Microsoft

Moscow, 2020-02-05

Architecting petabyte-scale analytics by scaling out Postgres on Azure with Citus

Alicja Kucharczyk
EMEA Global Blackbelt OSS Data Tech Specialist

Nothing Compares To VACUUM/The Ballad of Bloat

1406 wyświetleń • Data premiery: 21 sty 2020

Warsaw PostgreSQL Users Group

34 subskrypcje

Nothing Compares To VACUUM/The Ballad of Bloat

Questions?

Why am I here?

Agenda

- What?
- Why?
- Where?

The naming thing

Hyperscale

(Citus)

Hyperscale (Citus)

- Open source extension
- Pure Postgres, not a fork
- Turns Postgres into distributed, sharded database
- All the benefits of Postgres, without worry about scale

Hyperscale (Citus)

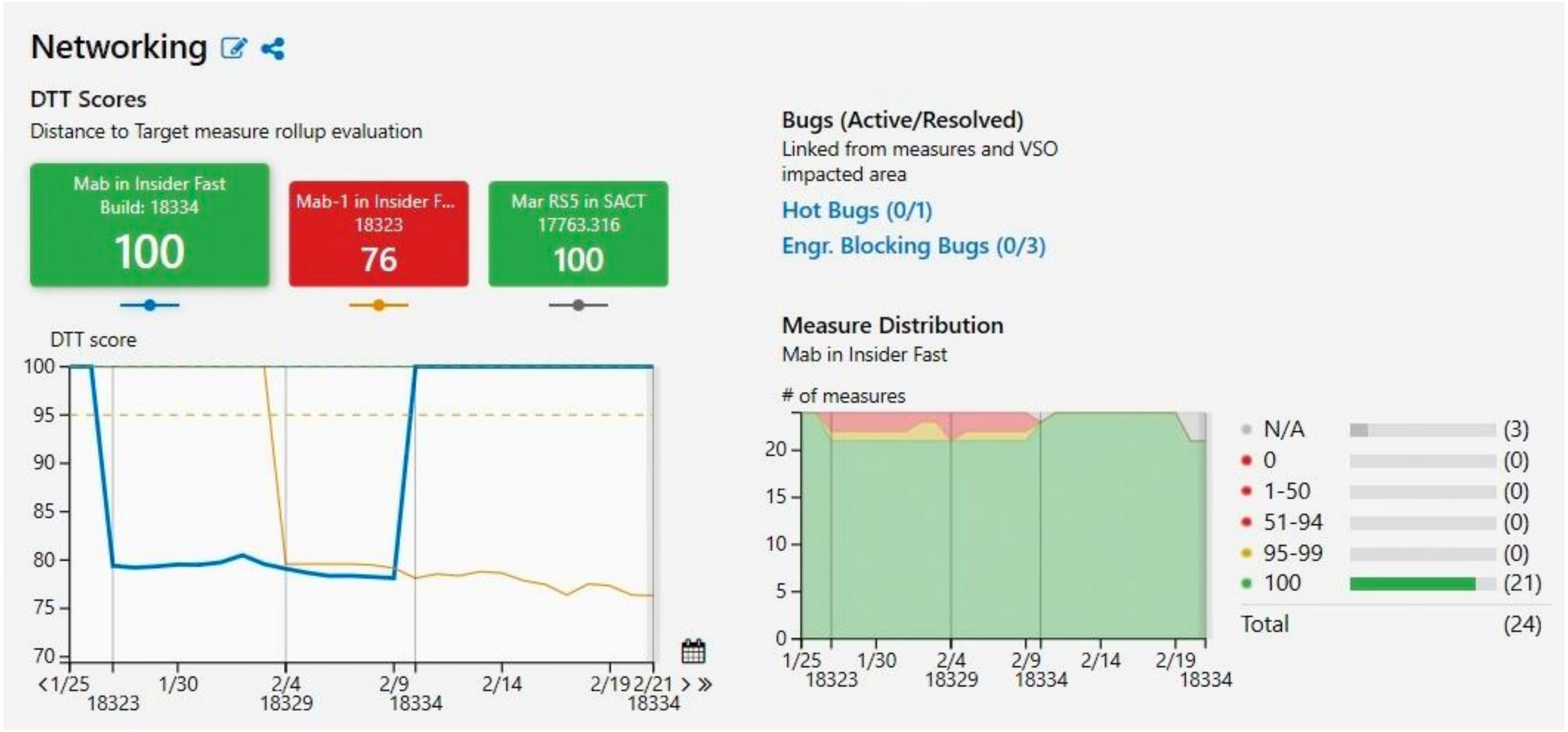
- Open source extension
- Pure Postgres, not a fork
- Turns Postgres into distributed, sharded database
- All the benefits of Postgres, without worry about scale

Why I like Hyperscale (Citus)?

