SDC 20

BY Developers FOR Developers

Storage Developer Conference
September 22-23, 2020

Scaling PostgreSQL with Persistent Memory

Naresh Kumar Inna and
Keshav Prasad

Memhive

Agenda

- Databases and PMEM
- PostgreSQL storage architecture
- Scaling PostgreSQL with Memhive and PMEM
- Benchmarks
- Conclusions

Databases and PMEM

- Databases are considered as one of the top use cases of PMEM - scaling capacity and performance

Multiple ways of using PMEM:

- Storing DB Logs including redo log, Write Ahead Log (WAL), etc - the most common use case (Eg:Redis AOF, Oracle)
- DB cache store (instead of storing in DRAM or as a cache tier)
- Relational data store (large "in-memory" store)

Databases and PMEM (contd..)

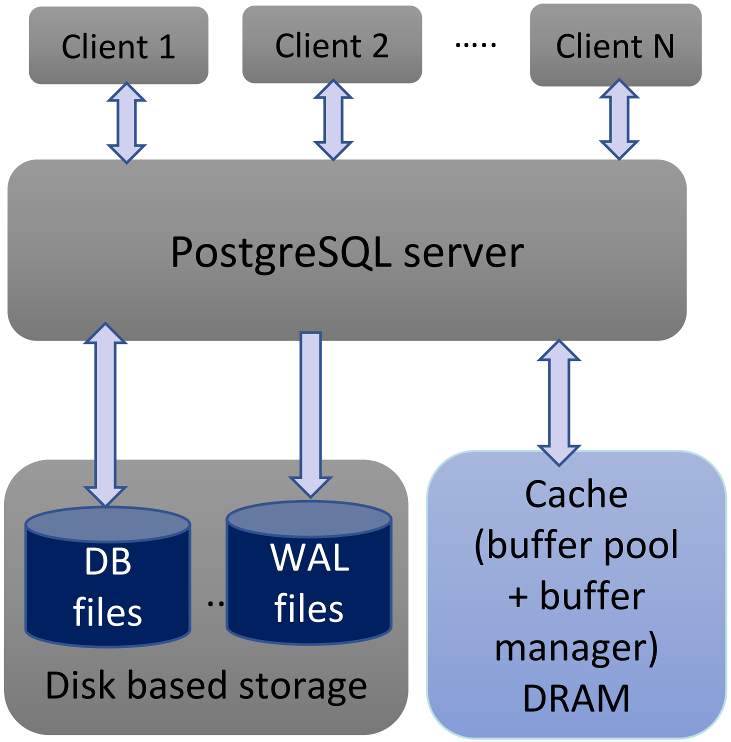
Conflicting modes of PMEM usage:

- Memory mode (transparent, but inefficient) cache
- AppDirect (complex but highly efficient)

PostgreSQL storage architecture

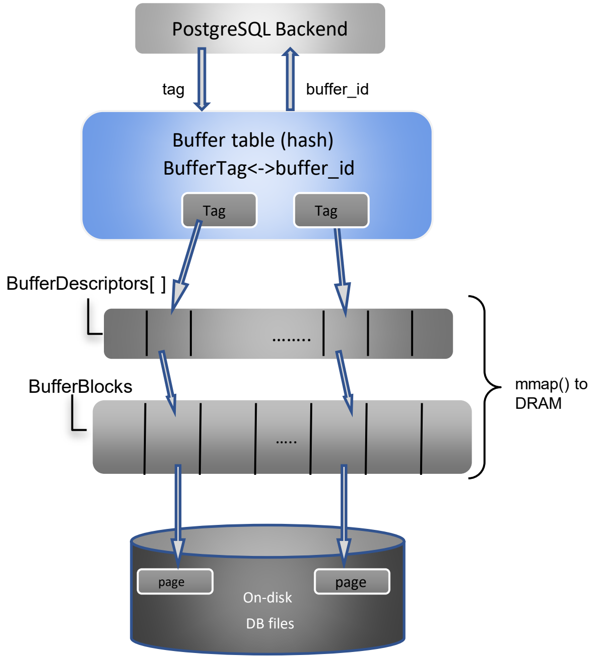
Traditional PostgreSQL

- PostgreSQL storage architecture
  - Cache on shared DRAM memory via `mmap(2)`
  - WAL and relation data laid out as directories and files (index, table) on a disk-based file system.



PostgreSQL cache layer

- Cache a.k.a shared buffer cache layer.
- Three layer buffer manager:
  - Buffer table (map buffer tag to buf ID)
  - Buffer descriptors (metadata)
  - Buffer blocks (data buffers) – 8KB
- Each 8KB buffer directly holds the page data of the on-disk table file it points to at the offset.



