld [%ld] swaps\n",
                r.ru_majflt - Save_r.ru_majflt,
                r.ru_minflt - Save_r.ru_minflt,
                r.ru_majflt, r.ru_minflt,
                r.ru_nswap - Save_r.ru_nswap,
                r.ru_nswap);
```

SKY mobi
新手机 新应用 新娱乐

监控数据库活动

```
        appendStringInfo(&str,
                "!\t%ld [%ld] signals rcvd, %ld/%ld [%ld/%ld] messages rcvd/sent\n",
                                r.ru_nsignals - Save_r.ru_nsignals,
                                r.ru_nsignals,
                                r.ru_msgrcv - Save_r.ru_msgrcv,
                                r.ru_msgsnd - Save_r.ru_msgsnd,
                                r.ru_msgrcv, r.ru_msgsnd);
         appendStringInfo(&str,
                        "!\t%ld/%ld [%ld/%ld] voluntary/involuntary context switches\n",
                                r.ru_nvcsw - Save_r.ru_nvcsw,
                                r.ru_nivcsw - Save_r.ru_nivcsw,
                                r.ru_nvcsw, r.ru_nivcsw);
#endif   /* HAVE_GETRUSAGE */
```

SKY mobi
新手机 新应用 新娱乐

监控数据库活动

- 使用举例2, 查看当前活动的会话信息(会话的启动时间, 事务的启动时间, 当前正在执行的SQL等)
- digoal=# select * from pg_stat_activity where state<>'idle' ; -- and pid<>pg_backend_pid();
- -[RECORD 1]----+---
- datid | 16384
- datname | digoal
- pid | 22164
- usesysid | 10
- usename | postgres
- application_name | psql
- client_addr |
- client_hostname |
- client_port | -1
- backend_start | 2013-12-17 15:35:09.22653+08
- xact_start | 2013-12-17 15:36:35.868056+08
- query_start | 2013-12-17 15:36:35.868056+08
- state_change | 2013-12-17 15:36:35.868061+08
- waiting | f
- state | active
- query | select * from pg_stat_activity where state<>'idle';

SKY mobi
新手机 新应用 新娱乐

监控数据库活动

- 使用举例3, 查看数据库的SQL级别统计信息, 按耗时倒序输出, 调用次数倒序输出, 单次SQL执行时间倒序输出.
- digoal=# create extension pg_stat_statements;
- CREATE EXTENSION
- digoal=# \q
- pg93@db-172-16-3-150-> cd $PGDATA
- pg93@db-172-16-3-150-> vi postgresql.conf
- shared_preload_libraries = 'pg_stat_statements'
- pg_stat_statements.max = 1000
- pg_stat_statements.track = all
- pg93@db-172-16-3-150-> pg_ctl restart -m fast

- 按SQL总耗时倒序输出

SKYY mobi
新手机 新应用 新娱乐

监控数据库活动

- digoal=# select * from pg_stat_statements order by total_time desc limit 1 offset 0;
- -[RECORD 1]-------+---
- userid | 10
- dbid | 16384
- query | insert into kuaidi_log1 values ($1, now(), ?, ?);
- calls | 100000000
- total_time | 1361916.61973994
- rows | 100000000
- shared_blks_hit | 102272487
- shared_blks_read | 1136660
- shared_blks_dirtied | 1136379
- shared_blks_written | 9385
- local_blks_hit | 0
- local_blks_read | 0
- local_blks_dirtied | 0
- local_blks_written | 0
- temp_blks_read | 0
- temp_blks_written | 0
- blk_read_time | 0 -- 以下两项需要开启trace_io_timing
- blk_write_time | 0

SKY mobi
新手机 新应用 新娱乐

监控数据库活动

- 调用次数倒序输出
- select * from pg_stat_statements order by calls desc limit 1 offset 0;

- 单次SQL执行时间倒序输出
- select * from pg_stat_statements order by total_time/calls desc limit 1 offset 0;

- 按shared buffer "未命中块读" 倒序输出
- select * from pg_stat_statements order by shared_blks_read desc limit 1 offset 0;

SKY mobi
新手机 新应用 新娱乐

监控数据库活动

- 每天以邮件形式发送CPU time TOP 20的SQL统计结果
- vi /home/postgres/script/report.sh
- #!/bin/bash

- export PGPORT=1921
- export PGDATA=/data01/pgdata/1921/pg_root
- export LANG=en_US.utf8
- export PGHOME=/opt/pgsql
- export LD_LIBRARY_PATH=$PGHOME/lib:/lib64:/usr/lib64:/usr/local/lib64:/lib:/usr/lib:/usr/local/lib
- export DATE=`date +"%Y%m%d%H%M"`
- export PATH=$PGHOME/bin:$PATH:.
