

PostgreSQL sharding based on FDW

digoal

阿里云

2016PostgreSQL中国用户大会

目录

- 水平拆分可选开源方案
- why sharding based on fdw?
- 继承、触发器
- pg_pathman
- CASE

2016Postgres中国用户大会

单节点性能指标参考数据

- 秒杀
 - 8 Core, 23万 qps
- KNN近邻查询
 - 16 Core, 100亿数据, 64并发, KNN查询平均响应时间**0.848毫秒**, qps **74151**.
- 模糊查询、正则匹配
 - 8Host, 16Core, 1008亿数据, 前后模糊、正则匹配, **秒级**响应
- 分词
 - 英语分词性能: ~900万 words每秒 (Intel(R) Xeon(R) CPU X7460 @ 2.66GHz)
 - 中文分词性能: ~400万字每秒 (Intel(R) Xeon(R) CPU X7460 @ 2.66GHz)
 - 英文分词+插入性能: ~666万字每秒 (Intel(R) Xeon(R) CPU X7460 @ 2.66GHz)
 - 中文分词+插入性能: ~290万字每秒 (Intel(R) Xeon(R) CPU X7460 @ 2.66GHz)
- 并行计算
 - CPU并行 32Core, 16亿(90GB), count (*) 7秒, bit(and, xor) 16秒, 非并行(141秒, 488秒).
 - GPU并行 (1张 1亿 table join 9张 10万 table) 21秒, 非并行520秒.

单节点性能指标参考数据

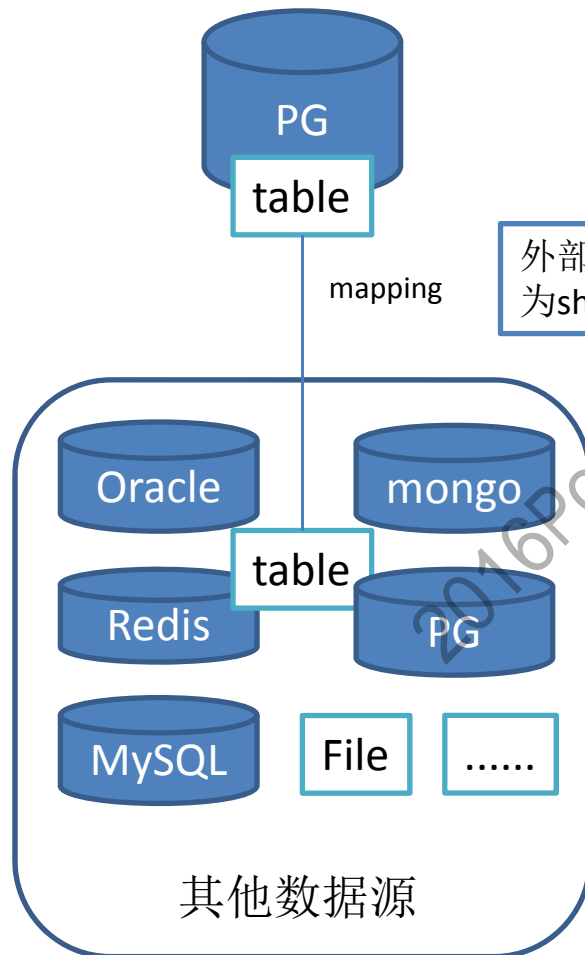
- 数据装载
 - 32Core, 512G, 2*Aliflash SSD
 - 连续24小时多轮数据批量导入测试(平均每条记录长度360字节, 时间字段索引)
 - 每轮测试插入12TB数据
 - 506万行/s, 1.78 GB/s, 全天插入4372亿, 154TB数据
 - (为什么这么快?) (BRIN, HEAP, 动态扩展FILE, prealloc XLOG, reuse XLOG)
- TPC-B (1 Select : 3 Update : 1 Insert)
 - 32Core, 512G, 2*Aliflash SSD 10亿数据量, 11万tps, 77万qps
 - Select-Only 100万tps (即使应用缓存失效, 也无大碍)
- TPC-C (新建订单45,支付43,订单查询4,发货4,库存查询4)
 - 4000个仓库, 400GB数据, 平均每笔事务10几条SQL
 - 12Core, 256GB, intel SSD , 61万TPmC (IO瓶颈严重,理论上可以达到200万)
- LinkBench (Facebook 社交关系应用)
 - 1亿个node, 4亿条关系, (32Core, 2 SSD, 512G)
 - (添加NODE, 更新NODE, 删除NODE, 获取NODE信息, 添加关系, 删除关系, 更新关系, 关系总数查询, 获取多个关系, 获取关系列表)
 - 12万 ops (默认测试用例)

