OSNABRÜCK UNIVERSITY

Embedded Software Systems

Project Report and Outlook: SMAUG

System-Level Modelling and Optimized Use of Disruptive Memory Technologies

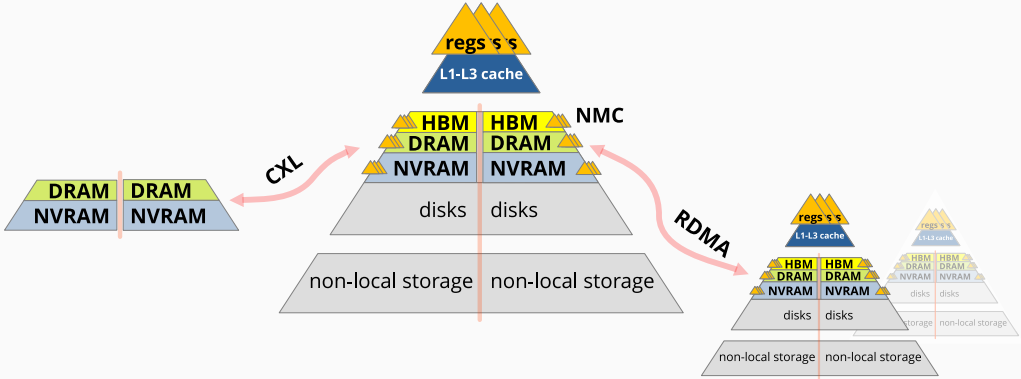
Birte Friesel, Michael Müller, Olaf Spinczyk

birte.friesel@uos.de

September 12th, 2025

Universität Osnabrück

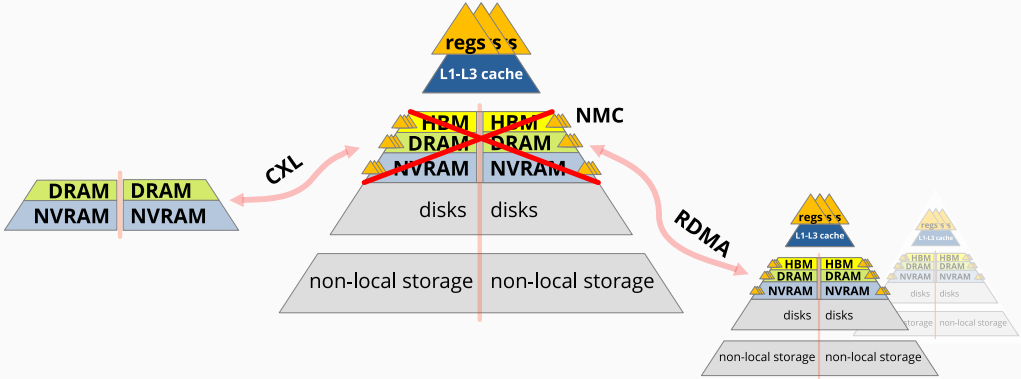
Motivation



Memory hierarchies are becoming increasingly complex

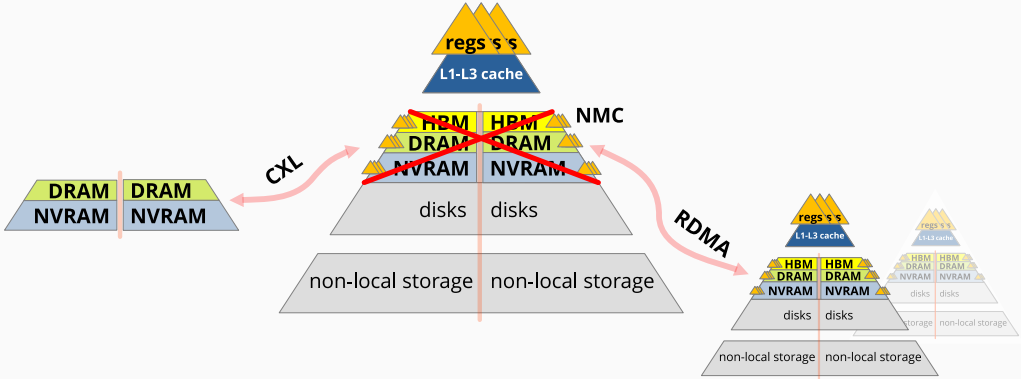
System software needs HW/SW models for resource management

Motivation



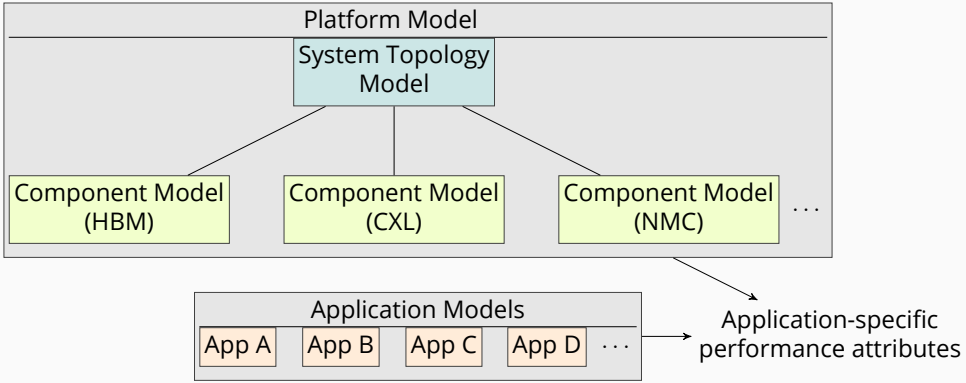
Disruptive memory technologies break out of the hierarchy:
Setup/transfer costs and access patterns must be considered

