OSNABRÜCK UNIVERSITY

Embedded Software Systems

Project Report: SMAUG

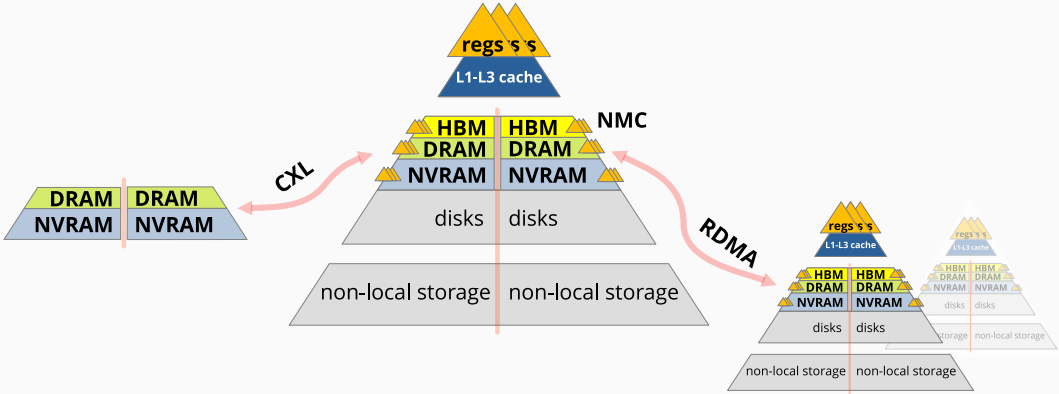
System-Level Modeling and Optimized Use of Disruptive Memory Technologies

Birte Friesel, Michael Müller, Olaf Spinczyk birte.friesel@uos.de

September 17, 2024

Universität Osnabrück

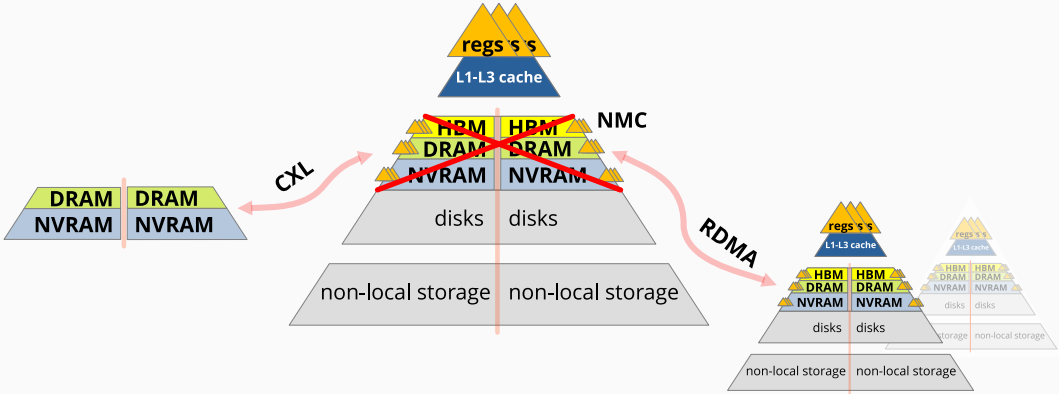
Motivation



Memory hierarchies are becoming increasingly complex

System software needs HW/SW models for resource management

Motivation – Year 2



Disruptive memory technologies break out of the hierarchy:
HBM has higher latency than DRAM → not a suitable DRAM cache
Today's NMC behaves like an accelerator → must be treated as such

Year 2 Results

- Performance models: High-Bandwidth Memory (Intel Xeon Max 9462)

- Performance models: Near-Memory Computing (UPMEM PIM)

Year 2 Results

- Performance models: High-Bandwidth Memory (Intel Xeon Max 9462)
  - Data transfer performance and data access latency
  - → HBM can slow down applications; depends on access patterns [FLS24] (verified by placement benchmarks on real-world applications)
- Performance models: Near-Memory Computing (UPMEM PIM)

