OSNABRÜCK UNIVERSITY

Embedded Software Systems

Project Report: SMAUG

System-Level Modeling and Optimized Use of Disruptive Memory Technologies

Birte Friesel, Olaf Spinczyk

birte.friesel@uos.de

October 17, 2023

Universität Osnabrück

Motivation

In-memory and **near-memory computing** can save bandwidth and offer parallelism

Multi-core and **many-core** CPUs with **shared** and **distributed caches**

Novel memory technologies have **different properties**

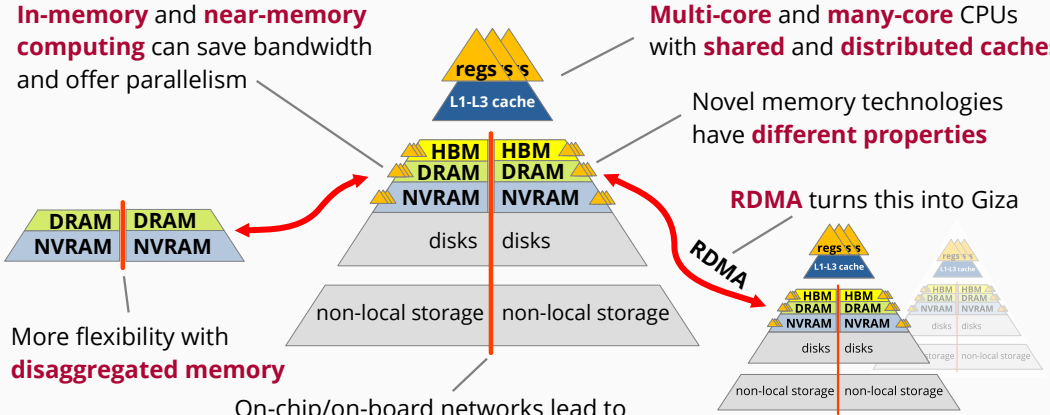
RDMA turns this into Giza

More flexibility with **disaggregated memory**

On-chip/on-board networks lead to **non-uniform I/O** and **memory architectures**

State-of-the-art models and placement algorithms:
view single technology in isolation or lack support for disruptive technologies

Motivation



⇒ Performance models and performance-aware placement algorithms with **full-system perspective** (contention, setup cost, . . .)

Workflow

- Test bed → measurements → modeling → validation
  - Non-Volatile RAM (Optane NVRAM)
  - Remote Direct Memory Access (Infiniband RDMA)
  - Near- and In-Memory Computing (UPMEM PIM)
  - High-Bandwidth Memory (Xilinx Alveo HBM)

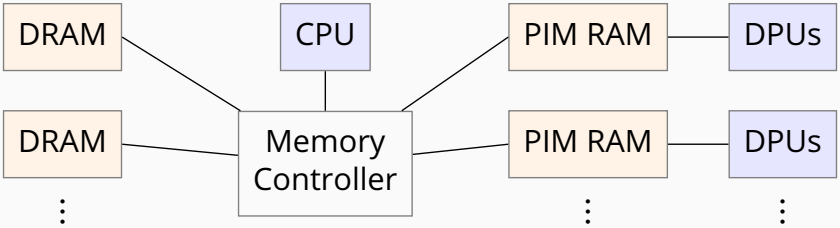
Workflow

- First step: suitable benchmark metrics and methodologies
- Test bed → measurements → modeling → validation
  - Non-Volatile RAM (Optane NVRAM)
  - Remote Direct Memory Access (Infiniband RDMA)
  - Near- and In-Memory Computing (case study: UPMEM PIM)
  - High-Bandwidth Memory (Xilinx Alveo HBM)

Outline

1. UPMEM PIM
2. Challenges
3. Implications

Near- and In-Memory Computing (UPMEM PIM)



- “Processing in Memory” promises transparent data processing, e.g. automatic (offloaded) transformation of data objects
- Common assumption: Data Processing Units (DPUs) ≈ additional CPUs: shared memory; same architecture; no overhead
- UPMEM PIM: First (and only) commercially available platform