- export PGHOST=$PGDATA
- export PGDATABASE=postgres

- psql -A -x -c "select row_number() over() as rn, * from (select query,' calls:'||calls||' total_time_s:'||round(total_time::numeric,2)||' avg_time_ms:'||round(1000*(total_time::numeric/calls),2) as stats from pg_stat_statements order by total_time desc limit 20) t;" >/tmp/stat_query.log 2>&1
- echo -e "$DATE avcp TOP20 query report yest"|mutt -s "$DATE avcp TOP20 query report yest" -a /tmp/stat_query.log digoal@126.com
- psql -c "select pg_stat_statements_reset()"
- # 9.2以及以上版本的total_time使用毫秒单位. 所以不需要乘以1000.
- crontab -e
- 1 8 * * * /home/postgres/script/report.sh

SKY mobi
新手机 新应用 新娱乐

监控数据库活动

- 使用举例4, 查看bgwriter的统计信息
- backend 的buffer 写过于频繁说明shared buffer不够用, 或者是bgwriter的sleep time太长需要调整,
- 让bgwriter多干点活(bgwriter_delay, bgwriter_lru_maxpages, bgwriter_lru_multiplier)或者调大shared buffer.
- http://www.postgresql.org/docs/9.3/static/runtime-config-resource.html#RUNTIME-CONFIG-RESOURCE-BACKGROUND-WRITER

Table 27-3. `pg_stat_bgwriter` View

Column	Type	Description
checkpoints_timed	bigint	Number of scheduled checkpoints that have been performed
checkpoints_req	bigint	Number of requested checkpoints that have been performed
checkpoint_write_time	double precision	Total amount of time that has been spent in the portion of checkpoint processing where files are written to disk, in milliseconds
checkpoint_sync_time	double precision	Total amount of time that has been spent in the portion of checkpoint processing where files are synchronized to disk, in milliseconds
buffers_checkpoint	bigint	Number of buffers written during checkpoints
buffers_clean	bigint	Number of buffers written by the background writer
maxwritten_clean	bigint	Number of times the background writer stopped a cleaning scan because it had written too many buffers
buffers_backend	bigint	Number of buffers written directly by a backend
buffers_backend_fsync	bigint	Number of times a backend had to execute its own fsync call (normally the background writer handles those even when the backend does its own write)
buffers_alloc	bigint	Number of buffers allocated
stats_reset	timestamp with time zone	Time at which these statistics were last reset

SKY mobi
新手机 新应用 新娱乐

监控数据库活动

- 使用举例5, 查看数据库级统计信息, 如数据库的 事务提交次数, 回滚次数, 未命中数据块读, 命中读, 行的统计信息(扫描, 输出,插入,更新,删除), 临时文件, 死锁, IOTIME等统计信息.

- 数据行的扫描和输出的区别是, 扫描的行不一定输出.
- 例如.
- digoal=# explain select count(*) from iso_test;
- QUERY PLAN
- ---
- Aggregate (cost=25.38..25.39 rows=1 width=0)
- -> Seq Scan on iso_test (cost=0.00..22.30 rows=1230 width=0)
- (2 rows)
- 扫描1230行, 输出1行.
- tup_returned 指扫描
- tup_fetched 指输出

SKY mobi
新手机 新应用 新娱乐

监控数据库活动

- digoal=# create table tbl(id int);
- CREATE TABLE
- digoal=# insert into tbl select generate_series(1,10000);
- INSERT 0 10000
- digoal=# analyze tbl;
- ANALYZE
- 重置统计信息, 便于查看
- digoal=# select pg_stat_reset();
- pg_stat_reset
- ---
-
- (1 row)
- digoal=# select tup_returned,tup_fetched from pg_stat_database where datname ='digoal';
- tup_returned | tup_fetched
- ---+---
- 0 | 0
- (1 row)

SKY mobi
新手机 新应用 新娱乐

监控数据库活动

- 以下SQL无索引, 全表扫描
- digoal=# select * from tbl where id<5;
- id
- ----
- 1
- 2
- 3
- 4
- (4 rows)
- 扫描1万行, 输出4行.