水平拆分可选开源方案

- OLTP
 - plproxy
 - pgpool-II
 - citusdata
 - pg-x2
 - FDW
- OLAP
 - greenplum
 - hawq
 - pg-xl

2016Postgres中国用户大会

FDW



外部表支持写入
为sharding 提供基础支持

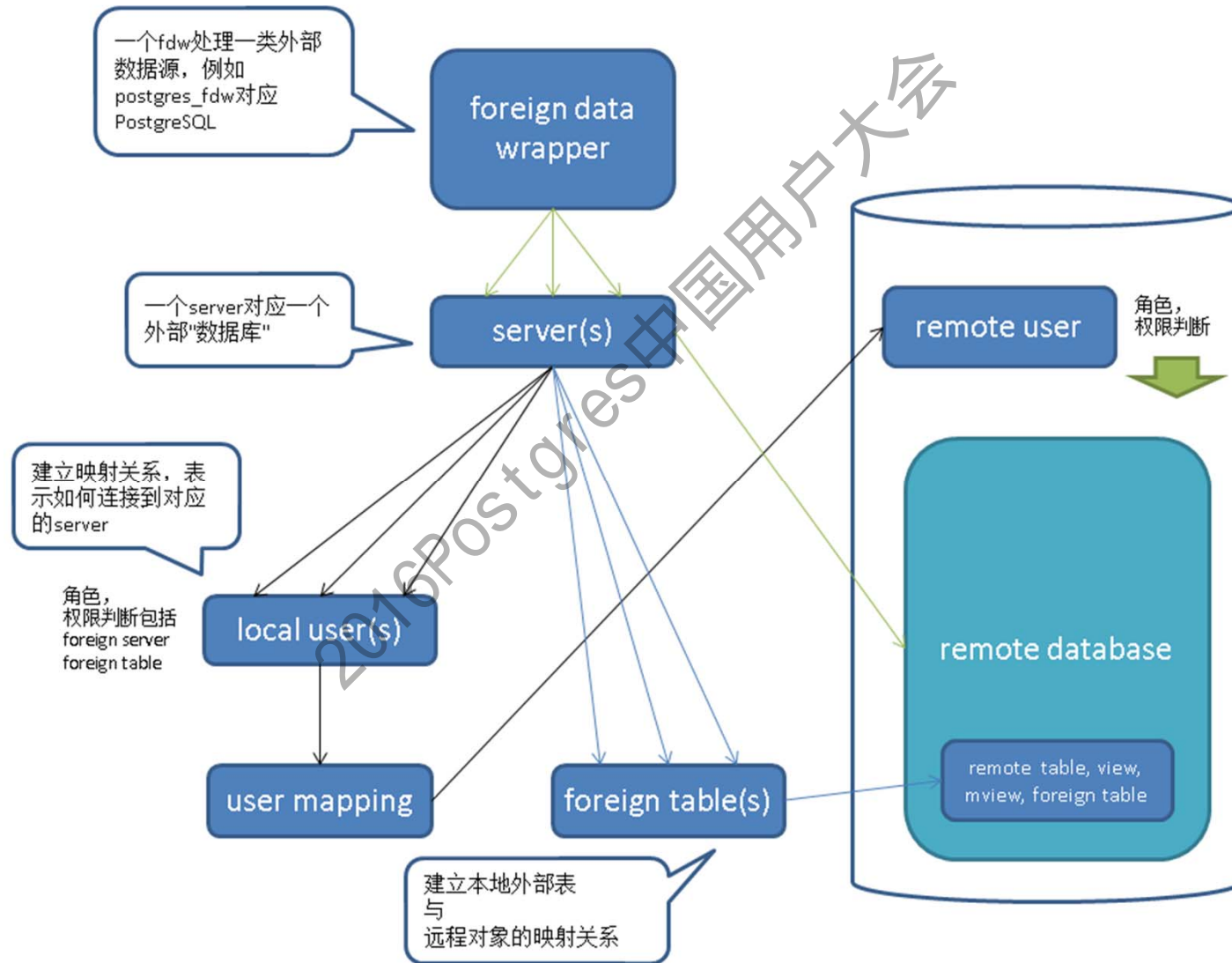
FDW API :

- scan api
- join api
- post scan/join api
- update api
- row lock api
- explain, analyze api
- import api
- parallel exec api

大象装进冰箱需要几步？

1. create server; -- host:port:dbname:other_ops
2. create user mapping; -- auth
3. create foreign table; -- table ddl mapping

FDW



why sharding based on fdw?