Why Microsoft likes Hyperscale (Citus)?

How do you know if the next update to your software is ready for hundreds of millions of customers?

Internal RQV analytics dashboard



Networking

DTT Scores

Distance to Target measure rollup evaluation

Mab in Insider Fast Build: 18334	Mab-1 in Insider F... 18323	Mar RS5 in SACT 17763.316
100	76	100

Bugs (Active/Resolved)

Linked from measures and VSO impacted area

Hot Bugs (0/1)

Engr. Blocking Bugs (0/3)

Measure Distribution

Mab in Insider Fast

● N/A	(3)
● 0	(0)
● 1-50	(0)
● 51-94	(0)
● 95-99	(0)
● 100	(21)
Total	(24)

RQV analytics dashboard is a critical tool for Windows engineers, program managers, and execs.

The short story

Min Wei, Principal Engineer at Microsoft

↓

discovered the open source Citus extension to Postgres by listening to a recorded conference talk on his drive home

↓

Impressed with the early results, he transitioned the project from a proof of concept into an official project.

↓

A few months later Microsoft had acquired Citus Data.

Measuring the quality of Windows

- “Release Quality View” (RQV) dashboard
- tracks 20,000 diagnostic and quality metrics
- over 800M unique devices monthly
- supports over 6 million queries per day
- hundreds of concurrent users
- 1000s of monthly active users
- 100s of dashboard pages

Production database cluster

2816 Cores, 18TB DRAM,

1PB Azure Premium Storage,

Multi-PB Azure Blob Storage - for the staging queue and raw Windows event data

- 2 Physical clusters behind a query router (Azure Web Service and Azure Redis Service)
- Ingest and delete ~5TB data per day
- P75 query latency ~90ms/200ms (response times for 75 percent of queries are less than 200 milliseconds)
- Support long running queries up to 4 mins.
- Support batch scheduled jobs that can run up for 2hours

Run Anywhere

On-Premises

In the Cloud - Azure Database for PostgreSQL

Azure Database for PostgreSQL is available in two deployment options

Enterprise-ready, fully managed community PostgreSQL with built-in HA and multi-layered security

Single Server

Fully-managed, single-node PostgreSQL

Example use cases

- Apps with JSON, geospatial support, or full-text search
- Transactional and operational analytics workloads
- Cloud-native apps built with modern frameworks

Hyperscale (Citus)

High-performance Postgres for scale out

Example use cases

- Scaling PostgreSQL multi-tenant, SaaS apps
- Real-time operational analytics
- Building high throughput transactional apps

We're talking about Hyperscale (Citus) today

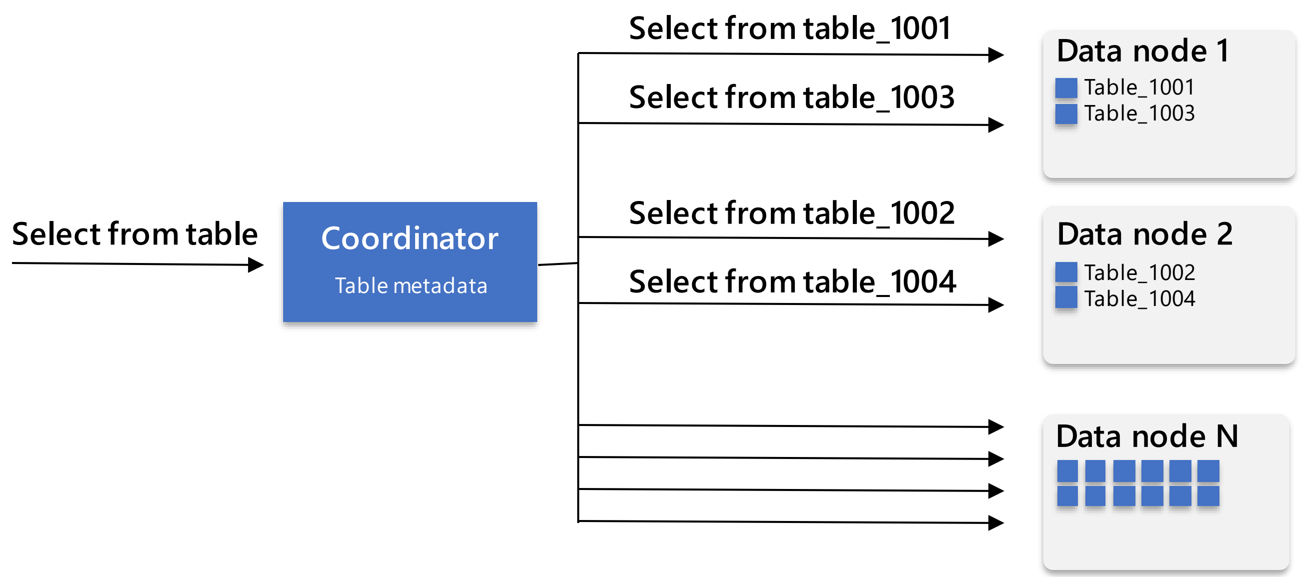
Scale horizontally across hundreds of cores with Hyperscale (Citus)

