好玩的PostgreSQL企业特性

digoal

11/2/2015

自我介绍

- 
  一枚PGer

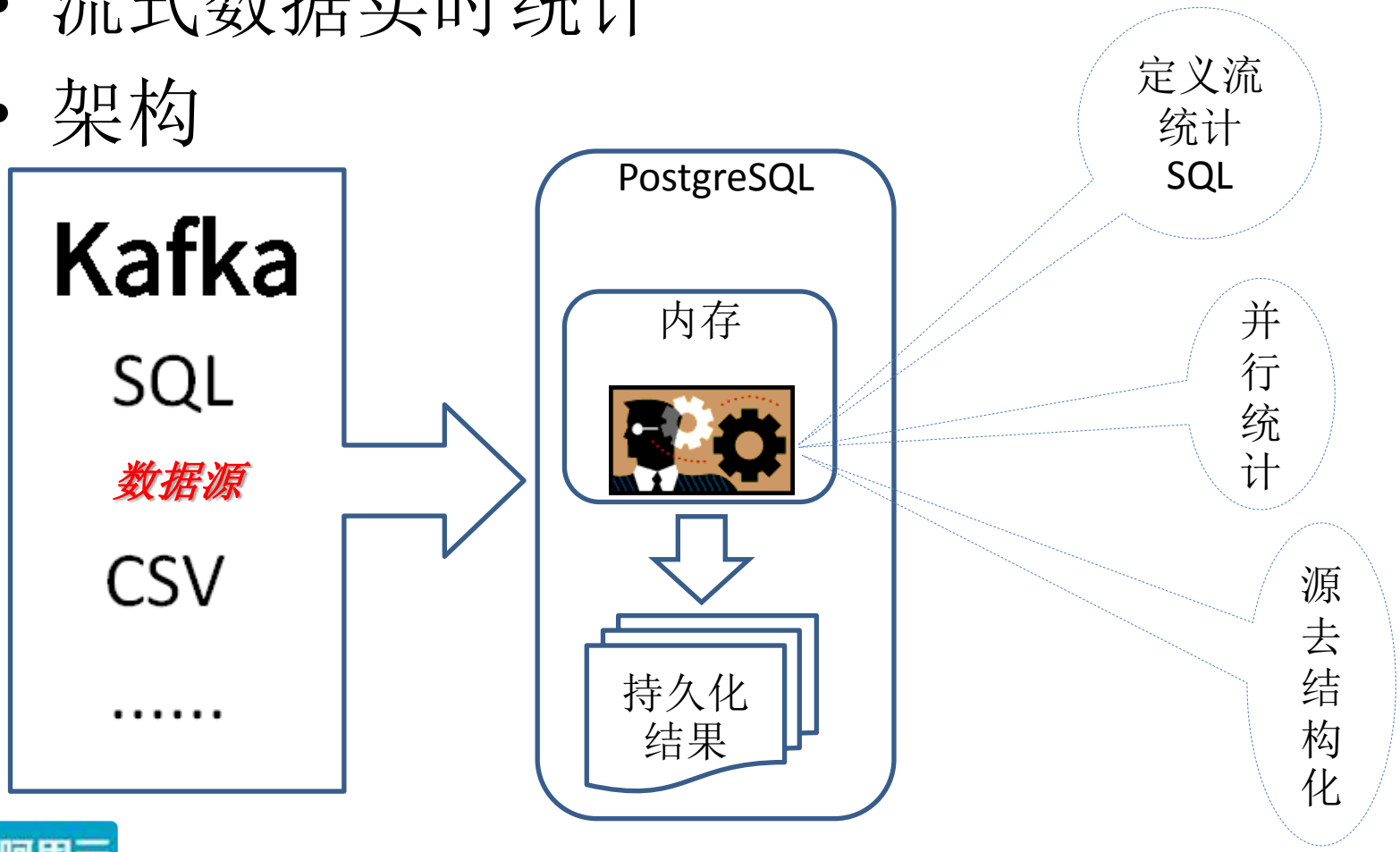
- PostgreSQL 中国社区发起人之一，社区CTO。
- PostgreSQL 中国社区杭州分会会长。
- PostgreSQL 中国社区大学发起人之一。
- DBA+社群联合发起人之一。
- 曾就职于斯凯网络，负责数据库部门，在上市前使用PostgreSQL完成去IOE任务，同时通过SOX审计，成功在纳斯达克上市。
- 现就职于阿里巴巴，RDS PG内核组。

目录

- 实时流式数据处理
- 数据挖掘(AGGFunc , PLR , PLPython , MADlib, UDF)
- 范围类型
- NoSQL特性(JSON,hstore类型)
- BRIN索引
- 部分索引
- 中文分词
- UDF
- 近似度查询
- 域
- 排他约束
- 递归查询
- 异构数据类型
- FDW
- 事件触发器
- 空间数据管理
- 数据预热
- advisory lock
- 流复制
- MPP(scale out)
- 安全
- GPU并行计算(scale up)
- 常见场景性能指标
- 用户案例

特性

- 流式数据实时统计
- 架构



应用场景

- 窗口：时间
- 维度：总和，总数，唯一值个数，最大值，最小值，平均值，众数，中位数，自定义百分位位点数。。。。。。

- 窗口：时间 + 传感器ID
- 维度：最低值，最高值，。。。。。。

- 窗口：时间 + 车辆ID
- 维度：最低时速，最高时速，平均时速，标准方差，噪点，轨迹，违章次数，。。。。。。油耗指数，刹车指数。。。。。。

- 金融行业实时计算

- 维度还可以结合自定义聚合函数，如: 柱状图，数据归类，MADlib，R 。。。。。。

应用场景

- CREATE CONTINUOUS VIEW name AS query
- query is a subset of PostgreSQL select :
- SELECT [DISTINCT [ON (expression [, ...])]]
  - expression [[AS] output_name] [, ...]
  - [FROM from_item [, ...]]
  - [WHERE condition]
  - [GROUP BY expression [, ...]] [HAVING condition [, ...]]
  - [WINDOW window_name AS (window_definition) [, ...]]

- where from_item can be one of:
  - stream_name [[AS] alias [(column_alias [, ...])]]
  - table_name [[AS] alias [(column_alias [, ...])]]
  - from_item [NATURAL] join_type from_item [ON join_condition]

应用场景

- 线性回归计算例子
- 实时计算每分钟y,x的截距，斜率。
- CREATE CONTINUOUS VIEW lreg AS
- SELECT date_trunc('minute', arrival_timestamp) AS minute,
- regr_slope(y::integer, x::integer) AS mx,
- regr_intercept(y, x) AS b
- FROM datapoints_stream GROUP BY minute;

- insert into lreg (arrival_timestamp,y,x,.....) values (.....);

应用场景

- 实时计算多少个GPS传感器位于某坐标方圆1000公里以内。
- -- PipelineDB ships natively with geospatial support
- CREATE CONTINUOUS VIEW sf_proximity_count AS
- SELECT COUNT(DISTINCT sensor_id::integer) FROM geo_stream WHERE ST_DWithin(
- -- Approximate SF coordinates
- ST_GeographyFromText('SRID=4326;POINT(37 -122)')::geometry, sensor_coords::geometry, 1000);

- insert into sf_proximity_count (sensor_id,sensor_coords,.....) values (.....);

应用场景

- 实时计算某WEB站点的请求延迟，90%低于多少毫秒，95%低于多少毫秒，99%低于多少毫秒。
- CREATE CONTINUOUS VIEW latency AS
- SELECT percentile_cont(array[90, 95, 99]) WITHIN GROUP (ORDER BY latency::integer) FROM latency_stream;

- insert into latency (latency,.....) values (.....);

性能指标

- CPU：E5-2650
- 每秒处理约600万行(5个维度)

特性

- 数据挖掘：AGGFunc , PLR , PLPython , MADlib ……

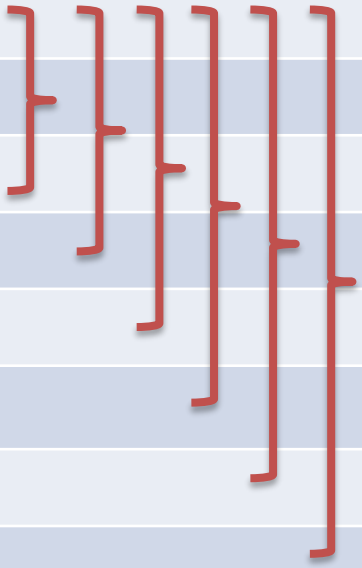
应用场景

- 举例
- p元线性回归
- y1=b0+b1x11+b2x12+…+bpx1p+ε1
- y2=b0+b1x21+b2x22+…+bpx2p+ε2
- ……
- 求截距，斜率。
- 预测yn
  - yn=b0+b1xn1+b2xn2+…+bpxnp+εn
- 公式
  - lm(y: 收盘价 ~ x1: 昨日收盘价 + x2: 昨日成交量, $DATA)

应用场景

- 某股票数据样本

日期	收盘价	昨日收盘价	昨日成交量	相关性
2015-10-08	?	x	x	计算数据线性相关性 自动选择相关性最高时的截距，斜率。 预测y。
2015-10-07	y	x	x	
2015-10-06	y	x	x	
2015-10-05	y	x	x	
2015-10-04	y	x	x	
2015-10-03	y	x	x	
2015-10-02	y	x	x	
2015-10-01	y	x	x	
……	y	x	x	



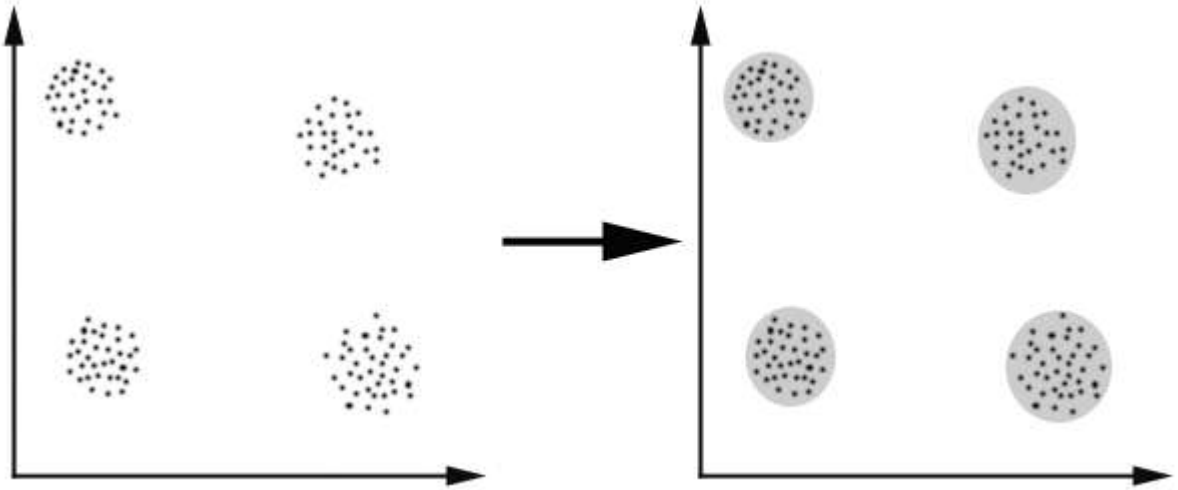
应用场景

- 预测股价

- http://blog.163.com/digoal@126/blog/static/16387704020152311265159 3/
- http://blog.163.com/digoal@126/blog/static/16387704020152512741921/

应用场景

- 数据聚集
- http://blog.163.com/digoal@126/blog/static/16387704020157174504812 1



特性

- 范围数据类型
  - intrange
  - iprange
  - ...range

应用场景

- 例子，快速范围查询，例如某个IP是否在某个IP地址段内
- 传统方法
- postgres=# create table tbl(id int,ip_start int8,ip_end int8);
- CREATE TABLE
- postgres=# create index idx_tbl on tbl using btree(ip_start,ip_end);
- CREATE INDEX

- 使用范围类型
- postgres=# create table tbl_r(id int,ip_range int8range);
- CREATE TABLE
- postgres=# create index idx_tbl_r on tbl_r using spgist(ip_range);
- CREATE INDEX
- 或
- postgres=# create index idx_tbl_r1 on tbl_r using gist(ip_range);
- CREATE INDEX

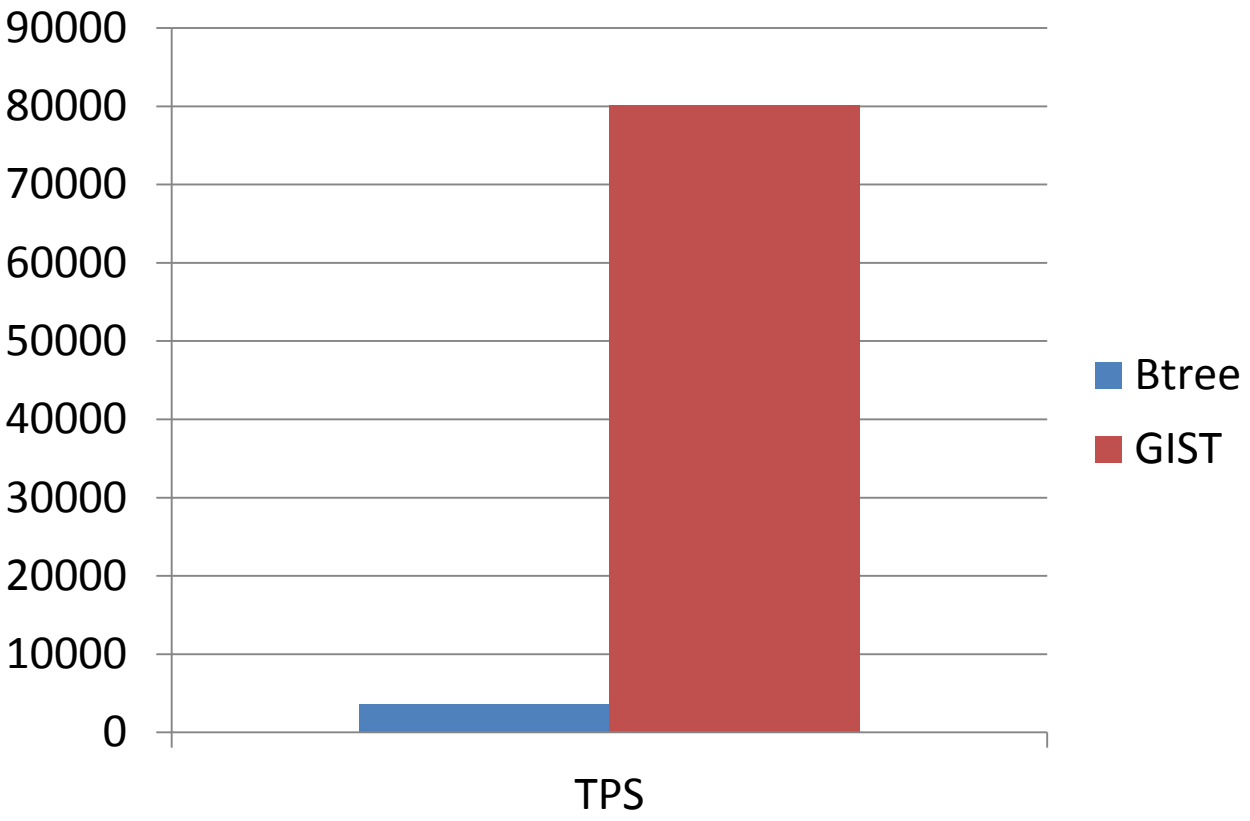
应用场景

- 传统范围查询
- postgres=# select * from tbl where ? between ip_start and ip_end;

- 使用范围类型的范围匹配操作符，利用gist/spgist索引
- postgres=# select * from tbl_r where ip_range @> ?

