Yandex

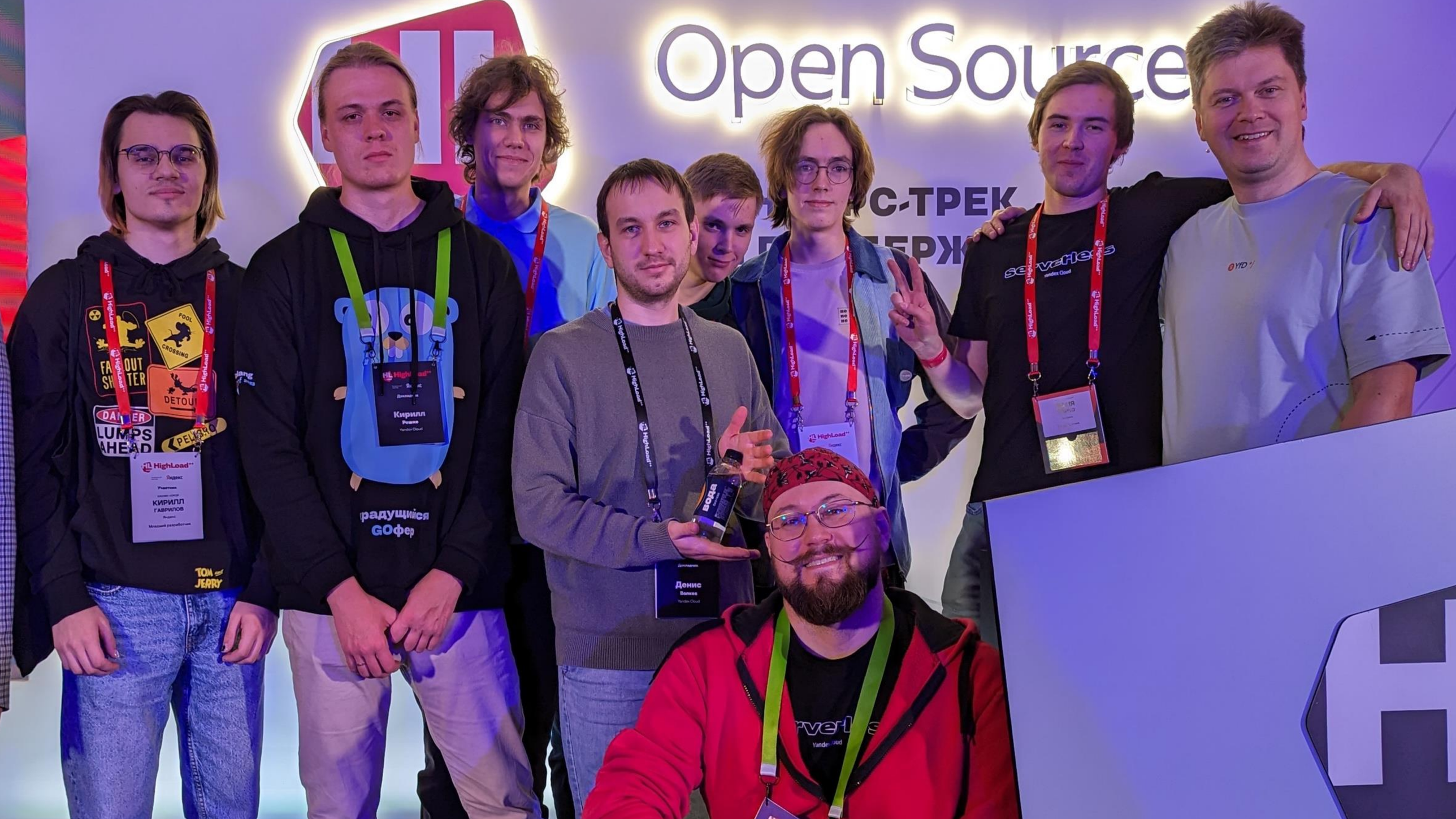
PGCONF.EU 2024

Hidden gems of WAL-G backup tool

Andrey Borodin, Postgres contributor

About me

- Postgres contributor on behalf of Yandex Cloud
- Maintain WAL-G, SPQR, Odyssey and some other stuff