Shard your Postgres database across multiple nodes to give your application more memory, compute, and disk storage

Easily add worker nodes to achieve horizontal scale

Scale up to 100s of nodes

Sharding data across multiple nodes



Each node PostgreSQL with Citus installed

1 shard = 1 PostgreSQL table

Terminology

Coordinator – Stores Metadata. Node which application connects to.

Worker / Data nodes – Nodes which store data in form of shards.

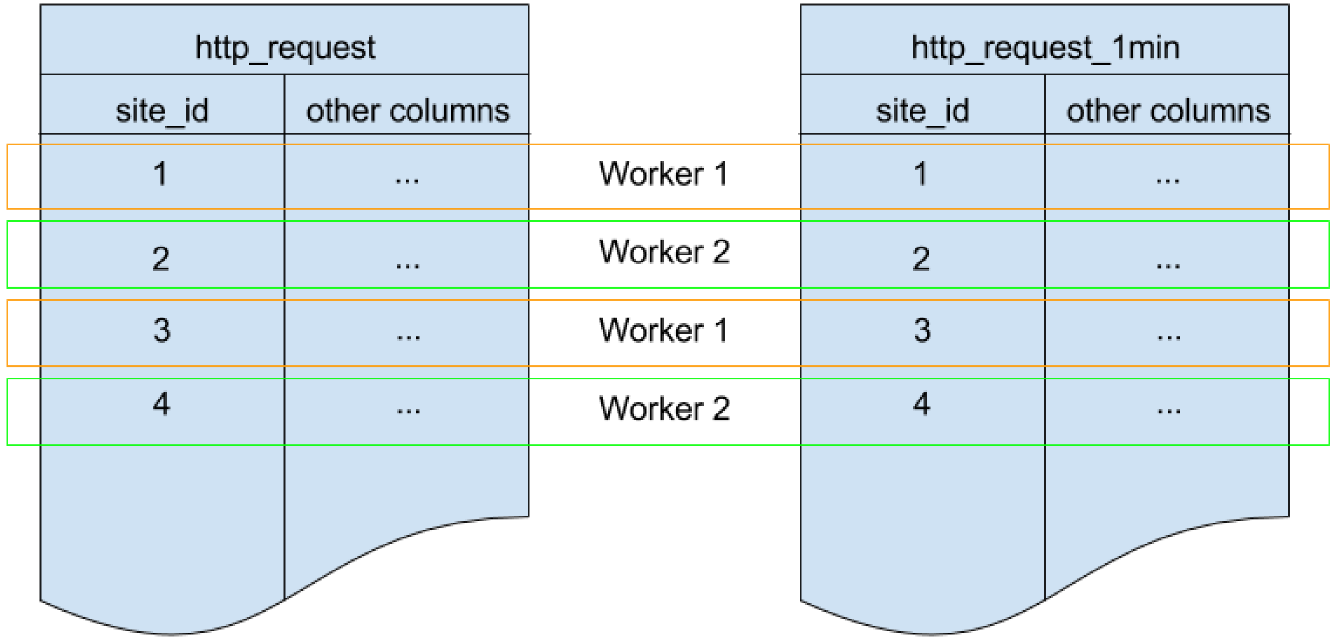
Sharding – Process of dividing data among nodes.

Shards – A partition of the data containing a subset of rows.

Co-location

- Tables sharded on the same distribution column are co-located.

Shards of both tables holding the same set of distribution column values are on the same worker.



http_request			http_request_1min	
site_id	other columns		site_id	other columns
1	...	Worker 1	1	...
2	...	Worker 2	2	...
3	...	Worker 1	3	...
4	...	Worker 2	4	...

Co-location based on data-type of the distribution column. Not the name of the column.

