Distributed Point-in-Time Recovery with Postgres

Eren Başak
Cloud Software Engineer
Citus Data

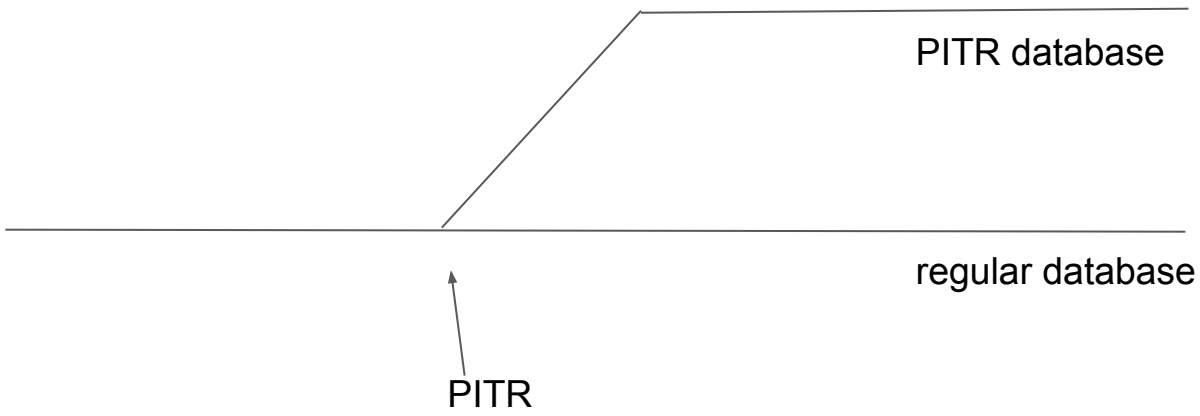
PGConf.Russia 2018

Overview

- What is Point-in-Time Recovery
- How to do point-in-time recovery
- Distributed Point-in-time-Recovery
- Citus Cloud way of doing PITR

Point-in-Time Recovery

- Point-in-the-past copy of an existing database
- “I want 02:45 pm yesterday copy of my database”
- A fork



Point-in-Time Recovery

Benefits and Use Cases

- DB Admin mistakes (DROP a wrong column)
- User deletes data by mistake

  “I ran our unit tests against the production database”

- Want an independent copy of the production database
  - Playground for data analysts
  - Understand the impact of bigger changes (a new index)

How to?
Prerequisites

- Periodic base backups

  Run pg_basebackup and archive the backups

- Wal Archiving

```
archive_command = ‘cp %p “/somewhere/reliable/%f”’
```

How to?
Recovery Steps

1. Determine a PITR target

   Timestamp or named restore points

2. Restore a proper backup

3. Prepare recovery.conf

How to?

recovery.conf

1. Restore Command to fetch necessary WAL files

```
restore_command = 'cp “/somewhere/reliable/%f” %p'
```

2. Recovery Target
   a. Named Restore Point: `recovery_target_name = ‘my-restore-point’`
   b. Time: `recovery_target_date = ‘2018-01-24 06:37:00 +0300’`
3. Other Settings
   a. `recovery_target_inclusive = true|false`
   b. `standby_mode = true|false`
   c. `recovery_target_action= shutdown|pause|promote`

How to?

Monitoring the progress

1. recovery.conf -> recovery.done after promotion
2. `SELECT pg_is_in_recovery()`
3. SELECT pg_last_xact_replay_timestamp()

Distributed PITR

1. Multiple PostgreSQL servers working together
   a. Citus
   b. Postgres-XL
   c. Application level sharding
2. PITR of all servers at once
3. May need to update metadata

   `pg_dist_node` for Citus

Distributed PITR

Simple approach

1. Use a suitable target time

   All servers should have backups before the selected time

2. PITR all servers to the target time

Distributed PITR

Simple approach: Problems

1. Clock differences

   A distributed transaction has completed in one node but not started at another node

