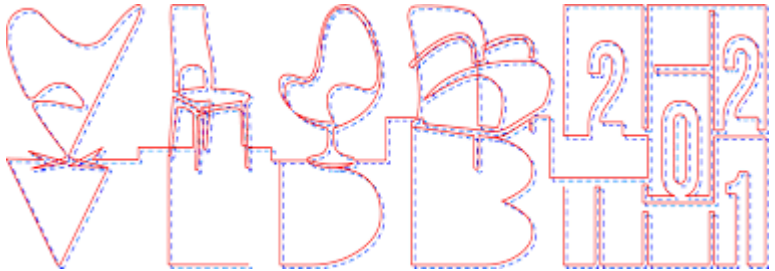
Extending In-Memory OLTP with Persistent Memory

Authors:
Hillel Avni, Aharon Avitzur, Nir Pachter, Vladi Vexler



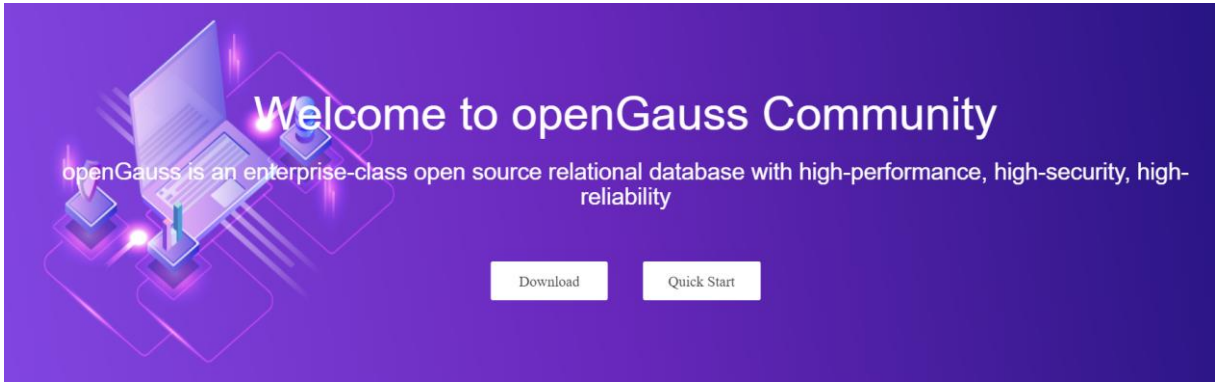
Tel-Aviv Research Center Huawei Technologies Co., Ltd.

PMEM as Drop-In Replacement in a Database

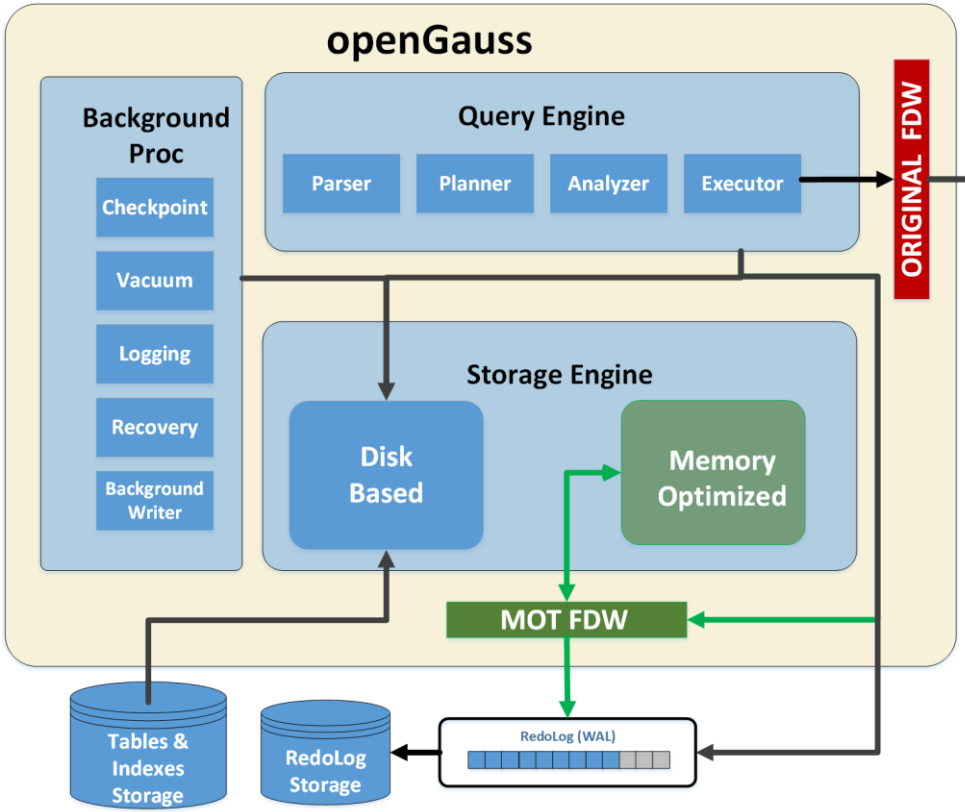
PMEM for MOT in openGauss

- **MOT is an in-memory storage engine in openGauss:**
  - Fully optimistic, lock-free indexes, NUMA aware
  - Strict durability and high availability
  - Weakness: Limited size and expensive DDR main memory
  - Opportunity: Larger and cheaper PMEM main memory

- **openGauss is a Postgres based open-source RDBMS:**
  - https://gitee.com/opengauss and https://github.com/opengauss-mirror
  - https://opengauss.org/



PMEM for MOT in openGauss



MOT FDW Extensions

- Creating Tables
- Creating Indexes
- Planning
- Execution
- Logging
- Checkpointing
- Vacuum
- DROP table/index

PMEM HARDWARE

Present - DIMM

- Plugs into the system's DDR slots.
- Allowed fast development but not scalable for the future.
- This interface is now deprecated due to inherent limitations:
  - Meant for small number of sockets, and not hot-plug.
  - No generic BIOS, implying hard maintenance.

