新手机 新应用 新娱乐

PostgreSQL 9.3 培训
Day 2

digoal.zhou

2013/12/5

SKY mobi
新手机 新应用 新娱乐

课程内容

- Day - 2
- PostgreSQL 索引介绍
- 目标:
  - 熟悉b-tree , hash , gist , gin , sp-gist几种索引类型的概念, 以及在什么样的场景应该使用何种索引
  - 了解全文检索的应用
- PostgreSQL查询优化
- 目标:
  - 了解explian SQL分析工具的使用, 理解explain 的代价计算原理, 并根据数据库硬件环境校准代价因子.
  - 理解explain 输出的含义 (如 组合行集 , 节点处理 , 合并连接, 哈希连接 等), 并可以结合explain的输出优化SQL.
- 连接池及数据库高速缓存
- 目标:
  - 以pgbouncer为例, 理解数据库连接池在短连接环境下的好处, 连接池的几种模式和使用场景
  - 本地和异地高速缓存的介绍, 如本地os 层缓存pgfincore, 异地K-V缓存pgmemcached的使用.
- 数据库扩展及复制
- 目标:
  - 了解数据库集群级流复制, 数据库热备份, 表级复制, 数据库在虚拟化环境下的使用注意事项

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 索引介绍

- 使用索引的好处
  - 利用索引进行排序减少CPU开销
  - 加速带条件的查询, 删除, 更新
  - 加速JOIN操作
  - 加速外键约束更新和删除操作
  - 加速唯一值约束, 排他约束
- 索引带来的弊端
  - 索引随着表的记录块的变迁需要更新, 因此会对这类操作带来一定的性能影响. (块不变更的情况下触发HOT特性, 可以不需要更新索引)
- 使用索引的注意事项
  - 正常创建索引时, 会阻断除查询以外的其他操作.
  - 使用并行CONCURRENTLY 选项后, 可以允许同时对表的DML操作, 但是对于频繁DML的表, 这种创建索引的时间非常长.
  - 某些索引不记录WAL, 所以如果有利于WAL进行数据恢复的情况(如crash recovery, 流复制, warm standby等), 这类索引在使用前需要重建. (HASH 索引)

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 索引介绍

- 索引类型
  - 根据不同的索引算法, PostgreSQL的算法分为 B-tree, Hash, GiST, SP-GiST, GIN
  - select amname from pg_am;
- 索引应用场景
  - PostgreSQL不同的索引类别支持的索引访问操作符也有区别, 以下为不同的索引类型对应的系统默认的索引策略
- Btree, 同时还支持前导模糊查询(like 'xxx%' 或 ~ '^xxx'), 忽略大小写字符前导模糊查询(ILIKE 'xxx%' 或 ~* '^xxx')
  - <
  - <=
  - =
  - >=
  - >
- Hash
  - =
- Gin, 支持多值列的索引, 例如数组类型, 全文检索类型, 例如以下为一维数组类型对应的GIN索引已实现的访问策略操作符
  - <@ -- 被包含 array[1,2,3] <@ array[2,3,1]
  - @> -- 包含 array[1,2,3] @> array[2]
  - = -- 相等 array[1,2,3] = array[1,2,3]
  - && -- 相交 array[1,2,3] && array[2]

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 索引介绍

- Gist, 并不是单类的索引, 可以认为它是一种索引框架, 支持许多不同的索引策略(operator class),
- 例如二维几何类型的以下操作符支持通过Gist索引访问
  - << -- 严格在左侧, 例如circle '((0,0),1)' << circle '((5,0),1)'
  - &< -- 表示左边的平面体不会扩展到超过右边的平面体的右边. 例如box '((0,0),(1,1))' &< box '((0,0),(2,2))'
  - &> -- 表示左边的平面体不会扩展到超过右边的平面体的左边. 例如box '((0,0),(3,3))' &> box '((0,0),(2,2))'
  - >> -- 严格在右
  - <<| -- 严格在下
  - &<| -- 不会扩展到超出上面
  - |&> -- 不会扩展到超出下面
  - |>> -- 严格在上
  - @> -- 包含
  - <@ -- 被包含
  - ~= -- 相同
  - && -- 相交
  - http://www.postgresql.org/docs/9.3/static/functions-geometry.html
  - 除此之外 , gist索引还支持近邻排序. 例如
    - SELECT * FROM places ORDER BY location <-> point '(101,456)' LIMIT 10;
  - 另外contrib中也提供了一些gist索引策略.

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 索引介绍

- Sp-Gist, 与gist类似, 也是一种索引框架, 支持基于磁盘存储的非平衡数据结构, 如四叉树, k-d树, radix树.
- 例如二维的point类型, gist索引支持的操作符如下
  - <<
  - >>
  - ~=
  - <@
  - <^ -- 在下面, circle '((0,0),1)' <^ circle '((0,5),1)'
  - >^ -- 在上面, circle '((0,5),1)' >^ circle '((0,0),1)'

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 索引使用场景举例

- 利用索引进行排序减少CPU开销
- 加速带条件的查询, 删除, 更新
- 加速JOIN操作
- 加速外键约束更新和删除操作
- 加速唯一值约束, 排他约束

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 索引使用场景举例

- 利用索引进行排序减少CPU开销
- 1. 查询条件就是索引列
- digoal=# create table test(id int, info text, crt_time timestamp);
- CREATE TABLE
- digoal=# insert into test select generate_series(1,10000), md5(random()::text),clock_timestamp();
- INSERT 0 10000
- digoal=# create index idx_test_1 on test(id);
- CREATE INDEX
- digoal=# explain analyze select * from test where id<100 order by id;
- QUERY PLAN
- ---
- Index Scan using idx_test_1 on test (cost=0.29..162.61 rows=3333 width=44) (actual time=0.036..0.069 rows=99 loops=1)
- Index Cond: (id < 100)
- Total runtime: 0.107 ms
- (3 rows)

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 索引使用场景举例

- 利用索引进行排序减少CPU开销
- 2. 查询条件不是索引列
- digoal=# explain analyze select * from test where info='620f5eaeaf0d7cf48cd1fa6c410bad49' order by id;
- QUERY PLAN
- ---
- Sort (cost=219.01..219.01 rows=1 width=45) (actual time=2.240..2.240 rows=1 loops=1)
- Sort Key: id
- Sort Method: quicksort Memory: 25kB
- -> Seq Scan on test (cost=0.00..219.00 rows=1 width=45) (actual time=0.016..2.201 rows=1 loops=1)
- Filter: (info = '620f5eaeaf0d7cf48cd1fa6c410bad49'::text)
- Rows Removed by Filter: 9999
- Total runtime: 2.273 ms
- (7 rows)

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 索引使用场景举例

- 利用索引进行排序减少CPU开销
- 2. 查询条件不是索引列
- digoal=# set enable_seqscan=off;
- SET
- digoal=# explain analyze select * from test where info='620f5eaeaf0d7cf48cd1fa6c410bad49' order by id;
- QUERY PLAN
- ---
- Index Scan using idx_test_1 on test (cost=0.29..299.29 rows=1 width=45) (actual time=0.027..3.628 rows=1 loops=1)
- Filter: (info = '620f5eaeaf0d7cf48cd1fa6c410bad49'::text)
- Rows Removed by Filter: 9999
- Total runtime: 3.661 ms
- (4 rows)
- 这个只是例子, 不一定适合实际应用场景.
- 如果info的选择性好的话, 在info上面加索引时比较妥当的.

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 索引使用场景举例

- 加速带条件的查询, 删除, 更新
- digoal=# explain analyze select * from test where id=1;
- QUERY PLAN
- ---
- Index Scan using idx_test_1 on test (cost=0.29..2.30 rows=1 width=45) (actual time=0.014..0.015 rows=1 loops=1)
- Index Cond: (id = 1)
- Total runtime: 0.039 ms
- (3 rows)
- 在没有索引的情况下的查询效率 :
- set enable_indexscan=off;
- set enable_bitmapscan=off;
- digoal=# explain analyze select * from test where id=1;
- QUERY PLAN
- ---
- Seq Scan on test (cost=0.00..219.00 rows=1 width=45) (actual time=0.017..1.744 rows=1 loops=1)
- Filter: (id = 1)
- Rows Removed by Filter: 9999
- Total runtime: 1.773 ms
- (4 rows)

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 索引使用场景举例

- 加速JOIN操作
- digoal=# create table test1(id int, info text, crt_time timestamp);
- CREATE TABLE
- digoal=# insert into test1 select generate_series(1,10000), md5(random()::text),clock_timestamp();
- INSERT 0 10000
- test1表没有索引时
- digoal=# explain analyze select t1.*,t2.* from test t1 join test1 t2 on (t1.id=t2.id and t2.id=1);
- QUERY PLAN
- ---
- Nested Loop (cost=0.29..221.31 rows=1 width=90) (actual time=0.028..1.708 rows=1 loops=1)
- -> Index Scan using idx_test_1 on test t1 (cost=0.29..2.30 rows=1 width=45) (actual time=0.015..0.016 rows=1 loops=1)
- Index Cond: (id = 1)
- -> Seq Scan on test1 t2 (cost=0.00..219.00 rows=1 width=45) (actual time=0.010..1.686 rows=1 loops=1)
- Filter: (id = 1)
- Rows Removed by Filter: 9999
- Total runtime: 1.768 ms
- (7 rows)

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 索引使用场景举例

- 加速JOIN操作
- digoal=# create index idx_test1_1 on test1(id);
- CREATE INDEX
- digoal=# explain analyze select t1.*,t2.* from test t1 join test1 t2 on (t1.id=t2.id and t2.id=1);
- QUERY PLAN
- ---
- Nested Loop (cost=0.57..4.61 rows=1 width=90) (actual time=0.045..0.046 rows=1 loops=1)
- -> Index Scan using idx_test_1 on test t1 (cost=0.29..2.30 rows=1 width=45) (actual time=0.012..0.012 rows=1 loops=1)
- Index Cond: (id = 1)
- -> Index Scan using idx_test1_1 on test1 t2 (cost=0.29..2.30 rows=1 width=45) (actual time=0.029..0.030 rows=1 loops=1)
- Index Cond: (id = 1)
- Total runtime: 0.089 ms
- (6 rows)

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 索引使用场景举例

- 加速JOIN操作
- MERGE JOIN也能用到索引.
- digoal=# explain analyze select t1.*,t2.* from test t1 join test1 t2 on (t1.id=t2.id);
- QUERY PLAN
-
- ---
- ---
- Merge Join (cost=0.57..698.57 rows=10000 width=90) (actual time=0.024..14.468 rows=10000 loops=1)
- Merge Cond: (t1.id = t2.id)
- -> Index Scan using idx_test_1 on test t1 (cost=0.29..274.29 rows=10000 width=45) (actual time=0.010..3.754 rows=10000 loops=1)
- -> Index Scan using idx_test1_1 on test1 t2 (cost=0.29..274.29 rows=10000 width=45) (actual time=0.007..3.715 rows=10000 loops=
- 1)
- Total runtime: 15.429 ms
- (5 rows)

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 索引使用场景举例

- 加速JOIN操作
- 在没有索引的情况下, merge join增加排序开销.
- digoal=# explain analyze select t1.*,t2.* from test t1 join test1 t2 on (t1.id=t2.id);
- QUERY PLAN
- ---
- Merge Join (cost=1716.77..1916.77 rows=10000 width=90) (actual time=8.220..17.291 rows=10000 loops=1)
- Merge Cond: (t1.id = t2.id)
- -> Sort (cost=858.39..883.39 rows=10000 width=45) (actual time=4.177..5.211 rows=10000 loops=1)
- Sort Key: t1.id
- Sort Method: quicksort Memory: 1018kB
- -> Seq Scan on test t1 (cost=0.00..194.00 rows=10000 width=45) (actual time=0.008..1.757 rows=10000 loops=1)
- -> Sort (cost=858.39..883.39 rows=10000 width=45) (actual time=4.035..5.300 rows=10000 loops=1)
- Sort Key: t2.id
- Sort Method: quicksort Memory: 1018kB
- -> Seq Scan on test1 t2 (cost=0.00..194.00 rows=10000 width=45) (actual time=0.006..1.752 rows=10000 loops=1)
- Total runtime: 18.420 ms
- (11 rows)

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 索引使用场景举例

- 加速外键约束更新和删除操作
- digoal=# create table p(id int primary key, info text, crt_time timestamp);
- CREATE TABLE
- digoal=# create table f(id int primary key, p_id int references p(id) on delete cascade on update cascade, info text, crt_time timestamp);
- CREATE TABLE
- digoal=# insert into p select generate_series(1,10000), md5(random()::text), clock_timestamp();
- INSERT 0 10000
- digoal=# insert into f select generate_series(1,10000), generate_series(1,10000), md5(random()::text), clock_timestamp();
- INSERT 0 10000

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 索引使用场景举例

- 加速外键约束更新和删除操作
- 在f表的p_id未加索引时, 更新p.id
- digoal=# explain (analyze,verbose,costs,buffers,timing) update p set id=0 where id=1;
- QUERY PLAN
- ---
- Update on postgres.p (cost=0.29..2.30 rows=1 width=47) (actual time=0.082..0.082 rows=0 loops=1)
- Buffers: shared hit=8
- -> Index Scan using p_pkey on postgres.p (cost=0.29..2.30 rows=1 width=47) (actual time=0.021..0.022 rows=1 loops=1)
- Output: 0, info, crt_time, ctid
- Index Cond: (p.id = 1)
- Buffers: shared hit=3
- Trigger RI_ConstraintTrigger_a_92560 for constraint f_p_id_fkey on p: time=2.630 calls=1
- Trigger RI_ConstraintTrigger_c_92562 for constraint f_p_id_fkey on f: time=0.059 calls=1
- Total runtime: 2.820 ms
- (9 rows)

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 索引使用场景举例

- 加速外键约束更新和删除操作
- 增加f表的p_id列上的索引
- digoal=# create index idx_f_1 on f(p_id);
- CREATE INDEX
- digoal=# explain (analyze,verbose,costs,buffers,timing) update p set id=1 where id=0;
- QUERY PLAN
- ---
- Update on postgres.p (cost=0.29..2.30 rows=1 width=47) (actual time=0.067..0.067 rows=0 loops=1)
- Buffers: shared hit=8
- -> Index Scan using p_pkey on postgres.p (cost=0.29..2.30 rows=1 width=47) (actual time=0.018..0.020 rows=1 loops=1)
- Output: 1, info, crt_time, ctid
- Index Cond: (p.id = 0)
- Buffers: shared hit=3
- Trigger RI_ConstraintTrigger_a_92560 for constraint f_p_id_fkey on p: time=0.471 calls=1
- Trigger RI_ConstraintTrigger_c_92562 for constraint f_p_id_fkey on f: time=0.053 calls=1
- Total runtime: 0.636 ms
- (9 rows)

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 索引使用场景举例

- 索引在排他约束中的使用
- 对排他操作符的要求, 左右操作数互换对结果没有影响. 例如x=y, y=x 结果都为true或unknown.
- 用法举例
- digoal=# CREATE TABLE test(id int,geo point,EXCLUDE USING btree (id WITH pg_catalog.=));
- CREATE TABLE
- digoal=# insert into test (id) values (1);
- INSERT 0 1
- digoal=# insert into test (id) values (1);
- ERROR: 23P01: conflicting key value violates exclusion constraint "test_id_excl"
- DETAIL: Key (id)=(1) conflicts with existing key (id)=(1).
- SCHEMA NAME: postgres
- TABLE NAME: test
- CONSTRAINT NAME: test_id_excl
- LOCATION: check_exclusion_constraint, execUtils.c:1337

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 索引使用场景举例

- 加速唯一值约束
- Primary KEY
- Unique KEY

SKY mobi
新手机 新应用 新娱乐

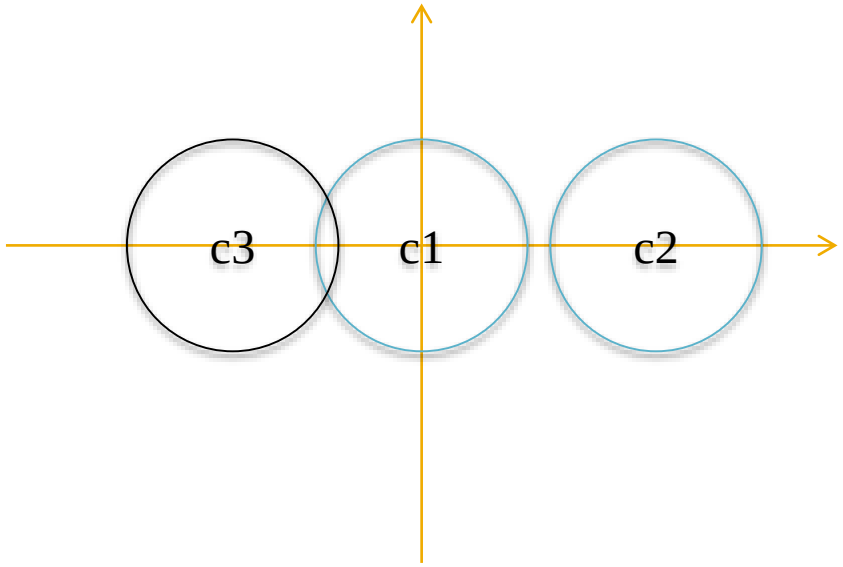
PostgreSQL 索引使用场景举例

- digoal=# CREATE TABLE test(id int,geo point,EXCLUDE USING spGIST (geo WITH pg_catalog.~=));
- CREATE TABLE
- 索引如下：
- digoal=# \d test
- Table "postgres.test"
- Column | Type | Modifiers
- --------+---------+-----------
- id | integer |
- geo | point |
- Indexes:
- "test_geo_excl" EXCLUDE USING spgist (geo WITH ~=)

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 索引使用场景举例

- 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>).



SKY mobi
新手机 新应用 新娱乐

PostgreSQL 索引使用场景举例

- digoal=# CREATE TABLE test(id int,geo circle,EXCLUDE USING GIST (geo WITH pg_catalog.&&));
- CREATE TABLE

- digoal=# CREATE TABLE test(id int,geo circle,EXCLUDE USING GIST (geo WITH pg_catalog.~=));
- CREATE TABLE

- 以下例子左右操作数互换后得到的结果不一致, 所以这类操作符不允许创建排他索引.
- digoal=# CREATE TABLE test(id int,geo point,EXCLUDE USING spGIST (geo WITH pg_catalog.<^));
- ERROR: 42809: operator <^(point,point) is not commutative
- DETAIL: Only commutative operators can be used in exclusion constraints.
- LOCATION: ComputeIndexAttrs, indexcmds.c:1132
- digoal=# CREATE TABLE test(id int,geo point,EXCLUDE USING btree (id WITH pg_catalog.>));
- ERROR: 42809: operator >(integer,integer) is not commutative
- DETAIL: Only commutative operators can be used in exclusion constraints.
- LOCATION: ComputeIndexAttrs, indexcmds.c:1132

SKY mobi
新手机 新应用 新娱乐

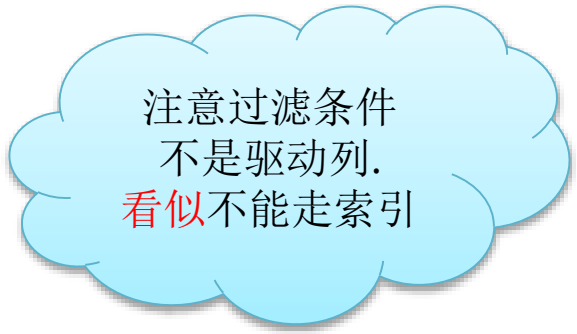
是否使用索引和什么有关?

