Postgres Cluster and Multimaster

PROFESSIONAL Postgres

postgrespro.ru

Ivan Panchenko
Postgres Pro

Cluster definition: several DBs working as one

- Redundancy
- Sharding
- Parallel query processing
- Failover
- Dynamic reconfiguration
- Cluster-wide data consistency (distributed transaction)
- Read scalability
- Write scalability
- Reliability

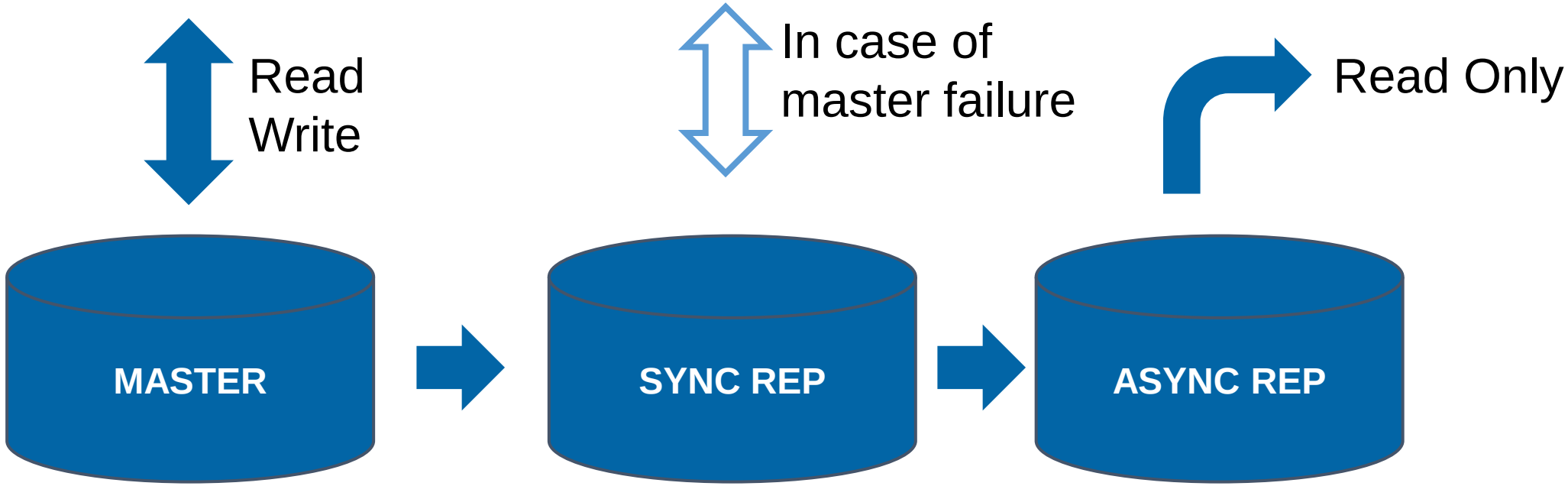
First attempts (~2006)

- Trigger based (SLONY/Londiste) or application level replication
- No real consistency : needs verifications
- No real failover (data loss possible)
- Read scalability
- Only application level sharding for write scalability

Mainstream evolution

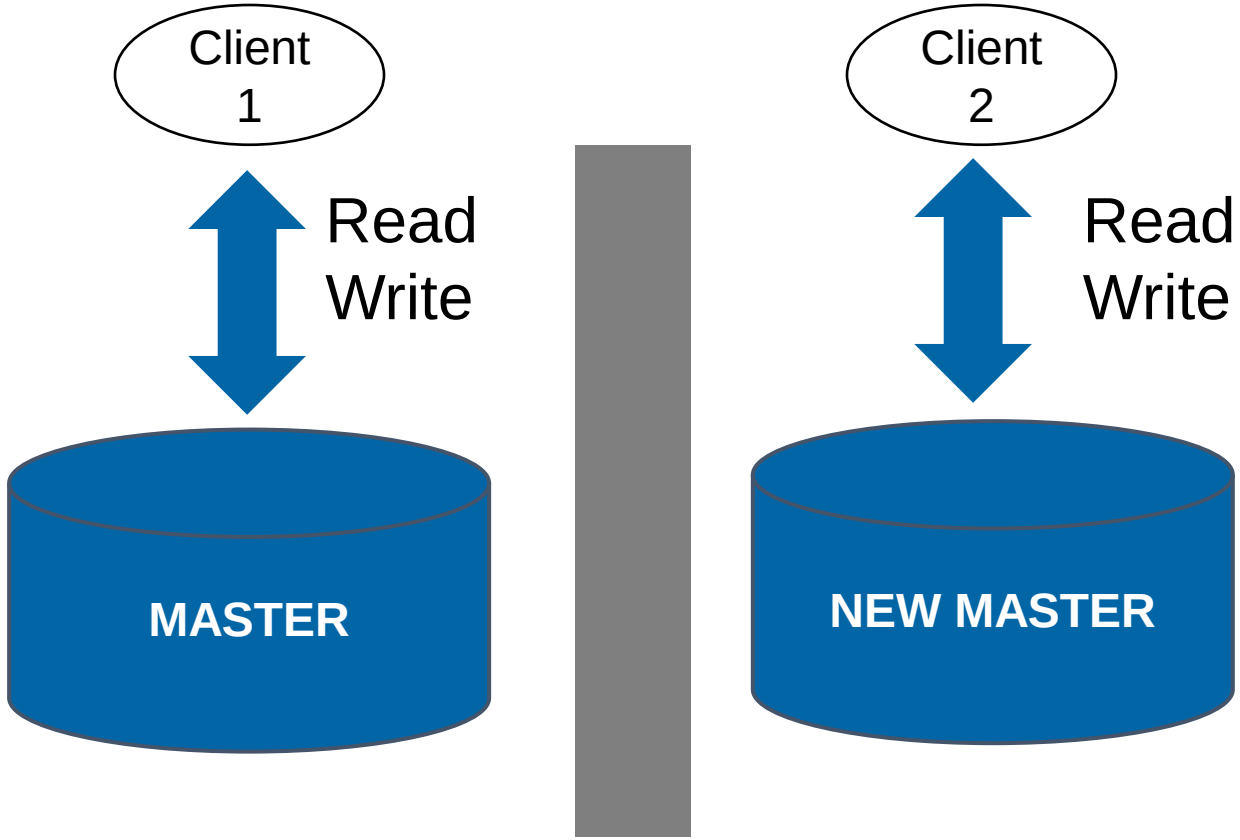
- WAL (write ahead logs)
- WAL shipping
- WAL streaming: on-line async replication.
- Logical replication: is more flexible
- Synchronous replication: necessary to exclude data loss
- HA provided by external tools or performed manually

