The Future of Postgres Sharding

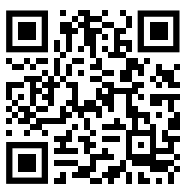
未来的Postgres分片

Bruce Momjian

This presentation will cover the advantages of sharding and future Postgres sharding implementation requirements.

本演示将探讨分片的优点以及未来的Postgres分片实现需求

https://momjian.us/presentations



Translated by: China PostgreSQL Association

由中国PostgreSQL分会协助翻译

Last updated: April, 2020

最后更新时间：2020年4月

Outline 议程

1. Scaling 扩展
2. Vertical scaling options 垂直扩展选项
3. Non-sharding horizontal scaling 非分片水平扩展
4. Existing sharding options 现有的分片选项
5. Built-in sharding accomplishments 内置的分片功能
6. Future sharding requirements 未来的分片需求

1. Scaling 扩展

Database scaling is the ability to increase database throughput by utilizing additional resources such as I/O, memory, CPU, or additional computers.
However, the high concurrency and write requirements of database servers make scaling a challenge. Sometimes scaling is only possible with multiple sessions, while other options require data model adjustments or server configuration changes.
Postgres Scaling Opportunities https://momjian.us/main/presentations/performance.html#scaling

数据库扩展是通过利用其他资源（例如I / O，内存，CPU或其他计算机）来提高数据库吞吐量的能力。
但是，数据库服务器的高并发和写入要求使扩展成为一个挑战。有时，只有在多个会话中才能进行扩展，而其他选项则需要调整数据模型或更改服务器配置。
Postgres扩展机会 *https://momjian.us/main/presentations/performance.html#scaling*

2. Vertical Scaling 垂直扩展

Vertical scaling can improve performance on a single server by:
垂直扩展可通过以下方式提高单个服务器的性能：

- Increasing I/O with 通过下面方式提高I/O
  - faster storage 更快的存储
  - tablespaces on storage devices 存储设备上的表空间
  - striping (RAID 0) across storage devices 跨存储设备进行条带化（RAID 0）
  - Moving WAL to separate storage 将WAL移至单独的存储
- Adding memory to reduce read I/O requirements
  增加内存以减少I/O读取
- Adding more and faster CPUs
  增加更多更快的CPU

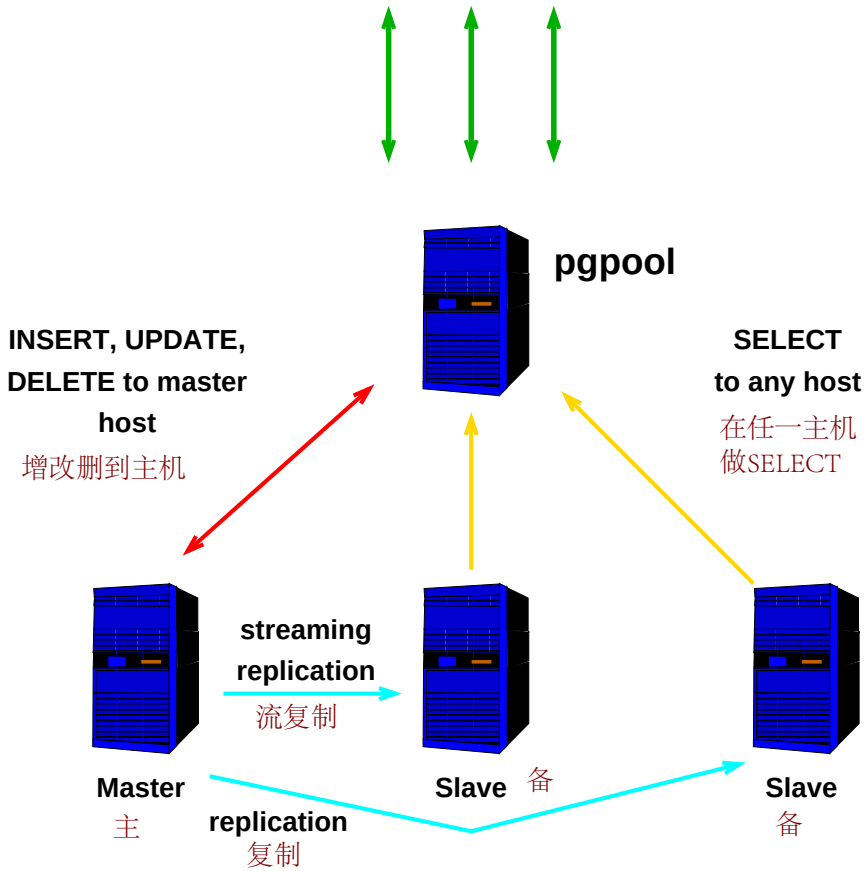
3. Non-Sharding Horizontal Scaling
非分片水平扩展

Non-sharding horizontal scaling options include:
非分片水平扩展选项包括:

- Read scaling using Pgpool and streaming replication
  使用Pgpool和流复制进行读取扩展
- CPU/memory scaling with asynchronous multi-master
  异步多主的CPU /内存扩展

The entire data set is stored on each server.
整个数据集存储在每个服务器上。

Pgpool II With Streaming Replication
具有流复制的Pgpool II



Streaming replication avoids the problem of non-deterministic queries producing different results on different hosts.

流复制避免了不确定查询在不同主机上产生不同结果的问题。

Why Use Sharding 为什么要使用分片?

