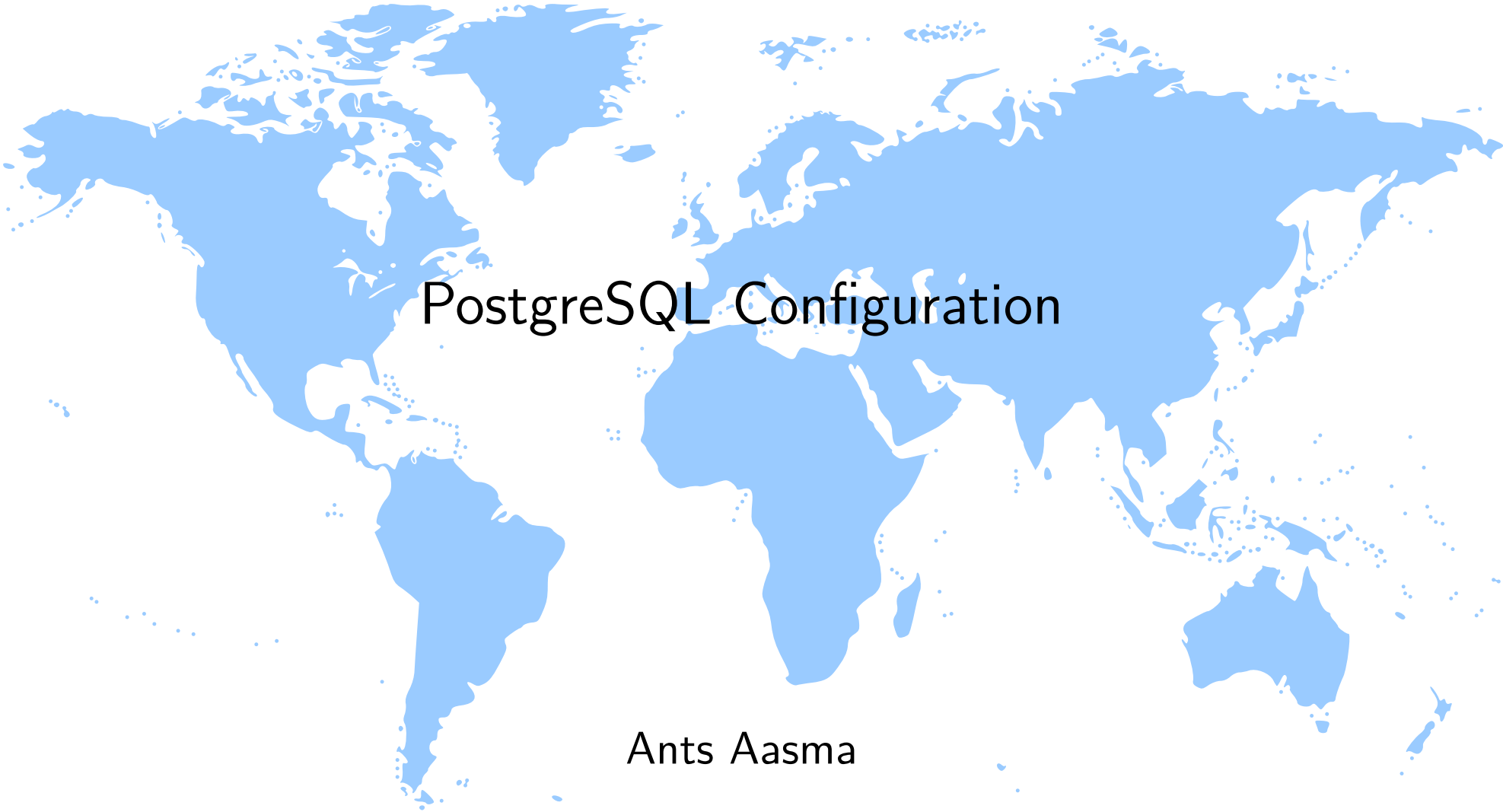

PostgreSQL Configuration

Ants Aasma

www.cybertec.at

GUCs

Introduction

- PostgreSQL configuration settings are called GUCs.
  - Grand Unified Configuration

Configuration sources

- `postgresql.conf`
- Command line parameters when start server process
- `ALTER DATABASE ... SET timezone = 'UTC';`
- `ALTER ROLE ... SET synchronous_commit = 'off';`
- `SET work_mem = '100MB';`
- `BEGIN; SET LOCAL random_page_cost = 1;`
- `CREATE FUNCTION ... SET enable_seqscan = off`
- `PGOPTIONS="-c post_auth_delay=0s" psql`

Configuration value datatypes

- Boolean
- String
- Numeric (Integer/Float)
- Numeric with unit (Memory/Disk/Time)
- Enum

Datatypes basic

- Boolean
  true/false on/off yes/no 1/0
  `track_io_timing = on`
- String
  - Generally use single quotes.
  - Double single quotes to escape.