Asymmetric (Single master) clusters



Reliable cluster: main challenge

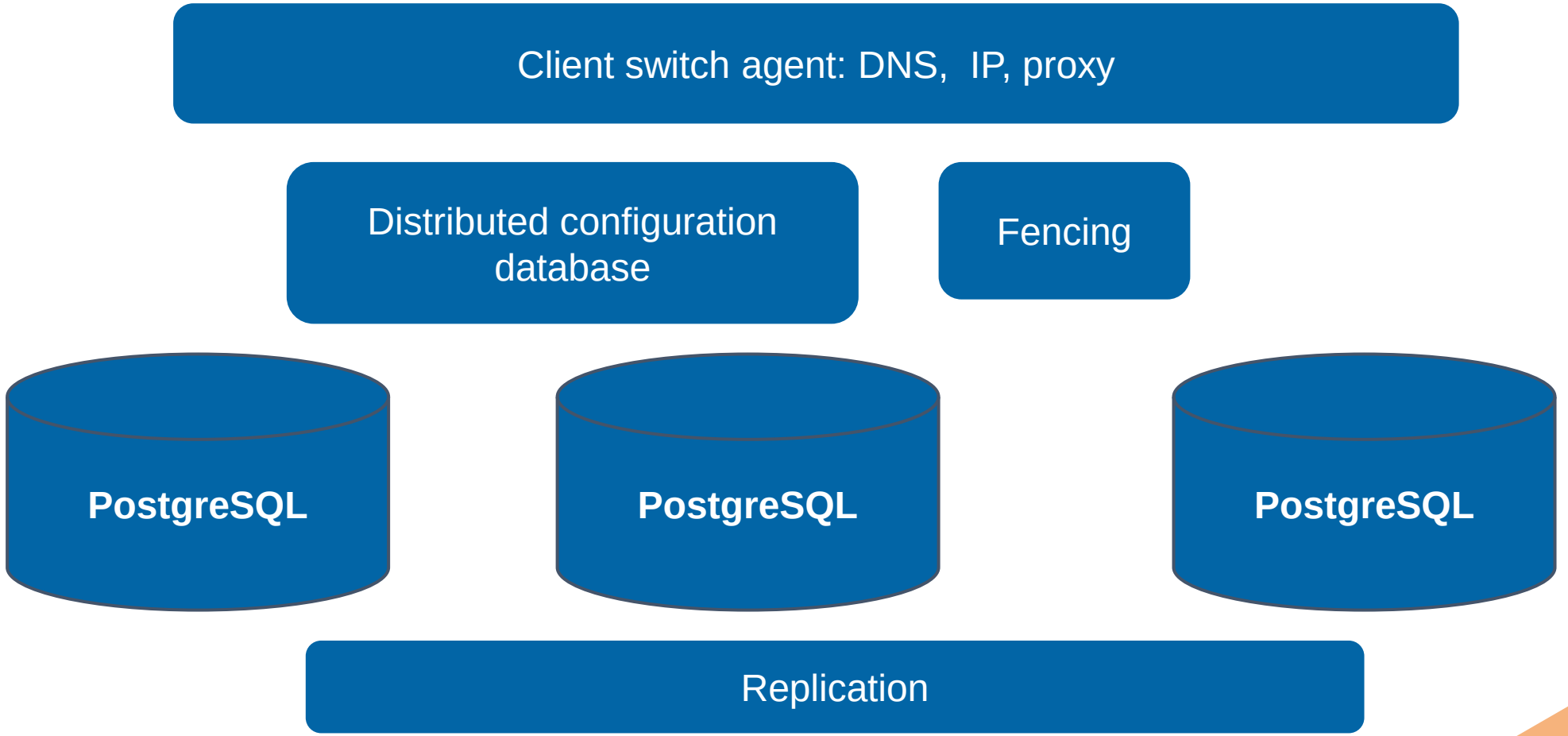
- Split-brain problem:



Temporary or long lasting Internal connection failure:

- Sync replica promotion
- Some clients connect to Old master
- Some clients connect to New master
- Chaos grows

Reliable cluster architecture



Reliable cluster: present solutions

Solution	Origin	License	Basis	Split brain protection
Patroni (engine, not a solution)	Zalando	MIT	Etcd OR zookeper OR consul	May be
PAF	Dalibo	Postgres	Corosync/ Pacemaker	Yes
Repmgr	2nd Quadrant	GPLv3	-	Should be
Postgres Pro	Postgres Pro	Commercial	Corosync/ Pacemaker	Yes

