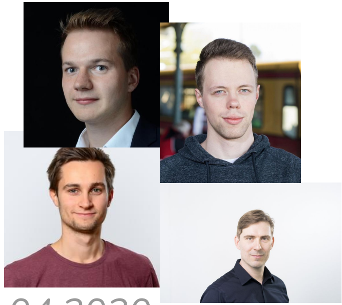
Maximizing Persistent Memory Bandwidth Utilization for OLAP Workloads

Björn Daase, Lars Jonas Bollmeier, Lawrence Benson, Tilmann Rabl
Data Engineering Systems Group, Hasso Plattner Institute
SIGMOD '21 | 19.04.2021

From Seminar to SIGMOD Paper



20.04.2020
Start of “Data Processing on Modern Hardware”

22.09.2020
Paper submitted

12.03.2021
Paper accepted

Data Processing on Modern Hardware

Instructors

Prof. Dr. Tilmann Rabl, Pedro Silva, Lawrence Benson, Ilin Tolovski, Wang Yue

Contents

In this project seminar, we will discuss data processing techniques on modern hardware. Specifically, we will look at the characteristics of modern processors (GPU, FPGA), memory (NVRAM), and network (RDMA) and research, how data processing can be done most efficiently on these devices. To this end, we will survey current trends in modern hardware, read and present research papers on data processing on modern hardware, identify a small research project and implement and evaluate a prototype for data processing on modern hardware.

31.08.2020
Official Ending of “Data Processing on Modern Hardware”

17.02.2021
Revision submitted

20-25.06.2021
2021 ACM SIGMOD Conference

HPI Data Lab

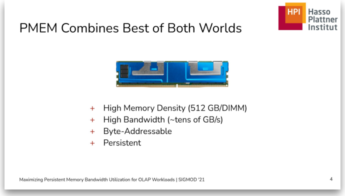
Infrastructure for Research Projects

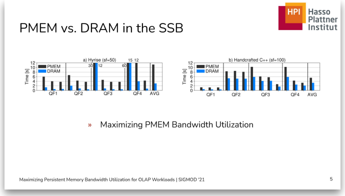
- FPGA
- GPU
- ARM
- **PMEM**

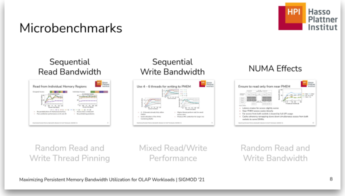
Great support by
Tobias Pape and Bernhard Rabe

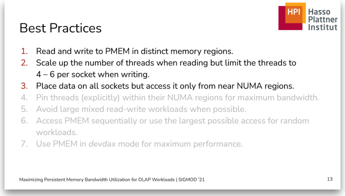
PMEM for OLAP

Evaluation and Optimization of PMEM for OLAP Workloads









DRAM vs. SSD

DRAM

[1]

- + Byte-Addressable
- + High Bandwidth
- - Low Memory Density (32 GB/DIMM)

SSD

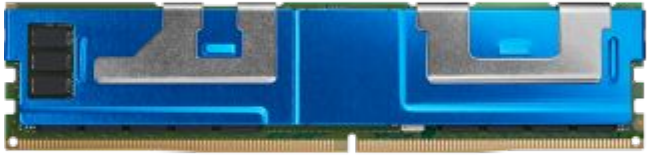
[2]

- + Persistent
- + High Memory Density
- - Low Bandwidth (~1 GB/s)

[1] https://www.goodram.com/wp-content/uploads/rdimm.jpg

[2] https://www.online-tech-tips.com/wp-content/uploads/2019/01/nvme-drive-ssd-e1548821393190.jpg.webp

PMEM Combines Best of Both Worlds

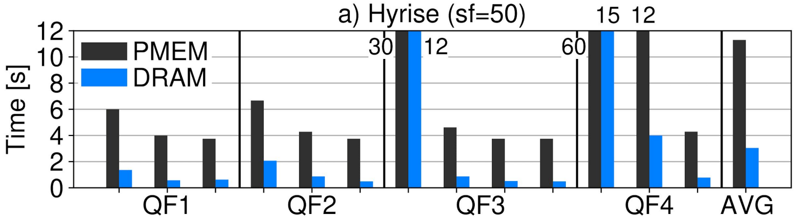


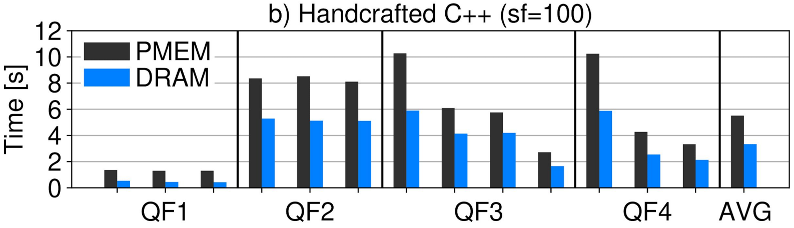
[3]

