The Future of Postgres Sharding

Bruce Momjian

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

 http://momjian.us/presentations

Last updated: May, 2017

Outline

1. Scaling
2. Vertical scaling options
3. Non-sharding horizontal scaling
4. Existing sharding options
5. Future sharding

Scaling

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 http://momjian.us/main/presentations/overview.html#scaling

Vertical Scaling

Vertical scaling can improve performance on a single server by:

- Increasing I/O with
  - faster storage
  - tablespaces on storage devices
  - striping (RAID 0) across storage devices
  - Moving WAL to separate storage
- Adding memory to reduce read I/O requirements
- Adding more and faster CPUs

Non-sharding Horizontal Scaling

Non-sharding horizontal scaling options include:

- Read scaling using Pgpool and streaming replication
- CPU/memory scaling with asynchronous multi-master

The entire data set is stored on each server.

Why Use Sharding?

- Only sharding can reduce I/O, by splitting data across servers
- Sharding benefits are only possible with a shardable workload
- The shard key should be one that evenly spreads the data
- Changing the sharding layout can cause downtime
- Additional hosts reduce reliability; additional standby servers might be required

Typical Sharding Criteria

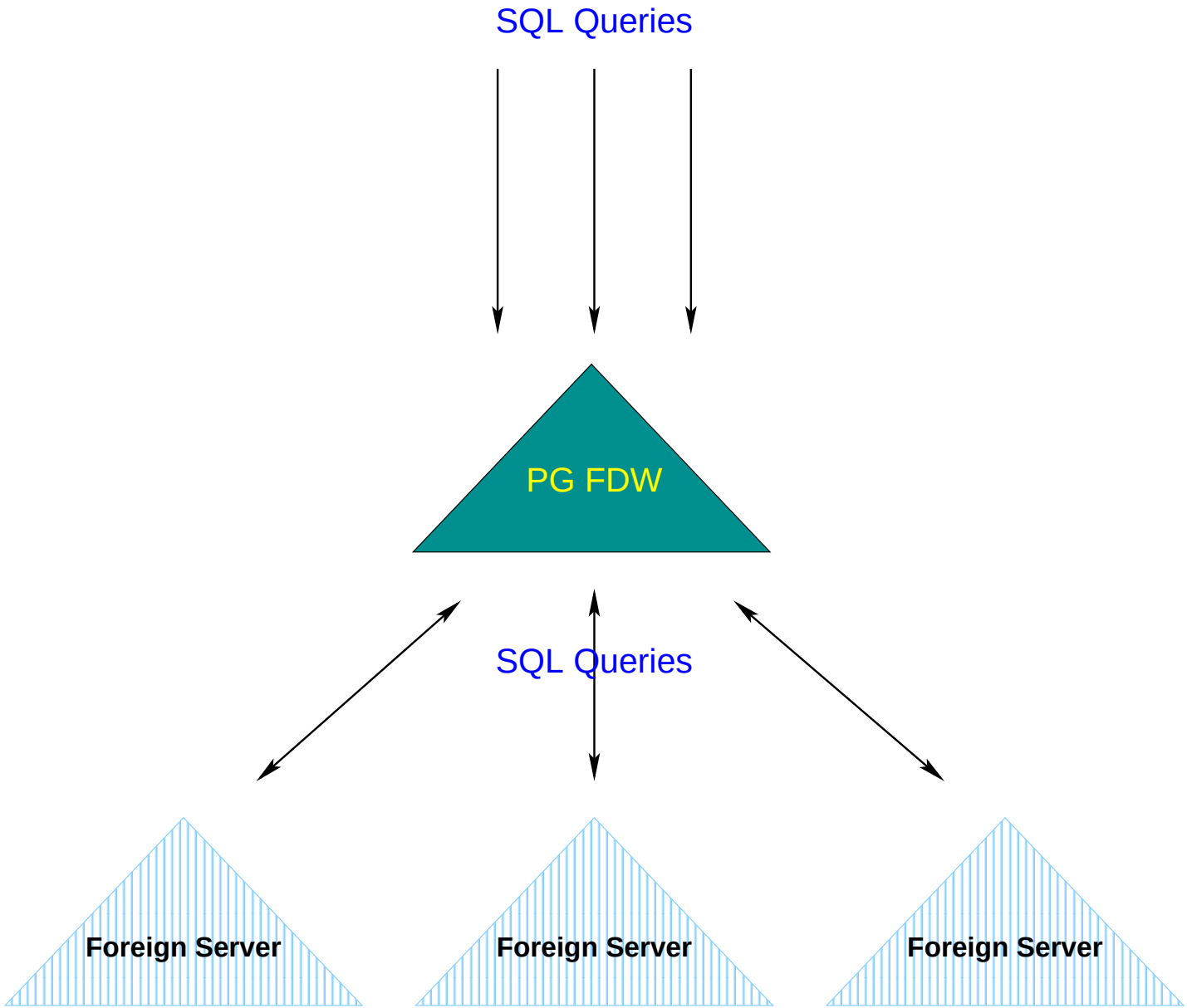
- List
- Range
- Hash

Existing Sharding Solutions

- Application-based sharding
- PL/Proxy
- Postgres-XC/XL
- pg_shard
- Hadoop

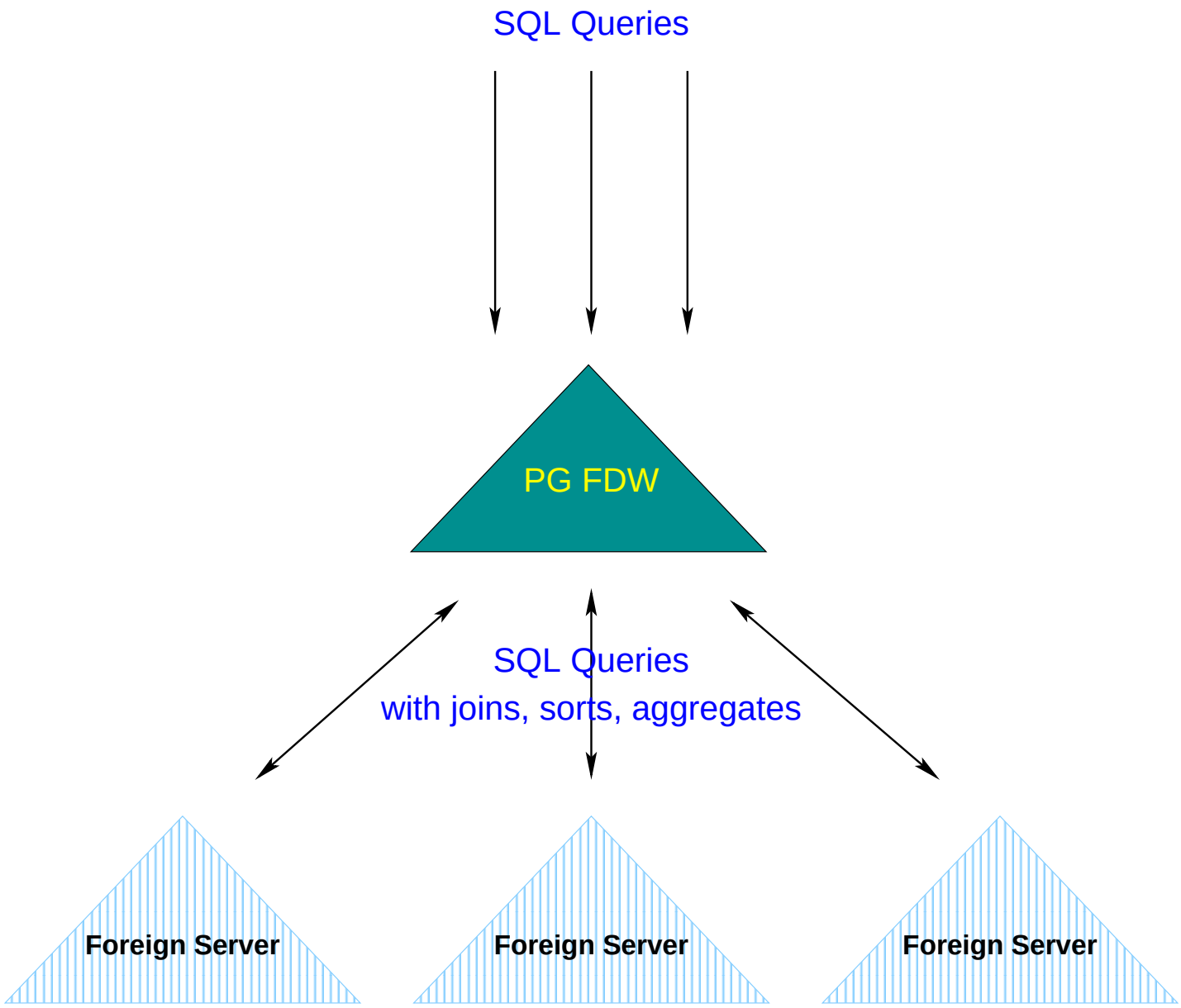
The data set is sharded (striped) across servers.