Corosync/Pacemaker

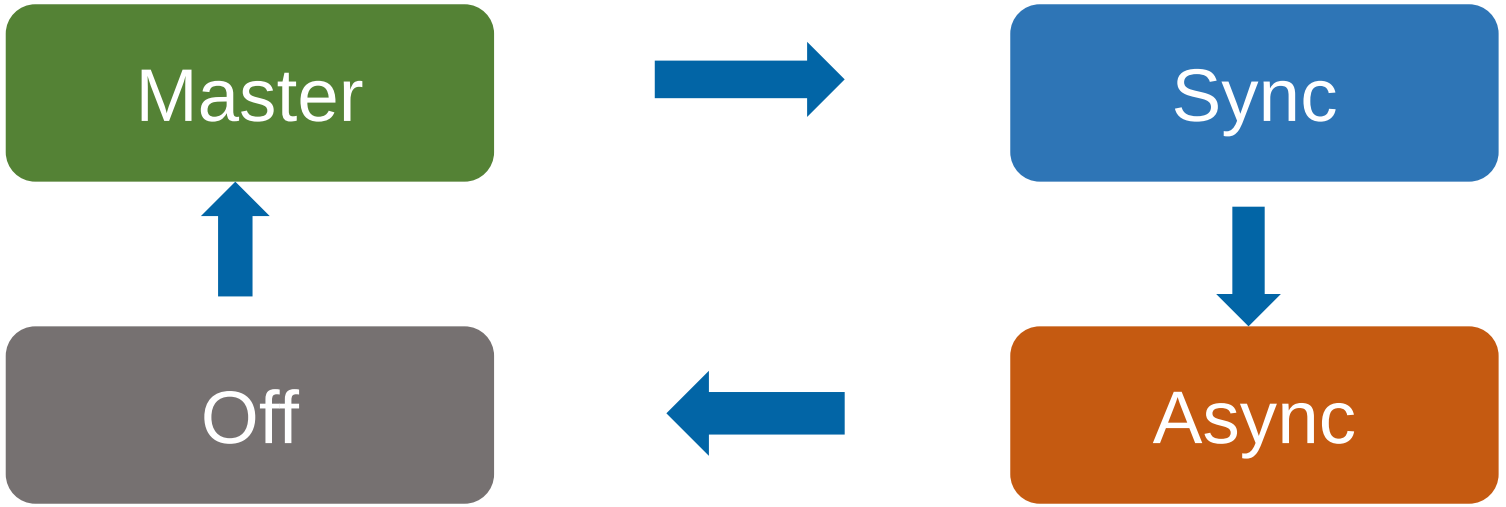
- Developed by Red Hat.
- Resource Agent – an interface utility to manage a resource. Must implement the following commands:

1. start
2. stop
3. status
4. monitor
5. promote
6. demote

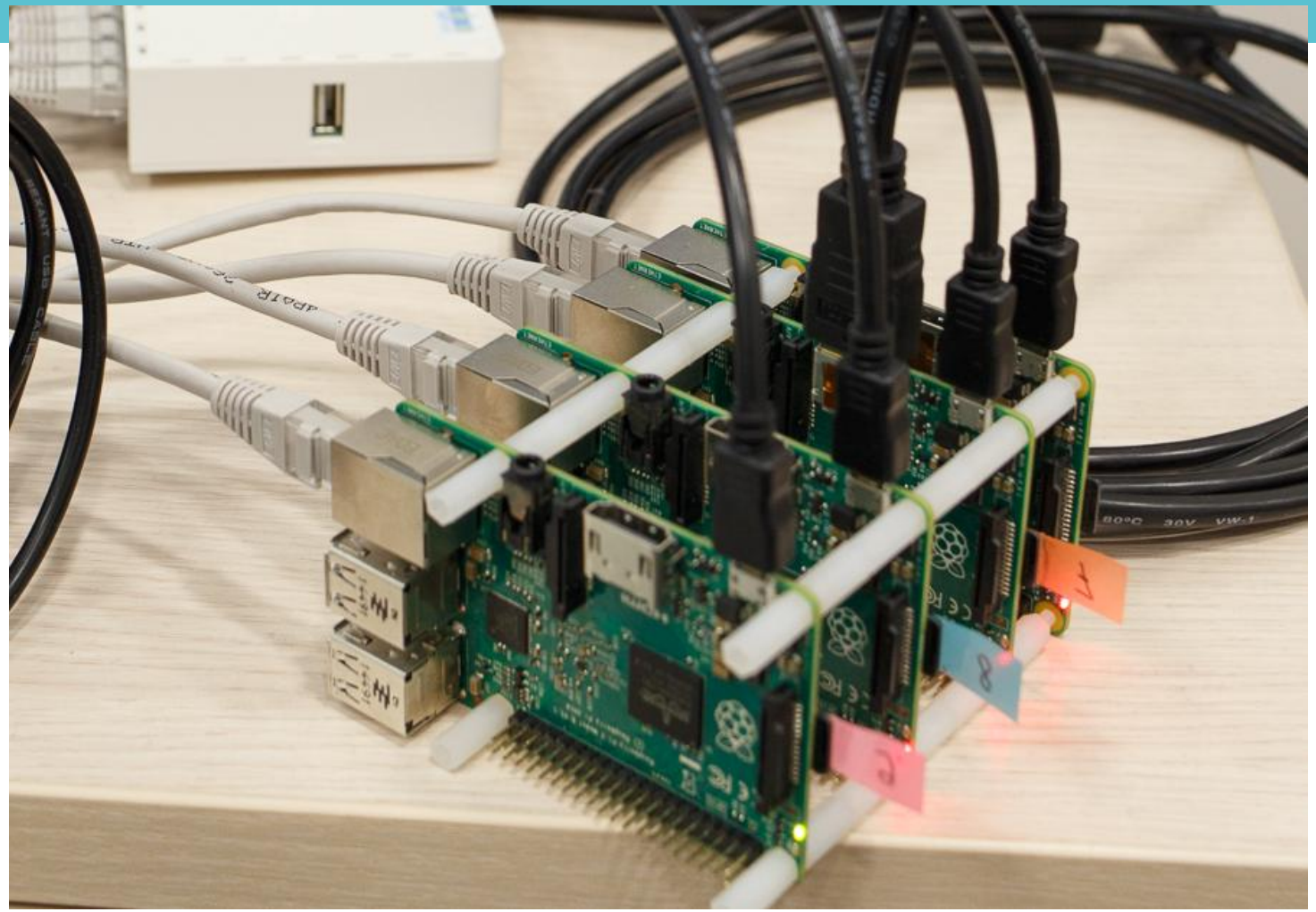
Cluster practice

- Diagnostics
- Failover
- Synchronous replica switchover
- Asynchronous replica switchover
- Recovery
- Split brain
- Deleting node
- Adding node

