OSNABRÜCK UNIVERSITY

Embedded Software Systems

A Full-System Perspective on UPMEM Performance

Birte Friesel, Marcel Lütke Dreimann, Olaf Spinczyk

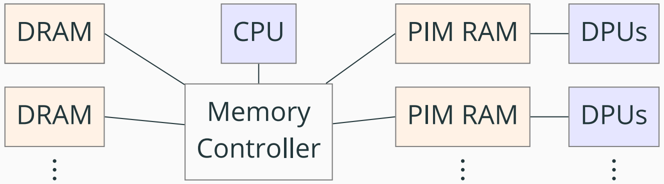
birte.friesel@uos.de

March 15, 2024

Osnabrück University

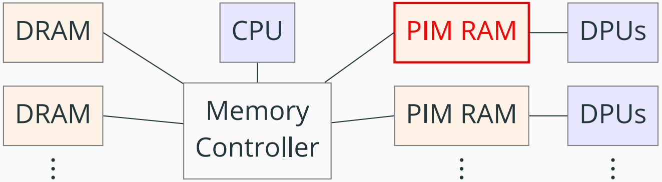
ess.cs.uos.de/~bf

“Processing in Memory” (PIM)



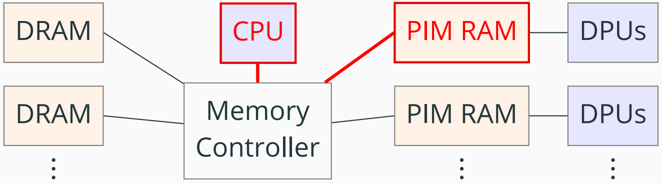
- Near-Memory Computing promises perfect world:
  - Transparent data processing in DRAM Processing Units (DPUs)
  - No memory controller bottleneck

"Processing in Memory" (PIM)



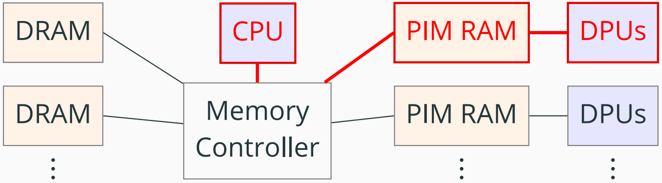
- Near-Memory Computing promises perfect world:
  - Transparent data processing in DRAM Processing Units (DPUs)
  - No memory controller bottleneck

“Processing in Memory” (PIM)



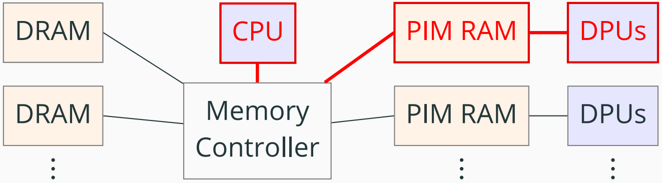
- Near-Memory Computing promises perfect world:
  - Transparent data processing in DRAM Processing Units (DPUs)
  - No memory controller bottleneck

“Processing in Memory” (PIM)



- Near-Memory Computing promises perfect world:
  - Transparent data processing in DRAM Processing Units (DPUs)
  - No memory controller bottleneck

“Processing in Memory” (PIM)

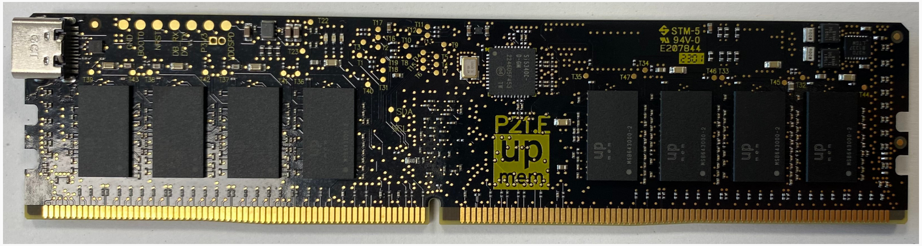


- Near-Memory Computing promises perfect world:
  - Transparent data processing in DRAM Processing Units (DPUs)
  - No memory controller bottleneck

→ How to determine PIM-friendly workloads?