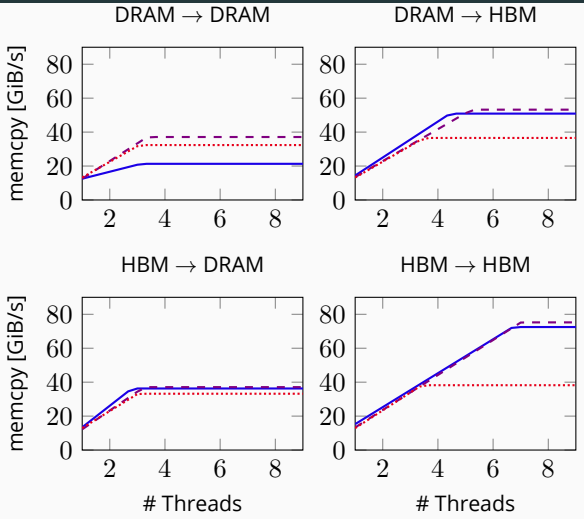
Year 2 Results

- Performance models: High-Bandwidth Memory (Intel Xeon Max 9462)
  - Data transfer performance and data access latency
  - → HBM can slow down applications; depends on access patterns [FLS24]
    (verified by placement benchmarks on real-world applications)
- Performance models: Near-Memory Computing (UPMEM PIM)
  - Setup cost and data transfer overhead diminish advantages of NMC [FLS23]
  - Models are suitable for placement decisions [FLS24]
    (evaluated by model integration in HetSim scheduling simulator [LFS24])
  - → **UpVec** cooperation (SMAUG + ParPerOS + Coordination):
    direct PIM RAM access via selective data transformation

High-Bandwidth Memory: Data Transfer Throughput

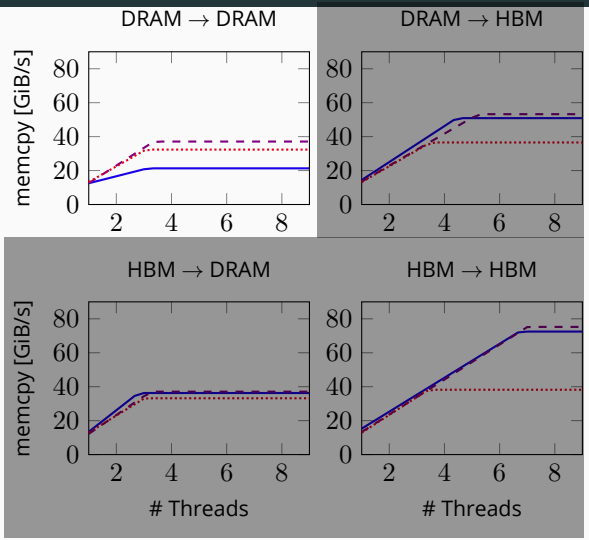
- 2× Xeon Max 9462 (`milos`)
  $2 \times 8\times$ (16 GiB DRAM + 8 GiB HBM)
- Roofline models:
  $\beta_0 + \beta_1 \cdot \min(\#\textit{threads}, \beta_2)$

High-Bandwidth Memory: Data Transfer Throughput



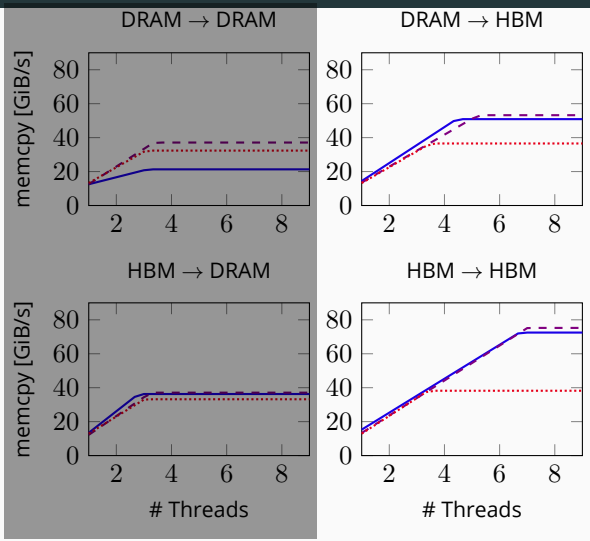
- 2× Xeon Max 9462 (`milos`)
  2 × 8× (16 GiB DRAM + 8 GiB HBM)
- Roofline models:
  $\beta_0 + \beta_1 \cdot \min(\#\mathit{threads}, \beta_2)$
- local / intra / cross package
  (10 % model error)