- digoal=# select tup_returned,tup_fetched from pg_stat_database where datname ='digoal';
- tup_returned | tup_fetched
- --------------+-------------
- 10005 | 5
- (1 row)

SKY mobi
新手机 新应用 新娱乐

监控数据库活动

able 27-4. pg_stat_database View

Column	Type	Description
datid	oid	OID of a database
datname	name	Name of this database
numbackends	integer	Number of backends currently connected to this database. This is the only column in this view that returns a value reflecting current state; all other columns return the accumulated values since the last reset.
xact_commit	bigint	Number of transactions in this database that have been committed
xact_rollback	bigint	Number of transactions in this database that have been rolled back
blks_read	bigint	Number of disk blocks read in this database
blks_hit	bigint	Number of times disk blocks were found already in the buffer cache, so that a read was not necessary (this only includes hits in the PostgreSQL buffer cache, not the operating system's file system cache)
tup_returned	bigint	Number of rows returned by queries in this database
tup_fetched	bigint	Number of rows fetched by queries in this database
tup_inserted	bigint	Number of rows inserted by queries in this database
tup_updated	bigint	Number of rows updated by queries in this database
tup_deleted	bigint	Number of rows deleted by queries in this database
conflicts	bigint	Number of queries canceled due to conflicts with recovery in this database. (Conflicts occur only on standby servers; see pg_stat_database_conflicts for details.)
temp_files	bigint	Number of temporary files created by queries in this database. All temporary files are counted, regardless of why the temporary file was created (e.g., sorting or hashing), and regardless of the log_temp_files setting.
temp_bytes	bigint	Total amount of data written to temporary files by queries in this database. All temporary files are counted, regardless of why the temporary file was created, and regardless of the log_temp_files setting.
deadlocks	bigint	Number of deadlocks detected in this database
blk_read_time	double precision	Time spent reading data file blocks by backends in this database, in milliseconds
blk_write_time	double precision	Time spent writing data file blocks by backends in this database, in milliseconds
stats_reset	timestamp with time zone	Time at which these statistics were last reset

SKY mobi
新手机 新应用 新娱乐

监控数据库活动

- 使用举例6, 查看表级统计信息, 区分全表扫描和索引扫描的次数和输出的行, 以及DML的行数, 评估的当前活动行数和垃圾行数, vacuum和analyze相关的统计信

able 27-5. pg_stat_all_tables View

Column	Type	Description
relid	oid	OID of a table
schemaname	name	Name of the schema that this table is in
relname	name	Name of this table
seq_scan	bigint	Number of sequential scans initiated on this table
seq_tup_read	bigint	Number of live rows fetched by sequential scans
idx_scan	bigint	Number of index scans initiated on this table
idx_tup_fetch	bigint	Number of live rows fetched by index scans
n_tup_ins	bigint	Number of rows inserted
n_tup_upd	bigint	Number of rows updated
n_tup_del	bigint	Number of rows deleted
n_tup_hot_upd	bigint	Number of rows HOT updated (i.e., with no separate index update required)
n_live_tup	bigint	Estimated number of live rows
n_dead_tup	bigint	Estimated number of dead rows
last_vacuum	timestamp with time zone	Last time at which this table was manually vacuumed (not counting VACUUM FULL)
last_autovacuum	timestamp with time zone	Last time at which this table was vacuumed by the autovacuum daemon
last_analyze	timestamp with time zone	Last time at which this table was manually analyzed
last_autoanalyze	timestamp with time zone	Last time at which this table was analyzed by the autovacuum daemon
vacuum_count	bigint	Number of times this table has been manually vacuumed (not counting VACUUM FULL)
autovacuum_count	bigint	Number of times this table has been vacuumed by the autovacuum daemon
analyze_count	bigint	Number of times this table has been manually analyzed
autoanalyze_count	bigint	Number of times this table has been analyzed by the autovacuum daemon

SKY mobi
新手机 新应用 新娱乐

监控数据库活动

- 使用举例7, 查看索引相关的统计信息, 例如该索引被使用的次数, 从这个索引读的行数和返回的行数.

