aiven

StandBy Database Shootout PostgreSQL vs. Oracle v2

Dirk Krautschick

PGconf.DE 12.04.2024, Munich

#whoami

Dirk Krautschick
Solution Architect

with Aiven since Nov 2023

16 years

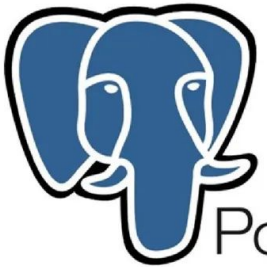
DBA, Trainer, Consulting, Sales Engineering

PostgreSQL, Oracle

Married, 2 Junior DBAs

Mountainbike, swimming, movies, music, hifi/home cinema, 8 bit computing

PostgreSQL User Group NRW



PostgreSQL

Founded Dec 2023

1st MeetUp (premiere!) in Feb 2024, Cologne @ ORDIX AG

2nd Meetup 15.05.2024 in **Aachen**

- *"Bringing Vectors to Postgres"* by Gülçin Yıldırım Jelínek
- *"pgBackRest Frequently Asked Questions"* by Stefan Fercot

Upcoming events in the pipeline, stay tuned!

- Target is at least 4 meetups a year
- All around NRW

Disclaimer

Different audience, different perspectives

My experience, my honest opinion

Let‘s stay open minded

Always open for discussions

Some History - Once upon a time…

High Availability was…

- Frequently dumps :-(
- Disk mirroring (drbd)
- Failover Cluster
- Trigger based solutions

wasn't perfect either….

- Similar "solutions"
- Oracle Failsafe / Failover Cluster …

…but then…

Some History - Evolution

2006 Log Shipping (8.2)

2010 Streaming Replication (9.0)

Including hot standby DBs

Synchronous with 9.1

2017 Logical Replication (10)

...

1996 manual standby DBs (7.3)

1999 managed standby DBs (8i)

2001 Data Guard Broker (9i), Real Application Cluster

2002 Oracle Streams (9.2)

2007 Active Data Guard (11g)

2009 Golden Gate

...

In memoriam...

PostgreSQL

Simon Riggs, † 26.03.2024

Long term Core Team Member and Contributor

Responsible for

- Sync Replication
- Hot Standby
- Point in Time Recovery

ORACLE DATABASE

Larry Carpenter, † 22.03.2024

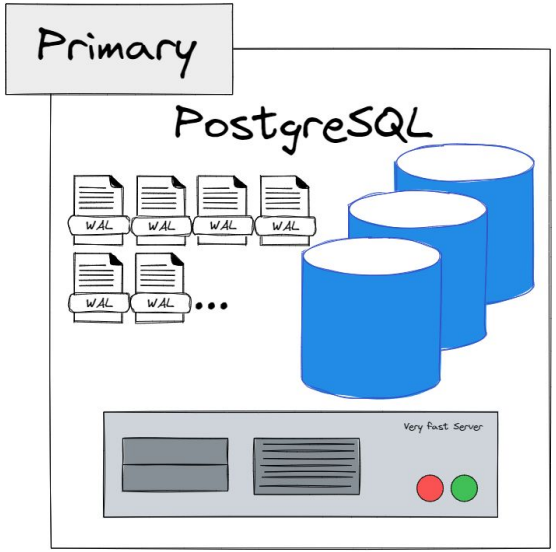
With Oracle since 1994

Data Guard Key Role since 2001

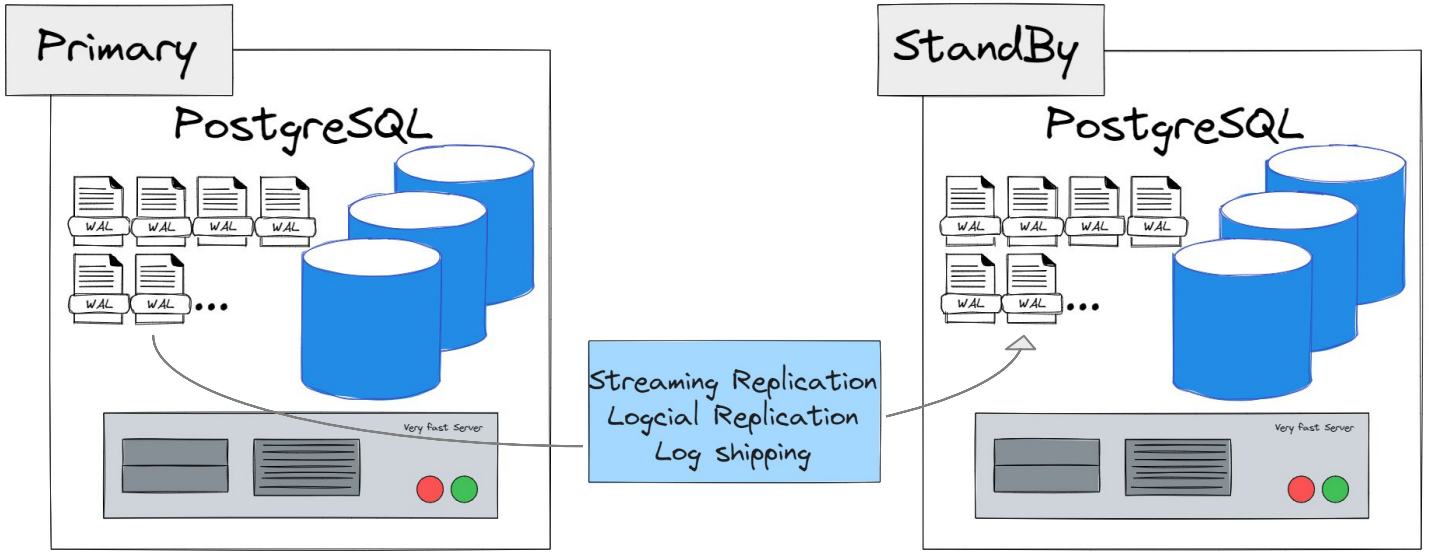
- Development
- Product Management

called as “Mr. Data Guard”