- 是否使用索引和什么有关?
- 首先是前面提到的Access Method, 然后是使用的operator class, 以及opc中定义的operator或function.
- 这些都满足后, 还要遵循CBO的选择.
  - #seq_page_cost = 1.0
  - #random_page_cost = 4.0
  - #cpu_tuple_cost = 0.01
  - #cpu_index_tuple_cost = 0.005
  - #cpu_operator_cost = 0.0025
  - #effective_cache_size = 128MB
- 遵循完CBO的选择, 还需要符合当前配置的Planner 配置.
  - #enable_bitmapscan = on
  - #enable_hashagg = on
  - #enable_hashjoin = on
  - #enable_indexscan = on
  - #enable_material = on
  - #enable_mergejoin = on
  - #enable_nestloop = on
  - #enable_seqscan = on
  - #enable_sort = on
  - #enable_tidscan = on

SKY mobi
新手机 新应用 新娱乐

多列索引的使用

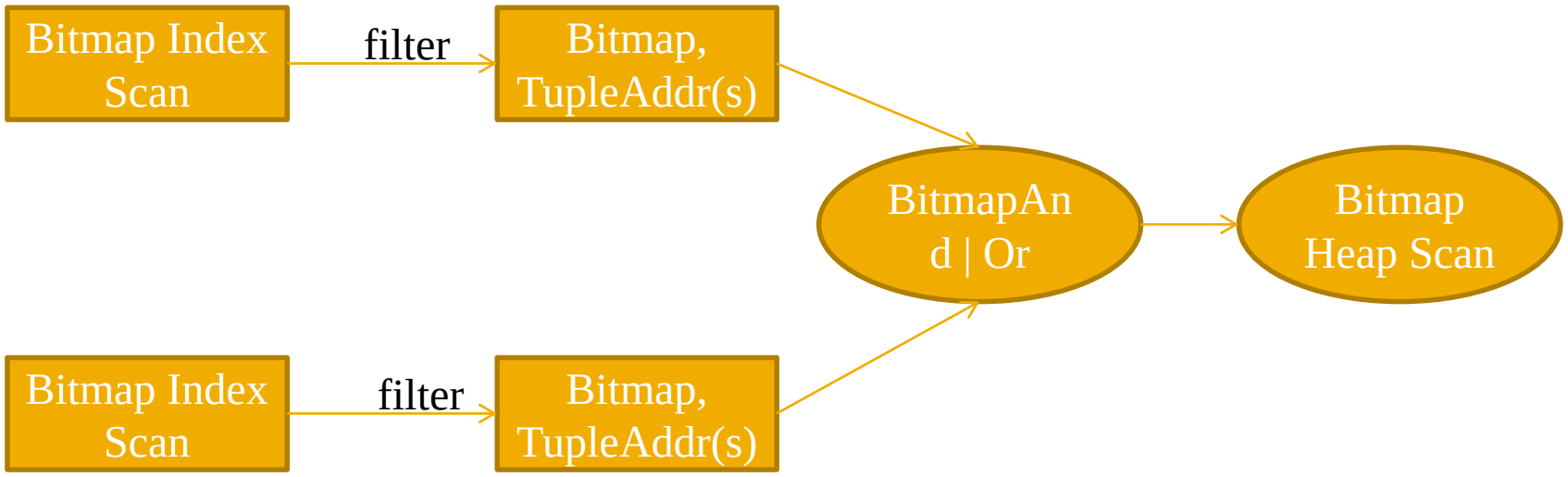
- Multicolumn Index
  - 多列索引，使用任何列作为条件，只要条件中的操作符或函数能满足opclass的匹配，都可以使用索引，索引被扫描的部分还是全部基本取决于条件中是否有索引的第一列作为条件之一.
- 例子
- postgres=# create table test (c1 int,c2 int);
- postgres=# insert into test select 1,generate_series(1,100000);
- postgres=# create index idx_test_1 on test(c1,c2);
- postgres=# analyze test;
- postgres=# explain select * from test where c2=100;
- Seq Scan on test (cost=0.00..1693.00 rows=1 width=8)
- Filter: (c2 = 100)
- postgres=# set enable_seqscan=off;
- postgres=# explain analyze select * from test where c2=100;
- Index Scan using idx_test_1 on test (cost=0.00..1858.27 rows=1 width=8) (actual time=0.104..7.045 rows=1 loops=1)
- Index Cond: (c2 = 100)



SKY mobi
新手机 新应用 新娱乐

索引合并查询

- Combining Multiple Indexes
- src/backend/executor
- 例如



SKY mobi
新手机 新应用 新娱乐

索引合并查询

- Combining Multiple Indexes
- 单列索引的多条件匹配合并
  - postgres=# create table test (id int primary key,info text unique);
  - postgres=# insert into test select generate_series(1,100000),'digoal'||generate_series(1,100000);
  - postgres=# explain analyze select * from test where id=1 or id=1000;
  - Bitmap Heap Scan on test (cost=8.54..16.20 rows=2 width=36) (actual time=0.034..0.036 rows=2 loops=1)
  - Recheck Cond: ((id = 1) OR (id = 1000))
  - -> BitmapOr (cost=8.54..8.54 rows=2 width=0) (actual time=0.023..0.023 rows=0 loops=1)
  - -> Bitmap Index Scan on test_pkey (cost=0.00..4.27 rows=1 width=0) (actual time=0.012..0.012 rows=1 loops=1)
  - Index Cond: (id = 1)
  - -> Bitmap Index Scan on test_pkey (cost=0.00..4.27 rows=1 width=0) (actual time=0.009..0.009 rows=1 loops=1)
  - Index Cond: (id = 1000)

SKY mobi
新手机 新应用 新娱乐

索引和collate的匹配

- collection
- 例子
  - CREATE TABLE test1c (
  - id integer,
  - content varchar COLLATE "x"
  -);

  - CREATE INDEX test1c_content_index ON test1c (content);
  - SELECT * FROM test1c WHERE content > constant;

  - -- 以下SQL不能使用索引test1c_content_index
  - SELECT * FROM test1c WHERE content > constant COLLATE "y";

  - -- 需建立与y COLLATE对应的索引, 以上这条SQL才会走索引.
  - CREATE INDEX test1c_content_y_index ON test1c (content COLLATE "y");

SKY mobi
新手机 新应用 新娱乐

部分值索引

- partial index
- 例子
- -- 部分约束
- --去除common值 id=1, 这个值有10W条, 走索引根本不合适. partial 索引很好的避免了此类情况.
  - postgres=# create table test(id int,info text);
  - postgres=# insert into test select 1,'digoal'||generate_series(1,100000);
  - postgres=# insert into test select generate_series(1,1000),'digoal'||generate_series(1,1000);
  - postgres=# create index idx_test_1 on test(id) where id<>1;
  - postgres=# explain select * from test where id=1;
  - Seq Scan on test (cost=0.00..1791.00 rows=100000 width=15)
  - Filter: (id = 1)
  - postgres=# explain select * from test where id=100;
  - Index Scan using idx_test_1 on test (cost=0.00..8.27 rows=1 width=15)
  - Index Cond: (id = 100)

SKY mobi
新手机 新应用 新娱乐

部分值索引

- -- 部分索引在非索引列的使用
  - postgres=# explain select * from test where info='digoal' and id=1;
  - QUERY PLAN
  - ----------
  - Seq Scan on test (cost=0.00..2041.00 rows=1 width=15)
  - Filter: ((info = 'digoal'::text) AND (id = 1))
  - postgres=# create index idx_test_2 on test(id) where info='digoal100';
  - postgres=# explain select * from test where info='digoal100';
  - QUERY PLAN
  - ----------
  - Index Scan using idx_test_2 on test (cost=0.00..8.27 rows=1 width=15)
  - (1 row)

SKY mobi
新手机 新应用 新娱乐

部分值索引

- -- 部分索引的好处, 为什么要去除common 值
  - postgres=# drop index idx_test_1;
  - postgres=# drop index idx_test_2;
  - postgres=# explain select * from test where id=1;
  - QUERY PLAN
  - ---
  - Seq Scan on test (cost=0.00..1791.00 rows=100000 width=15)
  - Filter: (id = 1)
  - -- 为什么会走全表扫描
  - postgres=# select id,count(*) from test group by id order by count(*) desc limit 10;
  - id | count
  - -----+--------
  - 1 | 100001
  - 120 | 1
  - 887 | 1
  - 681 | 1

SKY mobi
新手机 新应用 新娱乐

函数和表达式索引

- 函数索引和表达式索引
- 表达式索引
  - postgres=# explain select * from test where id+1=100;
  - QUERY PLAN
  - ----------
  - Seq Scan on test (cost=0.00..2059.86 rows=505 width=15)
  - Filter: ((id + 1) = 100)
  - postgres=# create index idx_test_1 on test((id+1));
  - CREATE INDEX
  - postgres=# explain select * from test where id+1=100;
  - QUERY PLAN
  - ----------
  - Bitmap Heap Scan on test (cost=12.18..577.45 rows=505 width=15)
  - Recheck Cond: ((id + 1) = 100)
  - -> Bitmap Index Scan on idx_test_1 (cost=0.00..12.05 rows=505 width=0)
  - Index Cond: ((id + 1) = 100)

SKY mobi
新手机 新应用 新娱乐

函数和表达式索引

- 函数索引
  - -- 以下区分大小写的场景无法使查询走普通的索引.
  - postgres=# create table test (id int,info text,crt_time timestamp(0));
  - postgres=# insert into test select generate_series(1,100000),'digoal'||generate_series(1,100000),clock_timestamp( );
  - postgres=# create index idx_test_info on test(info);
  - postgres=# explain select * from test where info ~* '^a';
  - Seq Scan on test (cost=0.00..1887.00 rows=10 width=23)
  - Filter: (info ~* '^a'::text)
  - -- 忽略大小写的ilike和~* 要走索引的话, 开头的字符只能是大小写一致的, 字母不行. 数字可以. 例如字母a区分大小写, 数字0不区分大小写.索引中的条目也就有差别.
  - postgres=# explain select * from test where info ~* '^0';
  - Index Scan using idx_test_info on test (cost=0.00..8.28 rows=10 width=23)
  - Index Cond: ((info >= '0'::text) AND (info < '1'::text))
  - Filter: (info ~* '^0'::text)

SKYMOBI 新手机 新应用 新娱乐

函数和表达式索引

- 函数索引
  - -- 要让字母也可以走忽略大小写的索引如何做呢?
  - -- 函数索引,但是函数必须是immutable状态的
  - 过滤条件中也必须使用和创建的索引相同声明
  - postgres=# select proname,provolatile from pg_proc where proname='lower';
  - proname | provolatile
  - lower | i
  - postgres=# create index idx_test_info_1 on test(lower(info));
  - CREATE INDEX
  - postgres=# explain select * from test where lower(info) ~ '^a';
  - Bitmap Heap Scan on test (cost=13.40..648.99 rows=500 width=23)
  - Filter: (lower(info) ~ '^a'::text)
  - -> Bitmap Index Scan on idx_test_info_1 (cost=0.00..13.27 rows=500 width=0)
  - Index Cond: ((lower(info) >= 'a'::text) AND (lower(info) < 'b'::text))
  - (4 rows)

SKY mobi
新手机 新应用 新娱乐

函数和表达式索引

- 作为查询条件的 函数 或 常量 或 变量 或 子查询
  - 优化器需要知道给operator的参数值才能通过pg_statistic中统计到的表柱状图来计算走索引还是走全表扫描或者其他planner的开销最小, 如果传入的是个变量则通常不能使用索引扫描.
  - 几个时间函数的稳定性：
  - postgres=# create index idx_test_1 on test (crt_time);
  - postgres=# select proname,proargtypes,provolatile from pg_proc where prorettype in (1114,1184) order by proargtypes;
  - proname | proargtypes | provolatile
  - ----------+---------+----------
  - transaction_timestamp | | s
  - statement_timestamp | | s
  - pg_stat_get_bgwriter_stat_reset_time | | s
  - pg_conf_load_time | | s
  - pg_postmaster_start_time | | s
  - pg_last_xact_replay_timestamp | | v

SKY mobi
新手机 新应用 新娱乐

函数和表达式索引

- 作为查询条件的函数 或 常量 或 变量 或 子查询
  - clock_timestamp | | v
  - now | | s

  - postgres=# explain select * from test where crt_time = clock_timestamp();
  - Seq Scan on test (cost=0.00..2137.00 rows=100000 width=23)
  - Filter: (crt_time = clock_timestamp())

  - postgres=# explain select * from test where crt_time = now();
  - Index Scan using idx_test_1 on test (cost=0.00..8.28 rows=1 width=23)
  - Index Cond: (crt_time = now())

  - postgres=# alter function now() strict volatile;
  - postgres=# explain select * from test where crt_time = now();
  - Seq Scan on test (cost=0.00..2137.00 rows=100000 width=23)
  - Filter: (crt_time = now())

SKY mobi
新手机 新应用 新娱乐

函数和表达式索引

- 作为查询条件的函数 或 常量 或 变量 或 子查询
  - postgres=# alter function clock_timestamp() strict immutable;
  - ALTER FUNCTION
  - postgres=# explain select * from test where crt_time = clock_timestamp();
  - QUERY PLAN
  - ---
  - Index Scan using idx_test_1 on test (cost=0.00..8.28 rows=1 width=23)
  - Index Cond: (crt_time = '2012-04-30 15:32:02.559888+08'::timestamp with time zone)

- 作为过滤条件的函数, immutable 和 stable 的函数在优化器开始计算COST前会把函数值算出来. 而volatile的函数, 是在执行SQL的时候运行的, 所以无法在优化器计算执行计划的阶段得到函数值, 也就无法和pg_statistic中的信息比对到底是走索引呢还是全表扫描或其他执行计划.

SKY mobi
新手机 新应用 新娱乐

函数和表达式索引

- 表达式作为过滤条件时, 同样的道理, 表达式不会在优化器计算执行计划的过程中运算, 所以也不能走最优的执行计划.
  - postgres=# explain select * from test where crt_time = (select now());
  - QUERY PLAN
  - ----------
  - Seq Scan on test (cost=0.01..1887.01 rows=100000 width=23)
  - Filter: (crt_time = $0)
  - InitPlan 1 (returns $0)
  - -> Result (cost=0.00..0.01 rows=1 width=0)
  - (4 rows)

- 绑定变量的使用场景, 通常需要5次后得到generic plan.
- http://blog.163.com/digoal@126/blog/static/163877040201211245243225 1/

SKY mobi
新手机 新应用 新娱乐

HOT Update

- 索引指针结构
- ItemPointers (index) ->
- ItemId数据结构
  - (Array of (lp_off:15bit, lp_flags:2bit,lp_len:15bit) pairs pointing to the actual items. 4 bytes per ItemId.)
- -> Item (tuple)

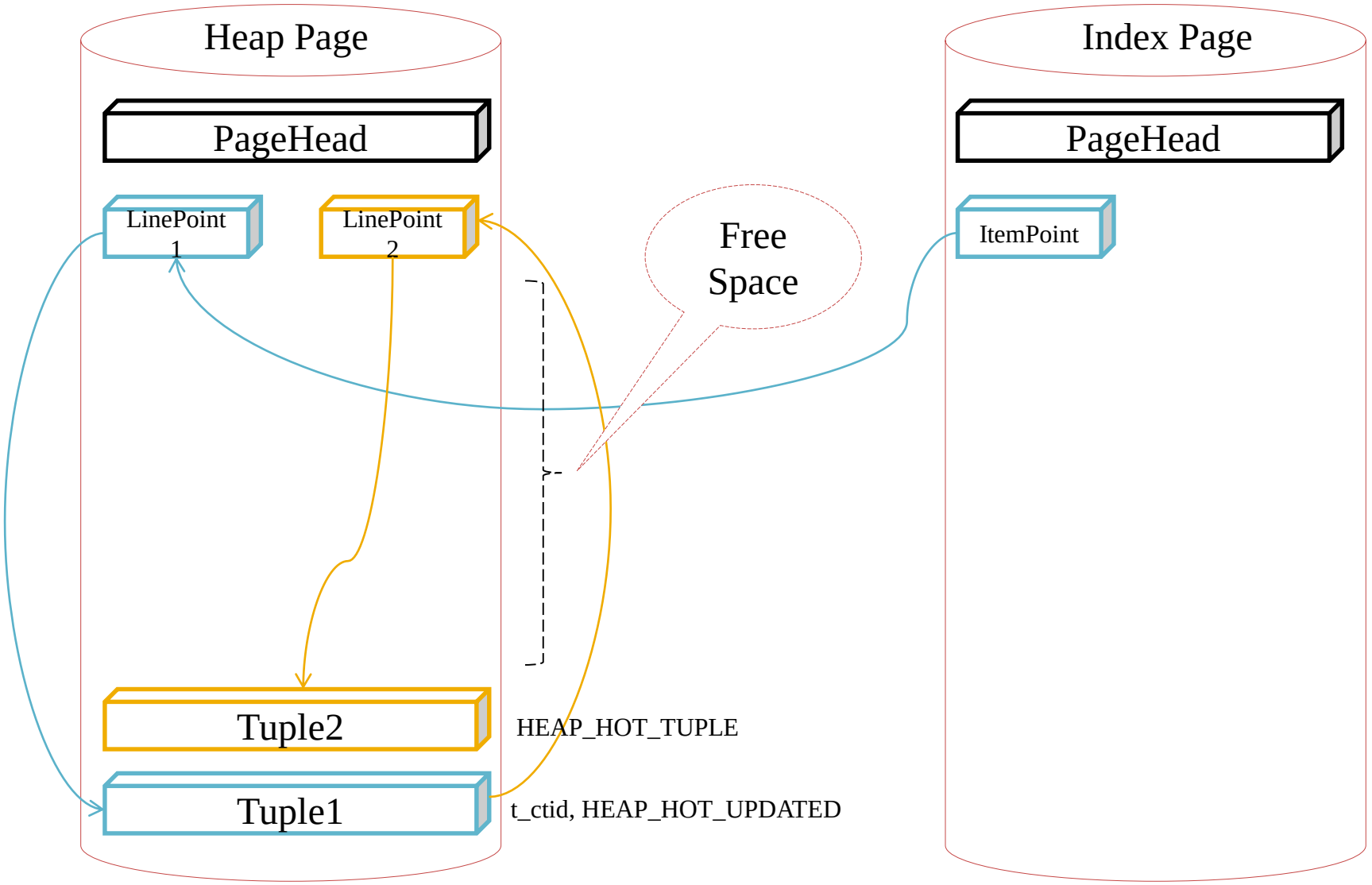
SKY mobi
新手机 新应用 新娱乐

HOT Update

- Heap-Only Tuple Benefit :
  - eliminates redundant index entries
  - allows the re-use of space taken by DELETEd or obsoleted UPDATEd tuples without performing a table-wide vacuum.
- Example
  - Update 1: Index points to 1
  - line points [1] [2]
  - Items [11111111]->[2222222222]
  - Update 2: Index points to 1
  - line point [1]->[2]
  - Items [2222222222]
  - Update 3: Index points to 1
  - line points [1]->[2] [3]
  - Items [2222222222]->[3333333333]
  - Update 4: Index points to 1
  - line points [1]------>[3]
  - Items [3333333333]

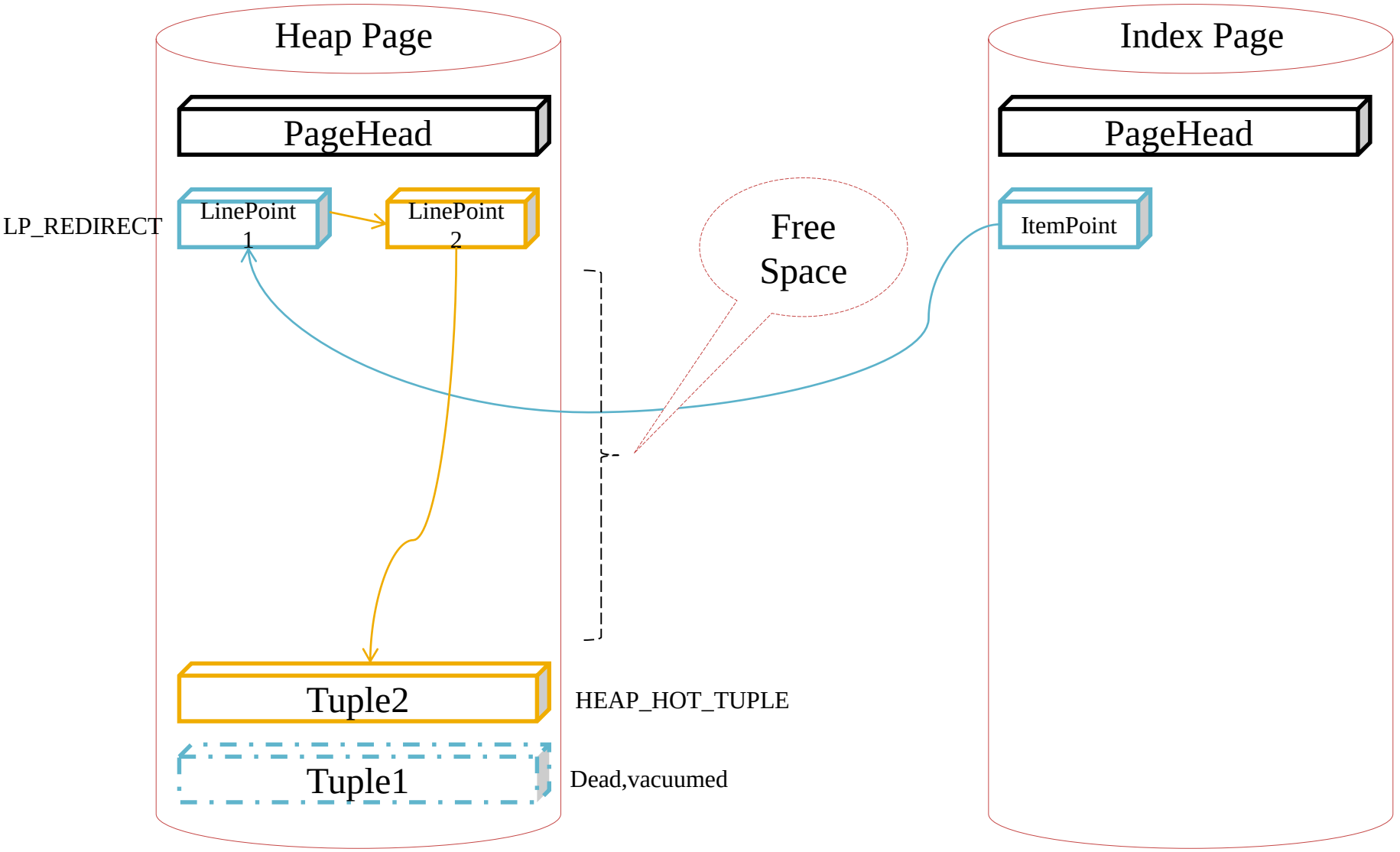
SKY mobi
新手机 新应用 新娱乐

HOT Update



SKY mobi
新手机 新应用 新娱乐

HOT Update



SKY mobi
新手机 新应用 新娱乐

HOT Update

- 利用pageinspect extension 来观察HOT

- postgres=# create extension pageinspect;
- postgres=# create table hot_test (id int primary key,info text);
- postgres=# insert into hot_test values (1,'digoal');
- -- 因为是从0号page开始插入, 这里就省去了查询ctid等过程.直接切入0号page.
- -- 当前的page信息
- postgres=# select * from page_header(get_raw_page('hot_test',0));
- lsn | tli | flags | lower | upper | special | pagesize | version | prune_xid
- 2/75B27878 | 1 | 0 | 28 | 8152 | 8192 | 8192 | 4 | 0

