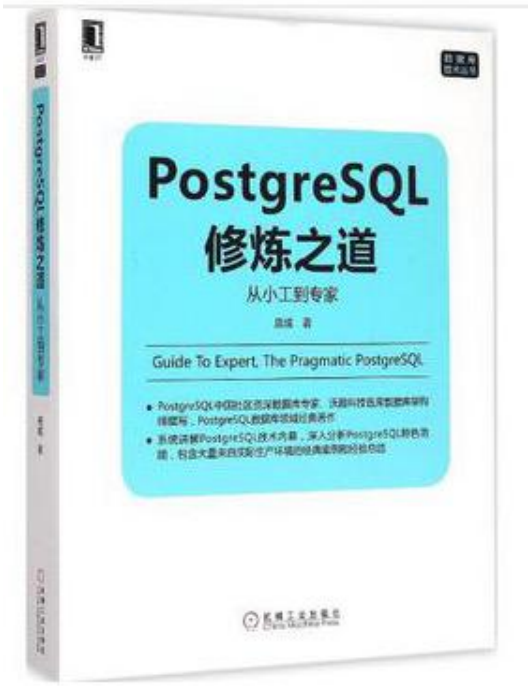
PostgreSQL中的分库分表解决方案

唐成 (网名osdba)
cheng.tang@postgres.cn
China Postgres User Group

个人介绍



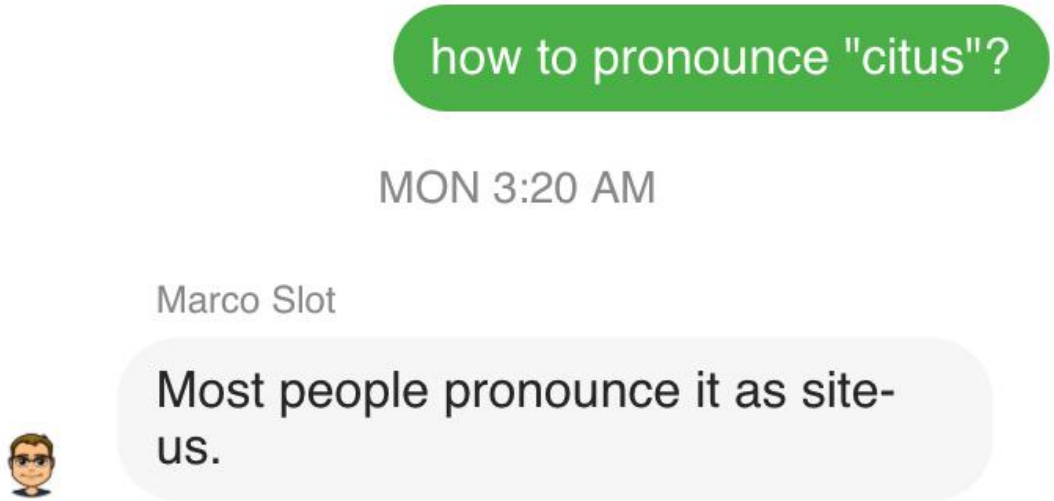
唐成（网名osdba）

云徙科技云技术总监

《PostgreSQL修炼之道：从小工到专家》的作者，目前为PostgreSQL中国用户会常务委员，PostgreSQL用户会杭州站负责人。从业近20年，拥有十几年数据库、操作系统、存储领域的工作经验，历任过阿里巴巴高级数据库专家，从事过阿里巴巴Greenplum、PostgreSQL数据库的架构设计和运维。曾任网易杭州研究院开发专家，主导了网易云计算中的云硬盘产品（类似Amazon EBS）的设计和开发。目前主要技术方向是数据库和分布式云存储以及周边的各种云计算技术。