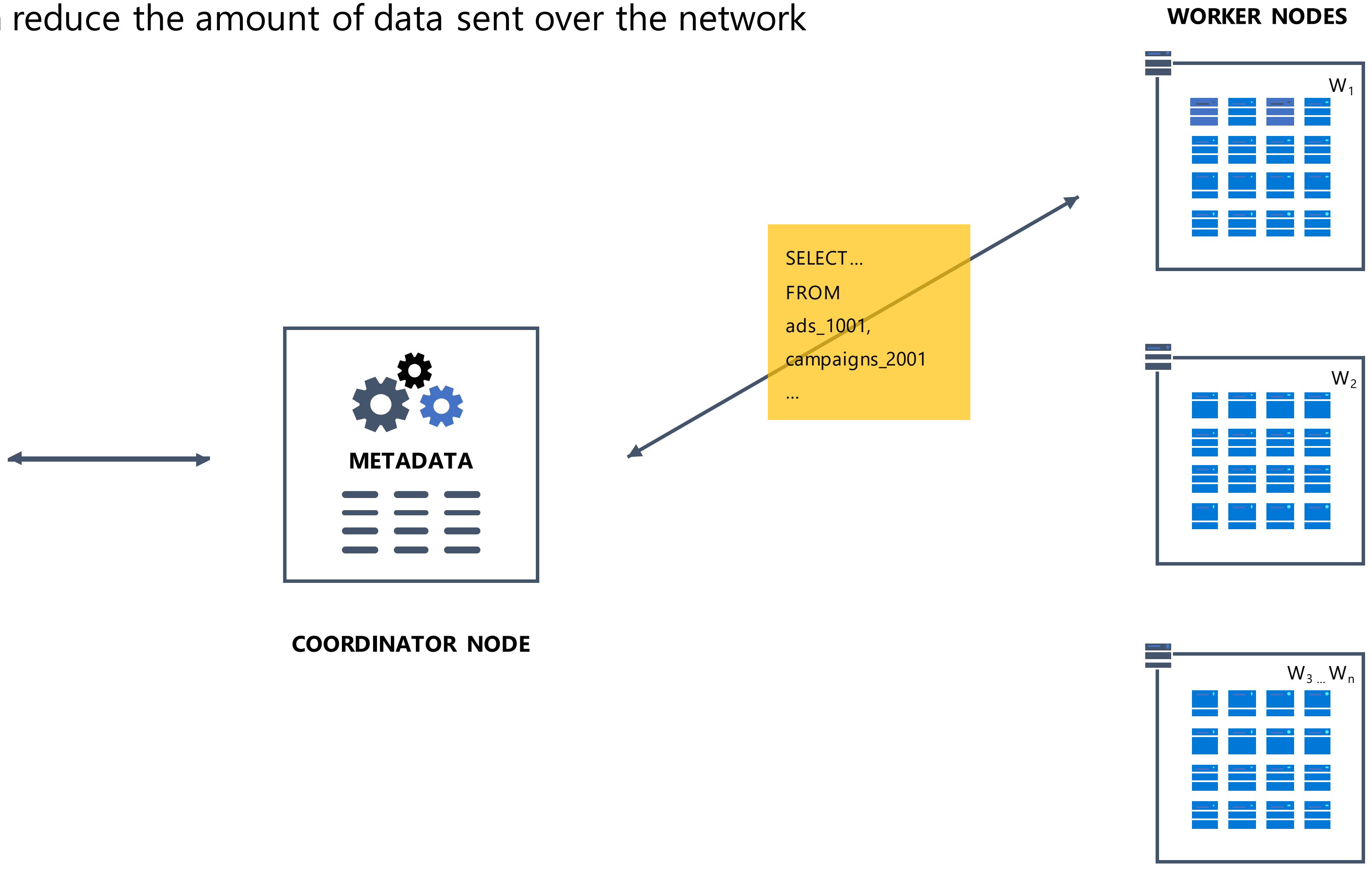
Co-location handles

- Joins
- Foreign keys/ Primary keys
- Rollups
- Others in future slides…

Co-located join

It’s logical to place shards containing related rows of related tables together on the same nodes

Join queries between related rows can reduce the amount of data sent over the network



- **APPLICATION**

```
SELECT count(*)
  FROM ads JOIN campaigns ON
       ads.company_id = campaigns.company_id
 WHERE ads.designer_name = ‘Isaac’
   AND campaigns.company_id = ‘Elly Co’
```

```
SELECT…
FROM
ads_1001,
campaigns_2001
…
```

