PGConf.Russia 2018

INGRAM MICRO®

PostgreSQL High Availability Cluster for Enterprise

Vadim Yatsenko

Sergey Kim

About Ingram Micro Company

1,400+ Vendors

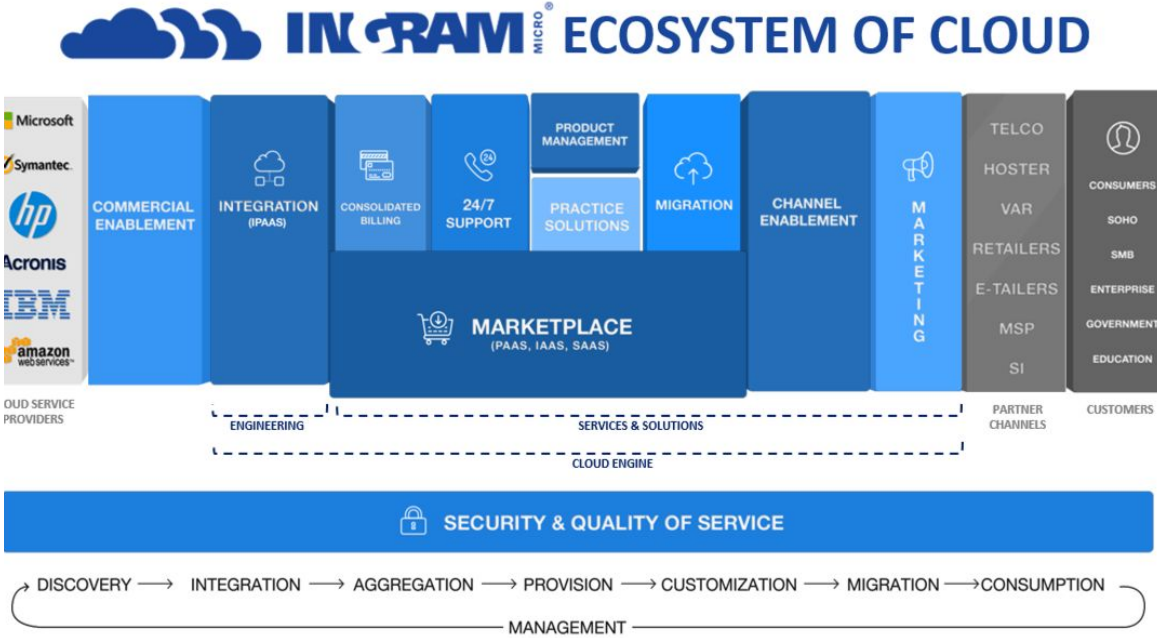
INGRAM MICRO®

200,000+ Customers

- Sales and Marketing
- Financing
- Technical & Pre-Sales Support
- Inventory Management
- Business Intelligence & Tools
- Vendor Relations
- Supply Chain Expertise
- Managed Services
- Configuration
- Communities
- Deep Understanding of Global and Local Requirements

Value-Added Resellers (VARs), Managed Service Providers (MSP), Enterprises, Government, Healthcare, etc.

Ingram Micro Cloud



At Ingram Micro Cloud, we develop and support a hyper scalable digital platform for the distribution of cloud services.

Differences for Products

Single project

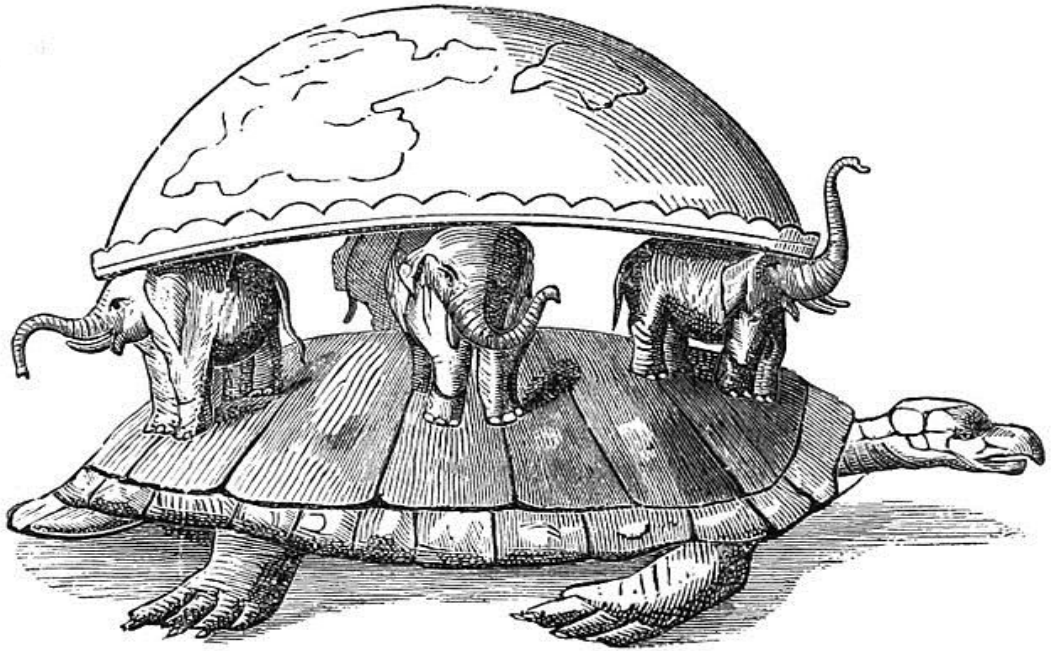
- DBAs and engineers
- DBA maintenance
- Online access to logs and monitoring
- Single configuration and project architecture

Enterprise Project

- Engineers
- Automatic or by cron maintenance
- Logs and monitoring on request (support)
- Multiple Product installation options

High availability?

- Failover and Failback
- Resilience to all the possible network partitioning problems
- Solve the split-brain problem
- Data Consistency
- Availability

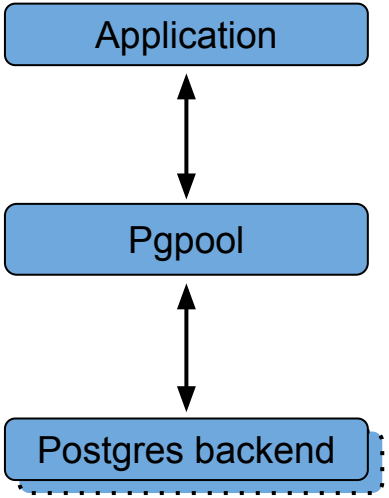


CAP Theorem?