- Research limited to simulators and custom FPGA builds until 2021

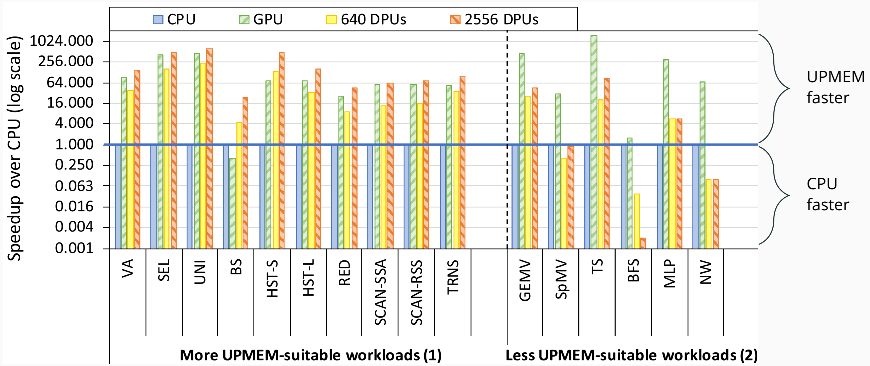
UPMEM PIM



- First (and only) commercially available processing-in-memory platform
  - 8 GB DDR4 modules (2 ranks × 4 GB)
  - 128 DPUs built into memory chips: 32-bit RISC @ 267 . . . 450 MHz
- Popular evaluation target [Góm+22; BJS23]

UPMEM: Promising Results

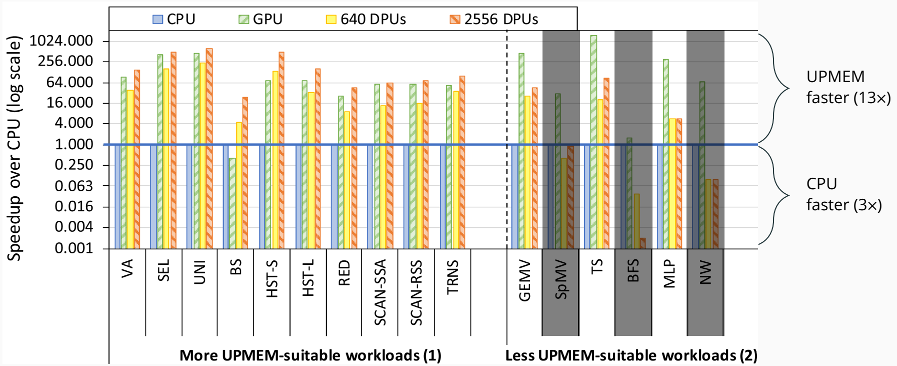
CPU: 4 threads on Intel Xeon E3-1240



PrIM suite: speedup of UPMEM PIM over quad-core CPU [Góm+22]

UPMEM: Promising Results

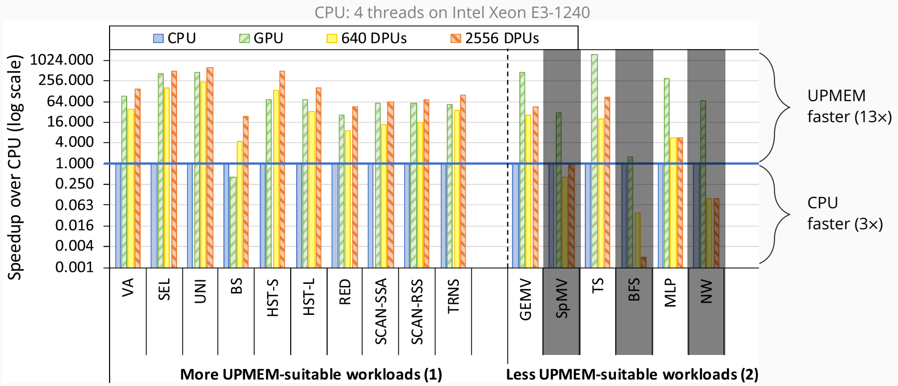
CPU: 4 threads on Intel Xeon E3-1240



PrIM suite: speedup of UPMEM PIM over quad-core CPU [Góm+22]

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

UPMEM: Promising Results(?)

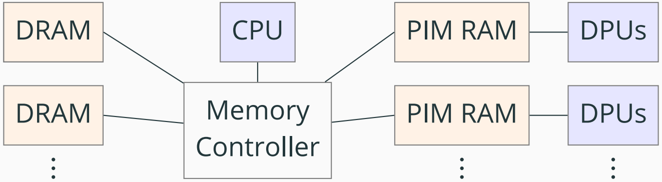


PrIM suite: speedup of UPMEM PIM over quad-core CPU [Góm+22]

. . . when leaving out overhead (assumption: amortized by chained kernels)