Future - CXL

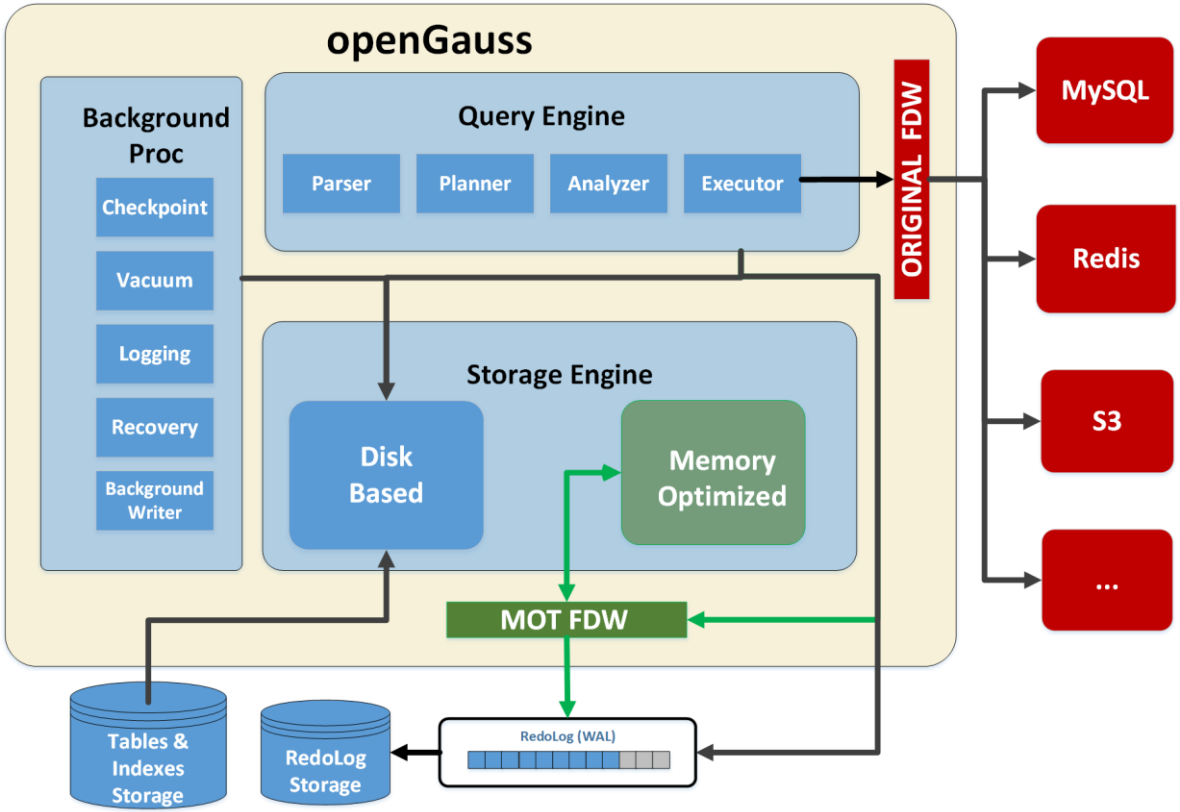
- Evolving standard.
- Vendor neutral so supported by OS driver.
- Based on PCI hot-plug to support unbounded memory size.
- Allows competition to increase performance and reduce price of PMEM.
- Code written for DDR-T DIMMs will work correctly on CXL.

Our Server

- Dual Socket Lenovo server
- CPU: 2 Intel Xeon Gold 6238L @ 2.1GHz
  - 22 Cores per socket
  - With hyper threading (44 virtual cores per socket)
- DRAM: 12 2666 MT/s 32Gb DDR4 (384Gb)
- PMEM: 8 256Gb Intel Optane 100 Series (2Tb) (appDirect mode)
  - PMEM0 is 1TB on Socket0
  - PMEM1 is 1TB on Socket1
- SSD: ThinkSystem 2.5" PM1645a 3.2TB Mainstream SAS 12Gb Hot Swap