Sharding Using Foreign Data Wrappers (FDW)



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

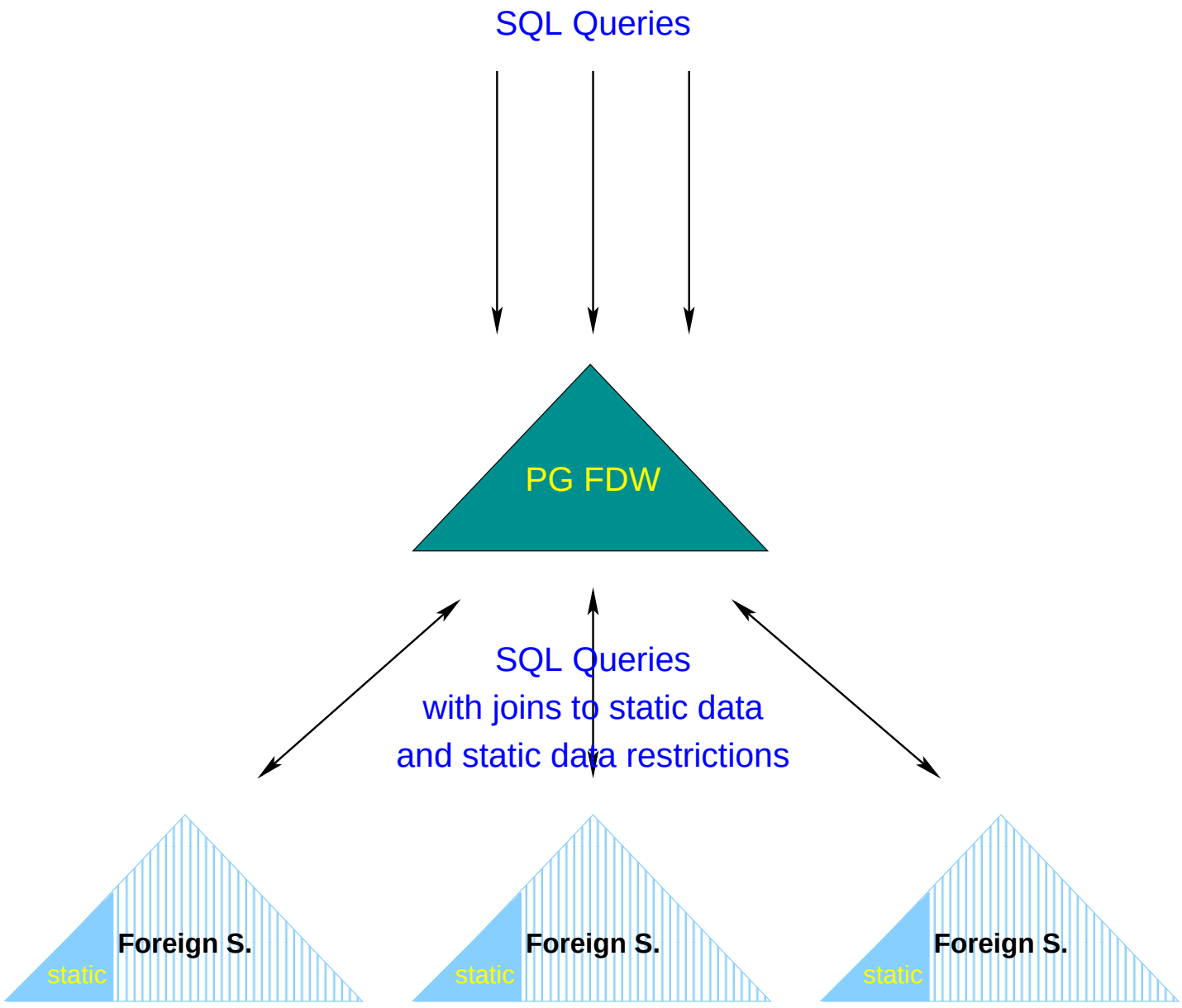
FDW Sort/Join/Aggregate Pushdown



Advantages of FDW Sort/Join/Aggregate Pushdown

- Sort pushdown reduces CPU and memory overhead on the coordinator
- Join pushdown reduces coordinator join overhead, and reduces the number of rows transferred
- Aggregate pushdown causes summarized values to be passed back from the shards
- WHERE clause restrictions are already pushed down