Open Source
HighLoad++
Кирилл
Гаврилов
Кирилл
Решко
Денис
Вода
ТОМ and JERRY

Usual features

Point-in-time recovery

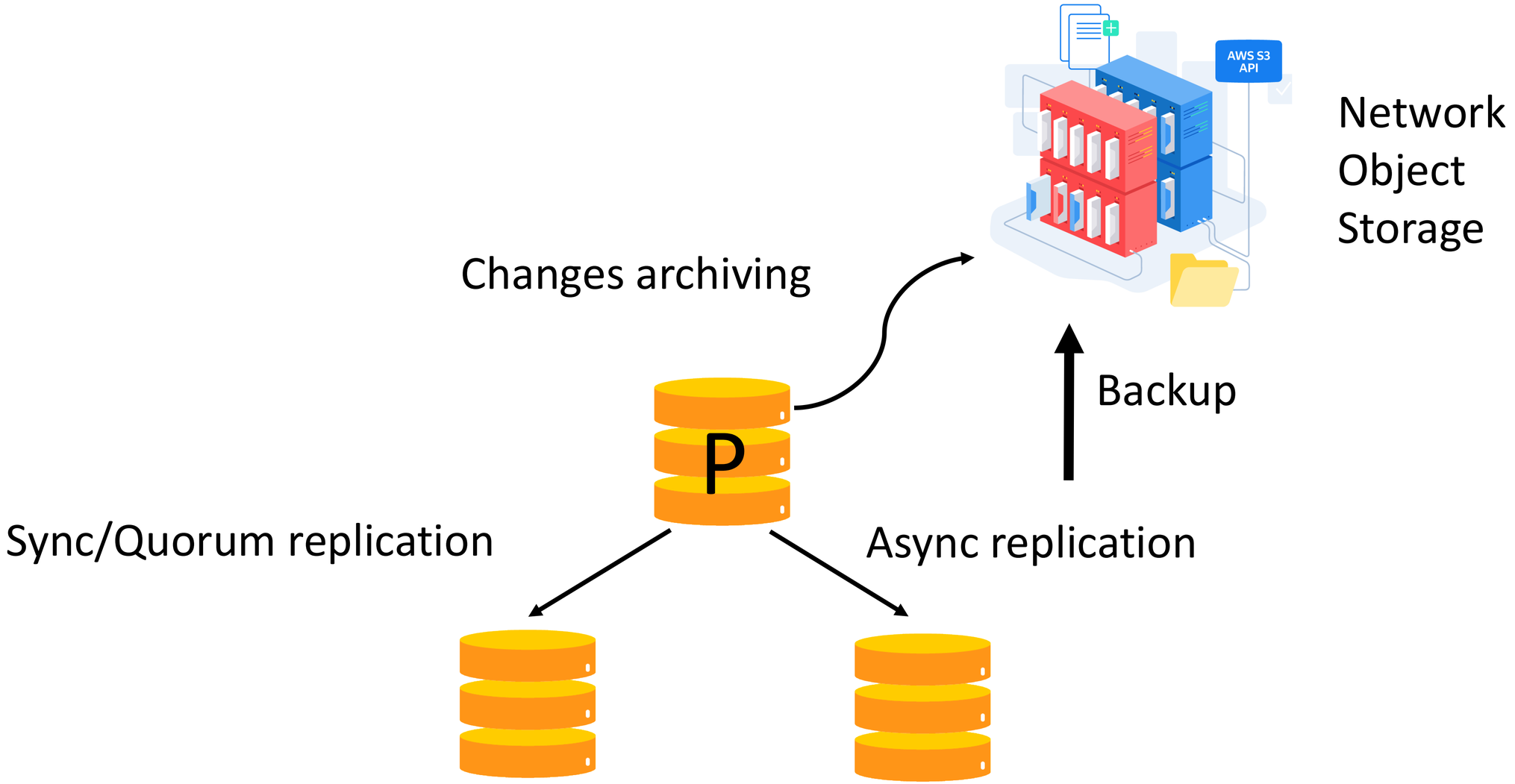
Backup + changes

- Scalable
- Reliable
- Efficient
- Fast

Scalability

- Data: from 10 GB to 10 TB on a host
- RAM: from 2 GB
- Number of CPUs: from 0.05 to ~100
- Async and parallel whenever possible
- Don’t spill anything on a local disk

HA cluster in the cloud



Resources

- Storage space

Resources

- ~~Storage space~~
- CPU
- Net bandwidth
- Disk IOPS

Reliability

- Protection from human error via automation and safety checks
- Prevention of data corruption
- Consistency monitoring
- Integration with other systems (HA tool)
- Extensibility and unification of approaches
- Encrypted data in storage

Fast recovery

- OLAP

From start to consistency point

- OLTP standby

To starting streaming replication

- OLTP primary

Until recovery target and accept of write queries

Unacceptable

- Data locks

Business can’t wait

- Data loss

We call it a “database” after all

pgProBackup

Barman

Backup and recovery manager for PostgreSQL

pgBackRest

WAL-G

Releases 53

v3.0.3 Latest
on Aug 8

+ 52 releases

Contributors 185

+ 171 contributors

Languages

Go 88.9% Shell 9.2%
Other 1.9%

3.2k stars

63 watching

457 forks

wal-g — -bash — 85×33

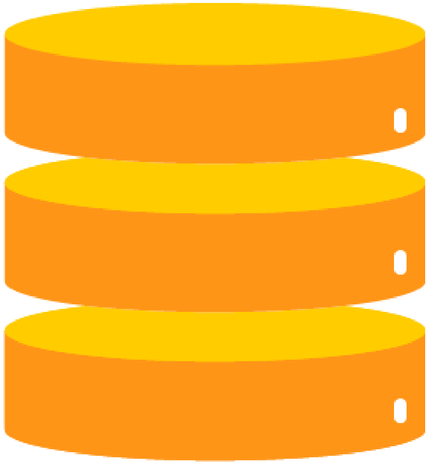
~/GoglandProjects/src/github.com/wal-g/wal-g/cmd/wal-g — -bash | ~/project/bin — psql postgres

```
x4mmm-osx:wal-g x4mmm$
x4mmm-osx:wal-g x4mmm$
x4mmm-osx:wal-g x4mmm$
x4mmm-osx:wal-g x4mmm$ AWS_ENDPOINT=https://storage.yandexcloud.net AWS_ACCESS_KEY_ID
=wIRAxxwOPLI3VrGwtYWL AWS_SECRET_ACCESS_KEY=ne[redacted]vsXX
WALE_S3_PREFIX=s3://wal-g-test/ ./wal-g backup-list
Path:
name                          last_modified        wal_segment_backup_start
base_000000010000000000000004 2019-02-02T18:39:30Z 000000010000000000000004
x4mmm-osx:wal-g x4mmm$
x4mmm-osx:wal-g x4mmm$
```

