Doing PITR Right
(Point-In-Time-Recovery)

Who Am I?

Stephen Frost

Sr. Database Engineer @ Resonate, Inc.

- Digital Media Company working with big data – PostgreSQL, Hadoop, etc.
- We're Hiring! techjobs@resonateinsights.com

PostgreSQL Major Contributor

- Implemented Roles (8.1)
- Column Level Privileges (8.4)
- Contributions to PL/pgSQL, PostGIS

What is PITR?

- Backup Strategy using PG's Write-Ahead-Log (WAL)
  - All changes are written to WAL first
  - WAL used for crash recovery
- PITR requires
  - Full backup
  - WAL files since last full backup
- Full backup can be taken while DB is on-line

Why PITR?

- What about pg_dump?
  - Single-threaded, not practical for large-scale databases
  - Restore can be parallel, but still very slow
    - Data has to be re-parsed
    - Indexes must be rebuilt
  - Keeps a very long running transaction open..
- But we have replication!
  - What happens when you drop a table on the master?

archive_command

- Simple – NEVER overwrite files, so check for them first
- test ! -f /mnt/server/archivedir/%f && \
- cp %p /mnt/server/archivedir/%f'
- Advanced – Test, test, test! Verify return codes.
- my_shell_script.sh %p %f
- Remote – Minimal and really insufficient- needs more
- scp %p remote:path/%f
- Ensure the archive command ONLY returns

Backing up PG

- Configure PG for archiving **first**!
  - (and check that's it's **working**)
- Before copying files, run:
  - psql -c “select pg_start_backup('mylabel', true);”
  - 'mylabel' can be anything; might use where the backup is stored to..
- Copy all files in the 'data' directory, using 'rsync' or 'tar'
  - Make sure to include all tablespace directories!

pg_basebackup

- Makes that whole backup thing WAY easier
- Configure PG for archiving **first**!
  - (and check that's it's **working**)
- Uses the PG replication protocol
  - Needs max_wal_senders set > 0
  - Connects to the running PG database
  - Streams the data files over the connection
- Important arguments
  - -D – directory to output files to; tablespaces go to same path as on master
  - -F format (plain or tar)

pg_receivexlog

- Streams transaction log to files from PG
- Connects to PG using replication protocol
- Removes the need for archive_timeout
- Important arguments:
  - -D dir; directory to store the files
- Still use archive_command!
  - Have it test that the file has been archived
  - sleep 5 && test -f /mnt/server/archivedir/%f
  - Sleep required due to async replication

WAL-e

- System to push PG backups and WAL to Amazon's S3
- http://github.com/wal-e/wal-e
- Includes:
  - Compression
  - Encryption
  - Full base backups
  - WAL files
  - Restore of base backups
  - Restore of WAL files
- Used extensively by Heroku

Restoring!

- Test your backups!
- Test by **doing a restore!**
- Test **regularly!** (more than once a year..)
- Test multiple scenarios
  - What if you had to restore from tape?
  - From off-site backups?
  - Fail-over to your 2nd site?

Restoring with PITR

- Restore your full backup
  - Ideally, somewhere **else**.
  - pg_xlog should be empty or not there
  - Ensure it exists with correct perms
  - Verify tablespace symlinks and files
  - If old system still around:
    - Copy files from the old pg_xlog into the new
    - May have unarchived files, allowing restore to closer to time of crash

recovery.conf

- Create a recovery.conf file
  - restore_command – command used to retrieve archived xlog files
    - %f – filename to be restores
    - %p – locataion to restore file to
    - Must only return zero on success
    - Will be asked for files that were not archived
  - Recovery target options: recovery_target_....
    - name – Point created with

Simple restore

- Simple recovery.conf
  - restore_command = 'cp /mnt/server/archivedir/%f "%p"'
  - recovery_target_time = '2013-03-19 12:00'
  - pause_at_recovery_target = false
- Recovers up to the specified time
- Immediately moves into 'on-line' mode at end

Advanced PITR Restore

- More complex recovery.conf
  - restore_command = 'myscript %f %p'
  - recovery_target_xid = '1234'
  - pause_at_recovery_target = true
- recovery_target_xid would need to come from user log files which include xids
- Pauses after recovery, allows user to connect and issue queries to check if they are at the right spot.
- If recovered to the right point, run to complete recovery:

Demo?

Questions?

resonate

Thank You

Stephen Frost