Motivation



System-Level Modelling and Optimized Use of DMTs

Motivation

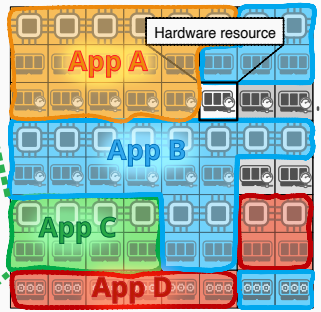


First funding period (Mar 2023 – Feb 2026):
System-Level Modelling and Optimized Use of DMTs

Motivation

Cells

- **Elastic** resource domain **for a single process**
- **Managed** by **MxVisor, directly**
- Use **task annotations** and resource metrics (e.g.,performance counters)
- To enable **fast reaction** to **load**



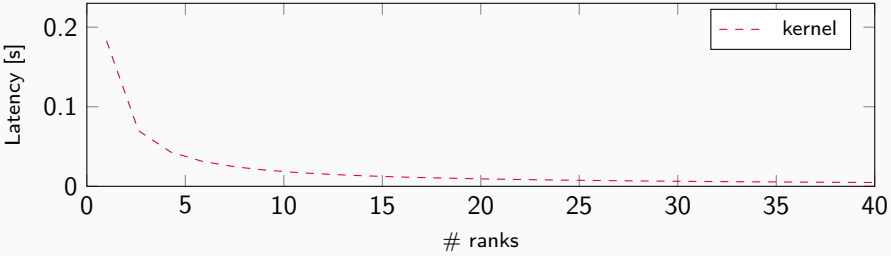
Second funding period (Mar 2026 – Feb 2029):
System-Level Modelling and Optimized Use of DMTs

Contents

Component Model: NMC (UPMEM PIM)

- UPMEM PIM: Near-Memory Computing ≙ offloading engine [Góm+22]
  - Up to 2560 cores: ideal for “embarrassingly parallel” problems

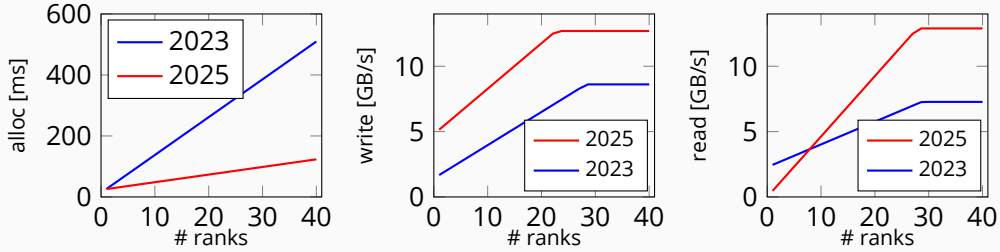
Component Model: NMC (UPMEM PIM)



- UPMEM PIM: Near-Memory Computing ≙ offloading engine [Góm+22]
  - Up to 2560 cores: ideal for “embarrassingly parallel” problems

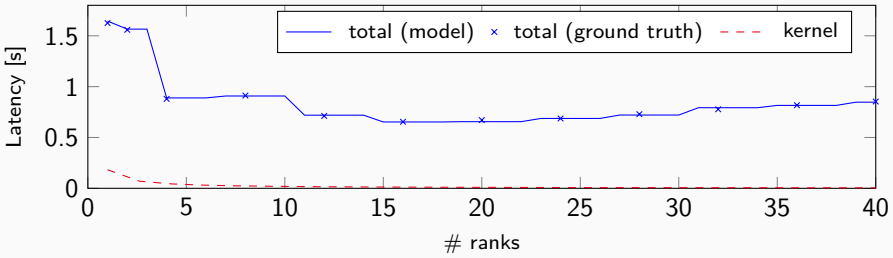
DBMS SELECT kernel latency: $237\,\mu s + 0.68\,ns \cdot \frac{\#rows}{\#ranks}$

Component Model: NMC (UPMEM PIM)



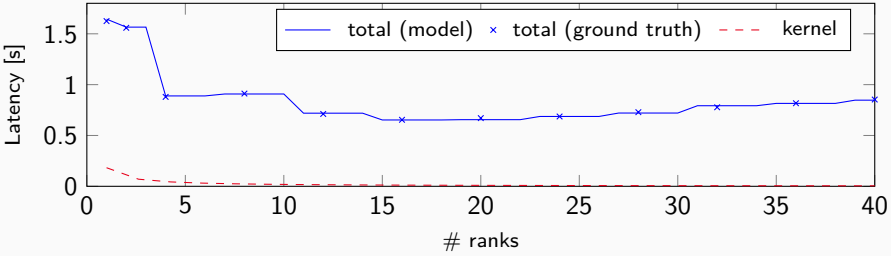
- UPMEM PIM: Near-Memory Computing ≙ offloading engine [Góm+22]
  - Up to 2560 cores: ideal for “embarrassingly parallel” problems
  - Setup and data transfer costs

Component Model: NMC (UPMEM PIM)



- UPMEM PIM: Near-Memory Computing ≙ offloading engine [Góm+22]
  - Up to 2560 cores: ideal for “embarrassingly parallel” problems
  - Setup and data transfer costs jeopardize kernel speedup [FLS23; FS25a]

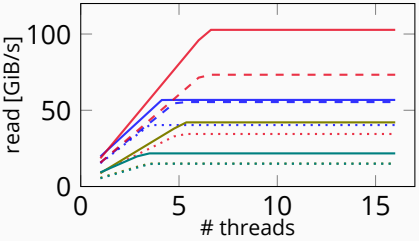
Component Model: NMC (UPMEM PIM)



- UPMEM PIM: Near-Memory Computing ≙ offloading engine [Góm+22]
  - Up to 2560 cores: ideal for “embarrassingly parallel” problems
  - Setup and data transfer costs jeopardize kernel speedup [FLS23; FS25a]

