PostgreSQL Unboxing

Christophe Pettus
PostgreSQL Experts, Inc.

thebuild.com
pgexperts.com

Welcome!

- Christophe Pettus
- Consultant with PostgreSQL Experts, Inc.
- Based in sunny San Francisco, California.
- Technical blog: **thebuild.com**
- Twitter: **@xof**
- **cpettus@pgexperts.com**

PGX POSTGRESQL EXPERTS, INC.

My background.

- PostgreSQL person since 1998.
- Came to databases as an application developer and architect.
- I had to suffer for my art.
- Now, it's your turn!

PGX
POSTGRESQL
EXPERTS, INC.

What is this?

- “Just enough” PostgreSQL for a new developer.
- PostgreSQL is a rich environment.
- Far too much to learn in a single tutorial.
- But enough to be dangerous!

PGX POSTGRESQL EXPERTS, INC.

The DevOps World

- "Integration between development and operations."
- "Cross-functional skill sharing."
- "Maximum automation of development and deployment processes."
- "We're way too cheap to hire real operations staff. Anyway: **Cloud!**"

This means…

- No experienced DBA on staff.
  - Have you seen how much those people *cost*, anyway?
- Development staff pressed into duty as database administrators.
- But it's OK… it's **PostgreSQL**!

Everyone Loves PostgreSQL!

- Fully ACID-compliant relational database management system.
- Richest set of features of any modern production RDMS.
- Relentless focus on quality, security, and spec compliance.
- Capable of very high performance.

PGX POSTGRESQL EXPERTS, INC.

But, it's hard to administer!

* This machine was bought in 1997.
* It is running PostgreSQL 9.2.1.
* I spend 10 minutes a year on maintenance.

Background

Elephant evolution.

- Derives from the 1986 POSTGRES project at the University of California, Berkeley.
- This also gave rise to Illustra, thence Informix.
- And thence Sybase, and from there MS SQL Server.
- Proudly open source since 1995.

Licenses matter.

- Licensed under the PostgreSQL License, similar to BSD/MIT.
- Allows for commercial derivatives, but…
- … not owned by a commercial organization.
- No one will take your elephant away.

Cross-Platform.

- Operates natively on all modern operating systems.
  - Plus Windows.
- Scales from development laptops to huge enterprise clusters.

PGX POSTGRESQL EXPERTS, INC.

Work in progress.

- Under constant development.
- A major release every 9-12 months or so.
- Constant minor releases.
- Vanishingly small security or data corruption bugs.
- Community focus on correctness and data integrity.

PGX POSTGRESQL EXPERTS, INC.

A quick spin around the elephant.

- PostgreSQL is the most feature-rich open source database, full stop.
  - Focus on “big database” features.
  - High-rate OLTP, data warehousing…
  - Equals or exceeds commercial DBs.
- Far more features than we can discuss here.

PGX POSTGRESQL EXPERTS, INC.

But we'll try!

- Huge range of integrated types.
- User-definable types.
- Built-in fast, multi-language full-text search.
- Extremely extensible.

Rich Data Types.

- Numeric
- Character
- Date/Time
- Boolean
- Enums
- Geometric
- Network Addresses
- Bit Strings
- Text-Search Related
- UUID
- XML

PGX POSTGRESQL EXPERTS, INC.

Powerful Extensions.

- PostGIS – De-facto standard geographic information system.
- Integrated programming languages
  - Python, Perl, Ruby, Java, R…
  - Coming soon, integrated JSON and V8.
  - It's WebScale™!

Installation

A variety of methods.

- Build from source.
  - Works on any platform.
  - Maximum control.
- Requires development tools.
- Does not come with platform-specific utility scripts (/etc/init.d, etc.).

Packages.

- Packages available for all major Linux platforms.
  - May need to use custom repositories.
- http://www.postgresql.org/download/linux/
- Debian-derived and RedHat-derived have different directory structures.
- We'll discuss those in a bit.

PGX POSTGRESQL EXPERTS, INC.

Other OSes.

- Windows: One-click installer available.
- OS X: One-click installer, MacPorts, Fink and Postgres.app from Heroku.
- For other OSes, check postgresql.org.

Creating a database cluster.

- A single PostgreSQL server can manage multiple databases.
- The whole group on a single server is called a “cluster”.
- This is very confusing, yes.

PGX POSTGRESQL EXPERTS, INC.

A Database.