Pushdown of Static Tables



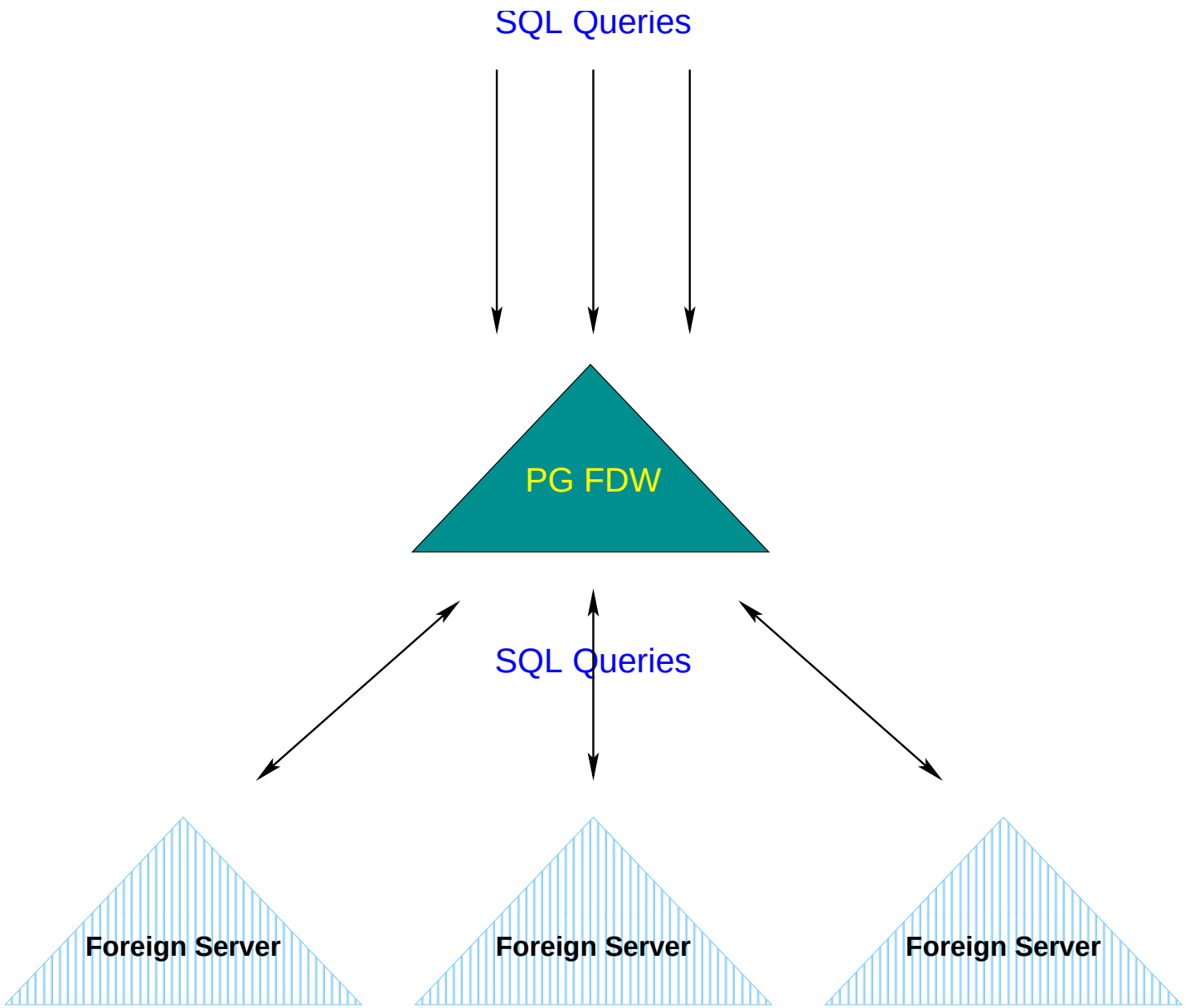
Implementing Pushdown of Static Tables

Static table pushdown allows join pushdown where the query restriction is on the static table and not on the shared key. Static tables can be pushed to shards using:

- Triggers with Slony-like distribution
- WAL logical decoding

The optimizer must also know which tables are duplicated on the shards for proper join pushdown.