Scaling PostgreSQL with PMEM

Design considerations with PMEM

- AppDirect *fsdax* choices PostgreSQL:
  - `libpmemobj`
  - `libpmem`
- `libpmemobj` challenges with PostgreSQL:
  - No pluggable storage engine like MySQL or MariaDB.
  - Introducing TX_xxx() API required re-designing core storage paths.
- `libpmem` :
  - Inline changes to existing storage paths, no design changes.

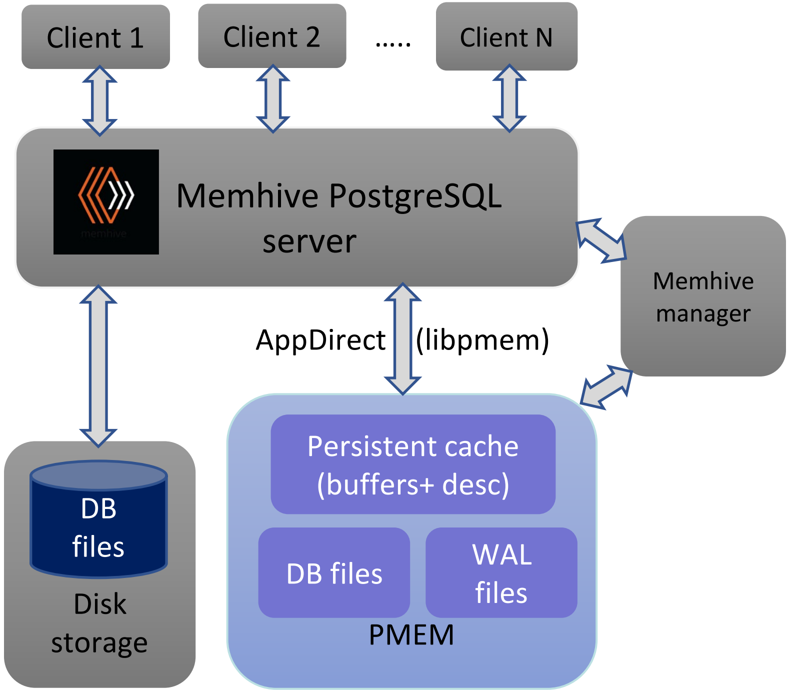
Additional design considerations

- `libpmem` provides no redundancy to protect against local DIMM failure, à la `libpmemobj` poolsets. *fsdax* has no LVM mirror support.
  - Critical for both WAL and DB relation files.

- NUMA effects: more pronounced with PMEM.

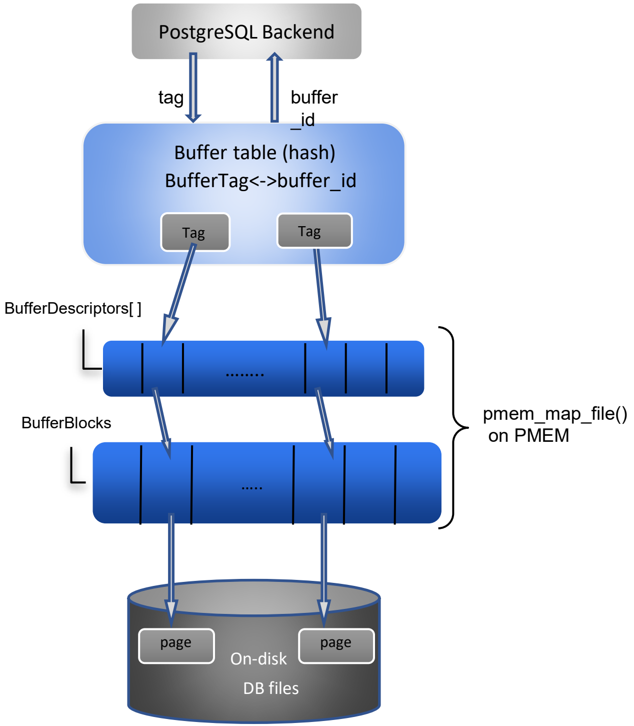
Memhive PostgreSQL

- PMEM based persistent cache
- WAL files on PMEM
- DB relation files on PMEM
- Manager



Persistent Cache

- PMEM based non-volatile cache.
- Buffer descriptors and buffers mapped to an *fsdax* namespace on PMEM.
- CPU cache flushes and batched drains at critical points of the cache manager. Uses both variants `pmem_memcpy_nodrain()` and `pmem_flush()`.



Persistent Cache (contd..)