Pgpool-II

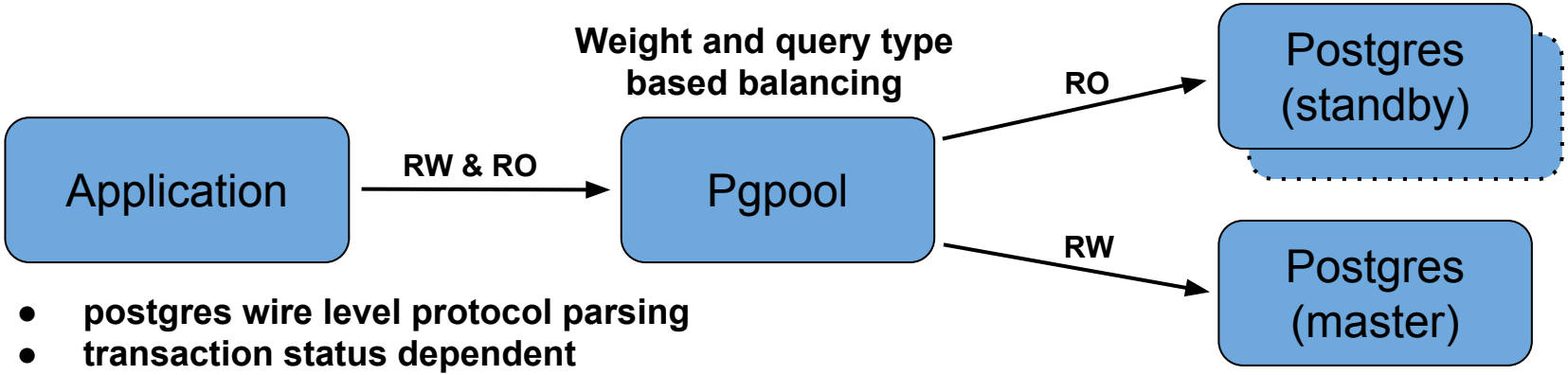
HA based on Pgpool-II

- Load Balancing
- Watchdog
- Master/Slave detection and promotion
- Query Replication
- Connections pooling / limiting exceeding connections



All strongly desired fruitful features

Load Balancing

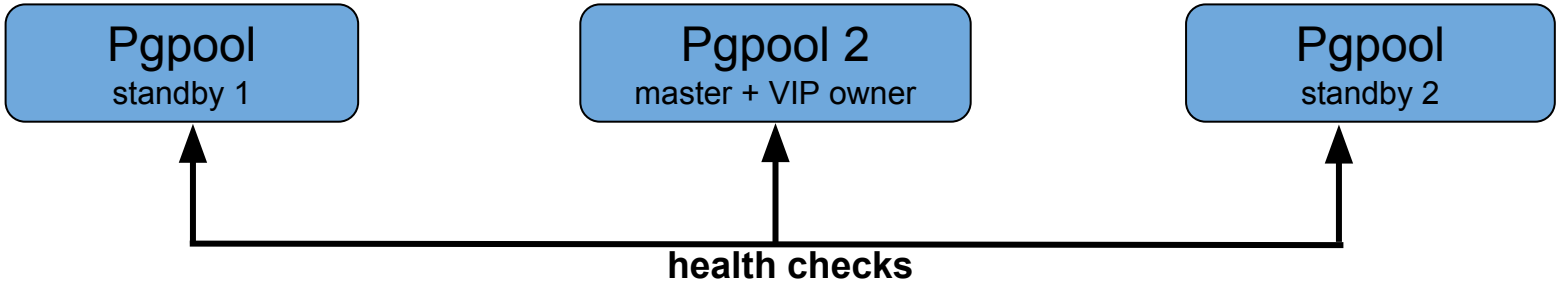


- postgres wire level protocol parsing
- transaction status dependent
- query type (RW/RO) dependent
- PG backend weight based

- scaled load (+ 50% performance for long queries)
- reduced loading of master backend

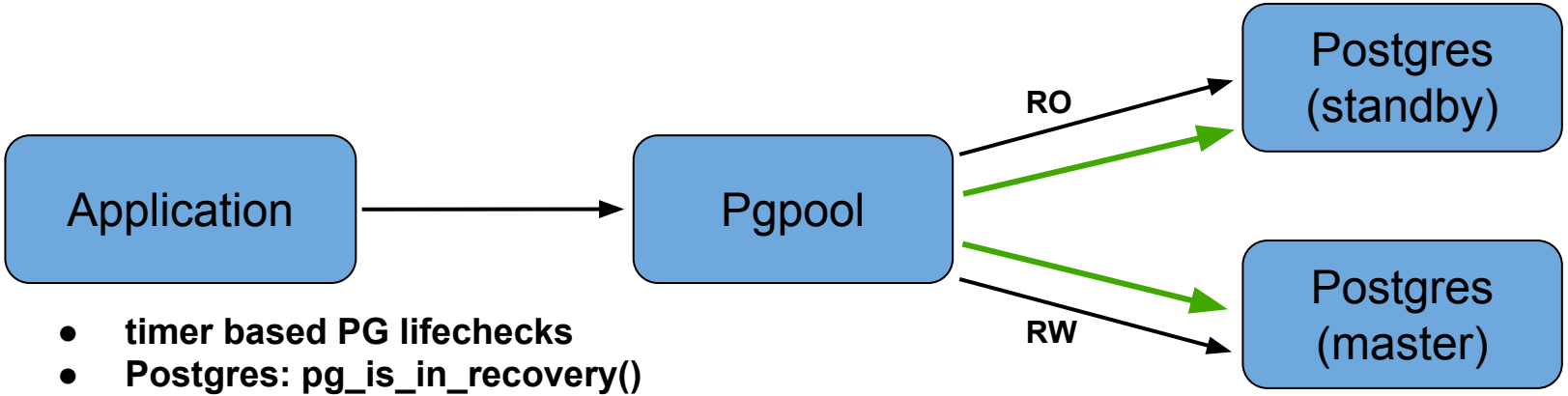
- balancing is session dependent & fixed
- impacted performance (25%) for PK based SELECTs
- impacted performance of modifications
- ...depends on certain series of SQLs