Table 27-6. pg_stat_all_indexes View

Column	Type	Description
relid	oid	OID of the table for this index
indexrelid	oid	OID of this index
schemaname	name	Name of the schema this index is in
relname	name	Name of the table for this index
indexrelname	name	Name of this index
idx_scan	bigint	Number of index scans initiated on this index
idx_tup_read	bigint	Number of index entries returned by scans on this index
idx_tup_fetch	bigint	Number of live table rows fetched by simple index scans using this index

SKY mobi
新手机 新应用 新娱乐

监控数据库活动

- 使用举例8, 表的IO级统计信息, 如heap主存储的块读(区分未命中shared buffer和命中shared buffer的统计)
- 这个表上的所有索引的块读统计总和 (区分未命中shared buffer和命中shared buffer的统计)
- 这个表上的TOAST表以及TOAST索引的块读(区分未命中shared buffer和命中shared buffer的统计)
- (查看这个视图的定义比较容易理解)

Table 27-7. pg_statio_all_tables View

Column	Type	Description
relid	oid	OID of a table
schemaname	name	Name of the schema that this table is in
relname	name	Name of this table
heap_blks_read	bigint	Number of disk blocks read from this table
heap_blks_hit	bigint	Number of buffer hits in this table
idx_blks_read	bigint	Number of disk blocks read from all indexes on this table
idx_blks_hit	bigint	Number of buffer hits in all indexes on this table
toast_blks_read	bigint	Number of disk blocks read from this table's TOAST table (if any)
toast_blks_hit	bigint	Number of buffer hits in this table's TOAST table (if any)
tidx_blks_read	bigint	Number of disk blocks read from this table's TOAST table index (if any)
tidx_blks_hit	bigint	Number of buffer hits in this table's TOAST table index (if any)

SKY mobi
新手机 新应用 新娱乐

监控数据库活动

- 使用举例9, 索引的IO级统计信息,
- 索引的块读(区分未命中shared buffer和命中shared buffer的统计)

Table 27-8. `pg_statio_all_indexes` View

Column	Type	Description
`relid`	`oid`	OID of the table for this index
`indexrelid`	`oid`	OID of this index
`schemaname`	`name`	Name of the schema this index is in
`relname`	`name`	Name of the table for this index
`indexrelname`	`name`	Name of this index
`idx_blks_read`	`bigint`	Number of disk blocks read from this index
`idx_blks_hit`	`bigint`	Number of buffer hits in this index

SKY mobi
新手机 新应用 新娱乐

监控数据库活动

- 使用举例10, 序列的IO级统计信息,
- 序列的块读(区分未命中shared buffer和命中shared buffer的统计)
- PostgreSQL中序列存储在表中.
- digoal=# create sequence seq2;
- digoal=# select * from seq2;
- -[RECORD 1]-+--------------------
- sequence_name | seq2
- last_value | 1
- start_value | 1
- increment_by | 1
- max_value | 9223372036854775807
- min_value | 1
- cache_value | 1
- log_cnt | 0
- is_cycled | f
- is_called | f(1 row)

Table 27-9. `pg_statio_all_sequences` View

Column	Type	Description
`relid`	`oid`	OID of a sequence
`schemaname`	`name`	Name of the schema this sequence is in
`relname`	`name`	Name of this sequence
`blks_read`	`bigint`	Number of disk blocks read from this sequence
`blks_hit`	`bigint`	Number of buffer hits in this sequence

SKY mobi
新手机 新应用 新娱乐

监控数据库活动

- 使用举例11, 函数的统计信息, 调用次数, 总的时间开销.
- 必须要先打开track_functions参数.

Table 27-10. `pg_stat_user_functions` View

Column	Type	Description
`funcid`	`oid`	OID of a function
`schemaname`	`name`	Name of the schema this function is in
`funcname`	`name`	Name of this function
`calls`	`bigint`	Number of times this function has been called
`total_time`	`double precision`	Total time spent in this function and all other functions called by it, in milliseconds
`self_time`	`double precision`	Total time spent in this function itself, not including other functions called by it, in milliseconds

SKYY mobi
新手机 新应用 新娱乐

监控数据库活动

- 使用举例12, 流复制的相关统计信息, 在上游节点查询, 显示直连的下游节点的统计信息.
- 每个standby节点一条记录, 包含wal的发送位置, 对端的接收位置, 写位置, flush位置, apply位置.

able 27-11. pg_stat_replication View

Column	Type	Description
pid	integer	Process ID of a WAL sender process
usesysid	oid	OID of the user logged into this WAL sender process
usename	name	Name of the user logged into this WAL sender process
application_name	text	Name of the application that is connected to this WAL sender
client_addr	inet	IP address of the client connected to this WAL sender. If this field is null, it indicates that the client is connected via a Unix socket on the server machine.
client_hostname	text	Host name of the connected client, as reported by a reverse DNS lookup of client_addr. This field will only be non-null for IP connections, and only when log_hostname is enabled.
client_port	integer	TCP port number that the client is using for communication with this WAL sender, or -1 if a Unix socket is used
backend_start	timestamp with time zone	Time when this process was started, i.e., when the client connected to this WAL sender
state	text	Current WAL sender state
sent_location	text	Last transaction log position sent on this connection
write_location	text	Last transaction log position written to disk by this standby server
flush_location	text	Last transaction log position flushed to disk by this standby server
replay_location	text	Last transaction log position replayed into the database on this standby server
sync_priority	integer	Priority of this standby server for being chosen as the synchronous standby
sync_state	text	Synchronous state of this standby server

SKYY mobi
新手机 新应用 新娱乐

监控数据库活动

- postgres=# select * from pg_stat_replication ;
- -[RECORD 1]----+------------------------------
- pid | 24085
- usesysid | 16391
- usename | replica
- application_name | walreceiver
- client_addr | 192.168.xxx.xxx
- client_hostname |
- client_port | 52568
- backend_start | 2014-02-11 16:54:57.788442+08
- state | streaming
- sent_location | 85F/F9F48EC0
- write_location | 85F/F9F48EC0
- flush_location | 85F/F9F48EC0
- replay_location | 85F/F9F48EC0
- sync_priority | 0
- sync_state | async

SKY mobi
新手机 新应用 新娱乐

监控数据库活动

- 使用举例13, standby的冲突统计视图,
- standby在apply xlog时可能与正在standby执行的SQL发生冲突.