- Minimal freelist updates during PostgreSQL server startup.
- Dual mode:
  - Always persistent: CPU cache flush/drain for buffer contents and selected descriptor fields. Persistence for both planned and unplanned server restarts.
  - Selective persistence: No flush/drain after buffer/meta updates to avoid penalty (albeit minimal). Persistence for planned server restarts only.
- Optimized for persisting meaningful buffers only:
  - Avoid flushes/drains on short lived cache buffers (eg: VACUUM, COPY IN)

WAL and relational data on PMEM

- WAL on PMEM:
  - Performance mode: *fsdax* type namespace, writes in the Xlog flush path replaced by `pmem_memcpy_xxx()` calls
  - Local (DIMM) redundancy mode: LVM mirror on *sector* type namespaces.
- Relational data files (indexes, tables) on *sector* type PMEM when DB size <= PMEM size, cache on DRAM.
- PostgreSQL replication for redundancy with both *sector* and *fsdax* types.

Possible configurations

- Persistent cache + WAL on PMEM:
  - Local redundancy: LVM mirror on *sector* (WAL)+ *fsdax* (cache) namespaces, non-interleaved DIMMs.
  - performance: *fsdax* (WAL + cache), interleaved DIMMs
  - Relational data files on existing DAS/SAN storage.
- Relational data files + WAL on PMEM:
  - Local redundancy: LVM mirror on *sector* namespace (WAL + relational data), non-interleaved DIMMs.
  - performance: LVM on *sector* (relational data) + *fsdax* (WAL).
  - Cache on DRAM

PostgreSQL file layout

Standard

```
[postgres@localhost ~]$ ls -l /usr/local/pgsql/data/
total 124
drwx------. 7 postgres postgres  4096 Jul  2 15:14 base
drwx------. 2 postgres postgres  4096 Sep  4 15:46 global
drwx------. 2 postgres postgres  4096 Jul  2 14:48 pg_commit_ts
drwx------. 2 postgres postgres  4096 Jul  2 14:48 pg_dynshmem
-rw-------. 1 postgres postgres  4513 Jul  2 14:48 pg_hba.conf
-rw-------. 1 postgres postgres  1636 Jul  2 14:48 pg_ident.conf
drwx------. 4 postgres postgres  4096 Sep  4 15:46 pg_logical
drwx------. 4 postgres postgres  4096 Jul  2 14:48 pg_multixact
drwx------. 2 postgres postgres  4096 Sep  4 15:46 pg_notify
drwx------. 2 postgres postgres  4096 Jul  2 14:48 pg_replslot
drwx------. 2 postgres postgres  4096 Jul  2 14:48 pg_serial
drwx------. 2 postgres postgres  4096 Jul  2 14:48 pg_snapshots
drwx------. 2 postgres postgres  4096 Sep  4 15:46 pg_stat
drwx------. 2 postgres postgres  4096 Sep  4 15:46 pg_stat_tmp
drwx------. 2 postgres postgres  4096 Jul  2 14:48 pg_subtrans
drwx------. 2 postgres postgres  4096 Jul  2 14:48 pg_tblspc
drwx------. 2 postgres postgres  4096 Jul  2 14:48 pg_twophase
-rw-------. 1 postgres postgres     3 Jul 10 18:13 PG_VERSION
drwx------. 3 postgres postgres  4096 Jul 16 23:14 pg_wal
drwx------. 2 postgres postgres  4096 Jul  2 14:48 pg_xact
-rw-------. 1 postgres postgres    88 Jul  2 14:48 postgresql.auto.conf
-rw-------. 1 postgres postgres 26804 Sep  4 15:45 postgresql.conf
-rw-------. 1 postgres postgres    59 Sep  4 15:46 postmaster.opts
-rw-------. 1 postgres postgres    89 Sep  4 15:46 postmaster.pid
[postgres@localhost ~]$
```