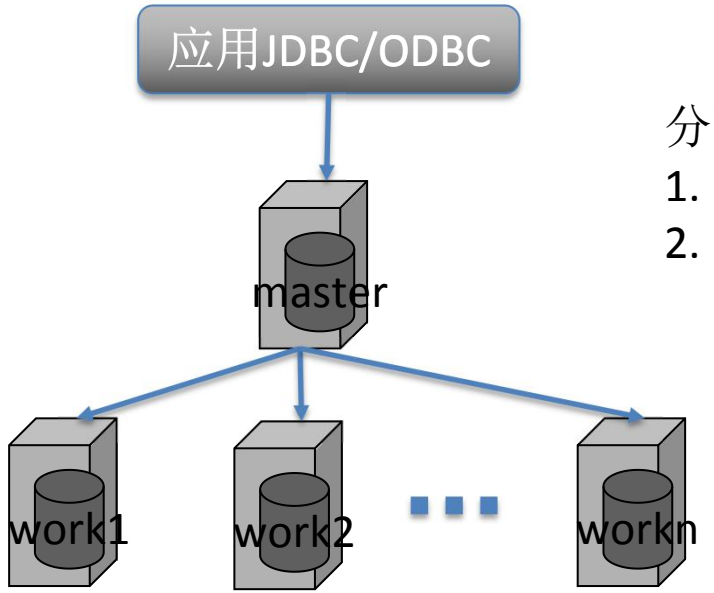
目录

How to pronounce “citus”