- Only sharding can reduce I/O, by splitting data across servers
  在服务器之间拆分数据，只有分片才能减少I/O
- Sharding benefits are only possible with a shardable workload
  分片带来的好处只有在可分片的工作负载下才有可能
- The shard key should be one that evenly spreads the data
  分片密钥应该是均匀分布数据的密钥
- Changing the sharding layout can cause downtime
  更改分片布局可能会导致停机
- Additional hosts reduce reliability; additional standby servers might be required
  附加主机会降低可靠性；可能需要其他备用服务器

Typical Sharding Criteria 典型分片标准

- List 列表
- Range 范围
- Hash 哈希

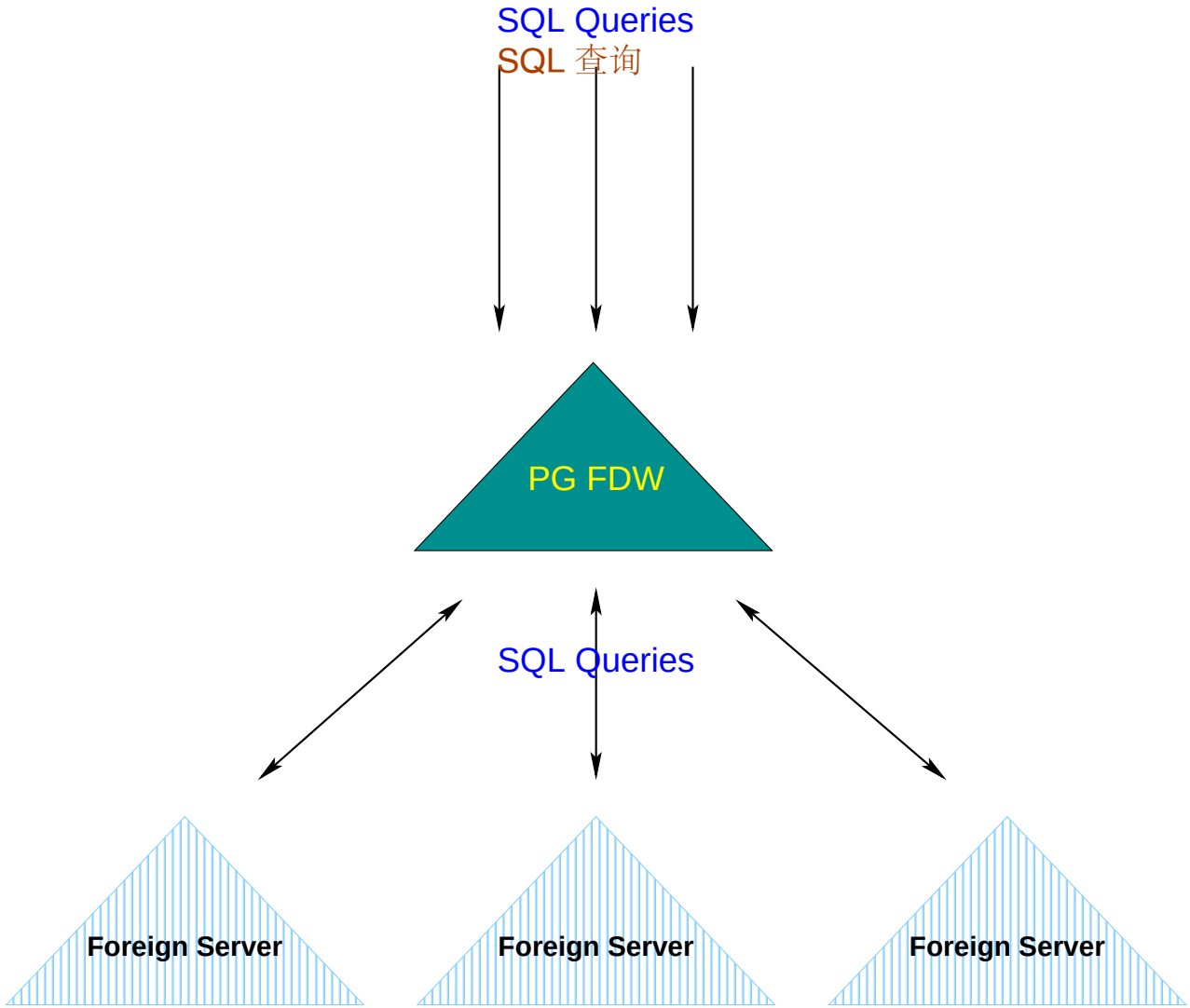
4. Existing Sharding Solutions
现有的分片方案

- Application-based sharding基于应用的数据分片
- PL/Proxy PL/代理
- Postgres-XC/XL
- Citus
- Hadoop

