PERCONA

Databases run better with Percona

October 2024

PERCONA

Building Out Multi-Master Replication Clusters,

Architectural Considerations

PERCONA

Robert Bernier
Robert.bernier@percona.com
Senior PostgreSQL Consultant
Percona

About This Talk

- About This Talk
- Active-Active Replication
  - Understanding Active-Active
  - Active-Active: 2 Node example
- Architectural Types
  - Daisy Chain
  - Star
  - Mesh
  - Xmas Tree
- Administering The Logical Replication Cluster
  - About
  - Best Practices
  - Caveat
  - Conflict Resolution
  - Monitoring

PERCONA

Active-Active

Understanding Active-Active

- Key Concerns
  - DML conflicts when the subscribed data conflicts with data already present in the relation.
  - The "echo effect" where data, initially propagated from the published table, returns to its original source host server as a DML operation.
- Mitigation Techniques
  - Maintaining consistent table definitions i.e. table columns and their constraints between PUBLISHED and SUBSCRIBED tables.
  - Filtering undesired rows when creating the PUBLICATION using the WHERE clause
  - Setting the ORIGIN parameter to "none" at SUBSCRIPTION creation.

Active-Active: 2 Node example

Step 1: Create Environment

```
-- common to both primary nodes
create role active_active with login replication password 'active';
create database db01;
```

Active-Active: 2 Node example

Step 1 cont'd

```
-- pg1
\c db01
drop table if exists t1;
create table t1(
     id uuid default gen_random_uuid()
    ,comments text default 'pg1-openai-poc'
    ,t_stamp timestamptz default clock_timestamp()
    ,PRIMARY KEY (t_stamp,id)
);

grant all on all tables in schema public to active_active;
```

```
-- pg2-openai-poc
\c db01
drop table if exists t1;
create table t1(
     id uuid default gen_random_uuid()
    ,comments text default 'pg2-openai-poc'
    ,t_stamp timestamptz default clock_timestamp()
    ,PRIMARY KEY (t_stamp,id)
);

grant all on all tables in schema public to active_active;
```

Active-Active: 2 Node example

Step 2: Start Replication

```
-- PUBLICATION

-- pg1
create publication pg1 for table t1;


-- pg2
create publication pg2 for table t1;
```

```
-- SUBSCRIPTION

-- pg1
create subscription pg1
    connection 'host=pg2-openai-poc.postgres.database.azure.com port=5432
dbname=db01
    user=active_active password=active'
    publication pg2
    with (origin=none, copy_data=false);

-- pg2
create subscription pg2
    connection 'host=pg1-openai-poc.postgres.database.azure.com port=5432
dbname=db01
    user=active_active password=active'
    publication pg1
    with (origin=none, copy_data=true);
```

Active-Active: 2 Node example

Step 3: Validation

```
-- pg1
insert into t1 values
    (default,default,default),
    (default,default,default),
    (default,default,default);
```

```
-- pg2
insert into t1 values
    (default,default,default),
    (default,default,default),
    (default,default,default);
```

Active-Active: Caveat

- In order that logical replication can function all tuples (records) must be unique in a table i.e. possess a PRIMARY, or UNIQUE, key..
- Where primary or unique keys do not exist in a table, the SQL command ALTER TABLE … REPLICA IDENTITY … can add the necessary information to the WAL making logical replication possible.

