PGCONF.DEV 2026 • FRIENDLY ARCHITECTURAL RESPONSE TRACK

Hey Andy Pavlo!
We Fixed the Part of PostgreSQL You Hate the Most.

Presenter: **Greg Burd** (greg@burd.me) | Solution Codebase Branch: **`tepid`** **CommitFest cf-5556**

THE CMU / PAVLO CRITIQUE

In his famous ecosystem roast, *"The Part of PostgreSQL We Hate the Most,"* Andy Pavlo targeted PostgreSQL's append-only MVCC engine layout. When an update occurs, the engine writes a brand new tuple version. If an update breaks the binary rules of classic HOT, it creates a massive write explosion.

> *"If you have a table with 64 indexes and you execute an UPDATE that changes just one single indexed column, classic HOT completely breaks. Postgres falls off a binary cliff and forcefully duplicates pointer entries into ALL 64 INDEXES!" — The Pavlo Manifesto*

The Resulting Write Amplification

Competing engines (like MySQL's InnoDB clustered configuration) avoid this penalty because secondary indexes point to a logical primary key descriptor. Classic Postgres optimization (2007) was strictly all-or-nothing: if even a single indexed attribute mutated, index write costs scaled linearly with the total index density of the schema layout.

THE SOLUTION: SELECTIVE HOT UPDATES

The `tepid` codebase eliminates this architectural penalty by inserting fine-grained conditional logic directly into the heap access layer. The modification sequence stays completely localized within the same storage block page, but pointer allocations are managed selectively:

- **Surgical Key Delta Writes:** Only secondary indexes tracking columns that were explicitly altered during the update receive a fresh leaf item tuple pointer.
- **Untouched Index Exemption:** Indexes matching unmodified columns are skipped entirely. They retain their current leaf addresses, evaluating the latest tuple state by resolving a fast in-page tombstone hop chain.

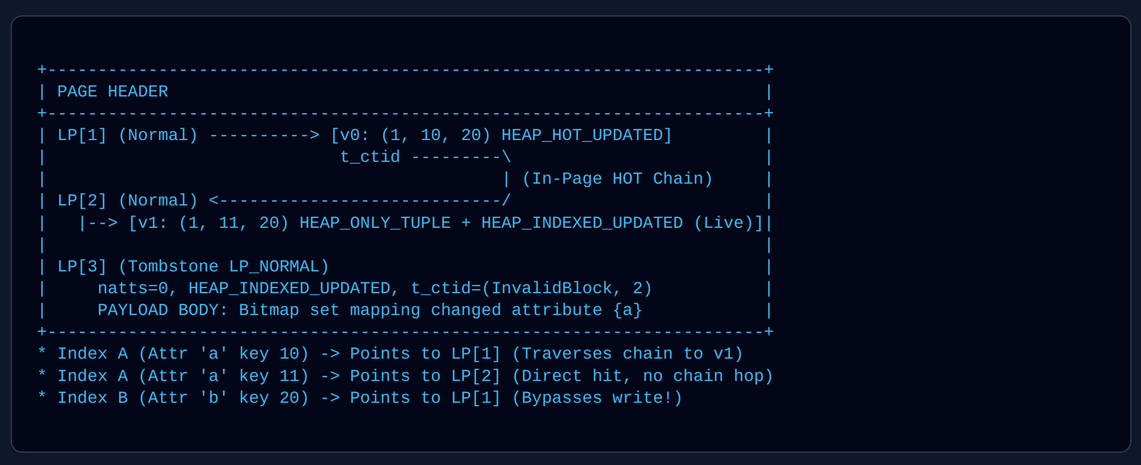
Performance Receipt Under Stress Churn

On a wide-table workload of random UPDATEs to one of 64 indexed columns (Linux RISC-V, 4 runs * 110 s/cell, 4 pgbench clients, autovacuum on), single-column updates (`wide_1`) gain **+74.3% TPS** while WAL drops **70.3%**; the `wide_2..wide_8` band averages **+68% TPS** / **-64% WAL**.