High-Bandwidth Memory: Data Transfer Throughput



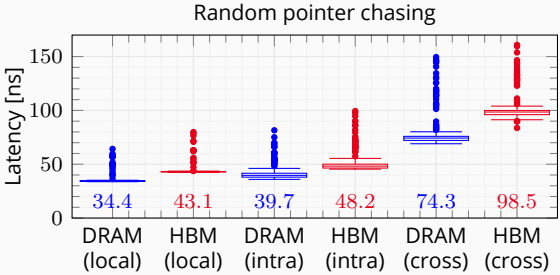
- 2× Xeon Max 9462 (`milos`)
  2 × 8× (16 GiB DRAM + 8 GiB HBM)
- Roofline models:
  $\beta_0 + \beta_1 \cdot \min(\#\mathit{threads}, \beta_2)$
- local / intra / cross package
  (10 % model error)
- DRAM: remote in/out is faster

High-Bandwidth Memory: Data Transfer Throughput



- 2× Xeon Max 9462 (`milos`)
  2 × 8× (16 GiB DRAM + 8 GiB HBM)
- Roofline models:
  $\beta_0 + \beta_1 \cdot \min(\#\mathit{threads}, \beta_2)$
- local / intra / cross package
  (10 % model error)
- DRAM: remote in/out is faster
- HBM: interconnect is bottleneck

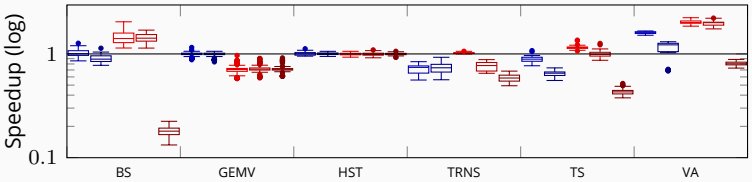
High-Bandwidth Memory: Random Access Latency



- Local HBM significantly slower than local and intra-package DRAM
- Cache hides latency for sequential and strided access patterns
- 4 % model error

High-Bandwidth Memory: Implications

- Evaluation on real-world parallel workloads

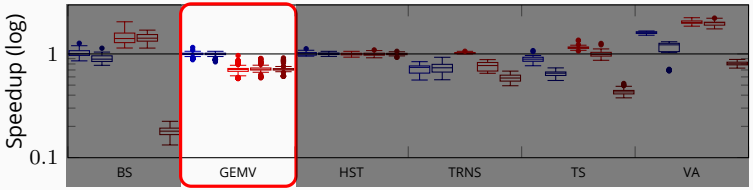


intra/cross-package DRAM ↔ local DRAM

local/intra/cross-pkg HBM ↔ local DRAM

High-Bandwidth Memory: Implications

- Evaluation on real-world parallel workloads



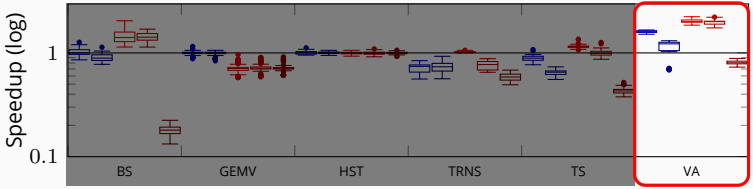
intra/cross-package DRAM ↔ local DRAM

local/intra/cross-pkg HBM ↔ local DRAM

- HBM can slow down workloads (matrix-vector multiplication)

High-Bandwidth Memory: Implications

- Evaluation on real-world parallel workloads



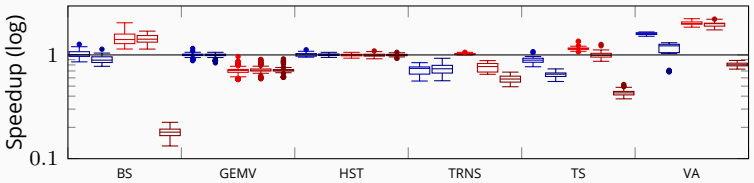
intra/cross-package DRAM ↔ local DRAM

local/intra/cross-pkg HBM ↔ local DRAM

- HBM can slow down workloads (matrix-vector multiplication)
- HBM or remote DRAM input can improve performance (vector addition)

