PostgreSQL Streaming Replication for Medium-sized Databases (10-20 TB)

Challenges and Solutions

PostgreSQL

Ivan Ivanov

TelebidPro

The Case

On the go

- Planning* and running replication from the beginning **is a must** for a new project
- But what if you need to add replication **to already existing cluster** with size 10-20 TB?

Size

- Everything is easy as a tutorial **when the size of the data is small**
- Problems and challenges occur **when the size increases significantly**

Network

- Running database replication in a **local network** is not as complex as running database replication **through VPN networks or over the internet** in different countries and cloud datacenters

* See Botros, S., Tinley, J. High Performance MySQL, 4th Edition, 2021.

Challenges deep-dive

On the go

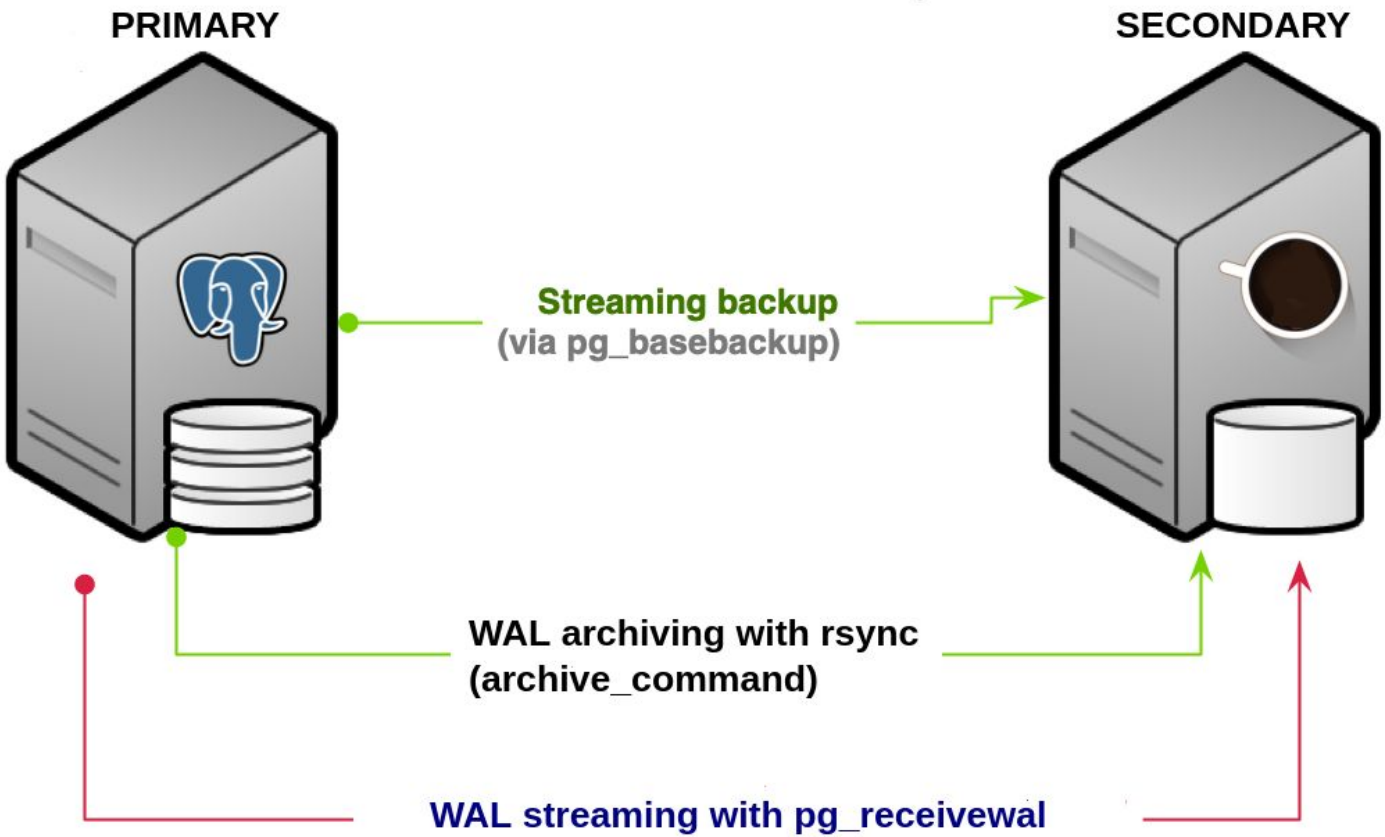
- Running **pg_basebackup** with the default option `--wal-method=stream`
- How to copy all necessary WAL files during the backup?