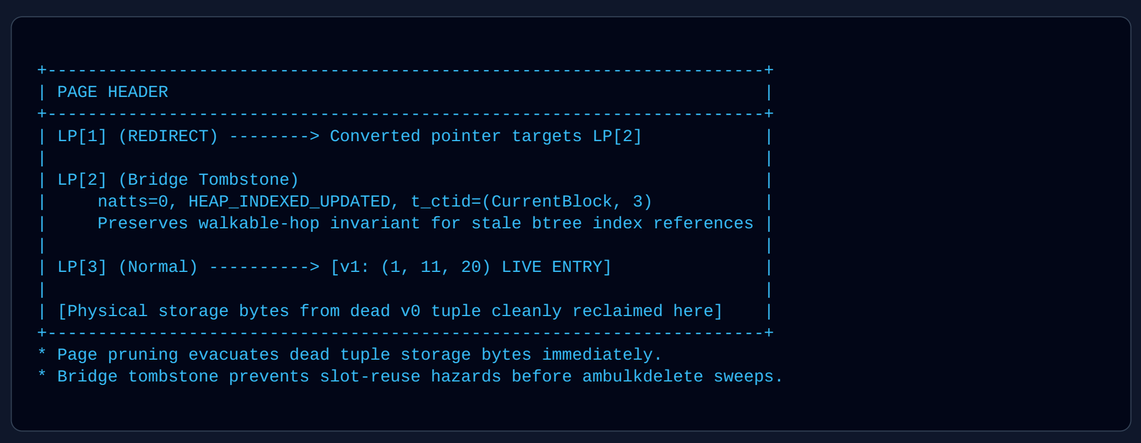
ON-PAGE STORAGE LAYOUT (ASCII ARCHITECTURE)

Exact structural visualization of physical PostgreSQL page frames showing line pointer redirections, bitmap-guided tombstones, and tuple chains.

STATE A: PAGE CONFIGURATION IMMEDIATELY POST-UPDATE



STATE B: LAYOUT POST IN-PAGE PRUNING OPERATIONS



OPERATIONAL CONTROLS & PRODUCTION SIZING GUIDANCE

HOT-Indexed Updates introducing high-precision database operator reference tunables to maximize break-even efficiency points out of the box. Most production OLTP environments require zero explicit modifications from their upstream baselines.

- **GUC Parameter: `hot_indexed_update_threshold` (Integer %, Default: `80`)** — Controls the maximum threshold percentage calculated via `(modified_idx_attrs / all_idx_attrs) * 100` at which heap_update executes the fine-grained selective path. If key churn crosses this value, it transparently triggers a plain fallback path to prevent edge-case structural regressions.
- **Adaptive Guardrails:** Classic HOT execution paths (zero indexed modifications) remain entirely unaffected by GUC parameters and always fire with absolute priority.
- **Observability Integrations:** Exposed explicitly via standard `EXPLAIN (SETTINGS)` through internal `GUC_EXPLAIN` tracking flags. Production DBAs can monitor real-time selective hit ratios seamlessly via catalog statistics expansion.

ECOSYSTEM MATRIX: PRIOR WRITE-AMPLIFICATION PROPOSALS VS. HOT-INDEXED UPDATES (TEPID)

Architectural Metric	Pre-HOT (1996)	Classic HOT (2007)	WARM (2017)	PHOT (2021)	HOT-Indexed (`tepid`)
Relaxes I1 ("no indexed change")	No	No	Yes	Yes	**Yes (Fine-grained selective isolation)**
Preserves I2 (entries at chain root)	No	Yes	Yes	No	**No (Allows multi-index mid-chain targeting)**
Changed-Index Btree Targets	New Tuple TID	n/a	Chain Root LP	Mid-chain TID	**Mid-chain Heap-Only Tuple TID**
Mid-chain LP Reclaim at Prune	n/a	Yes (Classic)	Yes (Classic)	Redirect on LP	**Bridge Tombstone (deferred to vacuum)**
Modified-Attrs Bitmap	n/a	n/a	Not Materialized	Computed at prune	**Explicitly materialized at write time**
Updates per Chain Restriction	Unbounded	Unbounded	Capped at 1	Unbounded	**Unbounded benefit optimization paths**
Per-AM Recheck Obligation	No	No	Required in all AMs	Unspecified	**Optional callback with fallback drop paths**
Page-level Infrastructure Bit	None	None	None	None	**`PD_HAS_HOT_INDEXED_BRIDGES`**

Ecosystem Collaboration & Project Status

Regress Integrity: 247/247 Core • 41/41 Isolation Passes Clean.
Recovery Suite passes identically with `wal_consistency_checking=all`.
Contact Presenter: **Greg Burd** <greg@burd.me>

Git Repo: https://github.com/gburd/postgres/tree/tepid
Wiki: https://wiki.postgresql.org/wiki/Heap_HOT_Selective_Index_Updates