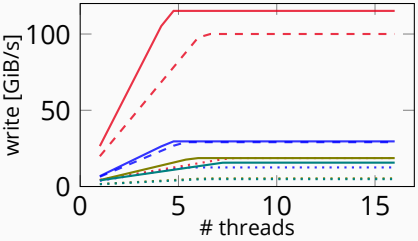
→ System-level models enable optimal resource utilization [FLS24; FLS25]

Platform Model: HBM, CXL



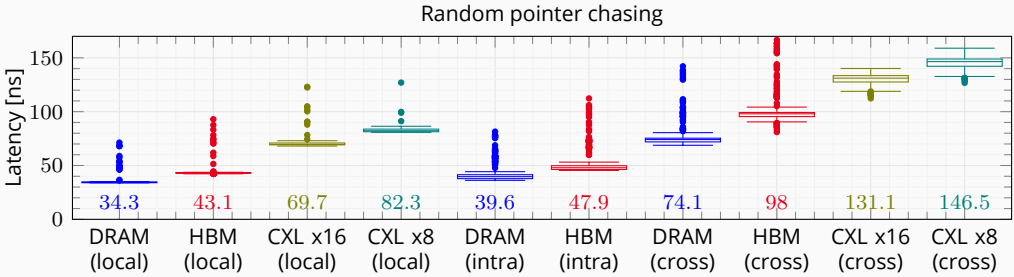
HBM
DRAM
CXL x16
CXL x8

local
intra -
cross -
pkg



- HBM: bandwidth ↑, but capacity ↓
- CXL: capacity ↑, but bandwidth ↓

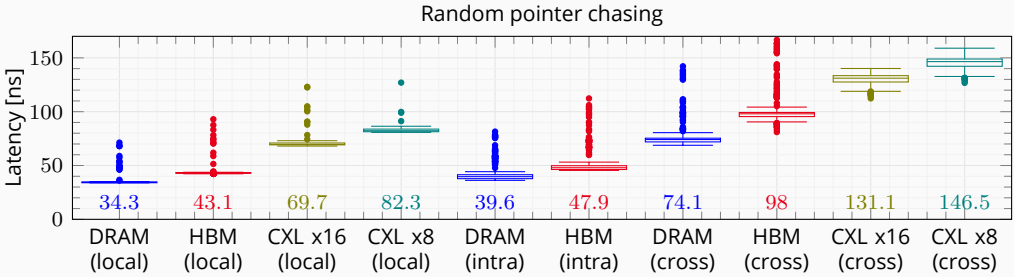
Platform Model: HBM, CXL



- HBM: bandwidth ↑, but capacity ↓, latency ↑

- CXL: capacity ↑, but bandwidth ↓, latency ↑ (worse than DRAM)

Platform Model: HBM, CXL

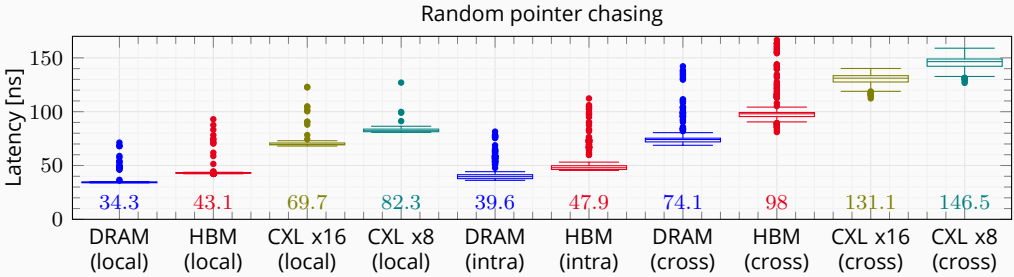


- HBM: bandwidth ↑, but capacity ↓, latency ↑

→ HBM can slow down applications; depends on access patterns [FLS24]

- CXL: capacity ↑, but bandwidth ↓, latency ↑ (worse than DRAM)

Platform Model: HBM, CXL



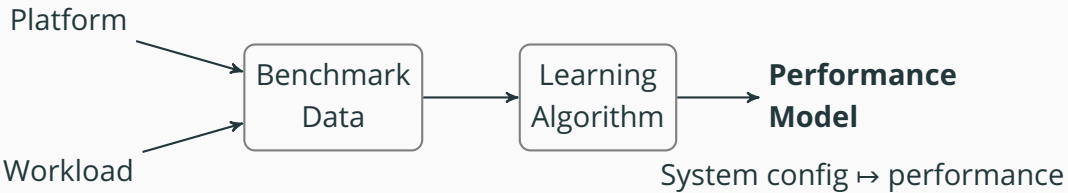
- HBM: bandwidth ↑, but capacity ↓, latency ↑

→ HBM can slow down applications; depends on access patterns [FLS24]

- CXL: capacity ↑, but bandwidth ↓, latency ↑ (worse than DRAM)
- Local CXL.mem ≋ remote DRAM → caching; contention avoidance

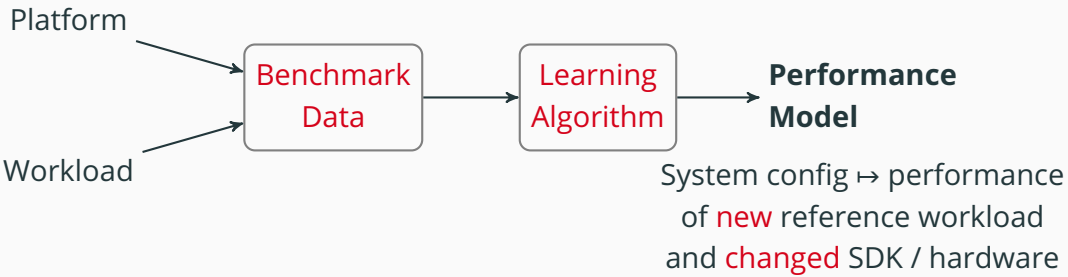
Application Models

- Existing approaches mix platform and application attributes



Application Models

- Existing approaches mix platform and application attributes
  - Changes in either → re-build model from scratch
  - New hardware, new SDK version, different workloads, . . .



