Hardware Acceleration & SIMD Optimizations in Postgres

Akash Shankaran

Intel

Past contributions

- Accelerate popcount using AVX-512 (PG17 - 2024)
  - Improves *bit_count* performance in Postgres
  - Resulted in PGVector Binary vector speed-up ~25-40%

- CRC32C using AVX-512 instructions where available (PG18 – 2025)

Current work

- Auto-vectorizing checksum calculation.
- Lock-Free Xlog reservation from WAL.
  - Improve TPCC performance on high core count systems.
- SORT – using x86-simd-sort library
  - TupleSort – improves ORDER BY queries
  - ListSort results in tangible improvements to PGVector

SORT proposal

- What do we want to do:
  - Introduce **x86-simd-sort** library to use for sort operations.
  - Library to be added as an optional compile-time dependency

  (e.g. `--with-x86-simd-sort`)
  - Extend existing templates and macros, to support simd based sort.
  - Observed improvements in scenarios using
    - tuple sort (ORDER BY)
    - list sort (PGVector)

X86-simd-sort library

- Supported Sorts: Quicksort, partial sort (top k), index sort for numeric types (16, 32, 64 bit signed/unsigned)
- Open source BSD3 license
- Builds using Meson
- Used in Numpy, PyTorch, JDK.

For Postgres, we plan to use quick sort for key-value pairs of numbers (32, 64bit integers/floats)

Results

- Tuple sort
  ORDER BY queries with randomly-generated data, with varying row count and cardinality (controlled by number of distinct values), single-column filter.

# of rows	# of distinct values	% gain or loss
100k	100k	+13.4
100k	10k	+9.8
100k	100	+3.0
100k	10	-1.9
1M	1M	+6.5
1M	10k	+1.6
1M	100	-2.9
1M	10	-5.1

Collected on AWS m7i.metal-24xl, Ubuntu 24.04, gcc13

Results

- List sort

Observed:

1. 7% reduction in index build time (HNSW) with the gist-960 dataset
2. 10% reduction with the dbpedia-openai-1000k dataset

(ANN-Benchmarks, HNSW index with halfvec, m=80)

Feedback

- Feasibility of the approach?
- What is the most common scenario for postgres sort, dataset size and cardinality for validating simd-sort?
- When to call simd-sort? query execution / query planning?
  - If planning, then what query statistics can be used (e.g. cardinality count)?