- -- 当前的item信息
- postgres=# select * from heap_page_items(get_raw_page('hot_test',0));
- lp | lp_off | lp_flags | lp_len | t_xmin | t_xmax | t_field3 | t_ctid | t_infomask2 | t_infomask | t_hoff | t_bits | t_oid
- 1 | 8152 | 1 | 35 | 1864 | 0 | 0 | (0,1) | 2 | 2050 | 24 | |

SKY mobi
新手机 新应用 新娱乐

HOT Update

- -- 当前索引的page信息
- postgres=# select * from page_header(get_raw_page('hot_test_pkey',0));
- lsn | tli | flags | lower | upper | special | pagesize | version | prune_xid
- 2/75B278B0 | 1 | 0 | 48 | 8176 | 8176 | 8192 | 4 | 0

- -- 当前索引的item信息
- postgres=# select * from heap_page_items(get_raw_page('hot_test_pkey',0));
- lp | lp_off | lp_flags | lp_len | t_xmin | t_xmax | t_field3 | t_ctid | t_infomask2 | t_infomask | t_hoff | t_bits | t_oid
- 1 | 12642 | 2 | 2 | | | | | | | | |
- 2 | 2 | 0 | 0 | | | | | | | | |
- 3 | 1 | 0 | 0 | | | | | | | | |
- 4 | 0 | 0 | 0 | | | | | | | | |
- 5 | 1 | 0 | 0 | | | | | | | | |
- 6 | 0 | 0 | 0 | | | | | | | | |

SKY mobi
新手机 新应用 新娱乐

HOT Update

- -- 更新一次后
- postgres=# update hot_test set info='new' where id=1;
- -- item信息
- postgres=# select * from heap_page_items(get_raw_page('hot_test',0));
- lp | lp_off | lp_flags | lp_len | t_xmin | t_xmax | t_field3 | t_ctid | t_infomask2 | t_infomask | t_hoff | t_bits | t_oid
- 1 | 8152 | 1 | 35 | 1864 | 1867 | 0 | (0,2) | 16386 | 258 | 24 | |
- 2 | 8120 | 1 | 32 | 1867 | 0 | 0 | (0,2) | 32770 | 10242 | 24 | |

itemID中的信息

tuple中的信息,对应第一幅图

- -- 索引的item信息(没有变化)
- postgres=# select * from heap_page_items(get_raw_page('hot_test_pkey',0));
- lp | lp_off | lp_flags | lp_len | t_xmin | t_xmax | t_field3 | t_ctid | t_infomask2 | t_infomask | t_hoff | t_bits | t_oid
- -- 内容略

SKY mobi
新手机 新应用 新娱乐

HOT Update

itemID中的信息
对应第二幅图

- -- vacuum 后
- postgres=# vacuum hot_test ;
- VACUUM
- postgres=# select * from heap_page_items(get_raw_page('hot_test',0));
- lp | lp_off | lp_flags | lp_len | t_xmin | t_xmax | t_field3 | t_ctid | t_infomask2 | t_infomask | t_hoff | t_bits | t_oid
- 1 | 2 | 2 | 0 | | | | | | | | |
- 2 | 8160 | 1 | 32 | 1867 | 0 | 0 | (0,2) | 32770 | 10498 | 24 | |
- -- 多次更新后
- postgres=# update hot_test set info='new' where id=1;
- postgres=# update hot_test set info='new' where id=1;
- postgres=# update hot_test set info='new' where id=1;
- postgres=# update hot_test set info='new' where id=1;
- postgres=# update hot_test set info='new' where id=1;
- postgres=# update hot_test set info='new' where id=1;

SKY mobi
新手机 新应用 新娱乐

HOT Update

- postgres=# select * from heap_page_items(get_raw_page('hot_test',0));
- lp | lp_off | lp_flags | lp_len | t_xmin | t_xmax | t_field3 | t_ctid | t_infomask2 | t_infomask | t_hoff | t_bits | t_oid
- 1 | 2 | 2 | 0 | | | | | | | | |
- 2 | 8160 | 1 | 32 | 1867 | 1868 | 0 | (0,3) | 49154 | 9474 | 24 | |
- 3 | 8128 | 1 | 32 | 1868 | 1869 | 0 | (0,4) | 49154 | 9474 | 24 | |
- 4 | 8096 | 1 | 32 | 1869 | 1870 | 0 | (0,5) | 49154 | 9474 | 24 | |
- 5 | 8064 | 1 | 32 | 1870 | 1871 | 0 | (0,6) | 49154 | 9474 | 24 | |
- 6 | 8032 | 1 | 32 | 1871 | 1872 | 0 | (0,7) | 49154 | 9474 | 24 | |
- 7 | 8000 | 1 | 32 | 1872 | 1873 | 0 | (0,8) | 49154 | 8450 | 24 | |
- 8 | 7968 | 1 | 32 | 1873 | 0 | 0 | (0,8) | 32770 | 10242 | 24 | |

注意redirect后,lp_off的值表示第几条itemid, 而不是offset_bytes.

SKY mobi
新手机 新应用 新娱乐

HOT Update

- -- vacuum后
- postgres=# vacuum hot_test ;
- postgres=# select * from heap_page_items(get_raw_page('hot_test',0));
- lp | lp_off | lp_flags | lp_len | t_xmin | t_xmax | t_field3 | t_ctid | t_infomask2 | t_infomask | t_hoff | t_bits | t_oid
- 1 | 8 | 2 | 0 | | | | | | | | |
- 2 | 0 | 0 | 0 | | | | | | | | |
- 3 | 0 | 0 | 0 | | | | | | | | |
- 4 | 0 | 0 | 0 | | | | | | | | |
- 5 | 0 | 0 | 0 | | | | | | | | |
- 6 | 0 | 0 | 0 | | | | | | | | |
- 7 | 0 | 0 | 0 | | | | | | | | |
- 8 | 8160 | 1 | 32 | 1873 | 0 | 0 | (0,8) | 32770 | 10498 | 24 | |

- Use pageinspect EXTENSION view PostgreSQL Page's raw infomation
- http://blog.163.com/digoal@126/blog/static/163877040201142732659 60/

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 全文检索

- 和全文检索相关的有两个数据类型
- tsvector -- 文本在经过全文检索标准化处理后得到类型, 处理后的文本包括(分词去重复后排序, 分词的位置, 分词的权重结构(一共可以指定4个权重ABCD, D默认不显示))
- tsquery -- 需要检索的分词组合, 组合类型包括&, | , ! (与, 或, 否). 同时还支持分词的权重, 分词的前导匹配.

- 详细介绍
- http://www.postgresql.org/docs/9.3/static/datatype-textsearch.html
- http://www.postgresql.org/docs/9.3/static/textsearch.html
- 字符串 到 tsvector 的默认转换例子 :
- digoal=# select $$hello world, i'm digoal.$$::tsvector;
- tsvector
- ----------
- 'digoal.' 'hello' 'i''m' 'world,'
- 这种转换后得到的tsvector不包含分词的位置信息和权重信息. 只有排序后的分词.
- 权重和位置信息可以提现在文本中(权重一般用来表示该分词所在的级别,如目录,或正文?),
- digoal=# select $$hello:1b world,:1a i'm:3D digoal.$$::tsvector;
- tsvector
- ----------
- 'digoal.' 'hello':1B 'i''m':3 'world,':1A

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 全文检索

- 使用to_tsvector, 可以指定不同的语言配置, 把文本根据指定的语言配置进行分词.
- 例如, 使用西班牙语和英语得到的tsvector值是不一样的.
- digoal=# select to_tsvector('english', $$Hello world, I'm digoal.$$);
- to_tsvector
- ----------
- 'digoal':5 'hello':1 'm':4 'world':2
- (1 row)

- digoal=# select to_tsvector('spanish', $$Hello world, I'm digoal.$$);
- to_tsvector
- ----------
- 'digoal':5 'hell':1 'i':3 'm':4 'world':2
- (1 row)

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 全文检索

- 查看系统中已经安装的全文检索配置.
- digoal=# \dF *
- List of text search configurations
- Schema | Name | Description
- pg_catalog | danish | configuration for danish language
- pg_catalog | dutch | configuration for dutch language
- pg_catalog | english | configuration for english language
- pg_catalog | finnish | configuration for finnish language
- pg_catalog | french | configuration for french language
- pg_catalog | german | configuration for german language
- pg_catalog | hungarian | configuration for hungarian language
- pg_catalog | italian | configuration for italian language
- pg_catalog | norwegian | configuration for norwegian language
- pg_catalog | portuguese | configuration for portuguese language
- pg_catalog | romanian | configuration for romanian language
- pg_catalog | russian | configuration for russian language
- pg_catalog | simple | simple configuration
- pg_catalog | spanish | configuration for spanish language
- pg_catalog | swedish | configuration for swedish language
- pg_catalog | turkish | configuration for turkish language
- 后面的例子中会有中文的安装

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 全文检索

- tsquery的例子
- &, | , ! 组合, 分组使用括号
- SELECT 'fat & rat'::tsquery;
- tsquery
- ---
- 'fat' & 'rat'

- SELECT 'fat & (rat | cat)'::tsquery;
- tsquery
- ---
- 'fat' & ('rat' | 'cat')

- SELECT 'fat & rat & ! cat'::tsquery;
- tsquery
- ---
- 'fat' & 'rat' & !'cat‘

- 支持指定权重
- SELECT 'fat:ab & cat'::tsquery;
- tsquery
- ---
- 'fat':AB & 'cat‘

- 支持前导匹配
- SELECT 'super:*'::tsquery;
- tsquery
- ---
- 'super':*

- 使用to_tsquery转换时, 也可以带上语言配置
- digoal=# SELECT to_tsquery('english', 'Fat:ab & Cats');
- to_tsquery
- ---
- 'fat':AB & 'cat'

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 全文检索

- tsquery的例子, @@操作符的使用, 文本匹配.
- digoal=# select to_tsvector('spanish', $$Hello world, I'm digoal.$$);
- to_tsvector
- ----------
- 'digoal':5 'hell':1 'i':3 'm':4 'world':2
- (1 row)

- digoal=# select to_tsvector('spanish', $$Hello world, I'm digoal.$$) @@ 'i:b'::tsquery;
- ?column?
- ----------
- f
- (1 row)

- digoal=# select to_tsvector('spanish', $$Hello world, I'm digoal.$$) @@ 'i:d'::tsquery;
- ?column?
- ----------
- t
- (1 row)

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 全文检索

- tsquery的例子, @@操作符的使用, 文本匹配.
- digoal=# select to_tsvector('spanish', $$Hello world, I'm digoal.$$) @@ 'i:*'::tsquery;
- ?column?
- ----------
- t
- (1 row)

- digoal=# select to_tsvector('spanish', $$Hello world, I'm digoal.$$) @@ 'i:d*'::tsquery;
- ?column?
- ----------
- t
- (1 row)

- digoal=# select to_tsvector('spanish', $$Hello world, I'm digoal.$$) @@ 'i:a*'::tsquery;
- ?column?
- ----------
- f
- (1 row)

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 全文检索

- 和全文检索相关的函数和操作符
- 详细介绍
- http://www.postgresql.org/docs/9.3/static/functions-textsearch.html

Operator	Description	Example	Result
@@	tsvector matches tsquery ?	to_tsvector('fat cats ate rats') @@ to_tsquery('cat & rat')	t
@@@	deprecated synonym for @@	to_tsvector('fat cats ate rats') @@@ to_tsquery('cat & rat')	t
||	concatenate tsvectors	'a:1 b:2'::tsvector || 'c:1 d:2 b:3'::tsvector	'a':1 'b':2,5 'c':3 'd':4
&&	AND tsquerys together	'fat | rat'::tsquery && 'cat'::tsquery	('fat' | 'rat') & 'cat'
||	OR tsquerys together	'fat | rat'::tsquery || 'cat'::tsquery	('fat' | 'rat') | 'cat'
!!	negate a tsquery	!! 'cat'::tsquery	!'cat'
@>	tsquery contains another ?	'cat'::tsquery @> 'cat & rat'::tsquery	f
<@	tsquery is contained in ?	'cat'::tsquery <@ 'cat & rat'::tsquery	t

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 全文检索

- 全文检索的索引使用
- digoal=# create table ts(id int, info tsvector, crt_time timestamp);
- CREATE TABLE
- digoal=# insert into ts values (1, $$Hello world, i'm digoal.$$, now());
- INSERT 0 1
- digoal=# create index idx_ts_1 on ts using gin (info);
- CREATE INDEX
- digoal=# select * from ts where info @@ 'digoal.'::tsquery;
- id | info | crt_time
- ----+----------+----------
- 1 | 'Hello' 'digoal.' 'i''m' 'world,' | 2013-12-09 16:35:55.635111
- (1 row)

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 全文检索

- 全文检索的索引使用(tsvector支持的索引策略gin, gist, btree)
- GIN索引策略, 可用于tsvector包含tsquery的查询匹配
- digoal=# set enable_seqscan=off;
- SET
- digoal=# explain analyze select * from ts where info @@ 'digoal.'::tsquery;
- QUERY PLAN
- ---
- Bitmap Heap Scan on ts (cost=2.00..3.01 rows=1 width=44) (actual time=0.021..0.021 rows=1 loops=1)
- Recheck Cond: (info @@ '"digoal."'::tsquery)
- -> Bitmap Index Scan on idx_ts_1 (cost=0.00..2.00 rows=1 width=0) (actual time=0.016..0.016 rows=1 loops=1)
- Index Cond: (info @@ '"digoal."'::tsquery)
- Total runtime: 0.061 ms
- (5 rows)

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 全文检索

- 全文检索的索引使用
- GiST索引策略, 可用于包含匹配
- digoal=# drop index idx_ts_1;
- DROP INDEX
- digoal=# create index idx_ts_1 on ts using gist (info);
- CREATE INDEX
- digoal=# explain analyze select * from ts where info @@ 'digoal.'::tsquery;
- QUERY PLAN
- ---
- Index Scan using idx_ts_1 on ts (cost=0.12..2.14 rows=1 width=44) (actual time=0.016..0.017 rows=1 loops=1)
- Index Cond: (info @@ '"digoal."'::tsquery)
- Total runtime: 0.055 ms
- (3 rows)

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 全文检索

- 中文全文检索举例
- http://blog.163.com/digoal@126/blog/static/16387704020125214101069 3/

- 中文全文检索语言配置安装简介
- 安装cmake
- tar -zxvf cmake-2.8.8.tar.gz
- cd cmake-2.8.8
- ./bootstrap --prefix=/opt/cmake2.8.8
- gmake
- gmake install

- vi ~/.bash_profile
- export PATH=/opt/cmake2.8.8/bin:$PATH
- . ~/.bash_profile

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 全文检索

- 安装crf
- tar -zxvf CRF++-0.57.tar.gz
- cd CRF++-0.57
- ./configure
- gmake
- gmake install

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 全文检索

- 安装nlpbamboo
- vi ~/.bash_profile
- export PGHOME=/opt/pgsql
- export PATH=$PGHOME/bin:/opt/bamboo/bin:/opt/cmake2.8.8/bin:$PATH:.
- export LD_LIBRARY_PATH=$PGHOME/lib:/lib64:/usr/lib64:/usr/local/lib64:/lib:/usr/lib:/usr/local/lib:.
- . ~/.bash_profile

- tar -jxvf nlpbamboo-1.1.2.tar.bz2
- cd nlpbamboo-1.1.2
- mkdir build
- cd build
- cmake .. -DCMAKE_BUILD_TYPE=release
- gmake all
- gmake install

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 全文检索

- 配置默认的lib库配置
- echo "/usr/lib" >>/etc/ld.so.conf (这个命令是bamboo对应的动态链接库)
- echo "/usr/local/lib" >>/etc/ld.so.conf (这个命令是CRF对应的动态链接库)
- ldconfig -f /etc/ld.so.conf

- 测试是否加入正常
- ldconfig -p|grep bambo
- libbamboo.so.2 (libc6,x86-64) => /usr/lib/libbamboo.so.2
- libbamboo.so (libc6,x86-64) => /usr/lib/libbamboo.so
- ldconfig -p|grep crf
- libcrfpp.so.0 (libc6,x86-64) => /usr/local/lib/libcrfpp.so.0
- libcrfpp.so (libc6,x86-64) => /usr/local/lib/libcrfpp.so

- 加入索引
- cd /opt/bamboo
- wget http://nlpbamboo.googlecode.com/files/index.tar.bz2
- tar -jxvf index.tar.bz2

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 全文检索

- 编译PostgreSQL支持模块
- export PATH=/opt/pgsql/bin:$PATH
- cd /opt/bamboo/exts/postgres/chinese_parser
- make
- make install
- touch $PGHOME/share/tsearch_data/chinese_utf8.stop

- cd /opt/bamboo/exts/postgres/pg_tokenize
- make
- make install

- 安装PostgreSQL支持模块
- su - postgres
- cd $PGHOME/share/contrib/
- psql -h 127.0.0.1 postgres postgres -f chinese_parser.sql
- psql -h 127.0.0.1 postgres postgres -f pg_tokenize.sql

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 全文检索

- 查看全文检索配置中加入了chinesecfg的配置.
- postgres=# select * from pg_ts_config;
- cfgname | cfgnamespace | cfgowner | cfgparser
- ------------+--------------+----------+-----------
- simple | 11 | 10 | 3722
- danish | 11 | 10 | 3722
- …….
- russian | 11 | 10 | 3722
- spanish | 11 | 10 | 3722
- swedish | 11 | 10 | 3722
- turkish | 11 | 10 | 3722
- chinesecfg | 11 | 10 | 33463
- (17 rows)

SKYY mobi
新手机 新应用 新娱乐

PostgreSQL 全文检索

- 测试tokenize分词函数
- postgres=# select * from tokenize('你好我是中国人');
- tokenize
- ----------
- 你好 我 是 中国 人
- (1 row)
- postgres=# select * from tokenize('中华人民共和国');
- tokenize
- ----------
- 中华人民共和国
- (1 row)
- postgres=# select * from tokenize('百度');
- tokenize
- ----------
- 百度
- (1 row)

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 全文检索

- postgres=# select * from tokenize('谷歌');
- tokenize
- ----------
- 谷歌
- (1 row)
- postgres=# select * from tokenize('今年是龙年');
- tokenize
- ---------------
- 今年 是 龙年
- (1 row)

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 全文检索

- 测试全文检索类型转换函数
- postgres=# select * from to_tsvector('chinesecfg', '你好,我是中国人.目前在杭州斯凯做数据库相关的工作.');
- to_tsvector
-
- ----------
- ',':2 '.':7,17 '中国':5 '人':6 '你好':1 '做':12 '在':9 '工作':16 '我':3 '数据库':13 '斯凯':11 '是':4 '杭州':10 '的':15 '目前':8 '相关':14
- (1 row)

- 索引的效果
- postgres=# create table blog (id serial primary key, user_id int8, title text, content text, ts_title tsvector, ts_content tsvector);
- NOTICE: CREATE TABLE will create implicit sequence "blog_id_seq" for serial column "blog.id"
- NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "blog_pkey" for table "blog"
- CREATE TABLE
- postgres=# create index idx_blog_ts1 on blog using gist(ts_title);
- CREATE INDEX
- postgres=# create index idx_blog_ts2 on blog using gist(ts_content);
- CREATE INDEX