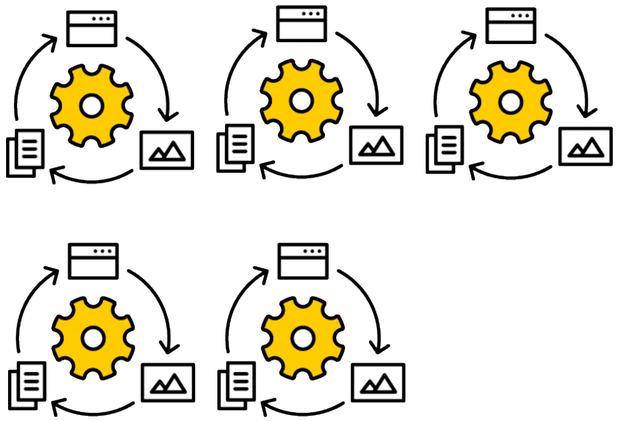
```
x4mmm-osx:wal-g x4mmm$
x4mmm-osx:wal-g x4mmm$ AWS_ENDPOINT=https://storage.yandexcloud.net AWS_ACCESS_KEY_ID
=wIRAxxwOPLI3VrGwtYWL AWS_SECRET_ACCESS_KEY=neh7EEYANpqGS5GJEbmOywhznxcIBukG3IamvsXX
WALE_S3_PREFIX=s3://wal-g-test/ ./wal-g backup-push ~/DemoDb
Path:
INFO: 2019/02/02 21:56:42.509465 Doing full backup.
WARNING: 2019/02/02 21:56:42.526434 It seems your archive_mode is not enabled. This w
ill cause inconsistent backup. Please consider configuring WAL archiving.
INFO: 2019/02/02 21:56:42.740377 Walking ...
INFO: 2019/02/02 21:56:42.742571 Starting part 1 ...
INFO: 2019/02/02 21:56:43.112485 Finished writing part 1.
INFO: 2019/02/02 21:56:48.744337 Starting part 2 ...
INFO: 2019/02/02 21:56:48.761395 /global/pg_control
INFO: 2019/02/02 21:56:48.764006 Finished writing part 2.
INFO: 2019/02/02 21:56:48.878931 Starting part 3 ...
INFO: 2019/02/02 21:56:48.894990 backup_label
INFO: 2019/02/02 21:56:48.895030 tablespace_map
INFO: 2019/02/02 21:56:48.895056 Finished writing part 3.
INFO: 2019/02/02 21:56:49.523658 Uploaded 3 compressed tar Files.
x4mmm-osx:wal-g x4mmm$
```

```
# - Archiving -

archive_mode = on

archive_command = '/usr/bin/envdir /etc/wal-g/envdir
/usr/bin/timeout 600 /usr/bin/wal-g wal-push %p'
```

Base backup



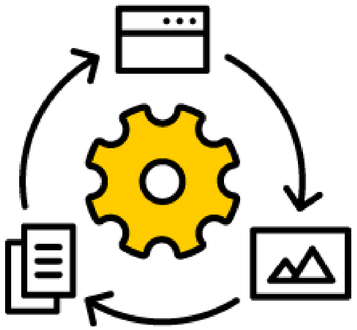
DB copy



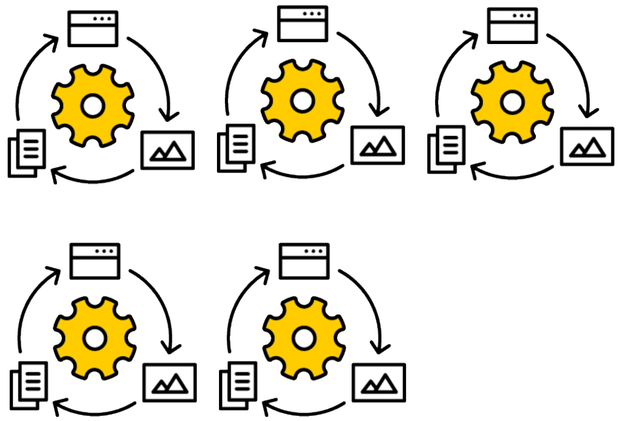
Changes (WAL)