PMEM vs. SSD for Logging in GaussDB

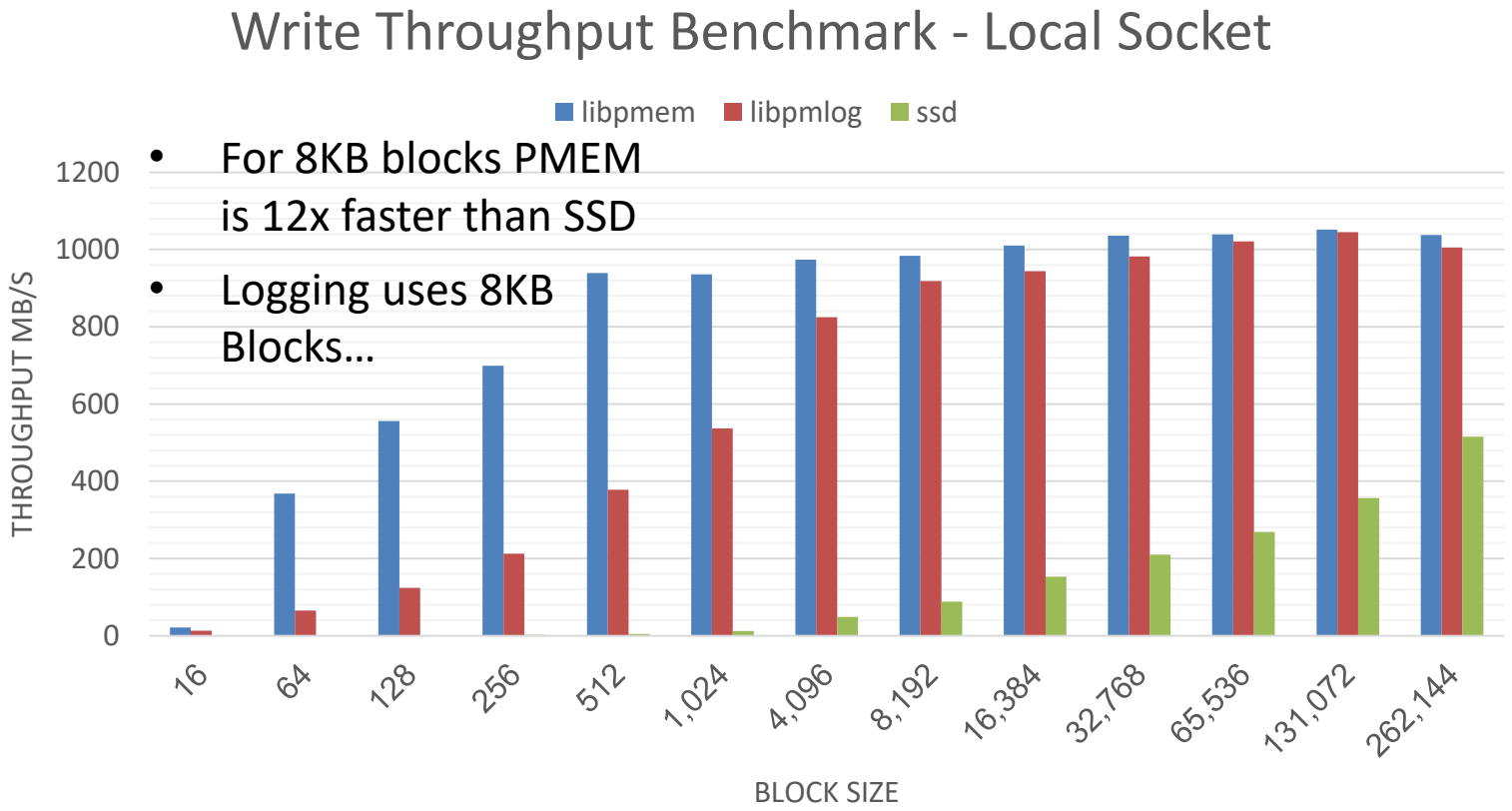
Logging in PMEM



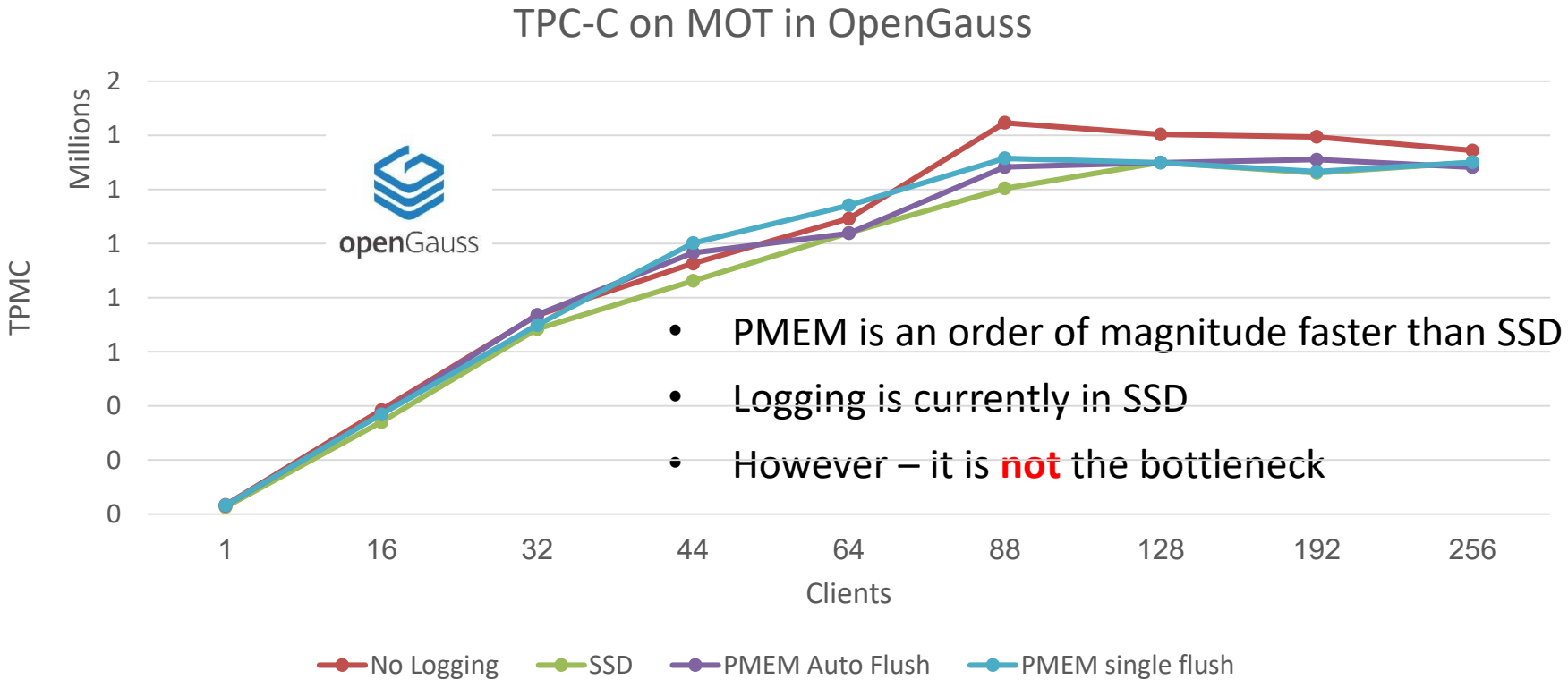
MOT FDW Extensions

- Creating Tables
- Creating Indexes
- Planning
- Execution
- **Logging**
- Checkpointing
- Vacuum