+ High Memory Density (512 GB/DIMM)
+ High Bandwidth (~tens of GB/s)
+ Byte-Addressable
+ Persistent

[3] https://www.online-tech-tips.com/wp-content/uploads/2019/01/nvme-drive-ssd-e1548821393190.jpg.webp

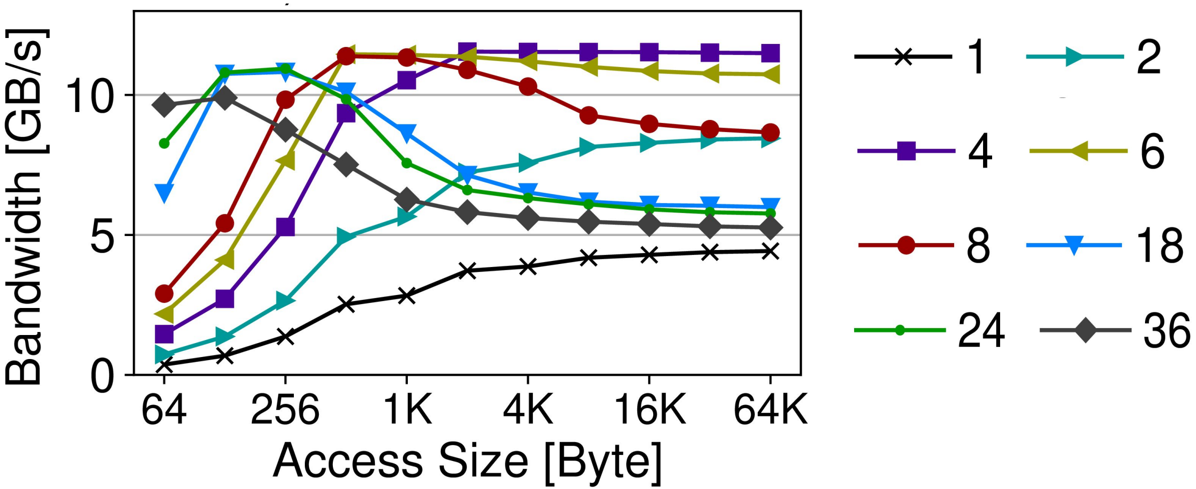
PMEM vs. DRAM in the SSB





» Maximizing PMEM Bandwidth Utilization

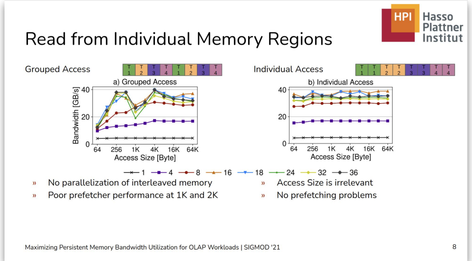
PMEM Bandwidth Behavior Unexpected



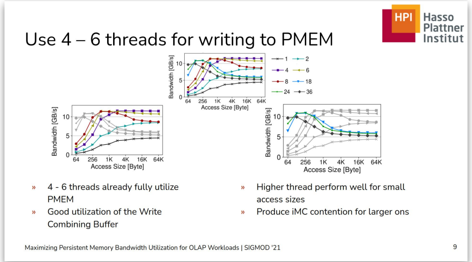
- Maximizing PMEM Bandwidth Utilization
- Understanding PMEM behavior