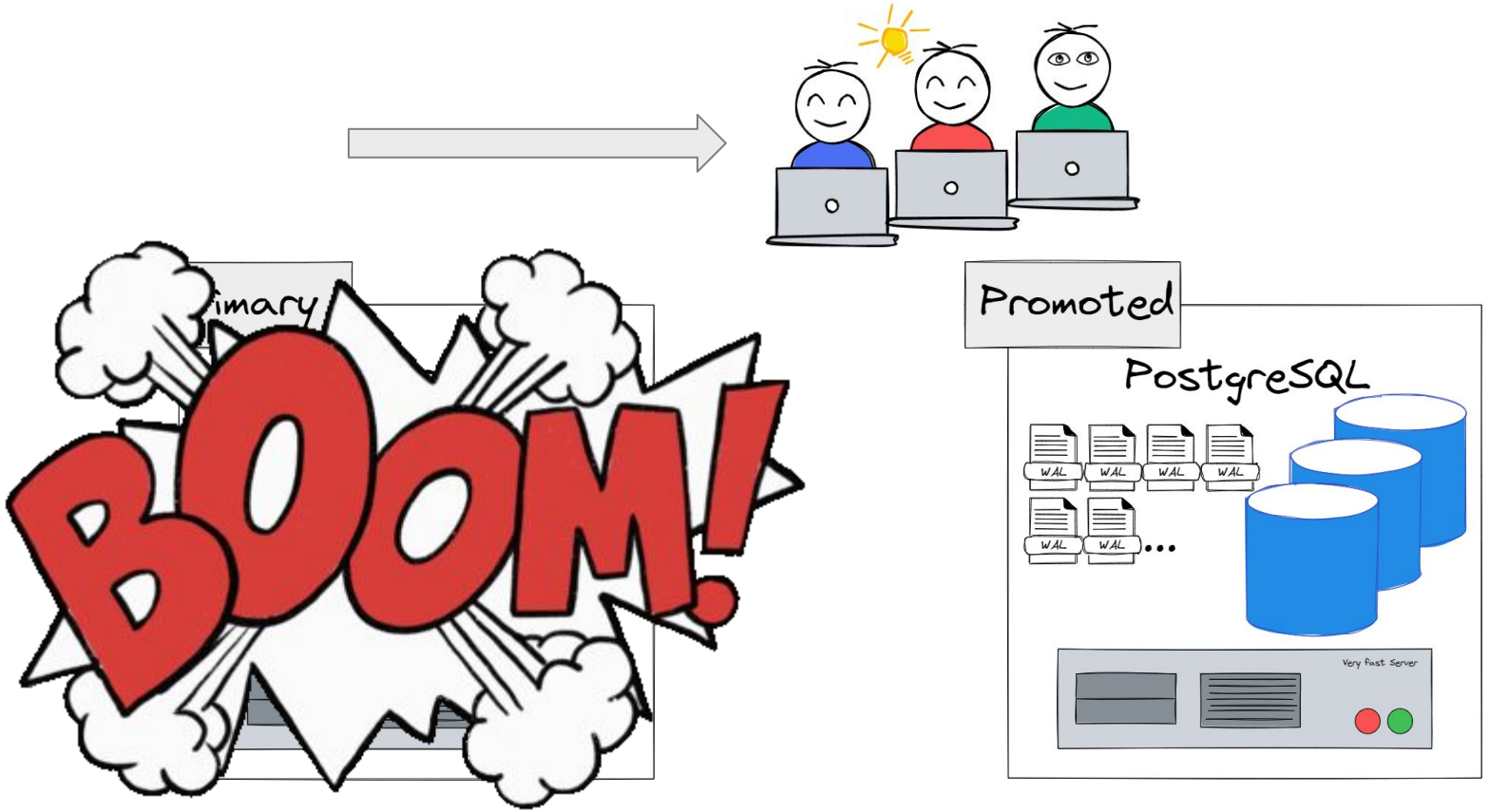
Concepts - The same Idea



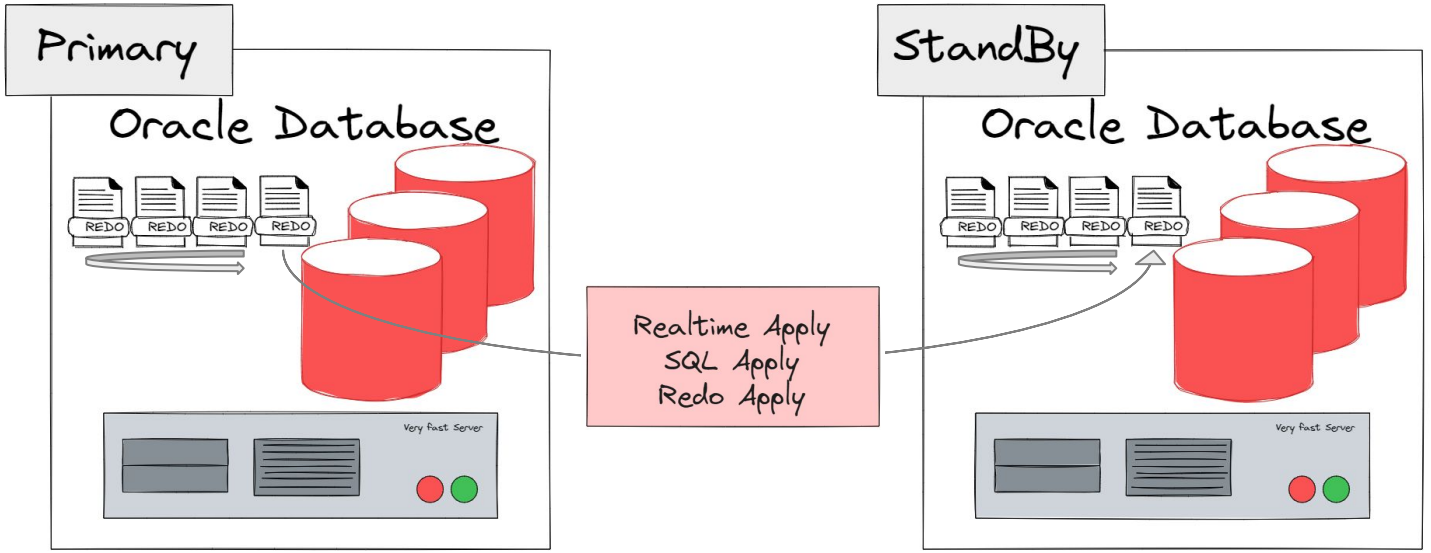
Concepts - The same Idea