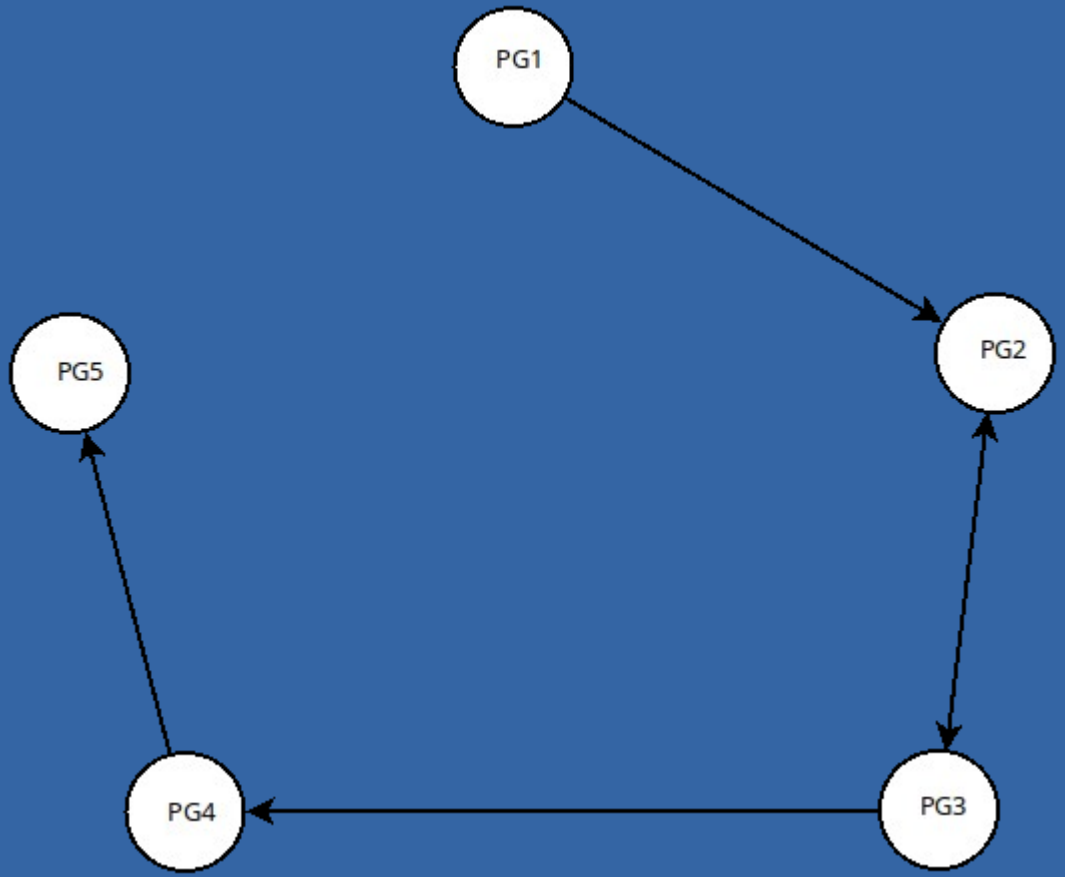
PERCONA

Architectural Types

Architectural Types

- Daisy Chain
- Star
- Mesh

Daisy Chain
(1 way-replication)



Daisy Chain

Step 1: Create Table and Publication

```
for u in 1 2 3 4 5
do
    echo "========= pg$u ========="
    psql -q "host=pg$u dbname=db01 user=postgres password=postgres" <<_eof_
        drop publication if exists pg$u;
        drop table if exists t1;
        create table t1(
            id uuid default gen_random_uuid()
            ,comments text default 'pg$u-openai'
            ,t_stamp timestamptz default clock_timestamp()
            ,PRIMARY KEY (t_stamp,id)
        );
        grant all on all tables in schema public to active_active;
        create publication pg$u for table t1;
_eof_
done
```

Daisy Chain

Step 2: Create the Subscription

```
for u in 2 3 4 5
do
    let x=$u-1
    echo "========= pg$x -> pg$u ========="
    psql -q "host=pg$u dbname=db01 user=postgres password=postgres" <<_eof_
        create subscription pg$x
            connection 'host=pg$x port=5432 dbname=db01 user=active_active
password=active'
            publication pg$x
            with (origin=any, copy_data=false, slot_name=pg$u);
_eof_
done
```

Daisy Chain
Step 3: Insert Records

```
for u in 1 2 3 4 5
do
    echo "========= pg${u} ========="
    psql -q "host=pg${u} dbname=db01 user=postgres password=postgres" <<_eof_
        insert into t1 values
            (default,default,default),
            (default,default,default);
_eof_
    sleep 1s
done
```