- OR 不改变原有数据结构，使用 函数索引
- create index idx on tbl using gist (**int8range(ip_start,ip_end+1)**);
- select * from tbl where int8range(ip_start,ip_end+1) @> ?;

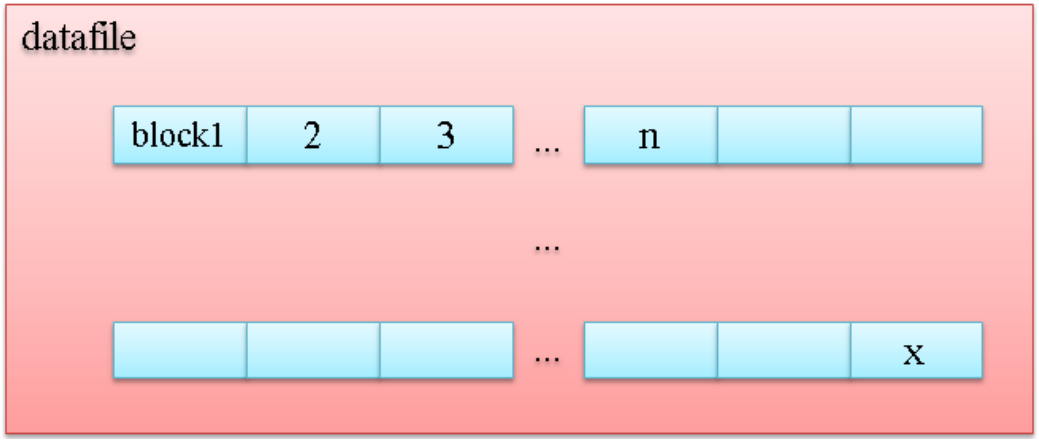
性能



- http://blog.163.com/digoal@126/blog/static/163877040201257010292 22/

特性

- BRIN(block range index)索引（lossy索引）
- btree
  - value1,ctid
  - value2,ctid
  - ……
  - big
  - full



- brin
  - blockid_left, blockid_right, value_min, value_max, allnull?, hasnull?
  - ……
  - small
  - lossy

应用场景

- 例子，流式大数据的快速范围检索

- 以时间字段为例，创建BRIN索引

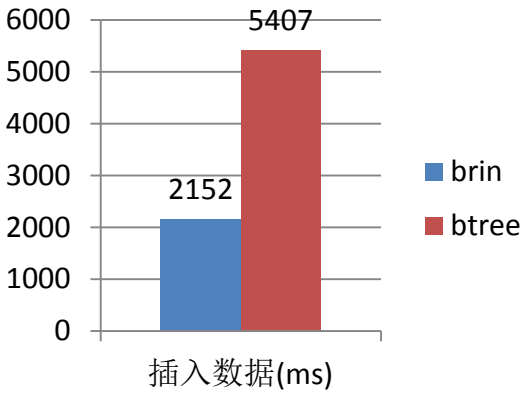
- BRIN索引
- block_range　value_min　value_max
- 1-127　mintime=? maxtime=?
- 128-255 mintime=? maxtime=?
- ... mintime=? maxtime=?

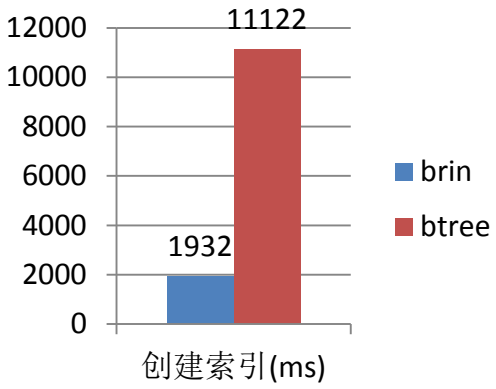
- 查询select * from tbl where crt_time between ? and ?;　or where crt_time = ?;
- 扫描符合条件的block，recheck条件。

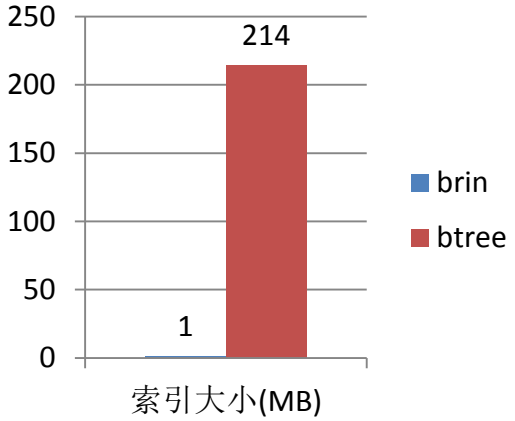
- 适合流式数据字段，不适合随机数据字段。
- 适合较大范围的检索，频繁的精确匹配检索不适合。

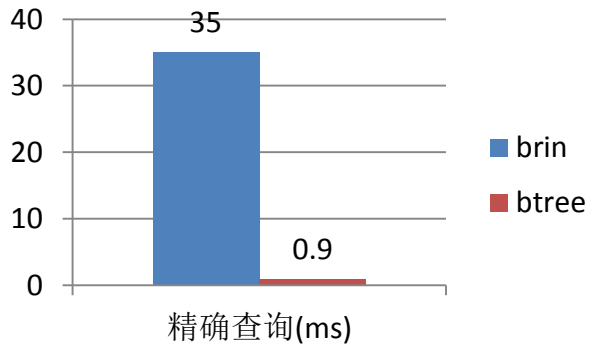
性能

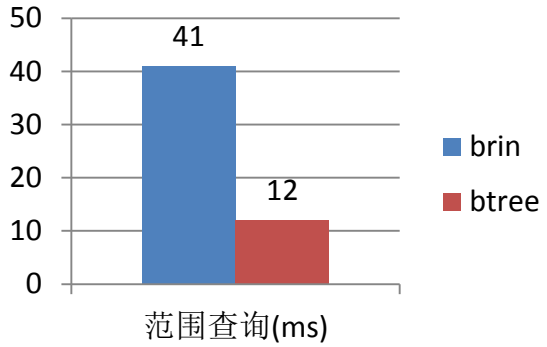
- 1000万记录为例，耗时(ms)比例











特性

- 部分索引

- 选择性差的值不进入索引
- 不被检索的值不进入索引

应用场景

- active字段表示用户的活跃度
- create table user (id int, active boolean, ...);
- 两种情况建议创建部分索引
  - 活跃用户占比少
  - 应用系统只对活跃用户进行检索
- create index idx_user_id on user(id) where active is true;
- 这些查询可以走部分索引
- select * from user where active is true; -- 全索引扫描
- select * from user where active is true and id=?; -- 精确扫描

特性

- 全文检索数据类型
- 例子, 中文分词与检索

- 分词类型：tsvector，支持分词，位置，段落
- 查询条件类型：tsquery，支持与，或，位置，段落，前缀等组合
- 分词索引：GIN

- to_tsvector('testzhcfg','“今年保障房新开工数量虽然有所下调，但实际的年度在建规模以及竣工规模会超以往年份，相对应的对资金的需求也会创历史纪录。”陈国强说。在他看来，与2011年相比，2012年的保障房建设在资金配套上的压力将更为严峻。');
- '2011':27 '2012':29 '上':35 '下调':7 '严峻':37 '会':14 '会创':20 '保障':1,30 '历史':21 '压力':36 '国强':24 '在建':10 '实际':8 '对应':17 '年份':16 '年度':9 '开工':4 '房':2 '房建':31 '数量':5 '新':3 '有所':6 '相比':28 '看来':26 '竣工':12 '纪录':22 '规模':11,13 '设在':32 '说':25 '资金':18,33 '超':15 '配套':34 '陈':23 '需求':19

应用场景

- to_tsquery('testzhcfg', '保障房资金压力');
  - to_tsquery
  - ---
  - '保障' & '房' & '资金' & '压力'
- SELECT 'super:*'::tsquery; -- super开头的单词
  - tsquery
  - ---
  - 'super':*

- 查询举例：
  - tsvector @@ to_tsquery('testzhcfg', '保障房资金压力'); -- 包含查询条件

性能

- 分词性能指标
- 英语分词性能：~ 900万 words每秒 (Intel(R) Xeon(R) CPU X7460 @ 2.66GHz)
- 中文分词性能：~ 400万 字每秒 (Intel(R) Xeon(R) CPU X7460 @ 2.66GHz)

- 英文分词+插入性能：~ 666万 字每秒 (Intel(R) Xeon(R) CPU X7460 @ 2.66GHz)
- 中文分词+插入性能：~ 290万 字每秒 (Intel(R) Xeon(R) CPU X7460 @ 2.66GHz)

- 查询性能和查询条件，数据量都有关系，没有很好的评估标准，大多数查询可以在毫秒级返回。
- https://github.com/amutu/zhparser

特性

- UDF
- c, python, java, perl, R, ……

应用场景

- 例子
- 把结巴中文分词的功能移植到PostgreSQL
- postgres=# create language plpythonu;
- CREATE LANGUAGE
- postgres=# create or replace function fenci(i_text text) returns ***tsvector*** as $$
- import jieba
- seg_list = jieba.cut(i_text, cut_all=False)
- return(" ".join(seg_list))
- $$ language plpythonu;
- CREATE FUNCTION

应用场景

- postgres=# select fenci('小明硕士毕业于中国科学院计算所，后在日本京都大学深造');
- ----------
- '中国科学院' '于' '后' '在' '小明' '日本京都大学' '毕业' '深造' '硕士' '计算所' ，'
- (1 row)

- postgres=# select fenci('结婚的和尚未结婚的');
- ----------
- '和' '尚未' '的' '结婚'
- (1 row)

特性

- 近似度查询

应用场景

- pg_trgm
- 近似度匹配，支持**GIN索引**检索

- 字符串前后各加2个空格，每连续的3个字符一组进行拆分并去重复，不区分大小写
  - digoal=> select show_trgm('digoal');
  - show_trgm
  - ---
  - {" d"," di","al ",dig,goa,igo,oal}

  - digoal=> select show_trgm('DIGOAL123456');
  - show_trgm
  - ---
  - {" d"," di",123,234,345,456,"56 ",al1,dig,goa,igo,l12,oal}
  - (1 row)
- 近似度算法
  - 两个字符串相同trigram个数 除以 总共被拆成多少个trigram

应用场景

- 大于等于近似度限制时，返回TRUE，同样可根据近似度高低排名，反映检索条件和数据之间的相关度。
  - digoal=> select show_limit();
  - show_limit
  - ------------
  - 0.3
  - (1 row)
  - postgres=# select similarity('postregsql','postgresql');
  - similarity
  - ------------
  - 0.375
  - (1 row)
  - postgres=# select 'postregsql' % 'postgresql'; -- 在记忆出现问题时，例如输错几个依旧可以匹配
  - ?column?
  - ----------
  - t
  - (1 row)

特性

- 域，约束
- 例子，限制输入格式，确保输入为一个正确的EMAIL地址。

- 域
- postgres=# create domain email as text check (value ~ '^.+@.+\..+$');
- CREATE DOMAIN
- postgres=# select 'a'::email;
- ERROR: value for domain email violates check constraint "email_check"
- STATEMENT: select 'a'::email;
- ERROR: value for domain email violates check constraint "email_check"
- postgres=# select 'digoal@126.com'::email;
- email
- ----------
- digoal@126.com
- (1 row)

应用场景

- 域，约束
- 例子，限制输入格式，确保输入为一个正确的EMAIL地址。

- 约束（支持数组，需自定义操作符配合数组约束使用）
- 创建模糊匹配对等函数
  - postgres=# create or replace function u_textregexeq(text,text) returns boolean as $$
  - select textregexeq(**$2,$1**);
  - $$ language sql strict;

  - 创建一个模糊匹配的对等操作符
  - postgres=# CREATE OPERATOR ~~~~ (procedure = u_textregexeq, leftarg=text,rightarg=text);
  - CREATE OPERATOR

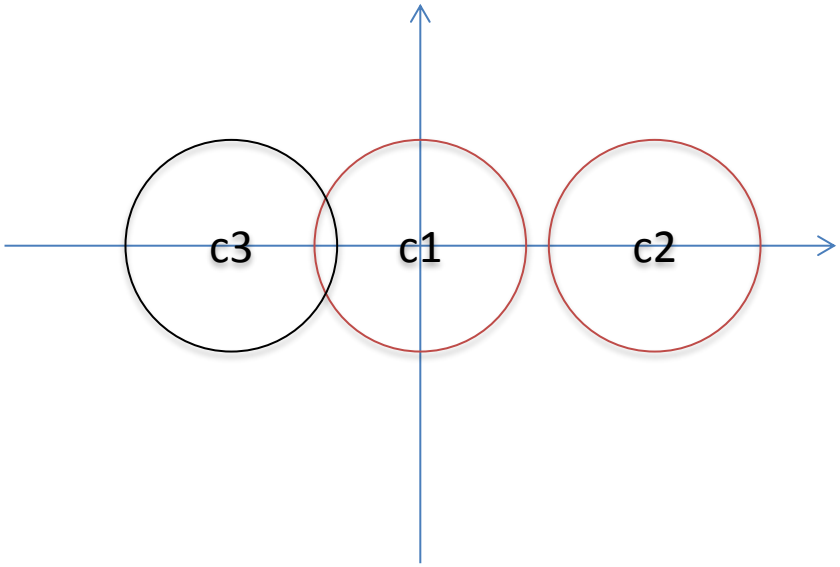
  - postgres=# select 'digoal@126.com' ~~~~ '^.+@.+\..+$';
  - -[RECORD 1]
  - ?column? | f
  - postgres=# select '^.+@.+\..+$' ~~~~ 'digoal@126.com';
  - -[RECORD 1]
  - ?column? | **t**

应用场景

- 约束（支持数组，需自定义操作符）
  - postgres=# create table t_email(id int, email text[] check ('^.+@.+\..+$' ~~~~ all (email)));
  - CREATE TABLE
  - postgres=# insert into t_email values (1, array['digoal@126.com','a@e.com']::text[]);
  - INSERT 0 1
  - postgres=# insert into t_email values (1, array['digoal@126.com','a@e']::text[]);
  - ERROR: new row for relation "t_email" violates check constraint "t_email_email_check"
  - DETAIL: Failing row contains (1, {digoal@126.com,a@e}).