High-Bandwidth Memory: Implications

- Evaluation on real-world parallel workloads

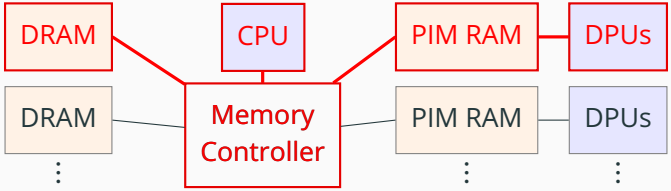


intra/cross-package DRAM ↔ local DRAM
local/intra/cross-pkg HBM ↔ local DRAM

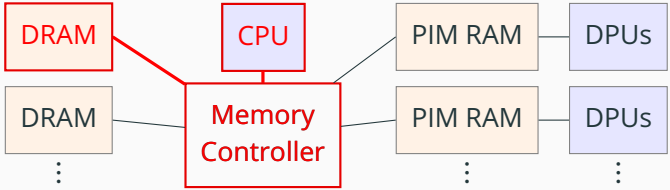
- HBM can slow down workloads (matrix-vector multiplication)
- HBM or remote DRAM input can improve performance (vector addition)

→ Placement algorithms need task-level access pattern annotations

Near-Memory Computing: UPMEM PIM

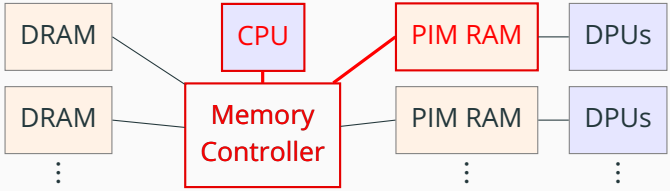


Near-Memory Computing: UPMEM PIM



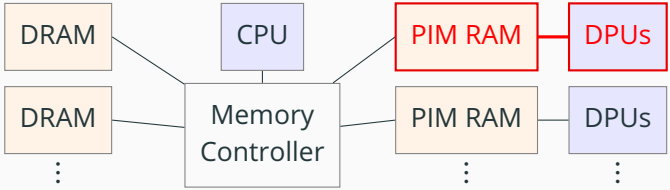
NMC ≠ UPMEM PIM

Near-Memory Computing: UPMEM PIM



NMC ≠ UPMEM PIM

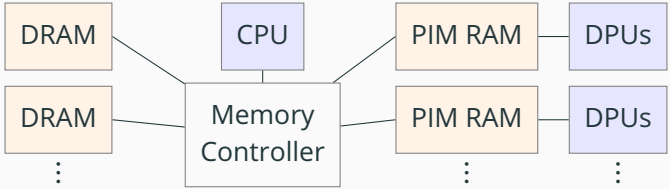
Near-Memory Computing: UPMEM PIM



NMC ≠ UPMEM PIM ≈ accelerator

- Speed-up on constant data due to massive parallelism [BJS23]; variable data or workloads are challenging [FLS23]

Near-Memory Computing: UPMEM PIM

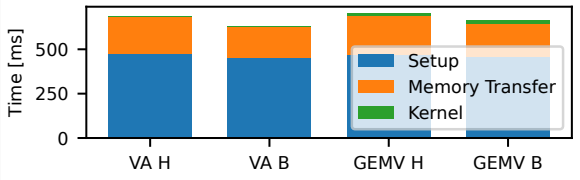


NMC ≠ UPMEM PIM ≈ accelerator

- Speed-up on constant data due to massive parallelism [BJS23]; variable data or workloads are challenging [FLS23]
- Performance model: setup cost and data transfer
- Notable improvements in UPMEM SDK 2024.1, but still costly

UPMEM PIM: Model Validation

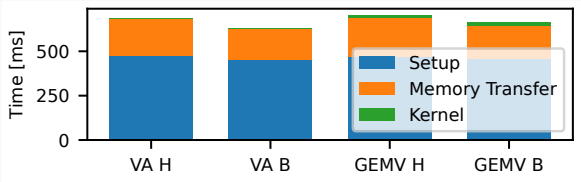
HetSim: Simulator for task-based scheduling on heterogeneous HW [LFS24]



HetSim vs **B**enchmark

UPMEM PIM: Model Validation

HetSim: Simulator for task-based scheduling on heterogeneous HW [LFS24]