Delta backups

DB copy

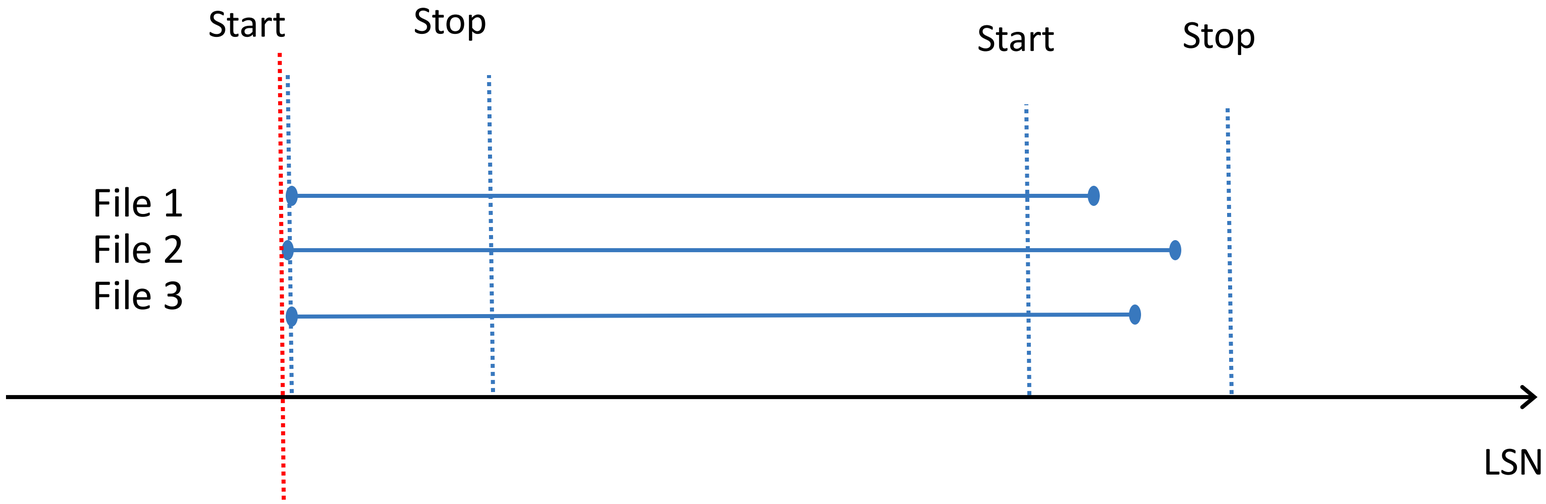


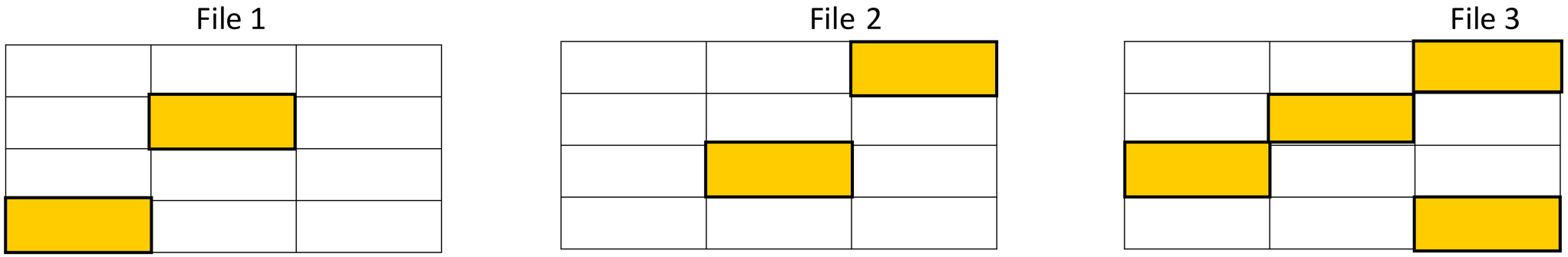
Delta copy



Changes (WAL)