Concepts - The same Idea



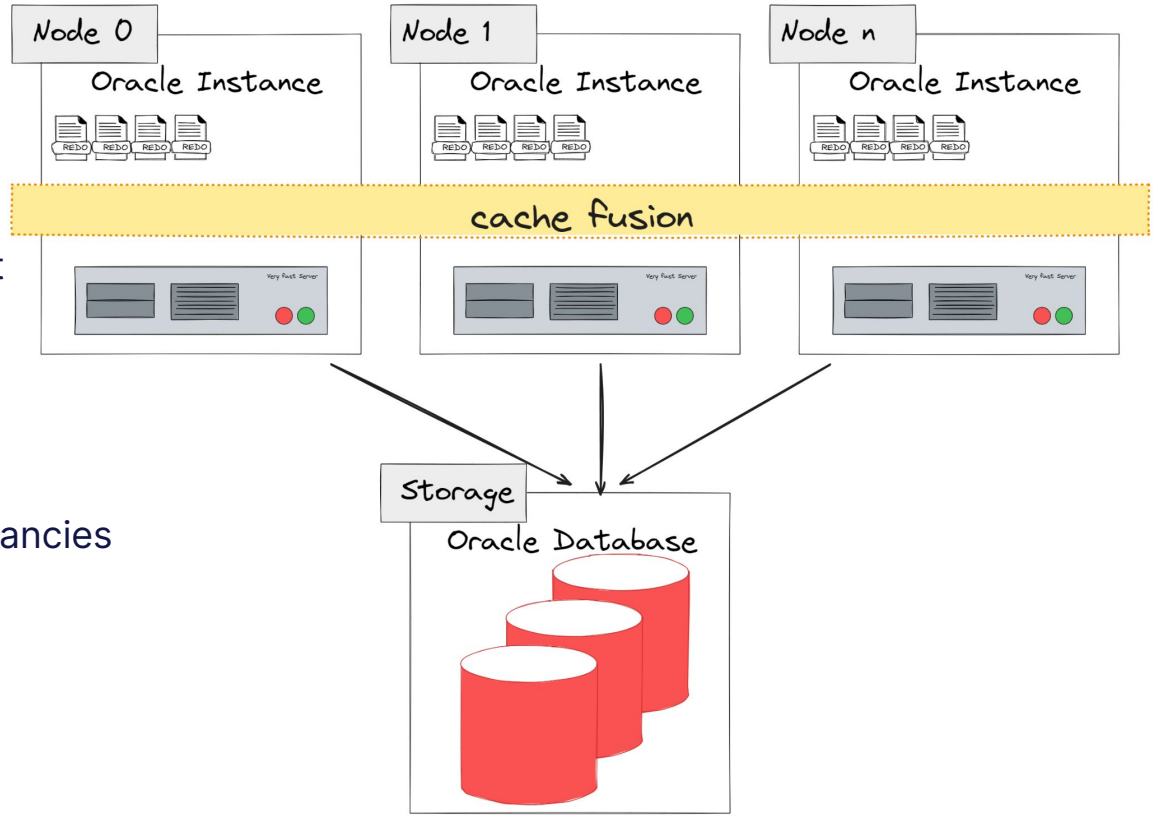
Concepts - The same Idea



Some words about RAC!