Daisy Chain
Step 4: Query Records

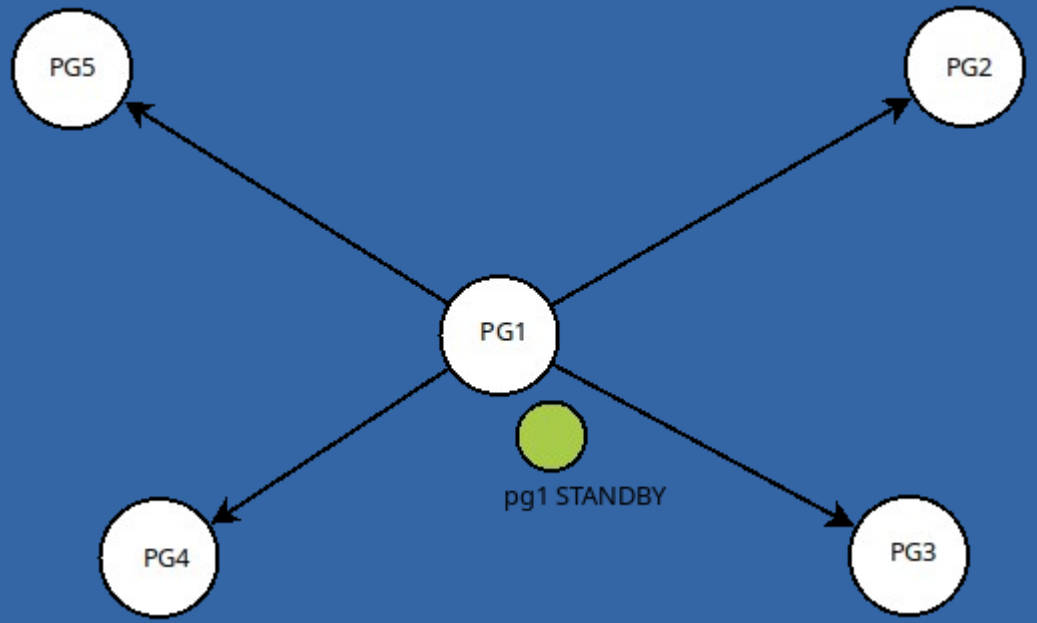
```
for u in 1 2 3 4 5
do
    echo "========= pg${u} ========="
    psql -q "host=pg${u} dbname=db01 user=postgres password=postgres" <<_eof_
        select * from t1 order by 3;
_eof_
done
```

Daisy Chain
Step 4 cont'd:

- Advantages ?
- Disadvantages?

```
========= pg1 =========
                  id                  |  comments  |            t_stamp
--------------------------------------+------------+-------------------------------
 8d32985a-b78e-421f-a679-1090275a0c38 | pg1-openai | 2024-06-25 21:17:11.958845+00
 868732b3-5e1a-46f8-8606-8c418f79240b | pg1-openai | 2024-06-25 21:17:11.958971+00
(2 rows)

========= pg2 =========
                  id                  |  comments  |            t_stamp
--------------------------------------+------------+-------------------------------
 8d32985a-b78e-421f-a679-1090275a0c38 | pg1-openai | 2024-06-25 21:17:11.958845+00
 868732b3-5e1a-46f8-8606-8c418f79240b | pg1-openai | 2024-06-25 21:17:11.958971+00
 266ce44c-a896-4f97-b2ce-cee4ef4fb15d | pg2-openai | 2024-06-25 21:17:13.000763+00
 8075f71c-5af5-45db-9db6-5e0e679e4ada | pg2-openai | 2024-06-25 21:17:13.000832+00
(4 rows)

========= pg3 =========
                  id                  |  comments  |            t_stamp
--------------------------------------+------------+-------------------------------
 8d32985a-b78e-421f-a679-1090275a0c38 | pg1-openai | 2024-06-25 21:17:11.958845+00
 868732b3-5e1a-46f8-8606-8c418f79240b | pg1-openai | 2024-06-25 21:17:11.958971+00
 266ce44c-a896-4f97-b2ce-cee4ef4fb15d | pg2-openai | 2024-06-25 21:17:13.000763+00
 8075f71c-5af5-45db-9db6-5e0e679e4ada | pg2-openai | 2024-06-25 21:17:13.000832+00
 e060482e-1e4d-48d9-88ec-733d738b741c | pg3-openai | 2024-06-25 21:17:14.041163+00
 d3414363-bb00-489f-94b8-e37908a729f4 | pg3-openai | 2024-06-25 21:17:14.041231+00
(6 rows)

========= pg4 =========
                  id                  |  comments  |            t_stamp
--------------------------------------+------------+-------------------------------
 8d32985a-b78e-421f-a679-1090275a0c38 | pg1-openai | 2024-06-25 21:17:11.958845+00
 868732b3-5e1a-46f8-8606-8c418f79240b | pg1-openai | 2024-06-25 21:17:11.958971+00
 266ce44c-a896-4f97-b2ce-cee4ef4fb15d | pg2-openai | 2024-06-25 21:17:13.000763+00
 8075f71c-5af5-45db-9db6-5e0e679e4ada | pg2-openai | 2024-06-25 21:17:13.000832+00
 e060482e-1e4d-48d9-88ec-733d738b741c | pg3-openai | 2024-06-25 21:17:14.041163+00
 d3414363-bb00-489f-94b8-e37908a729f4 | pg3-openai | 2024-06-25 21:17:14.041231+00
 6225fc7f-038d-4001-a8ad-e2f9c544bee2 | pg4-openai | 2024-06-25 21:17:15.081426+00
 3bbc856c-65e0-4789-a106-b3a2f29097dc | pg4-openai | 2024-06-25 21:17:15.081499+00
(8 rows)

========= pg5 =========
                  id                  |  comments  |            t_stamp
--------------------------------------+------------+-------------------------------
 8d32985a-b78e-421f-a679-1090275a0c38 | pg1-openai | 2024-06-25 21:17:11.958845+00
 868732b3-5e1a-46f8-8606-8c418f79240b | pg1-openai | 2024-06-25 21:17:11.958971+00
 266ce44c-a896-4f97-b2ce-cee4ef4fb15d | pg2-openai | 2024-06-25 21:17:13.000763+00
 8075f71c-5af5-45db-9db6-5e0e679e4ada | pg2-openai | 2024-06-25 21:17:13.000832+00
 e060482e-1e4d-48d9-88ec-733d738b741c | pg3-openai | 2024-06-25 21:17:14.041163+00
 d3414363-bb00-489f-94b8-e37908a729f4 | pg3-openai | 2024-06-25 21:17:14.041231+00
 6225fc7f-038d-4001-a8ad-e2f9c544bee2 | pg4-openai | 2024-06-25 21:17:15.081426+00
 3bbc856c-65e0-4789-a106-b3a2f29097dc | pg4-openai | 2024-06-25 21:17:15.081499+00
 a9690074-2eff-423d-bec6-8a81c5043f27 | pg5-openai | 2024-06-25 21:17:16.121258+00
 169d3989-c951-480d-8d21-6fc24dbde30d | pg5-openai | 2024-06-25 21:17:16.121351+00
(10 rows)
```