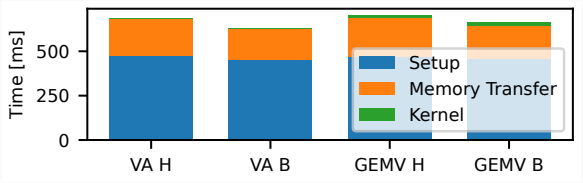


HetSim vs **B**enchmark

- Accurate prediction of setup cost and kernel run-time
- Sufficient for performance-aware scheduling decisions [FLS24]

UPMEM PIM: Model Validation

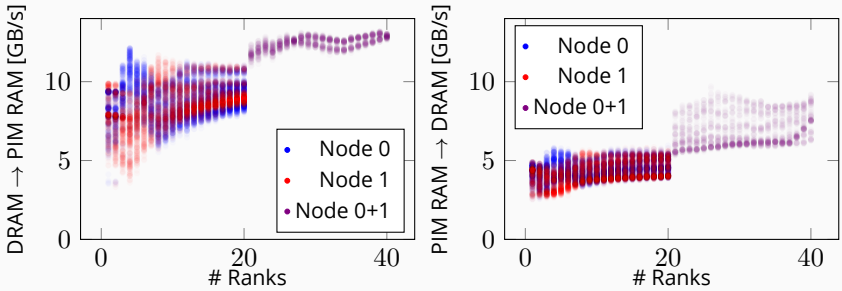
HetSim: Simulator for task-based scheduling on heterogeneous HW [LFS24]



HetSim vs **B**enchmark

- Accurate prediction of setup cost and kernel run-time
- Sufficient for performance-aware scheduling decisions [FLS24]
- 27 % mean error for memory transfers

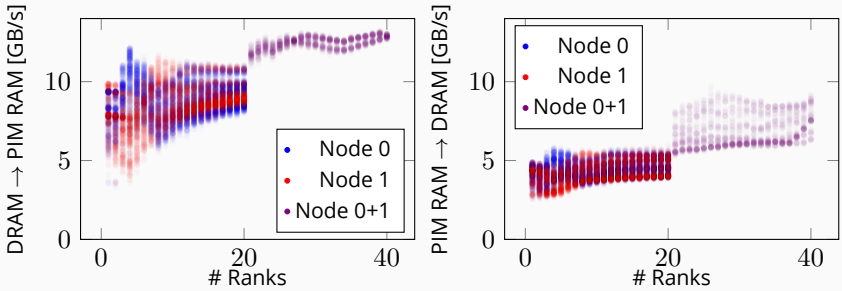
UPMEM PIM: Data Transfer



- Asymmetric, far below DRAM speed
- High variance due to SDK thread pool
- Transfer includes data transformation

- Single node: 8.7 GB/s / 4.2 GB/s
- Multi-node: 12.5 GB/s / 6.4 GB/s
- Model error: 12 %

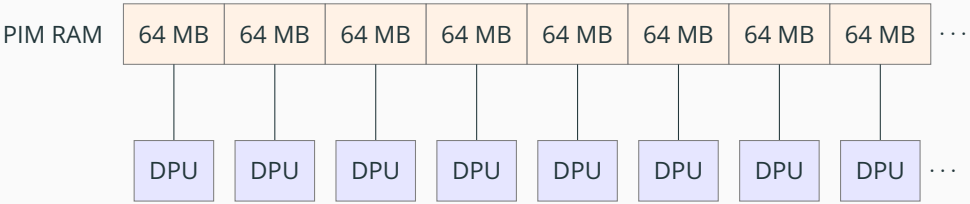
UPMEM PIM: Data Transfer



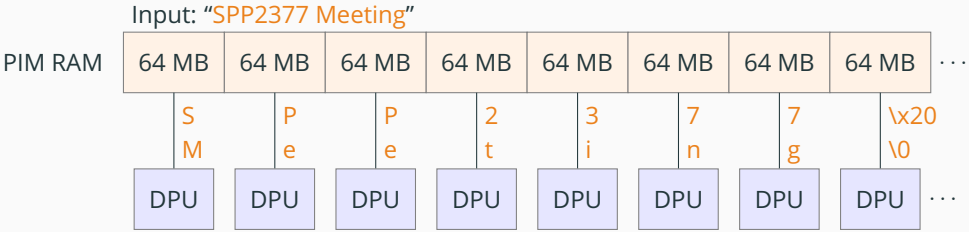
- Asymmetric, far below DRAM speed
- High variance due to SDK thread pool
- Transfer includes data transformation