Outline

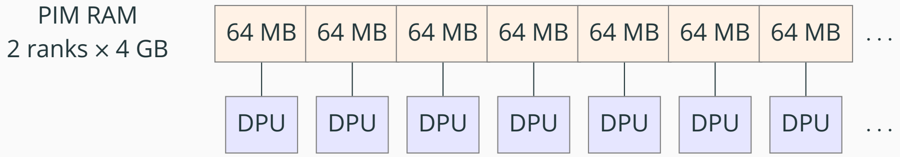
The Ideal PIM Architecture



Common assumption: DPUs behave like additional CPUs [Cor+21]

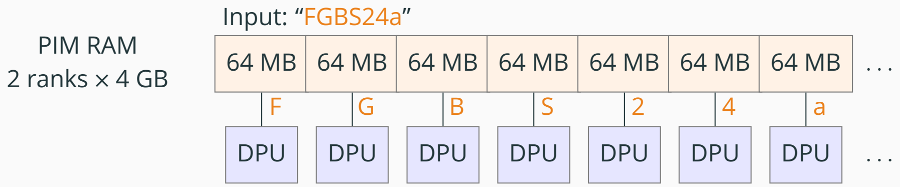
- No need to port or adjust algorithms
- No need to move data between DRAM and PIM RAM
- DPU execution overhead ≈ CPU scheduling overhead

A Real-World PIM Architecture (UPMEM)



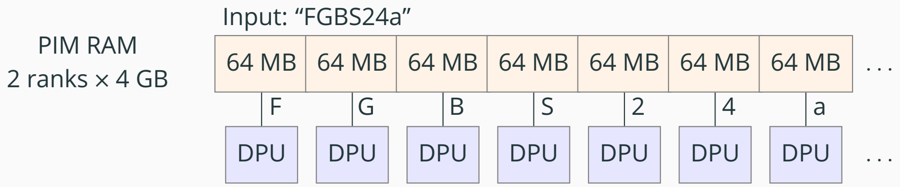
- 64MB chunk of DRAM per DPU, no shared memory
  → architecture ≠ CPU; algorithms may need adjustments

A Real-World PIM Architecture (UPMEM)



- 64MB chunk of DRAM per DPU, no shared memory
  → architecture ≠ CPU; algorithms may need adjustments
- Interleaving → PIM RAM ≠ DRAM; SDK mandatory [Dev19]
  → costly data exchange via SDK thread pool on host CPU

A Real-World PIM Architecture (UPMEM)



- 64MB chunk of DRAM per DPU, no shared memory
  → architecture ≠ CPU; algorithms may need adjustments
- Interleaving → PIM RAM ≠ DRAM; SDK mandatory [Dev19]
  → costly data exchange via SDK thread pool on host CPU

→ UPMEM PIM ≈ offloading engine; benchmarks must consider this

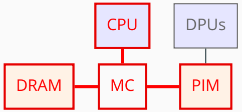
- Known challenges [Dav95; Lee+10; HB15]; lack of best practices for PIM

Challenges for a Full-System Perspective

- Goal: Identify UPMEM-suitable workloads
- Challenges for UPMEM vs. CPU benchmarks include:

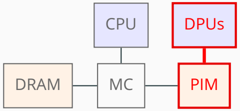
Challenges for a Full-System Perspective

- Goal: Identify UPMEM-suitable workloads
- Challenges for UPMEM vs. CPU benchmarks include:
    - Data transfer overhead



Challenges for a Full-System Perspective

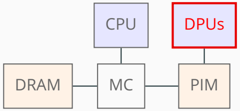
- Goal: Identify UPMEM-suitable workloads
- Challenges for UPMEM vs. CPU benchmarks include:



  - Data transfer overhead
  - Code and algorithm optimization (different architectures)

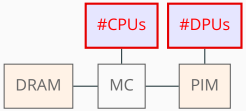
Challenges for a Full-System Perspective

- Goal: Identify UPMEM-suitable workloads
- Challenges for UPMEM vs. CPU benchmarks include:
  - Reconfiguration cost
  - Data transfer overhead
  - Code and algorithm optimization (different architectures)



Challenges for a Full-System Perspective

- Goal: Identify UPMEM-suitable workloads
- Challenges for UPMEM vs. CPU benchmarks include:



  - Reconfiguration cost
  - Data transfer overhead
  - Code and algorithm optimization (different architectures)
  - Resource allocation (1 . . . 100s of CPUs vs. 1 . . . 2560 DPUs)

Challenges for a Full-System Perspective

```
make dpus=512 tasklets=16
bin/scan-rss -i 1024
bin/scan-rss -i 3932160
make dpus=1024 tasklets=16
bin/scan-rss -i 1024
bin/scan-rss -i 3932160
```