- Databases are autonomous collections of objects (tables, schemas, etc.).
- You cannot directly join between them.
  - Foreign data wrappers coming soon!
- MySQL "databases" are PostgreSQL "schemas," more or less.

initdb

- The command to create a new database cluster is called initdb.
- It creates the files that will hold the cluster.
- It doesn't automatically start the server.
- Many packaging systems automatically create and start the server for you.

pg_ctl

- Built-in command to start and stop PostgreSQL.
- Frequently called by init.d, upstart or other scripts.
- Use the package-provided scripts.

psql

- Command-line interface to PostgreSQL.
- Run queries, examine the schema, look at PostgreSQL's various views.

PostgreSQL directories

- All of the data lives under a top-level directory.
- Let's call it $PGDATA.
  - Find it on your system, and do a ls.
  - The data lives in "base".
  - The transaction logs live in pg_xlog.

Configuration files.

- On most installations, the configuration files live in $PGBASE.
- On Debian-derived systems, they live in /etc/postgresql/9.2/main/...
- Find them. You should see:
  - postgresql.conf
  - pg_hba.conf

Configuration

Configuration files.

- Only two really matter:
  - postgresql.conf — most server settings.
  - pg_hba.conf — who gets to log in to what databases?

Users and roles.

- A “role” is a database object that can own other objects (tables, etc.), and that has privileges (able to write to a table).
- A “user” is just a role that can log into the system; otherwise, they’re synonyms.
- PostgreSQL’s security system is based around users.

pg_hba.conf

postgresql.conf

- Holds all of the configuration parameters for the server.
- Find it and open it up on your system.

postgresql.conf

We’re All Going To Die.

PGX POSTGRESQL EXPERTS, INC.

It Can Be Like This.

Important parameters.

- Logging.
- Memory.
- Checkpoints.
- Planner.
- You’re done.
- No, really, you’re done!

PGX POSTGRESQL EXPERTS, INC.

Logging.

- Be generous with logging; it's very low-impact on the system.
- It's your best source of information for finding performance problems.

PGX POSTGRESQL EXPERTS, INC.

Where to log?

- syslog — If you have a syslog infrastructure you like already.
  - LOCAL0.* -/var/log/postgresql
- Otherwise, CSV format to files.
- Do not use standard format; it's obsolete.

What to log?

```
log_destination = 'csvlog'
log_directory = 'pg_log'
logging_collector = on
log_filename = 'postgres-%Y-%m-%d_%H%M%S'
log_rotation_age = 1d
log_rotation_size = 1GB
log_min_duration_statement = 250ms
log_checkpoints = on
log_connections = on
log_disconnections = on
log_lock_waits = on
log_temp_files = 0
```

PGX POSTGRESQL EXPERTS, INC.

Memory configuration

- shared_buffers
- work_mem
- maintenance_work_mem

shared_buffers

- Below 2GB (?), set to 20% of total system memory.
- Below 32GB, set to 25% of total system memory.
- Above 32GB (lucky you!), set to 8GB.
- Done.

Shared memory follies.

- PostgreSQL allocates all shared memory at startup.
- Most Linux kernels don't allow much shared memory allocation.
- Relevant parameters are SHMMAX and SHMALL.

To adjust.

- Calculate: shared_memory in bytes, +20%.
- sysctl -w kernel.shmmax = (value)
- sysctl -w kernel.shmall = (value)/4096
- Sorry about that; there's a lot of history there.

OOM Killer Considered Harmful.

- The Linux OOM killer is a bug, not a feature, on PostgreSQL servers.
  - vm.overcommit_ratio = 100
  - vm.overcommit_memory = 2
  - Swap = RAM.

PGX POSTGRESQL EXPERTS, INC.

work_mem

- Start low: 32-64MB.
- Look for ‘temporary file’ lines in logs.
- Set to 2-3x the largest temp file you see.
- Can cause a **huge** speed-up if set properly!
- But be careful: It can use that amount of memory per planner node.

maintenance_work_mem

- 10% of system memory, up to 1GB.
- Maybe even higher if you are having VACUUM problems.
  - (We'll talk about VACUUM later.)

effective_cache_size

- Set to the amount of file system cache available.
- If you don't know, set it to 50% of total system memory.
- And you're done with memory settings.

Checkpoints.

- A complete flush of dirty buffers to disk.
- Potentially a lot of I/O.
- Done when the first of two thresholds are hit:
  - A particular number of WAL segments have been written.
  - A timeout occurs.

PGX POSTGRESQL EXPERTS, INC.

Checkpoint settings.

```
wal_buffers = 16MB

checkpoint_completion_target = 0.9

checkpoint_timeout = 10m-30m # Depends on restart time

checkpoint_segments = 32 # To start.
```

Checkpoint settings, 2.

