Postgres in Dockers on ZFS

Background and brief history

- Kobus Wolvaardt and I work at GoodX Software
- GoodX is a medical practice management ERP, CRM and EMR
- 1985 DOS pascal app, 2001 Delphi app on Windows
- 2006 Started moving from file data to PostgreSQL
- Local app and postgres deployments on 500 to 1000 sites (much pain)
- Cloud meant a windows machine in a DC over RDP running Delphi app
- 2013-2017 Business logic moved into database
- 2018 Python based web application launched (cloud solution)
- Growth of centrally hosted Postgresql dbs accelerated

How it grew from there

- Move postgres to Linux (postgres 8.4 performed badly on Windows)
- Put a couple of DBs in one cluster
- Run into performance issues -> buy more CPU and bigger/faster SSDs
- Put more DBs in one cluster
- Run into performance issues -> this time build some indexes :-) and buy more hardware
- Put some more DBs in the cluster
- Rinse repeat

Other design decisions

- Business logic inside the DB
- Printing was moved into DB functions with wkhtmltopdf
- HTML is bad at printing thus latex was installed
- HTTP requests made from plpython3u with requests library (and other things)

Problems

- Back when we ran into DB size issues >1TB SSDs cost $2000+
- Database count went up to 200 per cluster
  - Setting shared memory to 128GB (of 512GB)
  - Connection count set to 8000 and even 10000
  - One DB doing a long running query could push the whole machine over the edge
  - Soooo much money for SSDs
- Clusters started going over a TB each
  - Basebackups take forever
  - Backups take forever

More problems

- Python http requests would lock, killing them would result in cluster restart
- Wkhtmltopdf and latex packages and versions broke stuff
- Developer package choices dictated DB server OS and environment
- Security updates would restart postgres service (with all clusters) and break printing

ZFS - the what, why and the how

- Copy on write FS
- Each change to a block (even a single byte) results in the whole block being rewritten
- Consistent block state guaranteed
- Encryption, Compression and RAID built into FS (not block layer)
- ZFS has a ZIL where it can burst to fast drives (change log, kind of like WAL files)
- ZFS snapshots can be pushed and applied (almost like Postgres replication)
- Snapshots are cheap to make and mount
- Relatively easy raid (zraid1 and zraid2) that performs well
- ZFS has ARC cache to which fast SSD drives can be assigned
- Enterprise ready and widely deployed
- What could possibly go wrong?

Our first ZFS machine

- We needed replicas so we built 144TB 12x12TB spin disk (raidz2) machines for each DC
- Each live machine was replicated to a cluster on this machine
- Testing this machine showed decent performance
- The two replica servers pushed ZFS snapshots hourly, between them.
    - Networks in SA were very expensive until recently and replicating in real time was not feasible

ZFS - the good

- Hourly and daily snapshots going back months
- Seconds to restore (clone)
- No more copying and backing up and downtime for backups to complete
- We could build large clusters with cheap spin disks
- Zraid2 performs well and is really simple to manage
- Decent performance
- Backups and restore is very easy
- Compression built in
- Did I mention backups is a non issue?

ZFS - the bad (and the ugly)

- SSDs and ZFS don't play well
  - SSDs are liars (they lie about true block size)
  - The “copy on write” results in massive write amplification
  - We were expecting our 12 SSD machine to be faster than our 12 hdd machine
  - We put our most difficult and demanding client on it. Expecting a great day, the cluster just died!
  - Spin disks out performed SSDs with default setup (after some tweaking SSD were about equal to spin disks)
- ZFS has thrown a hissy fit at least three times, saying more than half of disks are broken and after a reset it seems only one was broken
- Resilvering a failed disk is stupidly slow. (Rebuilding a broken disk)
  - 20 to 45 days
  - Add SSDs for ZIL and caching… then it is just slow: 2 to 10 days

ZFS settings

- Setup the block size (recordsize) to match postgres (8kb or 16kb)
- ZFS (and most copy on write filesystems) guarantee atomic writes
  - Can set full_page_writes = off
  - Can turn wal_init_zero off as the zeros will not actually be overwritten in a CoW fs
- Disable atime
- Enable compression (zstd)
- Consider tweaking the primarycache (we use ‘all’)
- Set logbias=’latency’ (‘throughput’ results in fragmentation)

Docker - the why and how

- Devops drove our use of dockers
  - Instead of create DB -> restore DB -> apply migrations -> Run automated tests / dev cycle
  - Inherit postgres layer -> install needed packages -> build layer with base DB and data
    - Spin up docker (with no time to apply base DB and data) -> apply migrations -> Run tests
    - Great for automated tests
    - Near instant spin up
    - Concurrent DBs can be spun up, each in a docker subnet (very easy config)
- While discussing our sysadmin troubles docker was suggested
- We also had to update from 9.5 to 12 (some time ago)
  - We decided to break large clusters into smaller +-20 DB clusters (max 2000 connections)
- What could possibly go wrong?