- Goal: Identify UPMEM-suitable workloads
- Challenges for UPMEM vs. CPU benchmarks include:
  - Reproducible and adjustable benchmarks
  - Reconfiguration cost
  - Data transfer overhead
  - Code and algorithm optimization (different architectures)
  - Resource allocation (1 . . . 100s of CPUs vs. 1 . . . 2560 DPUs)

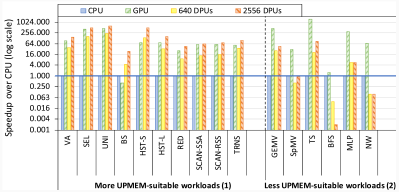
Challenges for a Full-System Perspective

- Goal: Identify UPMEM-suitable workloads
- Challenges for UPMEM vs. CPU benchmarks include:
  - Reproducible and adjustable benchmarks
  - Reconfiguration cost
  - Data transfer overhead
  - Code and algorithm optimization (different architectures)
  - Resource allocation (1 . . . 100s of CPUs vs. 1 . . . 2560 DPUs)

Outline

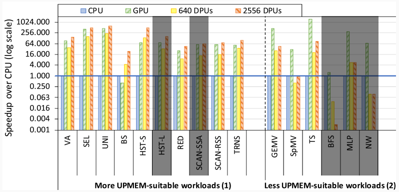
Foundation: the PrIM Benchmark Suite

- PrIM: comprehensive set of benchmark applications [Góm+22]
- CPU and UPMEM implementations, including
  - Database queries
  - Time series analysis
  - Image, matrix, vector processing



Foundation: the PrIM Benchmark Suite

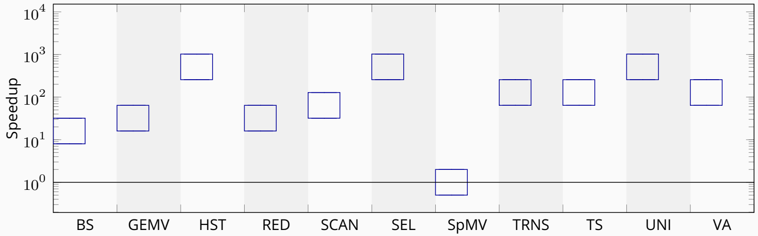
- PrIM: comprehensive set of benchmark applications [Góm+22]
- CPU and UPMEM implementations, including
  - Database queries
  - Time series analysis
  - Image, matrix, vector processing



- Source code and data sets (mostly) available
  - Suitable foundation for full-system performance perspective
  - Selection: 8 “UPMEM-suitable” + 3 “less suitable” (but promising) workloads

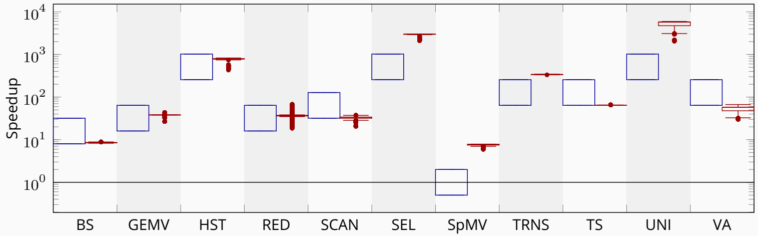
→ 1st Workshop on Disruptive Memory Systems (DIMES’23) [FLS23]

Reproduction



PIM (kernel only) on 2556 DPUs over four threads on Xeon E3-1240 @ 3.3 GHz (PrIM)

Reproduction

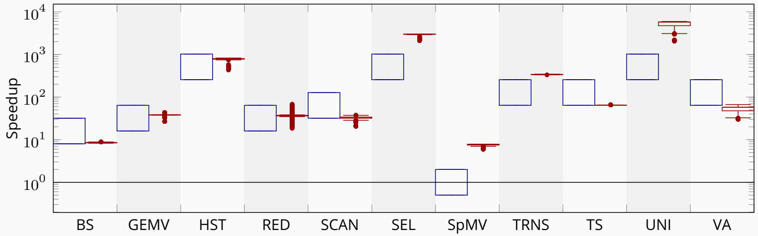


PIM (kernel only) on 2556 DPUs over four threads on Xeon E3-1240 @ 3.3 GHz (PrIM)

PIM (kernel only) on 2304 DPUs over four threads on Xeon Silver 4215 @ 2.5 GHz (ESS)

- 8/11 speedup results reproduced; 3 have higher speedup than PrIM

Reproduction



PIM (kernel only) on 2556 DPUs over four threads on Xeon E3-1240 @ 3.3 GHz (PrIM)
PIM (kernel only) on 2304 DPUs over four threads on Xeon Silver 4215 @ 2.5 GHz (ESS)