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 全文检索

```
postgres=# explain select user_id,title from blog where ts_content @@ to_tsquery('函数 & 中国');
                                 QUERY PLAN
-----------------------------------------------------------------------------
 Index Scan using idx_blog_ts2 on blog  (cost=0.00..4.27 rows=1 width=40)
   Index Cond: (ts_content @@ to_tsquery('函数 & 中国'::text))
(2 rows)

postgres=# explain select user_id,title from blog where ts_content @@ to_tsquery('函数 & 表');
                                 QUERY PLAN
-----------------------------------------------------------------------------
 Index Scan using idx_blog_ts2 on blog  (cost=0.00..4.27 rows=1 width=40)
   Index Cond: (ts_content @@ to_tsquery('函数 & 表'::text))
(2 rows)
```

SKY mobi
新手机 新应用 新娱乐

pg_trgm 近似匹配

- pg_trgm插件
- http://blog.163.com/digoal@126/blog/static/16387704020134161021418 01/
- http://www.postgresql.org/docs/9.3/static/pgtrgm.html

- 原理
- 首先把一个字符串拆分成多个独立的字符串(拆分间隔为非字符如数字空格标点)
- 然后再拆分后的独立字符串前加2个空格后加1个空格, 然后把字符串切分成相近的3个字符一组的一些单元.
- 两个字符串的相似度匹配和他们被切分成的单元共性有关.
- 宽字符的支持需要将数据库collate调整为C以外的值.
- 例如, aa b,c首先拆分成独立的字符串(aa , b , c)
- 然后在aa前加2个空格, 后面加一个空格
- 在b前加2个空格, 后面加一个空格
- 在c前加2个空格, 后面加一个空格
- 最后这几组字符串进行3个一组的切分
- digoal=# select show_trgm($$aa b,c$$);
- show_trgm
- ---
- {" a"," b"," c"," aa"," b "," c ","aa "}
- (1 row)

SKY mobi
新手机 新应用 新娱乐

pg_trgm 近似匹配

- 宽字符支持
- digoal=# \l
- List of databases
- Name | Owner | Encoding | Collate | Ctype | Access privileges
- -----------+----------+----------+------------+------------+-----------------------
- digoal | postgres | UTF8 | en_US.utf8 | en_US.utf8 |
- postgres | postgres | UTF8 | en_US.utf8 | en_US.utf8 |
- template0 | postgres | UTF8 | en_US.utf8 | en_US.utf8 | =c/postgres +
- | | | | | postgres=CTc/postgres
- template1 | postgres | UTF8 | en_US.utf8 | en_US.utf8 | =c/postgres +
- | | | | | postgres=CTc/postgres
- (4 rows)

- digoal=# select show_trgm('刘德华');
- show_trgm
- ---------------------------------
- {0xb207ac,0xd2efc5,67N,0x6ff95f}
- (1 row)

SKY mobi
新手机 新应用 新娱乐

pg_trgm 近似匹配

- 无法使用宽字符的场景
- ^
- digoal=# select show_trgm('你好') collate "zh_CN.utf8";
- show_trgm
- ----------
- {}
- (1 row)

- digoal=# \l
- List of databases
- Name | Owner | Encoding | Collate | Ctype | Access privileges
- ----------+----------+----------+---------+-------+----------------------
- digoal | postgres | UTF8 | C | C |

SKY mobi
新手机 新应用 新娱乐

练习

- 各种索引的合理使用
- 全文检索的使用以及中文分词插件的安装和使用
- 近似匹配插件pg_trgm的安装和使用

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 查询优化

- 了解explian SQL分析工具的使用, 理解explain 的代价计算原理, 并根据数据库硬件环境校准代价因子.
- 理解 explain 输出的含义 (如 组合行集 , 节点处理 , 合并连接, 哈希连接 等), 并可以结合explain的输出优化SQL.

SKY mobi
新手机 新应用 新娱乐

EXPLAIN 语法

- EXPLAIN [(option [, ...])] statement
- EXPLAIN [ANALYZE] [VERBOSE] statement

- where option can be one of:

- ANALYZE [boolean] -- 执行statement, 得到真实的运行时间以及统计信息
- VERBOSE [boolean] -- 输出详细信息
- COSTS [boolean] -- 输出cost值, 默认打开
- BUFFERS [boolean] -- 输出本次QUERY shared 或 local buffer的信息. 包括命中,未命中,脏, 写
- TIMING [boolean] -- 输出时间开销
- FORMAT { TEXT | XML | JSON | YAML } -- 输出格式

- 需要特别注意analyze的使用, 会真的的执行SQL, 所以一般不要使用, 特别是DML.

SKY mobi
新手机 新应用 新娱乐

EXPLAIN 输出的含义

- 例子讲解
- digoal=# explain (analyze, verbose, costs, buffers, timing) select count(*) from tbl_cost_align;
- QUERY PLAN
-
- Aggregate (cost=220643.00..220643.01 rows=1 width=0) (actual time=4637.754..4637.754 rows=1 loops=1)
- Output: count(*) -- 这个节点的输出, 聚合, 输出第一行前的开销是220643.00.
- -- 聚合的开销=220643.00 - 195393.00
- Buffers: shared hit=4925 read=89468 -- 这个节点以及下级节点的BUFFER统计项
- -> Seq Scan on postgres.tbl_cost_align (cost=0.00..195393.00 rows=10100000 width=0) (actual time=0.018..3119.291 rows=10100000 loops=1) -- 这个节点的路径(全表扫描)
- -- 0.00表示输出第一行前的成本, 如这里输出第一行前不需要排序为0.00. 后面是这个节点真实的时间.
- Output: id, info, crt_time -- 这个节点输出的列
- Buffers: shared hit=4925 read=89468 -- 这个节点的shared buffer命中4925个page, 从磁盘读取89468个page(如果shared buffer够大, 第二次执行的时候应该全部hit.)
- Total runtime: 4637.805 ms -- 总的执行时间
- (7 rows)

SKYY mobi
新手机 新应用 新娱乐

EXPLAIN 输出的含义

- 组合行集例子
- digoal=# explain (analyze, verbose, costs, buffers, timing) select 1 union select 1; -- union去重复, 所以有sort节点
- QUERY PLAN
- Unique (cost=0.05..0.06 rows=2 width=0) (actual time=0.049..0.051 rows=1 loops=1)
- Output: (1)
- Buffers: shared hit=3
- -> Sort (cost=0.05..0.06 rows=2 width=0) (actual time=0.047..0.047 rows=2 loops=1)
- Output: (1)
- Sort Key: (1)
- Sort Method: quicksort Memory: 25kB
- Buffers: shared hit=3
- -> Append (cost=0.00..0.04 rows=2 width=0) (actual time=0.006..0.007 rows=2 loops=1)
- -> Result (cost=0.00..0.01 rows=1 width=0) (actual time=0.003..0.003 rows=1 loops=1)
- Output: 1
- -> Result (cost=0.00..0.01 rows=1 width=0) (actual time=0.000..0.000 rows=1 loops=1)
- Output: 1
- Total runtime: 0.136 ms
- (14 rows)

SKY mobi
新手机 新应用 新娱乐

EXPLAIN 输出的含义

- 嵌套连接例子
- nested loop join: The right relation is scanned once for every row found in the left relation. This strategy is easy to implement but can be very time consuming. (However, if the right relation can be scanned with an index scan, this can be a good strategy. It is possible to use values from the current row of the left relation as keys for the index scan of the right.)

- for tuple in 左表查询 loop
- 右表查询(根据左表查询得到的行作为右表查询的条件依次输出最终结果)
- end loop;

- 适合右表的关联列发生在唯一键值列或者主键列上的情况.

SKYY mobi
新手机 新应用 新娱乐

EXPLAIN 输出的含义

- 嵌套连接例子
- digoal=# explain (analyze, verbose, costs, buffers, timing) select f.* from f,p where f.p_id=p.id and f.p_id<10;
- QUERY PLAN
- ---
- Nested Loop (cost=0.57..22.29 rows=9 width=49) (actual time=0.011..0.042 rows=9 loops=1)
- Output: f.id, f.p_id, f.info, f.crt_time
- Buffers: shared hit=31
- -> Index Scan using idx_f_1 on postgres.f (cost=0.29..2.45 rows=9 width=49) (actual time=0.005..0.010 rows=9 loops=1) -- 左表
- Output: f.id, f.p_id, f.info, f.crt_time
- Index Cond: (f.p_id < 10)
- Buffers: shared hit=4
- -> Index Only Scan using p_pkey on postgres.p (cost=0.29..2.19 rows=1 width=4) (actual time=0.002..0.003 rows=1 loops=9) -- 右表
- Output: p.id
- Index Cond: (p.id = f.p_id)
- Heap Fetches: 9
- Buffers: shared hit=27
- Total runtime: 0.072 ms
- (13 rows)

SKY mobi
新手机 新应用 新娱乐

EXPLAIN 输出的含义

- 哈希连接例子
- hash join: the right relation is first scanned and loaded into a hash table, using its join attributes as hash keys. Next the left relation is scanned and the appropriate values of every row found are used as hash keys to locate the matching rows in the table.

- 首先右表扫描加载到内存HASH表, hash key为JOIN列.
- 然后左表扫描, 并与内存中的HASH表进行关联, 输出最终结果.

SKY mobi
新手机 新应用 新娱乐

EXPLAIN 输出的含义

- 哈希连接例子
- digoal=# explain (analyze, verbose, costs, buffers, timing) select f.* from f,p where f.p_id=p.id and f.p_id<10;
- Hash Join (cost=2.56..234.15 rows=9 width=49) (actual time=0.047..3.905 rows=9 loops=1)
- Output: f.id, f.p_id, f.info, f.crt_time
- Hash Cond: (p.id = f.p_id) -- HASH join key, f.p_id.
- Buffers: shared hit=98
- -> Seq Scan on postgres.p (cost=0.00..194.00 rows=10000 width=4) (actual time=0.014..2.016 rows=10000 loops=1) -- 左表
- Output: p.id, p.info, p.crt_time
- Buffers: shared hit=94
- -> Hash (cost=2.45..2.45 rows=9 width=49) (actual time=0.017..0.017 rows=9 loops=1)
- Output: f.id, f.p_id, f.info, f.crt_time
- Buckets: 1024 Batches: 1 Memory Usage: 1kB -- 右表加载到内存, hash key是join key f.p_id
- Buffers: shared hit=4
- -> Index Scan using idx_f_1 on postgres.f (cost=0.29..2.45 rows=9 width=49) (actual time=0.005..0.012 rows=9 loops=1) -- 右表
- Output: f.id, f.p_id, f.info, f.crt_time
- Index Cond: (f.p_id < 10)
- Buffers: shared hit=4
- Total runtime: 3.954 ms
- (16 rows)

SKY mobi
新手机 新应用 新娱乐

EXPLAIN 输出的含义

- 合并连接例子
- merge join: Each relation is sorted on the join attributes before the join starts. Then the two relations are scanned in parallel, and matching rows are combined to form join rows. This kind of join is more attractive because each relation has to be scanned only once. The required sorting might be achieved either by an explicit sort step, or by scanning the relation in the proper order using an index on the join key.

- 首先两个JOIN的表根据join key进行排序
- 然后根据join key的排序顺序并行扫描两个表进行匹配输出最终结果.
- 适合大表并且索引列进行关联的情况.

SKYY mobi
新手机 新应用 新娱乐

EXPLAIN 输出的含义

- 合并连接例子
- digoal=# explain (analyze, verbose, costs, buffers, timing) select f.* from f,p where f.p_id=p.id and f.p_id<10;
- QUERY PLAN
- Merge Join (cost=0.57..301.85 rows=9 width=49) (actual time=0.030..0.049 rows=9 loops=1)
- Output: f.id, f.p_id, f.info, f.crt_time
- Merge Cond: (f.p_id = p.id)
- Buffers: shared hit=8
- -> Index Scan using idx_f_1 on postgres.f (cost=0.29..2.45 rows=9 width=49) (actual time=0.005..0.012 rows=9 loops=1)
- Output: f.id, f.p_id, f.info, f.crt_time
- Index Cond: (f.p_id < 10)
- Buffers: shared hit=4
- -> Index Only Scan using p_pkey on postgres.p (cost=0.29..274.29 rows=10000 width=4) (actual time=0.017..0.022 rows=10 loops=1)
- Output: p.id
- Heap Fetches: 10
- Buffers: shared hit=4
- Total runtime: 0.118 ms
- (13 rows)

SKY mobi
新手机 新应用 新娱乐

EXPLAIN 成本计算

- 成本计算相关的参数和系统表或视图
- 表或视图
  - pg_stats
  - pg_class -- 用到relpages和reltuples
- 参数
  - seq_page_cost -- 全表扫描的单个数据块的代价因子
  - random_page_cost -- 索引扫描的单个数据块的代价因子
  - cpu_tuple_cost -- 处理每条记录的CPU开销代价因子
  - cpu_index_tuple_cost -- 索引扫描时每个索引条目的CPU开销代价因子
  - cpu_operator_cost -- 操作符或函数的开销代价因子

SKY mobi
新手机 新应用 新娱乐

EXPLAIN 成本计算

- pg_stats

Name	Type	References	Description
schemaname	name	pg_namespace.nspname	Name of schema containing table
tablename	name	pg_class.relname	Name of table
attname	name	pg_attribute.attname	Name of the column described by this row
inherited	bool		If true, this row includes inheritance child columns, not just the values in the specified table
null_frac	real		Fraction of column entries that are null
avg_width	integer		Average width in bytes of column's entries
n_distinct	real		If greater than zero, the estimated number of distinct values in the column. If less than zero, the negative of the number of distinct values divided by the number of rows. (The negated form is used when ANALYZE believes that the number of distinct values is likely to increase as the table grows; the positive form is used when the column seems to have a fixed number of possible values.) For example, -1 indicates a unique column in which the number of distinct values is the same as the number of rows.
most_common_vals	anyarray		A list of the most common values in the column. (Null if no values seem to be more common than any others.)
most_common_freqs	real[]		A list of the frequencies of the most common values, i.e., number of occurrences of each divided by total number of rows. (Null when most_common_vals is.)
histogram_bounds	anyarray		A list of values that divide the column's values into groups of approximately equal population. The values in most_common_vals, if present, are omitted from this histogram calculation. (This column is null if the column data type does not have a < operator or if the most_common_vals list accounts for the entire population.)
correlation	real		Statistical correlation between physical row ordering and logical ordering of the column values. This ranges from -1 to +1. When the value is near -1 or +1, an index scan on the column will be estimated to be cheaper than when it is near zero, due to reduction of random access to the disk. (This column is null if the column data type does not have a < operator.)
most_common_elems	anyarray		A list of non-null element values most often appearing within values of the column. (Null for scalar types.)
most_common_elem_freqs	real[]		A list of the frequencies of the most common element values, i.e., the fraction of rows containing at least one instance of the given value. Two or three additional values follow the per-element frequencies; these are the minimum and maximum of the preceding per-element frequencies, and optionally the frequency of null elements. (Null when most_common_elems is.)
elem_count_histogram	real[]		A histogram of the counts of distinct non-null element values within the values of the column, followed by the average number of distinct non-null elements. (Null for scalar types.)

SKY mobi
新手机 新应用 新娱乐

EXPLAIN 成本计算 - 全表扫描

- 例子
- 全表扫描的成本计算
- digoal=# explain select * from f;
- QUERY PLAN
- ---
- Seq Scan on f (cost=0.00..12999.00 rows=640000 width=49)
- (1 row)

- Cost是怎么得来的? 全表扫描的成本计算只需要用到pg_class.
- digoal=# select relpages,reltuples from pg_class where relname='f';
- relpages | reltuples
- ----------+----------
- 6599 | 640000
- (1 row)

SKY mobi
新手机 新应用 新娱乐

EXPLAIN 成本计算 - 全表扫描

- digoal=# show seq_page_cost;
- seq_page_cost
- ---
- 1
- (1 row)
- digoal=# show cpu_tuple_cost;
- cpu_tuple_cost
- ---
- 0.01
- (1 row)
- COST值 :
- digoal=# select 6599*1+640000*0.01;
- ?column?
- ---
- 12999.00
- (1 row)

SKY mobi
新手机 新应用 新娱乐

EXPLAIN 行数评估

- 从柱状图评估行数的例子
- EXPLAIN SELECT * FROM tenk1 WHERE unique1 < 1000;

- QUERY PLAN
- ---
- Bitmap Heap Scan on tenk1 (cost=24.06..394.64 rows=1007 width=244)
- Recheck Cond: (unique1 < 1000)
- -> Bitmap Index Scan on tenk1_unique1 (cost=0.00..23.80 rows=1007 width=0)
- Index Cond: (unique1 < 1000)
- 在8.3以及以前版本default_statistics_target默认是10, 也就是10个bucket.
- SELECT histogram_bounds FROM pg_stats
- WHERE tablename='tenk1' AND attname='unique1';
- histogram_bounds
- ---
- {0,993,1997,3050,4040,5036,5957,7057,8029,9016,9995}

SKY mobi
新手机 新应用 新娱乐

EXPLAIN 行数评估

- 这个例子的行选择性如下
- selectivity = (1 + (1000 - bucket[2].min)/(bucket[2].max - bucket[2].min))/num_buckets
- = (1 + (1000 - 993)/(1997 - 993))/10
- = 0.100697

- 最终得到的行数是：
- rows = rel_cardinality * selectivity
- = 10000 * 0.100697
- = 1007 (rounding off)

- 这里rel_cardinality = pg_class.reltuples.

SKY mobi
新手机 新应用 新娱乐

EXPLAIN 行数评估

- 从MCV(most common values)评估行数的例子
- EXPLAIN SELECT * FROM tenk1 WHERE stringu1 = 'CRAAAA';

- QUERY PLAN
- ---
- Seq Scan on tenk1 (cost=0.00..483.00 rows=30 width=244)
- Filter: (stringu1 = 'CRAAAA'::name)

- SELECT null_frac, n_distinct, most_common_vals, most_common_freqs FROM pg_stats
- WHERE tablename='tenk1' AND attname='stringu1';
- null_frac | 0
- n_distinct | 676
- most_common_vals | {EJAAAA,BBAAAA,CRAAAA,FCAAAA,FEAAAA,GSAAAA,JOAAAA,MCAAAA,NAAAAA,WGAAAA}
- most_common_freqs | {0.00333333,0.003,0.003,0.003,0.003,0.003,0.003,0.003,0.003,0.003}

SKY mobi
新手机 新应用 新娱乐

EXPLAIN 行数评估

- 行选择性如下, most common vals对应的占比most common freqs.
- selectivity = mcf[3]
- = 0.003

- 得到行数
- rows = 10000 * 0.003
- = 30

SKY mobi
新手机 新应用 新娱乐

EXPLAIN 行数评估

- 从MCV(most common values)和distinct值个数评估行数的例子
- EXPLAIN SELECT * FROM tenk1 WHERE stringu1 = 'xxx';

- QUERY PLAN
- ----------
- Seq Scan on tenk1 (cost=0.00..483.00 rows=15 width=244)
- Filter: (stringu1 = 'xxx'::name)

- 1减去所有MCV的占比, 再乘以 distinct值的个数减去MCV的个数
- selectivity = (1 - sum(mvf))/(num_distinct - num_mcv)
- = (1 - (0.00333333 + 0.003 + 0.003 + 0.003 + 0.003 + 0.003 +
- 0.003 + 0.003 + 0.003 + 0.003))/(676 - 10)
- = 0.0014559

- rows = 10000 * 0.0014559
- = 15 (rounding off)

SKY mobi
新手机 新应用 新娱乐

EXPLAIN 行数评估

- 从MCV(most common values)和柱状图评估行数的例子
- 条件中即包含了MCV又落在柱状图中的情况, 柱状图的统计中不包含MCV的值, 所以从柱状图中计算行的选择性时, 要乘以一个系数, 这个系数是1减去MCF的总和.
- EXPLAIN SELECT * FROM tenk1 WHERE stringu1 < 'IAAAAA';

- QUERY PLAN
- ---
- Seq Scan on tenk1 (cost=0.00..483.00 rows=3077 width=244)
- Filter: (stringu1 < 'IAAAAA'::name)
- SELECT histogram_bounds FROM pg_stats
- WHERE tablename='tenk1' AND attname='stringu1';

- histogram_bounds
- ---
- {AAAAAA,CQAAAA,FRAAAA,IBAAAA,KRAAAA,NFAAAA,PSAAAA,SGAAAA,VAAAAA,XLAAAA,ZZAAAA}

SKY mobi
新手机 新应用 新娱乐

EXPLAIN 行数评估

- 本例选择性MCV的占比
- selectivity = sum(relevant mvfs)
- = 0.00333333 + 0.003 + 0.003 + 0.003 + 0.003 + 0.003
- = 0.01833333
- 柱状图占比
- digoal=# select 1-(0.00333333 + 0.003 + 0.003 + 0.003 + 0.003 + 0.003 +
- digoal(# 0.003 + 0.003 + 0.003 + 0.003);
- ----------
- 0.96966667
- 0.298387是柱状图中通过类似(1 + (1000 - bucket[2].min)/(bucket[2].max - bucket[2].min))/num_buckets的算法得到
- selectivity = mcv_selectivity + histogram_selectivity * histogram_fraction
- = 0.01833333 + 0.298387 * 0.96966667
- = 0.307669

- rows = 10000 * 0.307669
- = 3077 (rounding off)

SKY mobi
新手机 新应用 新娱乐

EXPLAIN 行数评估

- 多个列查询条件的选择性相乘评估例子
- EXPLAIN SELECT * FROM tenk1 WHERE unique1 < 1000 AND stringu1 = 'xxx';
- QUERY PLAN
- ---
- Bitmap Heap Scan on tenk1 (cost=23.80..396.91 rows=1 width=244)
- Recheck Cond: (unique1 < 1000)
- Filter: (stringu1 = 'xxx'::name)
- -> Bitmap Index Scan on tenk1_unique1 (cost=0.00..23.80 rows=1007 width=0)
- Index Cond: (unique1 < 1000)
- 多列的选择性相乘得到最终的选择性
- selectivity = selectivity(unique1 < 1000) * selectivity(stringu1 = 'xxx')
- = 0.100697 * 0.0014559
- = 0.0001466
- rows = 10000 * 0.0001466
- = 1 (rounding off)

SKY mobi
新手机 新应用 新娱乐

EXPLAIN 成本计算 - 索引扫描

- 索引扫描时 ,和全表扫描不同, 扫描PAGE的开销是pages*random_page_cost
- 另外, 索引扫描一般都涉及操作符, 例如大于, 小于, 等于.
- 这些操作符对应的函数的COST乘以cpu_operator_cost 就得到这个操作符的代价因子. 乘以实际操作的行数就得到CPU操作符开销. Rows Explain中可能无输出.
- 索引的CPU开销则是实际扫描的索引条目数乘以cpu_index_tuple_cost. Rows Explain中无输出
- 最后的一个开销是实际返回或丢给上层的TUPLE带来的CPU开销. cpu_tuple_cost*实际扫描的行数. Rows Explain中有输出

SKY mobi
新手机 新应用 新娱乐

explain代价因子校准

- http://blog.163.com/digoal@126/blog/static/16387704020131025571737 9/
- 不同的硬件环境CPU性能, IO性能各不相同, 所以默认的代价因子可能不适合实际的硬件环境.
- 校准方法是求未知数的过程. 其中要用到第三方的工具得到一些比较容易得到的值.
- 这里有个例子, 根据SQL实际的执行时间, 计算代价因子的值.
- seq_page_cost和cpu_tuple_cost的校准 :
- seq_page_cost通过stap测试得到.
- cpu_tuple_cost通过公式得到.