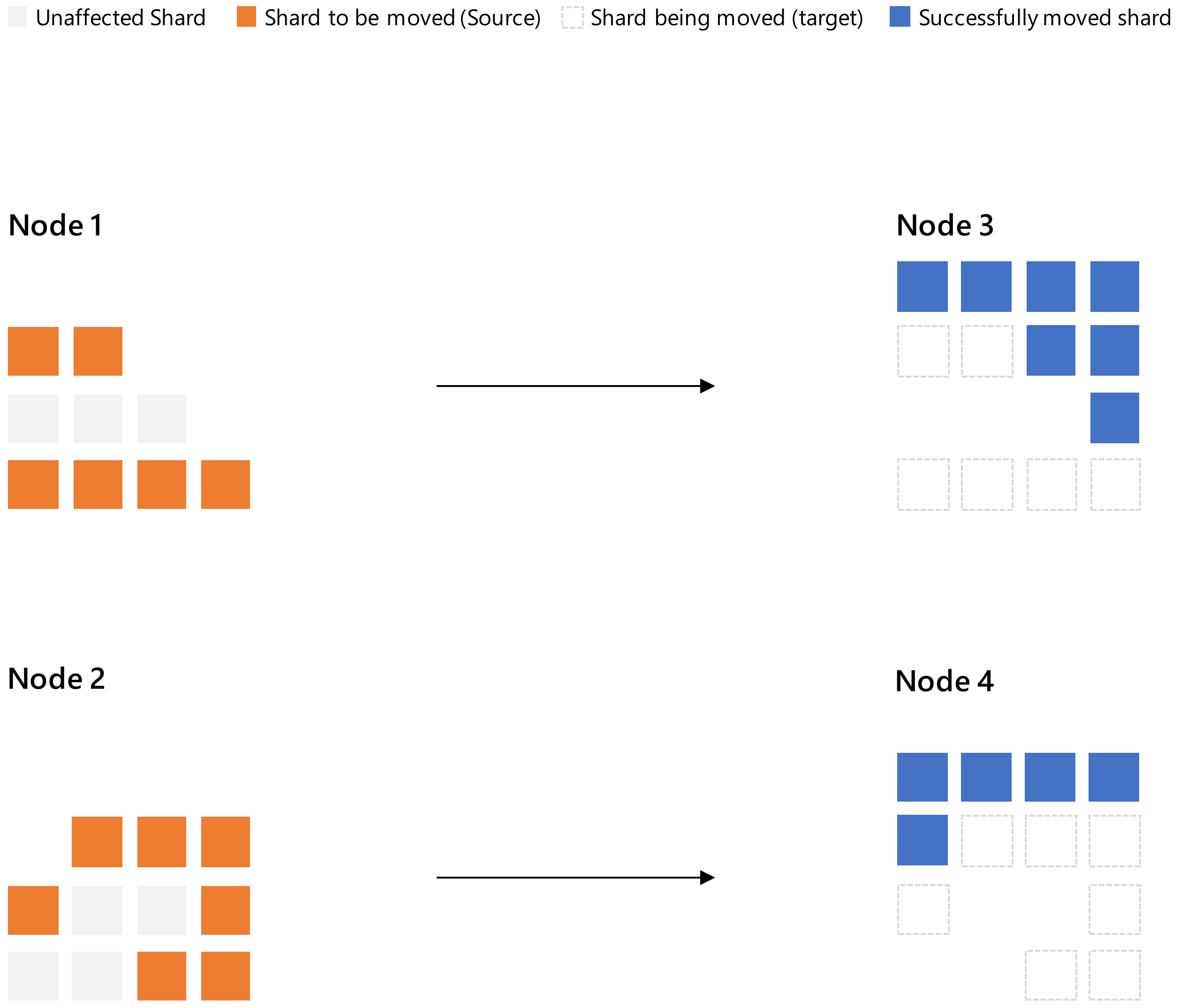
Effectively manage data scale out

Shard rebalancer redistributes shards across old and new worker nodes for balanced data scale out **without any downtime**.

Shard rebalancer will recommend rebalance when shards can be placed more evenly

For more control, use tenant isolation to easily allocate dedicated to specific tenants with greater needs