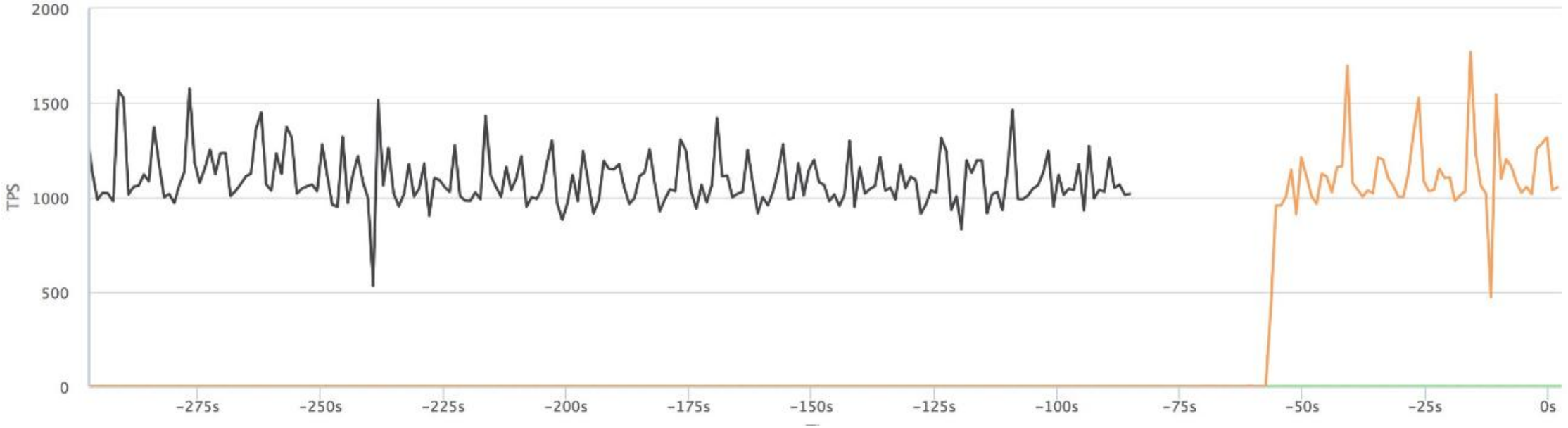
Cluster status transitions



A minimal cluster



Failover visualised

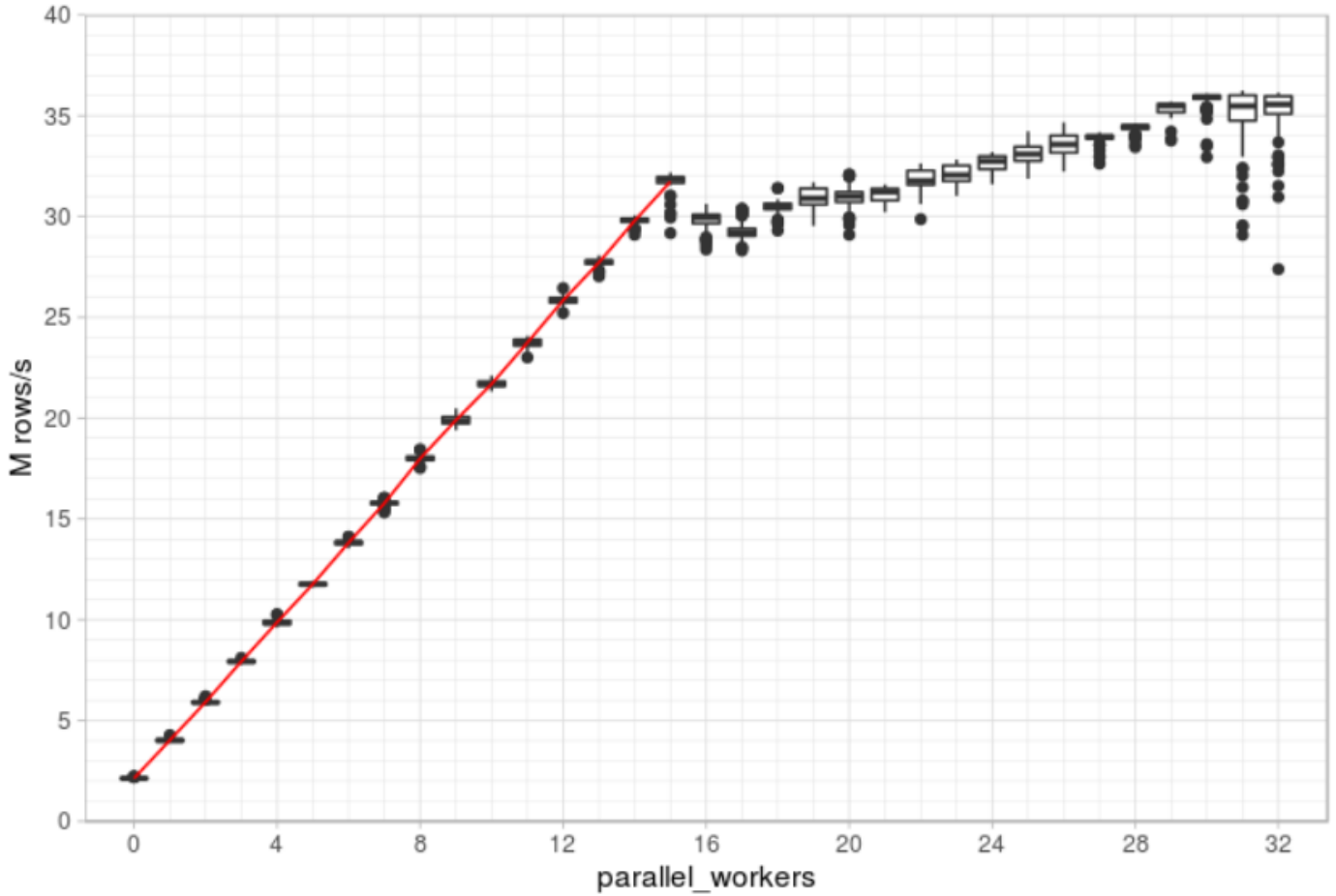


OLAP clusters

- Citus DB
- Green Plum

- Technology: Postgres fork
- License: Commercial; moving to open source
- Transaction consistency: None
- Scalability: good

Read scalability in vanilla Postgres



Single host 9.6:
Parallel query execution

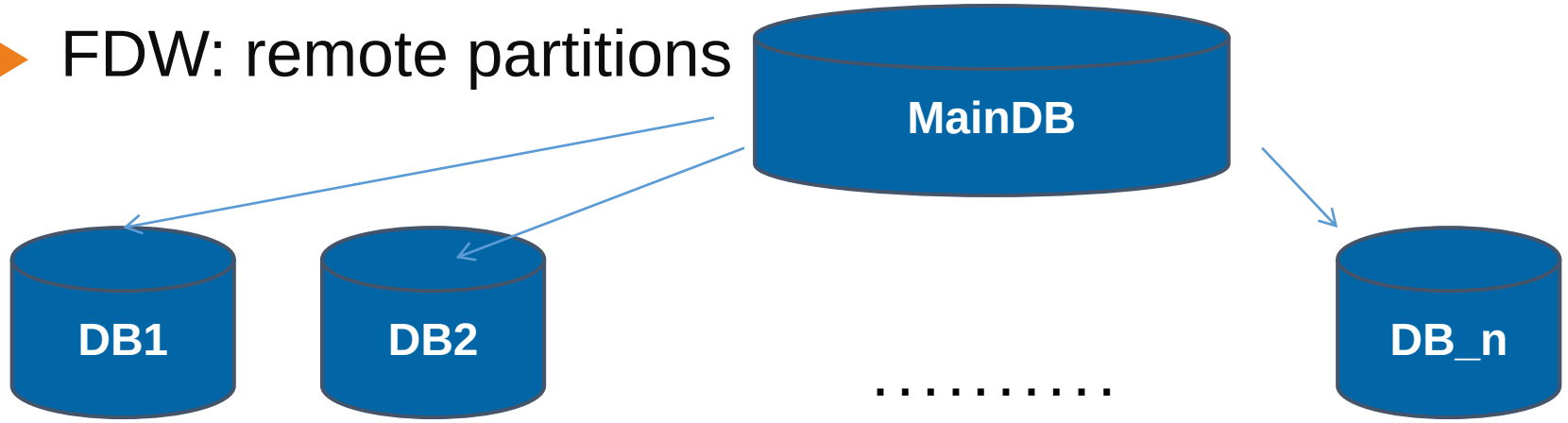
Read scalability in distributed database with sharding

- Table partitioning
- FDW: remote partitions

- No transaction integrity ☹

Read scalability in distributed database with sharding

- Table partitioning
- FDW: remote partitions



- No transaction integrity ☹

http://www.cybertec.at/experimenting-scaling-full-parallelism-postgresql

1 billion rows per second : Hans-Juergen Schoenig

Write-scalable clusters

- Postgres XC (dev.since 2010)
- Postgres XL (2014)
- Postgres X2 (2016)

- Technology: Postgres fork
- Write scalability: some
- Parallel processing: yes
- Failover: yes
- Transaction consistency: not enough

BDR (Bidirectional replication)

- Logical-replication based
- Post-commit replication
- Each transaction replicated to each node

- Technology: Postgres fork; moving to PostgreSQL
- License: Commercial; moving to open source
- Transaction consistency: None
- Read scalability: good

Postgres Pro Multimaster

- Logical replication based
- Each transaction replicated to each node
- Distributed transaction manager
- Internal failover engine
- Technology: Postgres extension
- License: Commercial; some parts - open source
- Transaction consistency: Yes
- Read scalability: good
- Write scalability: will have

Part of Postgres Pro Enterprise commercial distribution

Easy in use cluster

- No performance penalty for reads.
- Transaction can be issued to any node.
- No special actions required in case of failure (excl. client reconnect)