Star Topology
(1 way-replication)



Star Topology

Step 1: Create Table and Publication on Primary

```
psql "host=pg1 dbname=db01 user=postgres password=postgres" <<_eof_
    drop publication if exists pg1;
    drop table if exists t1;
    create table t1(
        id uuid default gen_random_uuid()
        ,comments text default 'pg1-openai'
        ,t_stamp timestamptz default clock_timestamp()
        ,PRIMARY KEY (t_stamp,id)
    );
    create publication pg1 for table t1;
    grant all on all tables in schema public to active_active;
_eof_
```

Star Topology
Step 1 cont'd: Create Table on REPLICA(s)

```
psql "host=pg1 dbname=db01 user=postgres password=postgres" <<_eof_
    drop publication if exists pg1;
    drop table if exists t1;
    create table t1(
        id uuid default gen_random_uuid()
        ,comments text default 'pg1-openai'
        ,t_stamp timestamptz default clock_timestamp()
        ,PRIMARY KEY (t_stamp,id)
    );
_eof_
```

Star Topology

Step 2: Create Subscription

```
for u in 2 3 4 5
do
    echo "========= pg1 -> pg$u ========="
psql -q "host=pg$u dbname=db01 user=postgres password=postgres" <<_eof_
create subscription pg1
connection 'host=pg1 port=5432 dbname=db01 user=active_active password=active'
           publication pg1
           with (origin=none, copy_data=false, slot_name=pg$u);
_eof_
echo "========= pg$u ========="
psql -q "host=pg$u dbname=db01 user=postgres password=postgres" <<_eof_
select subname, subenabled, subconninfo, subslotname from
       pg_subscription;
_eof_
done
```

Star Topology
step 2 cont'd:

```
done
========= pg1 -> pg2 =========
WARNING:  publication "pg2" does not exist on the publisher
NOTICE:  created replication slot "pg2" on publisher
========= pg2 =========
 subname | subenabled |                            subconninfo                                | subslotname
---------+------------+-----------------------------------------------------------------------+-------------
 pg1     | t          | host=pg1 port=5432 dbname=db01 user=active_active password=active | pg2
(1 row)

========= pg1 -> pg3 =========
WARNING:  publication "pg3" does not exist on the publisher
NOTICE:  created replication slot "pg3" on publisher
========= pg3 =========
 subname | subenabled |                            subconninfo                                | subslotname
---------+------------+-----------------------------------------------------------------------+-------------
 pg1     | t          | host=pg1 port=5432 dbname=db01 user=active_active password=active | pg3
(1 row)

========= pg1 -> pg4 =========
WARNING:  publication "pg4" does not exist on the publisher
NOTICE:  created replication slot "pg4" on publisher
========= pg4 =========
 subname | subenabled |                            subconninfo                                | subslotname
---------+------------+-----------------------------------------------------------------------+-------------
 pg1     | t          | host=pg1 port=5432 dbname=db01 user=active_active password=active | pg4
(1 row)

========= pg1 -> pg5 =========
WARNING:  publication "pg5" does not exist on the publisher
NOTICE:  created replication slot "pg5" on publisher
========= pg5 =========
 subname | subenabled |                            subconninfo                                | subslotname
---------+------------+-----------------------------------------------------------------------+-------------
 pg1     | t          | host=pg1 port=5432 dbname=db01 user=active_active password=active | pg5
(1 row)
```