Watchdog



- **Distributed quorum**
- **Split brain resolve / re-election**
- **VIP owning by Master Pgpool**
- **PG backend states detection and interexchange**

Master/Slave detection and promotion #1

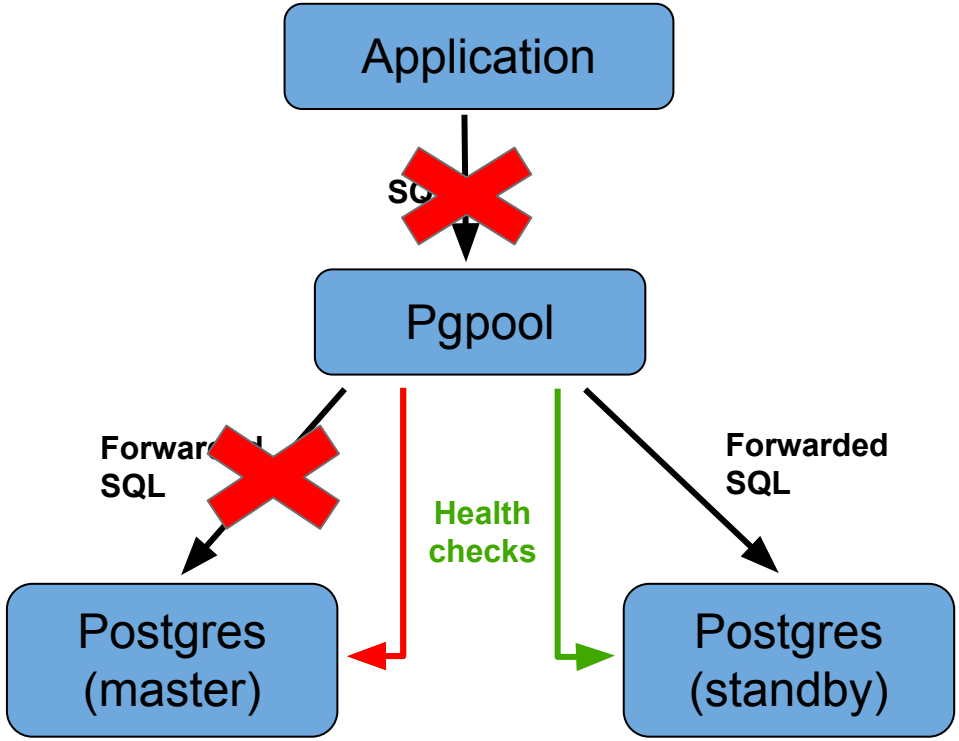


- timer based PG lifechecks
- Postgres: pg_is_in_recovery()
- Pgpool: failover_command

- background lifechecks
- application clients dont care about failed PG backend
-

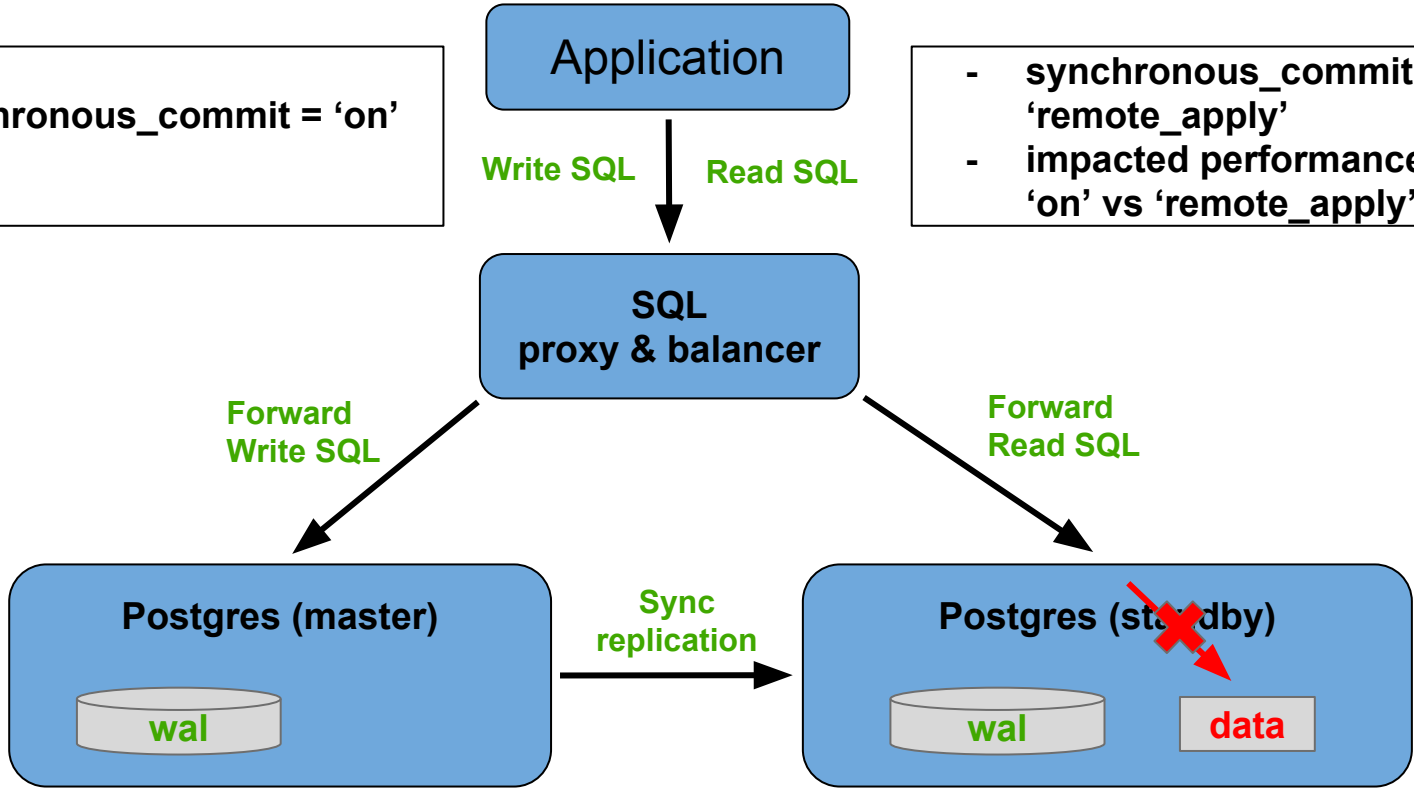
- time consuming quorum
- disconnection of balanced clients
- promotion is shell based