- 创建测试表, 插入测试数据
- digoal=# create table tbl_cost_align (id int, info text, crt_time timestamp);
- CREATE TABLE
- digoal=# insert into tbl_cost_align select (random()*2000000000)::int, md5(random()::text), clock_timestamp() from generate_series(1,100000);
- INSERT 0 100000
- digoal=# insert into tbl_cost_align select (random()*2000000000)::int, md5(random()::text), clock_timestamp() from generate_series(1,10000000);
- INSERT 0 10000000
- 分析表
- digoal=# analyze tbl_cost_align;
- ANALYZE

SKY mobi
新手机 新应用 新娱乐

explain代价因子校准

- 得到表的PAGE数
- digoal=# select relpages from pg_class where relname='tbl_cost_align';
- relpages
- ----------
- 94393
- (1 row)
- 检查点
- digoal=# checkpoint;
- CHECKPOINT
- 停库
- pg93@db-172-16-3-150-> pg_ctl stop -m fast
- waiting for server to shut down.... done
- server stopped
- 把操作系统的缓存刷入硬盘
- [root@db-172-16-3-150 ssd1]# sync; echo 3 > /proc/sys/vm/drop_caches
- 以1号CPU亲和启动数据库, 0号CPU会带来一定的额外开销问题.
- pg93@db-172-16-3-150-> taskset -c 1 /home/pg93/pgsql9.3.1/bin/postgres >/dev/null 2>&1

SKY mobi
新手机 新应用 新娱乐

explain代价因子校准

- 启动一个客户端
- pg93@db-172-16-3-150-> psql
- psql (9.3.1)
- Type "help" for help.
- digoal=# select pg_backend_pid();
- pg_backend_pid
- ----------------
- 5727
- (1 row)

SKYMOBI 新手机 新应用 新娱乐

explain代价因子校准

- 使用stap跟踪, 得到seq_page_cost.
- [root@db-172-16-3-150 ~]# taskset -c 7 stap -e '
- global a
- probe process("/home/pg93/pgsql9.3.1/bin/postgres").mark("query__start") {
- delete a
- println("query__start ", user_string($arg1), "pid:", pid())
- }
- probe vfs.read.return {
- t = gettimeofday_ns() - @entry(gettimeofday_ns())
- # if (execname() == "postgres" && devname != "N/A")
- a[pid()] <<< t
- }
- probe process("/home/pg93/pgsql9.3.1/bin/postgres").mark("query__done") {
- if (@count(a[pid()]))
- printdln("**", pid(), @count(a[pid()]), @avg(a[pid()]))
- println("query__done ", user_string($arg1), "pid:", pid())
- if (@count(a[pid()])) {
- # 未完

SKY mobi
新手机 新应用 新娱乐

explain代价因子校准

- println(@hist_log(a[pid()]))
- #println(@hist_linear(a[pid()],1024,4096,100))
- }
- delete a
- }' -x 5727

- 执行SQL
- digoal=# explain (analyze,verbose,costs,buffers,timing) select * from tbl_cost_align;
- QUERY PLAN
-
- ---
- Seq Scan on postgres.tbl_cost_align (cost=0.00..195393.00 rows=10100000 width=45) (actual time=0.839..3260.695 rows=10100000 loops=1)
- Output: id, info, crt_time
- Buffers: shared read=94393 -- 注意这个read指的是未命中shared buffer, 如果是命中的话会有hit=?
- Total runtime: 4325.885 ms
- (4 rows)

explain代价因子校准

- stap的输出
- query__start explain (analyze,verbose,costs,buffers,timing) select * from tbl_cost_align;pid:5727
- 5727**94417**14329
- query__done explain (analyze,verbose,costs,buffers,timing) select * from tbl_cost_align;pid:5727
- value |-- count
- 1024 | 0
- 2048 | 0
- 4096 | 153
- 8192 |@@ 86293
- 16384 |@ 1864
- 32768 | 116
- 65536 |@@@ 5918 -- 接近块设备readahead次数
- 131072 | 59
- 262144 | 7
- 524288 | 3
- 1048576 | 2
- 2097152 | 2
- 4194304 | 0
- 8388608 | 0

SKY mobi
新手机 新应用 新娱乐

explain代价因子校准

- 验证公式正确性
- digoal=# show seq_page_cost;
- seq_page_cost
- ---------------
- 1
- (1 row)
- digoal=# show cpu_tuple_cost;
- cpu_tuple_cost
- ----------------
- 0.01
- (1 row)
- 195393 = (shared read=)94393*1(seq_page_cost) + (rows=)10100000*0.01(cpu_tuple_cost)
- digoal=# select 94393+10100000*0.01;
- ?column?
- -----------
- 195393.00
- (1 row)

SKY mobi
新手机 新应用 新娱乐

explain代价因子校准

- 从stap中我们得到io的平均响应时间是14329纳秒(0.014329毫秒). 得到了seq_page_cost.
- 真实的执行时间是(3260.695 - 0.839). 套用到公式中, 求得cpu_tuple_cost :
- 3260.695 -0.839 = 94393*0.014329 + 10100000*cpu_tuple_cost
- cpu_tuple_cost = 0.00018884145574257426
- 重启数据库, 并刷系统缓存后, 调整这两个代价因子
- digoal=# set seq_page_cost=0.014329;
- SET
- digoal=# set cpu_tuple_cost=0.00018884145574257426;
- SET
- 得到的cost和实际执行时间基本一致.
- digoal=# explain (analyze,verbose,costs,buffers,timing) select * from tbl_cost_align;
- QUERY PLAN
-
- ---
- Seq Scan on postgres.tbl_cost_align (cost=0.00..3259.86 rows=10100000 width=45) (actual time=0.915..3318.443 rows=10100000 loops=1)
- Output: id, info, crt_time
- Buffers: shared read=94393
- Total runtime: 4380.828 ms

SKY mobi
新手机 新应用 新娱乐

explain代价因子校准

- random_page_cost, cpu_index_tuple_cost, cpu_operator_cost的校准.
- random_page_cost 本文还是通过stap跟踪来获得.
- cpu_index_tuple_cost 和 cpu_operator_cost 两个未知数需要两个等式求得,
- 除了公式以外, 本文利用cpu_index_tuple_cost 和 cpu_operator_cost的比例得到第二个等式.
- 首先我们还是要确定公式准确性, 为了方便公式验证, 把所有的常量都设置为1.
- digoal=# set random_page_cost=1;
- SET
- digoal=# set cpu_tuple_cost=1;
- SET
- digoal=# set cpu_index_tuple_cost=1;
- SET
- digoal=# set cpu_operator_cost=1;
- SET

SKY mobi
新手机 新应用 新娱乐

explain代价因子校准

- digoal=# set enable_seqscan=off; set enable_bitmapscan=off; explain (analyze,verbose,costs,buffers,timing) select * from tbl_cost_align where id>1998999963;
- QUERY PLAN
-
- ----------
- Index Scan using idx_tbl_cost_align_id on postgres.tbl_cost_align (cost=174.00..20181.67 rows=5031 width=45) (actual time=0.029..17.773 rows=5037 loops=1)
- Output: id, info, crt_time
- Index Cond: (tbl_cost_align.id > 1998999963)
- Buffers: shared hit=5054
- Total runtime: 18.477 ms
- (5 rows)
- 执行计划表明这是个索引扫描, 至于扫了多少个数据块是未知的, 索引的tuples也是未知的, 已知的是cost和rows.
- 20181.67 = blocks*random_page_cost + cpu_tuple_cost*5031 + cpu_index_tuple_cost*5031 + cpu_operator_cost*?

SKY mobi
新手机 新应用 新娱乐

explain代价因子校准

- 求这个问号, 可以通过更改cpu_operator_cost来得到.
- digoal=# set cpu_operator_cost=2;
- SET
- digoal=# set enable_seqscan=off; set enable_bitmapscan=off; explain (analyze,verbose,costs,buffers,timing) select * from tbl_cost_align where id>1998999963;
- SET
- SET
- QUERY PLAN
-
- ---
- Index Scan using idx_tbl_cost_align_id on postgres.tbl_cost_align (cost=348.00..25386.67 rows=5031 width=45) (actual time=0.013..5.785 rows=5037 loops=1)
- Output: id, info, crt_time
- Index Cond: (tbl_cost_align.id > 1998999963)
- Buffers: shared hit=5054
- Total runtime: 6.336 ms
- (5 rows)
- 25386.67-20181.67 = 5205 得到本例通过索引扫描的条数. 等式就变成了
- 20181.67 = blocks*random_page_cost + cpu_tuple_cost*5031 + cpu_index_tuple_cost*5031 + cpu_operator_cost*5205

SKY mobi
新手机 新应用 新娱乐

explain代价因子校准

- 接下来要求blocks, 也就是扫描的随机页数.
- 通过调整random_page_cost得到.
- digoal=# set random_page_cost = 2;
- SET
- digoal=# set enable_seqscan=off; set enable_bitmapscan=off; explain (analyze,verbose,costs,buffers,timing) select * from tbl_cost_align where id>1998999963;
- SET
- SET
- QUERY PLAN
-
- ---
- Index Scan using idx_tbl_cost_align_id on postgres.tbl_cost_align (cost=348.00..30301.33 rows=5031 width=45) (actual time=0.013..5.778 rows=5037 loops=1)
- Output: id, info, crt_time
- Index Cond: (tbl_cost_align.id > 1998999963)
- Buffers: shared hit=5054
- Total runtime: 6.331 ms
- (5 rows)
- 30301.33-25386.67 = 4914.66 --得到blocks = 4914.66.

SKY mobi
新手机 新应用 新娱乐

explain代价因子校准

- 更新等式 ：
- 20181.67 = 4914.66*random_page_cost + cpu_tuple_cost*5031 + cpu_index_tuple_cost*5031 + cpu_operator_cost*5205

- 接下来要做的是通过stap统计出random_page_cost.
- pg93@db-172-16-3-150-> taskset -c 1 /home/pg93/pgsql9.3.1/bin/postgres >/dev/null 2>&1
- [root@db-172-16-3-150 ~]# sync; echo 3 > /proc/sys/vm/drop_caches
- digoal=# select pg_backend_pid();
- pg_backend_pid
- ----------------
- 10009
- (1 row)

SKY mobi
新手机 新应用 新娱乐

explain代价因子校准

```
[root@db-172-16-3-150 ~]# taskset -c 2 stap -e '
global a
probe process("/home/pg93/pgsql9.3.1/bin/postgres").mark("query__start") {
  delete a
  println("query__start ", user_string($arg1), "pid:", pid())
}
probe vfs.read.return {
  t = gettimeofday_ns() - @entry(gettimeofday_ns())
  # if (execname() == "postgres" && devname != "N/A")
    a[pid()] <<< t
}
probe process("/home/pg93/pgsql9.3.1/bin/postgres").mark("query__done") {
  if (@count(a[pid()]))
    printdln("**", pid(), @count(a[pid()]), @avg(a[pid()]))
  println("query__done ", user_string($arg1), "pid:", pid())
  if (@count(a[pid()])) {
# 未完
```

SKY mobi
新手机 新应用 新娱乐

explain代价因子校准

- println(@hist_log(a[pid()]))
- #println(@hist_linear(a[pid()],1024,4096,100))
- }
- delete a
- }' -x 10009

- 执行SQL :
- digoal=# set enable_seqscan=off; set enable_bitmapscan=off; explain (analyze,verbose,costs,buffers,timing) select * from tbl_cost_align where id>1998999963;
- QUERY PLAN
-
- ---
- Index Scan using idx_tbl_cost_align_id on postgres.tbl_cost_align (cost=0.43..5003.15 rows=5031 width=45) (actual time=0.609..1844.415 rows=5037 loops=1)
- Output: id, info, crt_time
- Index Cond: (tbl_cost_align.id > 1998999963)
- Buffers: shared hit=152 read=4902
- Total runtime: 1846.683 ms
- (5 rows)

explain代价因子校准

- query__start explain (analyze,verbose,costs,buffers,timing) select * from tbl_cost_align where id>1998999963;pid:10009
- 10009**4946**368362 -- 得到random_page_cost
- query__done explain (analyze,verbose,costs,buffers,timing) select * from tbl_cost_align where id>1998999963;pid:10009
- value |--- count
- 2048 | 0
- 4096 | 0
- 8192 | 33
- 16384 | 2
- 32768 | 6
- 65536 | 4
- 131072 |@@@@@@@@@@@@@@@@@@@ 1193
- 262144 |@@ 2971
- 524288 |@@@@@@@@@@@@ 729
- 1048576 | 2
- 2097152 | 5
- 4194304 | 0
- 8388608 | 1
- 16777216 | 0
- 33554432 | 0

SKY mobi
新手机 新应用 新娱乐

explain代价因子校准

- 更新等式, 使用时间等式 :
- 等式1 :
- 1844.415 = 4914.66*0.368362 + 0.0001888414557425742 6*5031 + cpu_index_tuple_cost*5031 + cpu_operator_cost*5205
- cpu_tuple_cost用例子1中计算得到的0.0001888414557425742 6
- cpu_index_tuple_cost和cpu_operator_cost的比例用系统默认的2 : 1.
- 等式2 :
- cpu_index_tuple_cost/cpu_operator_cost = 2
- 最终得到 :
- cpu_index_tuple_cost = 0.0043349708521647999 0
- cpu_operator_cost = 0.0021674854260823999 5

SKY mobi
新手机 新应用 新娱乐

explain代价因子校准

- 结合例子1 得到的两个常量, 所有的5个常量值就调整好了.
- digoal=# set cpu_tuple_cost=0.00018884145574257426;
- SET
- digoal=# set cpu_index_tuple_cost = 0.00433497085216479990;
- SET
- digoal=# set cpu_operator_cost = 0.00216748542608239995;
- SET
- digoal=# set seq_page_cost=0.014329;
- SET
- digoal=# set random_page_cost = 0.368362;
- SET

- 校准代价因子练习

SKY mobi
新手机 新应用 新娱乐

auto_explain插件的使用

- http://blog.163.com/digoal@126/blog/static/163877040201158256121 45/
- auto_explain的目的是给数据库中执行的SQL语句一个执行时间阈值, 超过阈值的话, 记录下当时这个SQL的执行计划到日志中, 便于未来查看这个SQL执行计划有没有问题.
- 编译安装
- [root@db-172-16-3-150 ~]# export PAHT=/home/pg93/pgsql/bin:$PATH
- [root@db-172-16-3-150 ~]# which pg_config
- /usr/bin/pg_config
- [root@db-172-16-3-150 ~]# cd /opt/soft_bak/postgresql-9.3.1/contrib/auto_explain/
- [root@db-172-16-3-150 auto_explain]# gmake clean
- [root@db-172-16-3-150 auto_explain]# gmake
- [root@db-172-16-3-150 auto_explain]# gmake install

- auto_explain 有两种使用方法
- 会话级使用
- 数据库级使用

SKY mobi
新手机 新应用 新娱乐

auto_explain插件的使用

- 会话级使用举例
- digoal=# load 'auto_explain';
- LOAD
- digoal=# set auto_explain.log_min_duration=0; 设置SQL执行时间阈值
- SET
- digoal=# select * from t limit 1;
- 查看日志
- 2013-12-10 14:32:15.587 CST,"postgres","digoal",12933,"[local]",52a6b506.3285,11,"SELECT",2013-12-10 14:30:30 CST,2/180059,0,LOG,00000,"duration: 0.043 ms plan:
- Query Text: select * from t limit 1;
- Limit (cost=0.00..0.03 rows=1 width=108)
- -> Seq Scan on t (cost=0.00..1409091.04 rows=50000004 width=108)",,,,,,,,"explain_ExecutorEnd, auto_explain.c:320","psql"

SKY mobi
新手机 新应用 新娱乐

auto_explain插件的使用

- 数据库级使用
- vi $PGDATA/postgresql.conf
- shared_preload_libraries = 'pg_stat_statements, auto_explain'
- auto_explain.log_min_duration = 100ms
- 修改shared_preload_libraries需要重启数据库
- pg93@db-172-16-3-150-> pg_ctl restart -m fast

SKY mobi
新手机 新应用 新娱乐

练习

- 使用auto_explain跟踪慢SQL的执行计划
- explain代价因子的校准

SKY mobi
新手机 新应用 新娱乐

连接池及数据库高速缓存

- 连接池及数据库高速缓存
- 目标:
  - 以pgbouncer为例, 理解数据库连接池在短连接环境下的好处, 连接池的几种模式和使用场景
  - 几种外部高速缓存的介绍, 如os 层缓存pgfincore, K-V缓存pgmemcached的使用.