- Single node: 8.7 GB/s / 4.2 GB/s
- Multi-node: 12.5 GB/s / 6.4 GB/s
- Model error: 12 %

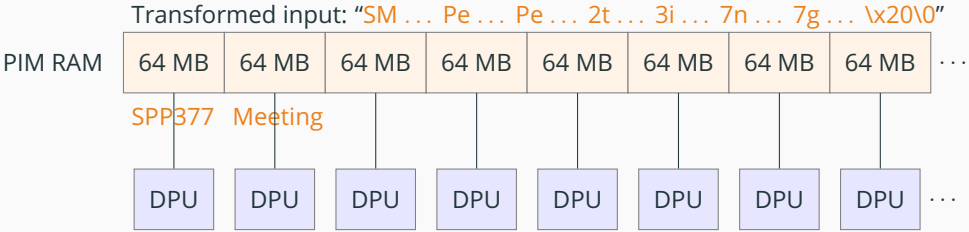
UPMEM PIM: Data Transformation



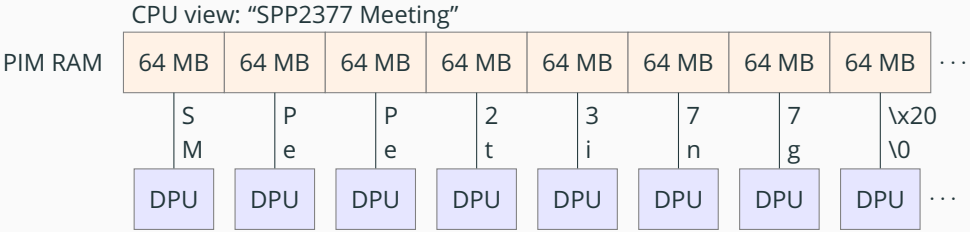
UPMEM PIM: Data Transformation



UPMEM PIM: Data Transformation



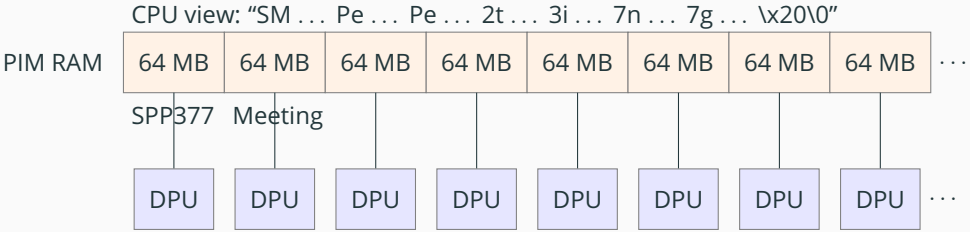
UpVec: Concept



UpVec: Store host data directly on PIM RAM

- Fine-grained on-demand transformation → avoid data transfer overhead
- Programs in PIM RAM; interpreter on DPUs → avoid setup costs

UpVec: Concept

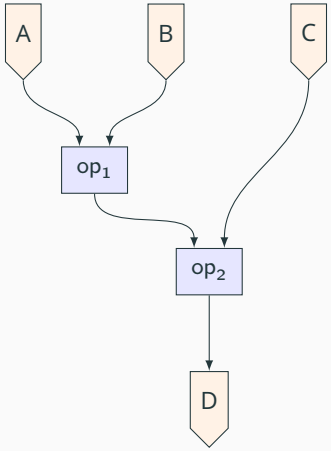


UpVec: Store host data directly on PIM RAM

- Fine-grained on-demand transformation → avoid data transfer overhead
- Programs in PIM RAM; interpreter on DPUs → avoid setup costs

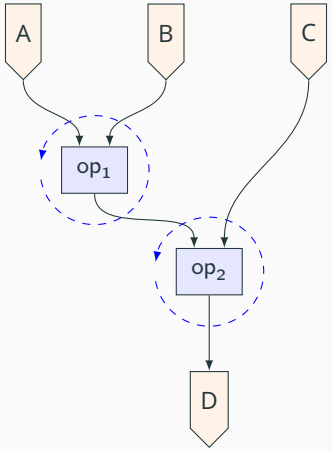
UpVec: Status

- STREAM benchmark with UpVec interpreter (worst-case scenario; promising results)
- Next: histogram on updatable vector of rows (existing works assume read-only data)