Master/Slave detection and promotion #2



Balancing is not always possible

- synchronous_commit = ‘on’



- synchronous_commit = ‘remote_apply’
- impacted performance: ‘on’ vs ‘remote_apply’ =>1.5

Discovered problems in Pgpool-II

Issue #	Description
329	Newly elected(promoted) Pgpool instance does connect to any backend
306	Standby Pgpool steals back MASTER status
295	Watchdog service fails to start
294	Pgpool service fails to start on the load of OS
281	"segmentation fault" when execute pcp_attach_node
280	Stack smashing detected
271	Balanced SELECT requests after started transaction and performed modification
248	Pgpool doesn’t restore backend connections
228	Depromoted Master Pgpool doesn’t de-escalate IP in case network restored
215	Promoted Pgpool doesn’t escalate ip in case of another node unavailability

Issues tracker: https://www.pgpool.net/mediawiki/index.php/Bug_tracking_system

Pgpool-II

Pros

Cons

Usage of Pgpool-II is under consideration...

Possible solutions?

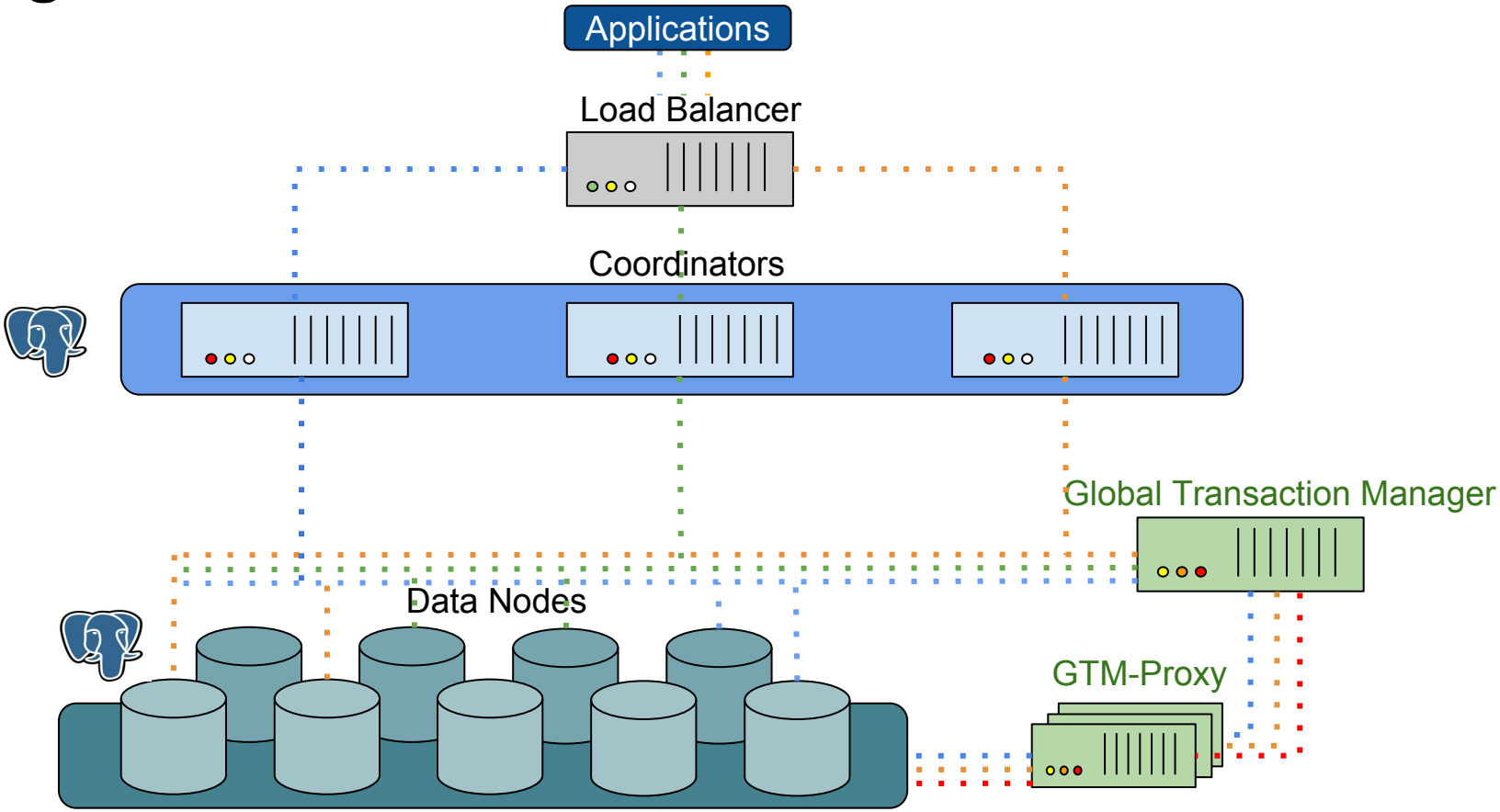
repmgr

Postgres-XL 9.5.5 R1.6

Postgres-XL 9.5.5 R1.6

- Connection pooling
- Load balancing
- High availability
- Scaling
- MVCC

Postgres-XL architecture



Postgres-XL performance tests

PostgreSQL 9.5.10

- CentOS 7.2
- 32 GB RAM