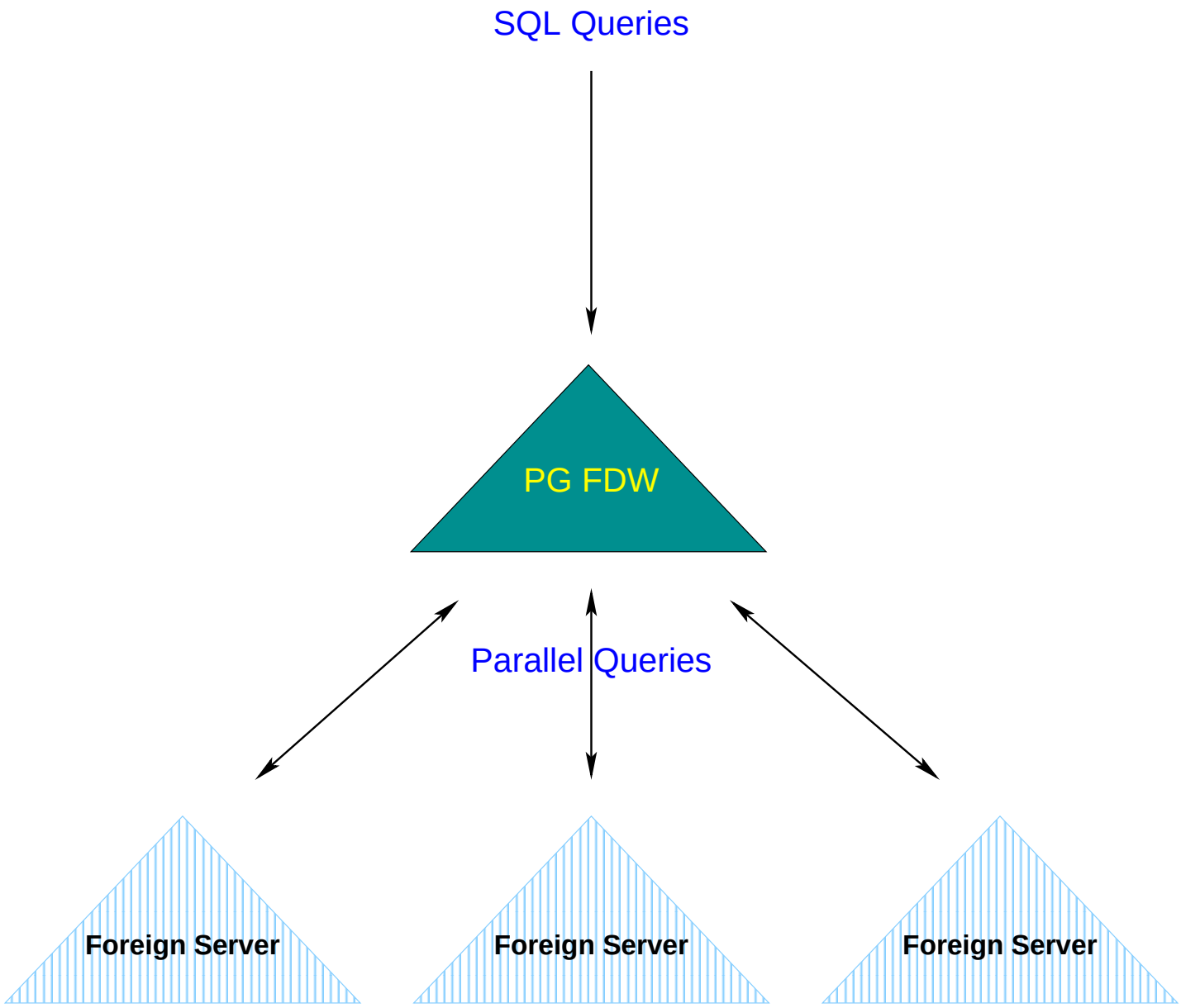
Query Routing and DDL to Proper Shards



Implementing Query Routing and DDL to Proper Shards

- Could use the existing partitioning system
  - inheritance
  - CHECK constraint
  - constraint exclusion
  - trigger for INSERT, UPDATE, and DELETE

