DATAVISION
by SODA

What are Containerized Postgres Storage Requirements

Alvaro Hernandez
@ahachete

`whoami`

- Founder & CEO, OnGres
- 20+ years Postgres user and DBA
- Mostly doing R&D to create new, innovative software on Postgres
- More than 135 tech talks
- Founder and President of the NPO Fundación PostgreSQL
- AWS Data Hero

Alvaro Hernandez
<aht@ongres.com>

aht.es

Goals for this talk

- Describe how Postgres works at the storage layer
- List the needs for an “ideal” storage system
- Explain how actually Postgres K8s operators use storage
- What may the future look like for Postgres Storage

POSTGRES STORAGE CHARACTERISTICS

DATAVISION by soda

Postgres Storage Architecture. Replication

- Like most RDBMSs Postgres is a shared-nothing architecture.
- Each instance/pod has a full copy of the data.
- Data is replicated via streaming replication (physical replication over TCP connection).
- Data is guaranteed to be the same (eventually or synchronously) but disks are not a exact copy.
- Replicas serve read-only queries but require a read-write filesystem (e.g. processing WAL, temporary queries, etc).

Block Size

- Postgres uses by default a 8192 bytes page size.
- It can be grown or shrink by recompiling Postgres.
- There’s little experience with different sizes. Almost everybody goes with the default.
- Depending on the storage block size, it may be convenient to experiment with different page sizes.

Filesystem

- Postgres relies on a POSIX filesystem.
- Most used ones are:
- ext4
- XFS
- There are some uses of ZFS (mostly on BSD). Performance is lower than ext4/XFS but may be worth. Requires explicit tuning.
- Requires careful operation if run on a distributed filesystem (e.g. NFS) as it may lead to data corruption.

DATAVISION by soda

Postgres Backups

- Backups are taken in physical form. They consist of a copy of the filesystem (PGDATA) along with a set of WAL files (generated over time).
- Filesystem copy does not require to be atomic iif pg_start_backup(‘label’) and pg_stop_backup() are called before and after.
- Snapshots can replace filesystem copy, and backup start/stop function calls are not needed, but are recommended.
- Differential backups are possible (e.g. PgBackRest).

WHAT A POSTGRES STORAGE SHOULD HAVE

Performance

- Many Postgres workloads are heavy I/O bound.
- Usually there’s a great difference between running local NVMEs vs network disk.
- Latency sentitivity for WAL files.
- Sequential throughput for table scans.
- Random opeartions for index traversal.

Multiple disks

- Each Postgres node can have one or multiple disks.
- Multiple are used when:
- Separating WAL files.
- Separating logs.
- Tablespaces.
- But separating into disks may introduce consistency issues.

DATAVISION by soda

Snapshots

- Snapshots are highly desirable characteristic, even though not very widely used in Postgres world.
- Can be used for many use cases:
- Backups (base backup replacement)
- Initializing replicas (requires scripting)
- Initializing logical replicas (requires even more scripting)
- Postgres upgrades (huge scripting required, but doable)
- Database branching! (see DBLab by postgres.ai)

ZFS

- Despite some performance hit, it is highly desirable.
- Snapshot support is great, including diffs between snapshots.
- Compression may provide fair storage and I/O savings.
- Built-in encryption solves one usual requirement for database encryption-on-rest (sometimes transparent data encryption, TDE, is required).
- ZFS is challenging on container environments due to the need to insert the module in the kernel.

HOW POSTGRES OPERATORS USE THE STORAGE

DATAVISION by soda

Storage for Postgres Kubernetes Operators

- For most, it’s quite straightforward: use StorageClass.
- One or more volumes (main PGDATA; WAL or tablespaces) provisioned from the StorageClass per pod.
- Occasionally, existing PVs may be used directly.
- CEPH is sometimes used. Works, but there’s not much production experience with it.

POSSIBLE FUTURE STORAGE USES FOR POSTGRES

DATAVISION by soda

What the Postgres storage future looks like?

- Leverage local storage (e.g. NVME)
  But make it as reliable as network storage.
- Make snapshots a first-class citizen in Kubernetes. Leverage them for Postgres DBOps.
- Move to shared storage architectures? Serverless Postgres (Aurora, Neon).
- Expose Object Storage (e.g. S3) to Postgres:
- Via specialized FS (exposed as POSIX e.g. ZFS proprietary mods)
- As a FDW

Q&A

More info/connect:

aht.es

@ahachete