- 例如查询的表在wal信息中有truncate的信息, 那么就发生冲突了. 冲突可以通过几个配置参数来决定是否需要cancel 冲突的sql.
- max_standby_archive_delay, max_standby_streaming_delay (catching up (sent loc=write loc)后的允许时长)
- http://www.postgresql.org/docs/9.3/static/hot-standby.html#HOT-STANDBY-CONFLICT

able 27-12. pg_stat_database_conflicts View

Column	Type	Description
datid	oid	OID of a database
datname	name	Name of this database
confl_tablespace	bigint	Number of queries in this database that have been canceled due to dropped tablespaces
confl_lock	bigint	Number of queries in this database that have been canceled due to lock timeouts
confl_snapshot	bigint	Number of queries in this database that have been canceled due to old snapshots
confl_bufferpin	bigint	Number of queries in this database that have been canceled due to pinned buffers
confl_deadlock	bigint	Number of queries in this database that have been canceled due to deadlocks

SKY mobi
新手机 新应用 新娱乐

OS层面的统计信息

- 例如sar出来的信息, 包括
- 系统的每秒读写IO请求
- 每秒系统写入或从磁盘读出的page数
- 系统每秒创建的进程数, 如果这个数字很大可能是应用程序连接数据库是短连接, 并且请求数据库频繁, 而PostgreSQL采用的是客户端连接过来fork新进程然后这个新进程与客户端进行交互的模式, 因此这种情况会造成数据库服务器大量的关闭和创建进程, sar -c能反映这种情况. 使用短连接还有一个坏处就是当系统中使用到sequence, 并且这个sequence被大量的这种短连接进程请求, 那么它设置的sequence cache没有效果并且会造成大量的跳号.
- 系统的负载
- 系统的内存和SWAP的使用情况
- 每秒被free的内存, 新增给buffer的内存, 新增给cache的内存
- 系统CPU在user, nice, system, iowait, steal, idle的使用占比
- inode, file 或其他内核表的报告
- 每秒上下文的切换数量
- SWAP每秒被换进或换出的数量

SKY mobi
新手机 新应用 新娱乐

OS层面的统计信息

- http://blog.163.com/digoal@126/blog/static/16387704020121135414570l/
- 将服务器每天产生的sar统计信息录入到数据库, 输出各sar维度的TOP10.
- 输出样本

```
---- DailyAvgValue TOP10: ----


1. ldavg_15 TOP10 :
```

get_info	get_ip	ldavg_15
[redacted]	192.[redacted]	4.27
[redacted]	192.[redacted]	2.84
[redacted]	192.[redacted]	2.69
[redacted]	192.[redacted]	2.55
[redacted]	192.[redacted]	1.82
[redacted]	192.[redacted]	1.65
[redacted]	192.[redacted]	1.52
[redacted]	192.[redacted]	1.46
[redacted]	192.[redacted]	1.41
[redacted]	192.[redacted]	1.41

(10 rows)

SKY mobi
新手机 新应用 新娱乐

监控数据库活动-nagios

- Nagios监控配置
- http://blog.163.com/digoal@126/blog/static/163877040201353133543 83/
- http://blog.163.com/digoal@126/blog/static/163877040201353341575 31/
- http://blog.163.com/digoal@126/blog/static/163877040201355625455 36

- nagios插件
- check_postgres
- http://bucardo.org/check_postgres/check_postgres.pl.html
- 或者自定义脚本
- http://blog.163.com/digoal@126/blog/static/163877040201412763135184/

SKY mobi
新手机 新应用 新娱乐

监控数据库活动-历史统计

- PostgreSQL第三方插件
- 例如 pgstatspack, pg_statsinfo, pgsnap
- 这些插件利用前面讲的那些统计信息表, 定时抓当前的统计镜像, 后期再根据需求输出分析报告.
- http://pgsnap.projects.pgfoundry.org/
- http://www.cybertec.at/postgresql_produkte/pgwatch-cybertec-enterprise-postgresql-monitor/
- http://pgstatsinfo.projects.pgfoundry.org/
- http://blog.163.com/digoal@126/blog/static/163877040201425856161 83/
- http://blog.163.com/digoal@126/blog/static/16387704020142502098 2/
- http://blog.163.com/digoal@126/blog/static/16387704020142511141 35/

- 自行定制执行计划
- 定时将统计数据插入历史记录表, 便于后期分析数据.