特性

- 排他约束



应用场景

- gist, spgist索引；平面/立体几何类型
- 例子, exclusion约束
- (适用于左右对等操作符，如等于，相交)

- CREATE TABLE test(id int,geo circle,EXCLUDE USING GIST (geo WITH pg_catalog.&&));
- INSERT INTO test values(1,'<(0,0),2>'::circle);
- INSERT INTO test values(1,'<(4.1,0),2>'::circle);

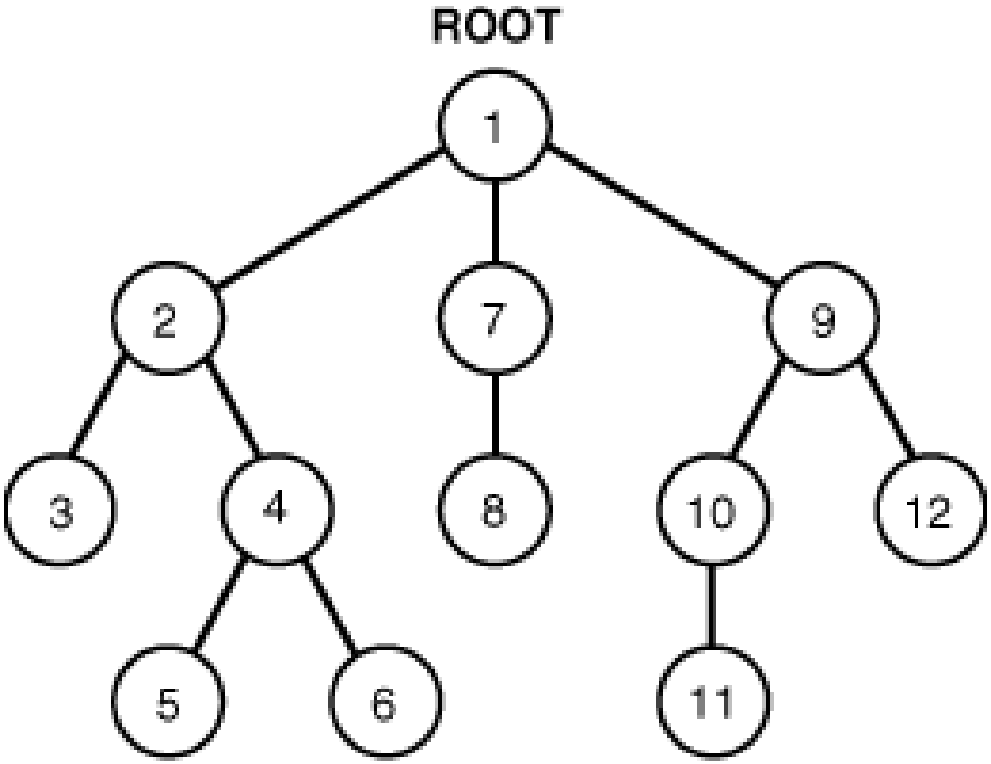
- INSERT INTO test values(1,'<(-1.9,0),2>'::circle);
- ERROR: conflicting key value violates exclusion constraint "test_geo_excl"
- DETAIL: Key (geo)=(<(-1.9,0),2>) conflicts with existing key (geo)=(<(0,0),2>).

特性

- 递归查询

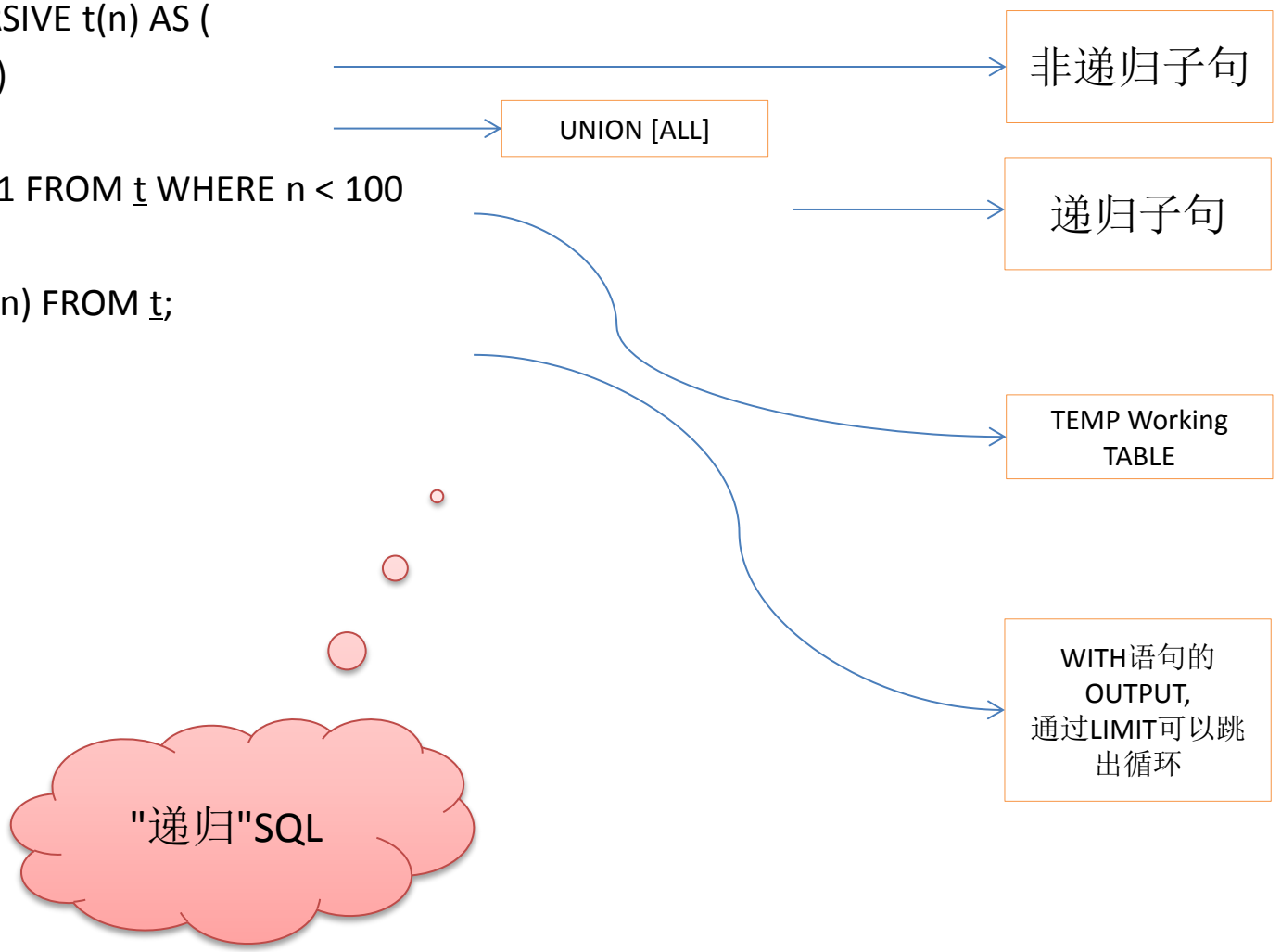
应用场景

- 异构查询，例如公交线路信息，可能包含当前站点，上一个站点的信息
- 某些多媒体分类信息，包括大类，小类，每条记录可能记录了父类



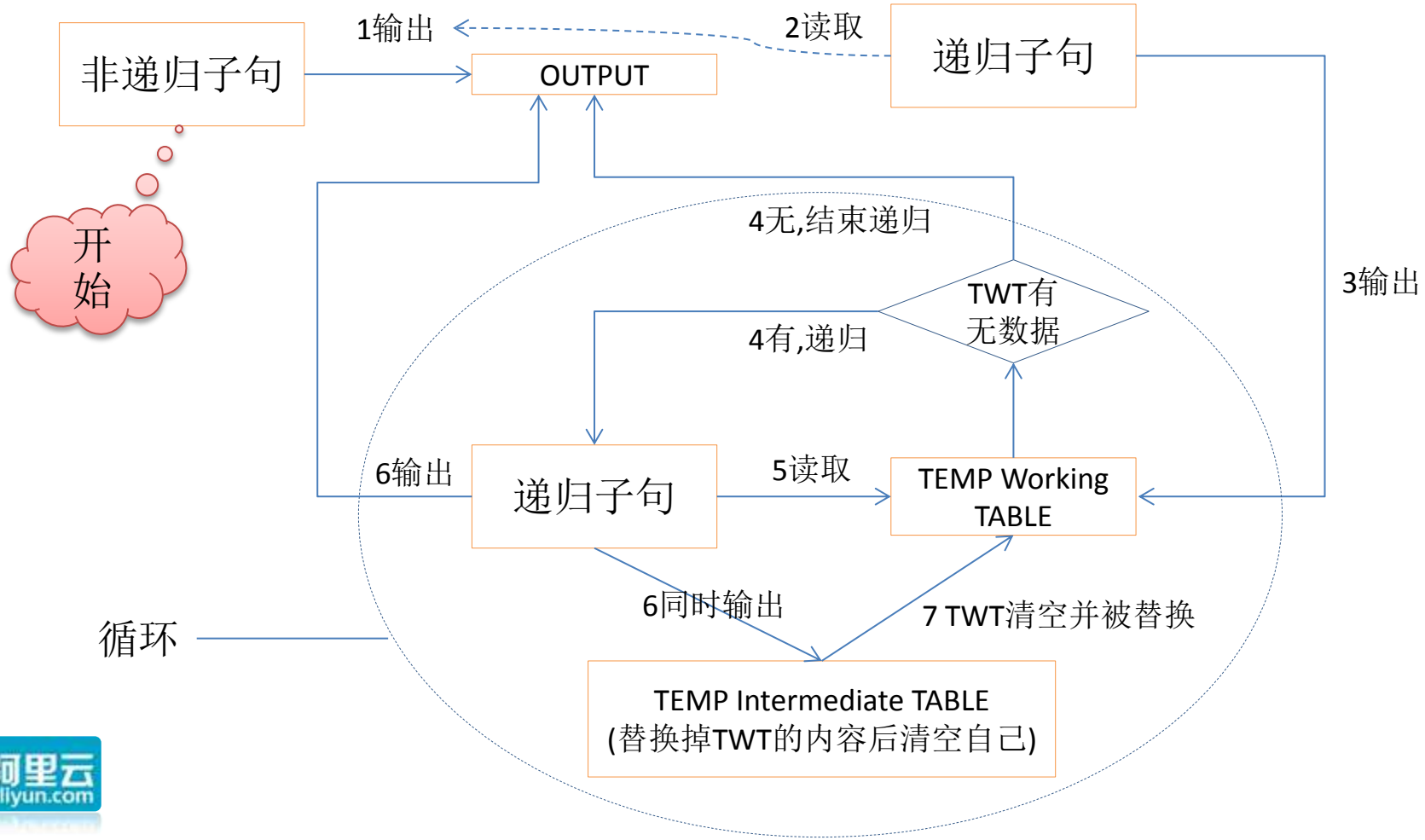
应用场景

```
WITH RECURSIVE t(n) AS (
    VALUES (1)
  UNION ALL
    SELECT n+1 FROM t WHERE n < 100
)
SELECT sum(n) FROM t;
```



应用场景

- UNION 去重复(去重复时NULL 视为等同)
- 图中所有输出都涉及UNION [ALL]的操作, 包含以往返回的记录和当前返回的记录



应用场景

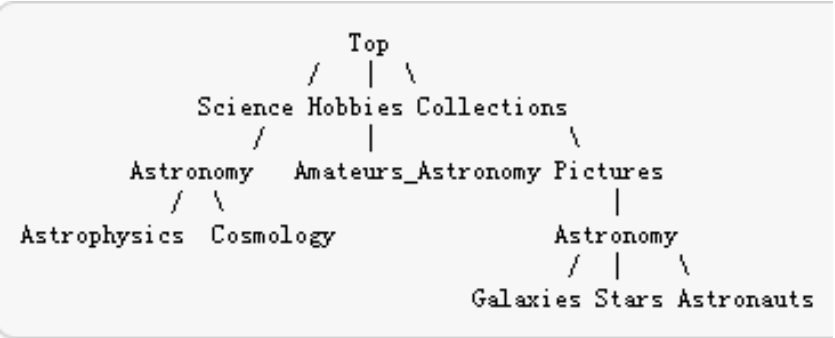
- 异构查询，例如公交线路信息，可能包含当前站点，上一个站点的信息
- 某些多媒体分类信息，包括大类，小类，每条记录可能记录了父类

```
with recursive t_result as
(
 select id,name,parentid,type,status,grade,filename,md5,
        brief,orderid,updatetime,createtime
   from tbl_menu as t_initial
   where name ~ '刘德华'
 union
 select t_working.id,t_working.name,t_working.parentid,
        t_working.type,t_working.status,t_working.grade,t_working.filename,
        t_working.md5,t_working.brief,t_working.orderid,t_working.updatetime,t_working.createtime
   from tbl_menu as t_working
 join
 t_result
 on (t_result.parentid = t_working.id)  -- 向上搜索
 -- 向下搜索 on (t_result.id = t_working.parentid)
)
select id,name,parentid,type,status,grade,filename,md5,brief,orderid,updatetime,createtime from t_result;
```

特性

- 异构数据类型
- ltree http://www.postgresql.org/docs/devel/static/ltree.html

```
CREATE TABLE test (path ltree);
INSERT INTO test VALUES ('Top');
INSERT INTO test VALUES ('Top.Science');
INSERT INTO test VALUES ('Top.Science.Astronomy');
INSERT INTO test VALUES ('Top.Science.Astronomy.Astrophysics');
INSERT INTO test VALUES ('Top.Science.Astronomy.Cosmology');
INSERT INTO test VALUES ('Top.Hobbies');
INSERT INTO test VALUES ('Top.Hobbies.Amateurs_Astronomy');
INSERT INTO test VALUES ('Top.Collections');
INSERT INTO test VALUES ('Top.Collections.Pictures');
INSERT INTO test VALUES ('Top.Collections.Pictures.Astronomy');
INSERT INTO test VALUES ('Top.Collections.Pictures.Astronomy.Stars');
INSERT INTO test VALUES ('Top.Collections.Pictures.Astronomy.Galaxies');
INSERT INTO test VALUES ('Top.Collections.Pictures.Astronomy.Astronauts');
CREATE INDEX path_gist_idx ON test USING gist(path);
CREATE INDEX path_idx ON test USING btree(path);
```