SKY mobi
新手机 新应用 新娱乐

连接池

- 为什么要使用连接池?
- 理由一, 由于PostgreSQL是进程模式, 短连接会带来性能问题. 看几个测试结果 :
- pg93@db-172-16-3-150-> vi test.sql
- select 1;
- 短连接模式的tps.
- pg93@db-172-16-3-150-> pgbench -M extended -n -r -f ./test.sql -c 16 -j 4 -C -T 30
- transaction type: Custom query
- scaling factor: 1
- query mode: extended
- number of clients: 16
- number of threads: 4
- duration: 30 s
- number of transactions actually processed: 36100
- tps = 1203.128160 (including connections establishing)
- tps = 97264.142873 (excluding connections establishing)
- statement latencies in milliseconds:
- 9.993634 select 1;

SKY mobi
新手机 新应用 新娱乐

连接池

- 长连接模式的tps
- pg93@db-172-16-3-150-> pgbench -M extended -n -r -f ./test.sql -c 16 -j 4 -T 30
- transaction type: Custom query
- scaling factor: 1
- query mode: extended
- number of clients: 16
- number of threads: 4
- duration: 30 s
- number of transactions actually processed: 2571870
- tps = 85724.228018 (including connections establishing)
- tps = 85767.412365 (excluding connections establishing)
- statement latencies in milliseconds:
- 0.185190 select 1;

SKY mobi
新手机 新应用 新娱乐

连接池

- 使用prepared模式的tps
- pg93@db-172-16-3-150-> pgbench -M prepared -n -r -f ./test.sql -c 16 -j 4 -T 30
- transaction type: Custom query
- scaling factor: 1
- query mode: prepared
- number of clients: 16
- number of threads: 4
- duration: 30 s
- number of transactions actually processed: 3695465
- tps = 123176.163882 (including connections establishing)
- tps = 123233.120481 (excluding connections establishing)
- statement latencies in milliseconds:
- 0.128769 select 1;

SKY mobi
新手机 新应用 新娱乐

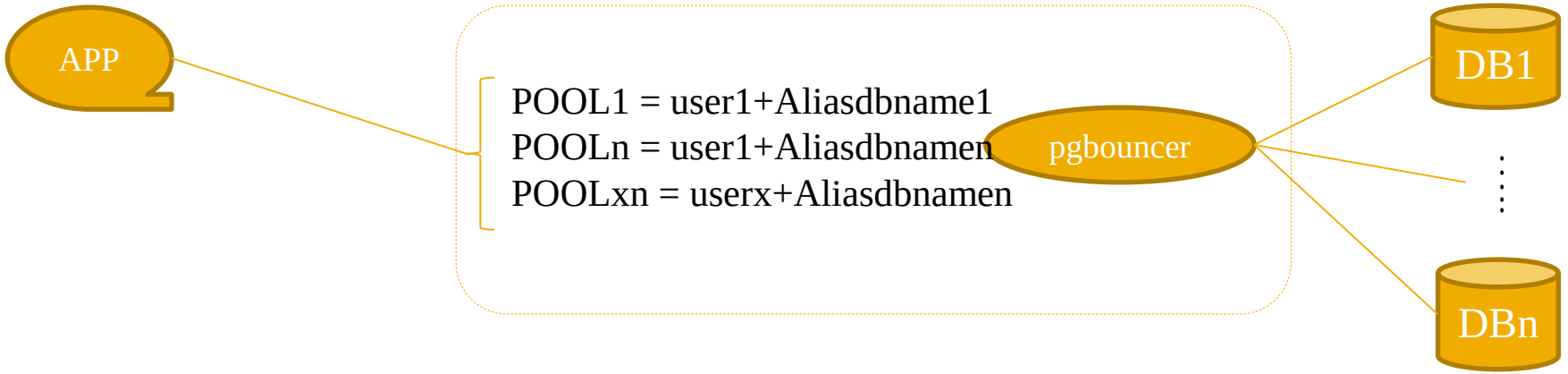
连接池

- 理由二, 当客户端非常多时, 大多数连接可能空闲, 但是长时间占据一个连接, 可能导致连接数超出数据库最大连接数配置, 正常发起的请求无法获得连接.
- 理由三, 连接池可以挡掉一些非法请求, 例如非法的访问非业务数据库的请求.
- 理由四, 连接池位于数据库和应用程序之间, 比较容易实现负载均衡的功能, 对应用程序透明.

SKY mobi
新手机 新应用 新娱乐

连接池

- pgbouncer连接池介绍
- 源码地址
- http://git.postgresql.org/gitweb/?p=pgbouncer.git;a=summary
- pgbouncer是一个比较小巧的PostgreSQL连接池插件, 采用线程模式, 每个连接仅需2K内存, 非常适合短连接的场景.
- pgbouncer支持三种连接复用模式:
  - 会话模式, 当客户端与pgbouncer会话断开时, 服务端的连接才可以被复用.
  - 事务模式, 当客户端与事务提交后, 服务端的连接才可以被复用.
  - 语句模式, 当客户端语句执行完后, 服务端的连接才可以被复用.
- 使用注意 :
  - 对于使用了绑定变量的客户端, 请使用会话模式, 因为会话中需要保存并复用named prepared statement的信息.



SKY mobi
新手机 新应用 新娱乐

连接池

- 安装pgbouncer
- 需求
- GNU Make 3.81+
- libevent 2.x
- http://monkey.org/~provos/libevent/
- 可选
- 异步DNS请求库, c-areq

- 安装libevent 2.x
- wget https://github.com/downloads/libevent/libevent/libevent-2.0.21-stable.tar.gz
- tar -zxvf libevent-2.0.21-stable.tar.gz
- cd libevent-2.0.21-stable
- ./configure && make && make install
- echo "/usr/local/lib" >> /etc/ld.so.conf

SKY mobi
新手机 新应用 新娱乐

连接池

- 配置lib库路径
- [root@db-172-16-3-150 libevent-2.0.21-stable]# ldconfig
- [root@db-172-16-3-150 libevent-2.0.21-stable]# ldconfig -p|grep libevent
- libevent_pthreads-2.0.so.5 (libc6,x86-64) => /usr/local/lib/libevent_pthreads-2.0.so.5
- libevent_openssl-2.0.so.5 (libc6,x86-64) => /usr/local/lib/libevent_openssl-2.0.so.5
- libevent_extra-2.0.so.5 (libc6,x86-64) => /usr/local/lib/libevent_extra-2.0.so.5
- libevent_extra-1.4.so.2 (libc6,x86-64) => /usr/lib64/libevent_extra-1.4.so.2
- libevent_core-2.0.so.5 (libc6,x86-64) => /usr/local/lib/libevent_core-2.0.so.5
- libevent_core-1.4.so.2 (libc6,x86-64) => /usr/lib64/libevent_core-1.4.so.2
- libevent-2.0.so.5 (libc6,x86-64) => /usr/local/lib/libevent-2.0.so.5
- libevent-1.4.so.2 (libc6,x86-64) => /usr/lib64/libevent-1.4.so.2

SKY mobi
新手机 新应用 新娱乐

连接池

- 安装异步DNS请求LIB库
- wget http://c-ares.haxx.se/download/c-ares-1.10.0.tar.gz
- tar -zxvf c-ares-1.10.0.tar.gz
- cd c-ares-1.10.0
- ./configure && gmake && gmake install
- [root@db-172-16-3-150 c-ares-1.10.0]# ldconfig
- [root@db-172-16-3-150 c-ares-1.10.0]# ldconfig -p|grep ares
- libcares.so.2 (libc6,x86-64) => /usr/local/lib/libcares.so.2
- libcares.so.2 (libc6,x86-64) => /usr/lib64/libcares.so.2
- libcares.so (libc6,x86-64) => /usr/local/lib/libcares.so

SKY mobi
新手机 新应用 新娱乐

连接池

- 安装pgbouncer
- git clone git://git.postgresql.org/git/pgbouncer.git
- cd pgbouncer
- git submodule init
- git submodule update
- ./autogen.sh
- ./configure --prefix=/opt/pgbouncer --with-libevent=/usr/local/lib --with-cares=/usr/local/lib
- make
- make install
- 命令行参数
- [root@db-172-16-3-150 opt]# /opt/pgbouncer/bin/pgbouncer --help
- Usage: pgbouncer [OPTION]... config.ini
- -d, --daemon Run in background (as a daemon)
- -R, --restart Do a online restart
- -q, --quiet Run quietly
- -v, --verbose Increase verbosity
- -u, --user=<username> Assume identity of <username>
- -V, --version Show version
- -h, --help Show this help screen and exit

SKY mobi
新手机 新应用 新娱乐

连接池

- 配置pgbouncer
- 1. 主配置文件
- mkdir -p /opt/pgbouncer/etc
- cd /opt/pgbouncer/etc/
- vi config.ini
- [databases]
- aliasdb1 = host=172.16.3.150 port=1921 dbname=digoal client_encoding=sql_ascii datestyle=ISO pool_size=20

- [pgbouncer]
- pool_mode = transaction
- listen_port = 6543
- listen_addr = 0.0.0.0
- auth_type = md5
- auth_file = /opt/pgbouncer/etc/users.txt
- logfile = /opt/pgbouncer/etc/pgbouncer.log
- 未完

SKY mobi
新手机 新应用 新娱乐

连接池

- pidfile = /opt/pgbouncer/etc/pgbouncer.pid
- unix_socket_dir = /opt/pgbouncer/etc
- admin_users = pgadmin
- stats_users = pgmon
- server_reset_query = DISCARD ALL
- server_check_query = select 1
- server_check_delay = 30
- max_client_conn = 50000
- default_pool_size = 20
- reserve_pool_size = 5
- dns_max_ttl = 15

SKY mobi
新手机 新应用 新娱乐

连接池

- 2. 用户密码配置文件
- cd /opt/pgbouncer/etc/
- vi users.txt
- "postgres" "md53175bce1d3201d16594cebf9d7eb3f9d"
- "pgadmin" "md55bde83786c10fc0f383464f6e56a6d6e"
- "pgmon" "123abc"

- MD5密码封装规则md5(密码+用户名), 与PostgreSQL中存储的md5一致.
- 例如pgadmin的密码123abc, 封装成md5为: md55bde83786c10fc0f383464f6e56a6d6e
- digoal=# select md5('123abcpgadmin');
- md5
- ----------
- 5bde83786c10fc0f383464f6e56a6d6e
- (1 row)

SKY mobi
新手机 新应用 新娱乐

连接池

- digoal=# alter role postgres encrypted password 'hello';
- ALTER ROLE
- digoal=# select md5('hellopostgres');
- md5
- ----------
- 0cfe8da07c510ab414c7da9b1acc8fbd
- (1 row)

- digoal=# select passwd from pg_shadow where usename='postgres';
- passwd
- ----------
- md50cfe8da07c510ab414c7da9b1acc8fbd
- (1 row)

SKY mobi
新手机 新应用 新娱乐

连接池

- 3. 启动
- chown -R pg93:pg93 /opt/pgbouncer
- /opt/pgbouncer/bin/pgbouncer -d -u pg93 /opt/pgbouncer/etc/config.ini

- 4. 日志
- 2013-12-10 17:22:15.570 20185 LOG listening on 0.0.0.0:6543
- 2013-12-10 17:22:15.570 20185 LOG listening on unix:/tmp/.s.PGSQL.6543
- 2013-12-10 17:22:15.570 20185 LOG process up: pgbouncer 1.6dev, libevent 2.0.21-stable (epoll), adns: c-ares 1.10.0

SKY mobi
新手机 新应用 新娱乐

连接池

- pgbouncer命令行管理
- pg93@db-172-16-3-150-> psql -h 127.0.0.1 -p 6543 -U pgadmin pgbouncer
- Password for user pgadmin:
- 这里输入pgadmin用户的密码.
- psql (9.3.1, server 1.6dev/bouncer)
- Type "help" for help.

- pgbouncer=# show help;
- NOTICE: Console usage
- DETAIL:
- SHOW HELP|CONFIG|DATABASES|POOLS|CLIENTS|SERVERS|VERSION
- SHOW STATS|FDS|SOCKETS|ACTIVE_SOCKETS|LISTS|MEM
- SHOW DNS_HOSTS|DNS_ZONES
- SET key = arg
- 未完

SKY mobi
新手机 新应用 新娱乐

连接池

- RELOAD
- PAUSE [<db>]
- RESUME [<db>]
- DISABLE <db>
- ENABLE <db>
- KILL <db>
- SUSPEND
- SHUTDOWN
- SHOW

- pgbouncer=# show config; -- 列出当前所有的配置
- ...
- 其他

SKY mobi
新手机 新应用 新娱乐

连接池

- 短连接的测试结果
- pg93@db-172-16-3-150-> pgbench -M extended -n -r -f ./test.sql -h /tmp -p 6543 -U postgres -c 16 -j 4 -C -T 30 aliasdb1
- transaction type: Custom query
- scaling factor: 1
- query mode: extended
- number of clients: 16
- number of threads: 4
- duration: 30 s
- number of transactions actually processed: 213378
- tps = 7110.818977 (including connections establishing)
- tps = 181282.711671 (excluding connections establishing)
- statement latencies in milliseconds:
- 1.698879 select 1;

SKY mobi
新手机 新应用 新娱乐

连接池

- pgbouncer各参数介绍
- 查看doc/config.txt

SKY mobi
新手机 新应用 新娱乐

数据库高速缓存

- 本地高速缓存pgfincore
  - OS Cache
  - http://git.postgresql.org/gitweb/?p=pgfincore.git;a=summary
  - 利用posix_fadvise修改文件的advice值. (参见 **pgfincore.c**　　posix_fadvise(fd, 0, 0, adviceFlag);)
  - man posix_fadvise

- 异地高速缓存pgmemcache
  - memcached

- 除了pgmemcache以外, pgredis也是类似的项目, 只不过是redis的一些封装好的API.
- 　参考: https://github.com/siavashg/pgredis

SKY mobi
新手机 新应用 新娱乐

本地高速缓存pgfincore

- 安装
- tar -zxvf pgfincore-b371336.tar.gz
- mv pgfincore-b371336 postgresql-9.3.1/contrib/
- cd postgresql-9.3.1/contrib/pgfincore-b371336/
- export PATH=/home/pg93/pgsql/bin:$PATH
- which pg_config
- /home/pg93/pgsql/bin/pg_config
- gmake clean
- gmake
- gmake install

- [root@db-172-16-3-150 pgfincore-b371336]# su - pg93
- pg93@db-172-16-3-150-> psql
- Type "help" for help.
- digoal=# create extension pgfincore;
- CREATE EXTENSION

SKY mobi
新手机 新应用 新娱乐

本地高速缓存pgfincore

- 测试
- digoal=# create table user_info(id int primary key, info text, crt_time timestamp);
- CREATE TABLE
- digoal=# insert into user_info select generate_series(1,5000000), md5(random()::text), clock_timestamp();
- 改参数, 便于观察本地OS缓存, 重启数据库.
- shared_buffers = 32MB
- echo 3 > /proc/sys/vm/drop_caches

- pgbench测试脚本
- vi test.sql
- \setrandom id 1 5000000
- select * from user_info where id=:id;

SKY mobi
新手机 新应用 新娱乐

本地高速缓存pgfincore

- 加载本地缓存前的测试结果
- pg93@db-172-16-3-150-> pgbench -M prepared -n -r -f ./test.sql -c 16 -j 4 -T 10 digoal
- transaction type: Custom query
- scaling factor: 1
- query mode: prepared
- number of clients: 16
- number of threads: 4
- duration: 10 s
- number of transactions actually processed: 27743
- tps = 2760.238242 (including connections establishing)
- tps = 2764.017578 (excluding connections establishing)
- statement latencies in milliseconds:
- 0.002850 \setrandom id 1 5000000
- 5.772589 select * from user_info where id=:id;

SKY mobi
新手机 新应用 新娱乐

本地高速缓存pgfincore

- 加载本地缓存
- digoal=# select pgfadvise_willneed('user_info');
- pgfadvise_willneed
- ----------
- (pg_tblspc/66422/PG_9.3_201306121/16384/92762,4096,93458,24384631)
- (1 row)

- digoal=# select pgfadvise_willneed('user_info_pkey');
- pgfadvise_willneed
- ----------
- (pg_tblspc/66422/PG_9.3_201306121/16384/92768,4096,27424,24355374)
- (1 row)
- 如果涉及TOAST表的查询, 还需要将toast表加载到缓存中.

SKY mobi
新手机 新应用 新娱乐

本地高速缓存pgfincore

- 加载本地缓存后的测试结果, 性能提升是非常明显的.
- pg93@db-172-16-3-150-> pgbench -M prepared -n -r -f ./test.sql -c 16 -j 4 -T 10 digoal
- transaction type: Custom query
- scaling factor: 1
- query mode: prepared
- number of clients: 16
- number of threads: 4
- duration: 10 s
- number of transactions actually processed: 578719
- tps = 57846.754016 (including connections establishing)
- tps = 57925.210019 (excluding connections establishing)
- statement latencies in milliseconds:
- 0.002523 \setrandom id 1 5000000
- 0.269790 select * from user_info where id=:id;

SKY mobi
新手机 新应用 新娱乐

异地高速缓存pgmemcache

- 异地高速缓存pgmemcache
- http://blog.163.com/digoal@126/blog/static/16387704020121017234125 7/

- pgmemcache是一系列的PostgreSQL函数, 用于memcache的读写操作.
- 通过pgmemcache以及PostgreSQL的触发器可以方便的对数据库中的数据进行缓存.
- 当然缓存的操作也可以挪至应用程序自己来处理. pgmemcache只是一种选择.

SKY mobi
新手机 新应用 新娱乐

异地高速缓存pgmemcache

- pgmemcache的安装

- pgmemcache 依赖 libmemcache 和 PostgreSQL
- libmemcache 依赖 libevent 和 memcached
- memcached 依赖 libevent

- 依次安装

SKY mobi
新手机 新应用 新娱乐

异地高速缓存pgmemcache

- 安装libevent
- wget https://github.com/downloads/libevent/libevent/libevent-2.0.20-stable.tar.gz
- tar -zxvf libevent-2.0.20-stable.tar.gz
- cd libevent-2.0.20-stable
- ./configure
- make
- make install
- 加入lib库路径
- vi /etc/ld.so.conf
- /usr/local/lib
- ldconfig
- ldconfig -p|grep libevent
- libevent_pthreads-2.0.so.5 (libc6,x86-64) => /usr/local/lib/libevent_pthreads-2.0.so.5
- libevent_openssl-2.0.so.5 (libc6,x86-64) => /usr/local/lib/libevent_openssl-2.0.so.5
- libevent_extra-2.0.so.5 (libc6,x86-64) => /usr/local/lib/libevent_extra-2.0.so.5
- libevent_core-2.0.so.5 (libc6,x86-64) => /usr/local/lib/libevent_core-2.0.so.5
- libevent-2.0.so.5 (libc6,x86-64) => /usr/local/lib/libevent-2.0.so.5

SKY mobi
新手机 新应用 新娱乐

异地高速缓存pgmemcache

- 安装memcached
- wget http://memcached.googlecode.com/files/memcached-1.4.15.tar.gz
- tar -zxvf memcached-1.4.15.tar.gz
- cd memcached-1.4.15
- ./configure --help
- ./configure --prefix=/opt/memcached-1.4.15 --enable-sasl --enable-64bit
- make
- make install
- cd /opt/memcached-1.4.15/share/man/man1
- man ./memcached.1

- 启动memcached
- memcached -d -u pg93 -m 800

- 小提示
- 64bit 对应pointer_size : 64, 所以将占用更多的空间. 如果没有超过20亿的key, 使用32位就够了.

SKY mobi
新手机 新应用 新娱乐

异地高速缓存pgmemcache

- 安装libmemcached
- wget http://download.tangent.org/libmemcached-0.48.tar.gz
- tar -zxvf libmemcached-0.48.tar.gz
- cd libmemcached-0.48
- ./configure --prefix=/opt/libmemcached-0.48 --with-memcached=/opt/memcached-1.4.15/bin/memcached
- make
- make install
- 修改动态库配置文件, 并使之生效 :
- vi /etc/ld.so.conf
- /opt/libmemcached-0.48/lib
- ldconfig
- 查看新增的动态库是否生效 :
- ldconfig -p|grep libmemcache
- libmemcachedutil.so.1 (libc6,x86-64) => /opt/libmemcached-0.48/lib/libmemcachedutil.so.1
- libmemcachedutil.so (libc6,x86-64) => /opt/libmemcached-0.48/lib/libmemcachedutil.so
- libmemcachedprotocol.so.0 (libc6,x86-64) => /opt/libmemcached-0.48/lib/libmemcachedprotocol.so.0
- libmemcachedprotocol.so (libc6,x86-64) => /opt/libmemcached-0.48/lib/libmemcachedprotocol.so
- libmemcached.so.6 (libc6,x86-64) => /opt/libmemcached-0.48/lib/libmemcached.so.6
- libmemcached.so (libc6,x86-64) => /opt/libmemcached-0.48/lib/libmemcached.so

SKY mobi
新手机 新应用 新娱乐

异地高速缓存pgmemcache

- 安装pgmemcache
- wget http://pgfoundry.org/frs/download.php/3018/pgmemcache_2.0.6.tar.bz2 -- 或者 https://github.com/ohmu/pgmemcache/
- tar -jxvf pgmemcache_2.0.6.tar.bz2
- cd pgmemcache
- 需要用到pg_config, 所以需要加入到PATH中.
- . /home/pg9.2.0/.bash_profile

- pgmemcache的头文件中包含了libmemcached的一些头, 如下, 所以需要将这些头文件拷贝到pgmemcache的目录中来.
- less pgmemcache.h
- #include <libmemcached/sasl.h>
- #include <libmemcached/memcached.h>
- #include <libmemcached/server.h>
- #include <sasl/sasl.h>

- 拷贝这些头文件到本地目录中,
- cp -r /opt/libmemcached-0.48/include/libhashkit ./
- cp -r /opt/libmemcached-0.48/include/libmemcached ./

SKY mobi
新手机 新应用 新娱乐

异地高速缓存pgmemcache

- 同时编译时需要用到libmemcached.so, 如下Makefile :
- less Makefile
- SHLIB_LINK = -lmemcached -lsasl2

- 但是没有指定库目录, 所以需要修改一下
- vi Makefile
- SHLIB_LINK = -L/opt/libmemcached-0.48/lib -lmemcached -lsasl2

- 接下来编译安装就可以了.
- gmake
- gmake install

SKY mobi
新手机 新应用 新娱乐

异地高速缓存pgmemcache

- 安装好pgmemcache后, 需要修改PostgreSQL的配置文件重启数据库,

- 这里假设172.16.3.150上已经启动了memcached.
- su - pg93
- cd $PGDATA
- vi postgresql.conf
- shared_preload_libraries = 'pgmemcache'
- pgmemcache.default_servers = '172.16.3.150:11211' #多个memcached用逗号隔开配置.
- pgmemcache.default_behavior = 'BINARY_PROTOCOL:1' #多个配置用逗号隔开配置.