- postgres_fdw 9.6 增强
- 连接复用
 - 只要server相同, remote user相同, 就可以复用连接
- 支持server extension option
 - 除了支持build-in func\ops, 也支持extension func\ops
- JOIN下推
 - server和user mapping相同的外表JOIN
 - JOIN的operation、function, 必须buildin -func immutable; 或者在server属性extension内, 属于extension中的immutable function、operations
- SORT下推
- WHERE子句算子下推
 - buildin -func immutable; 或者在server属性extension内, 属于extension中的immutable function、operations
- DML下推
- 允许设置表级或server级别fetch_size
- 允许cancel请求传递给remote session

why sharding based on fdw?

SORT

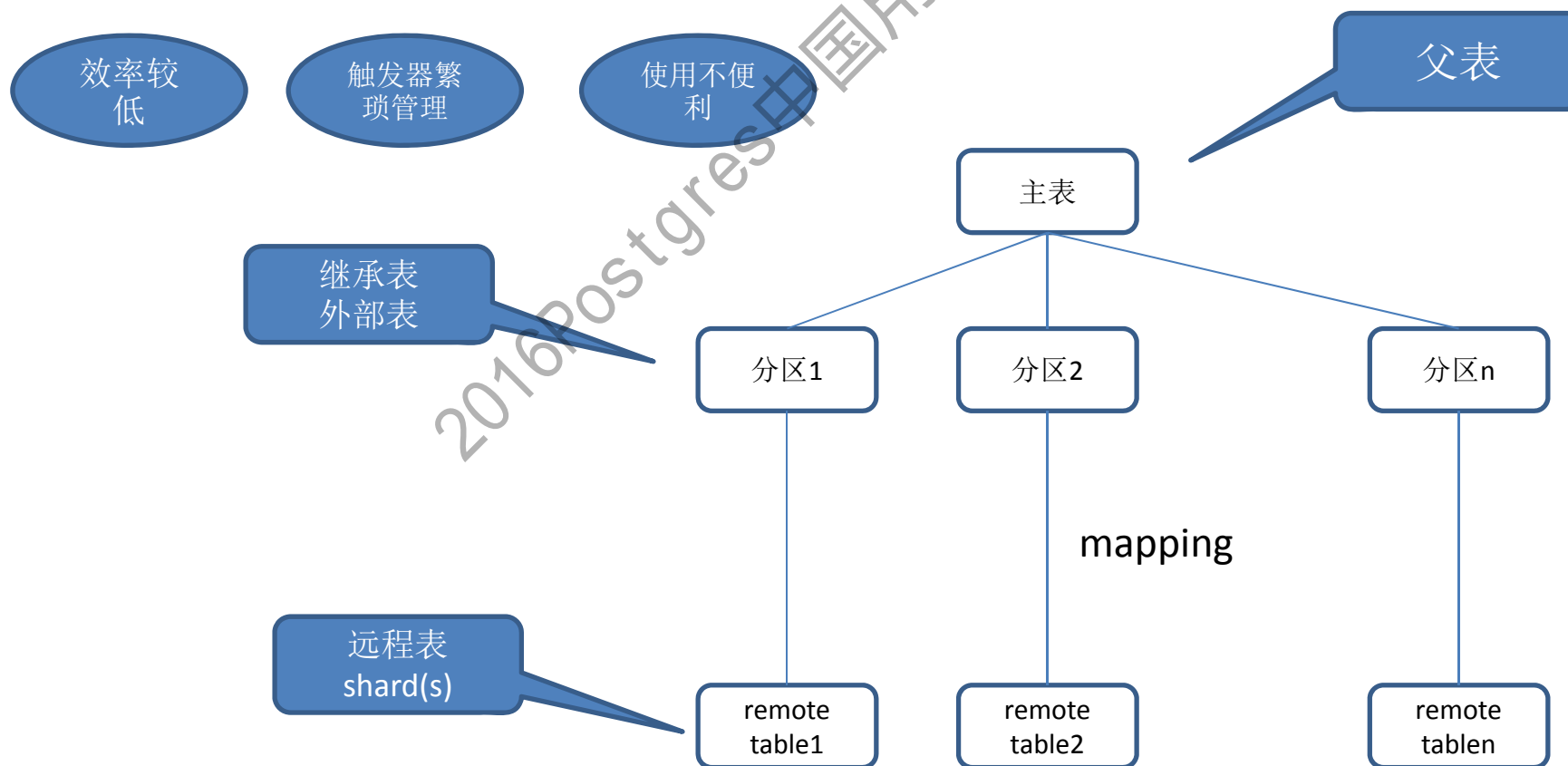
```
mas1=> explain (analyze,verbose,timing, costs,buffers) select * from userinfo order by uid desc limit 10;
QUERY PLAN
-----
Limit (cost=400.04..400.47 rows=10 width=48) (actual time=3.075..3.084 rows=10 loops=1)
  Output: userinfo_0.uid, userinfo_0.info, userinfo_0.crt_time
  -> Merge Append (cost=400.04..593.58 rows=4548 width=48) (actual time=3.073..3.080 rows=10 loops=1)
      Sort Key: userinfo_0.uid DESC
      -> Foreign Scan on digoal.userinfo_0 (cost=100.00..148.38 rows=1137 width=48) (actual time=0.844..0.845 rows=5 loops=1)
          Output: userinfo_0.uid, userinfo_0.info, userinfo_0.crt_time
          Remote SQL: SELECT uid, info, crt_time FROM role0.userinfo_0 ORDER BY uid DESC NULLS FIRST