The data set is sharded (striped) across servers.
数据集被分片（分割）到不同的服务器

5. Built-in Sharding Accomplishments: Sharding Using Foreign Data Wrappers (FDW)

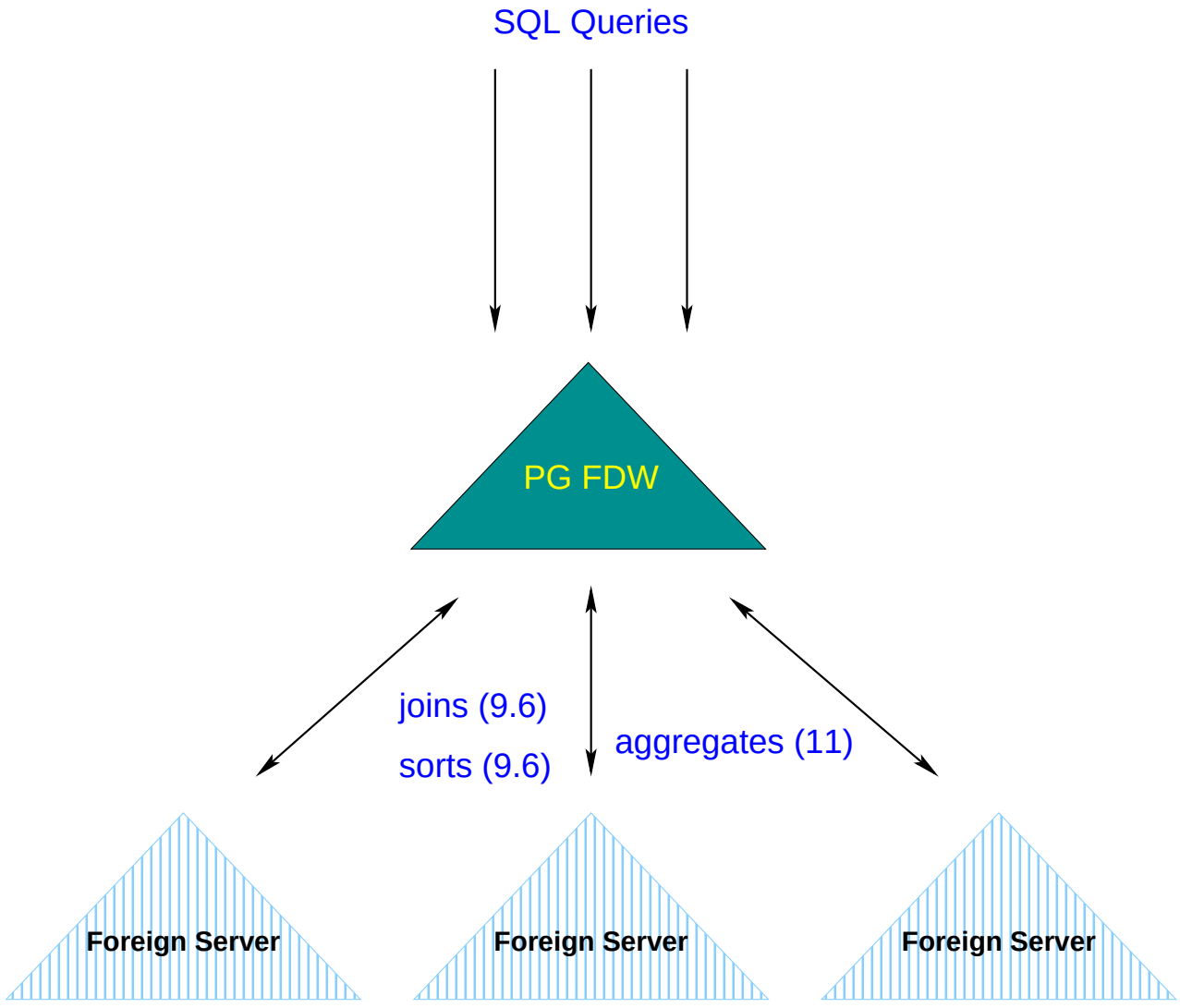
内置数据分片的实现：使用外部数据包装插件（FDW）进行数据分片



https://wiki.postgresql.org/wiki/Built-in_Sharding

FDW Sort/Join/Aggregate Pushdown

FDW 排序/合并/汇总下推



Advantages of FDW Sort/Join/Aggregate Pushdown

采用FDW排序/合并/汇总下推方案的优点

- Sort pushdown reduces CPU and memory overhead on the coordinator

  排序下推可以减少协调器对CPU的使用和内存开销
- Join pushdown reduces coordinator join overhead, and reduces the number of rows transferred

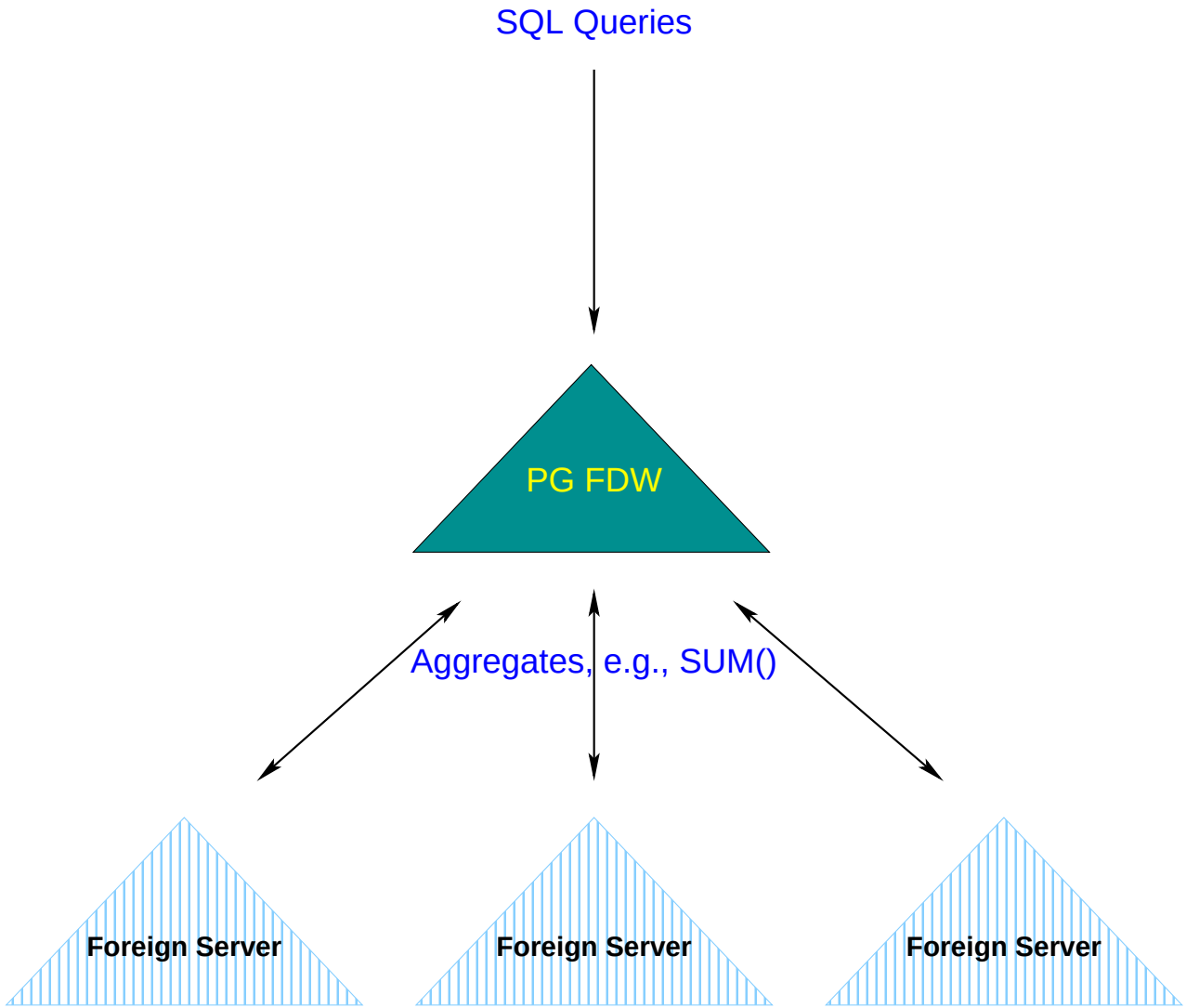
  合并下推减少协调器对合并的开销，并减少需要传输的行数
- Aggregate pushdown causes summarized values to be passed back from the shards