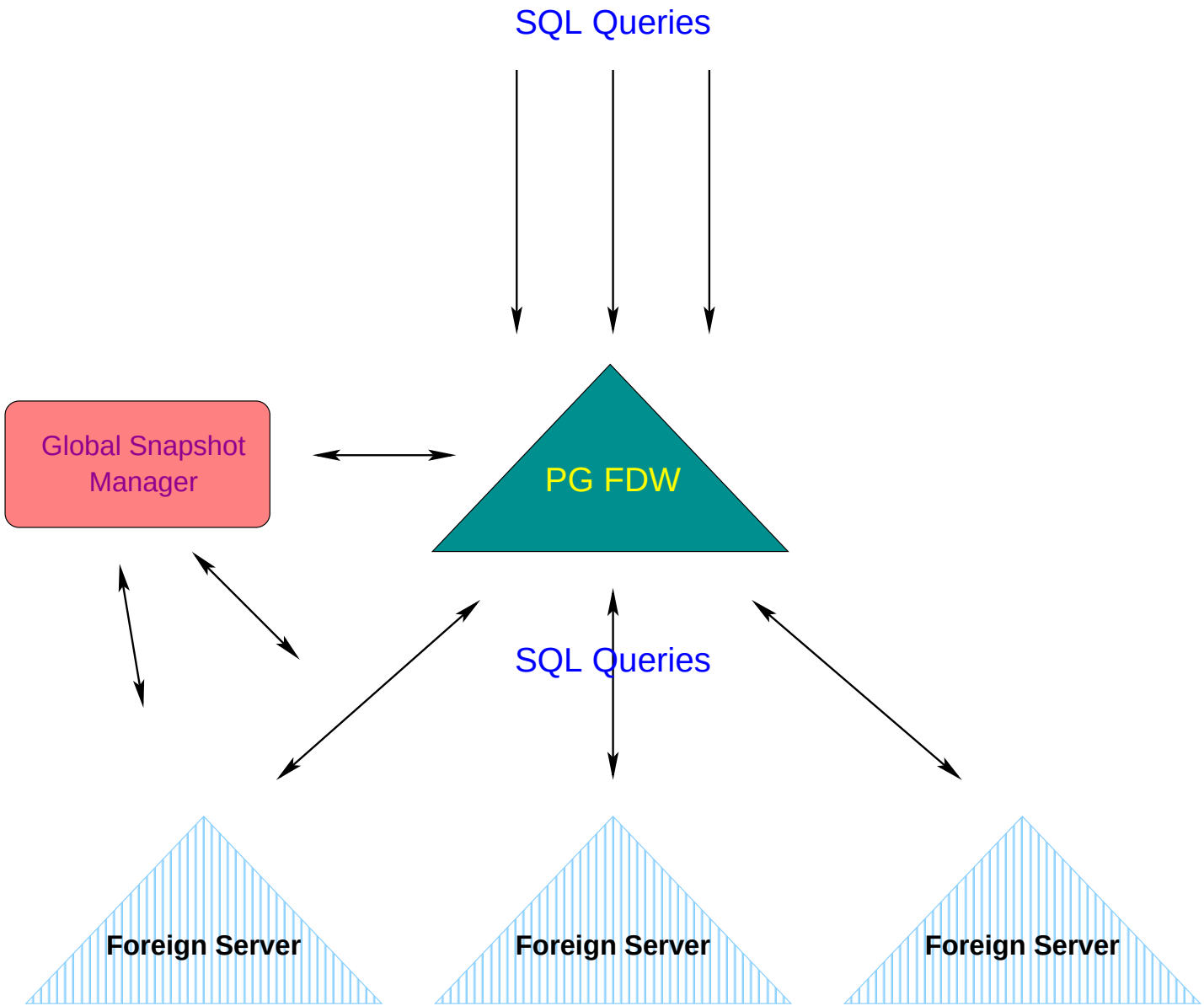
Parallel FDW Access



Advantages of Parallel FDW Access

- Can use libpq's asynchronous API to issue multiple pending queries
- Ideal for queries that must run on every shard, e.g.
  - restrictions on static tables
  - queries with no sharded-key reference
  - queries with multiple shared-key references
- Aggregates like AVG() must be computed on the controller by having shards return SUM() and COUNT()