LSN-based deltas





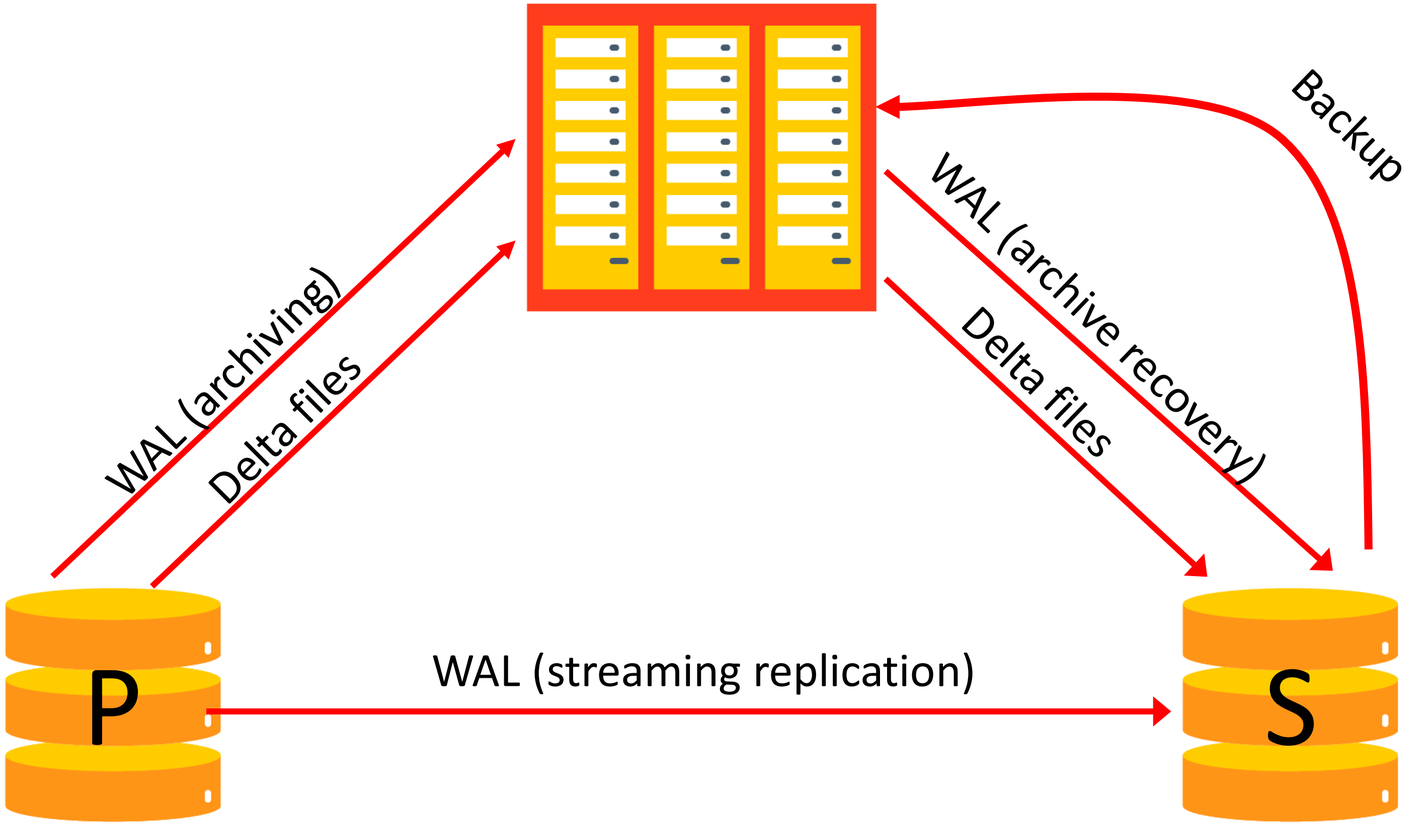
Unusual features

Parallel WAL interface

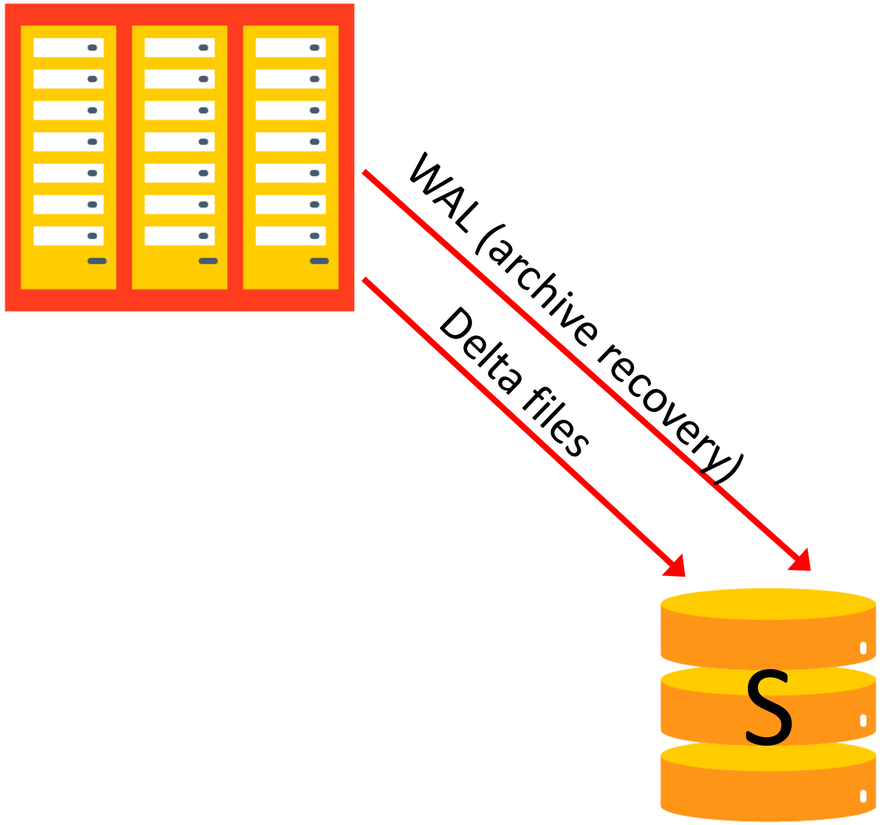
- archive_command
- restore_command

Synchronous PG calls are triggering prefetches

Data flows in the system



WAL-based prefault



Partial restore

Restore only several

- Databases
- Schemas
- Tables

Needs ignore_invalid_pages = true before promotion

Throttling

WALG_NETWORK_RATE_LIMIT
WALG_DISK_RATE_LIMIT

Throttling

WALG_NETWORK_RATE_LIMIT
WALG_DISK_RATE_LIMIT

- WALG_UPLOAD_DISK_CONCURRENCY

Throttling

WALG_NETWORK_RATE_LIMIT
WALG_DISK_RATE_LIMIT

- WALG_UPLOAD_DISK_CONCURRENCY

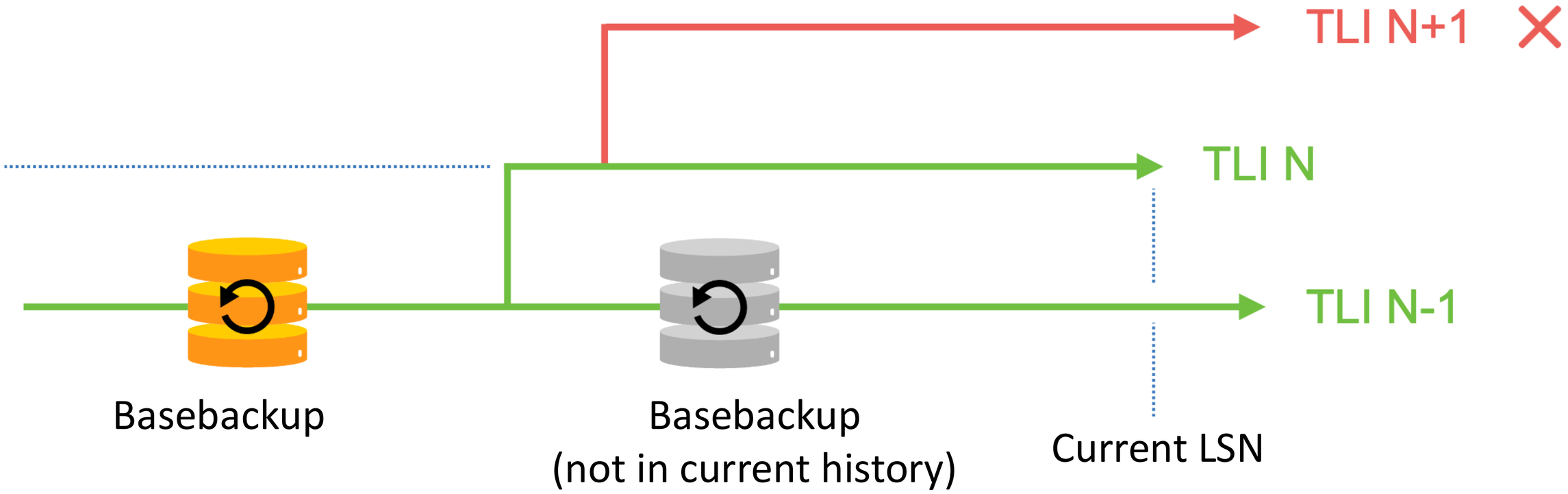
```
wal-g backup-push --turbo --config=/usual/wal-g.yaml
```

Consistency checks

WAL-verify

```
root@some-machine ~ # wal-g wal-verify timeline --config /etc/wal-g/wal-g.yaml
INFO: 2021/05/05 16:09:13.289940 Building check runner: timeline
INFO: 2021/05/05 16:09:13.289979 Running the check: timeline
[wal-verify] timeline check status: OK
[wal-verify] timeline check details:
Highest timeline found in storage: 4
Current cluster timeline: 4
```