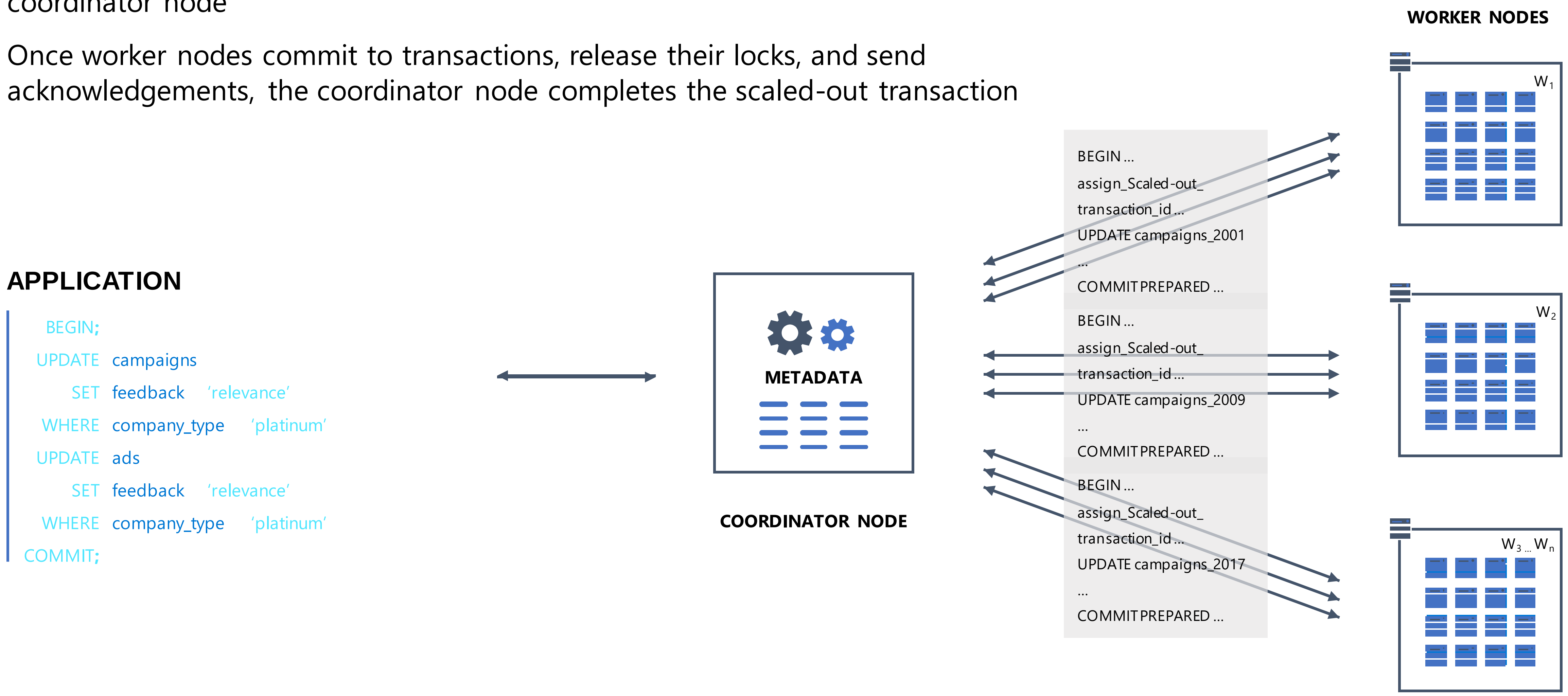
Shard Rebalancer



Scaled-out transaction

Hyperscale (Citus) leverages built-in 2PC protocol to prepare transactions via a coordinator node

Once worker nodes commit to transactions, release their locks, and send acknowledgements, the coordinator node completes the scaled-out transaction



```
  BEGIN;
UPDATE  campaigns
   SET  feedback   'relevance'
 WHERE  company_type   'platinum'
UPDATE  ads
   SET  feedback   'relevance'
 WHERE  company_type   'platinum'
COMMIT;
```

```
BEGIN ...
assign_Scaled-out_
transaction_id ...
UPDATE campaigns_2001
...
COMMIT PREPARED ...
```

```
BEGIN ...
assign_Scaled-out_
transaction_id ...
UPDATE campaigns_2009
...
COMMIT PREPARED ...
```

```
BEGIN ...
assign_Scaled-out_
transaction_id ...
UPDATE campaigns_2017
...
COMMIT PREPARED ...
```

Microsoft

Table Classification

3 Table Types

- Distributed Tables
- Reference Tables
- Local Tables

Distributed Tables

Definition:

- Tables that are sharded.