Microbenchmarks

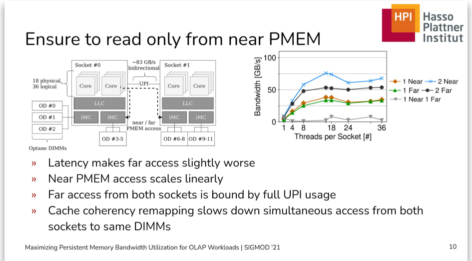
Sequential Read Bandwidth



Sequential Write Bandwidth



NUMA Effects

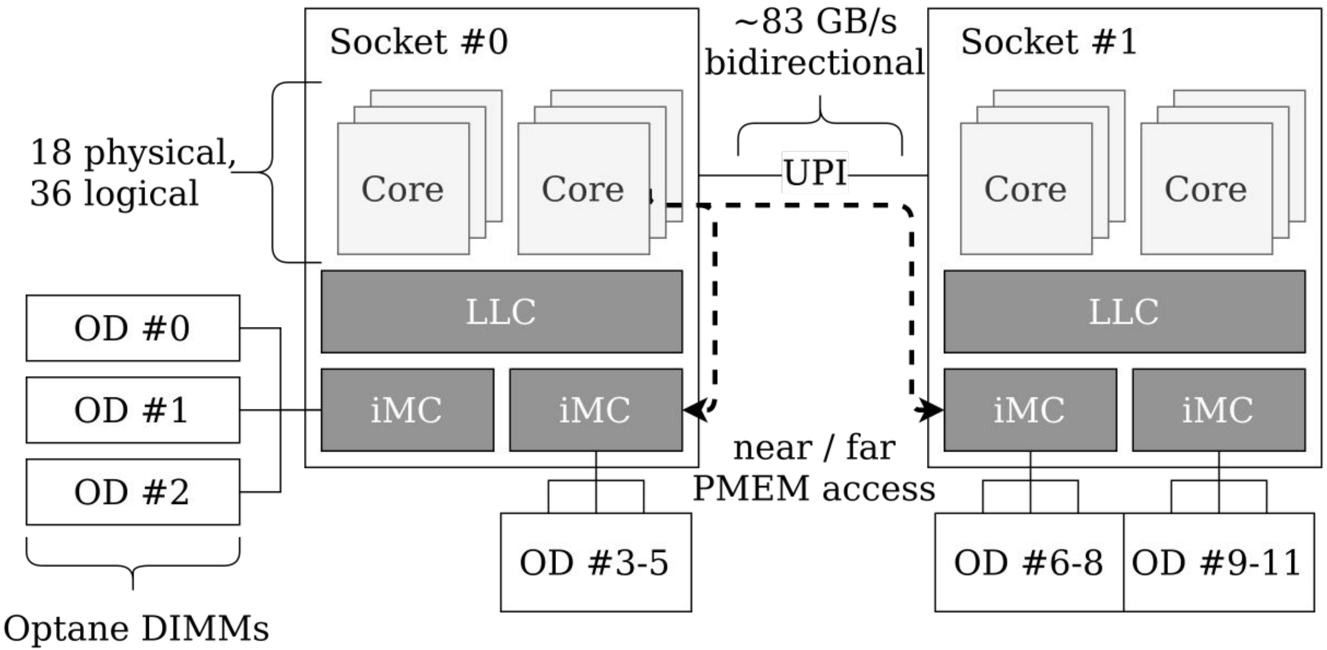


Read and Write Thread Pinning

Mixed Read/Write Performance

Random Read and Write Bandwidth

PMEM Benchmarking System

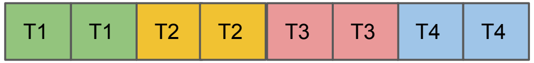


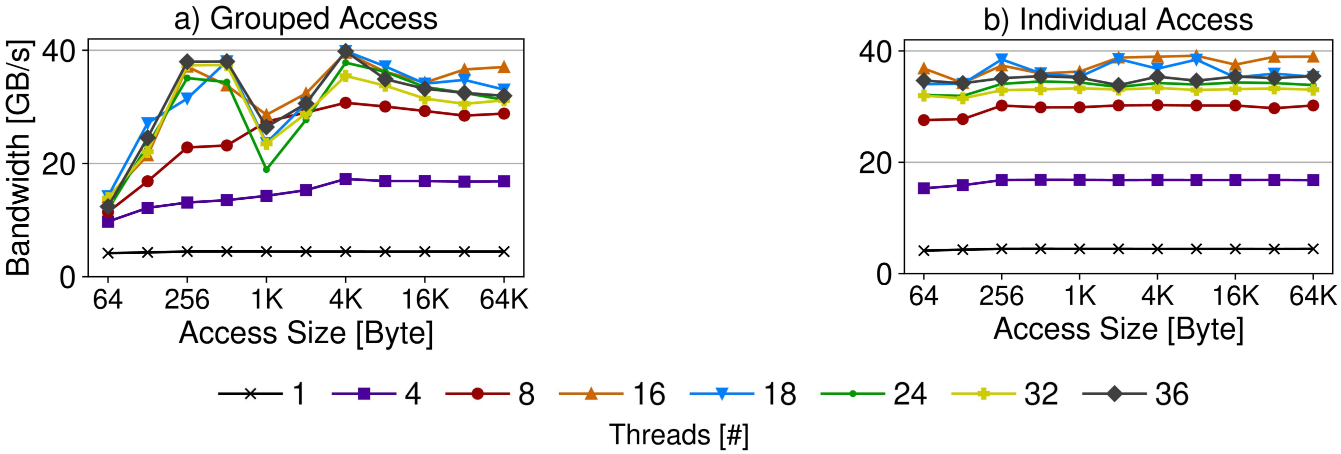
- 2 x Intel Xeon Gold 5220S @ 2.70 GHz
- 12 x 128 GB Intel Optane DIMMs
- 12 x 16 GB Samsung DDR4 DIMMs
- Ubuntu 18.04