- 16 Core CPU

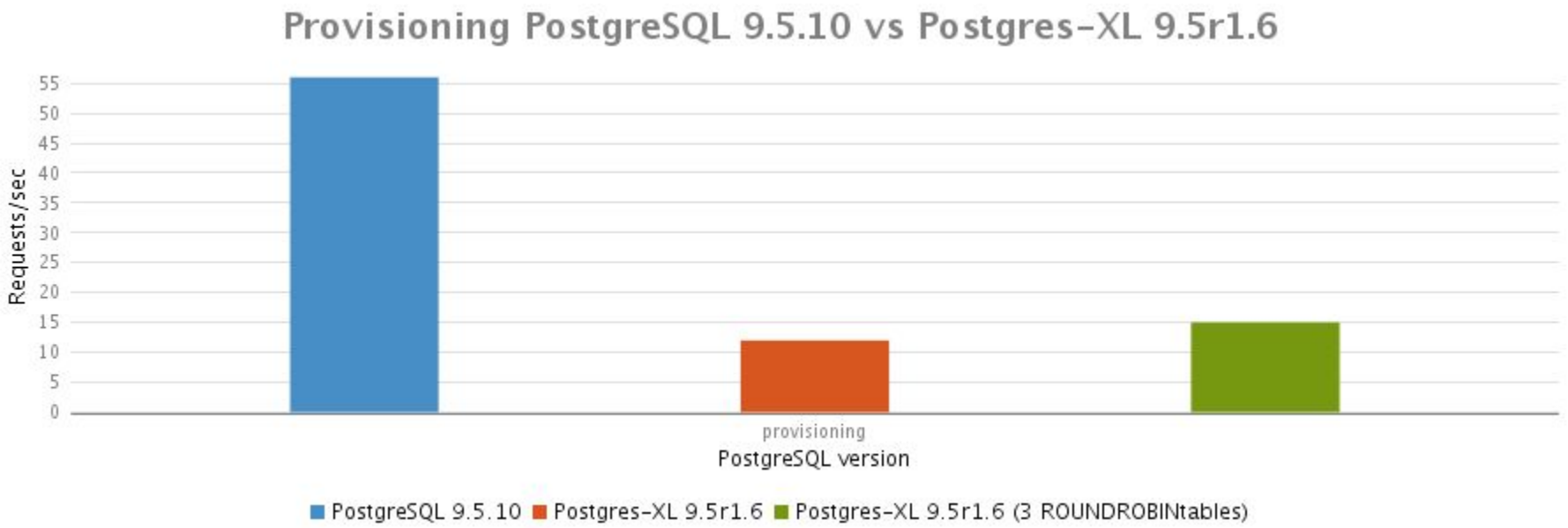
Postgres-XL 9.5r1.6 (Short Version)

- CentOS 7.2
- 2 Nodes:
  - 32 GB RAM
  - 10 CPU Core
- 1 Node GTM:
  - 8 GB RAM
  - 4 CPU Core
- Distribution:
  - All tables are distributed by replication
  - 3 big tables are distributed by ROUND ROBIN

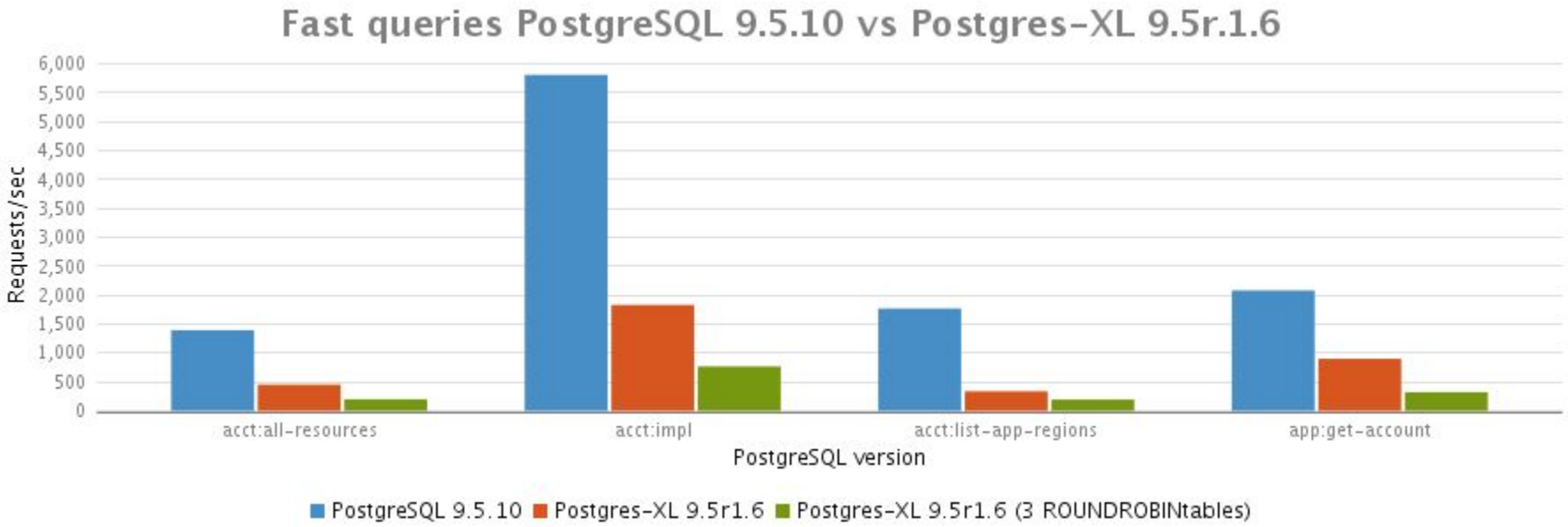
Limitations

- Hot Standby is not supported
- Distributed deadlock detection is not supported yet.
- Deadlock detection exists on each node, just not across nodes.
- Materialised views are currently maintained only on the coordinator.
- EXCLUSION CONSTRAINTS are enforced when both rows map to the same datanode.
- User defined functions have several limitations.
- There are restrictions on complex UPDATE/DELETE queries and updating distribution column values.
- TRIGGERs are not supported.
- EVENT TRIGGERs are not supported.
- SERIALIZABLE TRANSACTIONs are not supported.
- CREATE INDEX CONCURRENTLY is not supported.
- SAVEPOINTs are not supported.
- Large objects are not supported.
- Recursive queries work only in certain conditions.
- GROUPING SETS, ROLLUP or CUBE are not yet supported.
- Foreign Data Wrappers are not supported.
- INSENSITIVE/SCROLL/WITH HOLD cursors are not supported.
- LISTEN/NOTIFY is not supported.