      -> Foreign Scan on digoal.userinfo_1 (cost=100.00..148.38 rows=1137 width=48) (actual time=0.733..0.733 rows=1 loops=1)
          Output: userinfo_1.uid, userinfo_1.info, userinfo_1.crt_time
          Remote SQL: SELECT uid, info, crt_time FROM role1.userinfo_1 ORDER BY uid DESC NULLS FIRST
      -> Foreign Scan on digoal.userinfo_2 (cost=100.00..148.38 rows=1137 width=48) (actual time=0.714..0.714 rows=4 loops=1)
          Output: userinfo_2.uid, userinfo_2.info, userinfo_2.crt_time
          Remote SQL: SELECT uid, info, crt_time FROM role2.userinfo_2 ORDER BY uid DESC NULLS FIRST
      -> Foreign Scan on digoal.userinfo_3 (cost=100.00..148.38 rows=1137 width=48) (actual time=0.776..0.776 rows=3 loops=1)
          Output: userinfo_3.uid, userinfo_3.info, userinfo_3.crt_time
          Remote SQL: SELECT uid, info, crt_time FROM role3.userinfo_3 ORDER BY uid DESC NULLS FIRST
  Planning time: 0.261 ms
  Execution time: 4.007 ms
(18 rows)
```

```
mas1=> explain verbose update userinfo_1 set info=info where uid=1;
QUERY PLAN
-----
Update on digoal.userinfo_1 (cost=100.00..123.22 rows=5 width=54)
 -> Foreign Update on digoal.userinfo_1 (cost=100.00..123.22 rows=5 width=54)
     Remote SQL: UPDATE role1.userinfo_1 SET info = info WHERE ((uid = 1))
(3 rows)
```