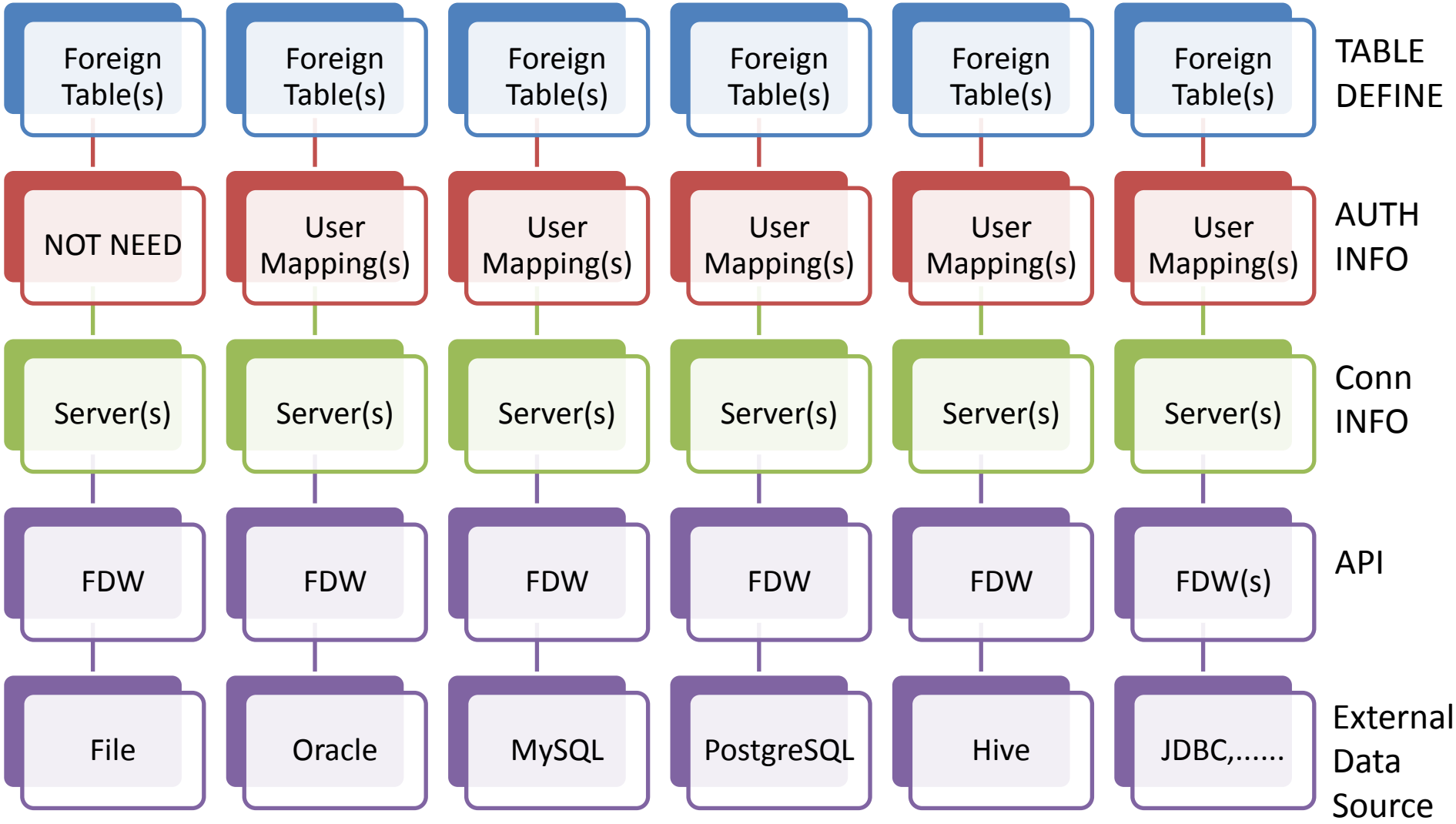

```
ltreetest=> SELECT path FROM test WHERE path <@ 'Top.Science';
                path
-----------------------------------
 Top.Science
 Top.Science.Astronomy
 Top.Science.Astronomy.Astrophysics
 Top.Science.Astronomy.Cosmology
(4 rows)
```

特性

- 外部表
- https://wiki.postgresql.org/wiki/Fdw

- 可以像操作
- 本地表一样
- join,read/write

特性



特性

- 事件触发器
- 例子,控制普通用户没有执行DDL的权限
- 目前支持的事件
  - ddl_command_start
  - ddl_command_end
  - table_rewrite
  - sql_drop
- 支持的SQL, (未完全截取)

Command Tag	ddl_command_start	ddl_command_end	sql_drop	table_rewrite
ALTER AGGREGATE	X	X	-	-
ALTER COLLATION	X	X	-	-
ALTER CONVERSION	X	X	-	-
ALTER DOMAIN	X	X	-	-
ALTER EXTENSION	X	X	-	-
ALTER FOREIGN DATA WRAPPER	X	X	-	-
ALTER FOREIGN TABLE	X	X	X	-
ALTER FUNCTION	X	X	-	-

特性

```
postgres=# create or replace function ev_reject_ddl() returns event_trigger language plpgsql as $$
declare
  v_rolesuper boolean;
begin
  --  判断是否为超级用户
  select rolsuper into v_rolesuper from pg_roles where rolname=current_user;
  if v_rolesuper then
    return;
  else
    raise exception 'event:%, command:%. Do not enable nonsuperuser execute ddl.', TG_EVENT, TG_TAG;
  end if;
end;
$$;
CREATE FUNCTION
postgres=# create event trigger tg_abort1 on ddl_command_start execute procedure ev_reject_ddl();
CREATE EVENT TRIGGER
postgres=# create table tt(id int);
CREATE TABLE
postgres=# \c postgres test
You are now connected to database "postgres" as user "test".
postgres=> create table tt(id int);
ERROR:  event:ddl_command_start, command:CREATE TABLE. Do not enable nonsuperuser execute ddl.
postgres=> create table tt1(id int);
ERROR:  event:ddl_command_start, command:CREATE TABLE. Do not enable nonsuperuser execute ddl.
postgres=> drop table tt;
ERROR:  event:ddl_command_start, command:DROP TABLE. Do not enable nonsuperuser execute ddl.
以上触发器函数稍微修改一下，可以限制特定用户，或者超级用户执行特定的DDL。
```

特性

- 空间数据管理
  - PostGIS, pgrouting
- GiST, SP-GiST 索引

应用场景

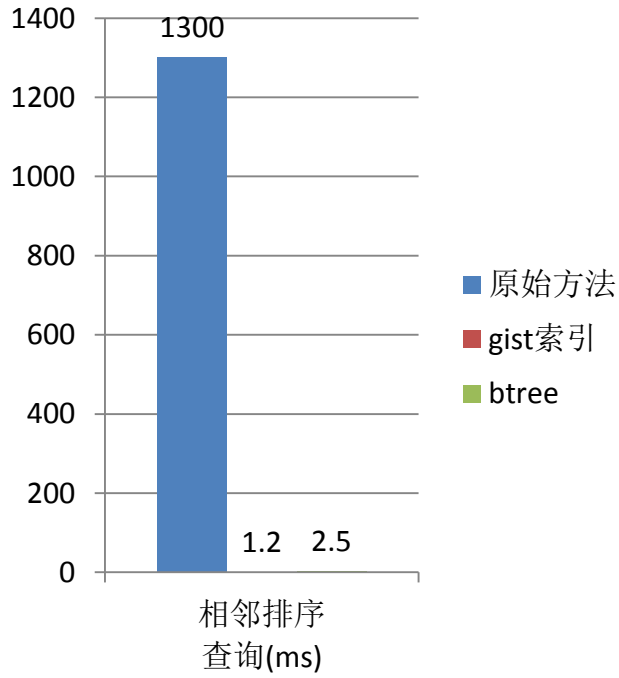
- O2O应用
- 位置交友
- 轨迹管理
- 电子栅栏，例如公车行驶在栅栏外时告警。
- KNN检索，距离排序，距离检索。
- (wifi热点定位) 商场的用户时空轨迹，商铺评估，促销效果评估。
- 最短驾驶路径计算
- ……
- NASA, ESA, ……

应用场景

- knn例子， gist索引
- select abs(id-100),* from test where abs(id-100)<10 order by abs(id-100) limit 10;

- select abs(id-100),* from
- (select * from (select * from test where id<=100 order by id desc limit 10) t
- union all
- select * from (select * from test where id>100 order by id limit 10) t
-) t
- where abs(id-100)<10 order by abs(id-100) limit 10;

- select id, * from test **order by id <-> 100** limit 10;



特性

- 数据预热
- 扩展
  - postgres=# create extension pg_buffercache;
  - postgres=# create extension pg_prewarm;
- 保存buffer快照(forknumber=0代表main，即数据)
  - postgres=# create table buf (id regclass,blk int8,crt_time timestamp);
  - postgres=# truncate buf;
  - postgres=# insert into buf select a.oid::regclass,b.relblocknumber,now() from pg_class a,pg_buffercache b where pg_relation_filenode(a.oid)=b.relfilenode and b.relforknumber=0 order by 1,2;
  - INSERT 0 32685

特性

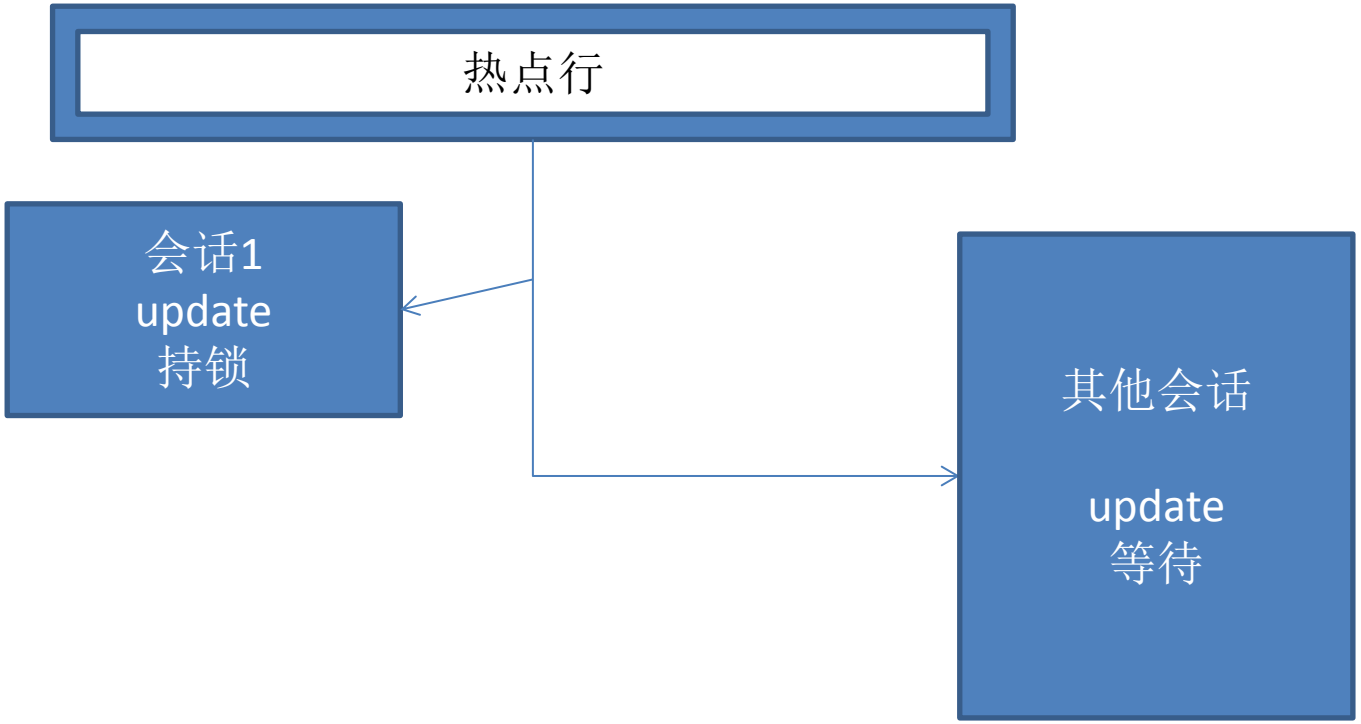
- 数据预热
- 重启数据库后的预热方法
  - pg95@db-172-16-3-150-> pg_ctl restart -m fast
  - pg95@db-172-16-3-150-> psql
  - postgres=# select pg_prewarm(id,'buffer','main',blk,blk) from buf;
  - 验证
  - postgres=# select a.oid::regclass,b.relblocknumber,relforknumber from pg_class a,pg_buffercache b where pg_relation_filenode(a.oid)=b.relfilenode and b.relforknumber=0 order by 1,2;
  - oid | relblocknumber | relforknumber
  - ----------+----------+----------
  - pg_default_acl_role_nsp_obj_index | 0 | 0
  - pg_tablespace | 0 | 0
  - pg_shdepend_reference_index | 0 | 0
  - .....