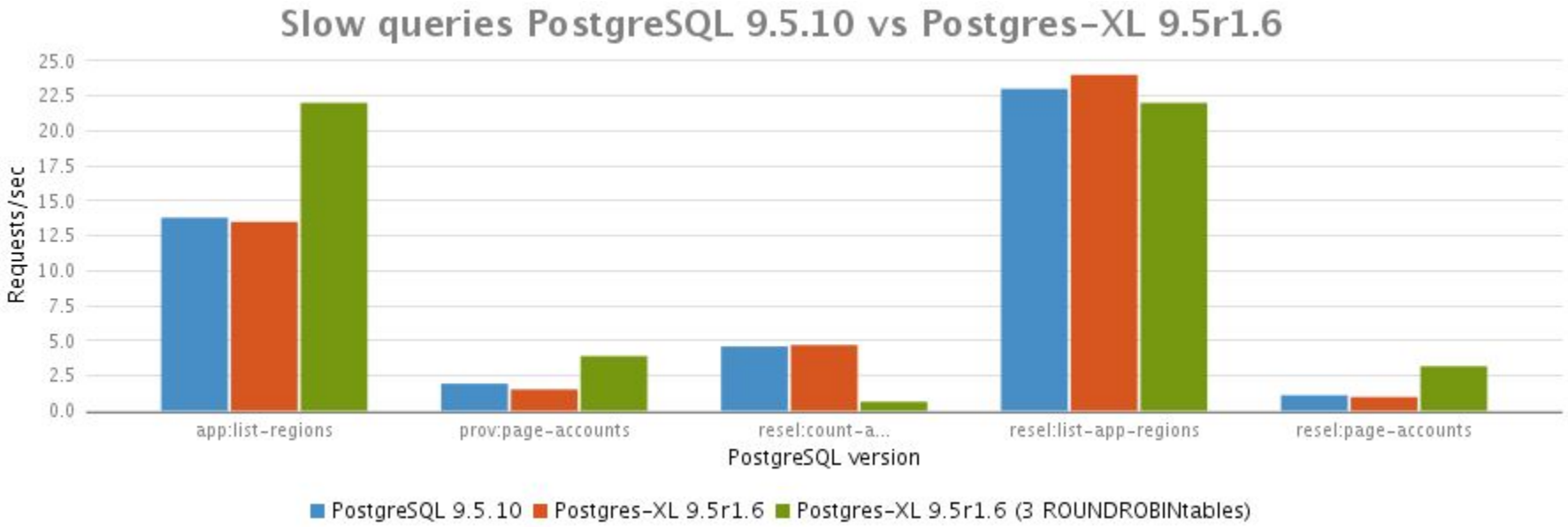
Provisioning test



Fast queries



Slow queries



Postgres-XL summary

- ACID
- Distributed MVCC
- Load balancing (nodes)
- HA (for replication table)

- Connections overhead
- Resharding
- Missing some features(cross FKs, materialised views ,triggers)
- Not suitable for OLTP

Requirements

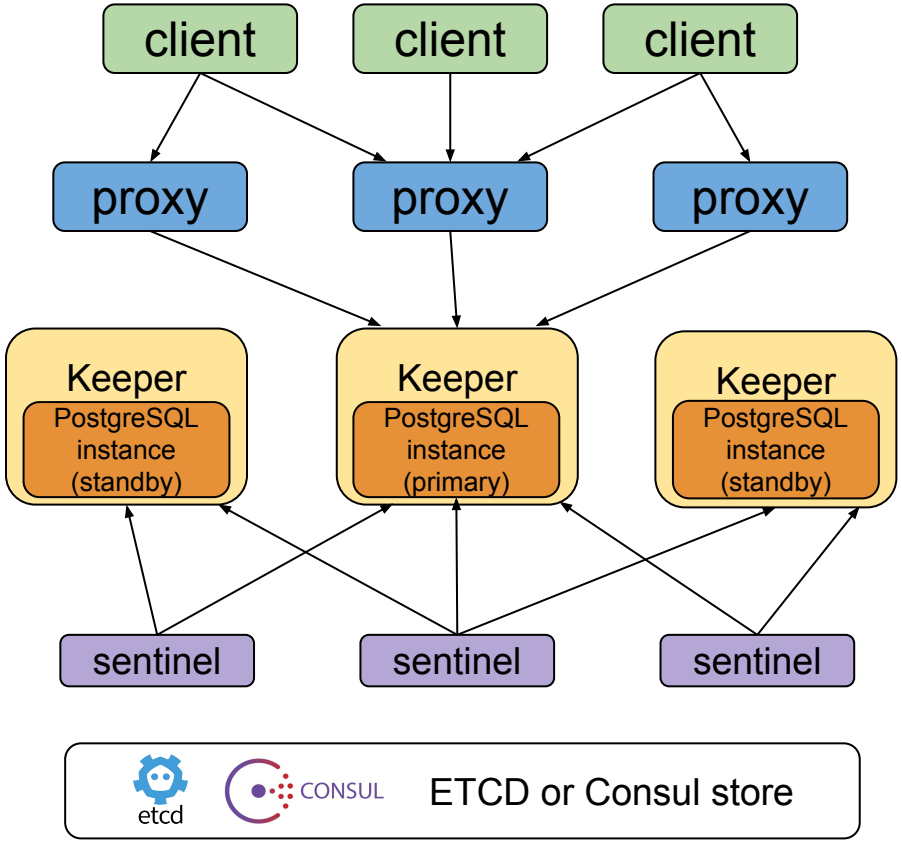
- High availability
- Easy cluster setup in minutes (from scratch or from an existing instance)
- Easy cluster administration and configuration
- Easy cluster scaling
- Can be used in Clouds

Simone Gotti

SORINT.lab OSS

https://github.com/sorintlab/stolon/

Stolon Architecture



Stolon Features

- Leverages PostgreSQL streaming replication.
- Resilient to any kind of partitioning. While trying to keep the maximum availability, it prefers consistency over availability.
- kubernetes integration letting you achieve postgreSQL high availability.
- Uses a cluster store like etcd or consul as an high available data store and for leader election
- Asynchronous (default) and synchronous replication.

Stolon Features

- Full cluster setup in minutes.
- Easy cluster administration
- Can do point in time recovery integrating with your preferred backup/restore tool.
- Standby cluster (for multi site replication and near zero downtime migration).
- Automatic service discovery and dynamic reconfiguration (handles postgres and stolon processes changing their addresses).
- Can use pg_rewind for fast instance resyncronization with current master.