citus是什么？

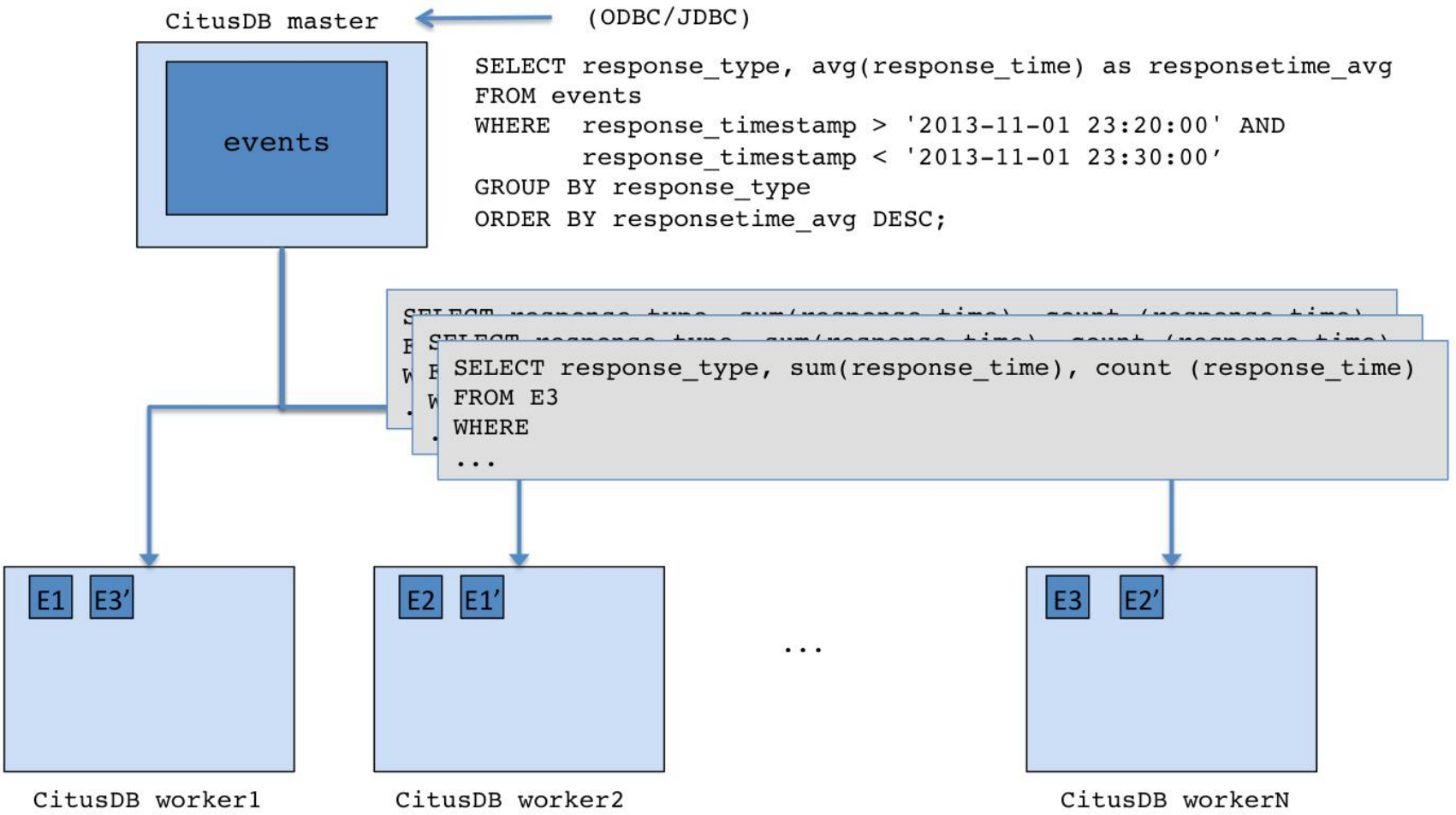
- citus是PostgreSQL数据库中的一个插件



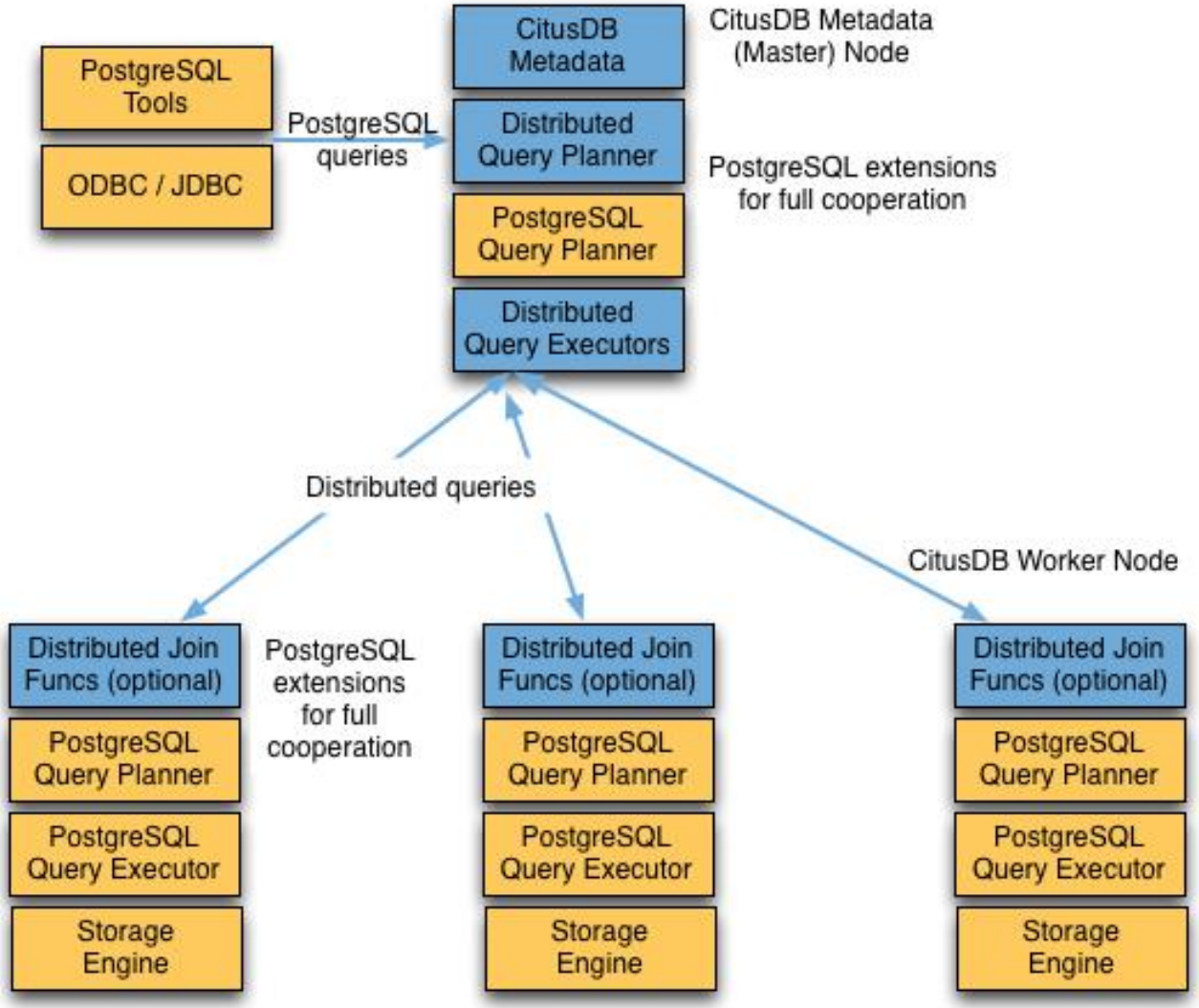
分两类节点，都是一台PostgreSQL数据库：

1. master节点：存分库分表的元数据
2. worker节点：存储真正的分片数据

什么是citus？


CitusDB master
(ODBC/JDBC)
events
SELECT response_type, avg(response_time) as responsetime_avg
FROM events
WHERE response_timestamp > '2013-11-01 23:20:00' AND
response_timestamp < '2013-11-01 23:30:00'
GROUP BY response_type
ORDER BY responsetime_avg DESC;
SELECT response_type, sum(response_time), count (response_time)
FROM E3
WHERE
...
E1
E3'
E2
E1'
...
E3
E2'
CitusDB worker1
CitusDB worker2
CitusDB workerN

citus查询的处理


PostgreSQL Tools
ODBC / JDBC
PostgreSQL queries
CitusDB Metadata
Distributed Query Planner
PostgreSQL Query Planner
Distributed Query Executors
CitusDB Metadata (Master) Node
PostgreSQL extensions for full cooperation
Distributed queries
CitusDB Worker Node
Distributed Join Funcs (optional)
PostgreSQL Query Planner
PostgreSQL Query Executor
Storage Engine
PostgreSQL extensions for full cooperation
Distributed Join Funcs (optional)
PostgreSQL Query Planner
PostgreSQL Query Executor
Storage Engine
Distributed Join Funcs (optional)
PostgreSQL Query Planner
PostgreSQL Query Executor
Storage Engine

目录

citus的软件安装

- 方法一：编译安装citus：
  - git clone https://github.com/citusdata/citus.git
  - cd citus
  - ./configure
  - make
- 前提：
  - 这台机器上安装了PostgreSQL，通过pg_config可以找到一些头文件

citus的软件安装

- 方法二：Linux包管理器安装编译好的citus：
  - Ubuntu:
    - aptitude install postgresql-9.6 postgresql-9.6-citus
- 安装说明：
  - 在ubuntu中已带有citus包
  - 在centos中也带有citus包

citus集群创建