Memhive with PMEM

```
[postgres@sdp data]$ mount | grep region0
/dev/pmem0 on /opt/pmem/region0 type ext4 (rw,relatime,seclabel,dax)
[postgres@sdp data]$ ls -l
total 128
drwx------. 7 postgres postgres  4096 Sep  3 05:25 base
-rw-------. 1 postgres postgres    30 Sep  4 00:00 current_logfiles
drwx------. 2 postgres postgres  4096 Sep  3 04:47 global
drwx------. 2 postgres postgres  4096 Sep  4 02:52 log
-rw-------. 1 postgres postgres     0 Sep  3 04:46 pcache
drwx------. 2 postgres postgres  4096 Sep  3 04:46 pg_commit_ts
drwx------. 2 postgres postgres  4096 Sep  3 04:46 pg_dynshmem
-rw-------. 1 postgres postgres  4269 Sep  3 04:46 pg_hba.conf
-rw-------. 1 postgres postgres  1636 Sep  3 04:46 pg_ident.conf
drwx------. 4 postgres postgres  4096 Sep  3 09:15 pg_logical
drwx------. 4 postgres postgres  4096 Sep  3 04:46 pg_multixact
drwx------. 2 postgres postgres  4096 Sep  3 04:46 pg_notify
lrwxrwxrwx. 1 postgres postgres    27 Sep  3 04:46 pg_pcache -> /opt/pmem/region0/pcachedir
drwx------. 2 postgres postgres  4096 Sep  3 04:46 pg_replslot
drwx------. 2 postgres postgres  4096 Sep  3 04:46 pg_serial
drwx------. 2 postgres postgres  4096 Sep  3 04:46 pg_snapshots
drwx------. 2 postgres postgres  4096 Sep  3 04:46 pg_stat
drwx------. 2 postgres postgres  4096 Sep  4 03:17 pg_stat_tmp
drwx------. 2 postgres postgres  4096 Sep  3 09:13 pg_subtrans
drwx------. 2 postgres postgres  4096 Sep  3 04:46 pg_tblspc
drwx------. 2 postgres postgres  4096 Sep  3 04:46 pg_twophase
-rw-------. 1 postgres postgres     3 Sep  3 04:46 PG_VERSION
lrwxrwxrwx. 1 postgres postgres    21 Sep  3 04:46 pg_wal -> /opt/pmem/region0/wal
drwx------. 2 postgres postgres  4096 Sep  3 09:08 pg_xact
-rw-------. 1 postgres postgres    88 Sep  3 04:46 postgresql.auto.conf
-rw-------. 1 postgres postgres 26678 Sep  3 04:46 postgresql.conf
-rw-------. 1 postgres postgres    63 Sep  3 04:46 postmaster.opts
-rw-------. 1 postgres postgres   109 Sep  3 04:46 postmaster.pid
```

The story in numbers

Strategic partnership with Intel® Optane™

- PMEM options: NVDIMM, Intel® Optane™.
- Optane™ PMEM is ideal for vertically scaling PostgreSQL due to the price/capacity advantage.
- All benchmarking tests performed on Intel's SDP cloud server with Optane.

Test environment

Hardware	
CPU	Intel Cascade Lake Xeon processor 24 cores x 2 (2 threads per core)
DRAM	16 GB x 12
PMEM	128 GB Optane x 12
SSD	800 GB SATA SSD, 480GB SATA SSD x 2
Software	
OS	Fedora Core-31 Linux 5.5.8-200
PMDK	1.7
Standard Postgres	PostgreSQL v12
Memhive	v1.0
File system	ext4

Test environment (contd..)

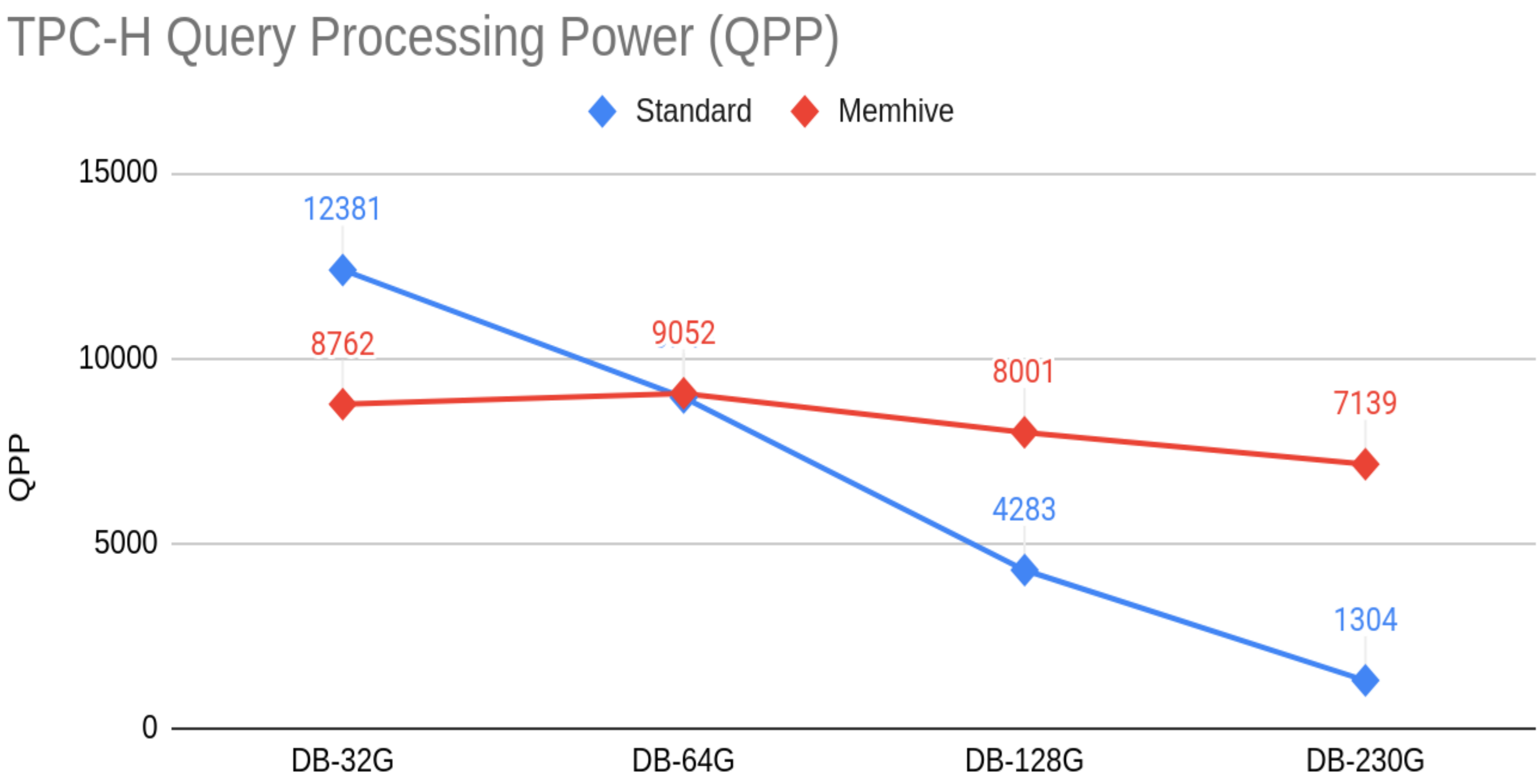
Benchmarks	
DBT-3 (TPC-H)	**Test parameters:** Database sizes: 32, 64, 128 and 230 GB Streams: 1, 5, 10 and 15
pgbench (TPC-B like)	**Test parameters:** Scaling factor: 24000, 350 GB database Clients: 5, 10, 20 and 40 Jobs: 5 Time: 20 minutes