- Look for checkpoint entries in the logs.
- Happening more often than checkpoint_timeout?
  - Adjust checkpoint_segments so that checkpoints happen due to timeouts rather filling segments.
- And you're done with checkpoint settings.

Checkpoint settings notes.

- The WAL can take up to 3 x 16MB x checkpoint_segments on disk.
- Restarting PostgreSQL can take up to checkpoint_timeout (but usually less).

Planner settings.

- effective_io_concurrency — Set to the number of I/O channels; otherwise, ignore it.
- random_page_cost — 3.0 for a typical RAID10 array, 2.0 for a SAN, 1.1 for Amazon EBS.
- And you're done with planner settings.

Do not touch.

- fsync = on
  - Never change this.
- synchronous_commit = on
  - Change this, but only if you understand the data loss potential.

Changing settings.

- Most settings just require a server reload to take effect.
- Some require a full server restart (such as shared_buffers).
- Many can be set on a per-session basis!

Concepts

Write-Ahead Log, an introduction.

- A continuous stream of (committed) database modifications.
- Broken into 16MB files, called "segments."
- Logically, starts with database cluster creation and lasts forever.
- In reality, that would be insane.

WAL, what is it good for?

- Used to restore the database on an abnormal termination.
- Absolutely essential to avoid data corruption.
- The replay has to happen from the last consistent state.
  - = the last time a checkpoint finished.

Important WAL Facts.

- It is time-ordered, so you can replay it to a particular point in time.
- It is append-only, so it pays to put it on its own file system.
- It is the basis for both warm standby and streaming replication.

MVCC

- Multi-Version Concurrency Control.
- Introduced by PostgreSQL, now used by pretty much everyone.
- Alternative to “pessimistic” locking strategies.
- Allows for much higher performance.

MVCC rules.

- Readers (to the same row) do not block readers.
- Writers do not block readers — readers get the old version of the row.
- Readers do not block writers.
- Writers **do** block writers to the same row.

Versioning.

- Multiple versions of the same row can exist.
- Deleted and updated rows are not immediately removed from the database.
  - Some other transaction might still be able to see them.
- Solution? VACUUM.

PGX
POSTGRESQL
EXPERTS, INC.

VACUUM

- Scans each table for “dead” versions of tuples, and marks them as free.
- Since 8.0, handled for you by the autovacuum daemon.
- Good to manually vacuum after major update/delete operations.

ANALYZE

- The planner requires statistics on each table to make good guesses for how to execute queries.
- ANALYZE collects these statistics.
- Done as part of VACUUM.
- Always do it after major database changes — especially a restore from a backup.

Locking.

- PostgreSQL takes implicit locks on objects to maintain concurrency control.
- Tuple locks on are on individual database rows.
- Table, schema and database locks are on higher-level objects.

Tuple locks.

- Share lock — Prevents the row from being modified, but it can be read; any number of sessions can hold a shared lock on the same row.
- Exclusive lock — Prevents the row from being modified by anyone else; only one session can hold an exclusive lock.

Surprising locks.

- Writing a dependent row can cause a share lock on the parent in a foreign key relationship.
- Updates on that parent row can then block.
- This is the reason for the fast/slow data rule.

Table-level locks.

- Taken during schema modifications.
- Held only for as long as the schema modification goes on.
- But this can be a very long time if you are adding a non-NULL column.
- Add column as NULL, set to non-NULL default later.

Explicit locking.

1. Taking an explicit lock on a table is a sign of an application problem.
2. If you think you can only solve your problem with an explicit lock, see #1.
3. If you are **sure** you can only solve your problem with an explicit lock, see #2.

Transaction modes.

- PostgreSQL supports multiple transaction modes.
  - READ COMMITTED
  - REPEATABLE READ
  - SERIALIZABLE

PGX POSTGRESQL EXPERTS, INC.

A reminder about MVCC.

- All transactions see a snapshot of the database at the start of a transaction.
- Only writes to the *same tuple* (row) block.
- The transaction isolation levels control how "perfect" this snapshot model is.

READ COMMITTED

- Each transaction sees a snapshot of the database at the time it starts.
- Pure read-only transactions are always consistent.
- Transactions lock, but do not fail.
- Conflicting writes to the same row can cause an inconsistent snapshot.

READ COMMITTED

- BEGIN;
- SELECT i FROM t WHERE k=3;
  -- Other transaction sets i to 7.
- UPDATE t SET i=12 WHERE k=3;
- COMMIT;

READ COMMITTED

- BEGIN;
- SELECT i FROM t WHERE k=3
  FOR UPDATE;
- UPDATE t SET i=12 WHERE k=3;
- COMMIT;

PGX POSTGRESQL EXPERTS, INC.