Check for unknown timeline



WAL-verify integrity

```
root@some-machine ~ # wal-g wal-verify integrity --config /etc/wal-g/wal-g.yaml
INFO: 2021/05/05 16:05:59.459085 Building check runner: integrity
INFO: 2021/05/05 16:05:59.496998 Detected earliest available backup: base_000000020000000000000000A
INFO: 2021/05/05 16:05:59.497033 Running the check: integrity
[wal-verify] integrity check status: OK
[wal-verify] integrity check details:
+-----+--------------------------+--------------------------+----------------+--------+
| TLI | START                    | END                      | SEGMENTS COUNT | STATUS |
+-----+--------------------------+--------------------------+----------------+--------+
|   2 | 000000020000000000000000A | 000000020000000100000AD |            420 |  FOUND |
|   3 | 000000030000000100000AE | 000000030000000200000027 |            122 |  FOUND |
|   4 | 000000040000000200000028 | 000000040000000600000CF |           1192 |  FOUND |
+-----+--------------------------+--------------------------+----------------+--------+
```

Absent WAL file

000000010000001300000E1

~~000000010000001300000E2~~

000000010000001300000E3

000000010000001300000E4

000000010000001300000E5

WALG_VERIFY_PAGE_CHECKSUMS

***_backup_sentinel.json:**

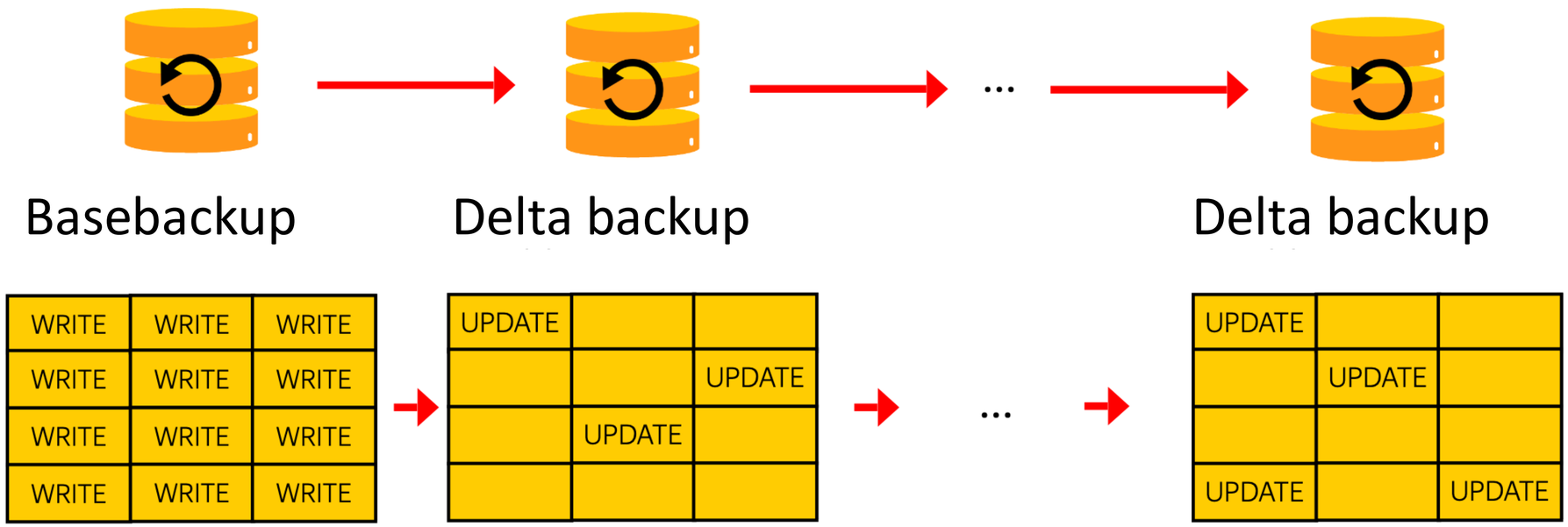
```
...
"/base/16384/16397":{
   "CorruptBlocks":{
      "SomeCorruptBlocks":[3,5,17],
      "CorruptBlocksCount":3
   },
   "IsIncremented":false,
   "IsSkipped":false,
   "MTime":"2020-08-27T14:31:06.483880188+05:00"
},
```

Prometheus metrics wanted

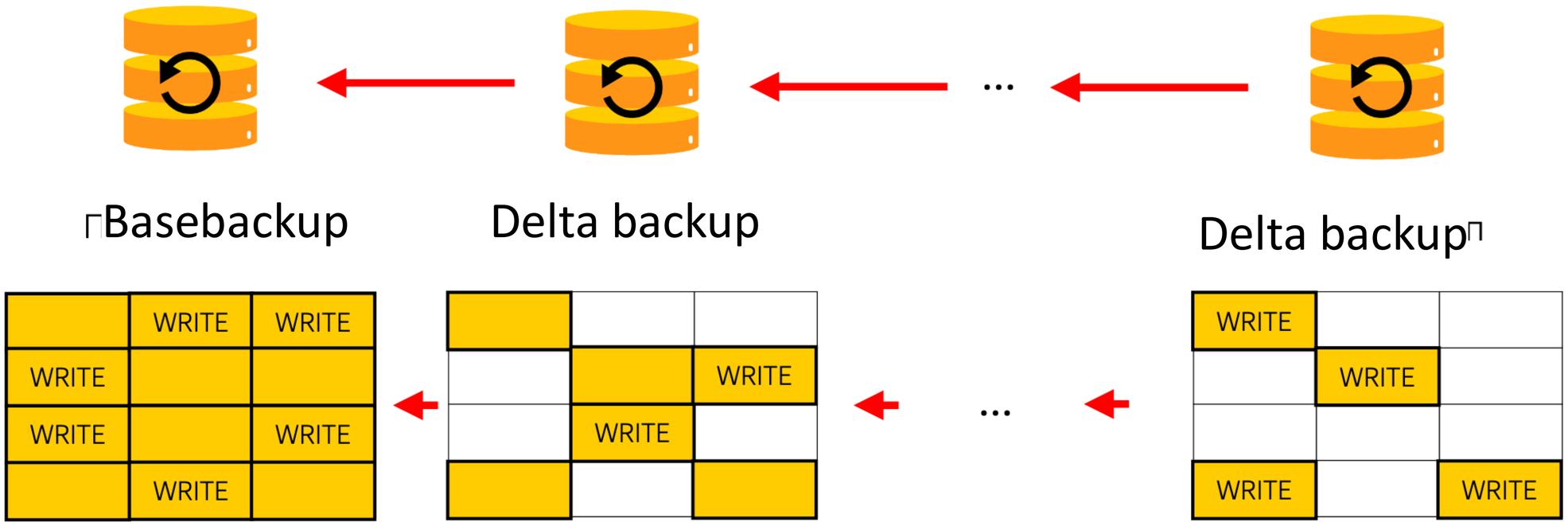
Most monitoring are based on **wal-g backup-list --json**