  汇总下推可以把结果回传给分片服务器
- WHERE clause restrictions are also pushed down

  WHERE条件限制也被下推到分片服务器

Aggregate Pushdown in Postgres 11

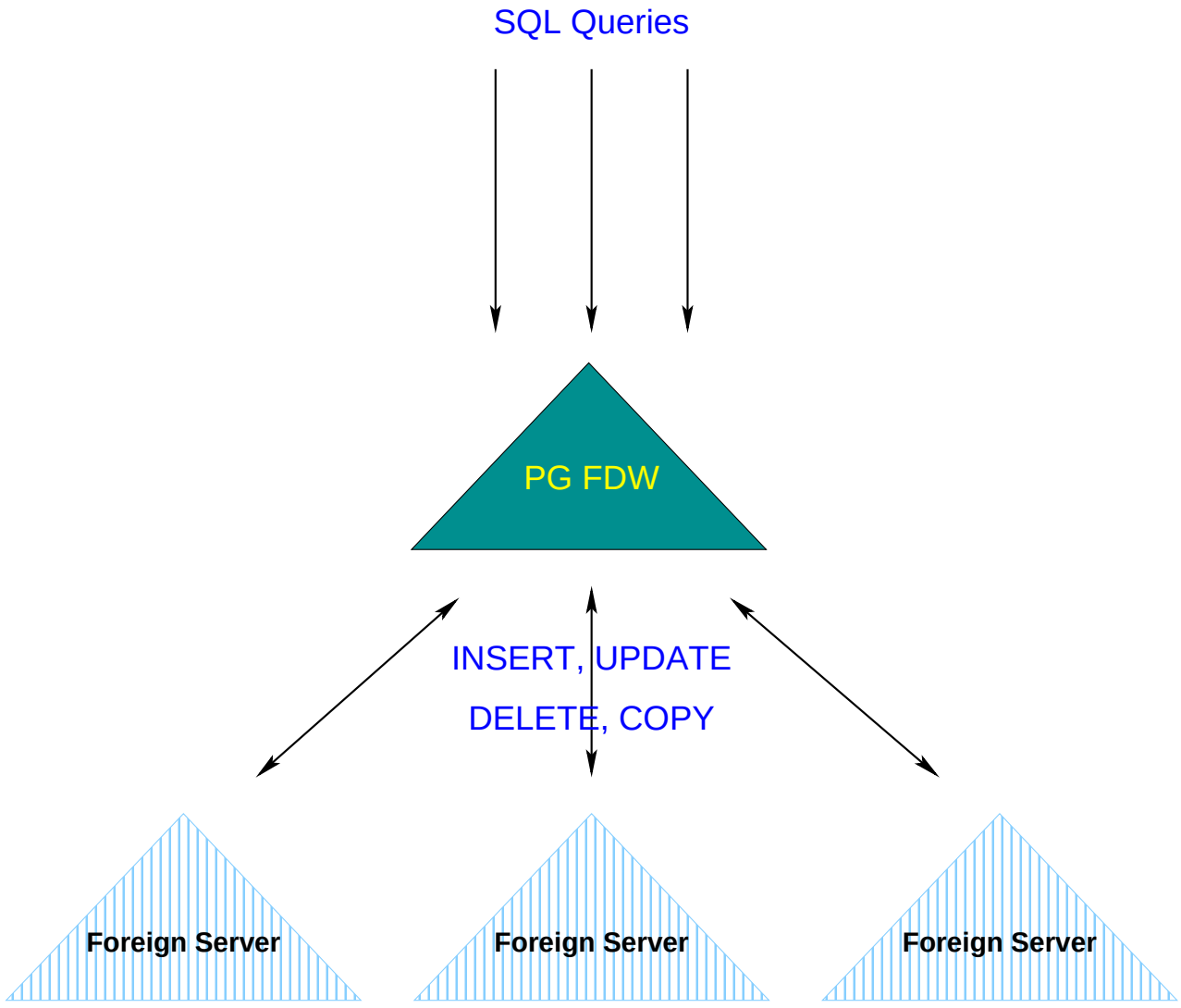
Postgres 11中的汇总下推



Unfortunately, aggregates are currently evaluated one partition at a time, i.e., serially. 不幸的是，目前的合并一次只能针对一个分区进行，即，串行的