- 例如分析数据库的增量趋势, 表的增量趋势, XLOG的增量趋势, 各时间段的TPS, 增量等.

SKY mobi
新手机 新应用 新娱乐

监控数据库活动-实时监控

- 动态跟踪
- 跟踪每条SQL的IO
- http://blog.163.com/digoal@126/blog/static/16387704020139151152545 43/
- http://blog.163.com/digoal@126/blog/static/16387704020139152191581/
- 跟踪每条SQL或每个会话的网络流量
- http://blog.163.com/digoal@126/blog/static/16387704020139153195701/
- 事务, QUERY, 检查点, buffer, 对象读写, 锁, xlog, 排序, 等探针用法
- http://blog.163.com/digoal@126/blog/static/16387704020139168401271 3/
- http://blog.163.com/digoal@126/blog/static/16387704020139161011173 67/
- http://blog.163.com/digoal@126/blog/static/16387704020139162245922 1/
- http://blog.163.com/digoal@126/blog/static/16387704020139164887 61/
- http://blog.163.com/digoal@126/blog/static/16387704020139165361610 3/
- http://blog.163.com/digoal@126/blog/static/16387704020139167492287 9/
- http://blog.163.com/digoal@126/blog/static/16387704020139162215 18/

SKY mobi
新手机 新应用 新娱乐

练习

- 了解各种维度的统计信息的解读(table, index, database, replication, sql...), 锁信息解读, 磁盘使用, 活动信息, 历史数据库活动统计报告
- pg_top 的部署
- nagios 的部署
- sar 统计信息收集和报告部署

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 日常维护和检查

- 日常维护
- http://www.postgresql.org/docs/9.3/static/maintenance.html

- 全局自动vacuum,
  - 开启postgresql.conf - autovacuum=on
- 表级计划性的vacuum,
  - 查询可能需要vacuum的表, 即表dead tuple的量或比例, 默认情况下可能有少于20%的dead tuple. autovacuum_vacuum_scale_factor=0.2
  - digoal=# select relname,n_live_tup,n_dead_tup from pg_stat_all_tables where n_dead_tup<>0 order by n_dead_tup desc;
  - relname | n_live_tup | n_dead_tup
  - ------------------+------------+------------
  - pg_depend | 6864 |
  - pg_proc | 2586 | 374
  - pg_class | 429 | 128
  - pg_attribute | 2958 | 117
  - pg_operator | 737 | 90
  - pg_type | 465 | 89
  - pg_amop | 414 | 59
  - 手工执行或者等待autovacuum进程执行.

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 日常维护和检查

- 膨胀表的维护,
  - postgres=# create extension pgstattuple;
  - CREATE EXTENSION
  - postgres=# select oid::regclass,(pgstattuple(oid)).* from pg_class where relkind='r' order by free_space desc limit 1 offset 0;
  - -[RECORD 1]------+-----------
  - oid | pg_rewrite
  - table_len | 98304
  - tuple_count | 107
  - tuple_len | 84992
  - tuple_percent | 86.46
  - dead_tuple_count | 0
  - dead_tuple_len | 0
  - dead_tuple_percent | 0
  - free_space | 12260
  - free_percent | 12.47
  - 如果浪费的空间太大,并且确实想回收空间的话,可以适当的vacuum full; -- vacuum full 会重建表. 特别注意.
  - 或者使用pg_reorg重组表,可以减少排他锁的时间.
  - http://blog.163.com/digoal@126/blog/static/163877040201411205420775/

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 日常维护和检查

- 膨胀索引|不平衡索引的重建,
  - postgres=# select oid::regclass,(pgstattuple(oid)).* from pg_class where relkind='i' order by free_space desc limit 1 offset 0;
  - -[RECORD 1]------+-----------------------
  - oid | pg_depend_depender_index
  - table_len | 237568
  - tuple_count | 6290
  - tuple_len | 150960
  - tuple_percent | 63.54
  - dead_tuple_count | 1
  - dead_tuple_len | 24
  - dead_tuple_percent | 0.01
  - free_space | 43120
  - free_percent | 18.15
  - 如果浪费的空间太大,并且确实想回收空间的话.可以新建同样的索引,然后删除老的索引,建索引时可以选择CONCURRENTLY参数.

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 日常维护和检查

- prevent xid wrapped 的处理, 因为事务ID为32位循环使用的, 所以如果不做处理的话, 会出现数据"disappear"的现象.