Star Topology

Step 3: Insert Records

```
for u in 1 2 3 4 5
do
    echo "========= pg${u} ========="
    psql -q "host=pg${u} dbname=db01 user=postgres password=postgres" <<_eof_
        insert into t1 values
            (default,default,default),
            (default,default,default);
_eof_
done
```

Star Topology
Step 4: Query Records

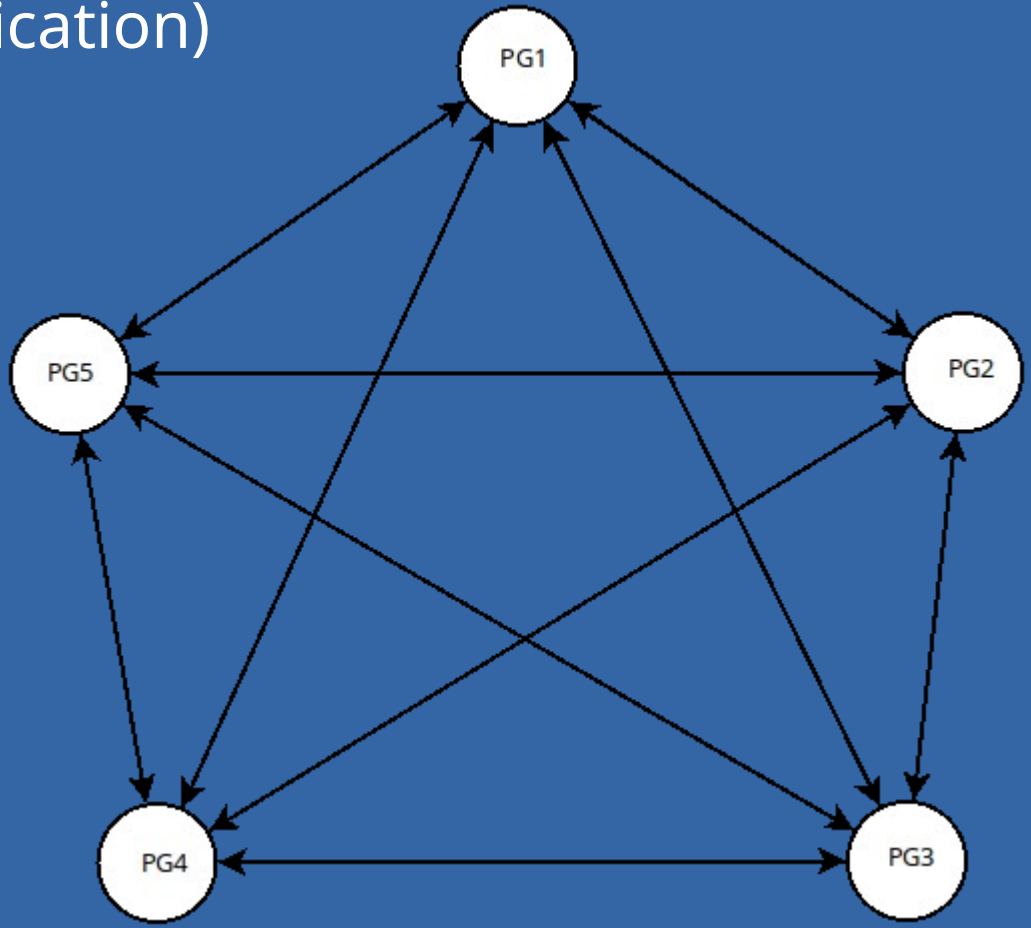
```
for u in 1 2 3 4 5
do
    echo "========= pg${u} ========="
    psql -q "host=pg${u} dbname=db01 user=postgres password=postgres" <<_eof_
        select * from t1 order by 3;
_eof_
done
```

Star Topology
step 4 cont'd:

- Advantages ?
- Disadvantages?

```
========== pg1 ==========
                  id                  |  comments  |           t_stamp
--------------------------------------+------------+-------------------------------
 8ede70f3-8409-4783-b03a-f05add1899e2 | pg1-openai | 2024-06-26 19:10:26.834392+00
 9ef5d0db-4a93-4886-b489-4a22f0950068 | pg1-openai | 2024-06-26 19:10:26.834529+00
(2 rows)

========== pg2 ==========
                  id                  |  comments  |           t_stamp
--------------------------------------+------------+-------------------------------
 8ede70f3-8409-4783-b03a-f05add1899e2 | pg1-openai | 2024-06-26 19:10:26.834392+00
 9ef5d0db-4a93-4886-b489-4a22f0950068 | pg1-openai | 2024-06-26 19:10:26.834529+00
 03dd34bc-2212-4837-a6bd-ebd9d10e0a51 | pg2-openai | 2024-06-26 19:10:26.874537+00
 69d982e2-445d-4add-86e6-5d4ee6ccf753 | pg2-openai | 2024-06-26 19:10:26.874612+00
(4 rows)

========== pg3 ==========
                  id                  |  comments  |           t_stamp
--------------------------------------+------------+-------------------------------
 8ede70f3-8409-4783-b03a-f05add1899e2 | pg1-openai | 2024-06-26 19:10:26.834392+00
 9ef5d0db-4a93-4886-b489-4a22f0950068 | pg1-openai | 2024-06-26 19:10:26.834529+00
 c4b7f0f1-eba2-4c16-ab69-930dd7ed701a | pg3-openai | 2024-06-26 19:10:26.910106+00
 ece3e682-9105-4aa7-ba43-7b75867285f0 | pg3-openai | 2024-06-26 19:10:26.910186+00
(4 rows)

========== pg4 ==========
                  id                  |  comments  |           t_stamp
--------------------------------------+------------+-------------------------------
 8ede70f3-8409-4783-b03a-f05add1899e2 | pg1-openai | 2024-06-26 19:10:26.834392+00
 9ef5d0db-4a93-4886-b489-4a22f0950068 | pg1-openai | 2024-06-26 19:10:26.834529+00
 6ed4c0e6-bc00-48b6-9817-3e8d0cee3e13 | pg4-openai | 2024-06-26 19:10:26.943257+00
 4894ce41-2930-4663-b1c2-76a530f485e0 | pg4-openai | 2024-06-26 19:10:26.943337+00
(4 rows)

========== pg5 ==========
                  id                  |  comments  |           t_stamp
--------------------------------------+------------+-------------------------------
 8ede70f3-8409-4783-b03a-f05add1899e2 | pg1-openai | 2024-06-26 19:10:26.834392+00
 9ef5d0db-4a93-4886-b489-4a22f0950068 | pg1-openai | 2024-06-26 19:10:26.834529+00
 5f68691c-1a3e-4d35-9a0b-a53ee8a2eae1 | pg5-openai | 2024-06-26 19:10:26.978153+00
 cbba47f3-a0f7-4735-86b2-96f01cbe7307 | pg5-openai | 2024-06-26 19:10:26.978228+00
(4 rows)
```

Mesh
(2 way-replication)



Mesh

Step 1: Create Table and Publication on nodes

```
for u in 1 2 3 4 5
do
echo "========= pg${u} ========="
psql "host=pg${u} dbname=db01 user=postgres password=postgres" <<_eof_
    drop publication if exists pg${u};
    drop table if exists t1;
    create table t1(
        id uuid default gen_random_uuid()
        ,comments text default 'pg${u}-openai'
        ,t_stamp timestamptz default clock_timestamp()
        ,PRIMARY KEY (t_stamp,id)
    );
    grant all on all tables in schema public to active_active;
    create publication pg${u} for table t1;
_eof_
done
```

Mesh
Step 2: Create Subscription

```
for u in 1 2 3 4 5
do
    for v in 1 2 3 4 5
    do
    if [ $u -ne $v ]
    then
        echo "========= pg$u -> pg$v ========="
        psql -q "host=pg$u dbname=db01 user=postgres password=postgres" <<_eof_
            create subscription pg$v
                connection 'host=pg$v port=5432 dbname=db01 user=active_active
password=active'
                publication pg$v
                with (origin=none, copy_data=false, slot_name=pg$u);
_eof_
    fi
    done
done
```

Mesh

Step 3: Insert Records

```
for u in 1 2 3 4 5
do
    echo "========= pg${u} ========="
    psql -q "host=pg${u} dbname=db01 user=postgres password=postgres" <<_eof_
        insert into t1 values
            (default,default,default),
            (default,default,default);
_eof_
    sleep 1s
done
```

Mesh
Step 4: Query Records

```
for u in 1 2 3 4 5
do
    echo "========= pg${u} ========="
    psql -q "host=pg${u} dbname=db01 user=postgres password=postgres" <<_eof_
        select * from t1 order by 3;
_eof_
done
```