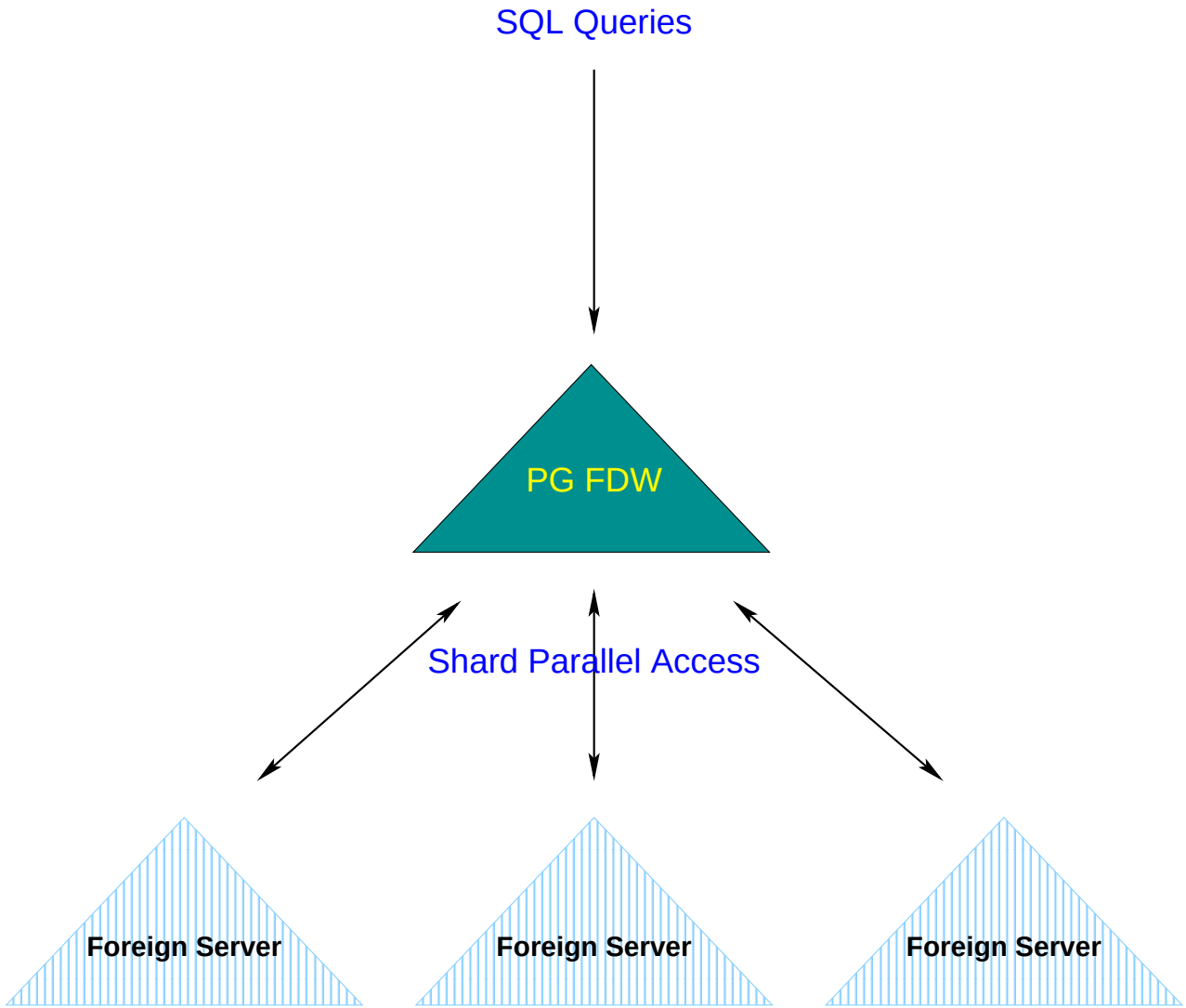
FDW DML Pushdown in Postgres 9.6 & 11

在Postgres 9.6和11中的FDW DML(数据操作语言)下推



6. Future Sharding Requirements: Parallel Shard Access

未来对数据分片的需求：并行数据分片访问



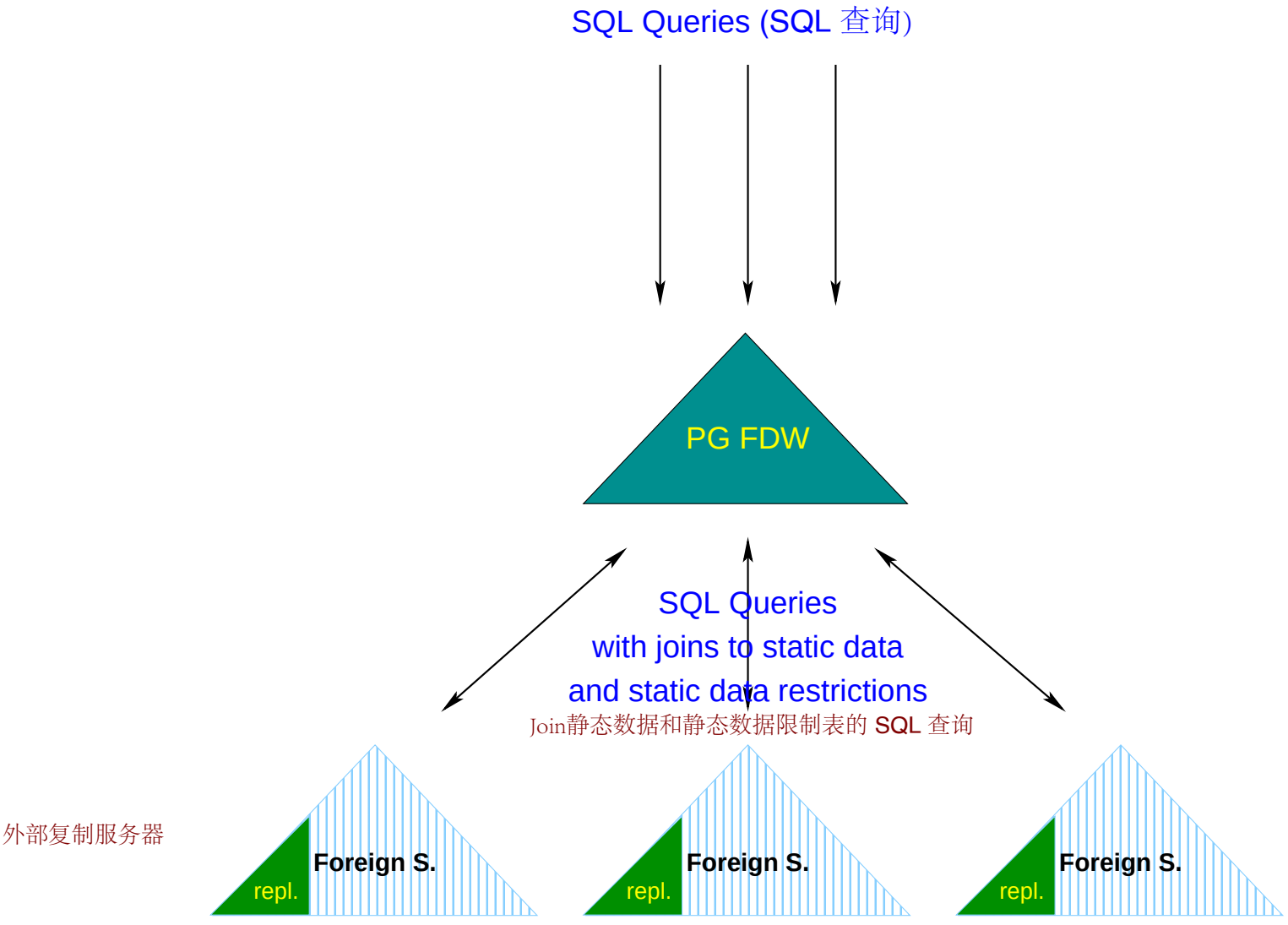
Parallel shard access is waiting for an executor rewrite, which is necessary for JIT improvements.