DML

继承、触发器问题

- 通过分区列约束筛选对应分区
- 通过触发器将数据写入对应分区
- 哈希分区的问题，查询时必须带上哈希函数

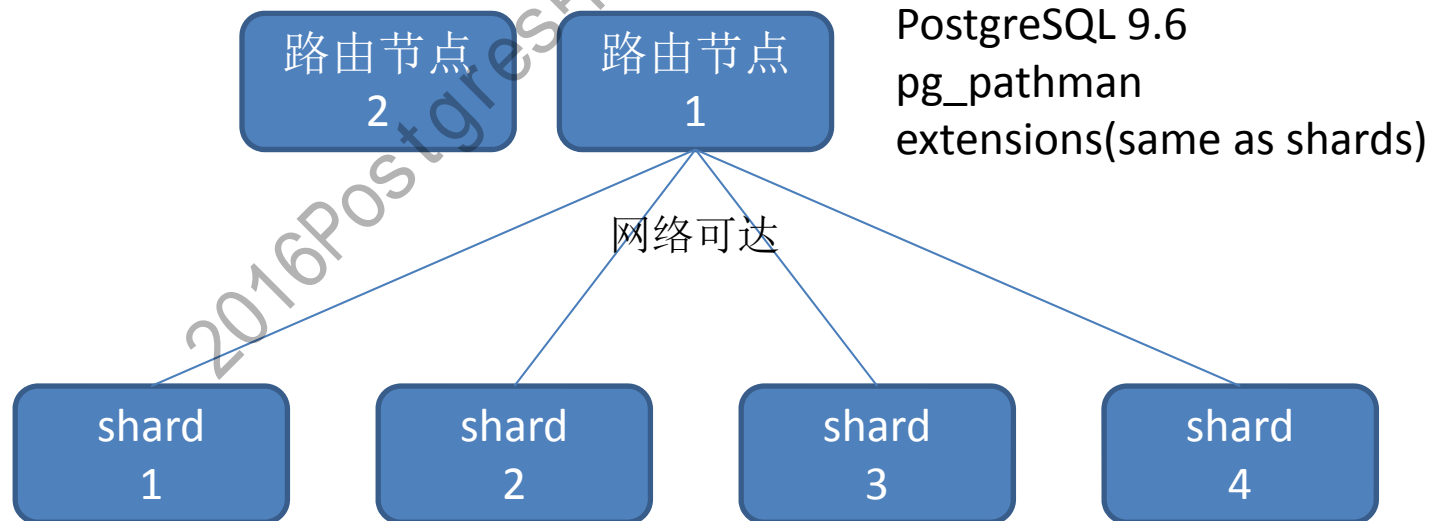


pg_pathman

- 支持HASH, range分区
- 支持自动分区、手动分区（合并、拆分、新增、删除、绑定）
- 支持大多数数据类型
- 执行计划高效（JOIN \ SUBQUERY）
- 动态选择分区（subquery）
- 高效插入、COPY FROM\TO（非触发器，使用HOOK替换rel oid）
- 支持自动扩展分区(范围分区)
- 支持更新分区列
- 支持分区DDL回调函数（如支持DDL逻辑复制）
- 非堵塞式迁移数据
- 支持FDW

CASE

segment 1 : 127.0.0.1:5281:db0:role0:pwd , schema_name role0
segment 2 : 127.0.0.1:5281:db1:role1:pwd , schema_name role1
segment 3 : 127.0.0.1:5281:db2:role2:pwd , schema_name role2
segment 4 : 127.0.0.1:5281:db3:role3:pwd , schema_name role3
master 1 : 127.0.0.1:5281:mas1:digoal:pwd , schema_name digoal
master 2 : 127.0.0.1:5281:mas2:digoal:pwd , schema_name digoal



PostgreSQL 任意版本
extensions (opt)