Design goals

- Identical data on all nodes
- Possibility to have local tables
- Maximum Postgres compatibility
- Writes to any node
- Fault tolerance
  - Next step: add sharding for write scalability

Transaction manager requirements

- No single point of failure

+: Spanner, Cockroach, Clock-SI

- : Pg-XL

- Read-only transactions from a single node without communication between nodes

+: SAP HANA, Spanner, Cockroach, Clock-SI

- : Pg-XL

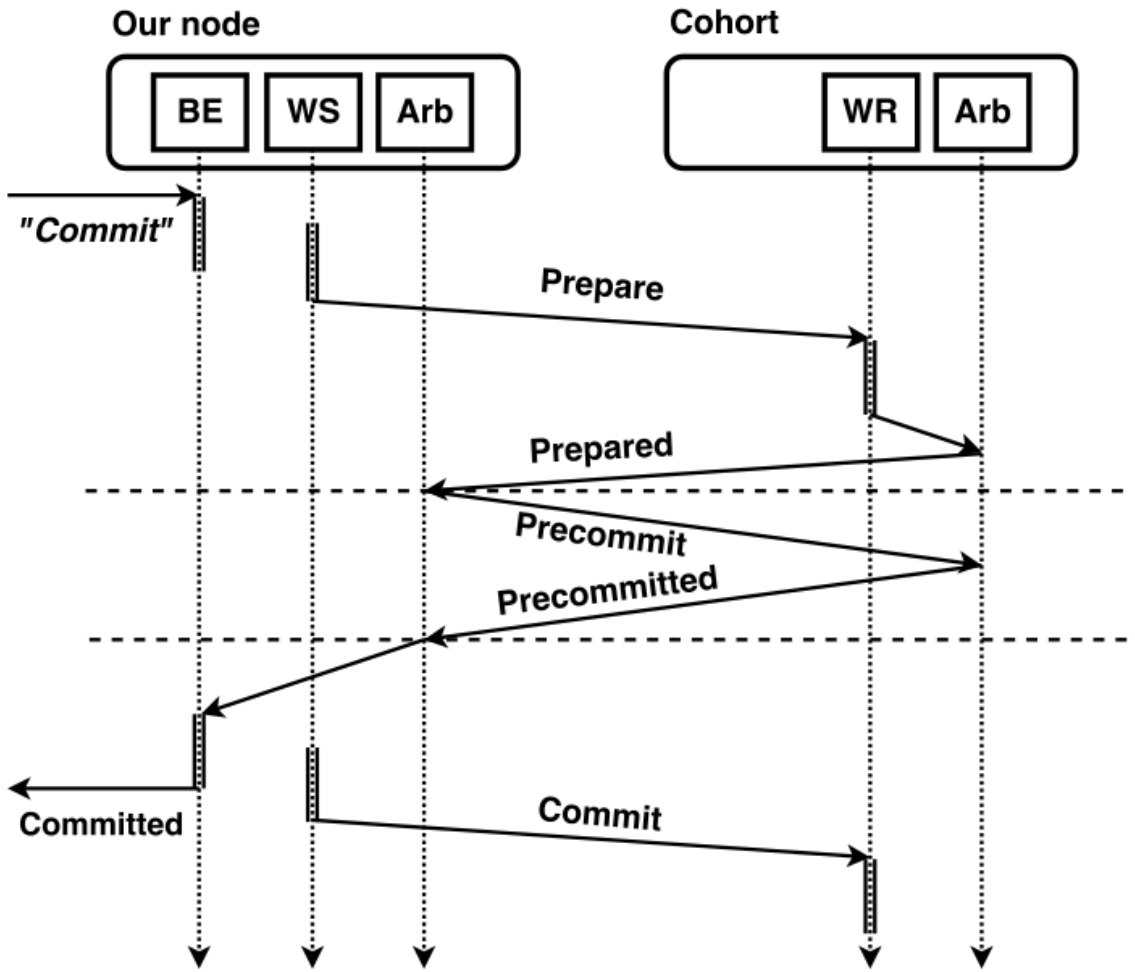
Why logical replication?

- Already existing open source solution by 2nd Quadrant
- Very flexible, i.e:
  - Can skip tables
  - Replicates between different versions

Why logical replication?

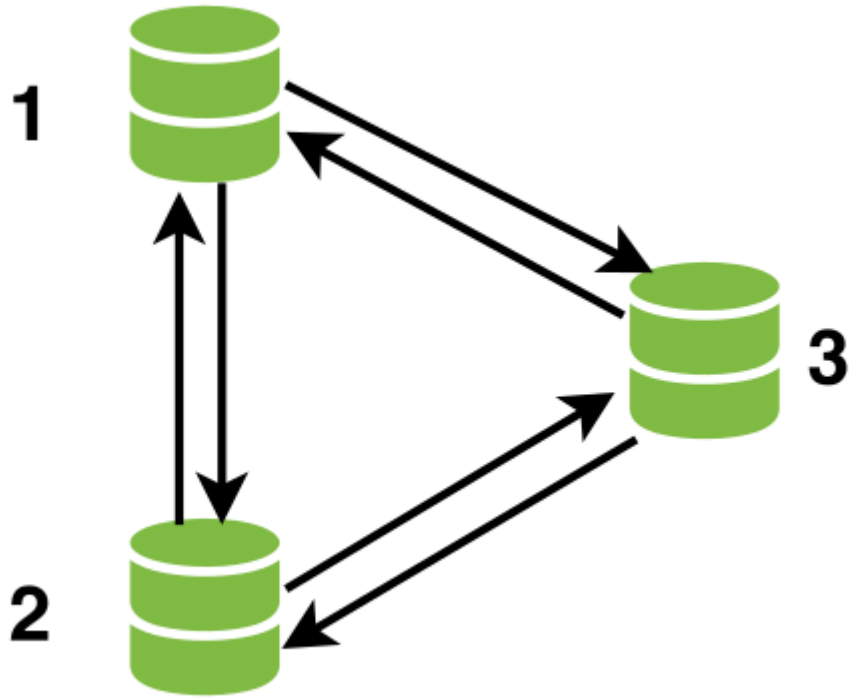
- Already existing open source solution by 2nd Quadrant
- Very flexible, i.e:
  - Can skip tables
  - Replicates between different versions