Read from Individual Memory Regions

Grouped Access

Individual Access

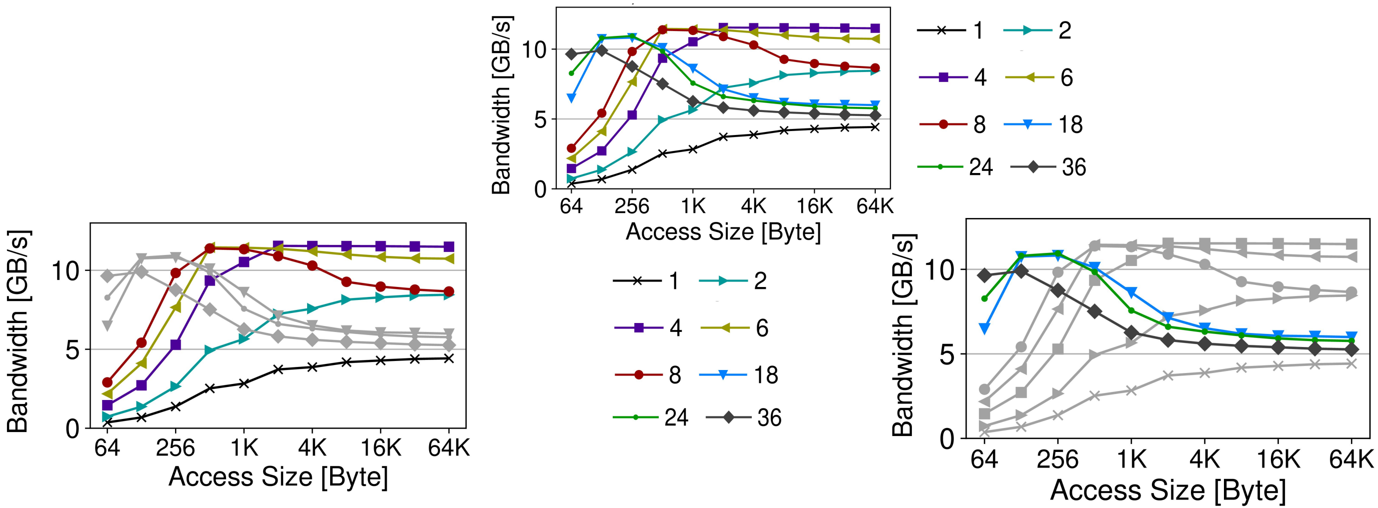




- » No parallelization of interleaved memory
- » Poor prefetcher performance at 1K and 2K

- » Access size is irrelevant
- » No prefetching problems

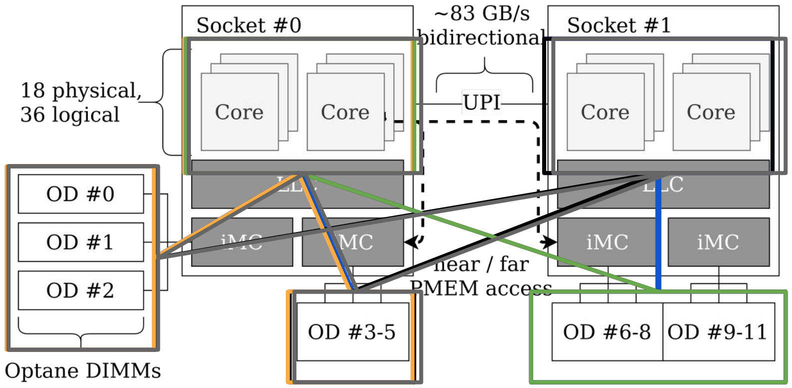
Use 4 – 6 Threads for Sequential Writing