Mesh
step 4 cont’d:

```
======== pg1 ========
                  id                  |  comments  |           t_stamp
--------------------------------------+------------+-------------------------------
 5fab5f67-6cea-465c-9eb7-e46787ab84b0 | pg1-openai | 2024-06-26 17:12:01.9294+00
 08537516-bd1a-4849-b2da-8eec727529de | pg1-openai | 2024-06-26 17:12:01.929519+00
 72e9f04e-74fb-4521-9824-971bf5093340 | pg2-openai | 2024-06-26 17:12:02.970595+00
 3f855892-899d-4da4-8ae1-1e4b14025c9d | pg2-openai | 2024-06-26 17:12:02.97066+00
 12f8318d-4116-49ba-9643-217429637c30 | pg3-openai | 2024-06-26 17:12:04.013467+00
 70655e17-0d0c-4c16-a4ba-15c259dbbe39 | pg3-openai | 2024-06-26 17:12:04.013531+00
 61d5fa4b-2efd-44d9-8c9b-fb0d426c0b62 | pg4-openai | 2024-06-26 17:12:05.054036+00
 0c70375a-606a-4d85-908d-b4e5dd313293 | pg4-openai | 2024-06-26 17:12:05.054114+00
 3e020797-7a64-42b9-91df-4ea3f7011c7c | pg5-openai | 2024-06-26 17:12:06.094513+00
 c856550c-975b-43a1-84e5-61194a76ffdd | pg5-openai | 2024-06-26 17:12:06.094584+00
(10 rows)

======== pg2 ========
                  id                  |  comments  |           t_stamp
--------------------------------------+------------+-------------------------------
 5fab5f67-6cea-465c-9eb7-e46787ab84b0 | pg1-openai | 2024-06-26 17:12:01.9294+00
 08537516-bd1a-4849-b2da-8eec727529de | pg1-openai | 2024-06-26 17:12:01.929519+00
 72e9f04e-74fb-4521-9824-971bf5093340 | pg2-openai | 2024-06-26 17:12:02.970595+00
 3f855892-899d-4da4-8ae1-1e4b14025c9d | pg2-openai | 2024-06-26 17:12:02.97066+00
 12f8318d-4116-49ba-9643-217429637c30 | pg3-openai | 2024-06-26 17:12:04.013467+00
 70655e17-0d0c-4c16-a4ba-15c259dbbe39 | pg3-openai | 2024-06-26 17:12:04.013531+00
 61d5fa4b-2efd-44d9-8c9b-fb0d426c0b62 | pg4-openai | 2024-06-26 17:12:05.054036+00
 0c70375a-606a-4d85-908d-b4e5dd313293 | pg4-openai | 2024-06-26 17:12:05.054114+00
 3e020797-7a64-42b9-91df-4ea3f7011c7c | pg5-openai | 2024-06-26 17:12:06.094513+00
 c856550c-975b-43a1-84e5-61194a76ffdd | pg5-openai | 2024-06-26 17:12:06.094584+00
(10 rows)

======== pg3 ========
                  id                  |  comments  |           t_stamp
--------------------------------------+------------+-------------------------------
 5fab5f67-6cea-465c-9eb7-e46787ab84b0 | pg1-openai | 2024-06-26 17:12:01.9294+00
 08537516-bd1a-4849-b2da-8eec727529de | pg1-openai | 2024-06-26 17:12:01.929519+00
 72e9f04e-74fb-4521-9824-971bf5093340 | pg2-openai | 2024-06-26 17:12:02.970595+00
 3f855892-899d-4da4-8ae1-1e4b14025c9d | pg2-openai | 2024-06-26 17:12:02.97066+00
 12f8318d-4116-49ba-9643-217429637c30 | pg3-openai | 2024-06-26 17:12:04.013467+00
 70655e17-0d0c-4c16-a4ba-15c259dbbe39 | pg3-openai | 2024-06-26 17:12:04.013531+00
 61d5fa4b-2efd-44d9-8c9b-fb0d426c0b62 | pg4-openai | 2024-06-26 17:12:05.054036+00
 0c70375a-606a-4d85-908d-b4e5dd313293 | pg4-openai | 2024-06-26 17:12:05.054114+00
 3e020797-7a64-42b9-91df-4ea3f7011c7c | pg5-openai | 2024-06-26 17:12:06.094513+00
 c856550c-975b-43a1-84e5-61194a76ffdd | pg5-openai | 2024-06-26 17:12:06.094584+00
(10 rows)

======== pg4 ========
                  id                  |  comments  |           t_stamp
--------------------------------------+------------+-------------------------------
 5fab5f67-6cea-465c-9eb7-e46787ab84b0 | pg1-openai | 2024-06-26 17:12:01.9294+00
 08537516-bd1a-4849-b2da-8eec727529de | pg1-openai | 2024-06-26 17:12:01.929519+00
```