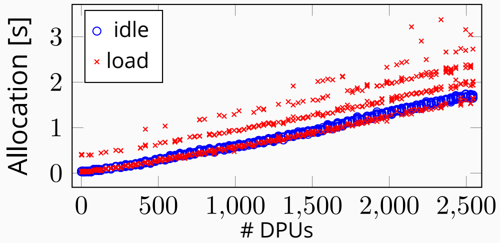
- 8/11 speedup results reproduced; 3 have higher speedup than PrIM
- 10/11 weak scaling results (1 . . . 64 DPUs, no CPU) reproduced

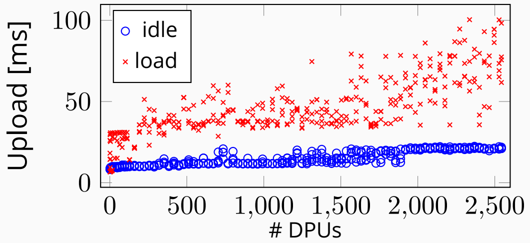
Reconfiguration Overhead

- UPMEM: allocate set of DPUs and upload application
- Key attribute in FPGA offloading research [Sch+20]
- Not considered in PIM/UPMEM benchmarks

Reconfiguration Overhead

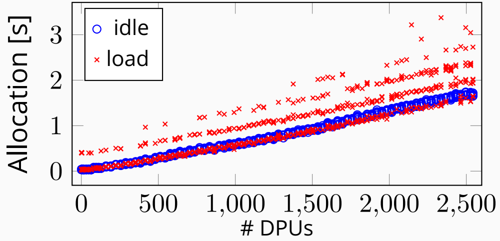
- UPMEM: allocate set of DPUs and upload application
- Key attribute in FPGA offloading research [Sch+20]
- Not considered in PIM/UPMEM benchmarks

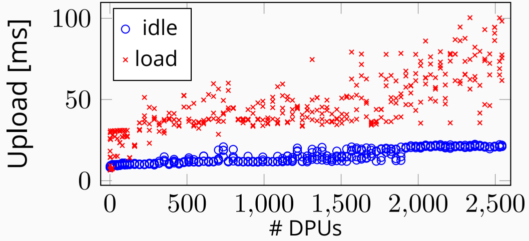




Reconfiguration Overhead

- UPMEM: allocate set of DPUs and upload application
- Key attribute in FPGA offloading research [Sch+20]
- Not considered in PIM/UPMEM benchmarks

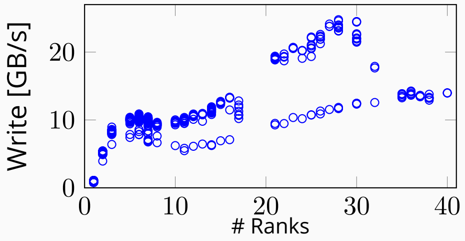


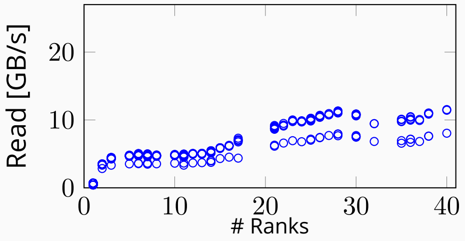


- Allocation can take seconds → short workloads infeasible with current SDK

Data Transfer Overhead

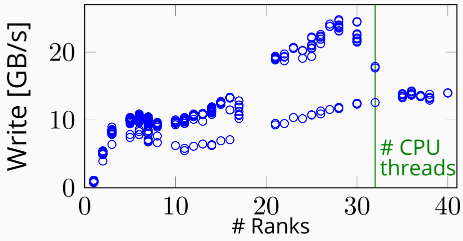
- UPMEM: explicit data transfers via SDK (with internal thread pool)

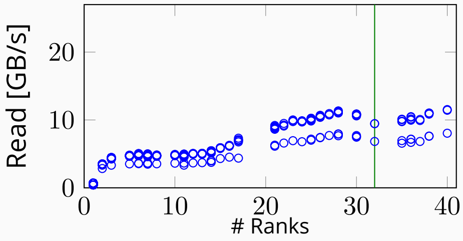




Data Transfer Overhead

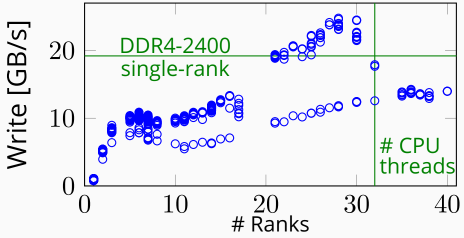
- UPMEM: explicit data transfers via SDK (with internal thread pool)