Commands invocation

- stolon-keeper
- stolon-sentinel
- stolon-proxy
- stolonctl

Stolon Client (stolonctl)

Available Commands:

clusterdata	Retrieve the current cluster data
help	Help about any command
init	Initialize a new cluster
promote	Promotes a standby cluster to a primary cluster
removekeeper	Removes keeper from cluster data
spec	Retrieve the current cluster specification
status	Display the current cluster status
update	Update a cluster specification
version	Display the version

Cluster Specification

Cluster Parameters

- **For Stolon components**

  (sleepInterval, requestTimeout, failInterval, deadKeeperRemovalInterval ...)
- **To manage replication**

  (maxStandbys, minSynchronousStandbys, maxSynchronousStandbys, usePgrewind, pitrConfig, standbySettings ...)
- **PITR (RecoveryTargetSettings)**

  (recoveryTarget, recoveryTargetLsn, recoveryTargetName ...)

Postgres Parameters

- **pgParameters**

  (shared_buffers, work_mem, temp_buffers, effective_cache_size, autovacuum_vacuum_cost_limit...)
- **NewConfig**

  (locale, encoding, dataChecksums)
- **StandbySettings**

  (primaryConnInfo, primarySlotName, recoveryMinApplyDelay)

Examples

Cluster Specification patching

```
stolonctl --cluster-name=mycluster update --patch '{ "synchronousReplication" : true }'
```

```
stolonctl --cluster-name=mycluster update --patch -f spec.json
```

Cluster Specification replace

```
stolonctl --cluster-name=mycluster update '{ "requestTimeout": "10s", "sleepInterval": "10s" }'
```

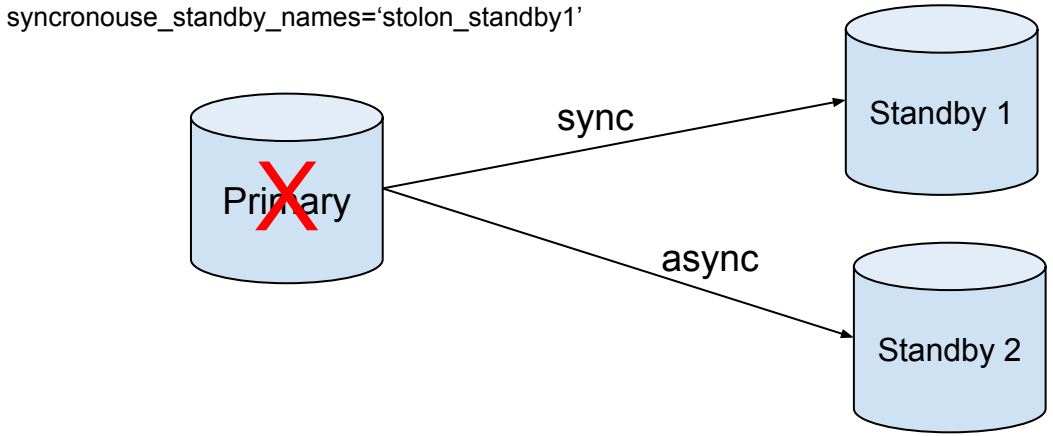
Set some postgres parameters

```
stolonctl --cluster-name=mycluster update --patch '{ "pgParameters" : {"log_min_duration_statement" : "1s" } }'
```

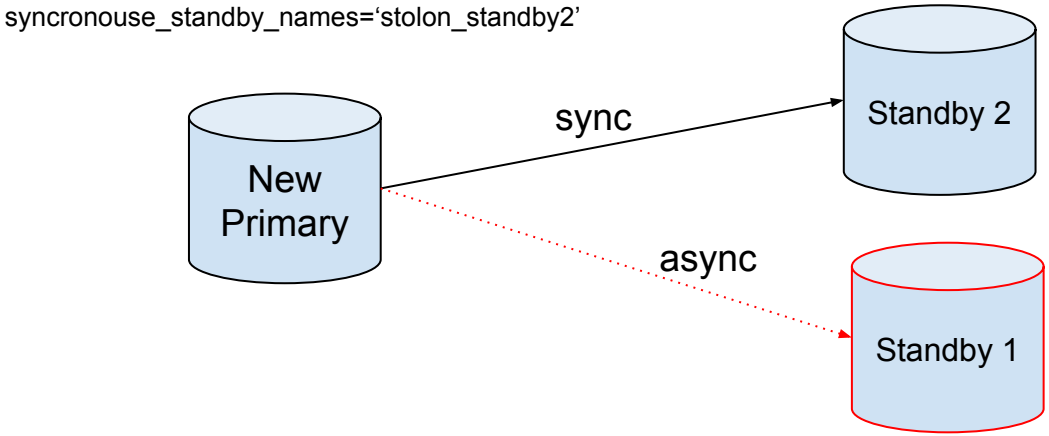
stolonctl status

```
stolonctl --cluster-name=kube-stolon --store-backend=etcd --store-endpoints="http://etcd-client.kube-db:2379" status
=== Active sentinels ===

ID              LEADER
431f50cd        false
554dd42f        true

=== Active proxies ===

ID
17616e94
bafec0d5

=== Keepers ===

UID             HEALTHY PG LISTENADDRESS        PG HEALTHY      PG WANTEDGENERATION     PG CURRENTGENERATION
postgres0       true    10.244.6.14:5432        true            3                       3
postgres1       true    10.244.8.7:5432         true            2                       2

=== Cluster Info ===

Master: postgres0

===== Keepers/DB tree =====

postgres0 (master)
└─postgres1
```