Size

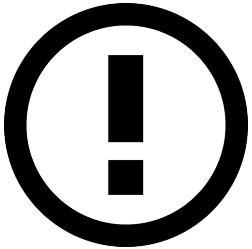
- **10-20 TB** database is a normal size for a medium-sized system
- How size can affect setting up and running PostgreSQL replication?

Network

- Copying 10-20 TB of data **through the internet** can be slow
- How to speed up the process of copying data files and WAL files?


PRIMARY
SECONDARY
Streaming backup
(via pg_basebackup)
WAL archiving with rsync
(archive_command)
WAL streaming with pg_receivewal

The problem: example

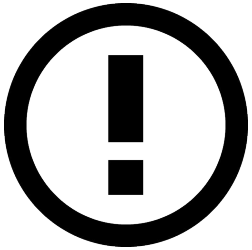


- command: **`pg_basebackup --progress --verbose --wal-method=stream -c fast -D /var/lib/postgresql/11/main/ -U replicant -h 5.5.5.5`**

- the option **`--wal-method`** has several settings, but all of them are not sufficient

- the best option to use is **`--wal-method=none`** (instead of **`fetch`** or **`stream`**)

The problem: explanation



- pg_basebackup: copying data files **without compression** (data files are copied 1:1)

- **the data copy can take several days to complete** for a **10 TB database** and a **100 Mbps internet connection** from a local datacenter to external cloud, e.g. Amazon (**this is going to take around 9 days**)

- pg_basebackup: streaming all necessary WAL files through the copy of data files, but **stops streaming them when the data copy finish**

- **PostgreSQL replication cannot start at the end of the process, because WAL files cannot catch-up due to the large size of the database and the speed of the network or have been already deleted on the primary host**

Solutions

Copy all WAL files using independent archiving (and not from pg_basebackup)

- set up **independent PostgreSQL archiving** the standard way

```
archive_mode = on
archive_command = 'test ! -f /pg/archive/%f && cp -p %p
/pg/archive/%f'
# archive_command = 'ssh postgres@5.5.5.5 "test ! -f
/var/lib/postgresql/archive/%f" && scp -p %p postgres@5.5.5.5:/var/li
b/postgresql/archive/%f'
```

- set up **a custom service to sync WAL files** from local server to replication server

```
while true ; do /usr/bin/rsync -avr --remove-source-files
--ignore-existing /pg/archive/
postgres@5.5.5.5:/var/lib/postgresql/archive/ ; sleep 0.5 ; done
```

Copy all WAL files using independent archiving (and not from pg_basebackup)

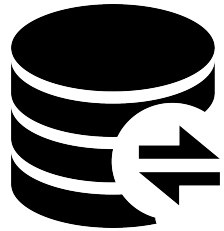
- now use **`--wal-method=none`** to copy only the data files (without any WAL files)

```
pg_basebackup --progress --verbose --wal-method=none -c fast -D
/var/lib/postgresql/11/main/ -U replicant -h 5.5.5.5
```

- now pg_basebackup **will only copy the needed data files**, but **providing all necessary WAL files is given to the independent archiving**, which will continue sending WAL files after the completion of the base backup