支持阿里云RDS PGSQL

场景设计

userinfo 增删改查

新增用户、销毁用户、
修改用户资料、查询用户资料

user_log 增查

新增用户登陆日志、查询用户日志

user_membership 增删查

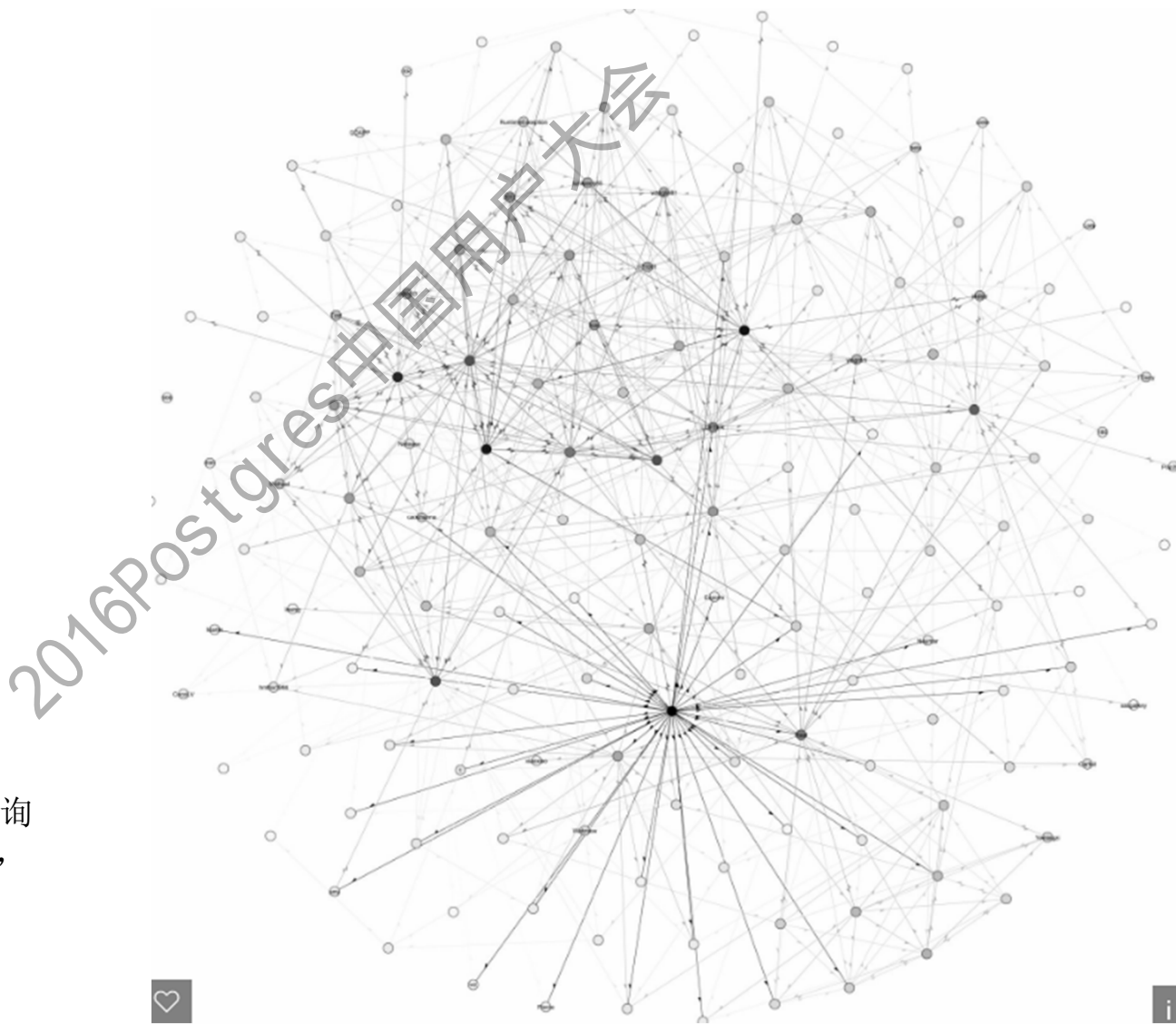
新增用户关系，
删除用户关系，
查询用户关系

user_membership_rev 增删查

新增反向用户关系，
删除反向用户关系，
查询反向用户关系

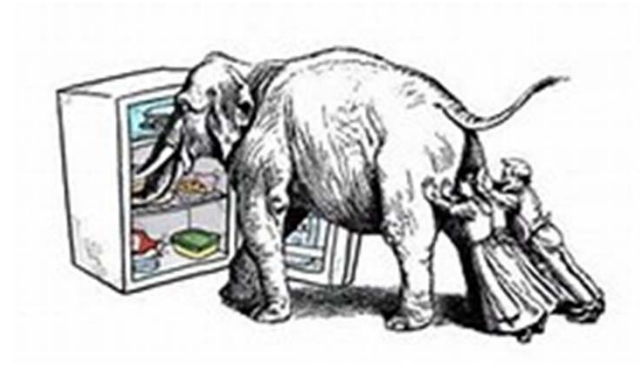
mv_user_membership 刷新、查询

mv_user_membership_rev 刷新、查询
聚物化视图，提高关系查询效率，
刷新物化视图，
查询物化视图



配置

- 准备路由节点、数据节点
- `func (arg jsonb(url,ddl)) $$`
 - create extension pg_pathman
 - create foreign server for all datanode on routenode
 - run DDL on all DATANODE
 - import foreign schema on routenode
 - create partition table on routenode
 - attach foreign table to partition table(specify **range or hash** bound)
- `$$`
- use partition table