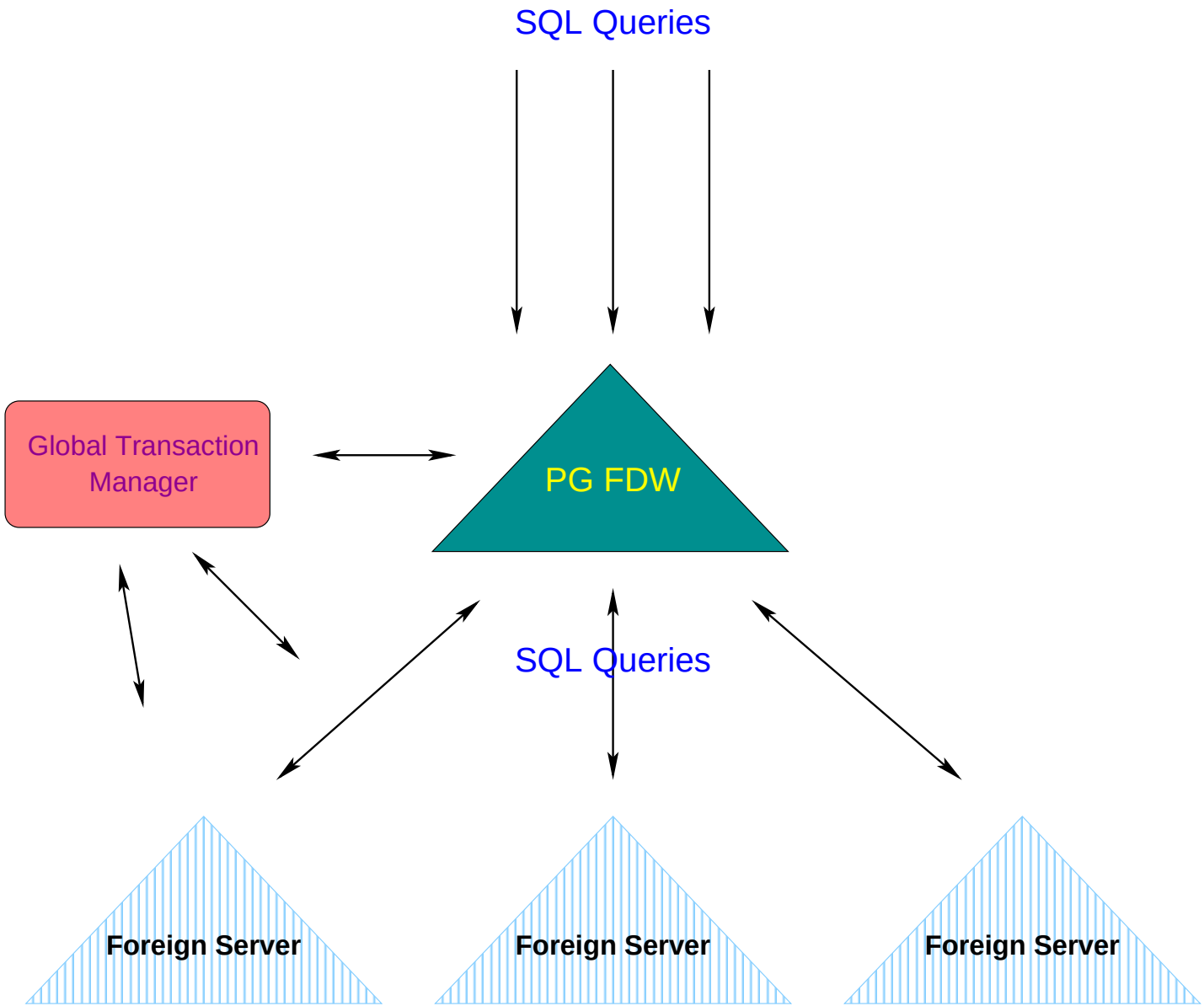
Global Snapshot Manager



Implementing a Global Snapshot Manager

- We already support sharing snapshots among clients with `pg_export_snapshot()`
- We already support exporting snapshots to other servers with the GUC `hot_standby_feedback`