- 重启数据库 :
- pg_ctl restart -m fast

- 在加载pgmemcache.sql前, 需要对这个脚本修改一下, 否则会报语法错误.
- cd $PGHOME/share/contrib
- vi pgmemcache.sql
- :%s/LANGUAGE\ 'C'/LANGUAGE\ C/g
- :x!

SKY mobi
新手机 新应用 新娱乐

异地高速缓存pgmemcache

- 在需要的库中执行脚本 :
- psql -h 127.0.0.1 -U postgres digoal -f ./pgmemcache.sql

- 测试 :
- digoal=> select memcache_set('key1', '1');
- memcache_set
- --------------
- t
- (1 row)

- digoal=> select memcache_get('key1');
- memcache_get
- --------------
- 1
- (1 row)

SKY mobi
新手机 新应用 新娱乐

异地高速缓存pgmemcache

- digoal=> select memcache_incr('key1',99);
- memcache_incr
- ---------------
- 100
- (1 row)

- digoal=> select memcache_incr('key1',99);
- memcache_incr
- ---------------
- 199
- (1 row)

SKY mobi
新手机 新应用 新娱乐

异地高速缓存pgmemcache

- digoal=> select memcache_stats();
- memcache_stats
- ------------------------------
- +
- Server: 172.16.3.150 (11211)+
- pid: 1918 +
- uptime: 13140 +
- time: 1353222576 +
- version: 1.4.15 +
- pointer_size: 64 +
- rusage_user: 0.999 +
- rusage_system: 0.1999 +
- curr_items: 1 +
- total_items: 3 +
- bytes: 72 +
- curr_connections: 6 +
- total_connections: 10 +
- connection_structures: 7 +

SKY mobi
新手机 新应用 新娱乐

异地高速缓存pgmemcache

- cmd_get: 1 +
- cmd_set: 1 +
- get_hits: 1 +
- get_misses: 0 +
- evictions: 0 +
- bytes_read: 207 +
- bytes_written: 3196 +
- limit_maxbytes: 67108864 +
- threads: 4 +

SKY mobi
新手机 新应用 新娱乐

异地高速缓存pgmemcache

- cmd_get: 1 +
- cmd_set: 1 +
- get_hits: 1 +
- get_misses: 0 +
- evictions: 0 +
- bytes_read: 207 +
- bytes_written: 3196 +
- limit_maxbytes: 67108864 +
- threads: 4 +

SKY mobi
新手机 新应用 新娱乐

异地高速缓存pgmemcache

- digoal=> select memcache_flush_all();
- memcache_flush_all
- ---
- t

- digoal=> select memcache_get('key1');
- memcache_get
- ---
-
- (1 row)

SKY mobi
新手机 新应用 新娱乐

异地高速缓存pgmemcache

- cache应用场景举例, 将用户-密码作为K-V存储到MEMCACHED中, 密码校验先从memcached进行匹配, 未匹配到再到数据库中检索.
- 1. 测试表
- digoal=> create table tbl_user_info (userid int8 primary key, pwd text);
- NOTICE: CREATE TABLE / PRIMARY KEY will create implicit index "tbl_user_info_pkey" for table "tbl_user_info"
- CREATE TABLE
- 2. 测试数据
- digoal=> insert into tbl_user_info select generate_series(1,10000000), md5(clock_timestamp()::text);
- INSERT 0 10000000
- 3. 更新触发器(不安全, SQL回滚后memcache的操作不能自动回滚)
- CREATE OR REPLACE FUNCTION tbl_user_info_upd() RETURNS TRIGGER AS $$
- BEGIN
- IF OLD.pwd != NEW.pwd THEN
- PERFORM memcache_set('tbl_user_info_' || NEW.userid || '_pwd', NEW.pwd);
- END IF;
- RETURN NULL;
- END;
- $$ LANGUAGE 'plpgsql' STRICT;
- CREATE TRIGGER tbl_user_info_upd AFTER UPDATE ON tbl_user_info FOR EACH ROW EXECUTE PROCEDURE tbl_user_info_upd();

SKY mobi
新手机 新应用 新娱乐

异地高速缓存pgmemcache

- 4. 插入触发器(不安全, 同理)
- CREATE OR REPLACE FUNCTION tbl_user_info_ins() RETURNS TRIGGER AS $$
- BEGIN
- PERFORM memcache_set('tbl_user_info_' || NEW.userid || '_pwd', NEW.pwd);
- RETURN NULL;
- END;
- $$ LANGUAGE 'plpgsql' STRICT;
- CREATE TRIGGER tbl_user_info_ins AFTER INSERT ON tbl_user_info FOR EACH ROW EXECUTE PROCEDURE tbl_user_info_ins();
- 5. 删除触发器(安全, 因为无法命中cache是安全的, 但是cache数据和table数据不一致是不安全的)
- CREATE OR REPLACE FUNCTION tbl_user_info_del() RETURNS TRIGGER AS $$
- BEGIN
- PERFORM memcache_delete('tbl_user_info_' || NEW.userid || '_pwd');
- RETURN NULL;
- END;
- $$ LANGUAGE 'plpgsql' STRICT;
- CREATE TRIGGER tbl_user_info_del AFTER DELETE ON tbl_user_info FOR EACH ROW EXECUTE PROCEDURE tbl_user_info_del();

SKY mobi
新手机 新应用 新娱乐

异地高速缓存pgmemcache

- 6. 用户密码校验函数 :
- CREATE OR REPLACE FUNCTION auth (i_userid int8, i_pwd text) returns boolean as $$
- declare
- v_input_pwd_md5 text;
- v_user_pwd_md5 text;
- begin
- v_input_pwd_md5 := md5(i_pwd);
- select memcache_get('tbl_user_info_' || i_userid || '_pwd') into v_user_pwd_md5;
- if (v_user_pwd_md5 <> '') then
- raise notice 'hit in memcache.';
- if (v_input_pwd_md5 = v_user_pwd_md5) then
- return true;
- else
- return false;
- end if;
- else
- -- 未完

SKY mobi
新手机 新应用 新娱乐

异地高速缓存pgmemcache

```
select pwd into v_user_pwd_md5 from tbl_user_info where userid=i_userid;
  if found then
    raise notice 'hit in table.';
    if (v_input_pwd_md5 = v_user_pwd_md5) then
      return true;
    else
      return false;
    end if;
  else
    return false;
  end if;
 end if;
exception
when others then
  return false;
end;
$$ language plpgsql strict;
```

SKY mobi
新手机 新应用 新娱乐

练习

- pgbouncer连接池搭建, 几种模式的使用对比
- pgbench压力测试, 测试短连接
- 本地高速缓存pgfincore的使用, 测试它带来的性能提升
- 异地高速缓存pgmemcached的使用

SKY mobi
新手机 新应用 新娱乐

数据库扩展及复制

- 数据库扩展及复制
- 目标:
  - 了解
  - 数据库热备份与还原,
  - 数据库集群级流复制,
  - 表级复制,
  - 数据库在虚拟化环境下的使用注意事项

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 数据库热备份与还原

- 数据库热备份与还原
- 物理备份与还原
  - 备份$PGDATA,归档文件,以及所有的表空间目录. 适用于跨小版本的恢复, 但是不能跨平台.
  - 需开启归档
  - 目前PG还不支持基于数据文件数据块变更的增量备份, 仅仅支持数据文件+归档的备份方式
  - 目前PG官方还不支持基于表空间的备份和还原. 但是可模拟.
    - http://blog.163.com/digoal@126/blog/static/163877040201232614225 81/
- 逻辑备份与还原
  - 备份数据, 适用于跨版本和跨平台的恢复

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 数据库物理备份

- 首先要开启归档, 日志模式>=archive, 步骤如下
- 创建归档目录
  - mkdir -p /ssd4/pg93/arch
  - chown -R pg93:pg93 /ssd4/pg93/arch
- 配置归档命令
  - %p 表示xlog文件名$PGDATA的相对路径, 如pg_xlog/000000010000000190000007D
  - %f 表示xlog文件名, 如000000010000000190000007D
  - vi $PGDATA/postgresql.conf
  - archive_mode = on
  - archive_command = 'DATE=`date +%Y%m%d`; DIR="/ssd4/pg93/arch/$DATE"; (test -d $DIR || mkdir -p $DIR) && cp %p $DIR/%f'
- 配置日志模式vi $PGDATA/postgresql.conf; (wal_level = hot_standby)
- 重启数据库(可选, 如果以前已经开启了归档的话则不需要重启数据库)
- 测试归档是否正常
  - digoal=# checkpoint;
  - digoal=# select pg_switch_xlog();
  - pg93@db-172-16-3-150-> cd /ssd4/pg93/arch/20131211/
  - pg93@db-172-16-3-150-> ll
  - -rw------- 1 pg93 pg93 16M Dec 11 09:28 000000010000000180000001F

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 数据库物理备份

- 物理备份, 方式很多, 达到目的即可.
- 通过pg_basebackup, 流复制协议备份, (本地使用时必须用tar模式. 异地无所谓, 如果要同目录结构的话使用p模式)
- 创建replication权限的角色, 或者超级用户的角色.
  - digoal=# create role rep nosuperuser replication login connection limit 32 encrypted password 'rep123';
  - CREATE ROLE
- 配置pg_hba.conf
  - host replication rep 0.0.0.0/0 md5
  - pg_ctl reload
- 备份, 因为使用流复制协议, 所以支持异地备份.
  - mkdir `date +%F` ; pg_basebackup -F t -x -D ./`date +%F` -h 172.16.3.150 -p 1921 -U rep
- 备份目录如下
- pg93@db-172-16-3-150-> ll
- total 13G
- -rw-rw-r-- 1 pg93 pg93 955M Dec 11 09:45 66372.tar
- -rw-rw-r-- 1 pg93 pg93 12G Dec 11 09:45 66422.tar
- -rw-rw-r-- 1 pg93 pg93 160M Dec 11 09:46 base.tar

- 或者手工拷贝目录的方式备份.

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 数据库物理备份

- 数字目录代表表空间的备份包
- pg93@db-172-16-3-150-> cd $PGDATA
- pg93@db-172-16-3-150-> cd pg_tblspc/
- pg93@db-172-16-3-150-> ll
- total 0
- lrwxrwxrwx 1 pg93 pg93 18 Oct 27 07:34 66372 -> /ssd4/pg93/tbs_idx
- lrwxrwxrwx 1 pg93 pg93 21 Oct 28 09:16 66422 -> /ssd3/pg93/tbs_digoal

- base目录代表$PGDATA的备份包, 查看tar包内容.
- pg93@db-172-16-3-150-> tar -tvf base.tar |less

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 数据库物理备份

- 手工拷贝目录的方式备份.
- 首先要打开强制检查点
  - pg93@db-172-16-3-150-> psql
  - psql (9.3.1)
  - Type "help" for help.
  - digoal=# select pg_start_backup(now()::text);
  - pg_start_backup
  - ----------
  - 18/27000028
  - (1 row)
- 然后备份$PGDATA和表空间目录, 例如拷贝到网络存储.

- 拷贝完后后, 关闭强制检查点
  - digoal=# select pg_stop_backup();
- 最后拷贝强制检查点之间的所有归档文件, 确保备份有效性.

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 数据库物理还原

- 物理还原, 顺序读取XLOG的信息进行恢复(xlog数据块中包含了DB数据块的变更, 事务状态信息等, 深究可以去看一下xlog的头文件以及pg_xlogdump).
- 还原点介绍
- #recovery_target_name = ‘’ # e.g. ‘daily backup 2011-01-26‘ -- 不支持inclusive配置. 因为它不需要从abort或commit判断结束点.
- #
- #recovery_target_time = ‘’ # e.g. ‘2004-07-14 22:39:00 EST‘ -- 时间格式使用当前系统配置的格式, 或从时间函数获取
- #
- #recovery_target_xid = ''
- #
- #recovery_target_inclusive = true
- 默认支持3种还原点.
- 如果不设置还原点则不会停止恢复, 一般用于建立流复制或容灾环境.
- http://blog.163.com/digoal@126/blog/static/163877040201303082942271/

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 数据库物理还原

- XID还原点, 以commit或abort的xid到达为准.

- xid按请求顺序分配, 但是abort和commit点的xid在XLOG顺序中是无序的, 只要从读取的XLOG abort/commit信息到达指定xid就停止恢复.

- 命名的还原点,
- 如果数据库中有多个重复命名的还原点, 遇到第一个则停止.
- 同时因为还原点的信息写在单独的xlog数据块中, 不是一条transaction record块, 所以也没有包含或不包含的概念, 直接截止.
- 不需要判断recovery_target_inclusive .

- 时间还原点
- 在同一个时间点, 可能有多个事务COMMIT/ABORT. 所以recovery_target_inclusive 在这里起到的作用是 :
- 截止于这个时间点的第一个提交的事务后 (包含这个时间点第一个遇到的提交/回滚的事务);
- 或者截止于这个时间点提交的最后一个事务后 (包括这个时间点提交/回滚的所有事务) .

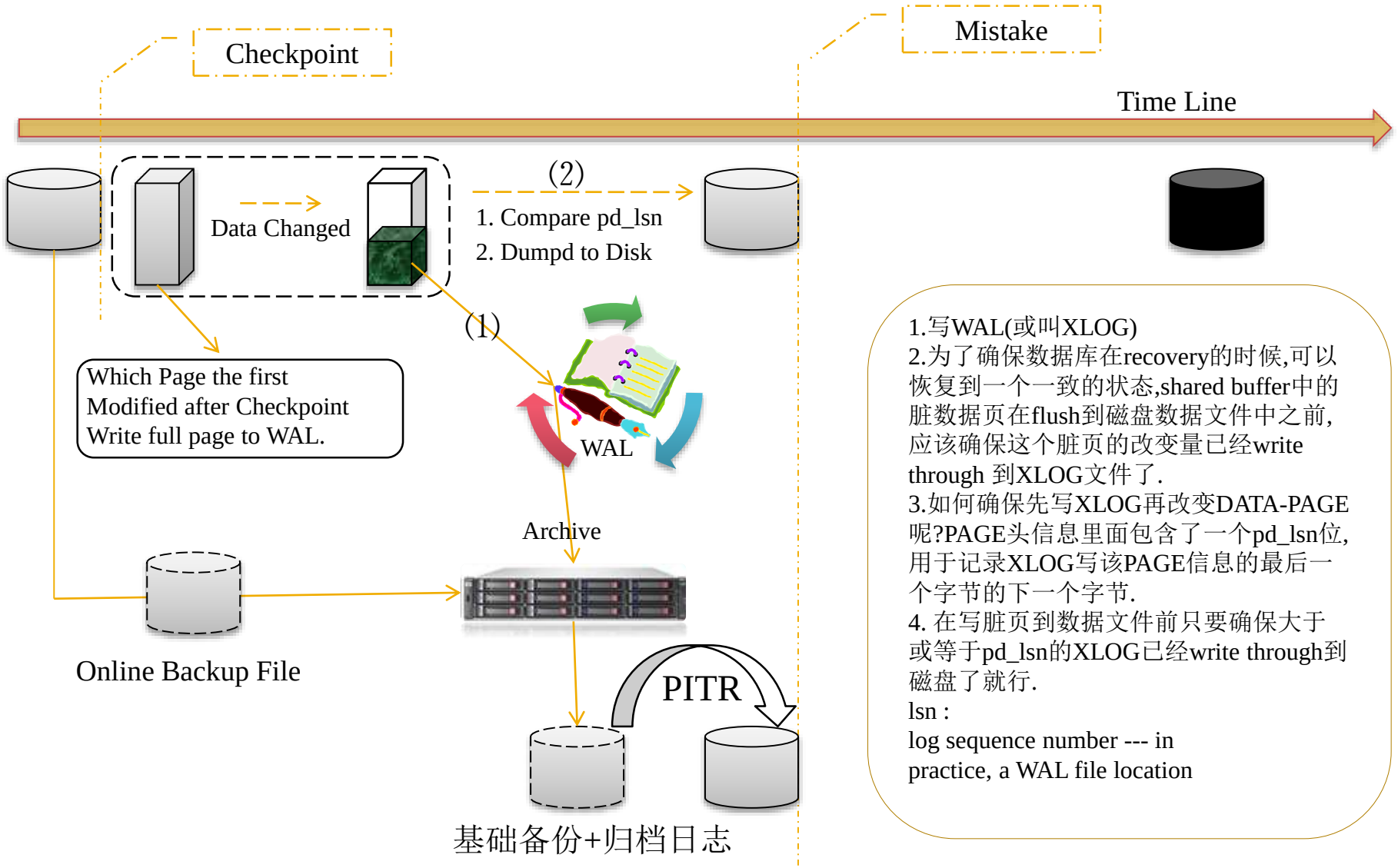
SKY mobi
新手机 新应用 新娱乐

PostgreSQL 数据库物理还原

- 物理还原
- 配置还原参数
  - 模板文件
  - $PGHOME/share/recovery.conf.sample
  - vi $PGDATA/recovery.conf
  - restore_command = 'cp /mnt/server/archivedir/%f %p'
  - recovery_target_timeline = 'latest'
- 启动数据库
  - pg_ctl start
- 配置hot_standby参数, 便于判断是否已经到达还原点. (可选, 仅做PITR时需要.一般都是恢复到最后)
- 检查是否到达指定还原点. (可选, 仅做PITR时需要.一般都是恢复到最后)
- 激活数据库
- 自由练习

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 数据库物理还原



1.写WAL(或叫XLOG)
2.为了确保数据库在recovery的时候,可以恢复到一个一致的状态,shared buffer中的脏数据页在flush到磁盘数据文件中之前,应该确保这个脏页的改变量已经write through 到XLOG文件了.
3.如何确保先写XLOG再改变DATA-PAGE呢?PAGE头信息里面包含了一个pd_lsn位,用于记录XLOG写该PAGE信息的最后一个字节的下一个字节.
4. 在写脏页到数据文件前只要确保大于或等于pd_lsn的XLOG已经write through到磁盘了就行.
lsn :
log sequence number --- in practice, a WAL file location

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 数据库逻辑备份

- pg_dump
  - -F c 备份为二进制格式, 压缩存储. 并且可被pg_restore用于精细还原
  - -F p 备份为文本, 大库不推荐.
- pg_dumpall
  - 可以备份全局元数据对象, 例如用户密码, 数据库, 表空间.
  - pg_dumpall 只支持文本格式.
- COPY命令, 在数据库库中执行COPY, 用于SQL子集或表的备份, 表的还原.