负载均衡、HA

or jdbc-balance
others

APP / HAproxy

应用可以直连任意master
或者通过haproxy之类的负载均衡代理连接master

master 1
meta only

master 2
meta only

master n
meta only

对等
水平扩展

or
多副本

DB1

DB2

DB3

DBn

水平扩展

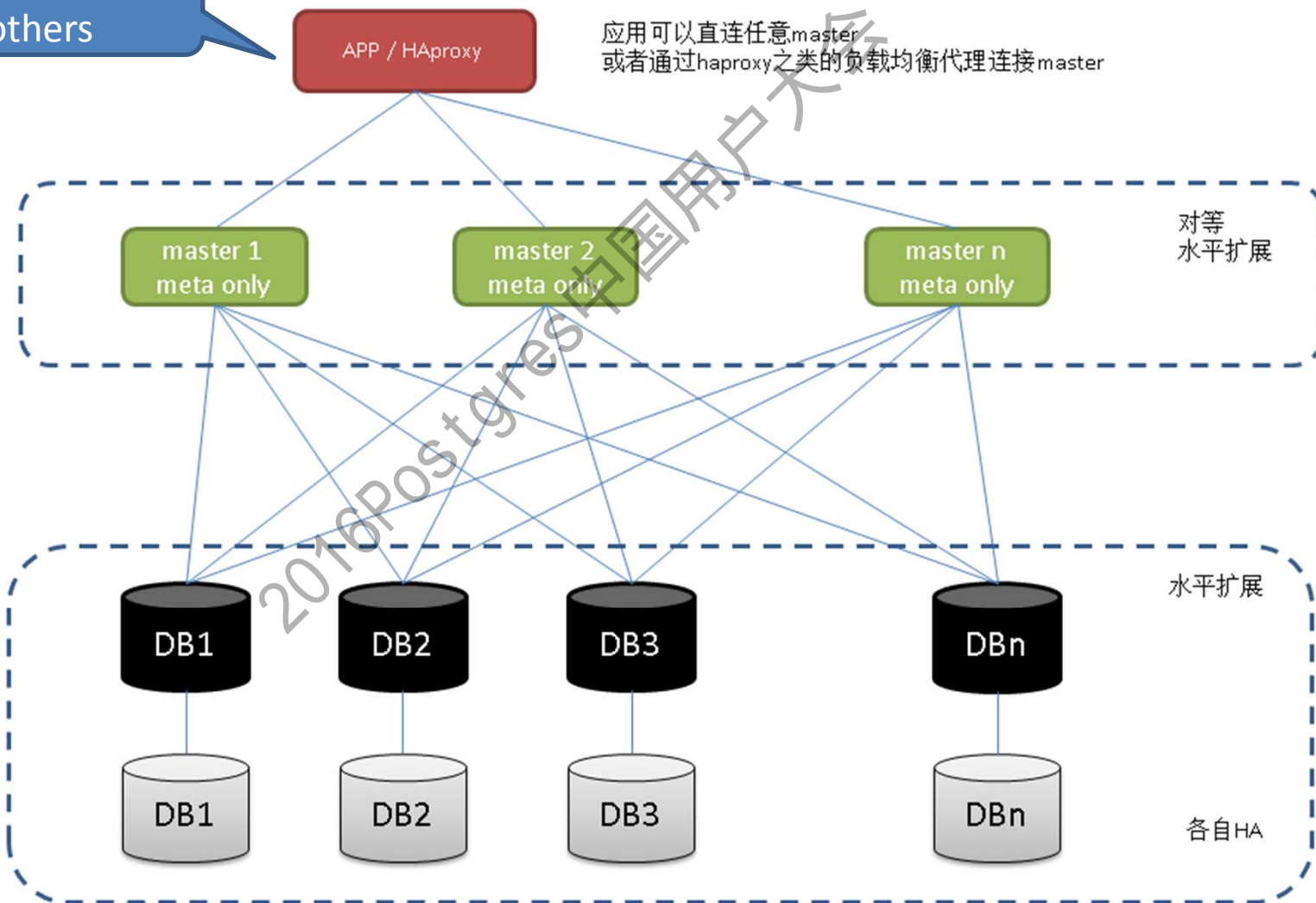
DB1

DB2

DB3

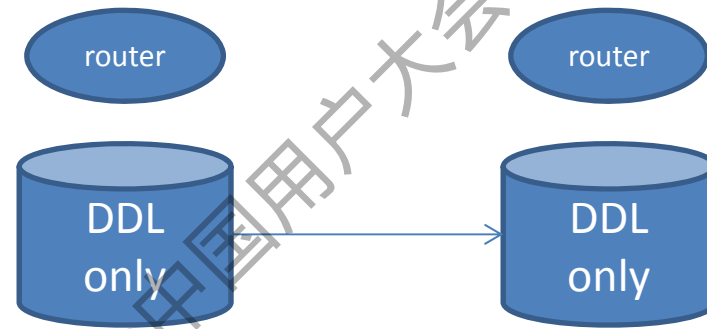
DBn

各自HA



路由节点/shard节点 扩容、缩容、迁移

- 路由节点
 - 扩容
 - 缩容
 - 迁移
- 数据节点
 - 扩容
 - 缩容
 - 迁移

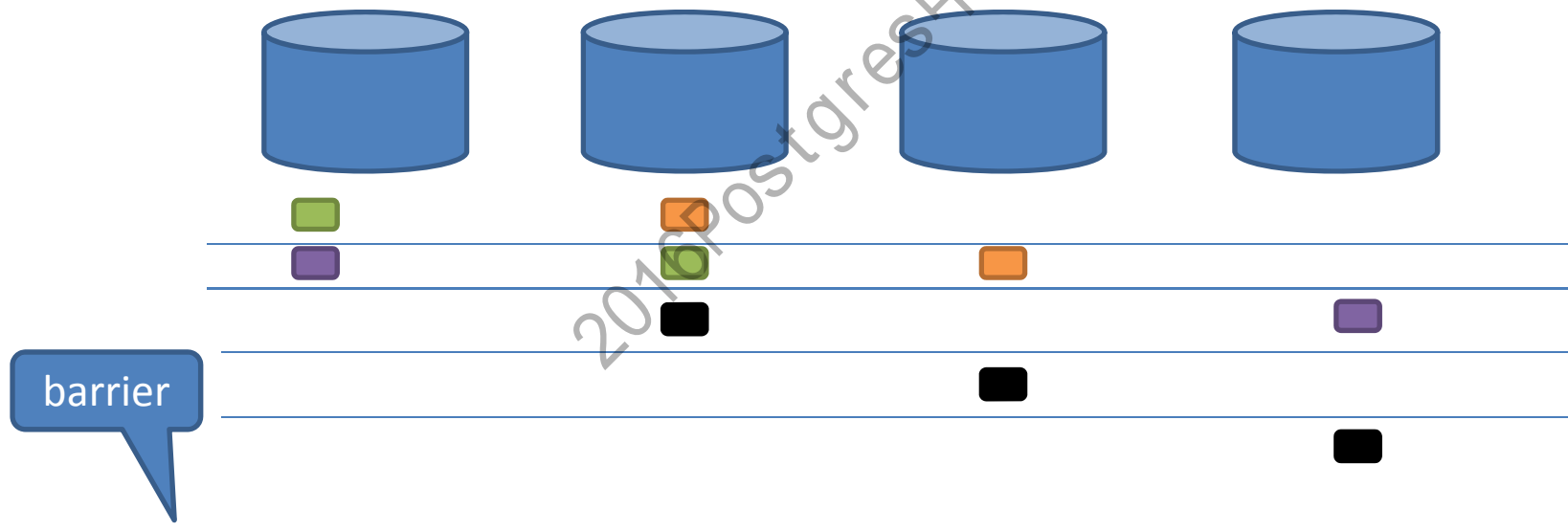


预分片
split 分片

logical replication
detach
attach

全局一致性备份

- barrier
- 时间点恢复



其他

- remote 连接稳定性, 超时
 - 认证超时、TCP超时、QUERY超时、LOCK超时
- 分布式事务
- 聚合 pushdown (10.0)
- append 并行化

- 本案详细文档
- https://github.com/digoal/blog/blob/master/201610/20161004_01.md
- https://github.com/digoal/blog/blob/master/201610/20161005_01.md
- https://github.com/digoal/blog/blob/master/201610/20161024_01.md
- https://github.com/digoal/blog/blob/master/201610/20161027_01.md