- All tests bound to one socket with `numactl(8)`
  - 128 GB Optane PMEM x 6 (interleaved)
  - Intel Xeon processor 24 cores x 1
  - 16 GB RAM x 6

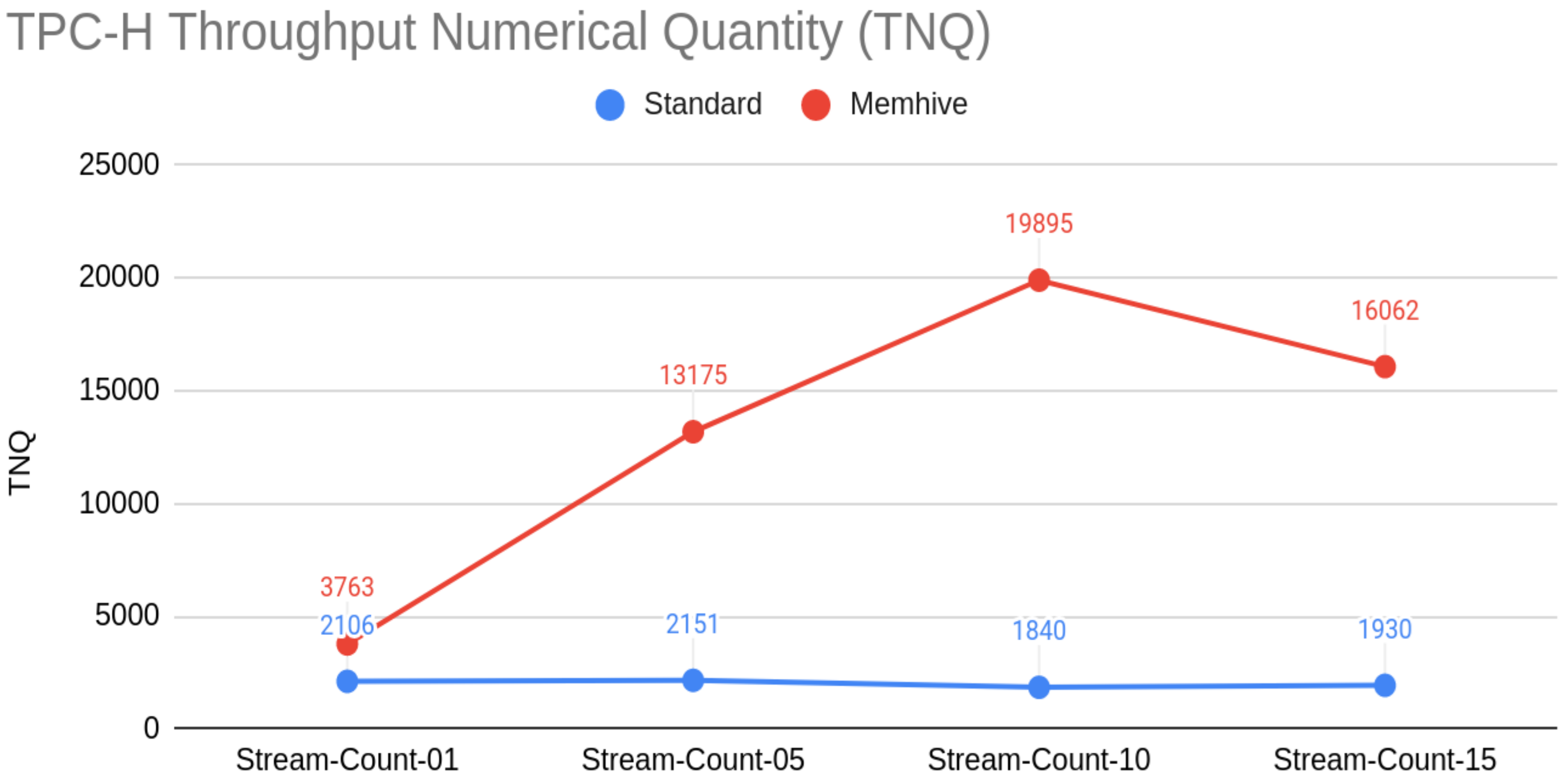
PostgreSQL config comparison

	Standard PostgreSQL v12	Memhive PostgreSQL
Optane Persistent Cache	N/A	400 GB
DRAM	90 GB	90 GB
WAL	On SSD	On Optane PMEM
Relation Data	On SSD	On SSD
Shared Buffers	On DRAM	On Optane PMEM