并行数据分片访问需要重新写执行器的代码，这对JIT（即时编译）改进也是必要的

Advantages of Parallel Shard Access
并行数据分片访问的优点

- Can use libpq's asynchronous API to issue multiple pending queries 可以使用libpq库的异步API执行多个需要等待的查询
- Ideal for queries that must run on every shard, e.g.,
  非常适合那些必须在每个分片服务器上运行的查询
  - restrictions on static tables 对静态数据表的限制
  - queries with no sharded-key reference 执行没有分片字段参考的查询
  - queries with multiple shared-key references 执行多分片字段参考的查询
- Parallel aggregation across shards 跨分片服务器数据合并

Joins With Replicated Tables 复制表的Join



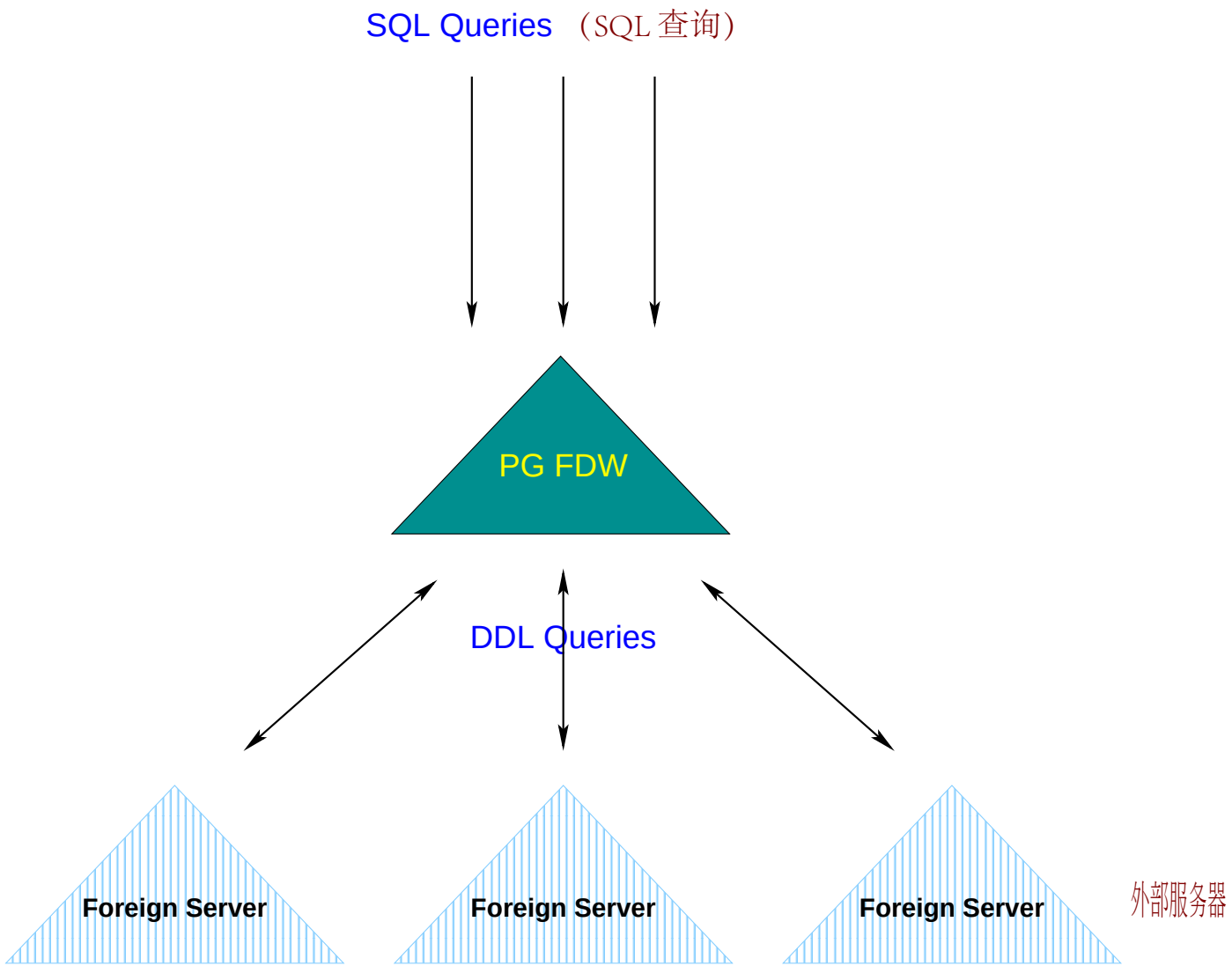
Implementing Joins With Replicated Tables

实现复制表的Join

Joins with replicated tables allow join pushdown where the query restriction is on the replicated (lookup) table and not on the sharded column. Tables can be replicated to shards using logical replication. The optimizer must be able to adjust join pushdown based on which tables are replicated on the shards.