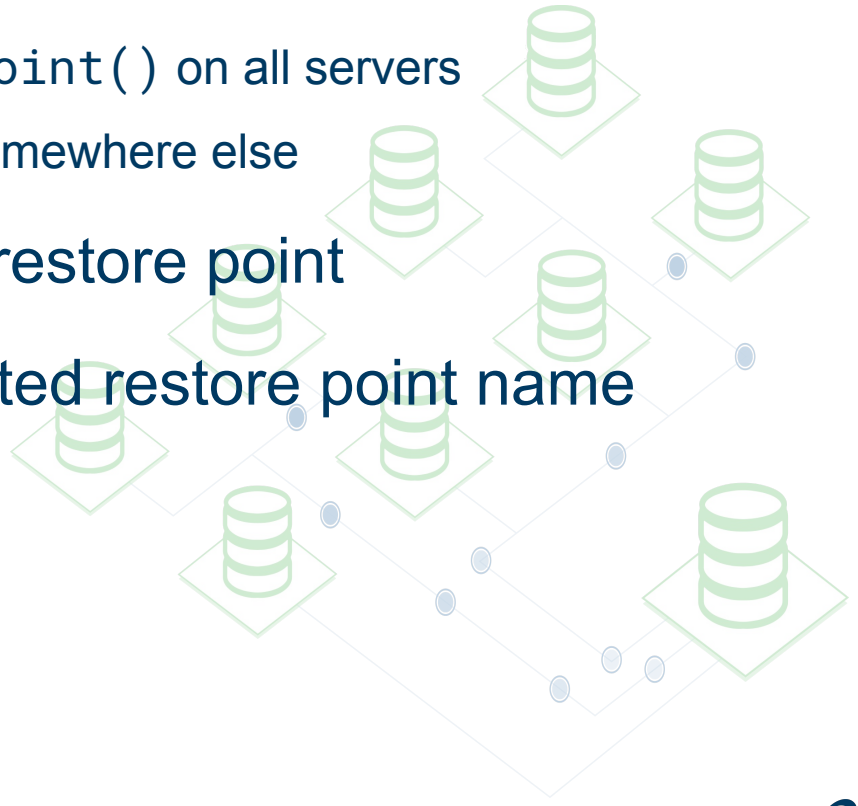
2. Ongoing transactions

   What if one transaction is aborted while others are ongoing

Distributed PITR

Distributed Restore Points

1. Periodically create distributed restore points
   a. Block all writes / take locks
   b. Run `pg_create_restore_point()` on all servers
   c. Store restore point name on somewhere else
2. Pick a suitable distributed restore point
3. Execute PITR with distributed restore point name



PITR at Citus Cloud

Citus is an extension to scale-out Postgres.

Citus Cloud: Managed Citus offering from Citus Data

- Nodes are on AWS EC2
- Daily backups of all servers to S3
- WAL archival to S3
- Backups are stored for 7 days
- Using WAL-E (and soon WAL-G)

PITR at Citus Cloud

1. User selects a target time and instance types.

   Temporary/non-production PITR clusters don't need to be as beefy as the production cluster.