Application Models

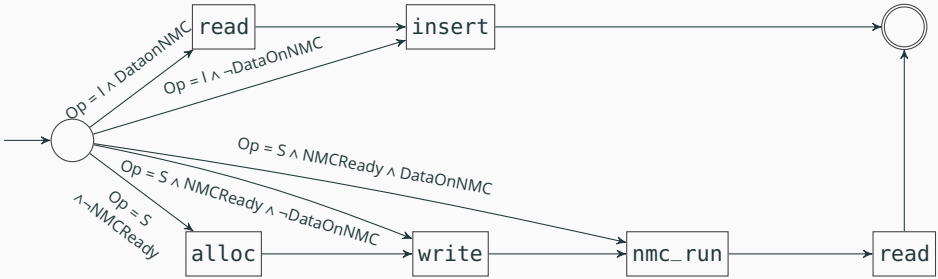
- Existing approaches mix platform and application attributes
  - Changes in either → re-build model from scratch
  - New hardware, new SDK version, different workloads, . . .
- Thus: incapable of dealing with workload changes;
  useless for compute/data placement decisions

Application Models

- Existing approaches mix platform and application attributes
  - Changes in either → re-build model from scratch
  - New hardware, new SDK version, different workloads, . . .
- Thus: incapable of dealing with workload changes;
  useless for compute/data placement decisions
- SMAUG: application model is independent of platform model
  - Decoupled from hardware or SDK performance
  - Can be learnt on simulated hardware
- Contribution: state-machine based behaviour models

Performance-Aware Behaviour Models

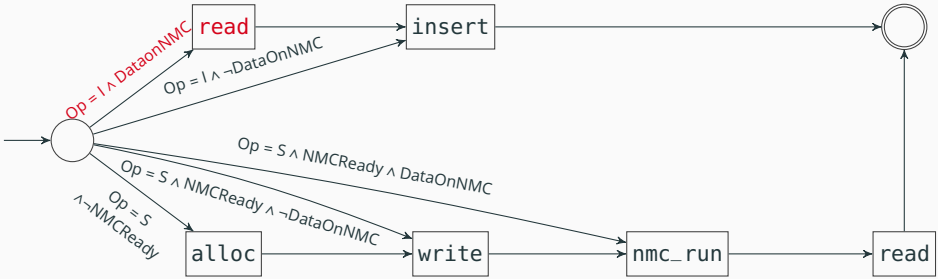
Simplified example: NMC-enabled DBMS, **I**NSERT on CPU, **S**ELECT on NMC



- States ≙ API or application kernel function calls
- Transitions guarded with workload-dependent runtime parameters

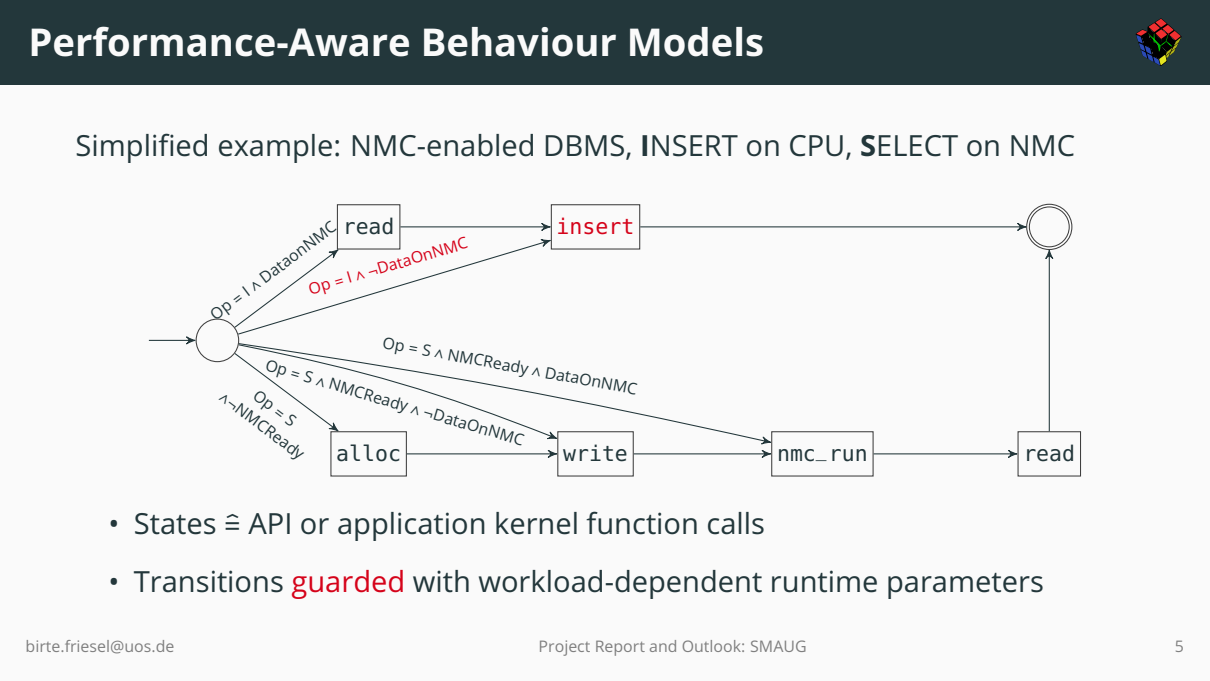
Performance-Aware Behaviour Models

Simplified example: NMC-enabled DBMS, **I**NSERT on CPU, **S**ELECT on NMC



- States ≙ API or application kernel function calls
- Transitions guarded with workload-dependent runtime parameters