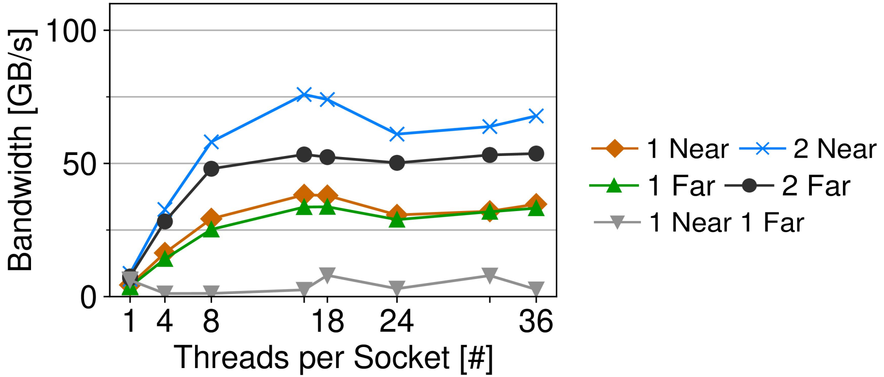


- 4 - 6 sequential threads already saturate PMEM
- Good utilization of the Write Combining Buffer

- Higher thread counts perform well for small access sizes
- iMC contention for larger access sizes

Read only from Near PMEM





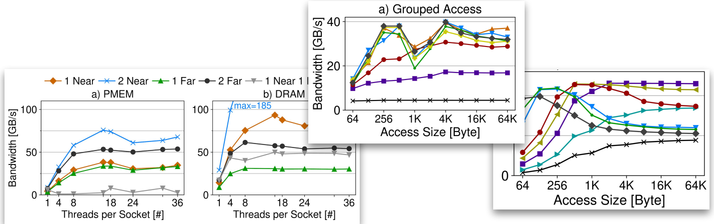
- » Latency makes far access slightly worse
- » Near PMEM access scales linearly
- » Far access from both sockets is UPI bound
- » Poor cross-socket access due to cache coherency

Summary

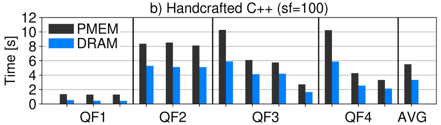
» Read behavior:  [1] ≈  [3]

» Write behavior:  [1] ≠  [3]

» Many aspects play in on multi-socket systems



» PMEM on average only 1.66x slower than DRAM for OLAP



Best Practices

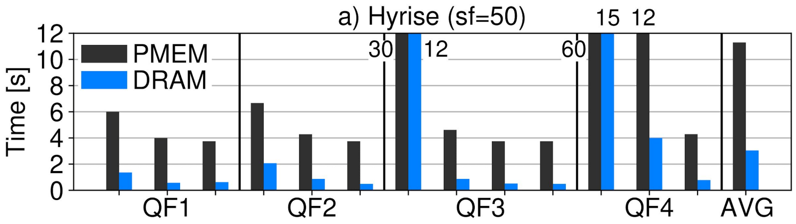
1. Read and write to PMEM in distinct memory regions.
2. Scale up the number of threads when reading but limit the threads to 4 – 6 per socket when writing.
3. Place data on all sockets but access it only from near NUMA regions.
4. Pin threads (explicitly) within their NUMA regions for maximum bandwidth.
5. Avoid large mixed read-write workloads when possible.
6. Access PMEM sequentially or use the largest possible access for random workloads.
7. Use PMEM in *devdax* mode for maximum performance.

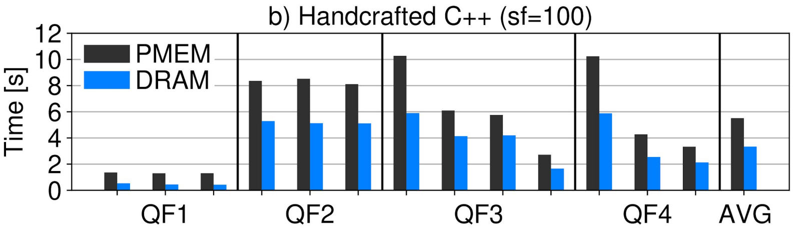
Questions?

https://hpi.de/rabl

bjoern.daase@student.hpi.de, lars.bollmeier@student.hpi.de

Star Schema Benchmark





- On average, PMEM is 3.7x slower than DRAM
- Over 90% of time lost in hash-operations

- On average, PMEM is 1.66x slower than DRAM
- PMEM-optimized hash index highly beneficial