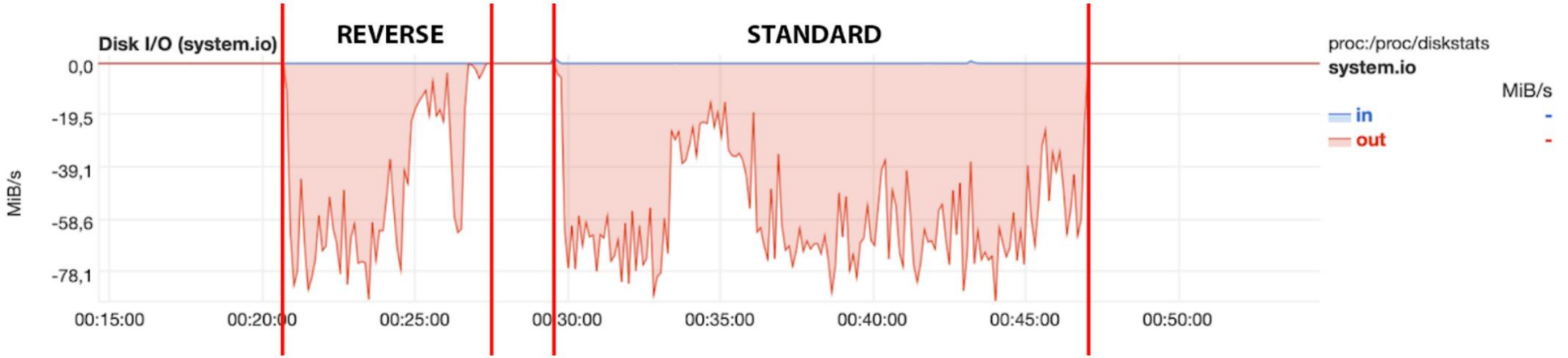
Delta unpack

Direct delta unpack



Reverse delta unpack




Disk I/O (system.io)
REVERSE
STANDARD
0,0
-19,5
-39,1
-58,6
-78,1
MiB/s
00:15:00
00:20:00
00:25:00
00:30:00
00:35:00
00:40:00
00:45:00
00:50:00
proc:/proc/diskstats
system.io
MiB/s
in
out

Catchup

Oracle: Rolling forward a standby database using RMAN

In this Document

WAL-G

```
wal-g catchup-send        # on Primary
wal-g catchup-receive     # on Standby
```

- Beware: wal-g must open tcp port on primary, not 5432\6432

Features

- Encrypted with WAL-G settings
- Safety checks
- Idempotent

Features

- Encrypted with WAL-G settings
- Safety checks
- Idempotent (TODO: move pg_control)

Features

- Encrypted with WAL-G settings
- Safety checks
- Idempotent (TODO: move pg_control)

Parallelism is not implemented yet

Changes tracking

Options

PTRACK

WAL Delta

Heap scan

PTRACK

- Most of distros do no support it
- We will support it, once there’s someone to maintain it in WAL-G

WAL delta

We had issues with VMs

https://github.com/wal-g/wal-g/issues/1296

Heap scan

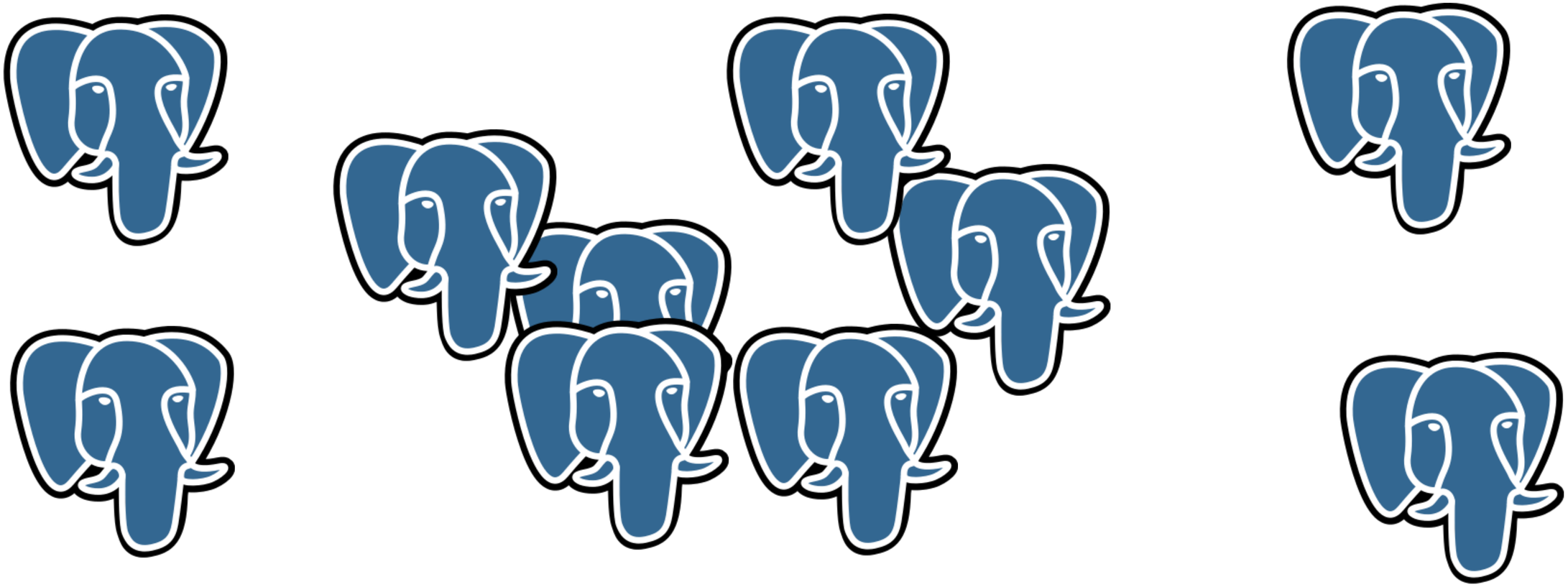
- Most io-consuming
- Reliable
- VMs and FSMs are copied anyway