Datatypes numeric

- Numeric
  Can be integer or floating point. Integers can't contain a decimal point.

```
max_connections = 100
random_page_cost = 1.23
```

Datatypes numeric with unit

- Numeric with unit
  - Have some implicit unit, for example blocks, seconds, kilobytes. Check pg_settings if you really want to know. Otherwise use human readable units
  - Memory is kB, MB, GB, TB (1024 based)
  - Time is ms, s, min, h, d

```
work_mem = 10MB
checkpoint_timeout = 30min
```

Datatypes continued

- ENUMs
  Predefined set of values, check pg_settings.enumvals, or the docs.
  `synchronous_commit = remote_write`

GUC contexts

- `user` - Can be set in each session. Can be set on Roles or databases.
- `superuser` - Can be set at runtime, but only by superusers.
- `backend` - Can be set by superuser when connecting. Not used much.
- `sighup` - Requires configuration reload.
- `postmaster` - Requires databse server restart.
- `internal` - Built in value

Configuring paths

- Paths are relative to data directory.

Connections

Connection options

- `listen_addresses = 'localhost'`
  - Usually `'*'` is the desired value.
- `port = 5432`
  - Use default if possible.
- `max_connections = 100`
  - 100 may not be enough
- `superuser_reserved_connections = 3`
  - Default is mostly ok. Useful to know that it's available.

SSL

- `ssl = off`, `ssl_cert_file`, `ssl_key_file`, `ssl_ca_file`, `ssl_crl_file`
- Must generate server keys to enable connection encryption.
- If you have PKI infrastructure in place it makes good sense to use SSL based authentication.
- If no PKI in place, self signed cert is better than nothing.

Keepalive

- `tcp_keepalives_idle`, `tcp_keepalives_interval`, `tcp_keepalives_count`
- Uses TCP protocol level keepalives.
- Useful if you have clients that keep getting their idle connection disconnected.
- Can also be set when connecting.

Resource usage

Buffers

- `shared_buffers = '128MB'`
  - Rule of thumb: 25% of memory
  - Changing requires restart
- `huge_pages = try`
  - Makes PostgreSQL use larger page size for shared_buffers allocation.
  - Noticeable performance boost for CPU bound workloads.
  - Less memory used per backend with huge shared_buffers setting.
  - Need to set `vm.nr_hugepages` in kernel

Buffers, continued

- `temp_buffers = '8MB'`
  - Same purpose as shared buffers, but backend local for temporary tables.
  - Can be changed by user as needed.
- `work_mem = '4MB'`
  - Controls how much memory backends are allowed to allocate for sorting, hash joins, etc.
  - Each executor node that needs a buffer will use this settings worth of memory.
  - User settable.

Buffers, continued 2

- `maintenance_work_mem = '64MB'`
  - Used for index creation, vacuuming and foreign key creation.
  - User changeable.
  - Each autovacuum worker will use up to this amount.
- `max_stack_depth = '2MB'`
  - Probably don't need to change this.
- `dynamic_shared_memory_type`
  - Relevant for background workers.
  - Default is OK.

Files

- `temp_file_limit = -1`
  - May want to set some reasonably high limit to avoid nasty surprises.
  - superuser setting
- `max_files_per_process = 1000`
  - Default is reasonable

2 phase transactions

- `max_prepared_transactions = 0`
  - Turned off by default to avoid a foot gun.
  - Java applications often want this.
  - Having a transaction manager or at the very least monitoring is required.

Background writer

- `bgwriter_delay = '200ms'`
  `bgwriter_lru_maxpages = 100`
  `bgwriter_lru_multiplier = 2.0`
- Default will write out 4MB/s (8kB*100/0.2s)
- Check `pg_stat_bgwriter.buffers_backend` if it's increasing it might be worth it to make background writer more aggressive

Async

- `effective_io_concurrency = 1`
- Sets how many async I/Os PostgreSQL will keep in flight.
- Currently only used for bitmap heap scans.

WAL settings

Logging level

- `wal_level = 'minimal'`
- minimal $<$ archive $<$ hot_standby $<$ logical
- Size and performance difference between archive, hot_standby and logical is pretty small.
- Minimal can skip significant amount of WAL logging for bulk operations, but PITR is not possible.

Durability

- `fsync = on`
  - Turning of never syncs anything to disk. Only use when data integrity is not important.
  - To safely go from off->on shut down database, change setting, issue OS level `sync` and then start up.
- `synchronous_commit = 'on'`
  - off - some transactions may be lost if server crashes
  - local - some transactions may not arrive on standby in case of a crash
  - remote_write - locally crash safe, all transactions are replicated to standby
  - on - all transactions are crash safe on local and standby
  - Can be set per transaction.