Performance-Aware Behaviour Models

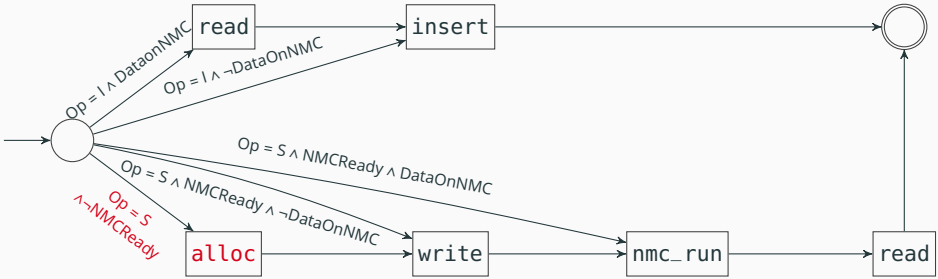


Simplified example: NMC-enabled DBMS, **I**NSERT on CPU, **S**ELECT on NMC

- States ≙ API or application kernel function calls
- Transitions guarded with workload-dependent runtime parameters

Performance-Aware Behaviour Models

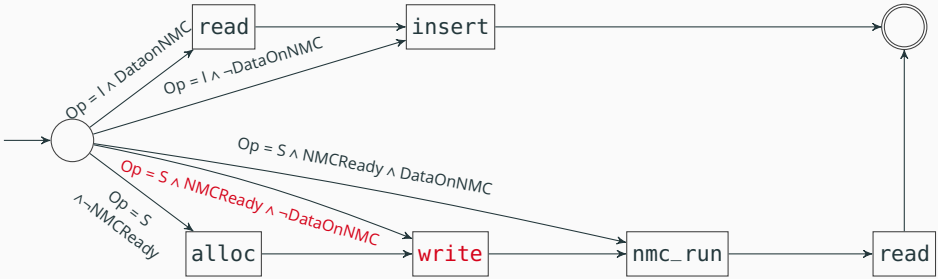
Simplified example: NMC-enabled DBMS, **I**NSERT on CPU, **S**ELECT on NMC



- States ≙ API or application kernel function calls
- Transitions guarded with workload-dependent runtime parameters

Performance-Aware Behaviour Models

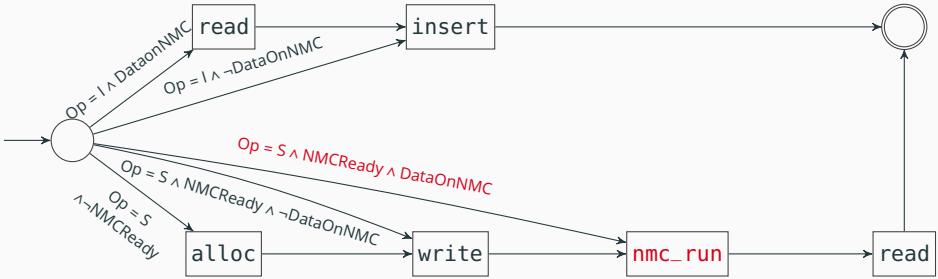
Simplified example: NMC-enabled DBMS, **I**NSERT on CPU, **S**ELECT on NMC



- States ≙ API or application kernel function calls
- Transitions guarded with workload-dependent runtime parameters

Performance-Aware Behaviour Models

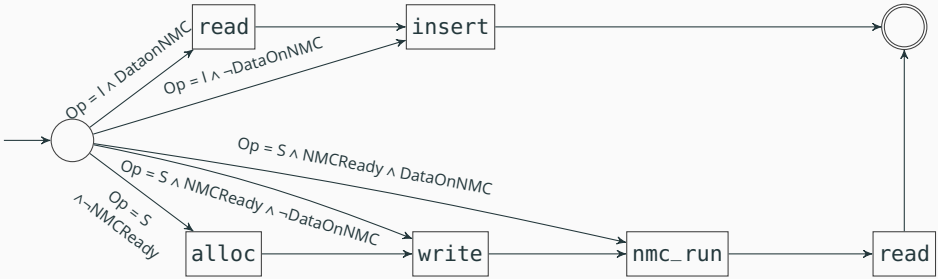
Simplified example: NMC-enabled DBMS, **I**NSERT on CPU, **S**ELECT on NMC



- States ≙ API or application kernel function calls
- Transitions guarded with workload-dependent runtime parameters

Performance-Aware Behaviour Models

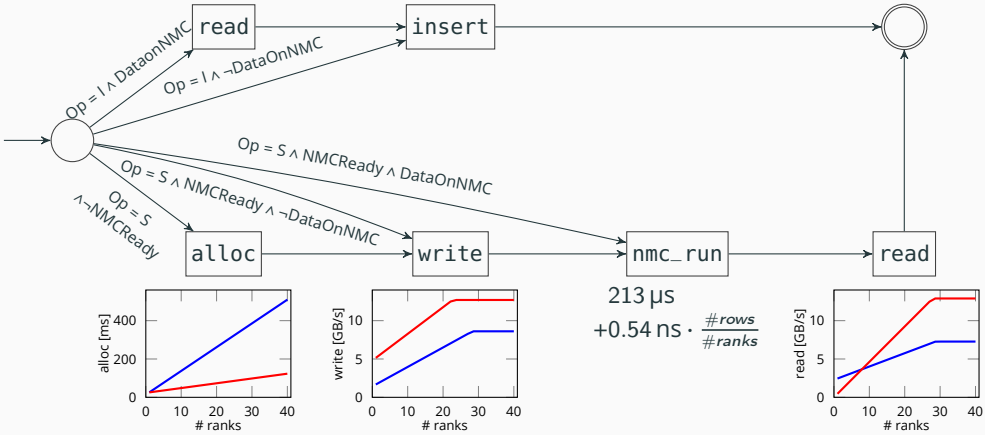
Simplified example: NMC-enabled DBMS, **I**NSERT on CPU, **S**ELECT on NMC