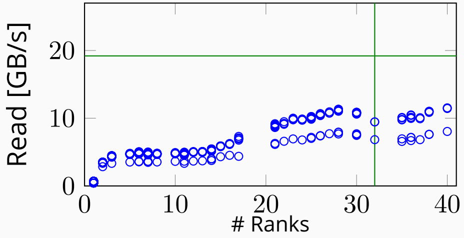




Data Transfer Overhead

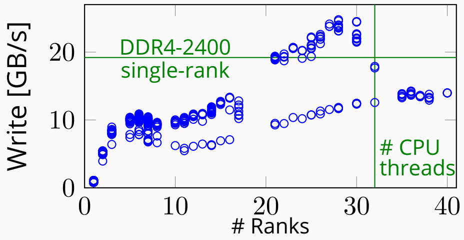
- UPMEM: explicit data transfers via SDK (with internal thread pool)

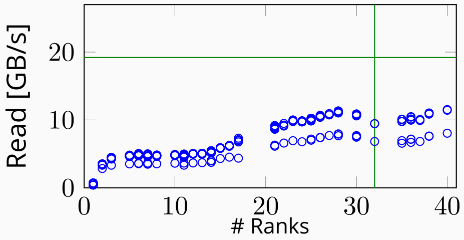




Data Transfer Overhead

- UPMEM: explicit data transfers via SDK (with internal thread pool)



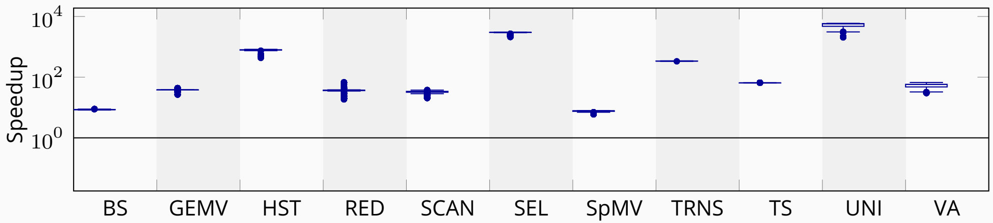


- Base latency in milliseconds range [Góm+22]
  - DRAM → MRAM on UPMEM module (write): 5 . . . 90 ms
  - DRAM ← MRAM on UPMEM module (read): 6 . . . 270 ms

Data Transfer in Benchmarks

- UPMEM mandates DRAM ↔ PIM RAM transfers
- “Amortized overhead”: weak argument (IRAM holds ≤ 4,096 instructions)

$2^{11} + 2^{8}$ = 2304 DPUs vs. 4 threads on Xeon Silver 4215

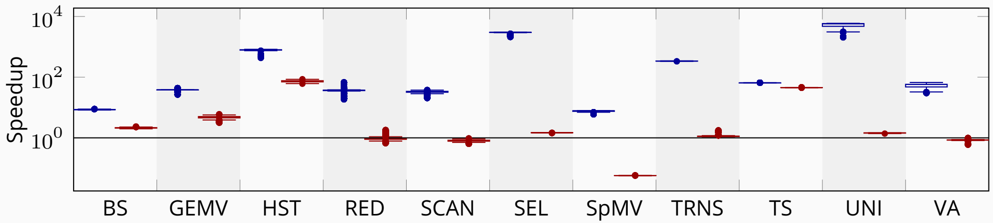


PIM (kernel only) over CPU (≙ PrIM)

Data Transfer in Benchmarks

- UPMEM mandates DRAM ↔ PIM RAM transfers
- “Amortized overhead”: weak argument (IRAM holds ≤ 4,096 instructions)

$2^{11} + 2^{8}$ = 2304 DPUs vs. 4 threads on Xeon Silver 4215



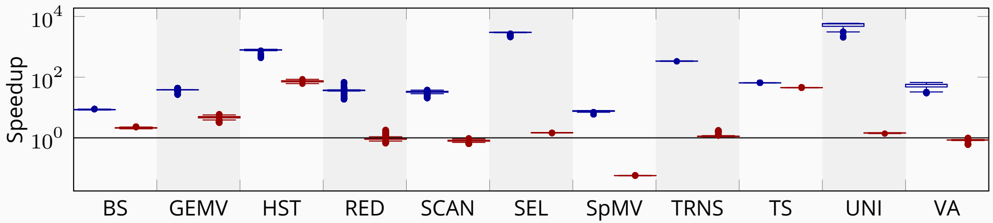
PIM (kernel only) over CPU (≙ PrIM)