- use the **`tar`** format and the **`--gzip`** option

Use pgBackRest for parallel and compressed backups



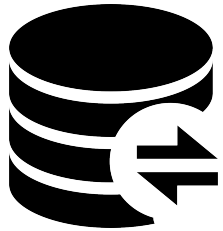
• the best alternative to pg_basebackup is **pgBackRest**, which has the following features that pg_basebackup does not have:

- ⇨ **parallel backup processing**
- ⇨ **compression**

• this means that you can dump the base backup faster because of the parallel processing and take advantage with the compression on the fly

• more information about **pgBackRest** is found here: **https://pgbackrest.org/**

Use pgBackRest for parallel and compressed backups



- **pgBackRest** is similar to **Barman**
- available as a **Debian** package and **RHEL** or **CentOS** package
- uses configuration files known as **stanza**
- **example** configuration:

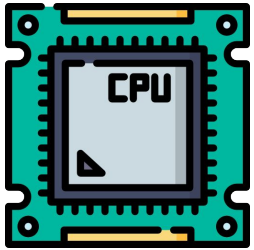
```
[demo]
pg1-host=pg-primary
pg1-path=/var/lib/postgresql/12/demo
[global]
compress-level=3
process-max=3
```

Network connection tips

• if your case includes setting up database replication between a **primary local PostgreSQL server**, e.g. in **Bulgaria**, and a **secondary PostgreSQL replica server** in the cloud, .e.g. in **Germany** (Amazon Web Services) I recommend copying data files and WAL files directly **through the internet with TLS enabled connection**

• **using VPN networks can slow down the connection even further** and for setting up the replication you can use TLS internet connection and then for running the replication you can switch to VPN access

Hardware resources tips



• if you are setting up PostgreSQL secondary replica server in the cloud, you have the luxury to choose not only the **CPU** and **RAM** resources, but also to **select different disks with different speeds and IOPS parameters**

• for setting up the replication (copying the data files and applying WAL files for the first time) choose a virtual machine with more vCPUs and RAM (**and increase `shared_buffers` and `wal_buffers`**)*

• for setting up the replication choose **faster disks** in order to be sure that this will not be a bottleneck when applying WAL files starts (then you can revert back to slower disks again)

* See https://www.enterprisedb.com/blog/tuning-sharedbuffers-and-walbuffers

The bottleneck: WAL files catching-up

• after completion of copying the data files with a pg_basebackup from the primary to secondary server, you can find yourself in a **situation** where your **10 TB** database was copied through the internet for **9 days** and now **PostgreSQL must replay all 500 000 WAL files** which have been copied for these 9 days

• **500 000 WAL files are around 8 TB**, which is almost the same as the size of your PostgreSQL cluster

• **WAL files are applied only by one recovery PostgreSQL process**: it must copy the WAL file from the archive (**write operation**), read it (**read operation**) and apply it to the database (**write operation**)

Optimize the applying of WAL files

Copy from archive

- You can put all WAL files from the master **directly in the pg_wal directory** of the secondary server, but this is **highly not recommended**, because can easily interrupt the replication process

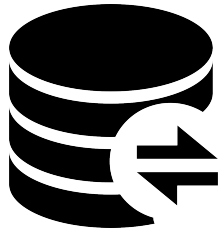
Read WAL files

- Use **pg_prefaulter** (https://github.com/TritonDataCenter/pg_prefaulter)
- It can **read-ahead** WAL files in order to **prewarm them in the filesystem cache** for faster reading
- Some people use **tar** to achieve the same result

Apply WAL files

- **Can not be skipped**
- **Optimize the application** to decrease filling of WAL files on the master
- Use all other tips in order to have **a fewer count of WAL files to apply**

Running the replication



• monitor system load, CPU and RAM usage and decrease or set **hardware resources** and **configuration settings** properly for the replica server

• monitor PostgreSQL replication **status** and **lag** through views `pg_stat_replication` and `pg_stat_wal_receiver`

• either use `archive_command`, or `pg_receivewal` with **replication slots** to preserve the replication when the network drops or there is a significant lag

Questions?

Thank you!

PostgreSQL

Ivan Ivanov

TelebidPro