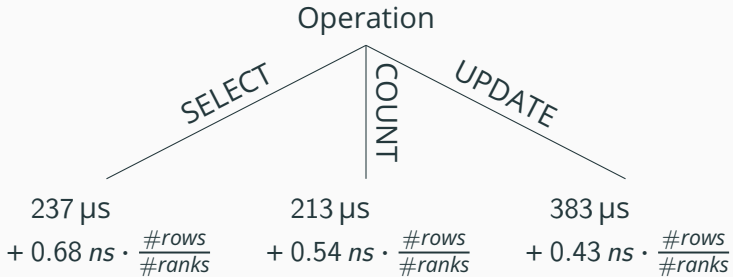
- States annotated with performance models
- Foundation: regression model trees (RMTs)

Performance-Aware Behaviour Models

Simplified example: NMC-enabled DBMS, **I**NSERT on CPU, **S**ELECT on NMC



Regression Model Trees (RMTs)

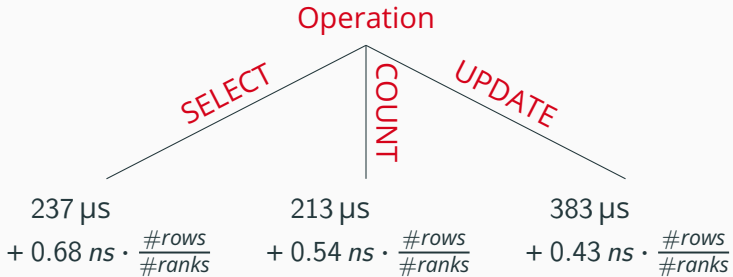


Regression model trees [FS22]

= regression trees [Bre+84]

\+ unsupervised least-squares [FBS18]

Regression Model Trees (RMTs)

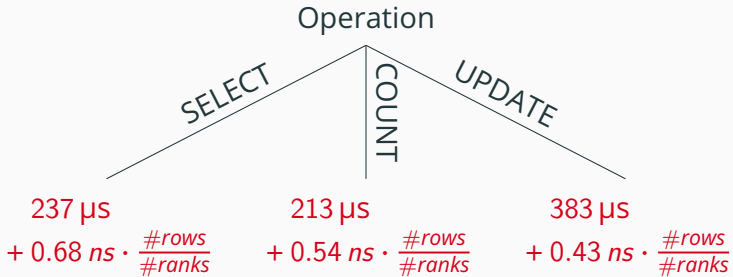


Regression model trees [FS22]

= regression trees [Bre+84]

\+ unsupervised least-squares [FBS18]

Regression Model Trees (RMTs)

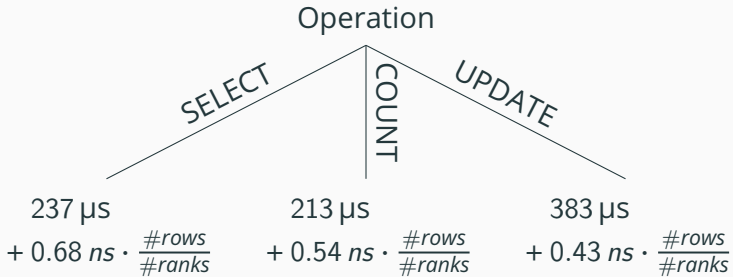


Regression model trees [FS22]

= regression trees [Bre+84]

\+ unsupervised least-squares [FBS18]

Regression Model Trees (RMTs)



Regression model trees [FS22]

= regression trees [Bre+84]

+ unsupervised least-squares [FBS18]

- Accurate and interpretable
- Input domain: arbitrary
- Output domain: $\mathbb{R}$

Application Models: Learning and Usage

- Performance prediction and model learning in two steps:
  1. Workload definition $\mapsto$ function call sequence (with function arguments)
     Example: Op = I, #rows = 2^{30}, DataOnNMC = $\top \mapsto$ `read`(8 GiB) $\cdot$ `insert`()
  2. For each function call: arguments $\mapsto$ latency/throughput
     Example: `read`(8 GiB) with #ranks = 40 $\mapsto$ 12.5 GiB/s

Application Models: Learning and Usage

- Performance prediction and model learning in two steps:
  1. Workload definition ↦ function call sequence (with function arguments)
     (state machine learnt from application traces; simulator is sufficient)
  2. For each function call: arguments ↦ latency/throughput
     (learnt from microbenchmarks or traces on real hardware)

Application Models: Learning and Usage

- Performance prediction and model learning in two steps:
  1. Workload definition ↦ function call sequence (with function arguments)
     (state machine learnt from application traces; simulator is sufficient)
  2. For each function call: arguments ↦ latency/throughput
     (learnt from microbenchmarks or traces on real hardware)

→ Predict latency/throughput of arbitrary workloads [FS25b; FS25a]

→ Understand bottlenecks and changes, e.g., between SDK releases [FS25c]

→ Simulation-based optimization of placement decisions [FLS25]

Application Models: Learning and Usage

- Performance prediction and model learning in two steps:
  1. Workload definition ↦ function call sequence (with function arguments)
     (state machine learnt from application traces; simulator is sufficient)
  2. For each function call: arguments ↦ latency/throughput
     (learnt from microbenchmarks or traces on real hardware)

→ Predict latency/throughput of arbitrary workloads [FS25b; FS25a]

→ Understand bottlenecks and changes, e.g., between SDK releases [FS25c]

→ Simulation-based optimization of placement decisions [FLS25]

- Compatible with HetSim (DMT-aware scheduling simulator) [LFS24]
- Both prediction steps use regression model trees (RMTs) [FS22]

Summary of First Funding Period

- 30/36 months elapsed (started in March, 2023)

(✓) Platform model: NMC (UPMEM PIM), HBM, CXL.mem (legacy mode)

† NVRAM (Intel Optane)

Summary of First Funding Period

- 30/36 months elapsed (started in March, 2023)

(✓) Platform model: NMC (UPMEM PIM), HBM, CXL.mem (legacy mode)