- 为了防止数据disappear的现象, 数据库的vacuum操作将记录的事务ID改写为FrozenTransactionId, 这个ID视为比所有XID更早的ID.
- src/include/access/transam.h
- /* Special transaction ID values
- *
- * BootstrapTransactionId is the XID for "bootstrap" operations, and
- * FrozenTransactionId is used for very old tuples. Both should
- * always be considered valid.
- *
- * FirstNormalTransactionId is the first "normal" transaction id.
- * Note: if you need to change it, you must change pg_class.h as well.
- */
- #define InvalidTransactionId ((TransactionId) 0)
- #define BootstrapTransactionId ((TransactionId) 1)
- #define FrozenTransactionId ((TransactionId) 2)
- #define FirstNormalTransactionId ((TransactionId) 3)
- #define MaxTransactionId ((TransactionId) 0xFFFFFFFF)

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 日常维护和检查

- 和FREEZE动作几个相关参数：
- autovacuum_freeze_max_age = 1900000000 # maximum XID age before forced vacuum
- # (change requires restart)
- 这个参数表示如果表的年龄(pg_class.relfrozenxid)超过autovacuum_freeze_max_age , 即使未设置autovacuum参数, 也将自动强制对该表执行vacuum freeze . 从而降低表的年龄.

- vacuum_freeze_min_age = 50000000
- 这个参数表示vacuum在扫描数据块时, 允许保留的事务ID的年龄, 年龄大于vacuum_freeze_min_age 值的事务ID将被替换为FrozenXID .

- vacuum_freeze_table_age = 1500000000
- 如果表的年龄大于vacuum_freeze_table_age 那么vacuum操作将扫描全表, 因此可用于降低表的年龄, 表的年龄将降到vacuum_freeze_min_age 设置的值.

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 日常维护和检查

- 加深对这几个参数的印象
- digoal=# truncate tbl_freeze_test ;
- TRUNCATE TABLE
- digoal=# insert into tbl_freeze_test select generate_series(1,100000);
- INSERT 0 100000
- digoal=# set vacuum_freeze_min_age=10000000;
- SET
- digoal=# select pg_relation_filepath('tbl_freeze_test');
- pg_relation_filepath
- ----------
- pg_tblspc/66422/PG_9.3_201306121/16384/93056
- (1 row)

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 日常维护和检查

- 第一次还没有生成VM文件, 所以如果现在执行vacuum是会扫描全表的.
- pg93@db-172-16-3-150-> ll pg_tblspc/66422/PG_9.3_201306121/16384/93056*
- -rw------- 1 pg93 pg93 3.5M Dec 17 21:46 pg_tblspc/66422/PG_9.3_201306121/16384/93056
- -rw------- 1 pg93 pg93 24K Dec 17 21:46 pg_tblspc/66422/PG_9.3_201306121/16384/93056_fsm

- digoal=# vacuum verbose tbl_freeze_test;
- INFO: vacuuming "postgres.tbl_freeze_test"
- INFO: "tbl_freeze_test": found 0 removable, 100000 nonremovable row versions in 443 out of 443 pages
- DETAIL: 0 dead row versions cannot be removed yet.
- There were 0 unused item pointers.
- 0 pages are entirely empty.
- CPU 0.00s/0.01u sec elapsed 0.01 sec.
- VACUUM
- 未生成VM文件前, vacuum进程必须扫描全表.
- vm文件也就是为了减少VACUUM进程扫描开销而设计的.

PostgreSQL 日常维护和检查

- 此时的xmin是normal xmin
- digoal=# select min(xmin::text),max(xmin::text) from tbl_freeze_test limit 10;
- min | max
- ----------+----------
- 316732599 | 316732599
- (1 row)
- 同时在vacuum后自动生成了vm文件
- pg93@db-172-16-3-150-> ll pg_tblspc/66422/PG_9.3_201306121/16384/93056*
- -rw------- 1 pg93 pg93 3.5M Dec 17 21:46 pg_tblspc/66422/PG_9.3_201306121/16384/93056
- -rw------- 1 pg93 pg93 24K Dec 17 21:46 pg_tblspc/66422/PG_9.3_201306121/16384/93056_fsm
- -rw------- 1 pg93 pg93 8.0K Dec 17 21:46 pg_tblspc/66422/PG_9.3_201306121/16384/93056_vm

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 日常维护和检查

- digoal=# set vacuum_freeze_min_age=0;
- SET
- digoal=# vacuum verbose tbl_freeze_test;
- INFO: vacuuming "postgres.tbl_freeze_test"
- INFO: "tbl_freeze_test": found 0 removable, 0 nonremovable row versions in 0 out of 443 pages
- DETAIL: 0 dead row versions cannot be removed yet.
- There were 0 unused item pointers.