- DROP table/index

PMEM vs. SSD Logging Microbenchmark



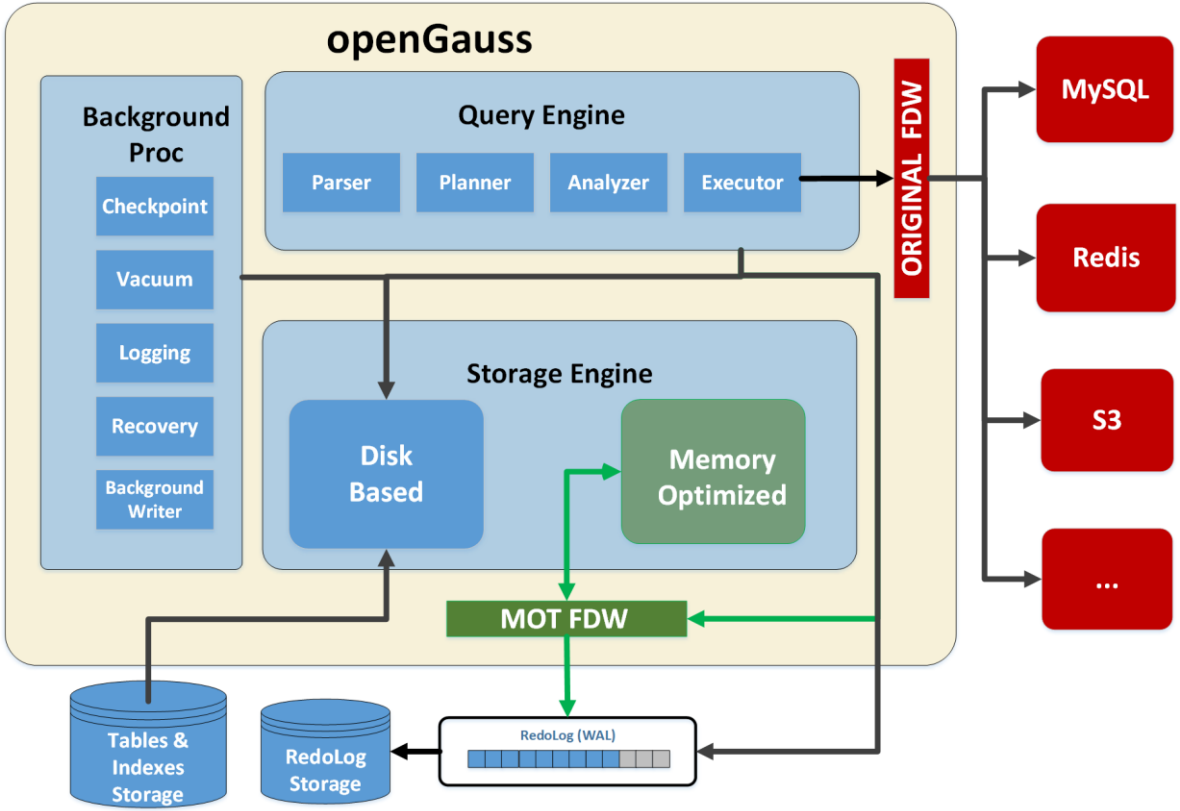
PMEM vs. SSD Logging in openGauss



- PMEM is an order of magnitude faster than SSD
- Logging is currently in SSD
- However – it is **not** the bottleneck

