pgstats in PostgreSQL 18

PostgreSQL Conference Japan 2025
November 21th, Tokyo
Michael Paquier

The lecturer

- French, based in Tokyo.
- PostgreSQL contributor since 2009
  - Committer
  - Hacker, Blogger, CF manager
- Website: https://paquier.xyz

Agenda

- Architecture
- Postgres 18
- Extensions

Architecture

Monitoring

- Aggregated view of cluster state
  - Objects: table, index, function
  - Processes: checkpointer, WAL receiver/sender, bgwriter
  - OS activity: WAL, I/O
  - Misc: Transaction, SLRU
- System views pg_stat_* (mostly, not pg_stat_activity!)
- https://www.postgresql.org/docs/devel/monitoring-stats.html

Up to Postgres 14

- Process: stats collector
- UDP for communication – pgstat_send() and stats socket
- Stats on disk – Point of consistency
  - Written periodically, couple of times per second.
  - Loaded from files by each process
  - One global file, plus one file per database
- Performance bottleneck
  - Addition of new tenants
  - Multi-tenant, many files

Postgres 15 – Shared memory

- No stats collector.
- No periodic writes.
- Single on-disk file.
- Each stats kind is assigned an ID
- Commit 5891c7a8a.

Durability

- Single file
- Loaded by startup process
- Written/synced by last process at shutdown:
  - Checkpointer
  - Postmaster (single-user mode)
- Lost on crash
  - Autovacuum and table problem.
  - Relation stats in relfilenodes – recovery (Postgres 19)

Source code

- Code parts
  - pgstat.c
  - pgstat_shmem.c
  - pgstat_internal.h and pgstat.h
- One file for each stats kind:
  - pgstat_database.c
  - pgstat_archiver.c, etc.
- src/backend/utils/activity/

Fixed-sized stats

- Chunk of shared memory
- Memory allocated at startup
- LWLocks included in each stats structure
  - pgstat_internal.h
  - PgStatShared_*
- Counters cover a known size
  - WAL
  - I/O
  - SLRU

Variable-sized stats - Details

- Hash table in dynamic shared memory (dshash.c, DSA)
- Hash key:
  - Stats kind (4 bytes)
  - Database OID (4 bytes)
  - 8-byte ID (4-byte OID in v15~17)
- Objects: database, table, functions, DDL-related.
- Locking internal, partition of dshash.
- Maximum number not capped.

Variable-sized stats - Reporting

- Limit shared memory interaction.
- Pending stats
  - Get reference to entry in shmem, report pending activity.
  - Static to each process.
  - Saved in memory context created in Top MemoryContext.
- Flush by pgstat_report_stat()
  - Transaction commit.
  - Hardcoded max interval (60s).
  - Can be forced, waiting on locks.

Snapshots - stats_fetch_consistency

- Behavior of stats in a single transaction
  - “none”, each access fetches data from shmem.
  - “cache”, first entry access cached until end of transaction.
  - “snapshot”, caches all stats of database connected to.
- pg_stat_clear_snapshot() to clean up cache.

Postgres 18

Backend statistics

- Hash key based on proc number.
- Variable-sized.
- pgstat_backend.c
- For WAL and I/O, same fields as pg_stat_wal and pg_stat_io.
- Functions
  - pg_stat_get_backend_wal(pid), single row
  - pg_stat_get_backend_io(pid), multiple rows
  - pg_stat_reset_backend_stats()
  - Join with pg_stat_activity

pg_stat_io

- Multiple rows:
  - Type of operation (IOOp): write, read, syncs..
  - Context (IOContext): VACUUM, bulk-read, bulk-write, init (new in 18!)
  - Backend type: backend, checkpointer, autovacuum..
- Counters
  - In bytes in v18
  - BLCKSZ and number of operations up to v17.
- Data of pg_stat_wal moved to pg_stat_io

pg_stat_database

- Parallel worker activity
- Fields
  - parallel_workers_to_launch, number decided by planner.
  - parallel_workers_launched, number actually launched.
- Tuning of parallel workers

pg_stat_statements - Fields

- Like pg_stat_database, for each query.
  - Number of workers planned.
  - Number actually launched.
  - Available only in EXPLAIN and logs up to v17.
- wal_buffers_full
  - Count when WAL buffers become full
  - Useful for tuning of GUC wal_buffers

pg_stat_statements - Normalization

- IN/ANY clauses
  - SELECT * FROM tab WHERE a IN ($1)
  - ORMs with large number of values
  - JDBC
- Relation name, not OID in query jumbling
  - Temp tables with same name: different query ID
  - Same table name + different schema: same query ID
  - Use alias in FROM clause for different query ID:
    SELECT * FROM s1.tab AS s1tab;
    SELECT * FROM s2.tab AS s2tab;

Extensions

Custom Cumulative Statistics

- Plugins and extensions can use pgstats!
- Similar to custom WAL RMGRs
  - Load with shared_preload_libraries
  - When removed from shared_preload_libraries, same as corrupted.
- Assigned fixed ID
- https://www.postgresql.org/docs/devel/xfunc-c.html#XFUNC-ADDIN-CUSTOM-CUMULATIVE-STATISTICS

Stats kind ID

- Whole range
  - PGSTAT_KIND_MIN 1
  - PGSTAT_KIND_MAX 32
- Invalid = 0
- Built-in: 1 to 12, up to 23 available
- Custom:
  - PGSTAT_KIND_CUSTOM_MIN = 24
  - PGSTAT_KIND_CUSTOM_MAX = PGSTAT_KIND_MAX
- pgstat_kind.h

Source code - Register

- PGSTAT_KIND_EXPERIMENTAL for development.
- ID overlap => stats file corruption.
- Reserve and ID: https://wiki.postgresql.org/wiki/CustomCumulativeStats
- See pgstat_register_kind().

Callbacks - Basics

- PgStat_KindInfo in pgstat_internal.h
- Some boolean fields:
  - fixed_amount: variable or fixed size?
  - accessed_across_databases
  - write_to_file
- Some functions
  - flush_pending_cb => Variable-sized
  - flush_static_cb => Fixed-sized
  - init_backend_cb => Initialization action for backends

Callbacks – Names and Object ID mapping

- For on-disk storage
  - Define custom object IDs when reading from disk, based on names.
  - Translate object IDs to names, when writing.
- to_serialized_name
- from_serialized_name
- Example: replication slot stats

Source code - Templates

- Statistics for injection points
- Hash key:
  - Invalid database OID
  - Hash of injection point name.
- src/test/modules/injection_points/
  - injection_stats.c for variable-sized stats
  - injection_stats_fixed.c for fixed-sized stats
- GUC to control if stats are enabled.

pg_stat_statements?

- Query text file
  - Performance bottleneck
  - Move to DSA
- Deallocation and pg_stat_statements.max
  - Track number of entries in hash table of pgstats.
  - New callback for deallocation?
- Do NOT move in core. Move to custom pgstats. Try at least. Promise.

Thanks!
Questions?