特性

- advisory Lock
  - 面向SQL的轻量级锁

应用场景

- 秒杀



应用场景

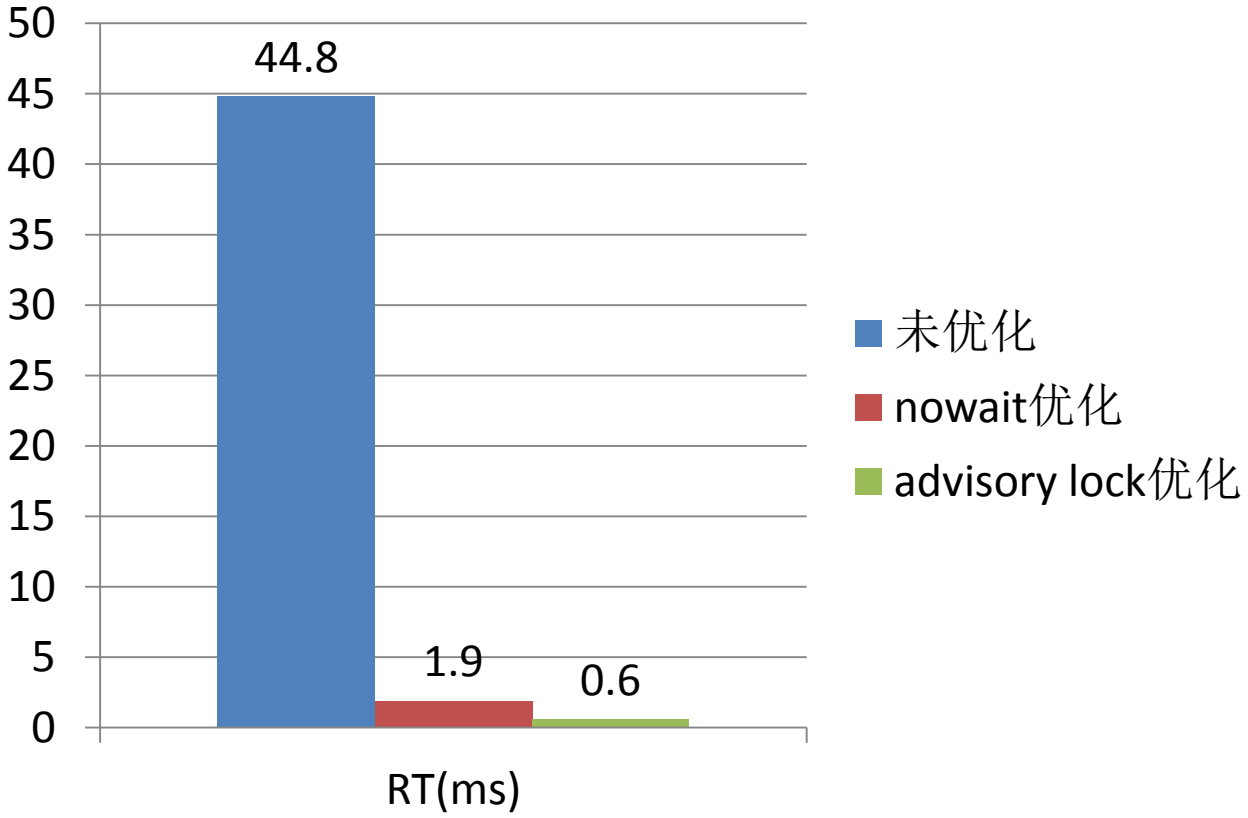
- 例子
- 原始方法
- update tbl set xxx=xxx,upd_cnt=upd_cnt+1 where id=pk and upd_cnt+1<=5;

- 用nowait 减少无效等待, 提高RT
- CREATE OR REPLACE FUNCTION public.f1(i_id integer)
- RETURNS void
- LANGUAGE plpgsql
- AS $function$
- declare
- begin
- perform 1 from t1 where id=i_id for update nowait;
- update t1 set info=now()::text where id=i_id;
- exception when others then
- return;
- end;
- $function$;

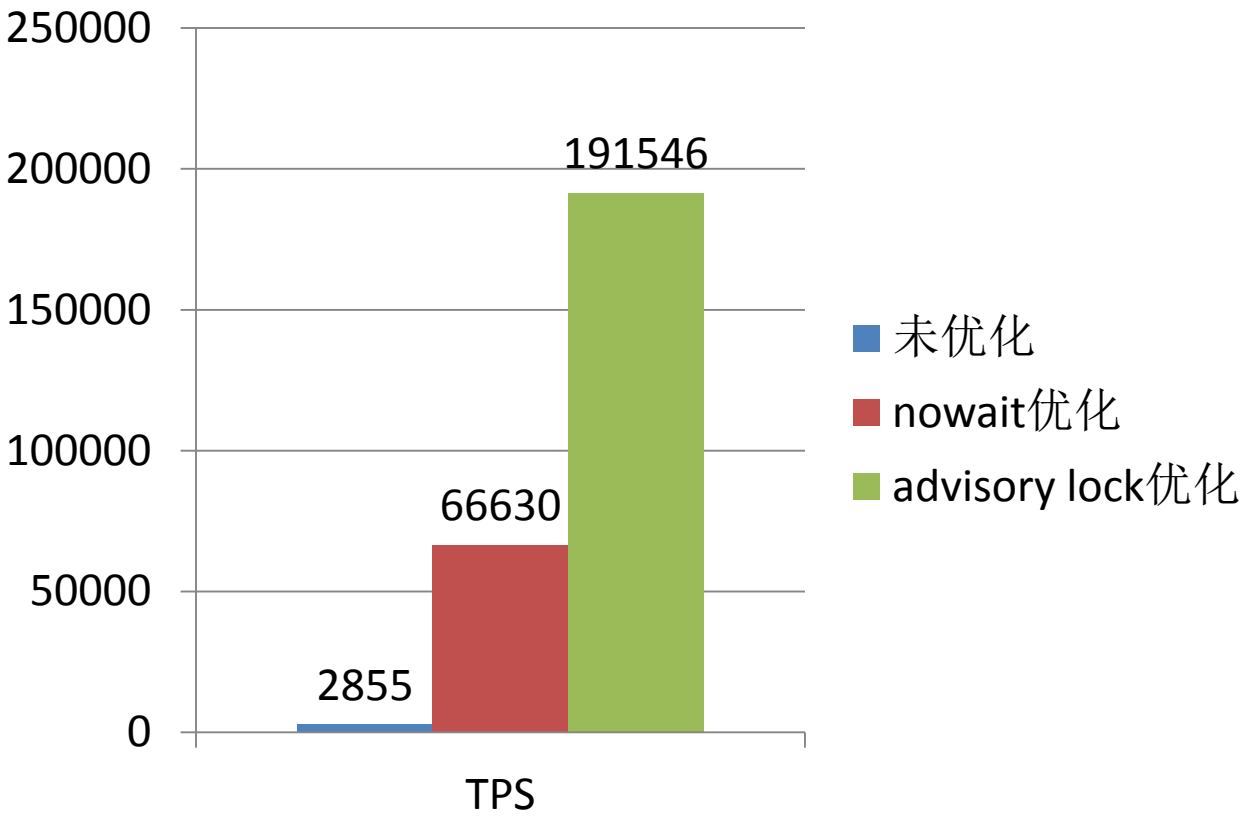
应用场景

- 用advisory lock优化：
- CREATE OR REPLACE FUNCTION public.f(i_id integer)
- RETURNS void
- LANGUAGE plpgsql
- AS $function$
- declare
- a_lock boolean := false;
- begin
- select pg_try_advisory_xact_lock(i_id) into a_lock;
- if a_lock then
- update t1 set info=now()::text where id=i_id;
- end if;
- exception when others then
- return;
- end;
- $function$;

性能



性能



特性

- Stream replication
- 同步or异步 Physical
- 同步or异步 Logical

应用场景

- HA
- 容灾
- 读写分离
- 多主，一主多从复制，多主一备数据合并。
- 延迟的备库
- 快照级备份
- PITR，数据可以恢复到任意时间点
- Aurora(共享存储一主多读)

性能

- 延迟
  - Physical Replication
  - 取决于网络质量，延迟通常 < 0.0x毫秒。
- 性能损耗
  - 上游节点TPS损耗 ~ 5%

特性

- MPP
  - HAWQ
  - Greenplum

特性

- Greenplum
- 架构

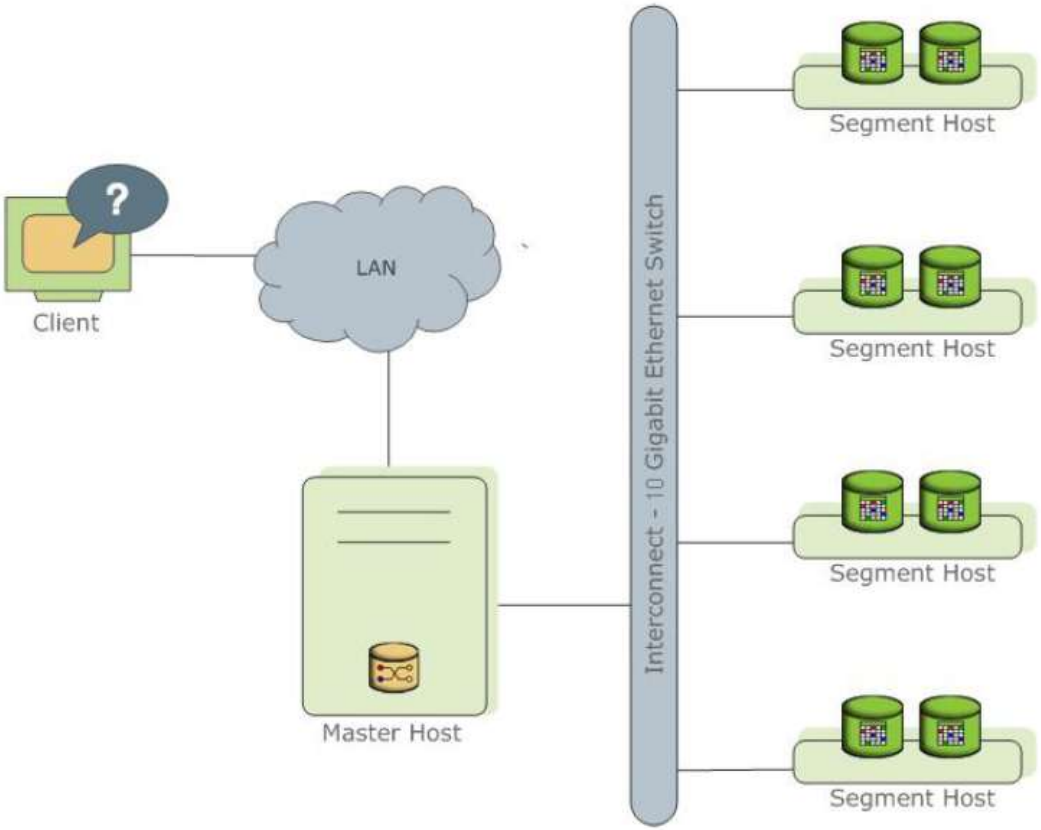


Figure 1: High-Level Greenplum Database Architecture

特性

- Greenplum
- 并行备份和还原

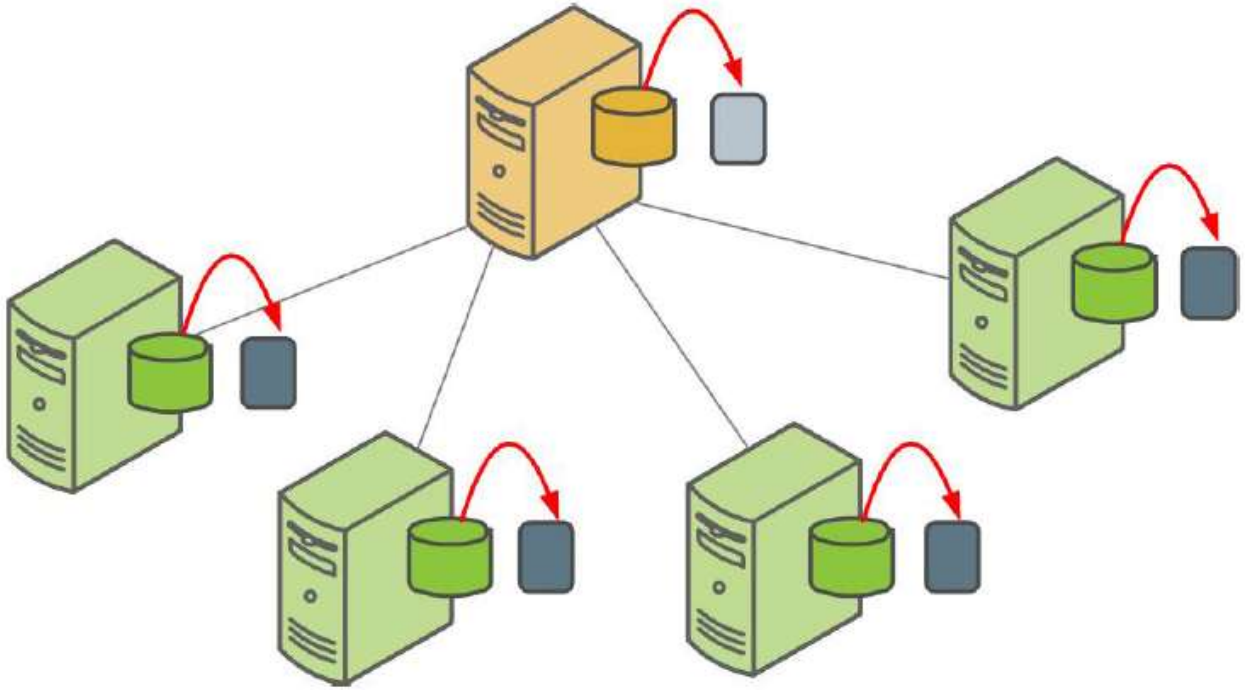
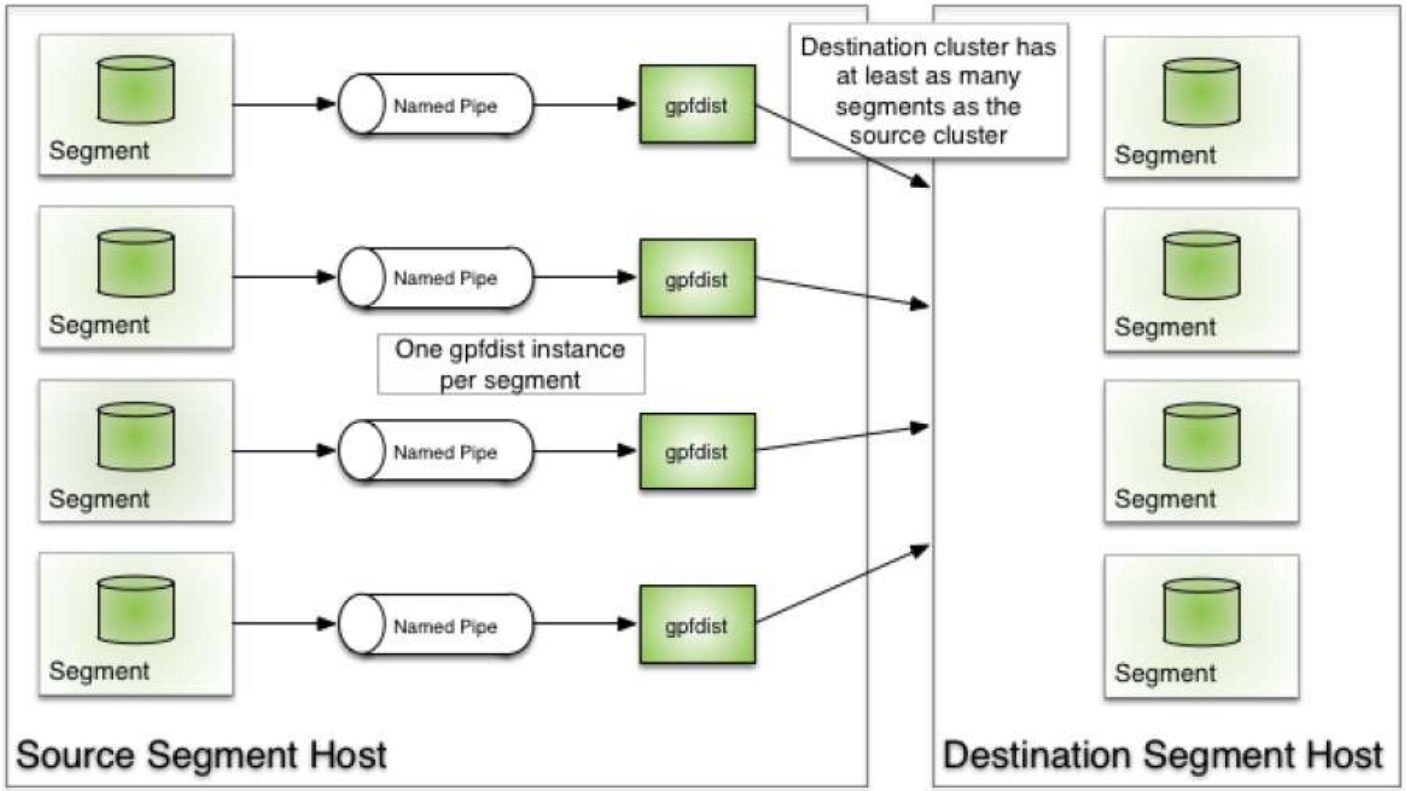