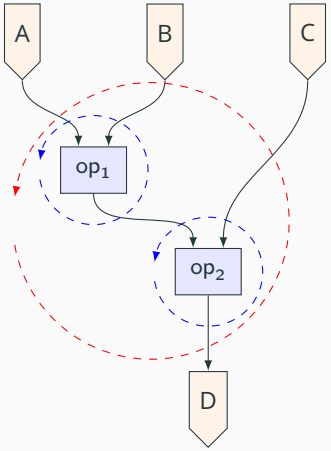
UpVec: Status

- STREAM benchmark with UpVec interpreter (worst-case scenario; promising results)
- Next: histogram on updatable vector of rows (existing works assume read-only data)



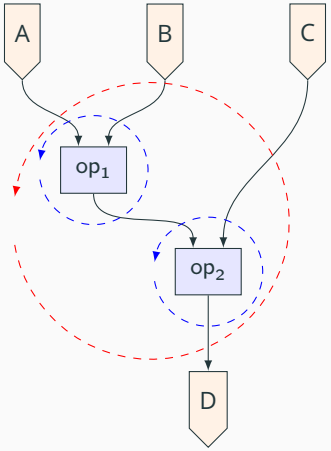
UpVec: Status

- STREAM benchmark with UpVec interpreter (worst-case scenario; promising results)
- Next: histogram on updatable vector of rows (existing works assume read-only data)



UpVec: Status

- STREAM benchmark with UpVec interpreter (worst-case scenario; promising results)
- Next: histogram on updatable vector of rows (existing works assume read-only data)



Cooperation

Ongoing:

- UpVec project (ParPerOS, Coordination)
- Survey on benchmarks for modern memory technologies (SMASH et al.)
- Using central hardware platform (HBM on milos, UPMEM on chios)

Cooperation

Ongoing:

- UpVec project (ParPerOS, Coordination)
- Survey on benchmarks for modern memory technologies (SMASH et al.)
- Using central hardware platform (HBM on milos, UPMEM on chios)

Opportunities:

- Experience with system-level HBM and UPMEM benchmarks
- Performance models → design space exploration

Conclusion and Outlook

✓ Performance models for HBM and NMC / UPMEM PIM

- Usable for task-level UPMEM scheduling; HBM needs annotations
- Published at WOSP-C ’24 [LFS24]; under submission at DIMES ’24 [FLS24]

Conclusion and Outlook

✓ Performance models for HBM and NMC / UPMEM PIM

- Usable for task-level UPMEM scheduling; HBM needs annotations
- Published at WOSP-C '24 [LFS24]; under submission at DIMES '24 [FLS24]

Up next:

- Workload annotations for HBM/DRAM placement
- Comparison with models for non-Linux OS, e.g. MxKernel
- Performance-aware placement on hardware (MxTasking integration)
- Metrics, benchmarks and models for CXL and RDMA

References i

[BJS23] Alexander Baumstark, Muhammad Attahir Jibril, and Kai-Uwe Sattler. **"Accelerating Large Table Scan Using Processing-In-Memory Technology".** In: Datenbank-Spektrum 23.3 (Oct. 10, 2023), pp. 199–209. ISSN: 1610-1995. DOI: 10.1007/s13222-023-00456-z.

[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. Koblenz, Germany: Association for Computing Machinery, 2023, pp. 1–7. ISBN: 9798400703003. DOI: 10.1145/3609308.3625266.

References ii

[FLS24] Birte Friesel, Marcel Lütke Dreimann, and Olaf Spinczyk. **“Performance Models for Task-based Scheduling with Disruptive Memory Technologies”.** In: Proceedings of the 2nd Workshop on Disruptive Memory Systems. DIMES ’24. Austin, TX, USA: Association for Computing Machinery, Nov. 2024.

[LFS24] Marcel Lütke Dreimann, Birte Friesel, and Olaf Spinczyk. **“HetSim: A Simulator for Task-based Scheduling on Heterogeneous Hardware”.** In: Companion of the 15th ACM/SPEC International Conference on Performance Engineering. ICPE ’24 Companion. London, UK: Association for Computing Machinery, May 2024, pp. 261–268. ISBN: 979-8-4007-0445-1. DOI: 10.1145/3629527.3652275.