Durability 2

- `wal_sync_method = open_datasync`
  - On Linux no reason to use anything else
- `full_page_writes = on`
  - Almost never safe to turn off. Useful with fsync=off.
  - In addition to safety speeds up recovery on standby.
- `wal_log_hints = off`
  - Useful for pg_rewind.

Durability 3

- `wal_buffers = -1`
  - Default = 3% of shared buffers, 16MB max.
  - Rarely useful to increase.
- `wal_writer_delay = 200ms`
  - Default is good enough.
- `commit_delay = 0`, `commit_delay_siblings = 5`
  - Waits before commit to merge multiple flushes.
  - Can be useful with WAL on spinning disks, no BBU and high write load. But SSD or BBU is a better solution.

Checkpoints

- `checkpoint_timeout = 5min`
  - Larger values result in less writes due to write merging.
  - More WAL to replay means more recovery time.
- `checkpoint_completion_target = 0.5`
  - Usually set to 0.9 for more uniform performance.
- `checkpoint_warning = 30s`

WAL size before 9.5

- `checkpoint_segments = 3`
  - Measured in 16MB segments.
  - Maximum disk use is around (2 + ckpt_compl_target)*ckpt_segments + 1 + wal_keep_segments

WAL size in 9.5

- `min_wal_size = '80MB'`
  `max_wal_size = '128MB'`
- Uses a moving average to estimate the number of files needed, doesn't use up all the space if it isn't needed.
- Soft limit, wal_keep_segments, archive_command or heavy load can still cause it to be exceeded.

Archiving

- `archive_mode = off`
- `archive_command = ''`
  - Turning archiving on causes WAL to be kept around until `arhcive_command` succesfully archives it.
- `archive_timeout = 0`
  - If you want WAL changes to reach the archive in a timely manner on idle systems use this to force a WAL segment switch after a timeout.

Replication settings (later)

Query planning

Execution methods

- `enable_bitmapscan`, `enable_hashagg`, ...
- Can disable problematic execution nodes to force a different planb.
- `enable_nestloop = off` is most commonly useful.

Planner costs

- `seq_page_cost`, `random_page_cost`, `cpu_tuple_cost`, `cpu_index_tuple_cost`, `cpu_operator_cost`
  - Discussed earlier.
- `effective_cache_size = '4GB'`
  - Does not allocate anything.
  - Larger values will make the optimizer think that nested loops with inner index lookups will hit cache and be cheap.

Genetic optimizer

- `geqo = on`, `geqo_threshold = 12`
  - Join planning is exponenitally hard problem.
  - Uses a genetic algorithm for optimizing large joins.
- `geqo_effort`, `geqo_pool_size`, `geqo_generations`, `geqo_selection_bias`, `geqo_seed`
  - Probably useful to have some experienece with tuning genetic algorithms before tweaking these.

Other options

- `from_collapse_limit = 8`, `join_collapse_limit = 8`
  - Merge up to this number of explicit JOINs or subqueries into one join level.
  - Setting these to 1 allows for explicit join order specification.
- `cursor_tuple_fraction = 0.1`
- `constraint_exclusion = partition`
- `default_statistics_target = 100`
  - Controls how much data ANALYZE collects by default. Larger values means more accurate stats (usually), but slower planning (always).

Logging

Where

- `log_destination = 'stderr'`
  - List of places to log to. Values: stderr, csvlog, syslog
- `logging_collector = on`
- `log_directory = 'pg_log'`
  - Can be convenient for to store outside data directory.
- `log_filename = postgresql-%a.log`
- `log_file_mode - 0600`

Log rotation

- `log_rotation_age`
- `log_rotation_size`
- `log_truncate_on_rotation`

Syslog

- `syslog_facility`
- `syslog_ident`

When

- DEBUG5..1, LOG, NOTICE, WARNING, ERROR, FATAL, and PANIC
- `client_min_messages = 'NOTICE'`
  - What the user receives
- `log_min_messages = 'WARNING'`
  - What is logged on the server
- `log_min_error_statement = 'ERROR`
- `log_min_duration_statement = -1`
  - When to log the offending SQL query.