Figure 10: Parallel Backups in Greenplum Database

特性

- Greenplum
- 并行数据导入



特性

- Greenplum
- HDFS外部表

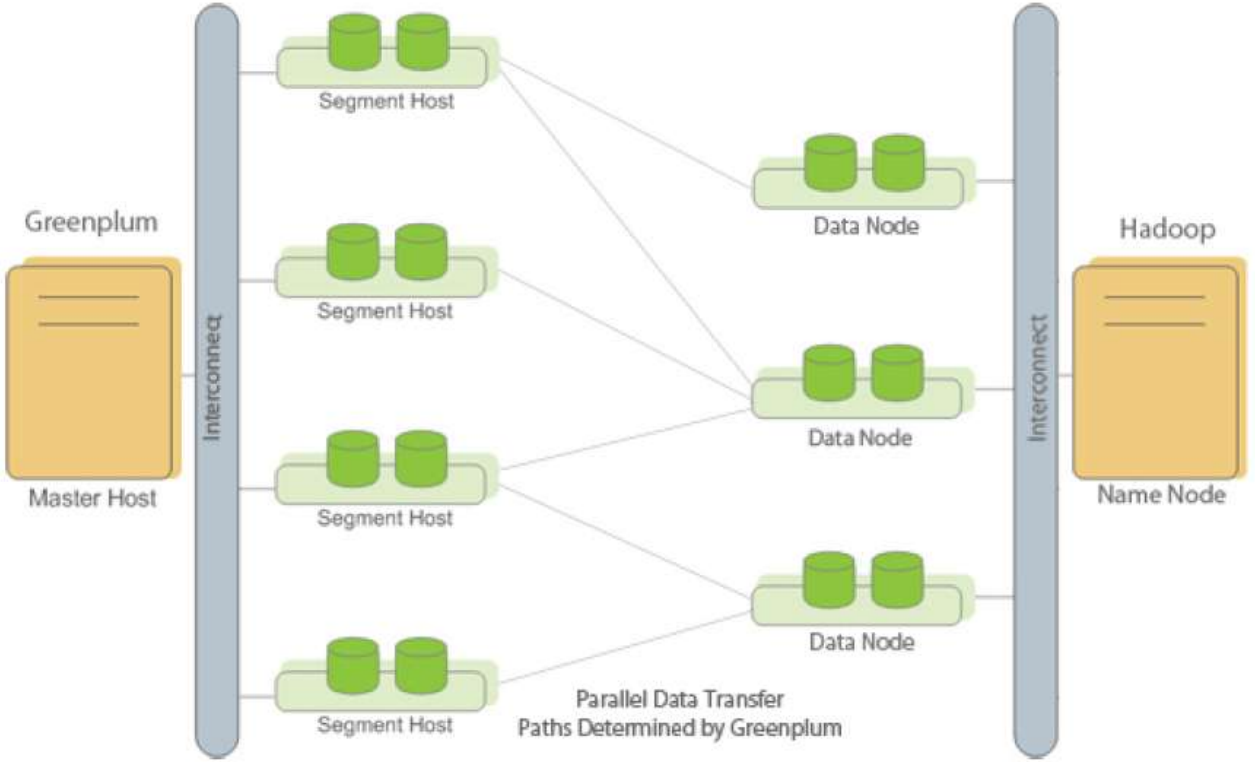


Figure 14: External Table Located on a Hadoop Distributed File System

特性

- Greenplum
- 分布式执行计划

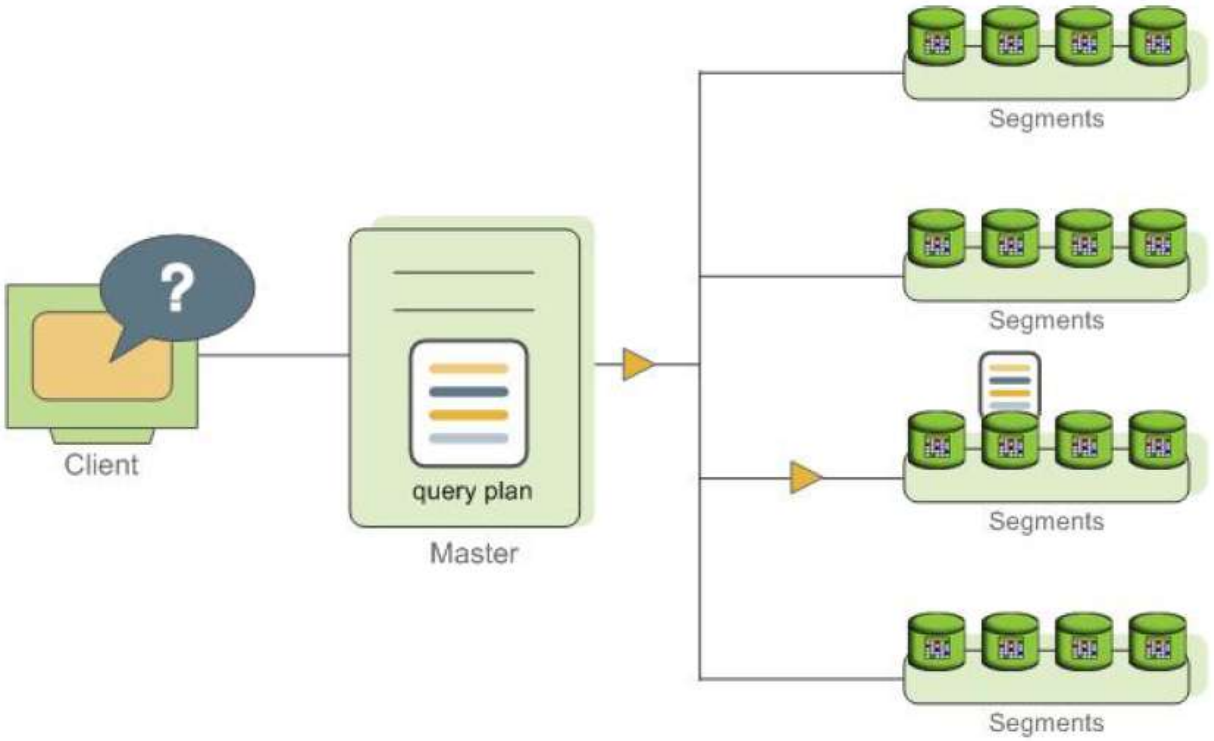


Figure 19: Dispatching a Targeted Query Plan

特性

- Greenplum
- 数据直接在数据节点之间重分布
- 并行处理

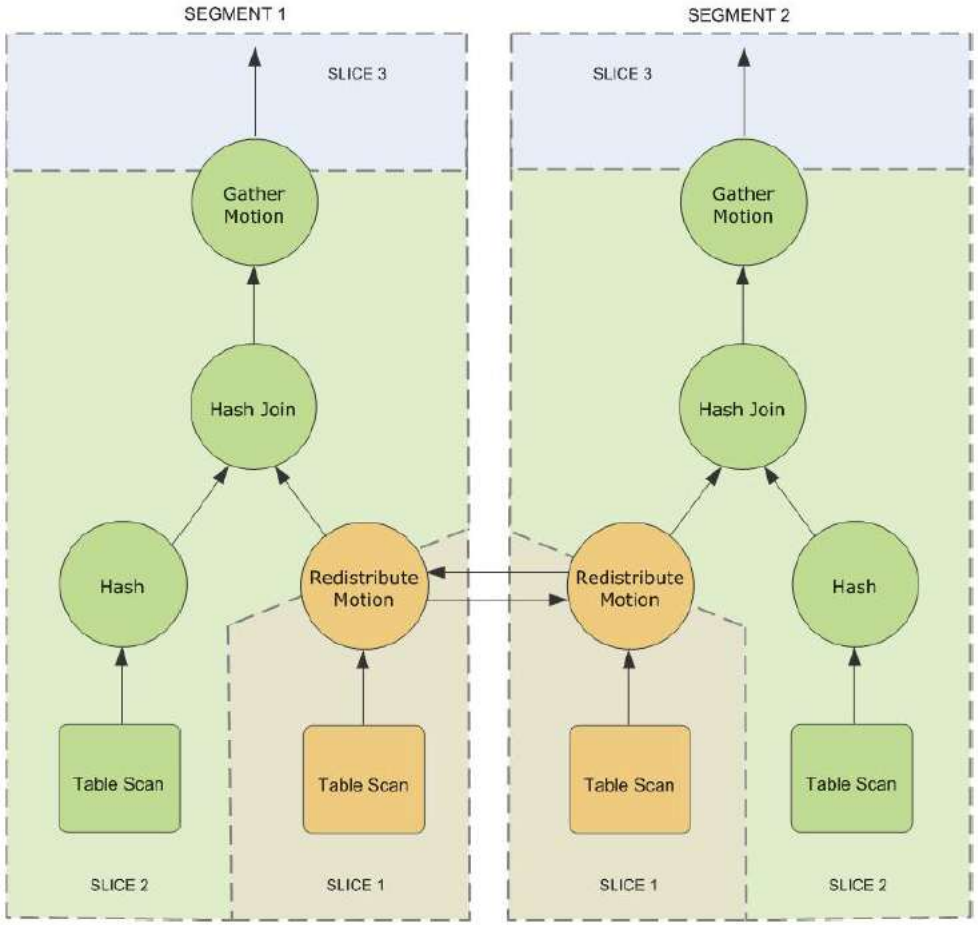


Figure 20: Query Slice Plan

特性

- Greenplum
- 并行处理

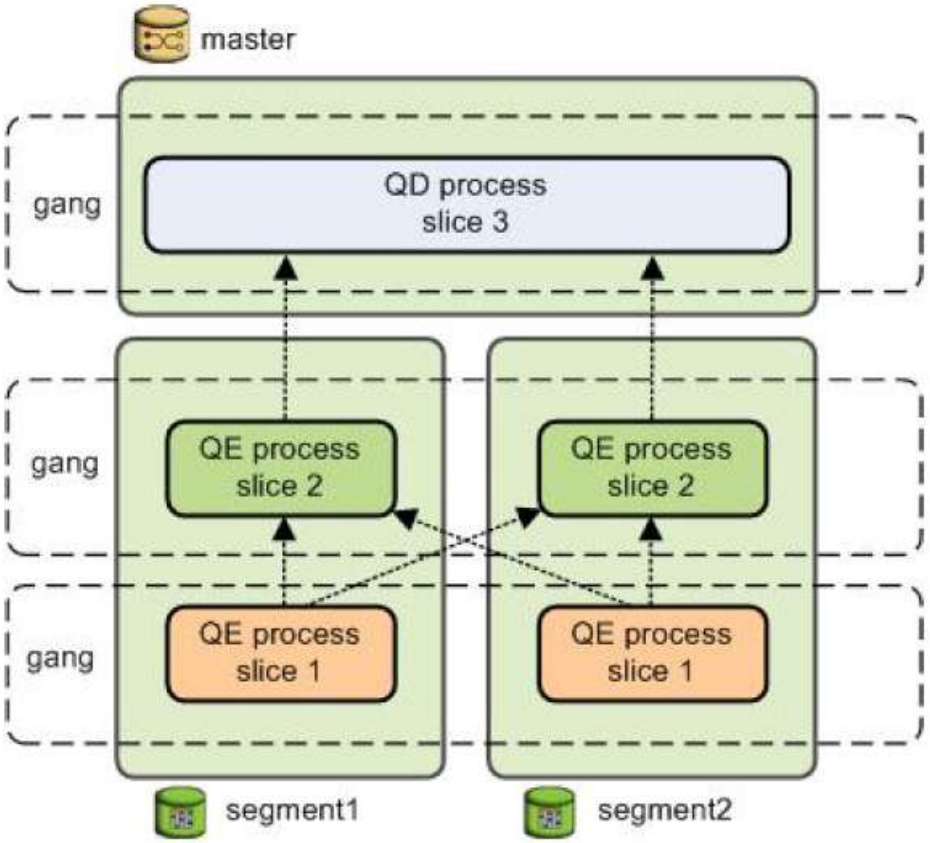
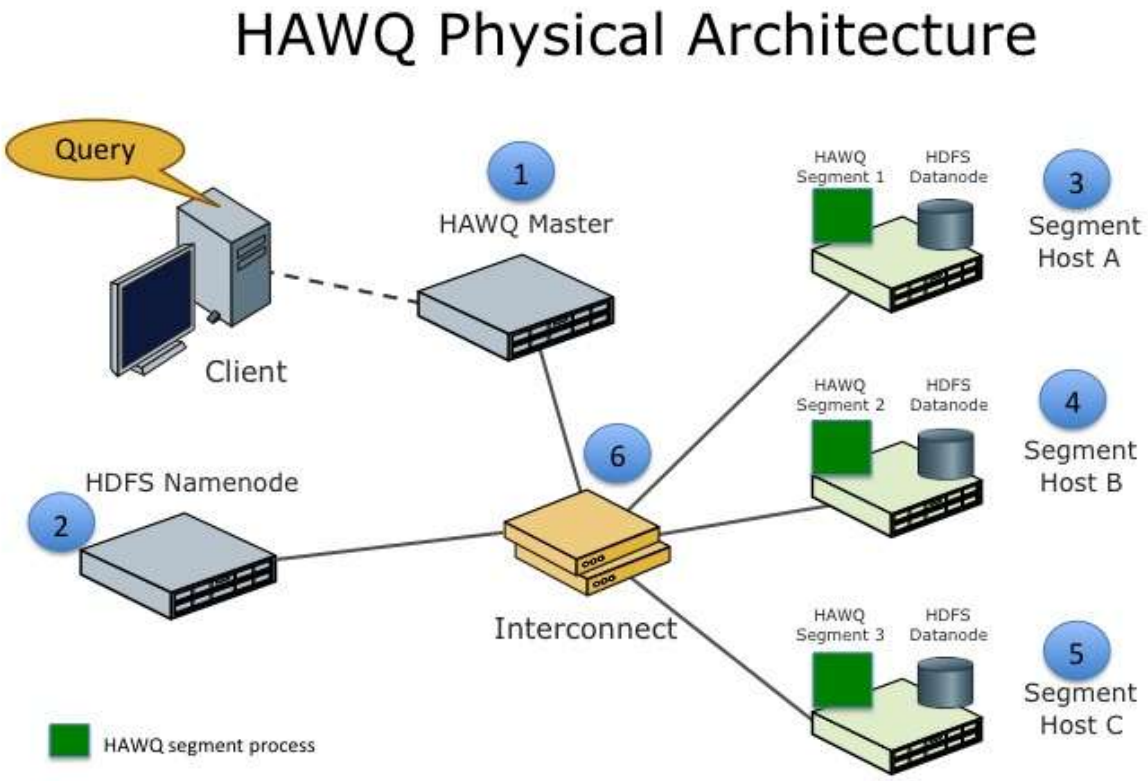