PMEM vs. DDR For Tables in MOT in GaussDB

Allocating Rows and Indexes in PMEM

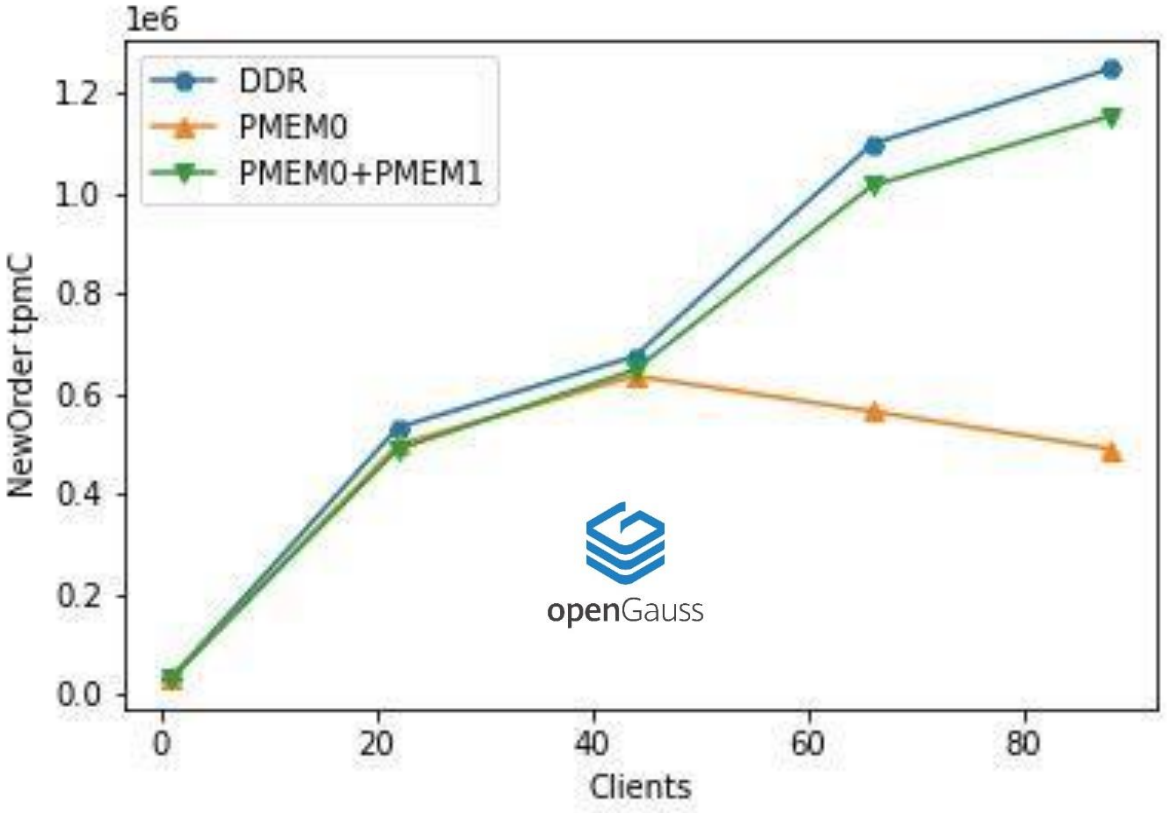


MOT FDW Extensions

- **Creating Tables**
- **Creating Indexes**
- Planning
- Execution
- Logging
- Checkpointing
- Vacuum
- DROP table/index