PIM (with data transfer) over CPU

Data Transfer in Benchmarks

- UPMEM mandates DRAM ↔ PIM RAM transfers
- “Amortized overhead”: weak argument (IRAM holds ≤ 4,096 instructions)

$2^{11} + 2^{8}$ = 2304 DPUs vs. 4 threads on Xeon Silver 4215



PIM (kernel only) over CPU (≙ PrIM)

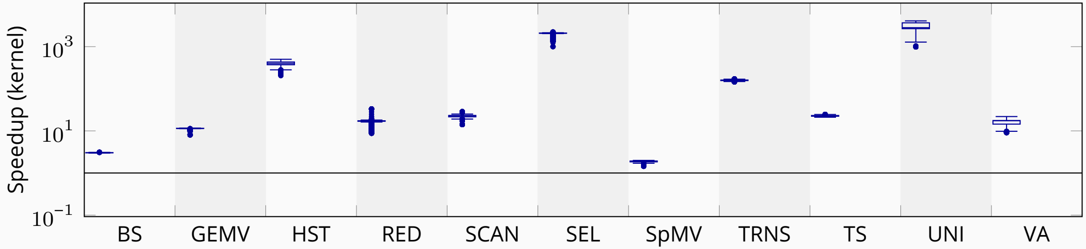
PIM (with data transfer) over CPU

→ Only 7 of 11 workloads still benefit from UPMEM PIM

Resource Allocation

- Speedup benchmarks: how many DPUs vs. how many CPU cores?

$2^{11} + 2^{8}$ = 2304 DPUs vs. Xeon Gold 6152



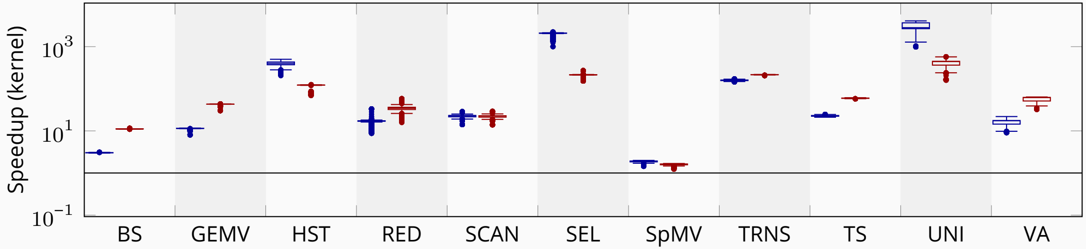
PIM over 4 CPU threads

(≘ PrIM; arbitrary)

Resource Allocation

- Speedup benchmarks: how many DPUs vs. how many CPU cores?

$2^{11} + 2^{8}$ = 2304 DPUs vs. Xeon Gold 6152



PIM over 4 CPU threads

(≙ PrIM; arbitrary)

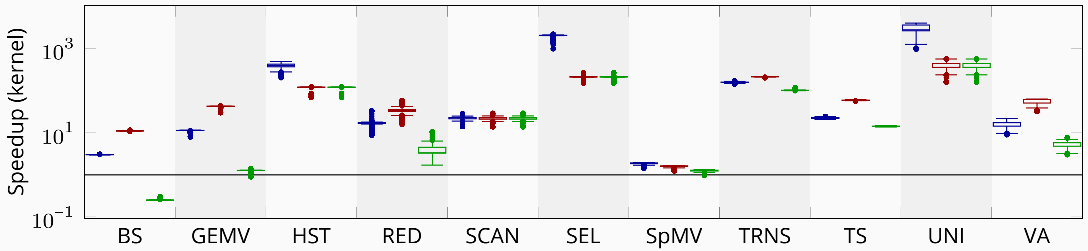
PIM over 1 CPU thread

(one core for housekeeping)

Resource Allocation

- Speedup benchmarks: how many DPUs vs. how many CPU cores?