2. Citus Cloud creates a new cluster.

3. Restore backups for each server.

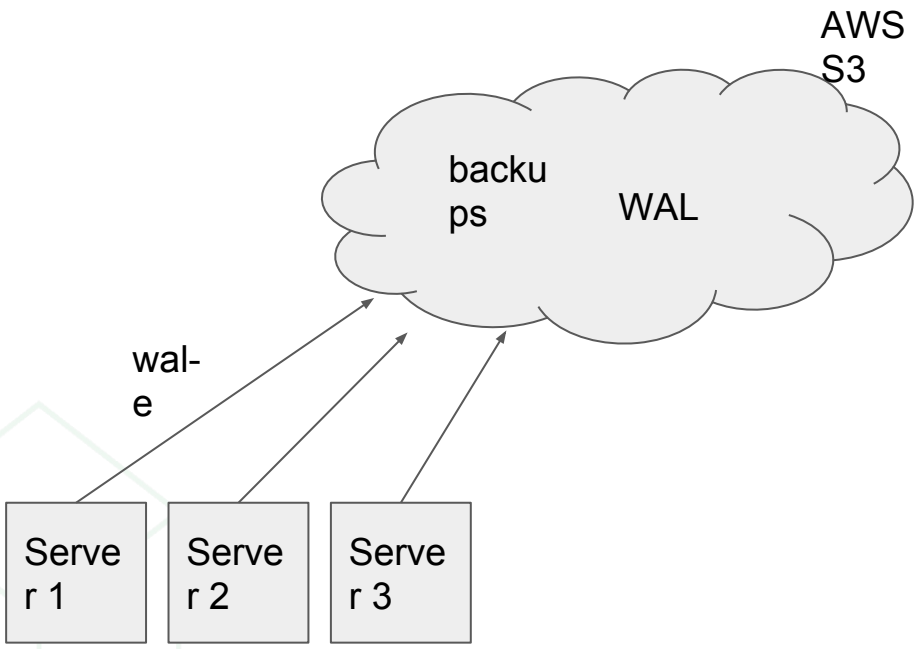
4. Update Coordinator Metadata after PITR is complete.

PITR at Citus Cloud

`citus_create_restore_point()`

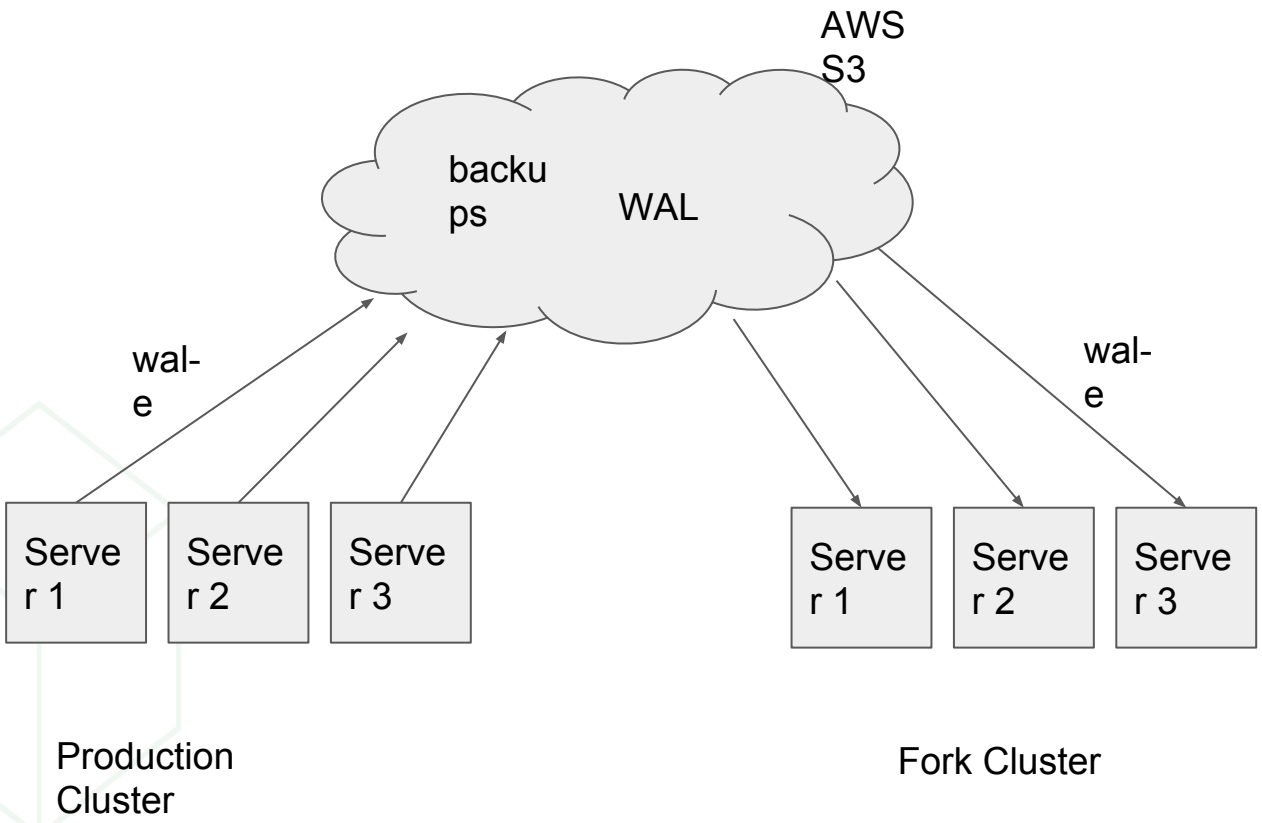
1. Open connections from coordinator to workers
2. Send BEGIN commands
3. Block distributed transactions by locking metadata
4. Run pg_create_restore_point() on the coordinator
5. Send pg_create_restore_point() commands