REPEATABLE READ

- Each transaction gets a perfectly consistent snapshot.
- Multiple writes to the same row can cause a transaction to be aborted.
- The aborting transaction can then be rerun.
- Not true serializable transactions.

REPEATABLE READ

- BEGIN;
- SELECT MAX(last_inserted_batch)…
- Insert any newer records.
- COMMIT;
- No traditional solution except a full table lock (or equivalent).

SERIALIZABLE

- New in 9.1!
- True mathematical serializability.
- Has overhead (although not much).
- As with REPEATABLE READ, transaction aborts can result.

SERIALIZABLE success.

- BEGIN;
- SELECT MAX(last_inserted_batch)…
- Insert any newer records.
  - Losing transaction aborts here.
- COMMIT;

Transaction philosophy.

- Keep transactions short.
- Do not leave transactions open during asynchronous events.
- Long-running transactions can create a myriad of problems.

Schema Design

A huge topic.

- We can only scratch the surface here.
- In general:
    - Keep your data in normal form.
    - Do not be afraid to do joins.
    - Do not denormalize except in response to a very real problem.

Fast/slow rule.

- Do not put fast changing data in the same table as slow changing data.
  - Especially if the table is the parent of a lot of other tables via foreign keys.
- This will avoid a large class of locking problems.

Indexing strategy.

- A good index is:
  - Highly selective.
  - Frequently used.
  - Or required to enforce a constraint.
- A bad index is:
  - Everything else.

Index creation.

- Create indexes on the basis of real-life queries.
- Look for sequential scans that can be sped up.
- Indexes are not cheap; drop any that are not being used.

Checking index usage.

- pg_stat_user_tables
- pg_stat_user_indexes
  - Look for lots of sequential scans, or
  - Not many index scans.

Pitfalls.

SELECT COUNT(*) FROM ...

- Everyone does this.
- SELECT COUNT(*) FROM MyHugeTable;
  - "Why is PostgreSQL so slow?"
- Implemented as a full table scan.
- So, don't do this.

PGX POSTGRESQL EXPERTS, INC.

In-place upgrade.

- Upgrading major versions (9.0 > 9.1) requires a pg_dump and pg_restore.
- No in-place upgrade in core yet.
- pg_upgrade is a thing.
- Trigger-based replication is another option.

autovacuum

- Background process that does VACUUMing.
- Handles most workloads well.
- Sometimes, can wake up at exactly the wrong time.
- Or run wild.

Manual VACUUM

- Disable AUTOVACUUM.
- Run VACUUM manually at low-load times.
- You **must** run VACUUM!
- Be sure to do ANALYZE at the same time.

Bulk loading data.

- Use COPY, not INSERT.
- COPY does full integrity checking and trigger processing.
- Do a VACUUM afterwards.

Debugging

“This query is slow.”

- EXPLAIN or EXPLAIN ANALYZE
- The output is… somewhat cryptic.
- http://explain.depesz.com/

“The database is slow.”

- What’s going on?
- pg_stat_activity
- tail -f the logs.
- Too much I/O? iostat 5

PGX POSTGRESQL EXPERTS, INC.

“The database isn’t responding.”

- Make sure it’s up!
- Can you connect with psql?
- pg_stat_activity
- pg_locks

Backup

pg_dump

- Built-in dump/restore tool.
- Takes a logical snapshot of the database.
- Does not lock the database or prevent writes to disk.
- Low (but not zero) load on the database.

pg_restore

- Restores database from a pg_dump.
- Is not a fast operation.
- Great for simple backups, not suitable for fast recovery from major failures.

PGX POSTGRESQL EXPERTS, INC.

Point-in-time recovery

- Combine file-system level snapshots of the database with archives of the WAL file.
- File system snapshot does not need to be atomic or consistent.
- Can be used to recover to a particular point-in-time in case of logical-level failures.

PITR process, an overview.

- Start archiving WAL segments.
- Make a base backup.
- Keep archiving WAL segments.
- Lather, rinse, repeat.
- The WAL segments plus the base backup are your backup.

PGX
POSTGRESQL
EXPERTS, INC.

Archive WAL segments.

- archive_mode = on
- archive_command = …
- Archive these to a different machine than your primary database server!
- On a cloud host? Make sure your archive machine is really on different physical hardware.

Do a base backup.

- pg_start_backup('label', true);
- Do a file-system level copy.
- Yes, it will be inconsistent. No, we don't care.
- PostgreSQL continues operating.
- pg_stop_backup('label', true);

Keep archiving WAL segments.