Transaction implementation

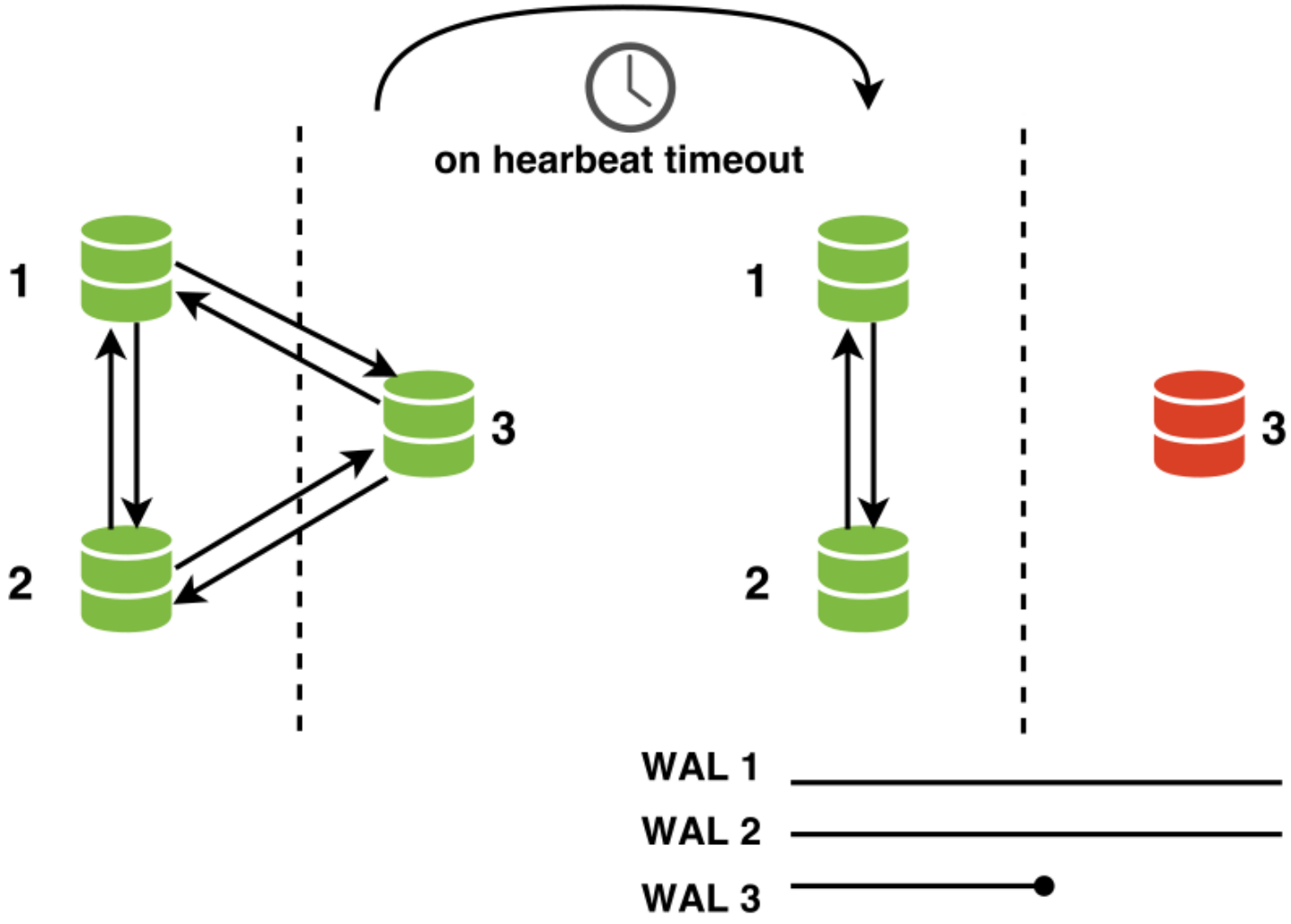


BE – backend,
WS – Walsender,
Arb – Arbiter,
WR – Walreceiver

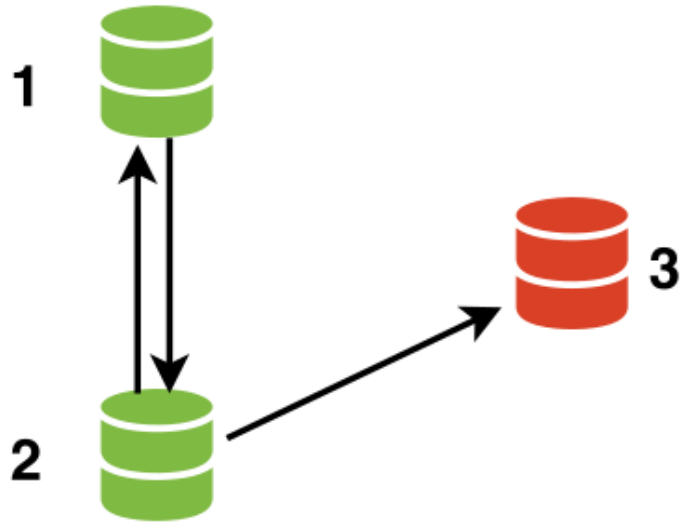
Normal work

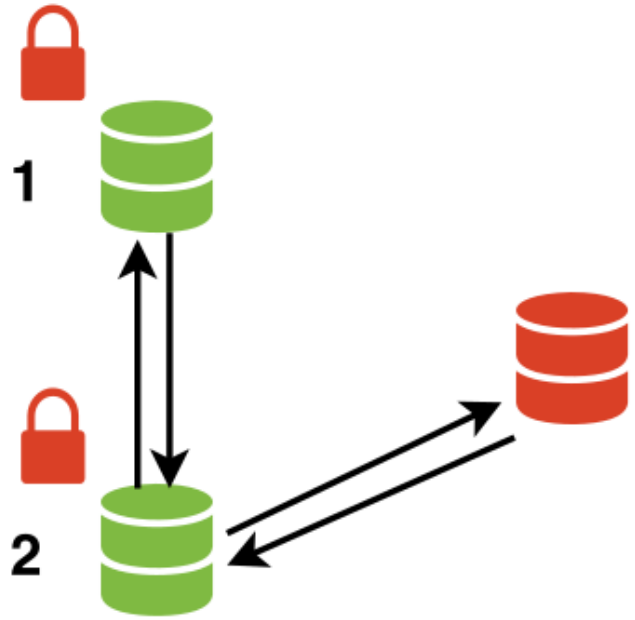


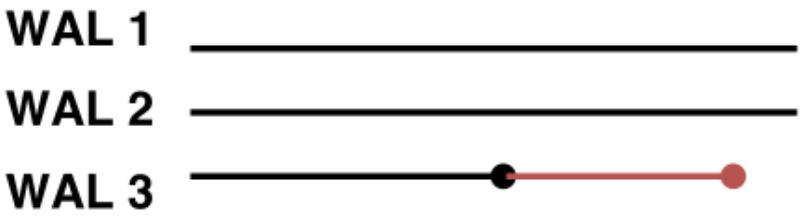
Internal network split

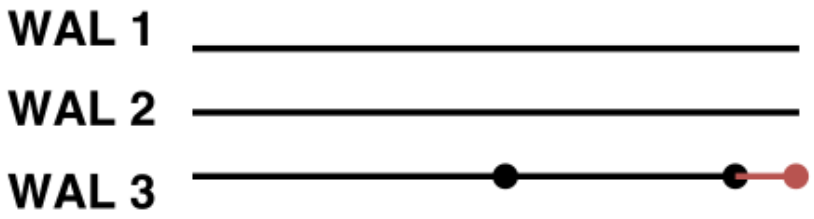


Internal network split: recovery

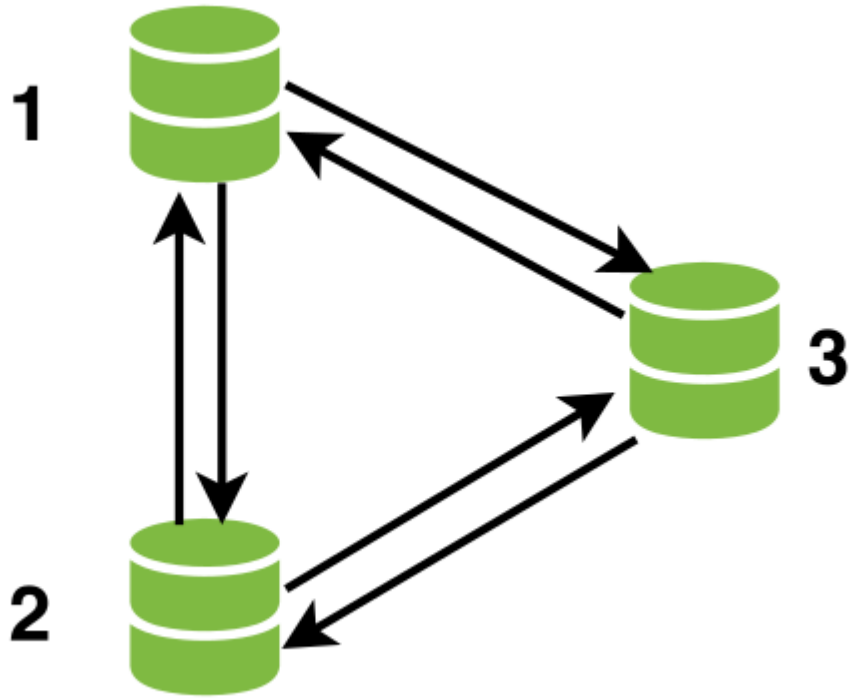








Normal work after recovery



Failures tested

- Node stop-start
- Node kill-start
- Simple network split
- Asymmetric network split
- Shift time
- Change clock speed on nodes (work in progress)

Performance

- Read-only performance is the same as in single instance
- Commit takes more time (two network roundtrips).
- Logical decoding slows down big transactions (to be fixed soon)

Information

- Postgres Pro documentation: http://postgrespro.com/docs
- PgConf.RU : international conference in Moscow – March 15-17.
  - http://pgconf.ru/
  - Russian and English with simultaneous translation
  - 7 Tutorials; > 50 talks

Postgres Miktzoanim

http://postgrespro.co.il/

+972 54 305 7642

info@postgrespro.co.il

PROFESSIONAL Postgres

postgrespro.ru