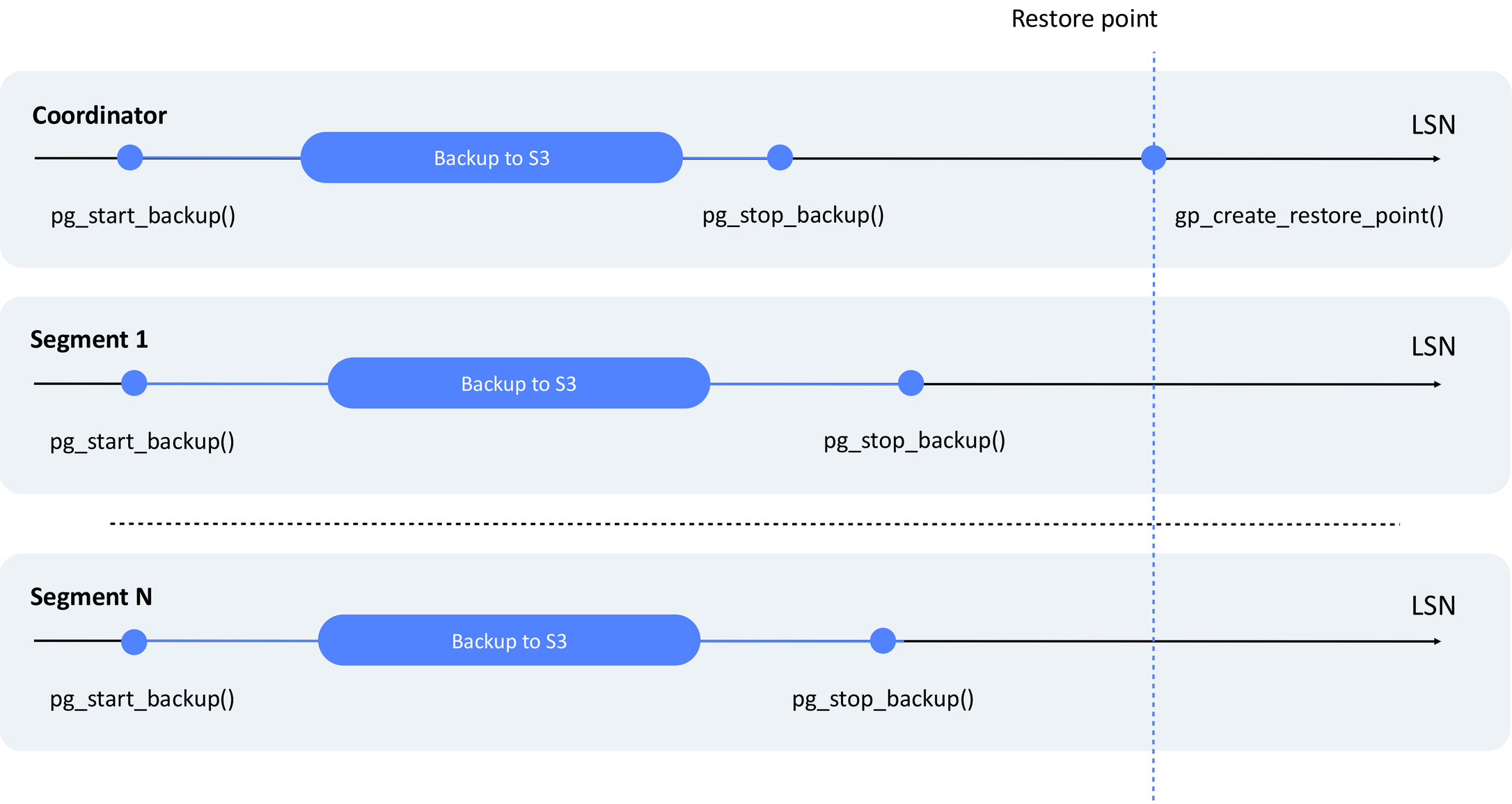
ModTime-based

Sharded clusters

Greenplum – it’s just many Postgreses!

Greenplum – it’s just many Postgreses!




Restore point
Coordinator
LSN
Backup to S3
pg_start_backup()
pg_stop_backup()
gp_create_restore_point()
Segment 1
LSN
Backup to S3
pg_start_backup()
pg_stop_backup()
Segment N
LSN
Backup to S3
pg_start_backup()
pg_stop_backup()

S3 features

- Snapshot backup
- Server side copy
- PATCH

Failover storages

- Useful when your primary archive storage can fail

Features not there yet

There’s too much of the stuff in a backup

- Indexes
- VMs
- FSMs

Can be recreated during restoration

Backup must be a side effect of a VACUUM

Help wanted

- Packaging
  - Apt
  - Yum
- Documentation

Thanks! ☺

Andrey Borodin

Postgres contributor

x4mmm @yandex-team.ru

x4mmm

PGCONF.EU
2024