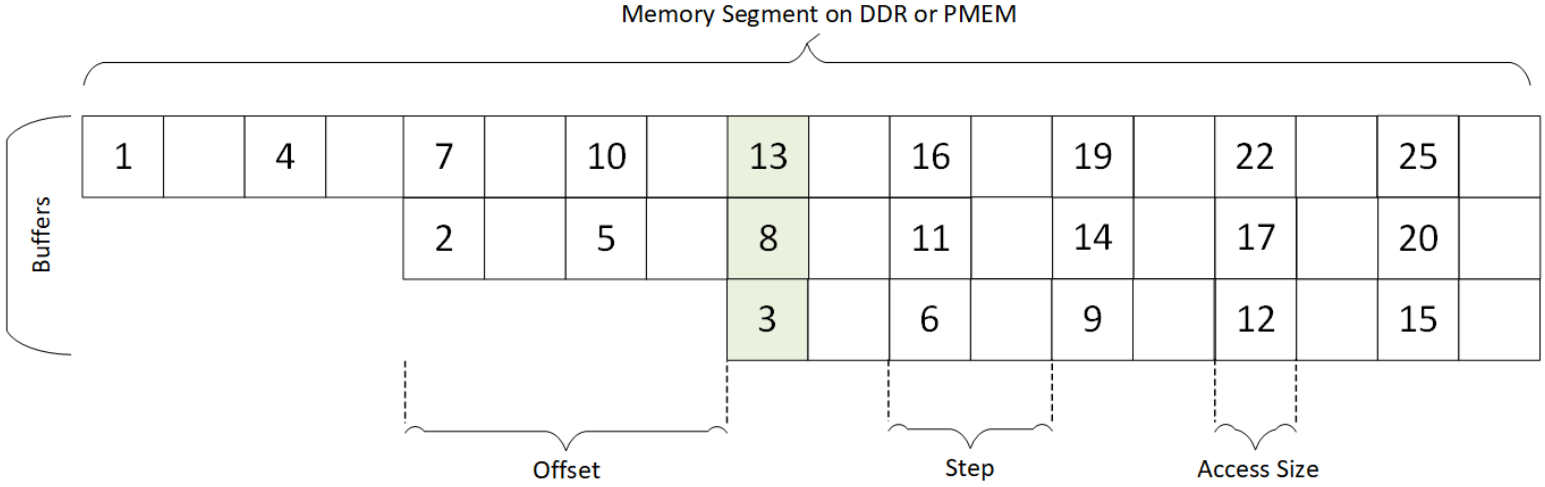
PMEM vs. DDR in MOT in openGauss

- PMEM reaches DDR performance.
- PMEM0 scales to 1 socket
- PMEM0 + PMEM1 scale to 88 clients (cores).
- Conclusion:
  - Bandwidth determines PMEM performance on openGauss.
  - The number of DIMMS determines the bandwidth
  - Latency is good enough to replace DDR in openGauss



PMEM vs. DDR Read / Write Microbenchmark

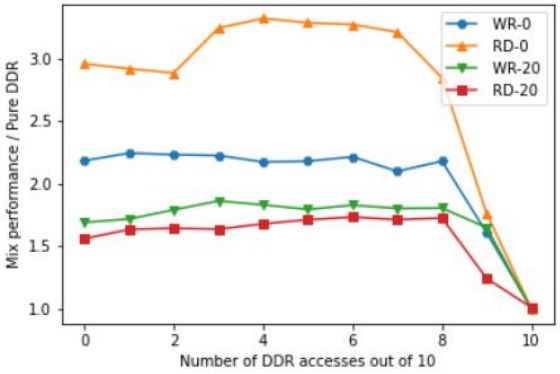
PMEM vs. DDR in RW Microbenchmark



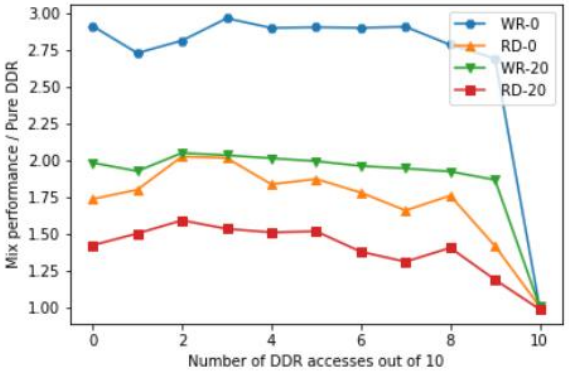
- Emulate non-sequential access
- Known number of accesses per address
- Control mix of DDR and PMEM accesses
- Known amount of memory accessed between consecutive accesses to the same address
- L2 cache misses are distributed uniformly
- Control the rate of L2 cache hits vs. L2 cache misses

PMEM vs. DDR in RW Microbenchmark - Results

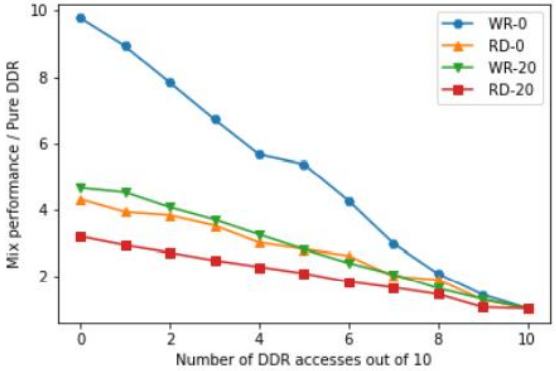
- Access patterns (lines):
  - **Hot:** In each phase the last access generates an L2 cache misse.
  - **Cold:** Each access generates an L2 cache miss.
  - **Index:** Access size is one cache line.
  - **Row:** Access size of 512.