Classification:

- Large tables (>10GB) – shard on same key (may require addition of shard key)
- All tables are be co-located
- Enables localized and fast joins on workers
- Ex: transactions, events etc

```
SELECT create_distributed_table(table_name, column_name);
```

Reference Tables

Definition:

- Replicated to all the nodes (extra latency)

Classification:

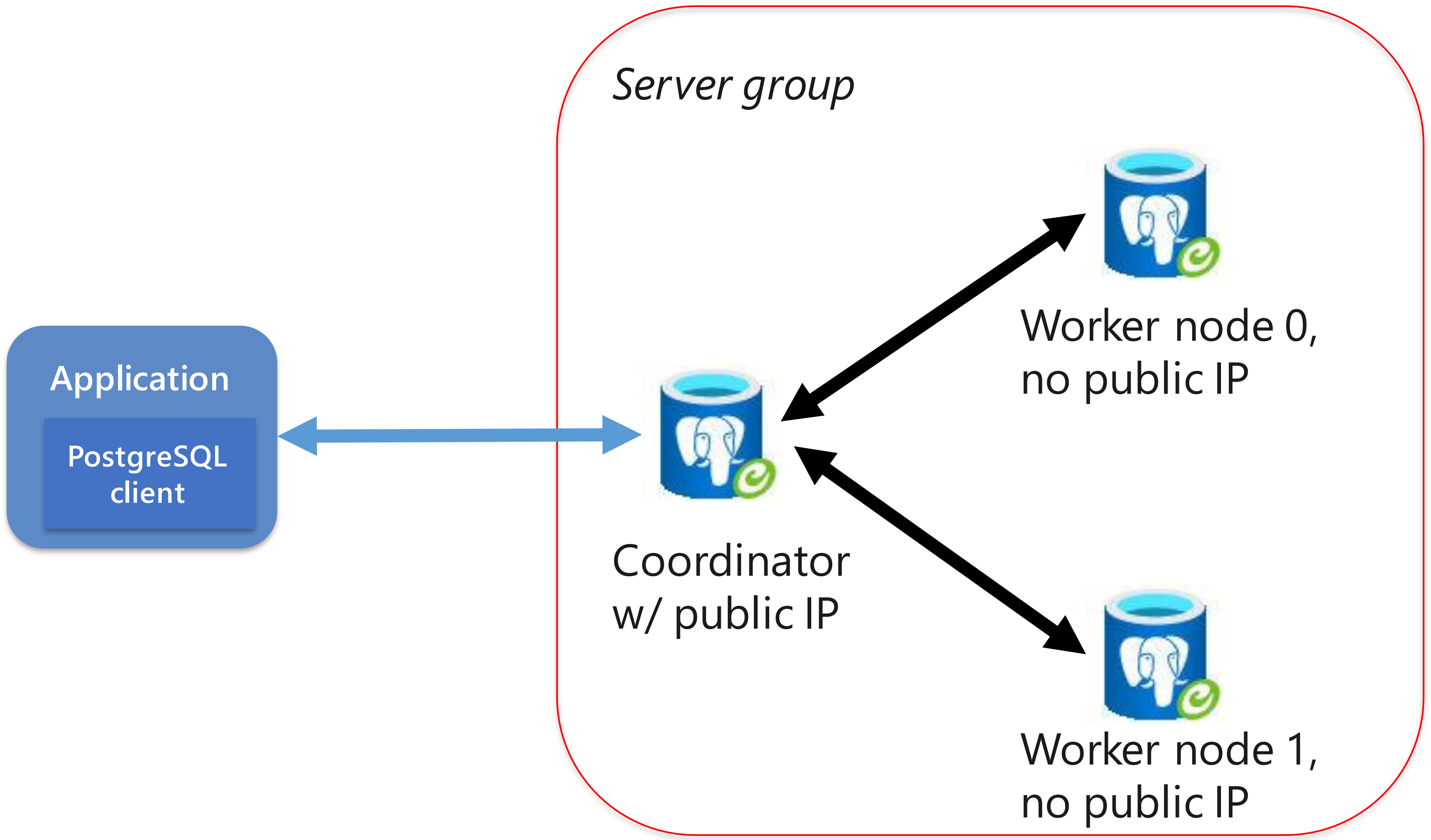
- Small tables < 10GB
- Efficient joins with distributed tables
- Cannot have sharding dimension
- Ex: countries, categories

```
SELECT create_reference_table(table_name);
```