$2^{11} + 2^8$ = 2304 DPUs vs. Xeon Gold 6152



PIM over 4 CPU threads

(≙ PrIM; arbitrary)

PIM over 1 CPU thread

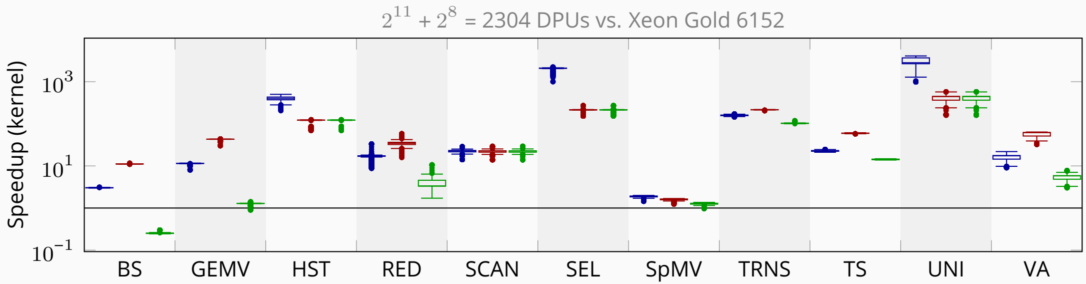
(one core for housekeeping)

PIM over best (≤ 88 threads)

(UPMEM cost ≈ 2× CPU-only)

Resource Allocation

- Speedup benchmarks: how many DPUs vs. how many CPU cores?



PIM over 4 CPU threads
(≙ PrIM; arbitrary)

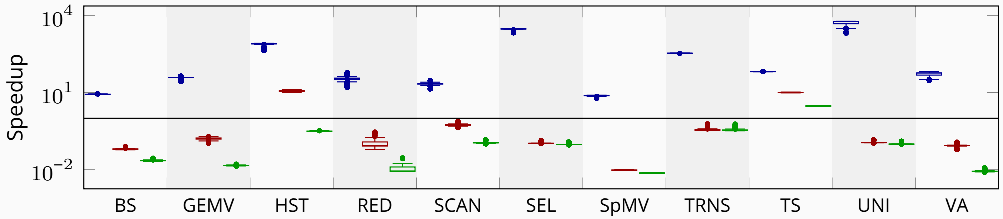
PIM over 1 CPU thread
(one core for housekeeping)

PIM over best (≤ 88 threads)
(UPMEM cost ≈ 2× CPU-only)

- Variable #DPUs vs. variable #CPUs (parallelization; sub-linear scaling)

Outline

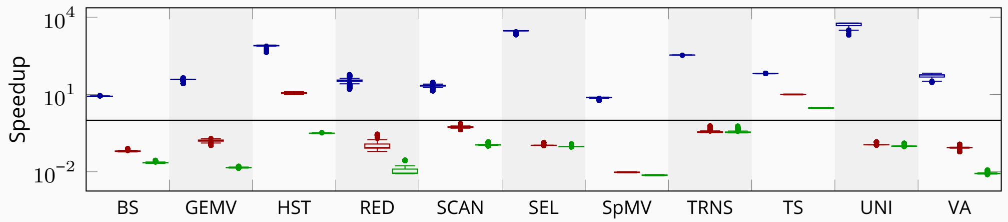
Conclusion



PIM (kernel) / 1 CPU thread PIM (+data) / best CPU PIM (+reconfiguration) / best CPU

- Assumptions matter: only **2 of 11** (**1 of 11**) individual workloads benefit

Conclusion



PIM (kernel) / 1 CPU thread   PIM (+data) / best CPU   PIM (+reconfiguration) / best CPU

- Assumptions matter: only **2 of 11** (**1 of 11**) individual workloads benefit
- Detailed benchmarks are key for evaluating novel technologies: reproducible; overhead-aware; variable resource allocation

References i

[BJS23] Alexander Baumstark, Muhammad Attahir Jibril, and Kai-Uwe Sattler. **“Accelerating Large Table Scan using Processing-In-Memory Technology”.** In: BTW 2023. Bonn: Gesellschaft für Informatik e.V., 2023, pp. 797–814. ISBN: 978-3-88579-725-8. DOI: `10.18420/BTW2023-51`.

[Cor+21] Stefano Corda et al. **“NMPO: Near-Memory Computing Profiling and Offloading”.** In: 2021 24th Euromicro Conference on Digital System Design (DSD). 2021, pp. 259–267. DOI: `10.1109/DSD53832.2021.00048`.

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

References ii

[Dev19] Fabrice Devaux. **“The true Processing In Memory accelerator”.** In: 2019 IEEE Hot Chips 31 Symposium (HCS). 2019, pp. 1–24. DOI: `10.1109/HOTCHIPS.2019.8875680`.

[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`.

[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 iii

[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.

References iv

[Sch+20] Robert Schmid et al. **“Accessible Near-Storage Computing with FPGAs”.** In: *Proceedings of the Fifteenth European Conference on Computer Systems*. EuroSys ’20. Heraklion, Greece: Association for Computing Machinery, 2020. ISBN: 9781450368827. DOI: `10.1145/3342195.3387557`. URL: `https://doi.org/10.1145/3342195.3387557`.