Figure 21: Query Worker Processes

特性

- HAWQ
- “HDFS”版
- 的GP



性能

- 摘录

DCA 系列规格概述

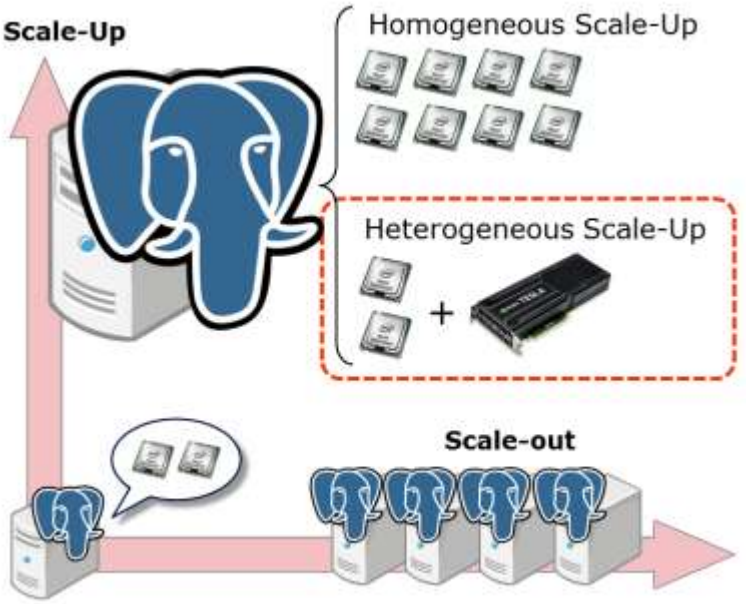
	DCAGP1000	大容量 DCA GP1000C
主服务器	2	2
分支服务器	16	16
CPU 内核总数	192	192
内存总量	768 GB	768 GB
分支硬盘的固态硬盘数量	192	192
可用容量（未压缩）	36 TB	124 TB
可用容量（压缩后）	144 TB	496 TB
最大扩展	6 个机架	6 个机架
扫描速度	24 GB/ 秒	14 GB/ 秒
数据加载速度	10 TB/ 小时	10 TB/ 小时

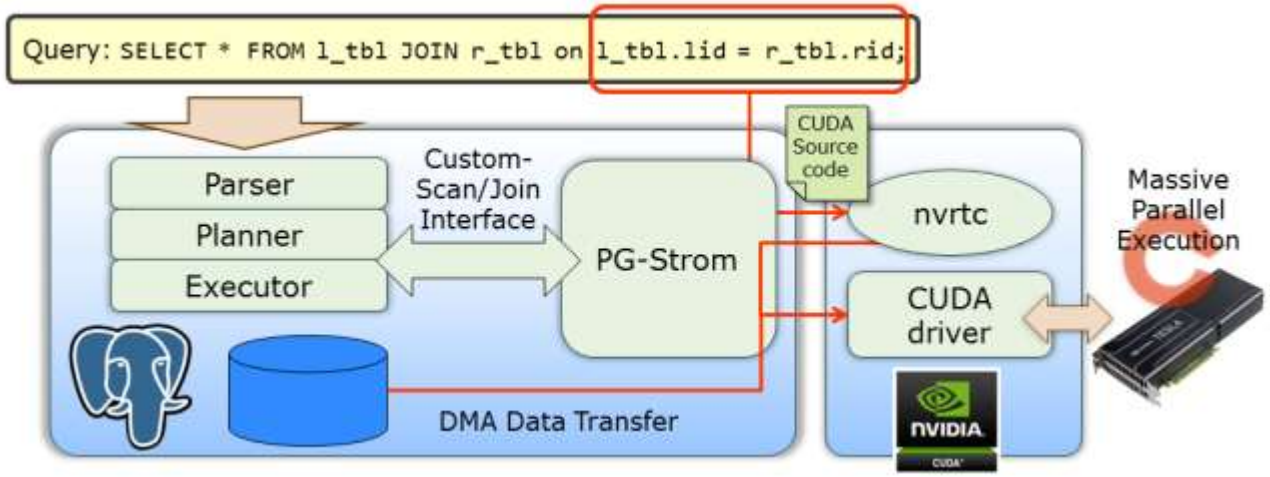
特性

- 安全
  - 行级安全策略
  - 防暴力破解
  - 密码复杂度策略
  - ACL列表
  - 数据加密
  - 数据传输加密
  - 函数体加密
  - 支持“trust”, “reject”, “md5”, “password”, “gss”, “sspi”, “ident”, “peer”, “pam”, “ldap”, “radius” or “cert”等认证方法
  - 审计功能
- ……

特性

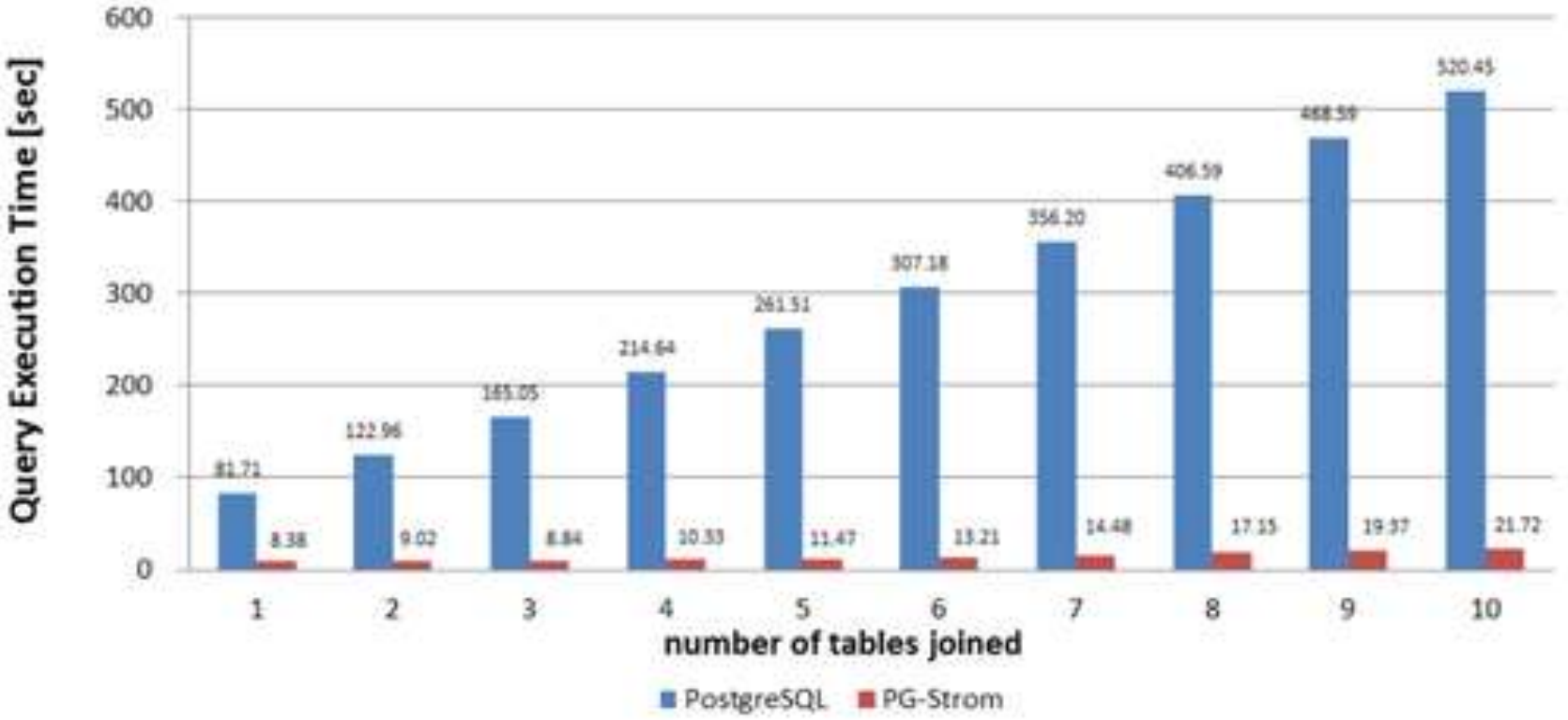
- GPU
- 并行计算





性能

- GPU



一般常见应用性能指标

- 测试用例，测试环境请参考，
  - http://blog.163.com/digoal@126/blog/static/163877040201541104656600/
  - http://blog.163.com/digoal@126/blog/static/16387704020154431045764/
  - http://blog.163.com/digoal@126/blog/static/163877040201542103933969/
  - http://blog.163.com/digoal@126/blog/static/16387704020154632523 87/
  - http://blog.163.com/digoal@126/blog/static/163877040201546516557 83/
  - http://blog.163.com/digoal@126/blog/static/163877040201546534228 92/
  - http://blog.163.com/digoal@126/blog/static/163877040201548114214 84/
  - http://blog.163.com/digoal@126/blog/static/16387704020154129958753/

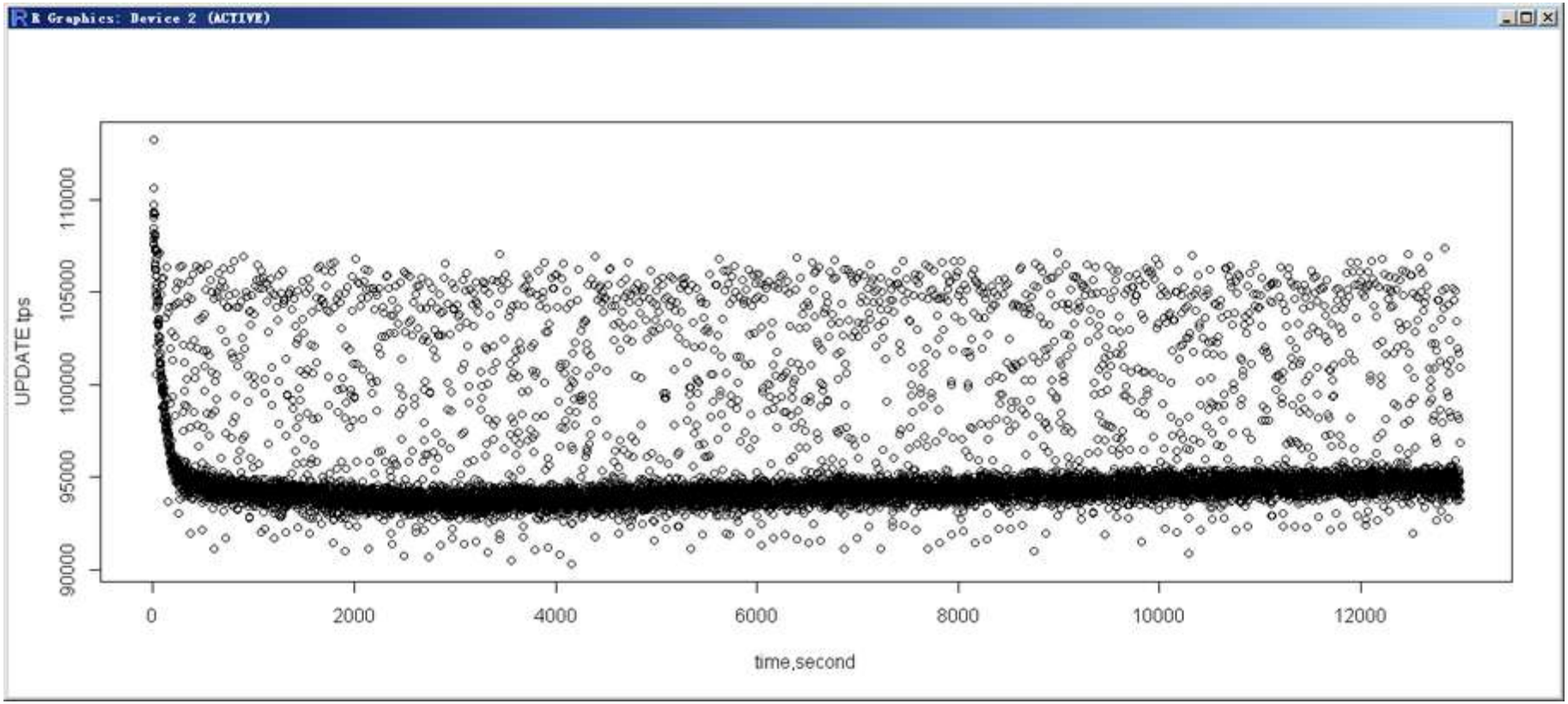
一般常见应用性能指标

- 服务器 2009 年 IBM X3950
- CPU 4 * 6核 Intel(R) Xeon(R) CPU X7460 @ 2.66GHz
- 内存 32 * 4GB DDR2 533MHz
- 硬盘 上海宝存 1.2TB Direct-IO PCI-E SSD
- 数据库 PostgreSQL 9.4.1
- 操作系统 CentOS 6.6 x64
- 文件系统 EXT4, noatime,nodiratime,nobarrier,discard

