Cybertec PostgreSQL Training

Ants Aasma

www.postgresql-support.de

Introduction

About me

- Software developer for 15 years, with 12 years of experience on PostgreSQL
- Doing PostgreSQL support and development for 4 years, including core PostgreSQL development.
- Some development and sysadmin experience with MySQL and Oracle.

Organizational matters

- Feel free to ask questions at any time.
- Schedule is flexible.

Structure of the training

- Day 1: Installation, Architecture, Indexes
- Day 2: Transactions, Vacuuming
- Day 3: Replication
- Day 4: Backups, Security
- Day 5: Window functions, stored procedures, JSON, FDW

Installing PostgreSQL

Installation sources

- Distribution packages
- PGDG packages
- Custom packages
- Build from source

Distribution packages

- Simplest, but usually out-of-date
- RHEL6 is on PostgreSQL 8.4, not supported since 2014
- RHEL7 is on PostgreSQL 9.2, 3+ years old, 2 years away from being EOL

PGDG packages

- RPM repository maintained by the PostgreSQL project
- Repos for RHEL, CentOS, Fedora, Scientific Linux, Amazon Linux, ...
- New package versions on official release days.
- http://yum.postgresql.org/

Custom packages

- Need an early version of a bugfix.
- Custom feature or a tweak.
- Need different filesystem layout (e.g. parallel installs of minor versions)
- Need special build settings (better profiling support)
- User SRPM from PGDG as a starting point

Build from source

- Mostly useful for developers
- Use it if you want to test out a new unreleased feature.
- Easy to do a standalone install for testing

```
yum-builddep postgresql
git clone git://git.postgresql.org/git/postgresql.git
cd postgresql
./configure --prefix=/tmp/my-custom-pg
make install
```

Other tips

- Install `-contrib` to have performance monitoring tools available
- If installing new packages is a long process, also install `-debuginfo` when deploying.

Setting up a new database cluster

- The first step is to initialize data directory
- PostgreSQL does initialization with initdb
- Redhat packaging includes a script to run initdb
  - `/usr/pgsql-9.4/bin/postgresql94-setup initdb`
  - Create a new version of `/etc/systemd/system/postgresql.service` to change data directory location
  - Set `PGSETUP_INITDB_OPTIONS` environment variable to specify initdb options

Configuring PostgreSQL

- Configuration files are stored in data directory (`/var/lib/pgsql/9.4/data/`)
- If in doubt `sudo -u postgres psql -c "SHOW config_file"`
- By default listens only on loopback interface, adjust in postgresql.conf
- Also modify pg_hba.conf to allow connections.

Starting PostgreSQL

- With systemd
  - `systemctl start postgresql-9.4.service`
  - ‘systemctl enable postgresql-9.4.service
- Using postgresql tools
  - `sudo -u postgres pg_ctl -D /path/to/data start`
  - `sudo -u postgres pg_ctl -D /path/to/data stop -m fast`

Practice

- Set up a PostgreSQL 9.4 installation.
- Use `--data-checksums` to enable data checksumming.
- Start PostgreSQL and confirm you are able to connect

Practice notes

- pg_hba.conf:

```
host    dbname    username    192.1.2.3/32    md5
```

-
```
CREATE USER foobar PASSWORD 'helloworld';
CREATE DATABASE testdb OWNER foobar;
```

PostgreSQL architecture

Overview

- Multiprocess model with a shared memory segment
- Multiversion concurrency control (MVCC)
- Transaction log redo based crash recovery
- Cost based query planner
- Lots of user visible extension capability

PostgreSQL processes

- Postmaster
  - Startup
  - Backend
  - WAL writer
  - Writer
  - Checkpointer
  - Autovacuum launcher
  - Autovacuum worker
  - Logger
  - Stats collector
  - Background worker
  - WAL Sender
  - WAL Receiver

Postmaster process

- Creates listening sockets
- Starts and supervises all other processes
- Forks a backend
- Controls shutdown
- Creates shared memory structures
- Responsible for restarting if one of the backends crashes

Shared_buffers

- Most important shared memory structure
- All table and index data is accessed by reading into shared_buffers

Startup process

- Recovers database to a safe state after a crash.
- Replays transaction logs in case of a replication slave.

Backend process

- One for each client connection
- Listens for and executes queries

WAL Writer process

- Flushes transaction logs so backends don't have to.
- Writes to operating system buffers, but does not sync to disk.
- By default runs 5x per second
- Unexpected exit causes Postmaster to restart

Writer process

- Previously also known as background writer (bgwriter).
- Writes out modified pages that might soon be evicted from shared_buffers so backends don't have to.

Checkpointer process

- Responsible for periodically flushing persistent data to disk to limit transaction log size and recovery time
- Was split out from background writer in version 9.2

Autovacuum launcher/worker processes

- Launcher periodically queries database statistics to see if they need cleaning
- Asks postmaster to fork worker processes to run vacuum on tables that need cleaning
- Also collects statistics about data distribution for query planning

Logger process

- Captures log output from all subprocesses and writes it to logs

Stats collector

- Collects, merges and writes out database usage statistics.
- Examples:
  - Number blocks read from an index
  - Number of deleted rows not cleaned up
  - Number of index scans performed
- Stats collector is not related to data statistics.

Background workers

- Extension mechanism for arbitrary tasks inside the database
- New in 9.3
- Similar to normal backend
- Examples:
  - Cron like execution of periodic tasks
  - Kill connections with idle transactions
  - Measure replication lag
  - Create and drop partitions
  - (in 9.6) execute parts of the query in parallel.

WAL Sender/WAL Receiver

- Replication processes that transfer transaction logs from master to slave.
- Sender is started as a regular backend, switches mode.
- Receiver is started by the startup process.