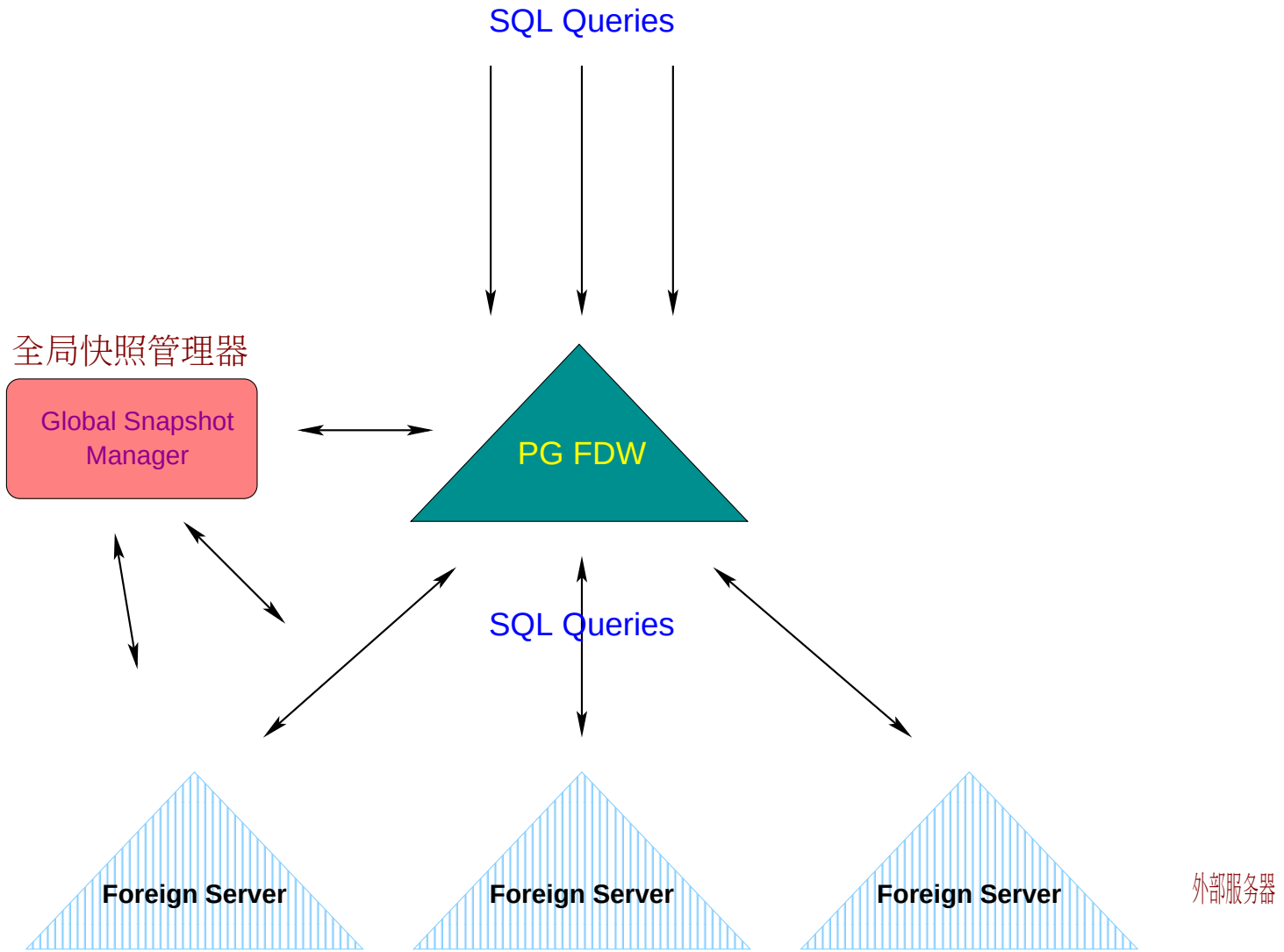
具有复制表的联接(JOIN)允许在查询中进行联接下推 (Join Pushdown)并
限制在复制（查找）表上，而不在分片字段上。可以使用逻辑复制将表复制到分片上。
优化器模块 （Optimizer） 必须能够根据复制到分片上的表来调整连接下推(Join Pushdown)

Shard Management 分片管理



Shard management will be added to the existing partitioning syntax, which was added in Postgres 10. 分片管理将添加到现有的分区语法，已加入PG 10版本