ORACLE DATABASE

- Real Application Cluster (RAC)
  - Cache fusion/fast interconnect
  - Compute HA only
  - Scaling capabilities
  - Data outages not covered
    - only over storage redundancies
  - Application dependencies
  - Unnecessary use cases



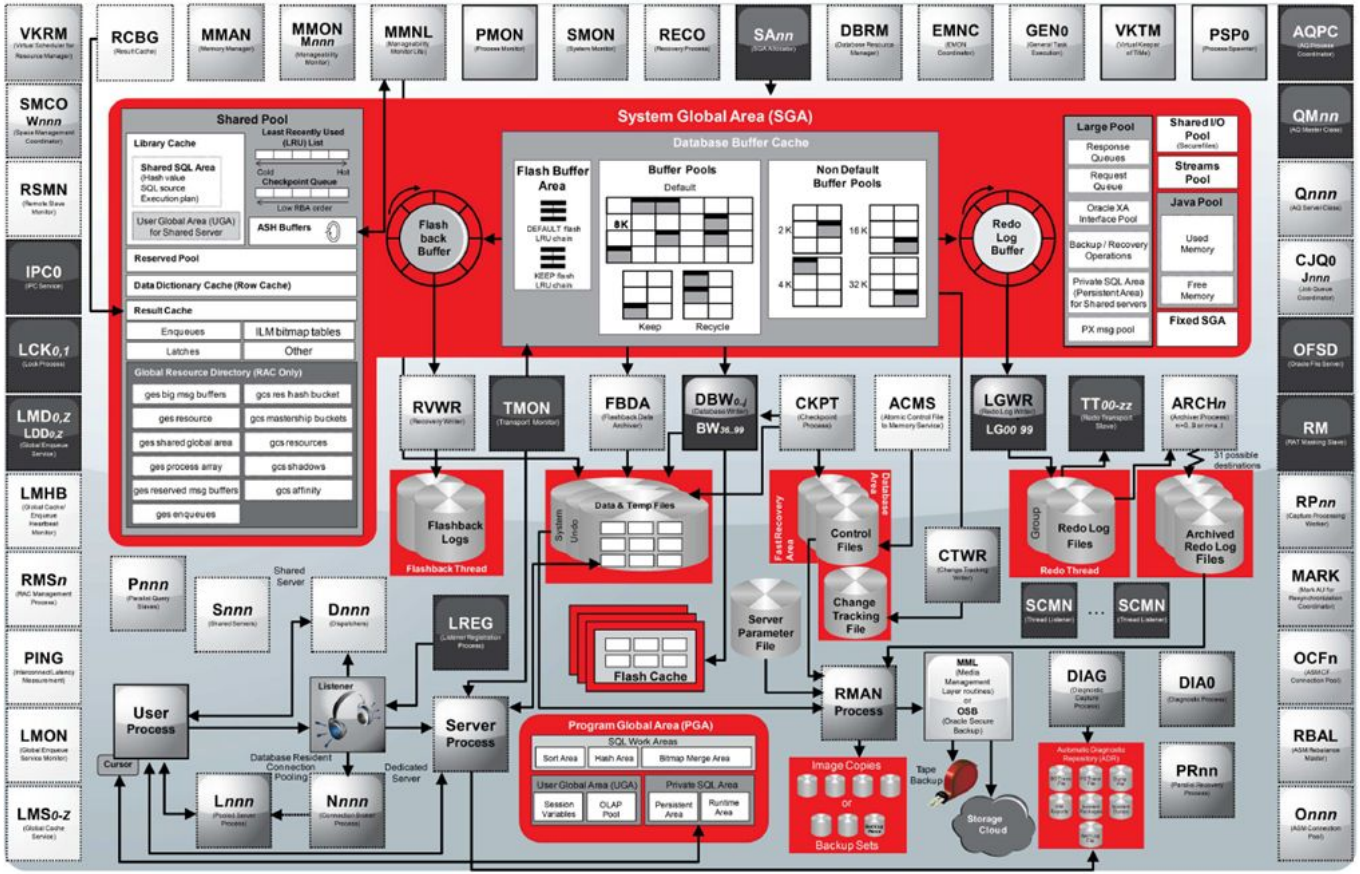
Under the Hood


```
postgres@node0 ~]# systemctl status postgresql-16-core.service

postgresql-16-core.service - PostgreSQL 16 database server
…
   Memory: 23.5M
   CGroup: /system.slice/postgresql-16-core.service
        ├─ 1884 /usr/pgsql-16/bin/postmaster -D /u00/postgres/16/data
        ├─ 1938 postgres: core16: checkpointer
        ├─ 1939 postgres: core16: background writer
        ├─ 1940 postgres: core16: walwriter
        ├─ 1941 postgres: core16: autovacuum launcher
        ├─ 1943 postgres: core16: stats collector
        ├─ 1944 postgres: core16: pg_wait_sampling collector
        ├─ 1945 postgres: core16: logical replication launcher
        ├─ 2016 postgres: core16: walsender replicator 192.168.0.200(55168)
```

```
postgres@node1 ~]# systemctl status postgresql-16-core.service

postgresql-16-core.service - PostgreSQL 16 database server
…
   Memory: 18.7M
   CGroup: /system.slice/postgresql-16-core.service
        ├─2032 /usr/pgsql-16/bin/postmaster -D /u00/postgres/16/data
        ├─2041 postgres: core16: startup recovering 000000020000…
        ├─2256 postgres: core16: checkpointer
        ├─2257 postgres: core16: background writer
        ├─2258 postgres: core16: stats collector
        ├─2259 postgres: core16: pg_wait_sampling collector
        └─3083 postgres: core16: walreceiver streaming 0/39000320
```

Under the Hood

ORACLE DATABASE



Under the Hood

~10 different backend process types

Primary DB

- `walsender`
  - (for each standby DB)

Standby DB

- `walreceiver`
- `recovery process`

Under the Hood

~10 different backend process types

Primary DB

walsender

(for each standby DB)