- 创建步骤：
  - 建多台PostgreSQL数据库，选择一台做为master节点，其它做为worker节点
  - 在master节点和worker节点上安装citus扩展：
    - create extension citus;
    - 在postgresql.conf中增加：shared_preload_libraries = 'citus'
  - 在master节点上添加worker节点：
    - SELECT * from master_add_node(’work01', 9702);
    - SELECT * from master_add_node(’work02', 9703);

目录

citus建表

- 两类表：
  - 分片表（distrubuted tables）
  - 广播表（Reference Tables）
- 建分片表
  - 在master，与原PostgreSQL库中一样的方法建一张表，如：
    - create table t01(id1 int, id2 int, t text);
  - 通过函数create_distributed_table把表定义成hash分片表：
    - **select** create_distributed_table('t01', 'id2');

citus建表

- 建广播表
  - 在master，与原PostgreSQL库中一样的方法建一张表，如：
    - create table t02(id1 int, id2 int, t text);
  - 通过函数create_reference_table把表定义成广播表：
    - **SELECT** create_reference_table('states');

citus建表

- 分片表的个数
  - 是每个节点一张吗？
  - 底层节点上看到的：

```
postgres=# \d
             List of relations
 Schema |     Name      | Type  | Owner
--------+---------------+-------+--------
 public | t01_102280 | table | citusr
 public | t01_102282 | table | citusr
 public | t01_102284 | table | citusr
 public | t01_102286 | table | citusr
 public | t01_102288 | table | citusr
 public | t01_102290 | table | citusr
 public | t01_102292 | table | citusr
 public | t01_102294 | table | citusr
 public | t01_102296 | table | citusr
 public | t01_102298 | table | citusr
 public | t01_102300 | table | citusr
 public | t01_102302 | table | citusr
 public | t01_102304 | table | citusr
 public | t01_102306 | table | citusr
 public | t01_102308 | table | citusr
 public | t01_102310 | table | citusr
(16 rows)
```