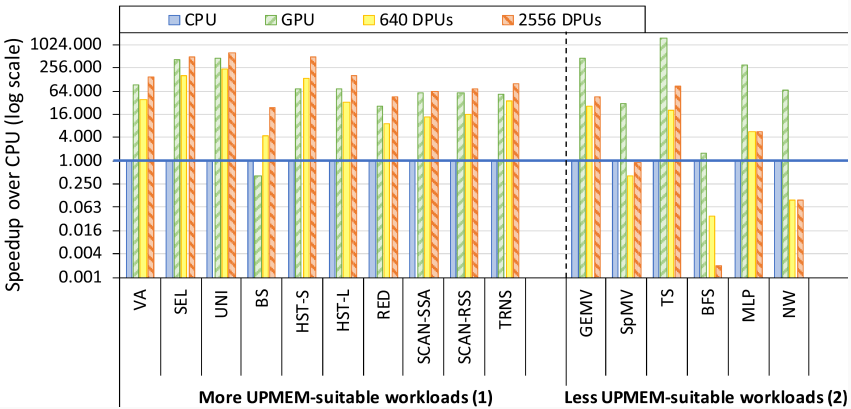
Benchmark and Model Metrics

- “Which technology is best suited for a given workload?”
- Simple and common approach: *speedup*

$$speedup = \frac{Throughput}{Throughput_{baseline}} \quad \text{or} \quad speedup = \frac{Latency_{baseline}}{Latency}$$

- $speedup > 1 \Rightarrow$ new technology is likely better (i.e., *speedup* times faster)

Case Study: UPMEM PIM



PrIM suite: speedup of UPMEM PIM over CPU execution [Góm+22]

⇒ PIM seems useful for DB acceleration, vector processing, data analysis . . .

Challenges

- Fair benchmarks are challenging: speedup is useless without context
- Well-known pitfalls in conventional benchmarks [HB15; Dav95; Lee+10]
- Lack of best practices for disruptive memory technologies

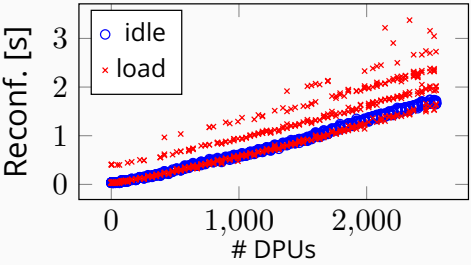
Challenges

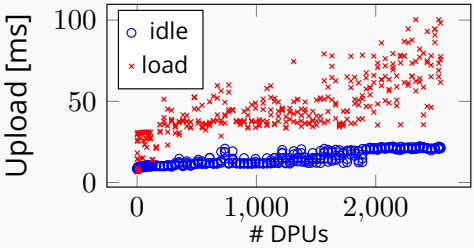
- Fair benchmarks are challenging: speedup is useless without context
- Well-known pitfalls in conventional benchmarks [HB15; Dav95; Lee+10]
- Lack of best practices for disruptive memory technologies
- UPMEM challenges include:
  1. Data transfer and reconfiguration cost
  2. Different architectures → different code and algorithm optimization
  3. Resource allocation (# CPU cores, # DPUs / UPMEM modules . . .)

① Data Transfer and Reconfiguration

- Wide-spread assumption: Negligible overhead

! UPMEM: costly and contention-sensitive setup and program upload

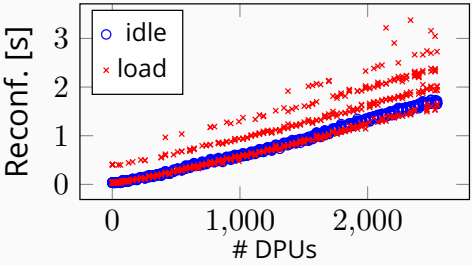


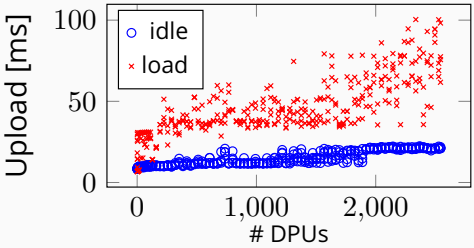


① Data Transfer and Reconfiguration

- Wide-spread assumption: Negligible overhead

! UPMEM: costly and contention-sensitive setup and program upload

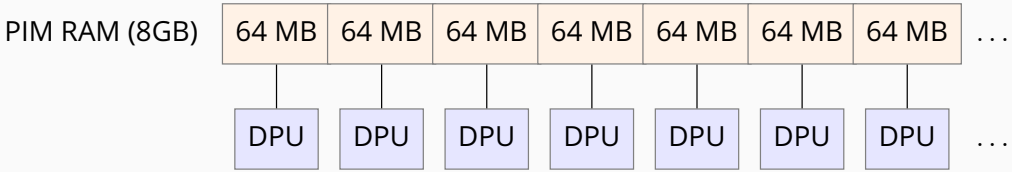




- UPMEM SDK treats PIM RAM like offloading engines

! Costly DRAM ↔ PIM RAM data transfers for each workload

② Code and Algorithm Optimization



- DPUs in PIM RAM ≠ CPU+RAM
- 64MB chunk of DRAM per DPU, no shared memory
- Synchronization and data exchange via host CPU (expensive)