Standby DB

walreceiver

recovery process

524 different backend processes (21.5)

```
SQL> select name, description from v$bgprocess;

FMON  File Mapping Monitor Process
ACMS  Atomic Controlfile to Memory Server
BRDG  KSRPS Message Bus Bridge
LCK1  Lock Process 1
...
S000  Shared servers
TT03  Redo Transport
M003  MMON slave class 1

524 rows selected.
```

Under the Hood

~10 different backend process types

Primary DB

- **`walsender`**
  (for each standby DB)

Standby DB

- **`walreceiver`**
- **`recovery process`**

...but

"only" ~70-100 in usual environment

```
# ps auxw | grep -c "ora_"
73
```

```
# ps auxwww | grep "ora_"

oracle    429876  0.0  1.6 1282444 61780 ?   Ss   00:06   0:00 ora_pmon_cdb2
oracle    429880  0.0  1.6 1282448 61900 ?   Ss   00:06   0:00 ora_clmn_cdb2
oracle    429884  0.0  1.6 1282196 62800 ?   Ss   00:06   0:00 ora_psp0_cdb2
...
oracle    501689  0.0  1.9 1283216 73588 ?   Ss   00:12   0:00 ora_m002_cdb2
oracle    548563  0.0  2.6 1284664 98448 ?   Ss   00:16   0:00 ora_m003_cdb2
```

Under the Hood

~10 different backend process types

Primary DB

- `walsender`
  (for each standby DB)

Standby DB

- `walreceiver`
- `recovery process`

DMON (Data Guard Broker Monitor)

RSM (Remote System Monitor)

NSV (DG Broker Network Slave)

NSS (Network Server Sync)

MRP (Managed Recovery Process)

RFS (Remote File Server)

LNS (LGWR Network Server)

Building Up - Preparations

WAL archiving not necessary!

Check defaults

```
wal_level = replica || logical
max_wal_senders
max_replication_slots
```

Create replication-user and -slot

Configure pg_hba.conf

Recommended: `track_commit_timestamp`

“ARCHIVELOG Mode” required

Force Logging

Prepare naming

```
DB_NAME
DB_UNIQUE_NAME
```

Create standby Redo Logfiles

Recommended: flashback database

Direct preparation of Primary DB

with Data Guard Broker (21c)

Building Up

Remote copy of primary DB

```
pg_basebackup -R …
```

Check Standby conf

```
primary_conninfo = '…'
primary_slot_name = slot1
```

Start Standby DB

Ready to go….

OracleNet Config

- Prepare tnsnames.ora on both sides
- Configure Listener on both sides
- including handle for Broker

Prepare environment on standby side

- Create mandatory folders
- Create dummy init.ora file
- Start NOMOUNT

Building Up

PostgreSQL

ORACLE DATABASE

...just chillin' with a beer

Connect Recovery Manager (RMAN)

- Target primary DB
- Auxiliary standby DB

Force RMAN duplicate

Or create standby DB with DBCA

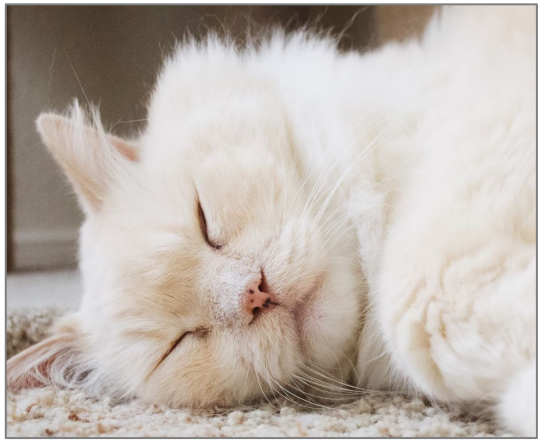
- Database Configuration Assistance (12.2)

Building Up

PostgreSQL

ORACLE DATABASE

...take a nap...



Activate the Data Guard Broker

Connect to the Broker

```
dgmgrl sys/secret@dbname
```

Create a Data Guard configuration

Add standby DB to configuration

Enable configuration

Ready

The Paranoia

Parameters

```
synchronous_standby_names
     <list of standbys>
synchronous_commit
     on | off | local |
     remote_apply | remote_write
```

Protection modes

- Maximum Protection (sync)
- Maximum Performance (async)
- Maximum Availability (*"maybe"* sync)

How many Standbys are possible?

Theoretically up to **262143**

Numerical limit of **`max_wal_sender`**

Cascading!

Since 11.2.0.1 up to **30** standby DBs

(< 11.2 up to 9 standby DBs)

Cascading!

Using standby DB as read replica?

PostgreSQL

Just there by default!

Parameters

```
hot_standby = on | off
```

ORACLE DATABASE

Active Data Guard

As optional License

But cool features, like e.g.