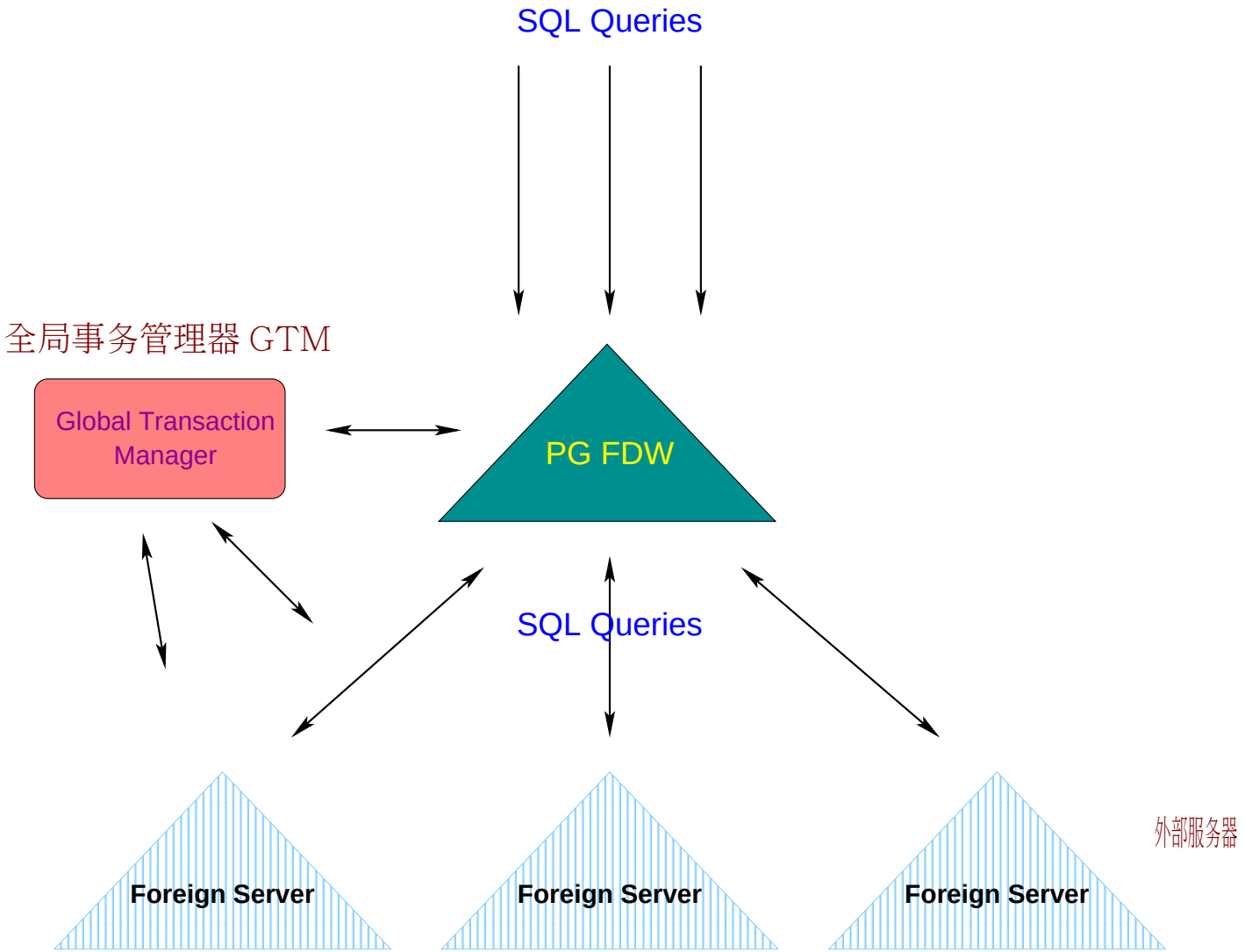
Global Snapshot Manager 全局快照管理器



Implementing a Global Snapshot Manager
实现全局快照管理器

- We already support sharing snapshots among clients with `pg_export_snapshot()`
  我们已经支持利用 pg_export_snapshot（） 函数调用来实现客户端间的快照共享
- We already support exporting snapshots to other servers with the GUC `hot_standby_feedback`
  我们已经支持利用 GUC参数 hot_standby_feedback 来实现快照导出到其他服务器

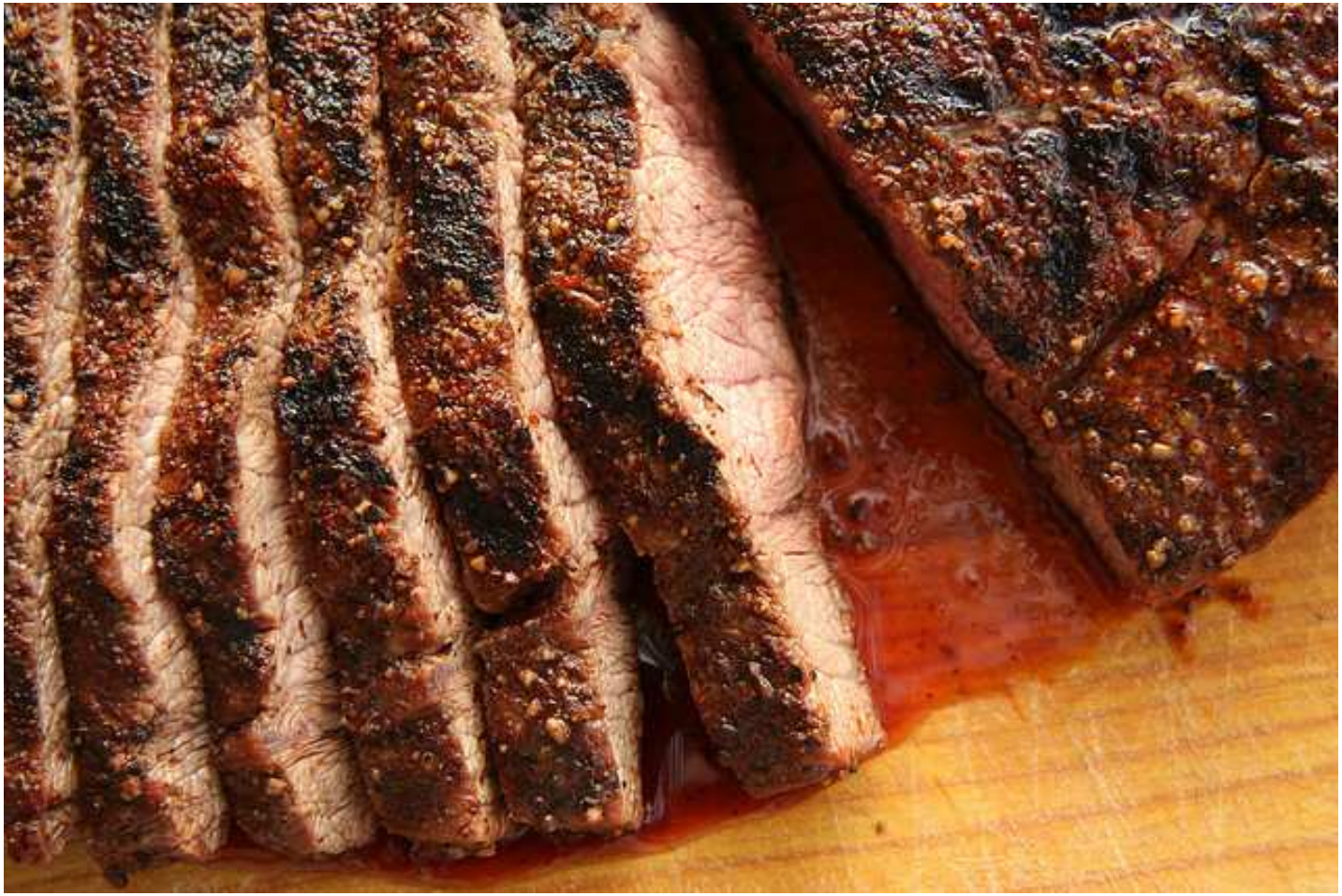
Global Transaction Manager 全局事务管理器GTM



Implementing a Global Transaction Manager
实现一个全局事务管理器GTM

- Can use prepared transactions (two-phase commit)
  使用prepared transaction （两阶段事务提交）
- Transaction manager can be internal or external
  事务管理器可以是内部或外部的
- Can use an industry-standard protocol like XA
  可以使用类似XA的行业标准协议

Conclusion 总结



https://momjian.us/presentations

https://www.flickr.com/photos/anotherpintplease/