- 自由练习

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 数据库逻辑还原

- 二进制格式的备份只能使用pg_restore来还原, 可以指定还原的表, 编辑TOC文件, 定制还原的顺序, 表, 索引等.
- 文本格式的备份还原, 直接使用用户连接到对应的数据库执行备份文本即可, 例如psql dbname -f bak.sql
- 自由练习

SKY mobi
新手机 新应用 新娱乐

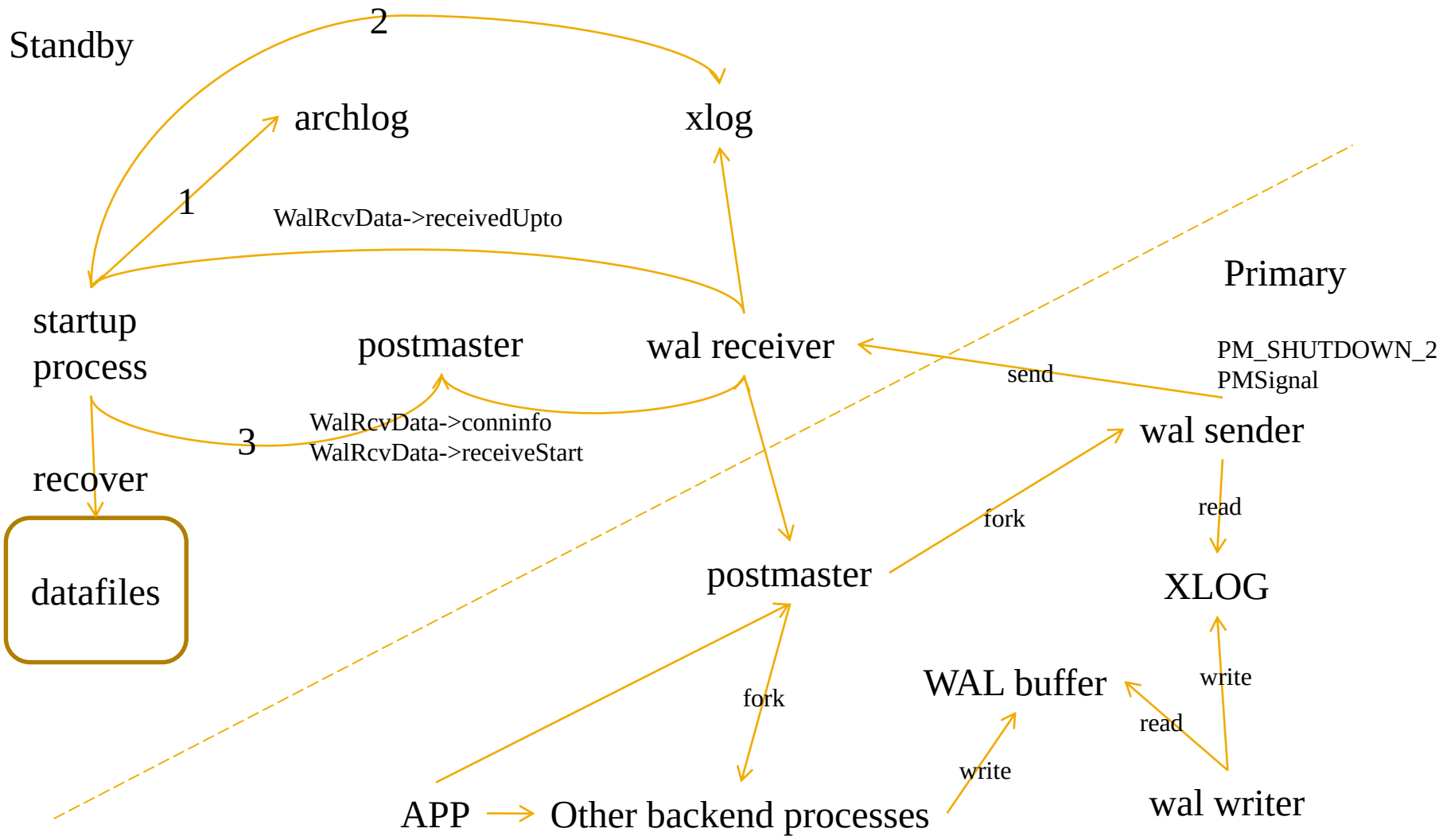
PostgreSQL 数据库流复制

- 数据库流复制
  - 9.0开始支持1+n的异步流复制.
  - 9.1支持1+1+n的同步和异步流复制
  - 9.2开始支持级联流复制
  - 9.3开始支持跨平台的流复制协议(目前可用于接收xlog).
  - 9.3开始流复制协议增加了时间线文件传输的协议, 支持自动切换时间线.

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 数据库流复制

- 异步流复制原理



SKY mobi
新手机 新应用 新娱乐

PostgreSQL 数据库流复制

Parameter Tuning :

Primary

max_wal_senders

wal_sender_delay (The sleep is interrupted by transaction commit)

wal_keep_segments

vacuum_defer_cleanup_age (the number of transactions by which VACUUM and HOT updates will defer cleanup of dead row versions.)

Standby

hot_standby

wal apply & SQL on standby conflict reference parameter

max_standby_archive_delay

(the maximum total time allowed to apply any one WAL segment's data.)

max_standby_streaming_delay

(the maximum total time allowed to apply WAL data once it has been received from the primary server)

wal_receiver_status_interval

(minimum frequency, The standby will report the last transaction log position it has written, the last position it has flushed to disk, and the last position it has applied.)

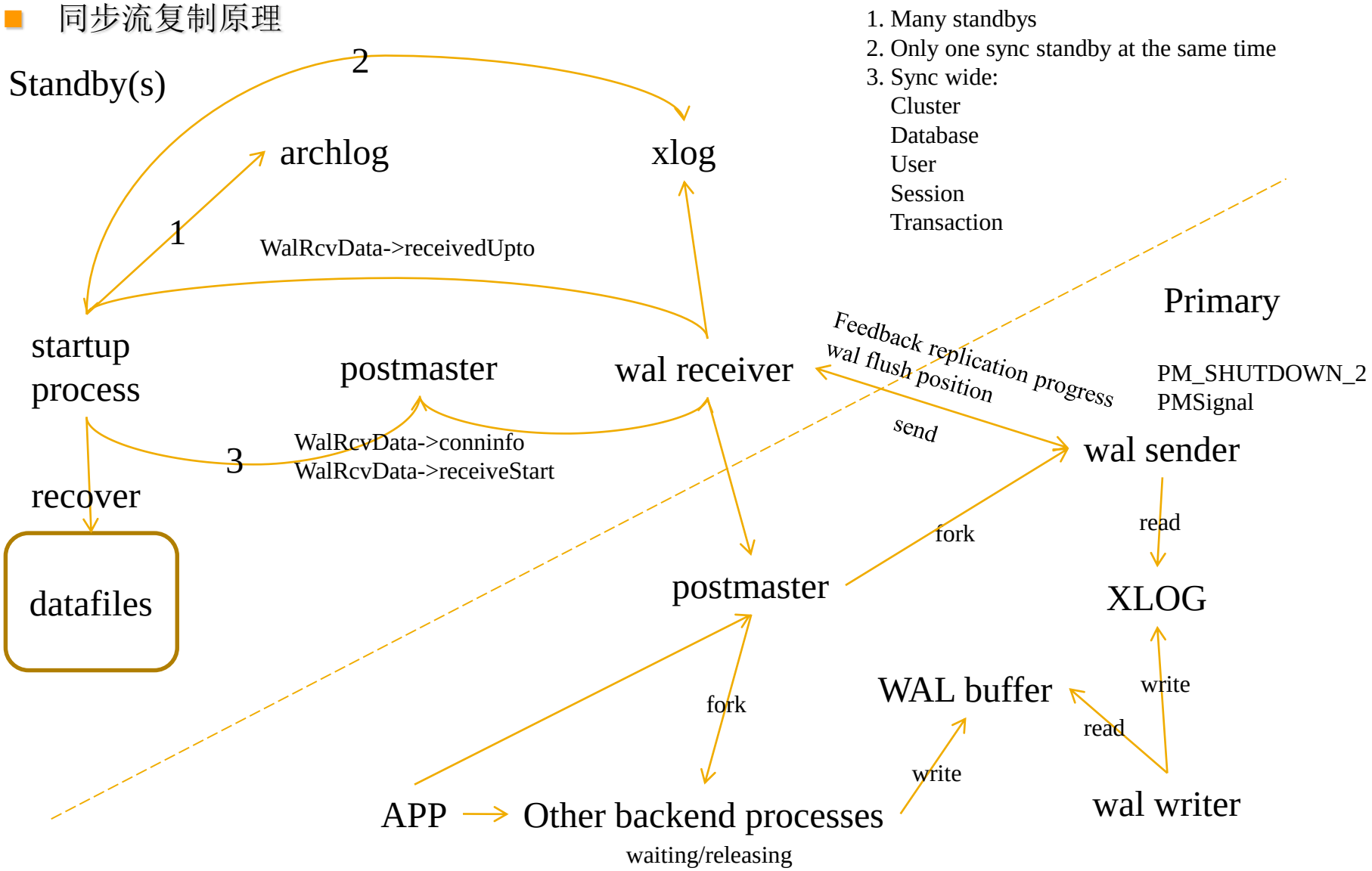
hot_standby_feedback

(send feedback to the primary about queries currently executing on the standby.)

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 数据库流复制

- 同步流复制原理



1. Many standbys
2. Only one sync standby at the same time
3. Sync wide:
   Cluster
   Database
   User
   Session
   Transaction

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 数据库流复制

Parameter Tuning :

Primary

- max_wal_senders
- wal_sender_delay
- wal_keep_segments
- vacuum_defer_cleanup_age
- synchronous_replication
- synchronous_standby_names
  - (primary_conninfo in standby's primary_conninfo)

Standby

- hot_standby
- max_standby_archive_delay
- max_standby_streaming_delay
- wal_receiver_status_interval
- hot_standby_feedback

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 数据库流复制

- 流复制hot_standby演示
  - 规划主机, 网络, 存储, 同步主备机器的时间
  - 生成主库
  - 配置主库postgresql.conf, pg_hba.conf
  - 新建replication 角色
  - 配置hot_standby .pgpass, 数据目录
  - 使用pg_basebackup创建备库基础备份
  - 配置备库recovery.conf, postgresql.conf
  - 启动hot_standby
  - 测试, 新建用户, 表空间, 数据库, schema, 数据表.
  - 使用pgbench进行压力测试, 查看流复制统计信息表pg_stat_replication
  - 角色切换测试

- 练习

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 数据库表级复制

- 使用触发器实现表级复制的介绍
- 物化视图介绍
- 支持表级复制的第三方插件(代理层SQL分发复制, 数据库端触发器复制)
  - Slony-I
  - Londiste3
  - Bucardo
  - Pgpool-II
  - Pl/proxy
  - ...
- 选取londiste3介绍

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 表复制-触发器

- 多主复制
- http://blog.163.com/digoal@126/blog/static/16387704020133211252201 34/

- 演示, 练习.

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 表复制-物化视图

- 本地物化视图
- Command: CREATE MATERIALIZED VIEW
- Description: define a new materialized view
- Syntax:
- CREATE MATERIALIZED VIEW table_name
- [(column_name [, ...])]
- [WITH (storage_parameter [= value] [, ...])]
- [TABLESPACE tablespace_name]
- AS query
- [WITH [NO] DATA]

- Command: REFRESH MATERIALIZED VIEW
- Description: replace the contents of a materialized view
- Syntax:
- REFRESH MATERIALIZED VIEW name
- [WITH [NO] DATA]

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 表复制-物化视图

- digoal=# create table tbl (id int primary key, info text, crt_time timestamp);
- CREATE TABLE
- digoal=# insert into tbl select generate_series(1,100000), md5(random()::text), clock_timestamp();
- INSERT 0 100000
- digoal=# create materialized view mv_tbl as select * from tbl where id<1000 with no data;
- SELECT 0
- digoal=# refresh materialized view mv_tbl ;
- REFRESH MATERIALIZED VIEW
- 9.4将添加物化视图增量刷新的功能.
- http://blog.163.com/digoal@126/blog/static/163877040201362383382/
- digoal=# select * from pg_matviews ;
- schemaname | matviewname | matviewowner | tablespace | hasindexes | ispopulated | definition
- ------------+-------------+--------------+------------+------------+-------------+-------------------------
- postgres | mv_tbl | postgres | | f | t | SELECT tbl.id, +
- | | | | | | tbl.info, +
- | | | | | | tbl.crt_time +
- | | | | | | FROM tbl +
- | | | | | | WHERE (tbl.id < 1000);
- (1 row)

SKY mobi
新手机 新应用 新娱乐

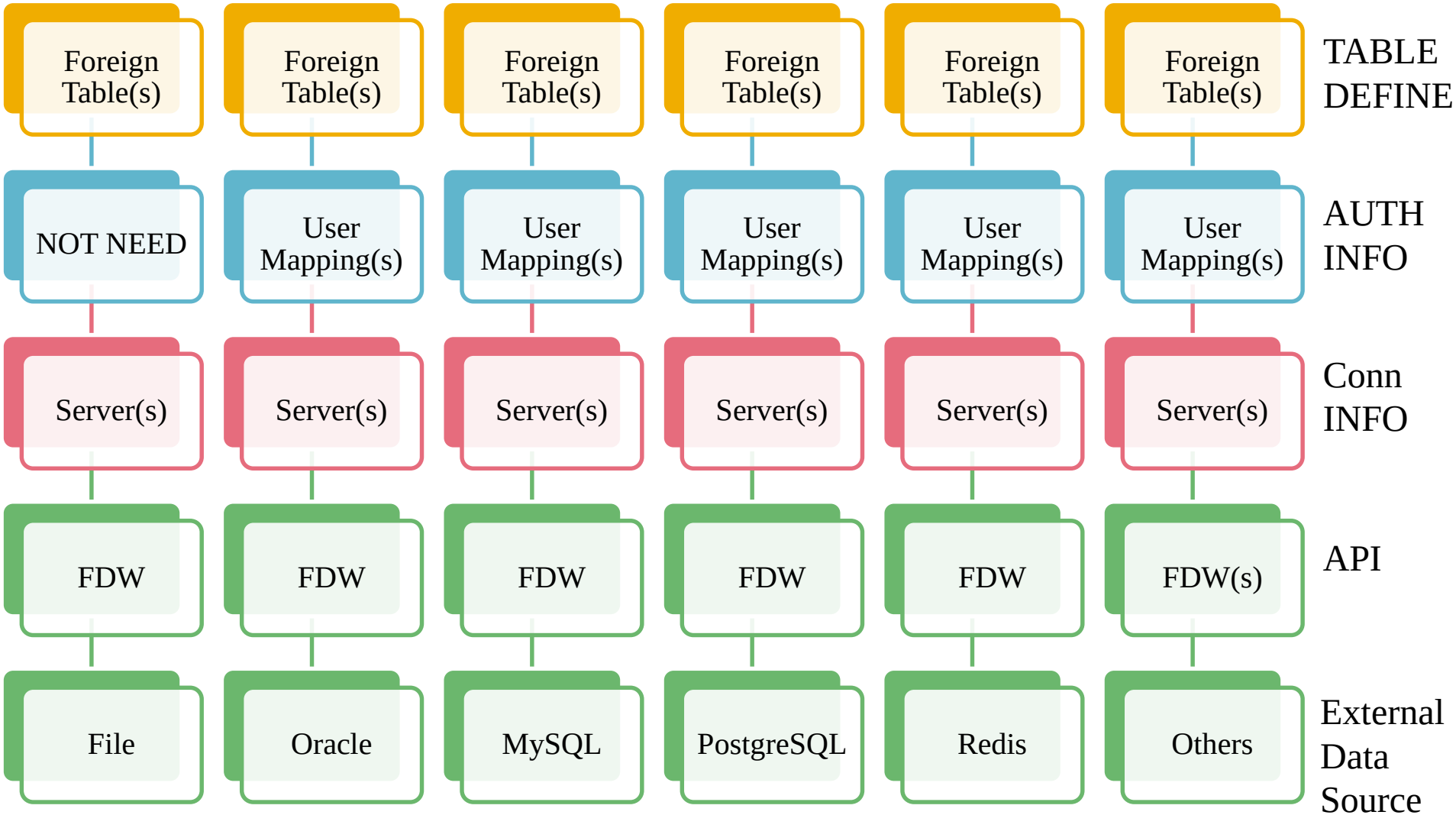
PostgreSQL 表复制-物化视图

- 异地物化视图, 基于外部表

- 外部表的概念
- Foreign data wrapper
- A foreign data wrapper is a library that can communicate with an external data source, hiding the details of connecting to the data source and fetching data from it.

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 表复制-物化视图



SKY mobi
新手机 新应用 新娱乐

PostgreSQL 表复制-物化视图

- PostgreSQL Foreign Table - pgsql_fdw
- http://blog.163.com/digoal@126/blog/static/163877040201231514057303/
- PostgreSQL Foreign Table - oracle_fdw 1
- http://blog.163.com/digoal@126/blog/static/163877040201181505331588/
- PostgreSQL Foreign Table - oracle_fdw 2
- http://blog.163.com/digoal@126/blog/static/163877040201181511623 40/
- PostgreSQL Foreign Table - oracle_fdw 3
- http://blog.163.com/digoal@126/blog/static/163877040201189519534 08/
- PostgreSQL Foreign Table - file_fdw
- http://blog.163.com/digoal@126/blog/static/163877040201141641148311/
- PostgreSQL Foreign Table - redis_fdw
- http://blog.163.com/digoal@126/blog/static/163877040201191811882 47/
- PostgreSQL Foreign Table - mysql_fdw 1
- http://blog.163.com/digoal@126/blog/static/163877040201111123352 4987/
- PostgreSQL Foreign Table - mysql_fdw 2
- http://blog.163.com/digoal@126/blog/static/163877040201211085516 98/

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 表复制-物化视图

- 异地物化视图例子
- 首先创建外部表, 以postgresql外部表为例子
- http://blog.163.com/digoal@126/blog/static/16387704020132141031444 14/
- http://blog.163.com/digoal@126/blog/static/16387704020131254491985 8/

- digoal=# CREATE EXTENSION postgres_fdw;
- CREATE EXTENSION
- digoal=# CREATE SERVER s1 FOREIGN DATA WRAPPER postgres_fdw;
- CREATE SERVER
- digoal=# alter server s1 options (add hostaddr '172.16.3.150', add port '1921', add dbname 'digoal');
- ALTER SERVER

- digoal=# grant usage on foreign server s1 to digoal;
- GRANT
- digoal=# CREATE USER MAPPING FOR digoal server s1 options (user 'postgres', password 'postgres');
- CREATE USER MAPPING

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 表复制-物化视图

- 创建外部表
- digoal=# \c digoal digoal
- You are now connected to database "digoal" as user "digoal".
- digoal=> create foreign table ft1 (id int, info text, crt_time timestamp) server s1 options(schema_name 'public', table_name 'test123');
- CREATE FOREIGN TABLE

- 在远程创建基表
- digoal=# create table public.test123(id int, info text, crt_time timestamp);
- CREATE TABLE

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 表复制-物化视图

- 在本地查看外部表, 9.3的postgres_fdw接口支持写外部表, 往里面写一些数据.
- digoal=# \c digoal digoal
- You are now connected to database "digoal" as user "digoal".
- digoal=> select * from ft1;
- id | info | crt_time
- ----+------+----------
- (0 rows)

- digoal=> insert into ft1 select generate_series(1,100), md5(random()::text), clock_timestamp();
- INSERT 0 100

- 基于外部表创建物化视图
- digoal=> create materialized view mv_ft1 as select * from ft1 with no data;
- SELECT 0

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 表复制-物化视图

- 刷新物化视图
- digoal=> refresh materialized view mv_ft1;
- REFRESH MATERIALIZED VIEW
- digoal=> select count(*) from mv_ft1 ;
- count
- -------
- 100
- (1 row)

- 物化视图不允许refresh, vacuum, analyze以及select以外的操作. 在pg_rewrite中限制了:hasForUpdate false
- digoal=> insert into mv_ft1 select * from ft1;
- ERROR: cannot change materialized view "mv_ft1"
- digoal=> delete from mv_ft1;
- ERROR: cannot change materialized view "mv_ft1"
- digoal=> truncate mv_ft1;
- ERROR: "mv_ft1" is not a table

SKY mobi
新手机 新应用 新娱乐

PostgreSQL 表复制-londiste3

- 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/

- 演示, 练习.

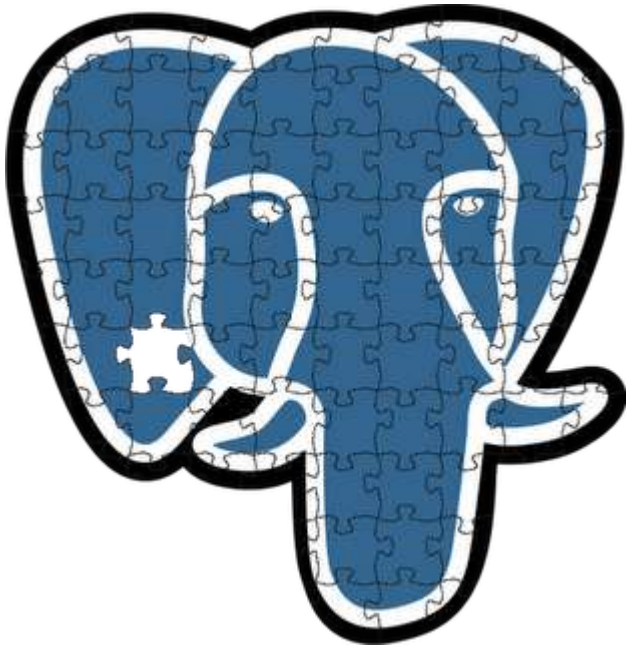
SKY mobi
新手机 新应用 新娱乐

练习

- 数据库热备份与还原练习
- 流复制练习
- 表级复制练习

SKY mobi
新手机 新应用 新娱乐

Thanks



- 关于本PPT有问题请发邮件至 digoal@126.com
- 保持联系, 个人QQ: 276732431
- 群: 3336901
- 【参考】
- 《PostgreSQL 9.3 Manual》
- 【更多内容请关注】
- http://blog.163.com/digoal@126/