DML Redirection (since 19c)

Or just simply switch to “SQL Apply”

Cluster Handling - Overall

Manual steps

Customized scripts

Command Line Interfaces

- Data Guard Broker (DGMGRL)
- SQLplus

Enterprise Manager Cloud Control

Cluster Handling - Status Check

Logging

View `pg_stat_replication`

(on primary DB)

View `pg_stat_wal_receiver`

(on standby DB)

External solutions

e.g. Grafana dashboards

Views

Data Guard Broker

```
DGMGRL> show configuration;

Configuration - myconfig
Protection Mode: MaxPerformance
Members:
      node0      - Primary database
      node1      - Physical standby database

Fast-Start Failover: DISABLED

Configuration Status:
SUCCESS   (status updated 11 seconds ago)
```

Cluster Handling - Status Check

ORACLE DATABASE

```
DGMGRL> show database node0;

Database - node0

  Role:              PRIMARY
  Intended State:    TRANSPORT-ON
  Instance(s):
       node0

Database Status:
SUCCESS
```

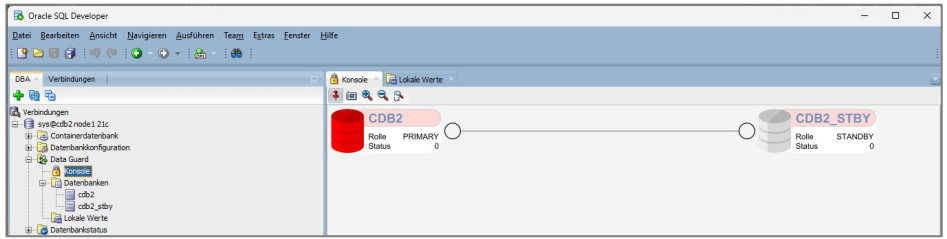
ORACLE DATABASE

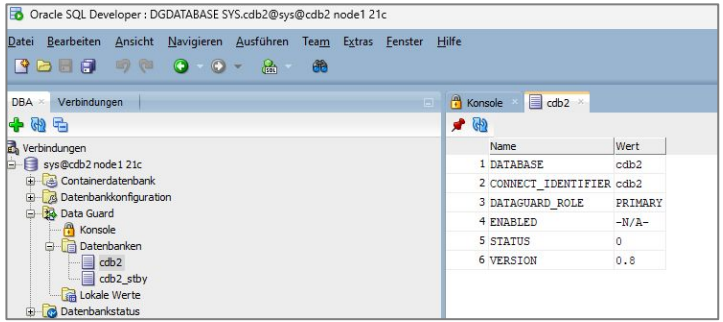
```
DGMGRL> show database node1;

Database - node1

  Role:              PHYSICAL STANDBY
  Intended State:    APPLY-ON
  Transport Lag:     0 seconds (computed 3 second ago)
  Apply Lag:         0 seconds (computed 3 second ago)
  Average Apply Rate: 6.00 KByte/s
  Real Time Query:   OFF
  Instance(s):
       node1

Database Status:
SUCCESS
```

Cluster Handling - Status Check

ORACLE DATABASE

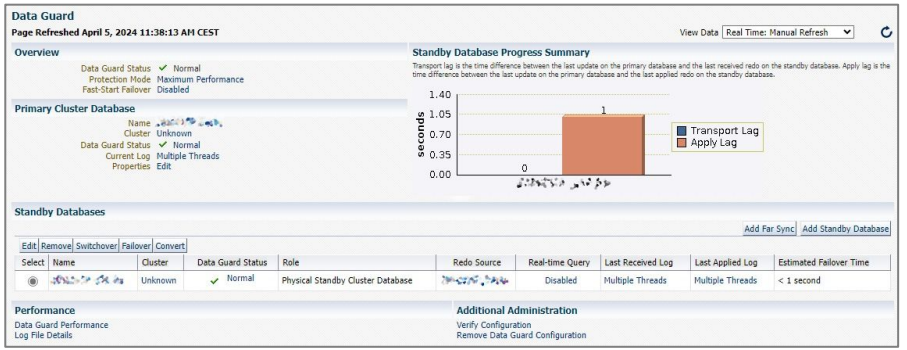
Oracle SQL Developer





ORACLE DATABASE

Oracle Enterprise Manager Cloud Control



Cluster Handling - Failover

Promote Standby DB

```
pg_ctl promote …

promote_trigger_file
```

Take care for the devoted node

- STONITH
- Prevention of split brain

Failover with DGMGRL

```
DGMGRL> failover to node1;
Performing failover NOW, please wait...
Failover succeeded, new primary is "node1"
DGMGRL>
```

Broker detects devoted node...

- ...with working communication
- So no 100% guarantee

Cluster Handling - Automatic Failover?

Not included in core

Customized scripts

External initiation with trigger file

Additional Tools/Extensions

Fast-Start Failover, FSFO (since 10.2)

Independent Observer

Up to 3 Observers (since 12.2)

Included in Oracle Client Package

Pre-defined or individual thresholds

Cluster Handling - Reinstate

Manual reinstate

Drop broken old primary DB

...and yet again...

`pg_basebackup -R …`

Add node as follower

Reinstate of old primary

```
DGMGRL> reinstate database node0;
Reinstating database "node0", please wait...
...
Continuing to reinstate database "node0" ...
Reinstatement of database "node0" succeeded
DGMGRL>
```

Cluster Handling - Switchover

ORACLE DATABASE

Manual steps

- Restrict all connections out
- Stop primary DB consistent
- Check if all WALs are applied on standby

DB

- Change config on primary
- Promote Standby DB

Quite fiddly :-(

Easy with Broker

```
DGMGRL> switchover to node1;
Performing switchover NOW, please wait...
Operation requires a connection to instance …
Connecting to instance "node0"...
Connected as SYSDBA.
New primary database "node1" is opening...
Operation requires start up of instance …
Starting instance "node0"...
ORACLE instance started.
Database mounted.
Switchover succeeded, new primary is "node1"
```

Client Connection Handling

- Core
  - Multiple hosts (libpq)
- External
  - Virtual IP
  - HAproxy
  - PgBouncer
  - Pgpool-II