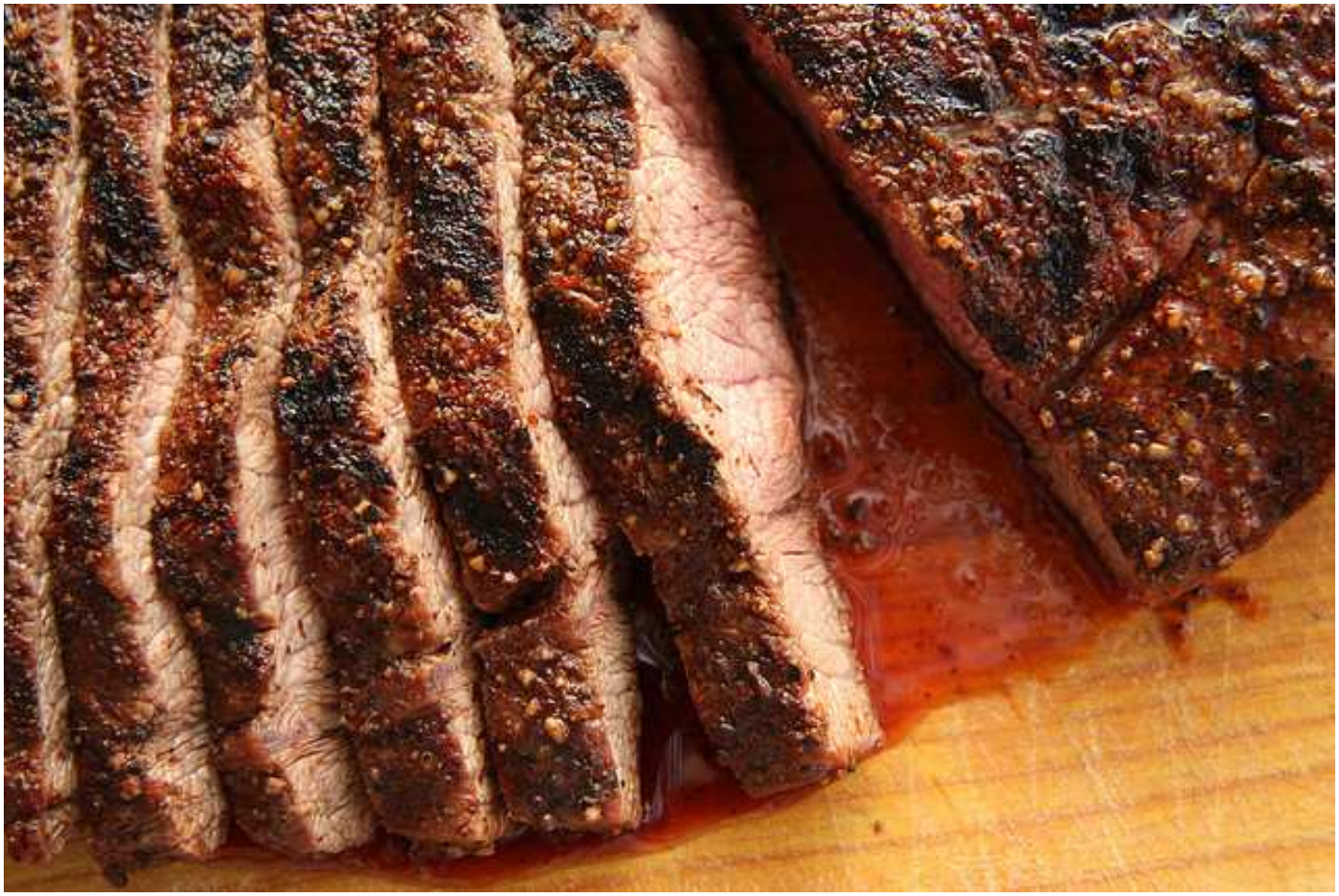
Global Transaction Manager



Implementing a Global Transaction Manager

- Can use prepared transactions (two-phase commit)
- Transaction manager can be internal or external
- Can use an industry-standard protocol like XA

Conclusion



http://momjian.us/presentations

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