† NVRAM (Intel Optane)

(✓) Application models: concept, learning algorithm, evaluation

Summary of First Funding Period

- 30/36 months elapsed (started in March, 2023)

(✓) Platform model: NMC (UPMEM PIM), HBM, CXL.mem (legacy mode)

† NVRAM (Intel Optane)

(✓) Application models: concept, learning algorithm, evaluation

✓ Using models for optimal compute/data placement decisions
(proof of concept; in-depth research in second funding period)

Summary of First Funding Period

- 30/36 months elapsed (started in March, 2023)

(✓) Platform model: NMC (UPMEM PIM), HBM, CXL.mem (legacy mode)

† NVRAM (Intel Optane)

(✓) Application models: concept, learning algorithm, evaluation

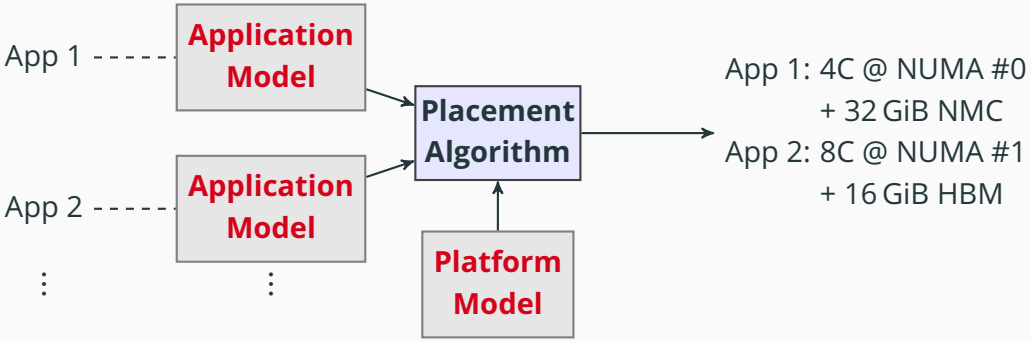
✓ Using models for optimal compute/data placement decisions
(proof of concept; in-depth research in second funding period)

- Findings published at DIMES (2×), WOSP-C, VaMoS, NoDMC, SPLC, CCMCC
  [FLS23; LFS24; FLS24; FS25b; FLS25; FS25c; FS25a]
  - Artifacts for all publications available at `ess.cs.uos.de/git/artifacts`
  - SPLC: artifacts evaluated: available and functional

Contents

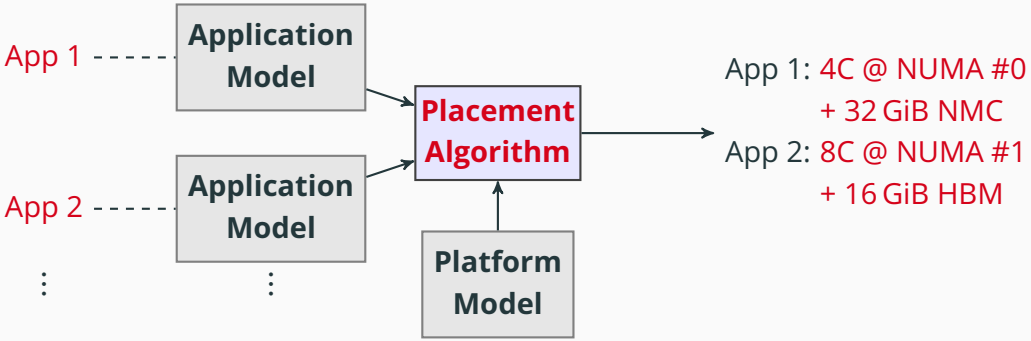
Second Funding Period: Optimized Use of DMTs

Goal: Use models from first period for holistic, model-guided strategies



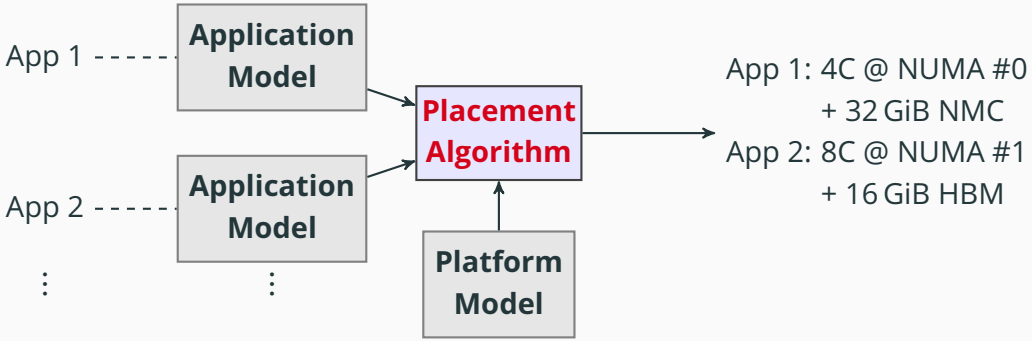
Second Funding Period: Optimized Use of DMTs

Goal: Use models from first period for holistic, model-guided strategies



Second Funding Period: Optimized Use of DMTs

Goal: Use models from first period for holistic, model-guided strategies

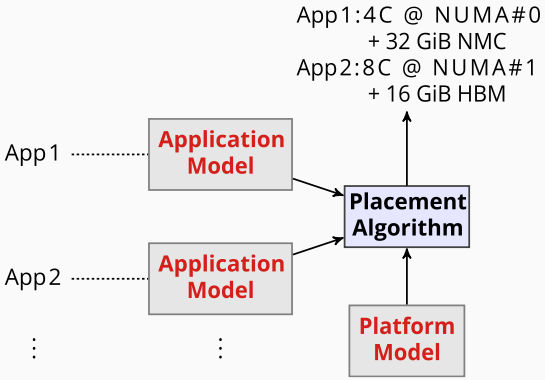


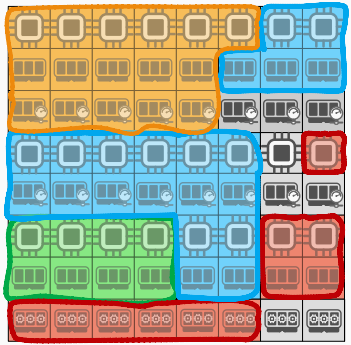
⇒ Benefit of resource management strategies for systems with DMTs?