- Simple
  - Multiple Hosts in TNSnames.ora
  - Services
- Sophisticated / Application based
  - Transparent Application Failover (TAF)
  - Fast Connection Failover (FCF)
  - Transaction Guard (TG)
  - Application Continuity (AC)
  - Transparent Application Continuity (TAC)

Getting closer to Data Guard …

Extension and/or solutions available for providing proper

- Open Source
- Replication management
- Failover-/Switchover handling
- Automatic Failover, Observer-like functionality (witness)

Getting closer to Data Guard …

repmgr

Initial maintained by 2ndQuadrant

GPL license

https://github.com/EnterpriseDB/repmgr

Release v1.0 (May 2010)

Actual Release v5.4.1 (July 2023)

C-based

patroni

Initiated by Zalando (October 2015)

as fork of governor

MIT license

https://github.com/zalando/patroni

Release v1.0 (July 2016)

Actual Release v3.3.0 (April 2024)

Python-based

Getting closer to Data Guard ...repmgr

Cluster overview in repmgr

```
$ repmgr --config-file=/etc/repmgr/14/repmgr.conf cluster show

 ID | Name  | Role    | Status        | Upstream | Location | Priority | Timeline | Connection string
----+-------+---------+---------------+----------+----------+----------+----------+-------------------------------------------------
 1  | node0 | primary | * running     |          | default  | 100      | 11       | host=node0 port=50141 dbname=repmgr user=repmgr
 2  | node1 | standby |   running     | node0    | default  | 100      | 11       | host=node1 port=50141 dbname=repmgr user=repmgr
 3  | node2 | standby |   running     | node0    | default  | 100      |          | host=node2 port=50141 dbname=repmgr user=repmgr
 4  | node3 | witness |   running     | node0    | default  | 0        | n/a      | host=node3 port=50141 dbname=repmgr user=repmgr
```

Example for switchover command

```
$ repmgr --config-file=/etc/repmgr/14/repmgr.conf standby switchover --siblings-follow
```

Getting closer to Data Guard ...repmgr

Event listing in repmgr

```
$ repmgr --config-file=/etc/repmgr/14/repmgr.conf cluster event
 Node ID | Name  | Event                     | OK | Timestamp           | Details
---------+-------+---------------------------+----+---------------------+-------------------------------------------------------
 1       | node0 | child_node_new_connect    | t  | 2022-06-09 23:18:37 | new standby "node1" (ID: 2) has connected
 2       | node1 | repmgrd_start             | t  | 2022-06-09 23:18:32 | monitoring connection to upstream node "node0" (ID: 1)
 1       | node0 | repmgrd_start             | t  | 2022-06-09 08:06:26 | monitoring cluster primary "node0" (ID: 1)
 1       | node0 | child_node_disconnect     | t  | 2022-05-24 13:35:25 | standby node "node1" (ID: 2) has disconnected
 1       | node0 | child_node_new_connect    | t  | 2022-05-23 20:38:02 | new standby "node1" (ID: 2) has connected
...
 4       | node3 | witness_register          | t  | 2022-04-11 14:32:40 | witness registration succeeded; upstream node ID is 1
 3       | node2 | standby_follow            | t  | 2022-04-11 14:32:39 | standby attached to upstream node "node0" (ID: 1)
 1       | node0 | standby_switchover        | t  | 2022-04-11 14:32:37 | node "node0" (ID: 1) promoted to primary, node "node1"...
 1       | node0 | standby_promote           | t  | 2022-04-11 14:32:15 | server "node0" (ID: 1) was successfully promoted to primary
 2       | node1 | child_node_new_connect    | t  | 2022-04-11 14:15:13 | new witness "node3" (ID: 4) has connected
 4       | node3 | witness_register          | t  | 2022-04-11 14:15:08 | witness registration succeeded; upstream node ID is 2
```

Getting closer to Data Guard …Patroni

Cluster overview

```
$ sudo patronictl -c /etc/patroni/patroni.yml list

+ Cluster: pgd14A   (7011110722654005156)  -----------+
| Member | Host  | Role    | State   | TL | Lag in MB |
+--------+-------+---------+---------+----+-----------+
| node0  | node0 | Leader  | running | 11 |         0 |
| node1  | node1 | Replica | running | 11 |         0 |
| node2  | node2 | Replica | running | 11 |           |
+--------+-------+---------+---------+----+-----------+
```

Example for switchover

```
$ sudo patronictl -c /etc/patroni/patroni.yml switchover
```

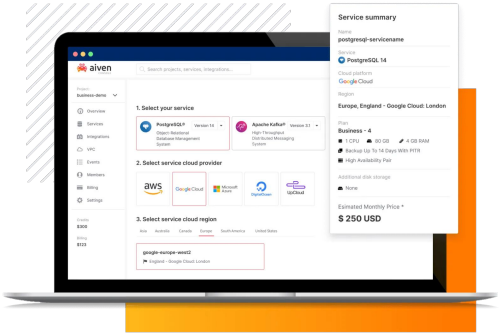
Getting closer to Data Guard …Cloud :-)

Use a fully managed PostgreSQL with

e.g. “Aiven for PostgreSQL”

having all necessary bells and whistles and

with up to 2 read replicas for high availability



What about logical replication?

Same basic setup, except...

```
wal_level = logical
```

Schema transfer

- Script
- pg_dump

```
CREATE PUBLICATION / SUBSCRIPTION
```

Or use Extension pglogical!