此集群中有两个worker节点，每个节点中都出现16张分片表，总共32张表

citus建表

- 分片表的个数（续）
  - 默认是32张，是由参数citus.shard_count指定的
  - citus.shard_count可以动态改变，改变后，后面再创建表的分片数就会改变。
- 注意：
  - 分片数不同的表进行join时，可能会出现问题

citus建表

- 第二种分片表：Append Distribution
  - 常常用做时间序列的数据使用
  - 每一个分片是一个时间范围
  - 提供了函数，方便删除某个时间范围的数据
  - 分片数是按添加数据时动态创建
    - 其中参数citus.shard_max_size限制了每个分片表的大小
- 建表语句：

```
create table append_table_01(id1 int, id2 int, t text, tm timestamp);
SELECT create_distributed_table('append_table_01', 'tm', 'append');
```

citus建表：append表的例子

citus建表

- Append Distribution过期数据的删除

SELECT * from master_apply_delete_command('DELETE FROM append_table01 WHERE tm <=''2013-06-01 00:00:00''');

- 上面的语句
  - 是drop各个分片表，而不是执行delete from语句，如果要一条一条删除，应该执行下面的SQL:

SELECT master_modify_multiple_shards('DELETE FROM append_table01 WHERE tm <=''2013-06-01 00:00:00''');

citus对DDL语句的支持

- 删除表：
  - 可以使用drop table命令
- alter table
  - 支持大数多的alter table，把语句直接发到各个底层节点
- 提供了手工执行的方
  - citus.enable_ddl_propagation
- 保证一致性
  - SET citus.multi_shard_commit_protocol TO '2pc'

citus对DML语句的支持

- insert：
  - 支持单条insert
  - 不支持insert into t(id, t) values(1,’11’),(2,’22)；这样的多条插入
  - 也不支持insert … select … from tab;
- update和delete
  - where条件上有分布键，能定位到某一个shard中
  - 如果要在所有的shard中删除：
    - SELECT master_modify_multiple_shards('DELETE FROM github_events WHERE repo_id IN (24509048, 24509049)');

citus查询的处理

- 两种执行器
  - real-time
    - 不支持跨节点join
    - 但几乎支持各种聚合函数
  - Task Tracker Executor
    - 支持跨节点join
- 由参数citus.task_executor_type指定
  - 可以在线修改

citus查询的处理

- 查询的支持情况
  - 支持各种聚合函数
    - Count (Distinct)： **支持**HyperLogLog algorithm 得到一个近似值，此近似值的错误率可以小于参数citus.count_distinct_error_rate指定的值
  - Limit Pushdown
    - 可以下推
    - 支持加限制，不过这样的结果是近似的
      - SET citus.limit_clause_row_fetch_count to 10000;