Docker - benefits

- Environment consistent (wkhtmltopdf and latex is always correct)
- Installing custom and our own pg extensions in the docker makes life easy
- Upgrades can now be done one cluster at a time
- Postgresql.conf and hba.conf and everything lives in the data folder
  - Base backups copies config with
  - One directory contains it all, config and data
  - Snapshots also snapshots config (nice for investigations)

Docker - issues

- Logging is a bit different and needed to be setup correctly
- Shared memory default to low in dockers, had to set higher than shared buffers

Other Postgres settings

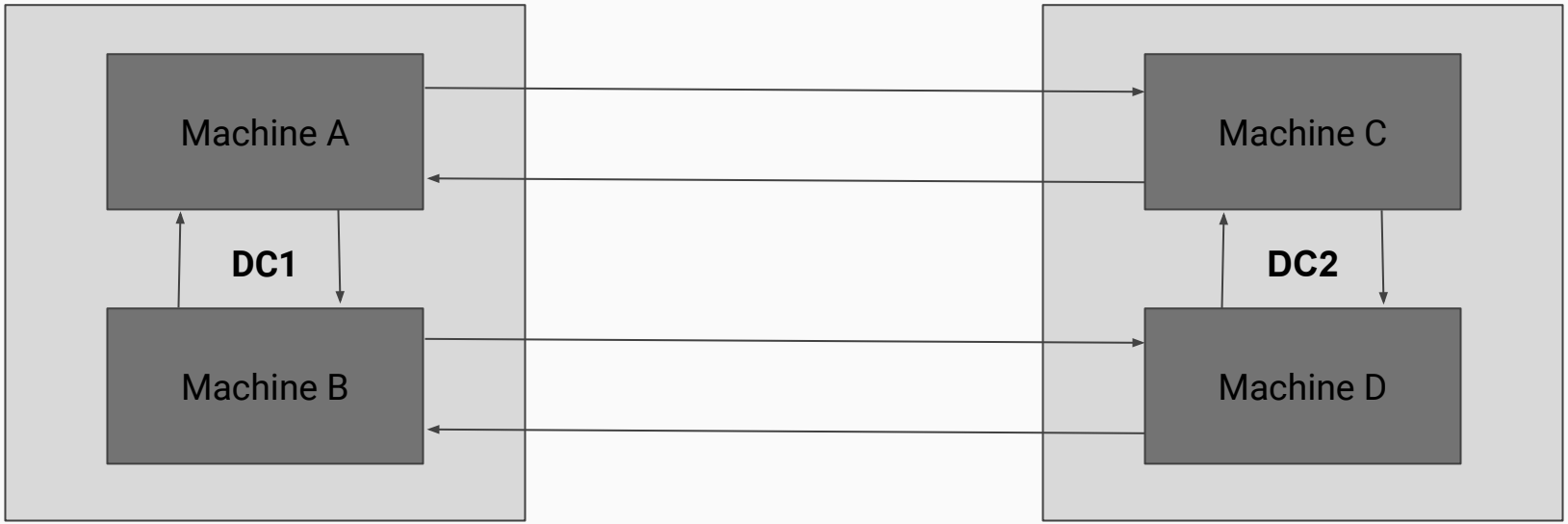
- Stats collector directory should be RAM disk (slow spin disks and ZFS)
- sync_commit = ‘off’
  - We’re not a bank and if 500ms or even 2 seconds goes lost with a server crash we will take it.
- Checkpoint = 15 to 30 minutes reduced IO
  - Slow recovery time after a crash
  - We really don't have crashes often
  - Waiting 10 minutes for a cluster to start is not that bad

What next?

- Having broken up the clusters into many smaller clusters
- Replica machines worked great until we needed them
  - Replication needs to be turned off when using it as a live machine
- Network speeds/costs improved
- New architecture was decided on:
  - ZFS and docker everywhere
  - Lower machine utilization with easier failover
- Upgrading to postgres 16 sometime

Next/Current architecture

3 to 4 clusters of max 20 databases per cluster replicating two directions
Each group of 4 forms a redundant cluster



Thanks

- We are idiots, and still learning… hope this is of use
- Welcome to find me afterward and ask anything or please tell us where we are being silly
- Much of what we do is because we really have to manage costs to host many tiny doctor practices
- Our typical machine is Dell 730 12 drive (4 or 8 TB) 768GB ram and 36c/72t CPU we have more than 30 such machines with around 100 clusters and almost 2200 databases