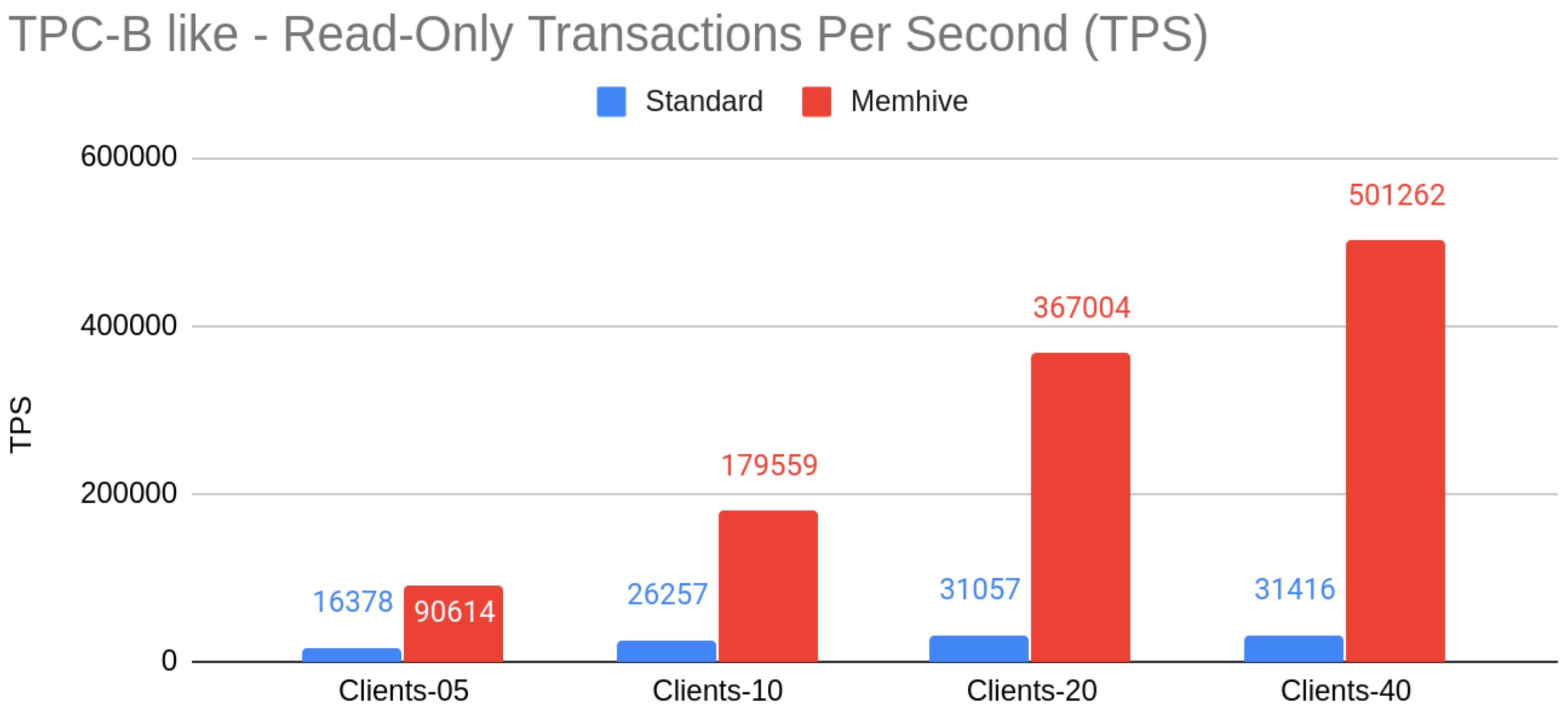
Benchmark results: OLAP - TPC-H DBT-3



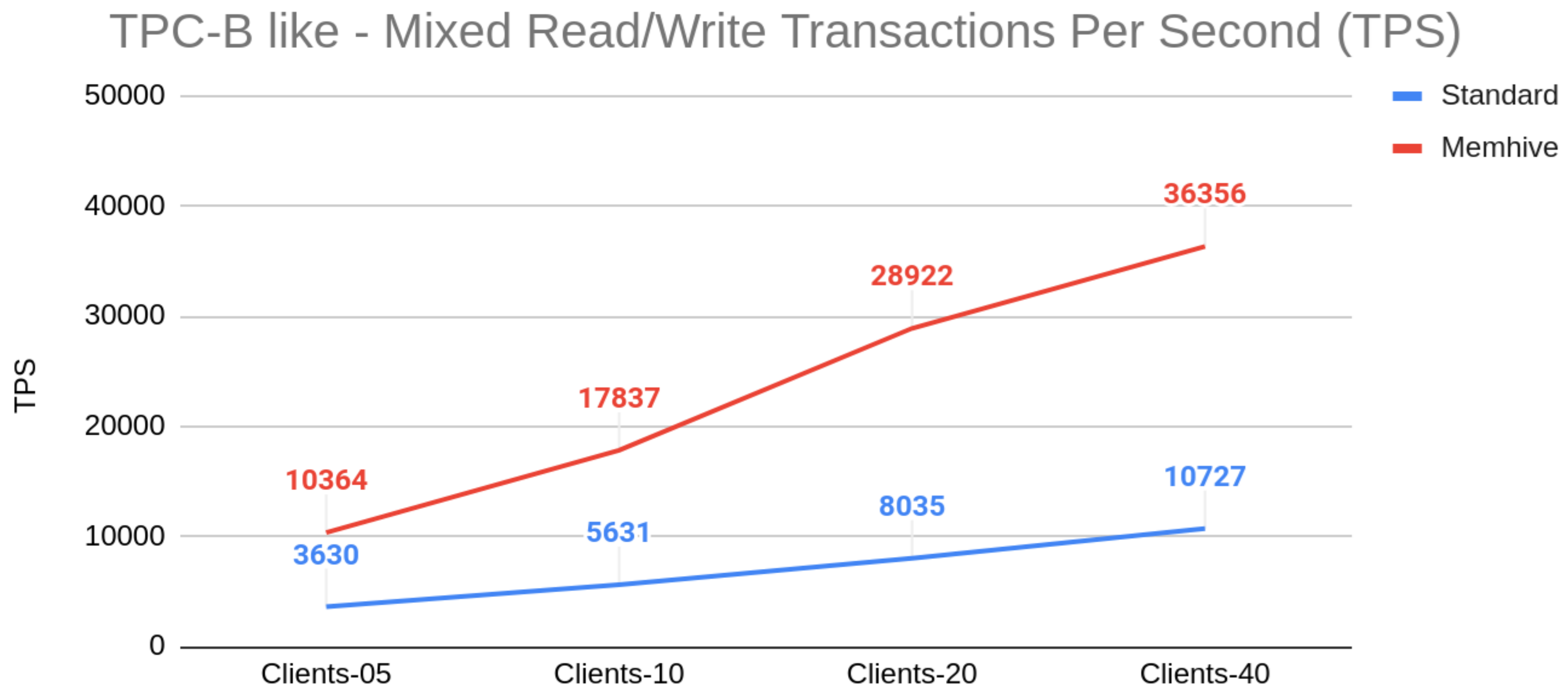
Benchmark results: OLAP - TPC-H DBT-3



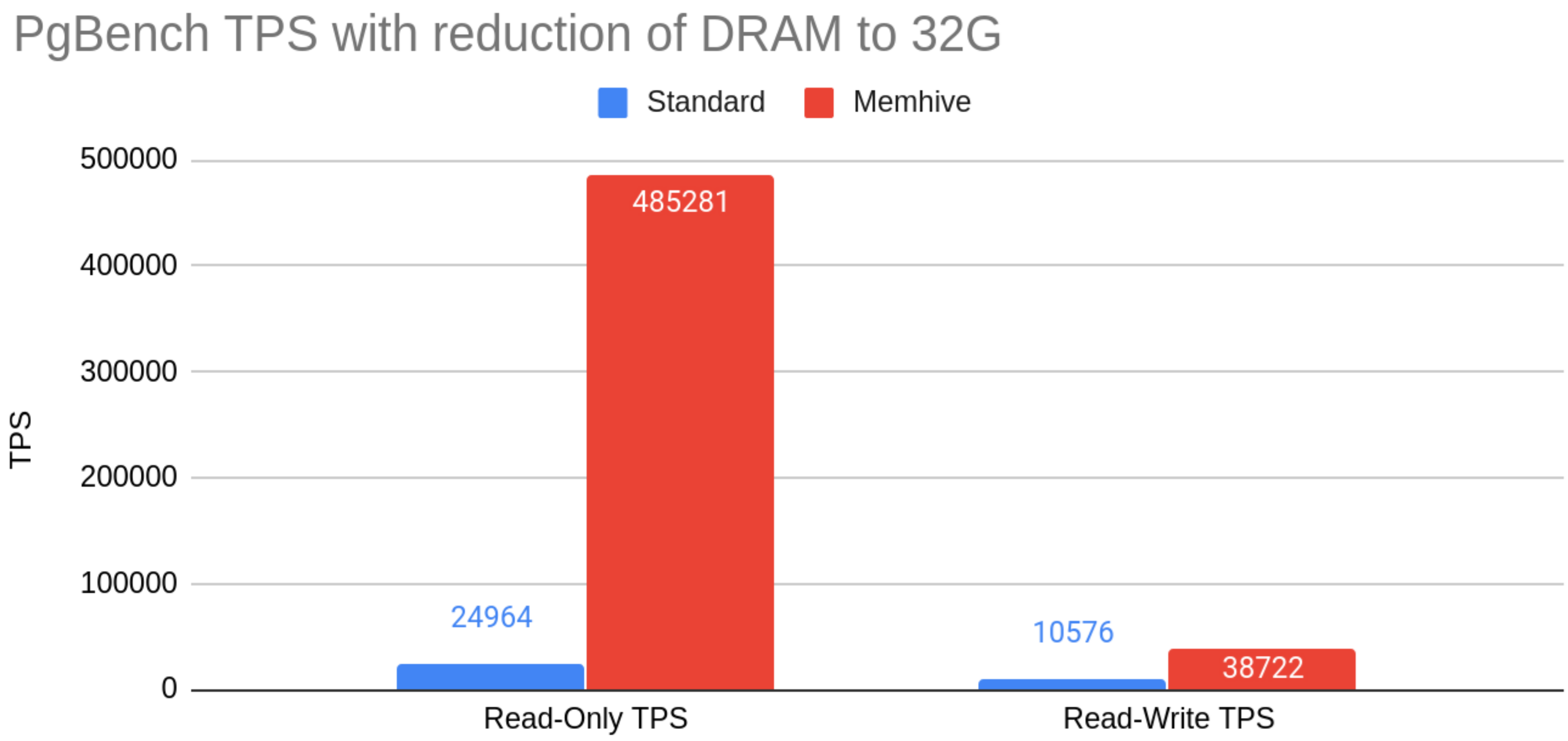
Benchmark results: OLTP - TPC-B like - Pgbench



Benchmark results: OLTP - TPC-B like - Pgbench



Benchmark results: Reduced RAM to 32G



Performance summary

- **Upto 10x** throughput in OLAP DBT-3 TPC-H workload
- **Upto 5x** query processing power in OLAP DBT-3 TPC-H workload
- **Upto 15x** Read transactions per second in OLTP TPC-B like PgBench
- **Upto 3.5x** Mixed Read/Write transactions per second in OLTP TPC-B like PgBench
- Negligible (2%-3%) impact of flush/drains.

Conclusions:
PostgreSQL storage on PMEM

Conclusions

- **PMEM as a persistent PostgreSQL cache**
  - PostgreSQL cache scales almost linearly with memory, making it ideal to reside on PMEM due to $/GB advantage.
  - Access to a large cache turns PostgreSQL into in-memory DB when DB size <= PMEM, ideal for OLAP.
  - Flushes/drains have minimal impact.
  - Instant startup, constantly warm cache.
  - Dramatic reduction in DRAM requirements for PostgreSQL.
  - No strict need for redundancy. Upon PMEM DIMM failures/bad blocks/unsafe shutdowns, cache is rebuilt from on-disk DB data files.

Conclusions

- **PMEM for PostgreSQL data**
  - Ideal for storing relational objects such as WAL, table and index files.
  - Combination of cache and WAL on PMEM leads to significant OLTP and OLAP performance gains.
- **`libpmem`: Device redundancy versus performance**

  Pure performance/no redundancy: *fsdax* for cache and WAL.

  Performance/recoverable from H/W errors: *fsdax* for cache.

  Local redundancy for critical data: LVM mirror over *sector* for WAL and relational files.

….else, use `libpmemobj`.

Thank you!