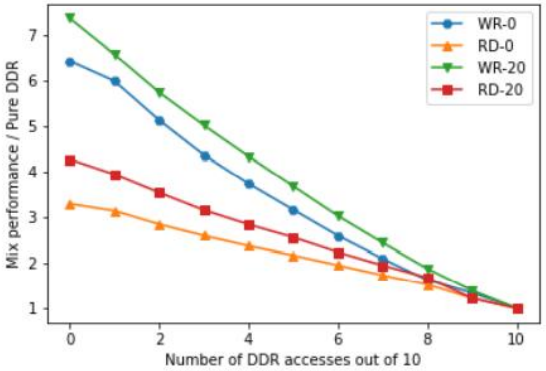
(a) Hot index



(b) Hot rows



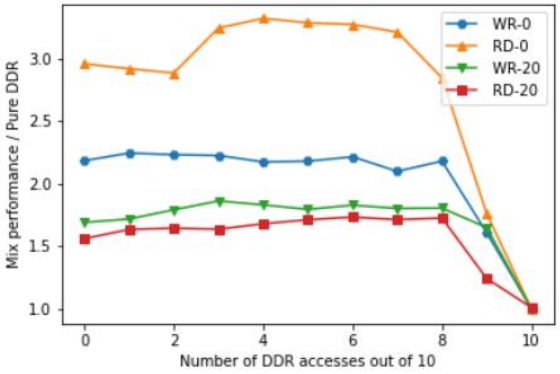
(c) Cold index



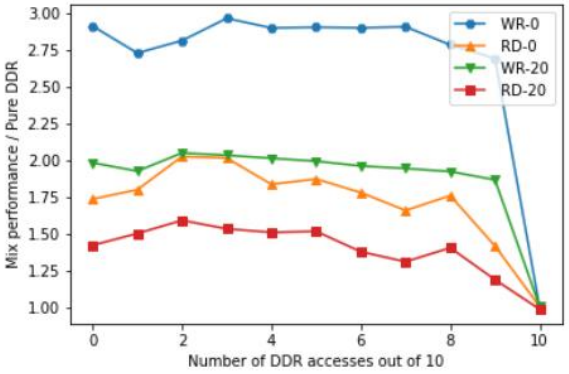
(d) Cold rows

PMEM vs. DDR in RW Microbenchmark - Results

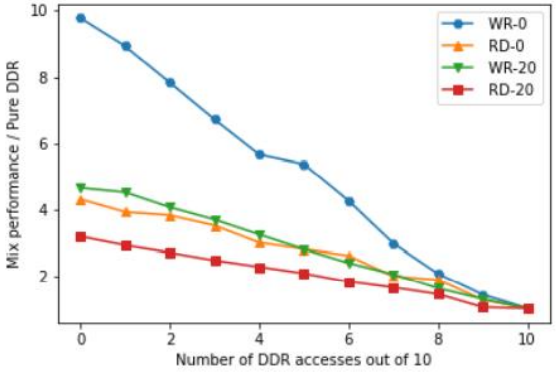
- Hot accesses are not benefiting from mixing with DDR.
- In hot index, compared to DDR, writes perform better than reads, while in hot rows reads are closer to DDR performance than writes
- Cold accesses are accelerated relative to the amount of DDR in the workload.
- Cold writes to indexes are slower than cold writes to rows.
- While in cold indexes delays make PMEM performance closer to DDR, in rows, without delays PMEM is closer to DDR performance.



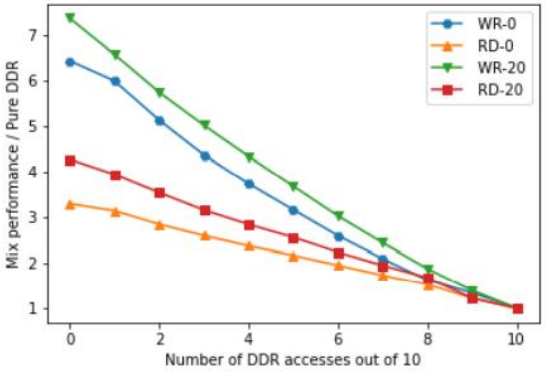
(a) Hot index



(b) Hot rows



(c) Cold index

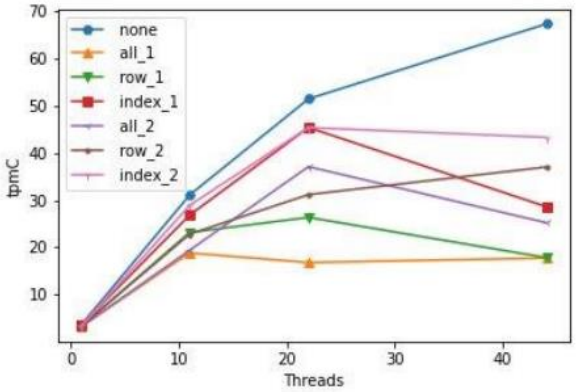


(d) Cold rows

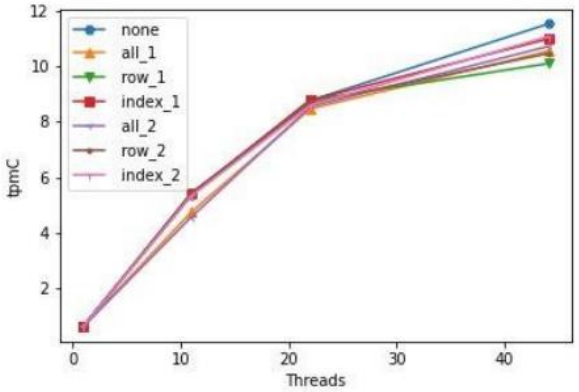
PMEM vs. DDR For Tables in MOT Microbenchmark

PMEM vs. DDR in MOT Microbenchmark

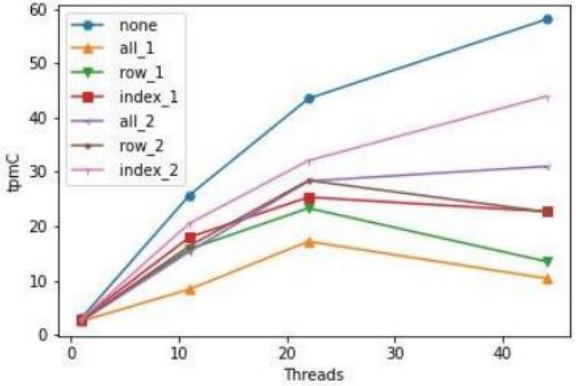
- Configurations (lines):
  - **none:** everything on DDR.
  - **cfg_1:** Use only PMEM0.
  - **cfg_2:** Use both PMEM0 and PMEM1.
  - **all:** Everything on PMEM.
  - **index:** Index on PMEM and rows on DDR.
  - **row:** Rows on PMEM and index on DDR.
  - Drafts are always on DDR.