3 PostgreSQL nodes

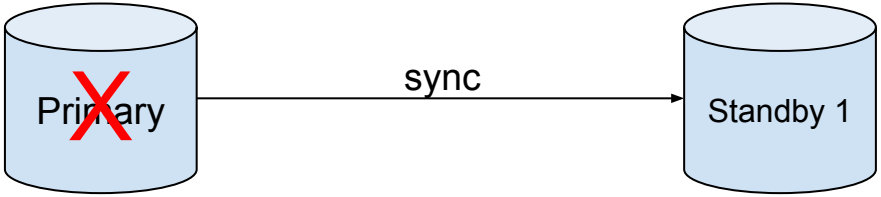


3 PostgreSQL nodes

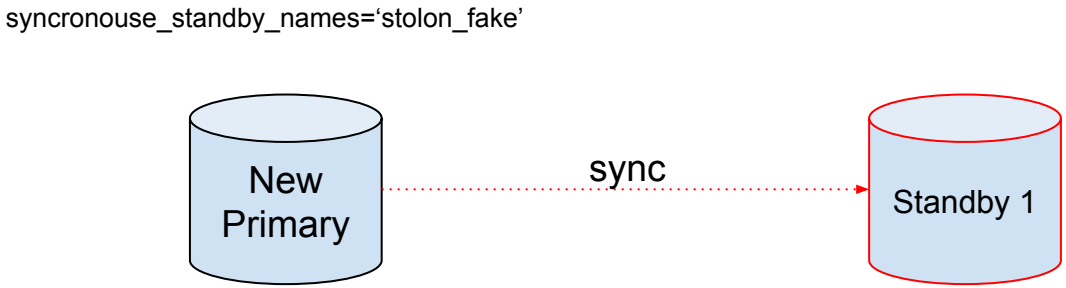


2 PostgreSQL nodes

syncronouse_standby_names='stolon_standby1'



2 PostgreSQL nodes



The primary server is not available for writing new data

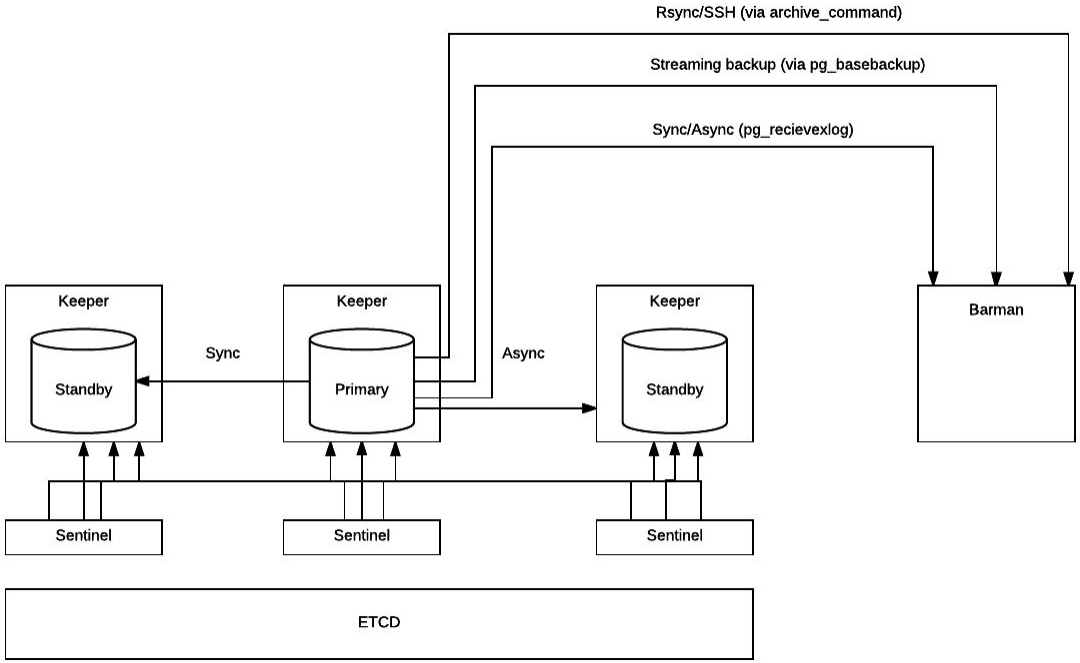
Problems

1. 2 nodes only in asynchronous mode (lack of consistency)
2. Backup is not possible with RPO=0

Put in two cents' worth

+

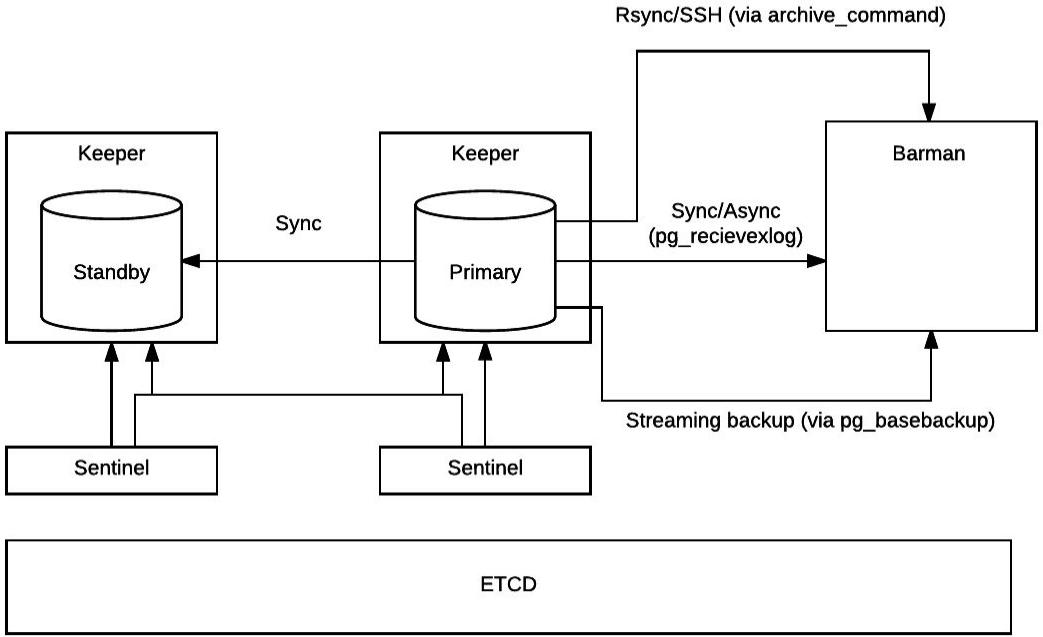
Classic Stolon Architecture and Barman Backup



Solution

- Implemented "AdditionalSyncStandbyNames" parameter;
  - configure additional value for the "synchronous_standby_names" parameter
- Implemented "AdditionalReplicationSlotName";
  - provide additional replication slot with specified configurable name

Stolon with 2 Nodes and Barman as 3rd Nodes



Stolon Release 0.8.0

https://github.com/sorintlab/stolon/pull/410

To be continued

Realize the Promise of Technology™