What

- `application_name` - set by the client connecting.
- `debug_print_parse/rewritten/plan` - Probably not too useful
- `log_checkpoints = off`
  - Use this to see how much data checkpoints are writing out and what fsyncing latency is at the end.
  - Very useful if you have tools that can produce a graph from this data.
- `log_connections = off`, `log_disconnections = off`
  - Useful for auditing
- `log_duration`
- `log_error_verbosity = default`

What 2

- `log_hostname = off`
- `log_line_prefix = '< %t >'`
  - Including remote host, username and database name is useful
- `log_lock_waits = off`
- `log_statement = none`
  - Mostly for auditing. Values: none, ddl, mod, all
- `log_temp_files = -1`
- `log_timezone = 'Europe/Tallinn'`

Runtime statistics

Stats collector settings

- `track_activities = on`, `track_activity_query_size = 1024`
  - Enables pg_stat_activities. Very useful.
- `track_counts = on`
  - Don't turn this off. Needed for autovacuum.
- `track_io_timing = off`
  - Helps understanding where I/O time is spent
  - If pg_test_timing shows <100ns then turning this on is practically free.
- `track_functions = none`
  - Values, none, pl, all. pl would be sensible default.

Stats collector settings

- `update_process_title = on`
- `stats_temp_directory = 'pg_stat_tmp'`

Vacuum configuration

General

- `autovacuum = on`
  - Don't turn it off!
- `log_autovacuum_min_duration = -1`
- `autovacuum_max_workers = 3`
  - Probably increase this
- `autovacuum_naptime = 1min`
  - Usually ok

When to vacuum

- `autovacuum_vacuum_threshold = 50`
- `autovacuum_analyze_threshold = 50`
  - If less than this number of rows changed, don't touch.
- `autovacuum_vacuum_scale_factor = 0.2`
  - Percentage of dead rows in table before vacuuming. Decrease, especially for big tables.
- `autovacuum_analyze_scale_factor = 0.1`
  - Usually decrease.

When definitely vacuum

- `autovacuum_freeze_max_age = 200000000`
  - Maybe increase
- `autovacuum_multixact_freeze_max_age = 400000000`

Autovacuum aggressiveness

- `autovacuum_vacuum_cost_delay = 20ms`
  - Sleep for this long everytime cost is hit
- `autovacuum_vacuum_cost_limit = -1`
  - -1 = use vacuum settings

Vacuum aggressiveness

- `vacuum_cost_delay = 0`
  - Foreground vacuum runs at full tilt.
- `vacuum_cost_page_hit = 1`
- `vacuum_cost_page_miss = 10`
- `vacuum_cost_page_dirty = 20`
- `vacuum_cost_limit = 200`
  - Clean up maximum of (200/20)*8kB/0.02s = 4MB/s
  - Read from disk max 8MB/s
  - Read from cache 80MB/s
  - Increase cost limit for autovacuum!

What to vacuum

- `vacuum_freeze_min_age = 50000000`
  - Decrease this to freeze early
- `vacuum_freeze_table_age = 150000000`
  - Increase this and autovacuum_freeze_max_age to reduce number of anti-wraparound vacuums.
- `vacuum_multixact_freeze_table_age`, `vacuum_multixact_freeze_min_age` ** Same story

Tuning vacuum per table

- `ALTER TABLE ... SET (autovacuum_vacuum_scale_factor = 0.01)`
- `autovacuum_enabled`
- `autovacuum_{vacuum,analyze}_{threshold,scale_factor}`
- `autovacuum_vacuum_cost_{delay, limit}`
- `autovacuum_[multixact_]freeze_{min,max,table}_age`
- `log_autovacuum_min_duration`

Other

Misc 1

- `search_path = "$user", public`
- `default_tablespace = ''`
- `temp_tablespaces = ''`
- `client_encoding = ''`

Transaction isolation

- `default_transaction_isolation = 'read committed'`
- `default_transaction_read_only = false`
- `default_transaction_deferrable = false`

Timeouts

- `statement_timeout = 0`
- `lock_timeout = 0`

Extension module loading

- `local_preload_libraries`
- `session_preload_libraries`
- `shared_preload_libraries`

Lock management

- `deadlock_timeout = '1s'`
- `max_lock_per_transaction = 64`
  - Increase if you have thousands of tables.
- `max_pred_locks_per_transaction = 64`
  - Serializable transactions use these

Error handling

- `exit_on_error = off`
  - Errors kill the connection
- `restart_after_crash = true`
  - May be useful to turn of in a cluster environment.

Overview

Always

- listen_addresses
- shared_buffers
- checkpoint_segments (max_wal_size)

Usually

- work_mem, maintenance_work_mem
- wal_level
- checkpoint_completion_target
- autovacuum_max_workers, autovacuum_analyze_scale_factor, autovacuum_vacuum_scale_factor, autovacuum_cost_limit

To avoid support calls

- temp_file_limit
- statement_timeout

Nice to have

- track_io_timing
- log_line_prefix
- log_checkpoints
- shared_preload_libraries = 'pg_stat_statements'