- 更新,查询数据量 5000万，基于主键的更新，查询。
- 插入数据量 100亿，插入带主键约束的表。

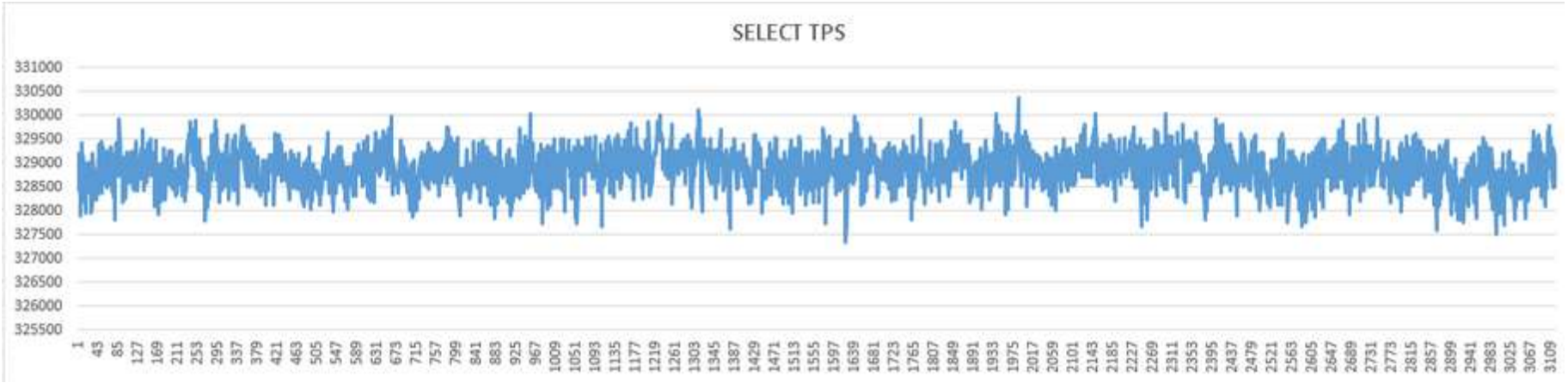
一般常见应用性能指标

- 更新 tps 分布情况：



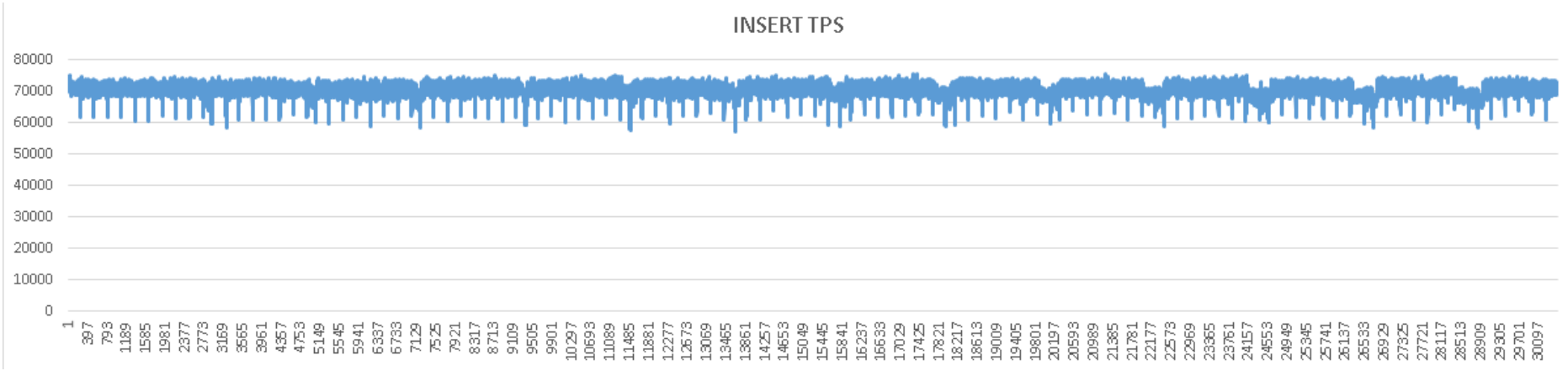
一般常见应用性能指标

- 查询 tps 分布情况：



一般常见应用性能指标

- 插入 tps 分布情况：
- 单表达到20亿后TPS表现平稳。

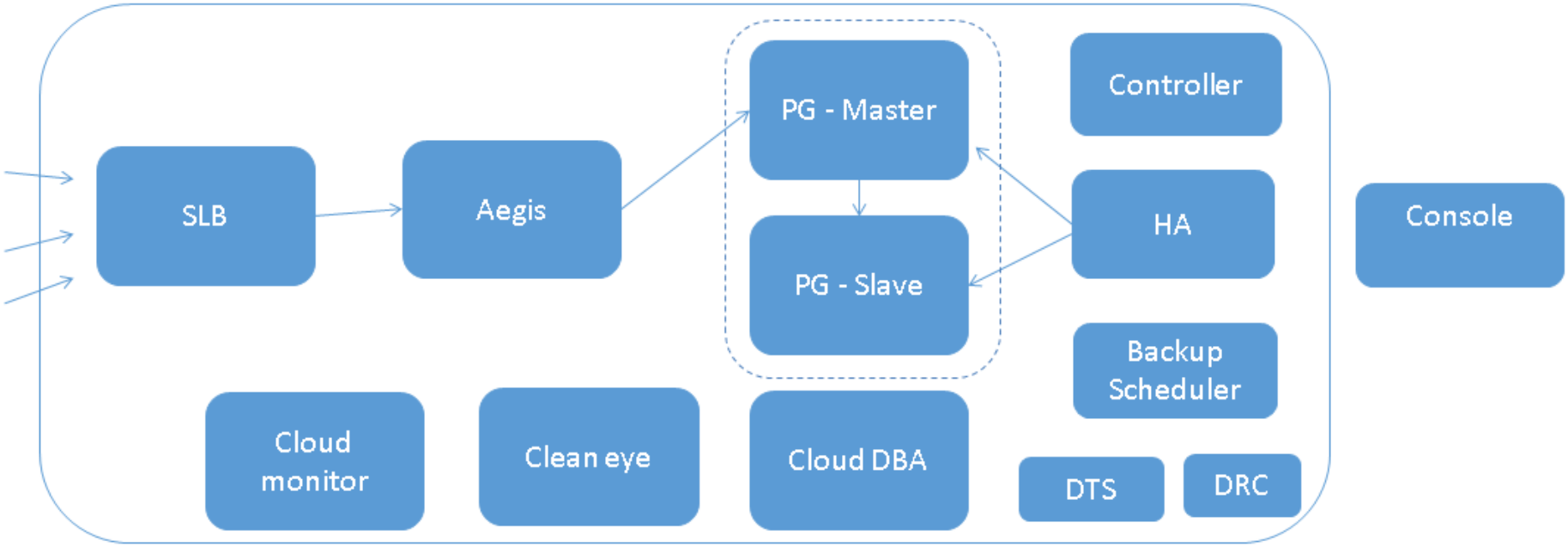


哪些用户在用PostgreSQL

- 生物制药 {Affymetrix(基因芯片), 美国化学协会, gene(结构生物学应用案例), ...}
- 电子商务 { CD BABY, etsy(与淘宝类似), whitepages, flightstats, Endpoint Corporation ...}
- 学校 {加州大学伯克利分校, 哈佛大学互联网与社会中心, .LRN, 莫斯科国立大学, 悉尼大学, ...}
- 金融 {Journyx, LLC, trusecommerce(类似支付宝), 日本证券交易交所, 邮储银行, 同花顺...}
- 游戏 {MobyGames, ...}
- 政府 {美国国家气象局, 印度国家物理实验室, 联合国儿童基金, 美国疾病控制和预防中心, 美国国务院, 俄罗斯杜马...}
- 医疗 {calorieking, 开源电子病历项目, shannon医学中心, ...}
- 制造业 {Exoteric Networks, 丰田, 捷豹路虎, 中芯国际}
- 媒体 {IMDB.com, 美国华盛顿邮报国会投票数据库, MacWorld, 绿色和平组织, ...}
- 开源项目 {Bricolage, Debian, FreshPorts, FLPR, PostGIS, SourceForge, OpenACS, Gforge, ...}
- 零售 {ADP, CTC, Safeway, Tsutaya, Rockport, ...}
- 科技 {Sony, MySpace, Yahoo, Afilias, APPLE, 富士通, Omniti, Red Hat, Sirius IT, SUN, 国际空间站, Instagram, Disqus, 去哪儿, 腾讯, 华为, 中兴, 斯凯, 云游, 阿里, 高德, 沃趣 ...}
- 通信 {Cisco, Juniper, NTT(日本电信), 德国电信, Optus, Skype, Tlestra(澳洲电讯), 中国移动...}
- 物流 {SF}
- 基础设施 {国家电网}
- More : http://www.postgresql.org/about/users/

阿里云RDS PG的架构

- 定制、增强的内核
- 一主一备架构（可配置为同步或异步模式）
- HA模块自动检测存活和控制主从切换
- 连接池、流量控制、防闪断
- 七天内任意时间点恢复、SQL审计



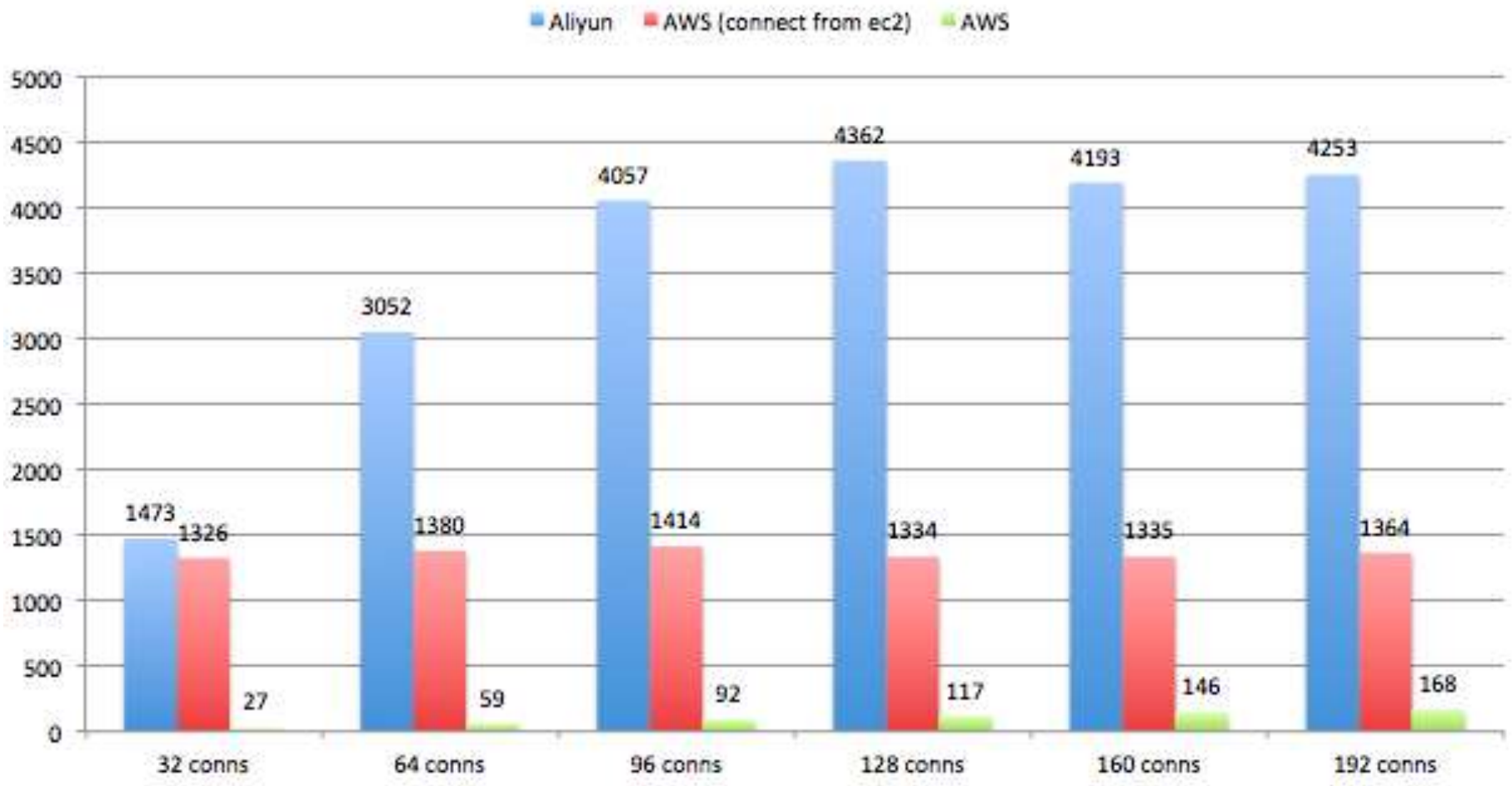
阿里云RDS PG的内核增强

- 稳定性与性能
  - Checkpointer优化
  - Clog优化
- 安全与权限管理
  - 防SQL注入
  - 新的rds_superuser角色
  - 允许用户自己管理插件
  - 修补安全上的漏洞（函数安全性增强）
- 监控与审计
  - 日志输出SQL的更多信息
- 对Proxy的支持
  - 对用户透明
  - 连接池
  - …
- 逻辑和增量复制
  - 增加日志解析插件，配合DRC完成增量迁移与异构复制
- 自定义中文分词
- ……

阿里云RDS PG的云运维

- 监控
  - 秒级的主机、网络链路监控（Cloudmonitor）
  - 数据库性能数据收集，间隔为30s（Controller）
  - 自动监控脚本部署与报警（Alimonitor）
  - 控制器任务状态监控
- 问题处理
  - 数据库错误日志分析系统（Cleaneye）
  - 自动修复系统（Robot）
  - “速报”系统
  - 全量SQL审计
- 智能运维
  - CloudDBA

阿里云RDS PG的性能



谢谢！

- ***PostgreSQL，让数据变得好玩！***
- QQ/微信 ： 276732431
- BLOG ： http://blog.163.com/digoal@126

- 2015 PostgreSQL 用户大会
  - 11月20,21日　北京
  - http://postgres2015.eventdove.com/

- PostgreSQL 社区沟通渠道
- 微信公众号: postgres用户会
- 微信群: PG圈
- 微博: PostgreSQL用户会
- Q群:　3336901 ，100910388 ，5276420 ，191516184
- 邮件列表: **pggeneral@groups.163.com** **pgadmin@groups.163.com** pglecturers@groups.163.com
- WEB:　http://bbs.postgres.cn/　http://www.postgres.cn/