Extended Vacuum Statistics in PostgreSQL

This patch collects vacuum statistics and ships an extension that persists them, with filters for surgical monitoring control.

Alena Rybakina — lena.ribackina@yandex.ru
Andrei Lepikhov — lepihov@gmail.com
Andrey Zubkov — a.zubkov@postgrespro.ru

Why we collect these statistics

#monitoring #vacuum #statistics #postgres

PURPOSE OF VACUUM

- Clean dead versions of tuples
- Update the visibility map (VM)
- Update the free space map (FSM)
- Freeze tuples, blocks and tables
- Update catalog information

MAIN DIFFICULTIES

- **Long-running transactions**
  hold back the xmin horizon — vacuum can't reclaim recently dead tuples
- **Bloated indexes**
  every vacuum re-reads the entire index
- **Many parameters**
  hard to predict the effect of changing them without statistics

THE (AUTO)VACUUM OBSERVABILITY GAP

pg_stat_all_tables	pg_stat_progress_vacuum	Server log
✓ Counts: vacuum_count, n_dead_tup, last_vacuum	✓ Current phase	✓ Has the numbers
✗ No I/O	✗ In-flight only	✗ Unstructured text
✗ No WAL	✗ No history	✗ Memory consuming

What statistics we add

COMMON — shared by all three views

- **WAL:** wal_records · wal_fpi · wal_bytes
- **BUFFER I/O:** blks_fetched · blks_hit · blks_read · blks_dirtied · blks_written
- **TIME:** total_time · delay_time
- **COUNTERS:** interrupts_count · wraparound_failsafe_count

pg_stats_vacuum_database

ALL COMMON FIELD (TABLES + INDEXES)

vacuum_count · per-DB failsafe interrupts · delay_time

pg_stats_vacuum_tables

ALL COMMON FIELDS

- **PAGES:** pages_scanned · pages_removed
- **TUPLES:** tuples_deleted · tuples_frozen · recently_dead · missed_dead_tuples
- **VISIBILITY MAP:** vm_new_visible · vm_new_frozen

pg_stats_vacuum_indexes

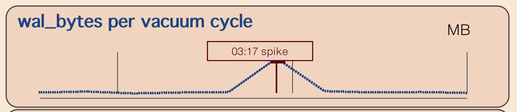
ALL COMMON FIELDS

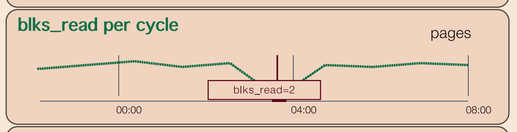
tuples_deleted · pages_deleted

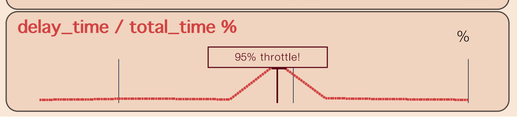
SCAN · PATCH #5012

How it lands in core

1. **Scan table** — find dead row versions
   - start heap stats
2. **Vacuum indexes** — remove index references pointing to dead TIDs
   - start idx stats
   - end idx stats & diff stats
   - report idx stats
3. **Clean up index** — recycle empty pages, updates index stats
   - start idx stats
   - end idx stats & diff stats
   - report idx stats
4. **Clean up heap tuples** — free dead line pointers, compact pages, update FSM/VM
   - end heap stats
   - exclude idx stats
   - report heap stats







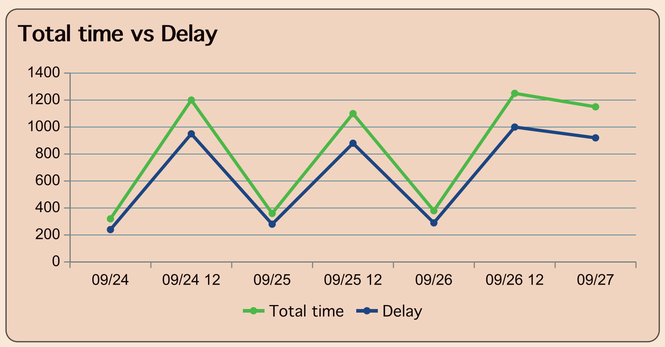
DBA controls — filters & anomalies hunting

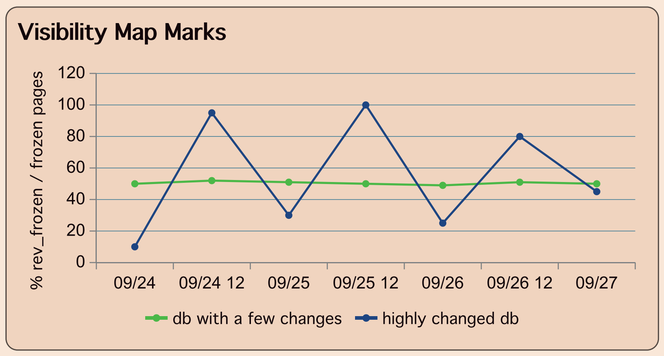
track_relations = all, system, user
track_stat_types: wal, buffers, timing, general, all
relation_whitelist = orders, events
database_whitelist = mydb, analytics

DBA · read & act

«I see a WAL spike on orders at 03:17 — and zero blocks read. Probably maintenance_work_mem too small. Let's check.»

Tracking dynamics & anomalies are possible now...





load extension

1. **Load extension**
   Add custom vacuum statistics EXT_VAC_DB and EXT_VAC_REL

set the filters

1. **FILTERS**

check if stats should be saved

#	KIND	DATABASE	OBJECT	COUNTERS (CUMULATIVE)	
1	RELATION	mydb	orders	seq_scan: 42	tup_ins: 15203
2	DATABASE	mydb	calc_total	calls: 5840	total_time: 12.4
3	FUNCTION	mydb	—	xact_commit: 48201	blks_read: 9104
4	EXT_VAC_REL	mydb	orders	blks_read: 340	pages_scanned: 128
5	EXT_VAC_REL	mydb	orders_pke	blks_read: 52	pages_deleted: 3
6	EXT_VAC_DB	mydb	—	blks_read: 392	wal_bytes: 81920
...	...	...	...	...	

3. **Hook calls the function that writes to hash table**
   The hook receives the diff, applies the filters from ①, then takes the per-entry LWLock and adds counters into the dshash

2. **Cleans heap & indexes, calls `report()` via hook**
   Each phase ends with a snapshot diff of statistics or new accumulated counters

get the statistics through views in extension

5. **SQL views**
   pg_stats_vacuum_tables
   pg_stats_vacuum_indexes
   pg_stats_vacuum_database

4. **Flush to disk**
   Checkpointer & shutdown serialize the dshash atomically.