SQL apply

- Setup physical replication
- Stop redo apply on standby DB
- Prepare primary for logical standby DB
- Transition to logical standby DB
- Open logical standby DB

What about Multimaster Replication?

- Thoughts
  - Only with logical replication
  - Oracle RAC is NOT Multimaster Replication
  - Often more a wish than a real need
  - Application dependencies
    - Of course the same with Oracle RAC or GoldenGate

What about Multimaster Replication?

Have seen manual implementations

«On your own risk!»

pg_logical

Commercial Solutions

Oracle Streams

- Deprecated with 12c
- Desupported with 19c

Oracle Golden Gate

- Additional product
- Oracle ↔ Oracle
- Oracle ← (PostgreSQL, MySQL,...)

Conclusion

Pro

- Lightweight configuration, architecture
- No archived WAL files necessary
- Very flexible with logical replication
- Extensibility

Cons

- Fiddly Failover/Switchover handling
- No proper native connection handling
- No automatism

Pro

- Proper Automatic Failover solution included
- Handling with Broker CLI, native tooling
- Interesting features

Cons

- Not that easy, still straight forward
- Only for Enterprise Edition
- Active Data Guard additional license
- Poor logical capabilities

Final Words

Core replication technologies are quite equal

Oracle Data Guard Licensing!

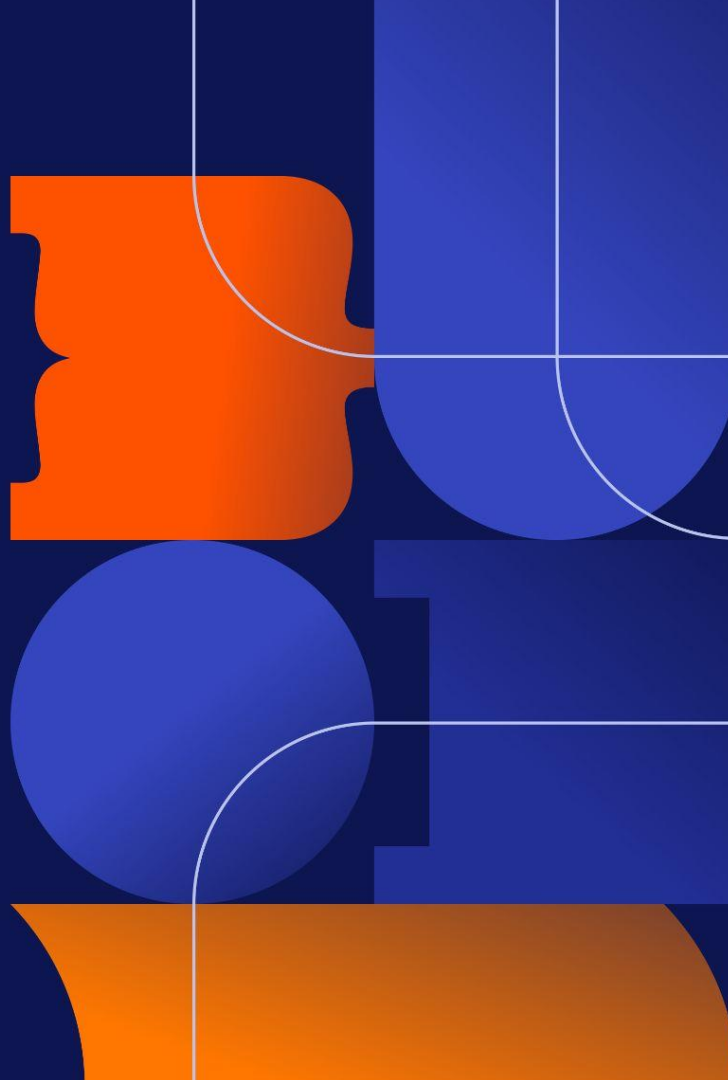
…but Oracle brings a bunch of features

PostgreSQL “Core only” not practical in critical environments

…but extensibility brings flexibility and lots of options

Solutions like patroni, repmgr, etc. nearly on par with Data Guard

The trusted open source data platform for everyone



One data platform for your cloud needs

Event streaming	Event stream processing	Relational databases	Key-value database	Wide column database	Data warehouse	Time series database	Search engine	Data visualization
Aiven for **Apache Kafka®** and **Kafka® Connect**	Aiven for **Apache Flink®**	Aiven for **PostgreSQL®**, Aiven for **MySQL**	Aiven for **Redis®**	Aiven for **Apache Cassandra®**	Aiven for **ClickHouse®**	Aiven for **M3**	Aiven for **OpenSearch®**	Aiven for **Grafana®**

STREAM | **STORE** | **ANALYZE**

Host

aws
Google Cloud
DigitalOcean
Microsoft Azure
Bring your own cloud

Deploy

Terraform
Kubernetes
API REST API
CLI Aiven CLI
Aiven Console

Integrate

Datadog
Prometheus
AWS CloudWatch
GCP Monitoring
MongoDB
AWS S3
GCP BigQuery
Couchbase
Snowflake
splunk> Splunk
Sumologic
Debezium
GCP Pub/Sub
GCP Storage

Customers


okta


snyk


DOORDASH


priceline


fiverr.


Norauto


DECATHLON


GTL


ACTIVISION
BLIZZARD


MIRAKL


GOV.UK


goto financial


spare


Schibsted


TOYOTA


paf


CONRAD


adeo


ometria


WÄRTSILÄ