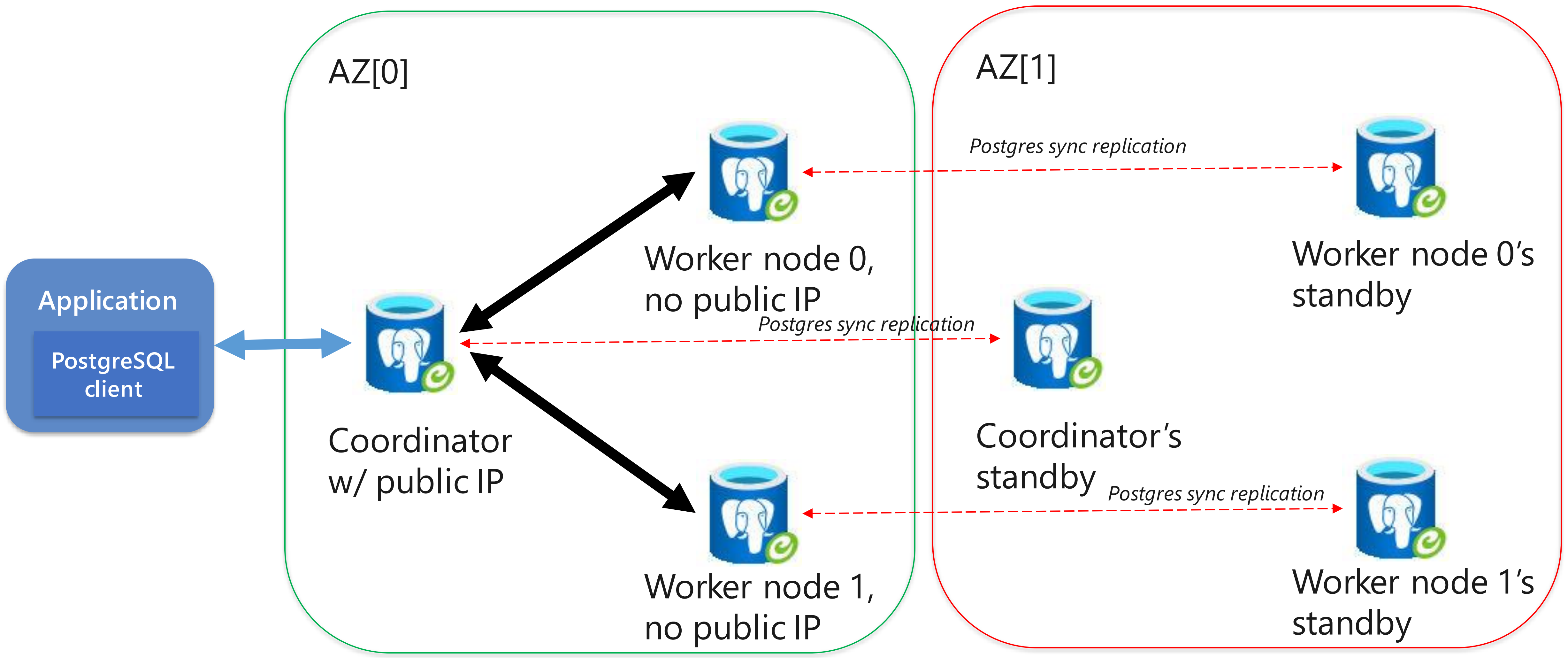
Local Tables

- Plain Postgres tables on the coordinator node.
- Admin Tables that don’t interact with main tables
- Separate micro-service that doesn’t need sharding

Hyperscale (Citus): Customer view



Hyperscale (Citus): High availability



AZ[0]
AZ[1]
Application
PostgreSQL client
Coordinator w/ public IP
Worker node 0, no public IP
Worker node 1, no public IP
Postgres sync replication
Postgres sync replication
Postgres sync replication
Coordinator's standby
Worker node 0's standby
Worker node 1's standby

Features: High availability (HA)

- Standby nodes for each primary node in Hyperscale (Citus)
  - Standby nodes are created in another AZ selected by service
  - Synchronous Postgres replication
  - Transparent for apps: Same connection string after failover
- Detection, failover, new standby creation
    - Detection: Up to 150 seconds (five 30 sec probes)
    - Failover: Up to 90 seconds
    - Total downtime: Up to 240 seconds
    - New standby creation: Up to 1 hour

Features: Connectivity and security

- Connection security (data-in-motion)
  - Connection to coordinator only
  - Firewall rules set for server group/coordinator
    - Specific IP/IP range
    - Allow all Azure services and resources
    - The whole world (0.0.0.0-255.255.255.255)
    - You can set it at Create time or after creation in Networking blade
  - Always TLS 1.2
- Storage security (data-at-rest)
  - Data, logs and backups encrypted with AES-256 cypher on storage level

Backup and restore

- Fully automated backup
  - Enabled on each node
  - Stored for 35 days
  - Deleted server
    - Backup is taken as a part of dropping the server and only this last backup is preserved
- Restore
  - Can restore to a date stamp with 5-minute increment
  - Need to open a support ticket to request PITR

Want to learn more?

http://tiny.cc/80lljz - Hyperscale
http://tiny.cc/n2lljz - ora2pg

Warsaw
Prague
Stuttgart
Geneva
Munich
Cologne
Paris
London
Amsterdam
Madrid
Oslo
Milan
Rome
Istanbul

Thank you!