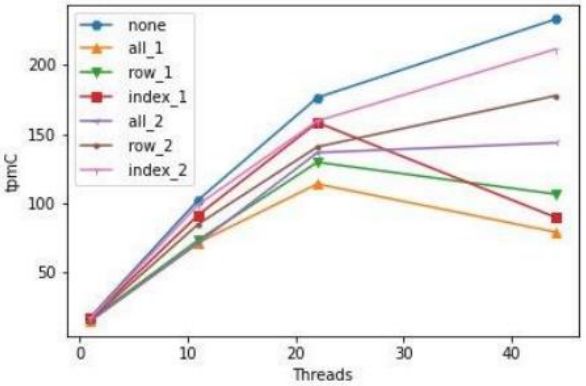
(a) Standard TPC-C mixture



(b) Read oriented TPC-C



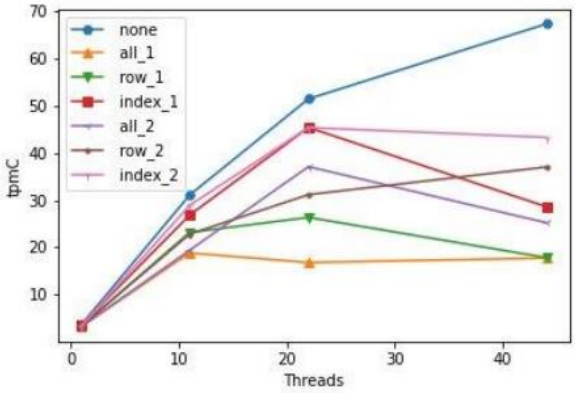
(c) Only NewOrder



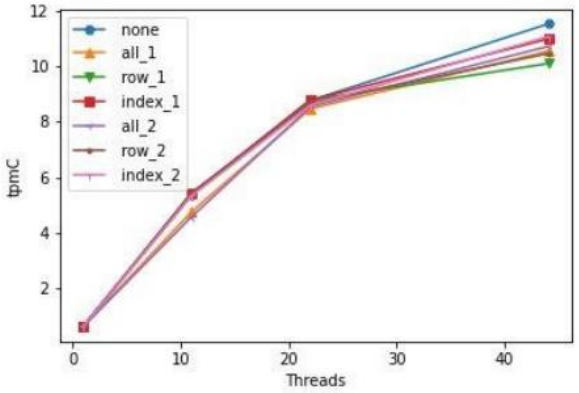
(d) Only Payment

PMEM vs. DDR in MOT Microbenchmark

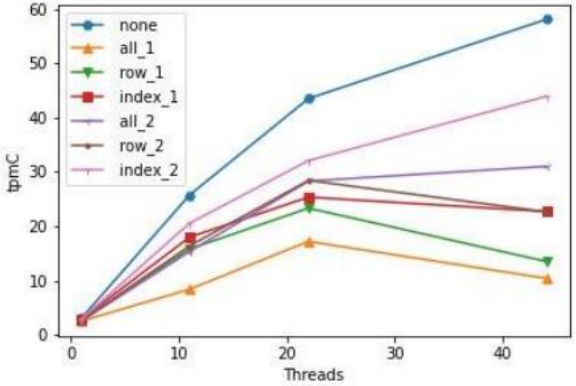
- In all workloads using the second PMEM modules gives performance advantage already with 11 threads.
- Even though PMEM1 is on the remote socket, the added bandwidth is boosting execution speed, and the NUMA effect is not stopping scalability.
- Putting only the indexes on PMEM is faster than putting only the rows on PMEM. The reason is both that index accesses are shorter and that the internal nodes are hot
- Figure (b) is mostly read, and as there are fewer NewOrder transactions, i.e. less inserts, the working set is smaller. As a result even when all data and indexes reside on PMEM execution speed is comparable to DDR.



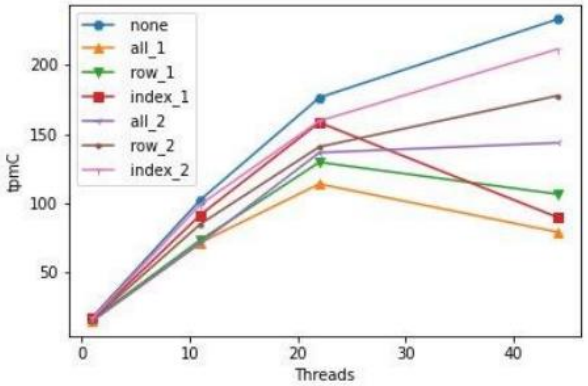
(a) Standard TPC-C mixture



(b) Read oriented TPC-C



(c) Only NewOrder



(d) Only Payment

Conclusion

Conclusions

- When MOT is embedded in the fully functional GaussDB DBMS PMEM preserves DDR performance.
- In microbenchmarks we see that Intel Optane 100 Series is not a transparent replacement for DDR as its lower bandwidth and higher latency can slow execution and limit scalability.
- Used in the right dosage, PMEM can extend memory size with only a small performance hit also in microbenchmarks.
- As faster PMEM products will appear, we expect it to be even more useful for in-memory databases.

THANK YOU!
謝謝

Code: https://gitee.com/opengauss and https://github.com/opengauss-mirror

Docs: https://opengauss.org/en/docs/2.0.1/docs/Developerguide/mot.html