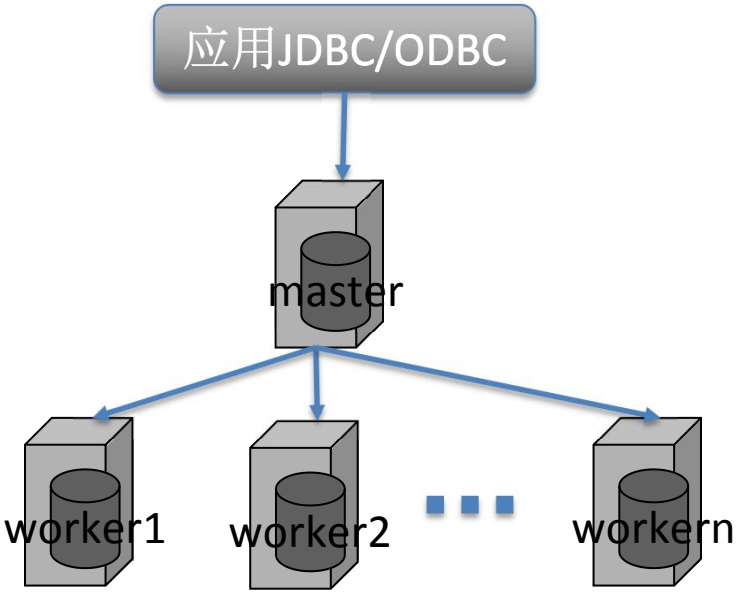
citus查询的处理

- 查询的支持情况（续）
  - Join
    - 只支持等值join
    - 支持加限制，不过这样的结果是近似的
    - Broadcast joins
    - Co-located joins
    - Repartition joins

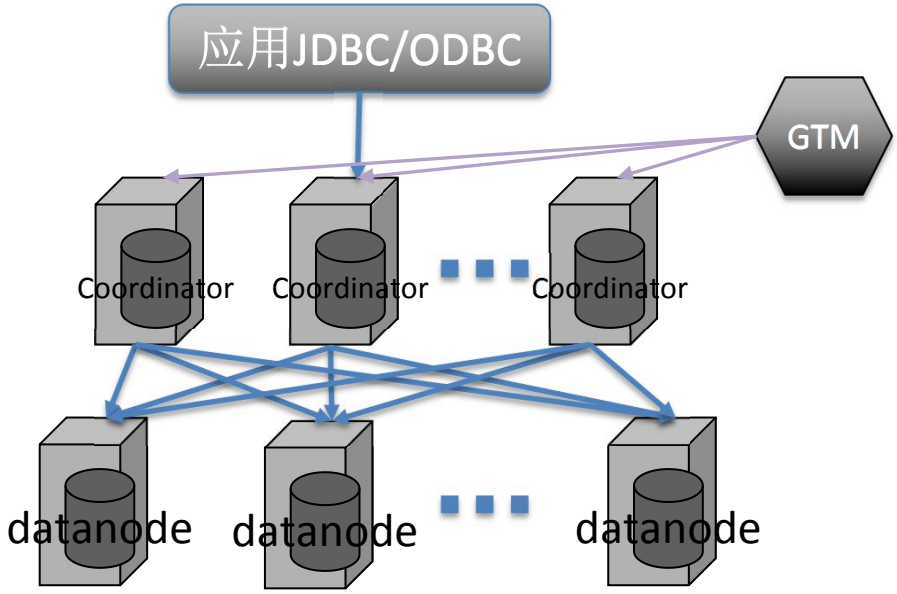
目录

架构对比

citus



Postgres-x2



开源的citus不太能支持多master，而Postgres-X2支持多个Coordinator

分布式事务

- citus不支持分布式事务
  - 如果所有更新都是在一个分片中，是可以使用事务的。
- Postgres-X2支持分布式事务
  - 因为有GTM，所以可以支持分布式事务
  - 但这会导致性能低

- Postgres-X2几乎拥有与单机数据库的所有功能
  - 支持复杂的SQL和跨节点JOIN
  - 全局事务的强一致性
  - 支持Read commited事务隔离级别
  - 几乎支持所有单机数据库的DDL语句
  - 支持跨节点的视图
  - 支持跨节点的存储过程
- citus：SQL有较多的限制

高可用对比

- Postgres-X2本身不提供高可用
  - 需要自己把datanode通过standby的方式做高可用
- citus
  - 数据可以做多副本，当一个副本出现问题时，可以自动高可用

总体的对比

- citus的插件的好外
  - 是数据库的一个插件，可以轻松跟着PostgreSQL的版本升级而升级
  - 而Postgres-X2是通过改造PostgreSQL代码而形成的独立的一套软件，当PostgreSQL新版本出来后，很难跟随升级
- 运维
  - Postgres-X2复杂，运维难度大，但功能强大
  - citus相对简单一些

Thanks!

Q & A