- The WAL segments plus the base backup are your backup.
- You can replay the WAL segments to any point in time (after the backup was complete).
- When it's been too long to recover quickly, do another base backup.

Downsides.

- 16MB a piece for each WAL segments.
- Replaying WAL segments takes a while.
- Doing the base backup may be prohibitively long.
- The next step in data security is…

PGX POSTGRESQL EXPERTS, INC.

Warm standby.

- Secondary server continually integrates the WAL segments.
- Can come back up instantly in the event of primary failure.
- You still need the base backup and WAL segments to do PITR recovery though.
  - You can't go back in time.

Warm standby, the bad news.

- The secondary cannot be used for queries.
  - No load balancing here.
- Managing the WAL segments can be a pain.
- The secondary moves forward only as fast as the WAL segments can be moved.
- The next step, then, is…

Replication

Replication options.

- PostgreSQL's built-in streaming replication.
- Trigger-based replication:
  - Slony
  - Bucardo
  - Londiste

PGX
POSTGRESQL
EXPERTS, INC.

Built-in replication.

- Available in the core since 9.0.
- Read/write master.
- Read-only secondaries.
- Single-level tree of secondaries to one master.

Advantages.

- Very fast.
- Secondaries can be queried, making it great for load balancing.
- DDL changes are automatically pushed to secondaries.

PGX
POSTGRESQL
EXPERTS, INC.

The bad news.

- All-or-nothing: The entire database cluster must be replicated.
- Any change is immediately propagated, including your mistakes.
- Requires some tuning for query cancellation issues.

Setting up.

- Do a base backup.
- Set up recovery.conf correctly.
- Bring up the secondary.
- Profit.

Tools for replication.

- pg_basebackup — Built in tool for doing a base backup to prime a secondary.
- repmgr.org — Prepackaged tools for setting up and monitoring replication.

PGX POSTGRESQL EXPERTS, INC.

Trigger-based replication.

- Installs triggers on tables on master.
- A daemon process picks up the changes and applies them to the secondaries.
- Third-party add-ons to PostgreSQL.

Trigger-based rep: Good.

- Highly configurable.
- Can push part or all of the tables; don't have to replicate everything.
- Multi-master setups possible (Bucardo).

Trigger-based rep: Bad.

- Fiddly and complex to set up.
- Schema changes must be pushed out manually.
- Imposes overhead on the master.

Pooling, etc.

Why pooling?

- Opening a connection to PostgreSQL is expensive.
- It can easily be longer than the actual query time.
- Above 200-300 connections, use a pooler.

pgbouncer

- Developed by Skype.
- Easy to install.
- Very fast, can handle 1000s of connections.
- Does not to failover, load-balancing.
    - Use HAProxy or similar.

pgPool II

- Does query analysis.
- Can route queries between master and secondary in replication pairs.
- Can do load balancing, failover, and secondary promotion.
- Higher overhead, more complex to configure.

Hardware, System

Cloud hosting.

- Cloud hosting has terrible I/O.
- Databases (above a certain size) are I/O bound.
- You can see where this is going.

PGX POSTGRESQL EXPERTS, INC.

Making the best of a bad situation.

- Get as much RAM as you can afford (up to 2x database size).
- CPU capacity is not as important as RAM.
- Make sure the underlying storage system is reliable.

Playing safe.

- Always use replication.
- Make sure your replica is on a different physical machine than the primary.
  - EC2 has client-affinity for boxes.
- Store configurations, etc. in a VCS; machines can die unexpected.

PGX POSTGRESQL EXPERTS, INC.

Speaking of EC2.

- EBS striping can get you some performance benefit…
- … at the expense of the EBS snapshot capability.
- Ubuntu 11.04 seems to be the most stable.
- EBS can fail.
- Check out WAL-E from Heroku.

Your own(-ish) hardware.

- SSDs if you can afford it, SAS drives otherwise.
  - RAID10.
- Put pg_xlog on its own volume.
- Move pg_stat_tmp to a RAM disk.
- xfs (or ext4).

Tools

Monitor, monitor, monitor.

- Use Nagios / Ganglia to monitor:
  - Disk space — at minimum.
  - CPU usage
  - Memory usage
  - Replication lag.
- check_postgres.pl (bucardo.org)

PGX
POSTGRESQL
EXPERTS, INC.

Graphical clients

- pgAdmin III
  - Comprehensive, open-source.
- Navicat
  - Commercial product, not PostgreSQL-specific.

Log Analysis

- pgFouine
  - Traditional, not maintained much anymore.
  - Requires a patch for 9.1 log files.
- pgbadger
  - Brand new, actively maintained.

Questions?

Thank you!

@xof
cpettus@pgexperts.com