PITR at Citus Cloud

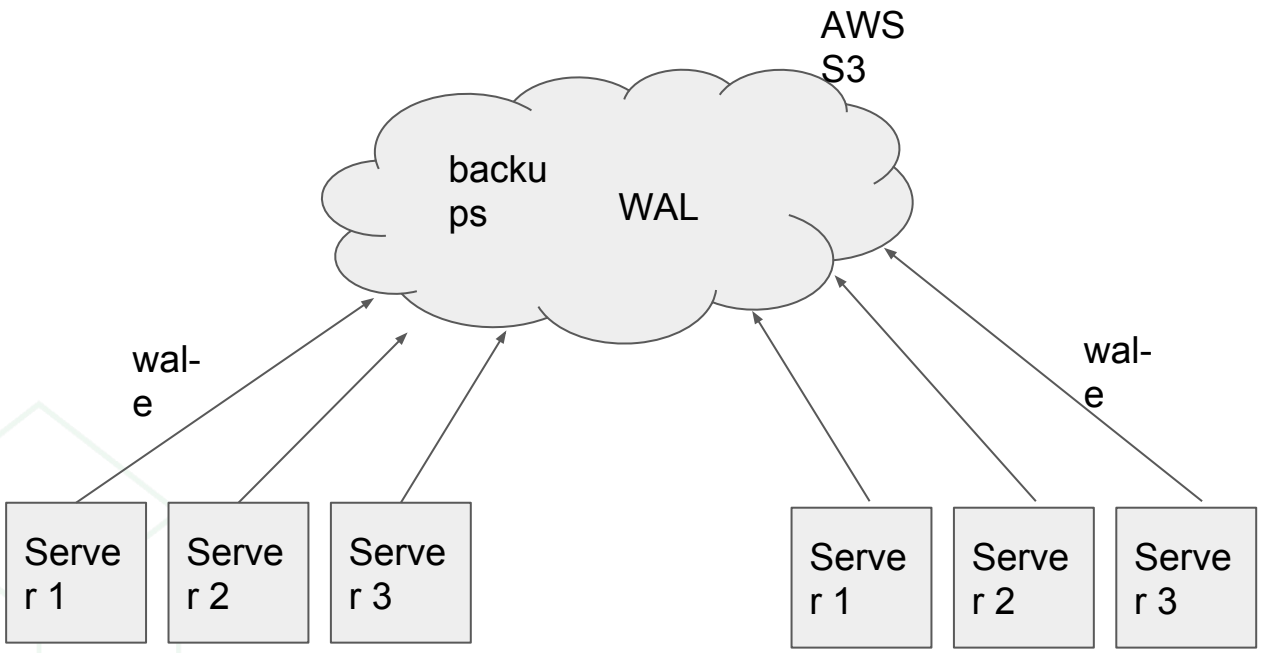


Normal State of a Citus Cluster

PITR at Citus Cloud



PITR at Citus Cloud



After PITR is completed

Thank You!

Eren Başak
eren@citusdata.com

www.citusdata.com  @aamederen