! Workloads must be carefully optimized for UPMEM

③ Resource Allocation

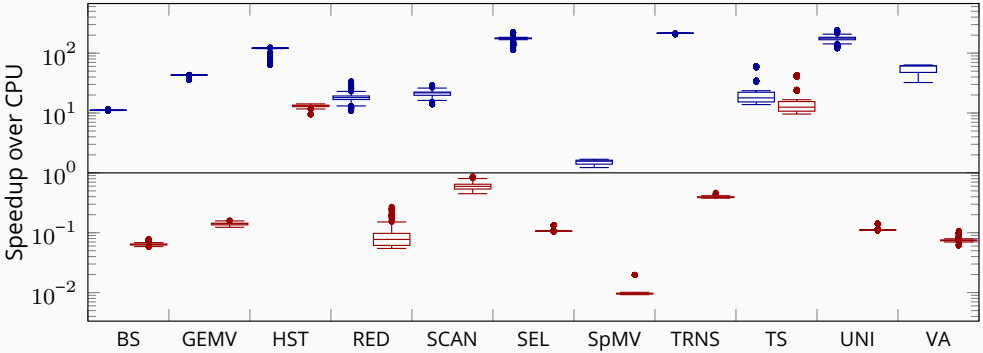
- Workloads run on some number of threads or DPUs → how many?
- All DPUs vs. single thread (UPMEM SDK needs one housekeeping core)
- All DPUs vs. all threads (UPMEM server cost ≈ 2× conventional server)
- some DPUs vs. some threads (parallelization; sub-linear scaling)

③ Resource Allocation

- Workloads run on some number of threads or DPUs → how many?
- All DPUs vs. single thread (UPMEM SDK needs one housekeeping core)
- All DPUs vs. all threads (UPMEM server cost ≈ 2× conventional server)
- some DPUs vs. some threads (parallelization; sub-linear scaling)

! Models must account for variable #CPU threads and #DPUs

Wrap-Up



left: PIM (kernel only) over single-threaded CPU right: PIM (with overhead) over best CPU config

⇒ Only **2 out of 11** workloads benefit in all cases

- Paper to appear at DIMES'23 [FLS23]

Conclusion and Next Steps

Done:

- Performance metrics and benchmark pitfalls; focus: UPMEM PIM

→ DIMES'23 paper, WSOS'23 lecture, WSOS'23 UPMEM Hackathon

Next:

- Contention-aware UPMEM PIM model
- Metrics, benchmarks and models for:
  - RDMA
  - HBM
  - NVRAM

References i

[Dav95] Andrew Davison. **“Twelve Ways to Fool the Masses When Giving Performance Results on Parallel Computers”.** In: Supercomputing Review (Aug. 1995), pp. 54–55.

[FLS23] Birte Friesel, Marcel Lütke Dreimann, and Olaf Spinczyk. **“A Full-System Perspective on UPMEM Performance”.** In: Proceedings of the 1st Workshop on Disruptive Memory Systems. DIMES’23. to appear. Koblenz, Germany: ACM Press, 2023.

[Góm+22] Juan Gómez-Luna et al. **“Benchmarking a New Paradigm: Experimental Analysis and Characterization of a Real Processing-in-Memory System”.** In: IEEE Access 10 (2022), pp. 52565–52608. DOI: 10.1109/ACCESS.2022.3174101.

References ii

[HB15] Torsten Hoefler and Roberto Belli. **“Scientific Benchmarking of Parallel Computing Systems: Twelve Ways to Tell the Masses When Reporting Performance Results”.** In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. SC ’15. Austin, Texas: Association for Computing Machinery, 2015. ISBN: 9781450337236. DOI: 10.1145/2807591.2807644. URL: https://doi.org/10.1145/2807591.2807644.

[Lee+10] Victor W. Lee et al. **“Debunking the 100X GPU vs. CPU Myth: An Evaluation of Throughput Computing on CPU and GPU”.** In: SIGARCH Comput. Archit. News 38.3 (June 2010), pp. 451–460. ISSN: 0163-5964. DOI: 10.1145/1816038.1816021. URL: https://doi.org/10.1145/1816038.1816021.