- 0 pages are entirely empty.
- CPU 0.00s/0.00u sec elapsed 0.00 sec.
- VACUUM
- 这里执行vacuum时不会扫描任何块, 所以xmin还是normal xid, 没有被改为frozenxid
- digoal=# select min(xmin::text),max(xmin::text) from tbl_freeze_test limit 10;
- min | max
- ----------+----------
- 316732599 | 316732599
- (1 row)

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 日常维护和检查

- digoal=# set vacuum_freeze_table_age = 0;
- SET
- digoal=# vacuum verbose tbl_freeze_test;
- INFO: vacuuming "postgres.tbl_freeze_test"
- INFO: "tbl_freeze_test": found 0 removable, 100000 nonremovable row versions in 443 out of 443 pages
- DETAIL: 0 dead row versions cannot be removed yet.
- There were 0 unused item pointers.
- 0 pages are entirely empty.
- CPU 0.00s/0.03u sec elapsed 0.03 sec.
- VACUUM
- vacuum_freeze_table_age=0的话是告诉VACUUM进程, 如果表的年龄超出0, 要扫描全表. (所以这里就达到目的了)
- digoal=# select min(xmin::text),max(xmin::text) from tbl_freeze_test limit 10;
- min | max
- -----+-----
- 2 | 2
- (1 row)

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 日常维护和检查

- 空闲时段的人为干预freeze (set vacuum_freeze_table_age = 0; vacuum table; 或者直接执行vacuum freeze tbl; 效果一样), 可以减少自动触发force whole table vacuum for prevent wrapped xid的概率.
- 因为自动触发如果发生在数据库繁忙节点, 会带来较大的IO性能影响.

- 查找年龄较老的表, 手工降低年龄.
- digoal=# select age(relfrozenxid),relname from pg_class where relkind='r' order by age(relfrozenxid) desc;
- vacuum freeze tablename;

- vacuum重要性以及如何定制vacuum计划
- http://blog.163.com/digoal@126/blog/static/16387704020141228245597 8/
- 数据"消失"实验
- http://blog.163.com/digoal@126/blog/static/16387704020118304315362 2/

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 日常维护和检查

- 日志文件的维护
  - 压缩保存
- 日志内容的检查
  - 日志的错误输出
  - 长SQL
  - 锁等待
  - 错误级别的日志
- 日志文件查询
  - 外部表
- http://www.postgresql.org/docs/9.3/static/runtime-config-logging.html#RUNTIME-CONFIG-LOGGING-CSVLOG
- http://www.postgresql.org/docs/9.3/static/file-fdw.html

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 日常维护和检查

```
CREATE TABLE postgres_log
(
  log_time timestamp(3) with time zone,
  user_name text,
  database_name text,
  process_id integer,
  connection_from text,
  session_id text,
  session_line_num bigint,
  command_tag text,
  session_start_time timestamp with time zone,
  virtual_transaction_id text,
  transaction_id bigint,
  error_severity text,
  sql_state_code text,
  message text,
  detail text,
```

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 日常维护和检查

- hint text,
- internal_query text,
- internal_query_pos integer,
- context text,
- query text,
- query_pos integer,
- location text,
- application_name text,
- PRIMARY KEY (session_id, session_line_num)
-);
- To import a log file into this table, use the COPY FROM command:

- COPY postgres_log FROM '/full/path/to/logfile.csv' WITH csv;
- 导入查询

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 日常维护和检查

- 外部表查询
- CREATE EXTENSION file_fdw;
- CREATE SERVER pglog FOREIGN DATA WRAPPER file_fdw;
- CREATE FOREIGN TABLE pglog (
- log_time timestamp(3) with time zone,
- user_name text,
- database_name text,
- process_id integer,
- connection_from text,
- session_id text,
- session_line_num bigint,
- command_tag text,
- session_start_time timestamp with time zone,
- virtual_transaction_id text,
- transaction_id bigint,

PostgreSQL 日常维护和检查

```
    error_severity text,
    sql_state_code text,
    message text,
    detail text,
    hint text,
    internal_query text,
    internal_query_pos integer,
    context text,
    query text,
    query_pos integer,
    location text,
    application_name text
) SERVER pglog
OPTIONS ( filename '/data/pgdata/pg_root/pg_log/postgresql-2013-12-20_000000.csv', format 'csv' );
```

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 日常维护和检查

- 检查集群是否正常
- 检查 standby的延迟(pg_stat_replication)

- PostgreSQL 数据库巡检模板
- http://blog.163.com/digoal@126/blog/static/163877040201425281649 7/

SKY mobi
新手机 新应用 新娱乐

练习

- 日志记录类型, 日志的处理, 表的维护, 数据的维护.