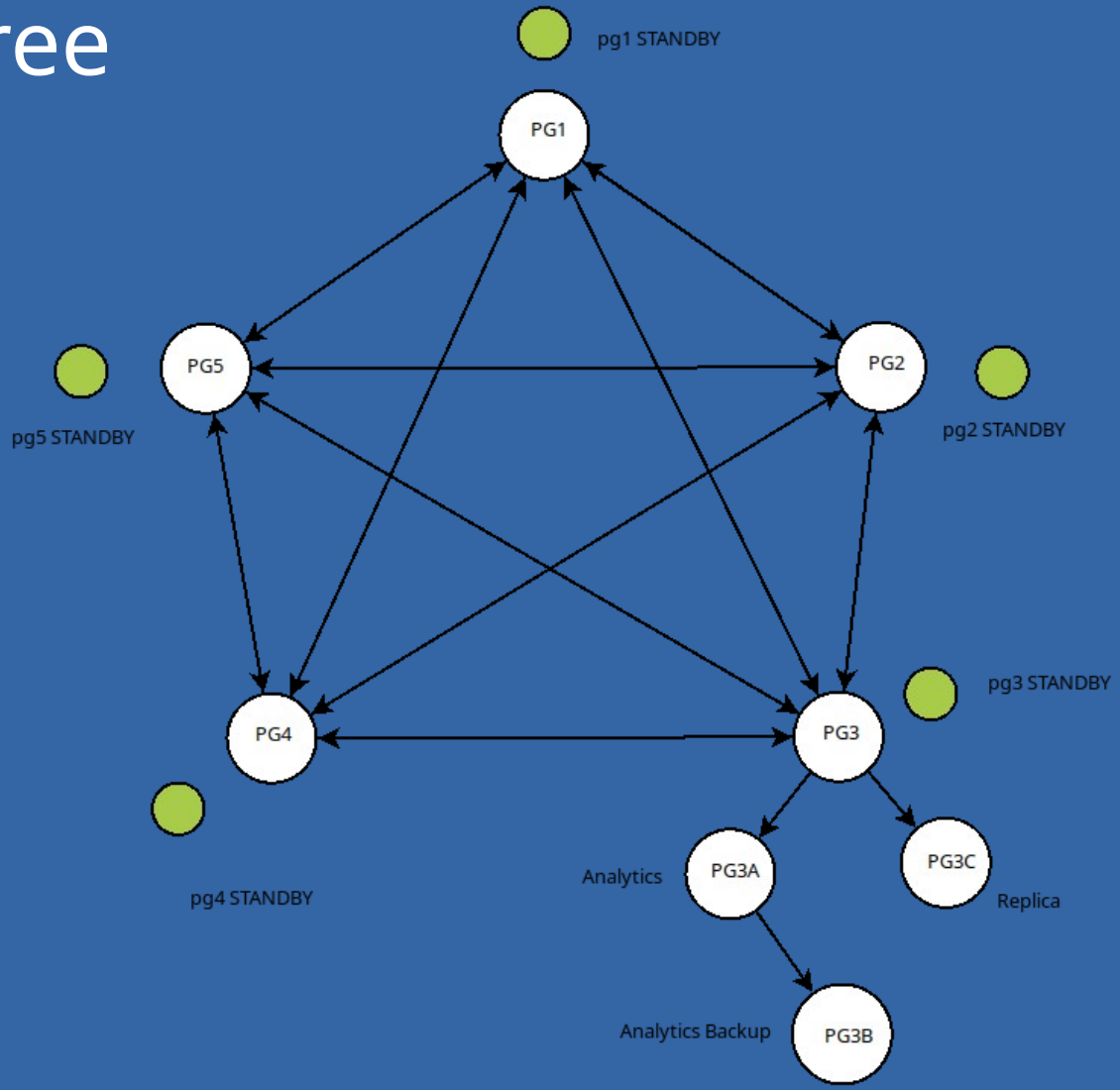
Mesh

- Advantages: Not only can multiple DML operations be executed on each host but one can failover to any one of the PRIMARY read-write nodes without downtime or SUBSCRIPTION configuration updates.
- Disadvantages: Topology complexity and network overhead increases as the system scales.

Mesh

Number Of Nodes	Total Number Slots req'd
2	2
3	6
4	12
5	20
6	30

XMAS Tree



Questions?

https://github.com/rbernierZulu/pg_conf_Seattle-2024

Thank You!

PASS24
Data Community Summit