Research Objectives

① Coarse-grained (system-wide) resource partitioning strategies

→ Avoid interference for latency-critical applications

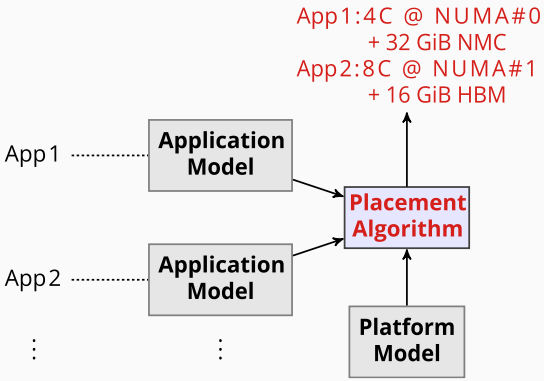


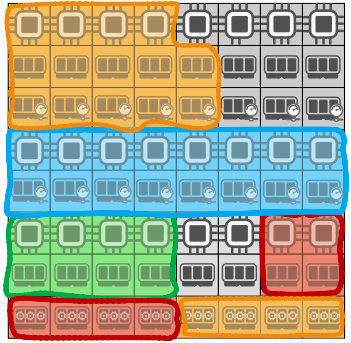


Research Objectives

① Coarse-grained (system-wide) resource partitioning strategies

→ Avoid interference for latency-critical applications

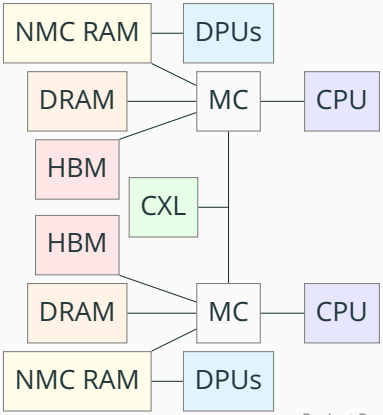




Research Objectives

② Fine-grained resource usage optimization

→ Place compute and data of given application on appropriate DMTs

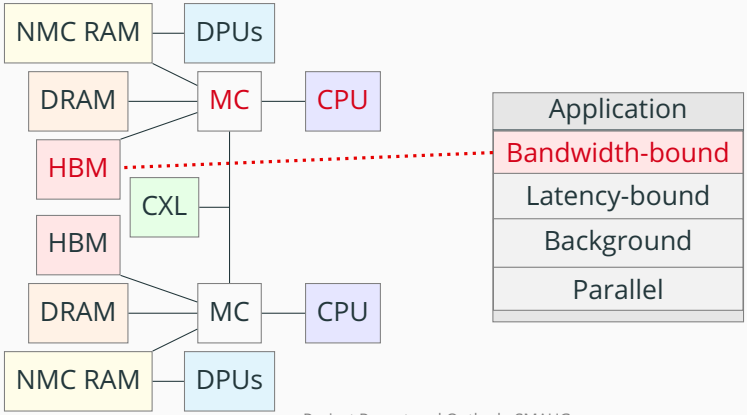


Application
Bandwidth-bound
Latency-bound
Background
Parallel

Research Objectives

② Fine-grained resource usage optimization

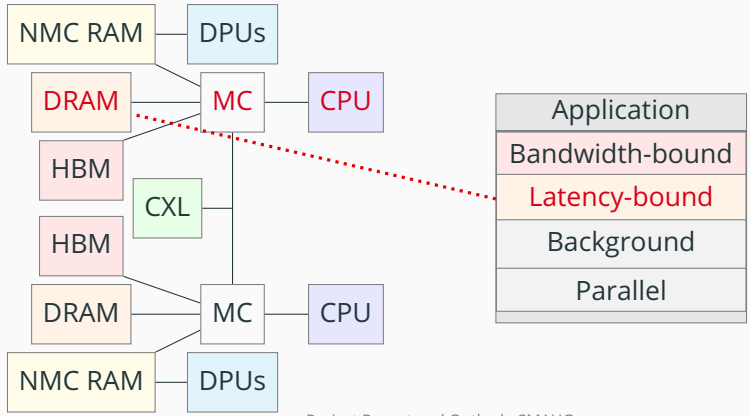
→ Place compute and data of given application on appropriate DMTs



Research Objectives

② Fine-grained resource usage optimization

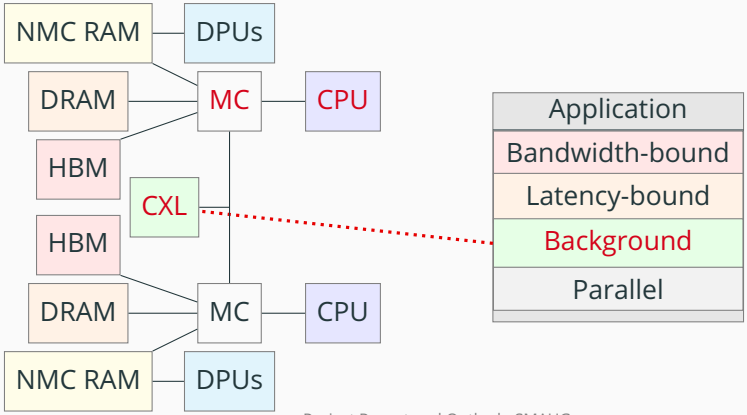
→ Place compute and data of given application on appropriate DMTs



Research Objectives

② Fine-grained resource usage optimization

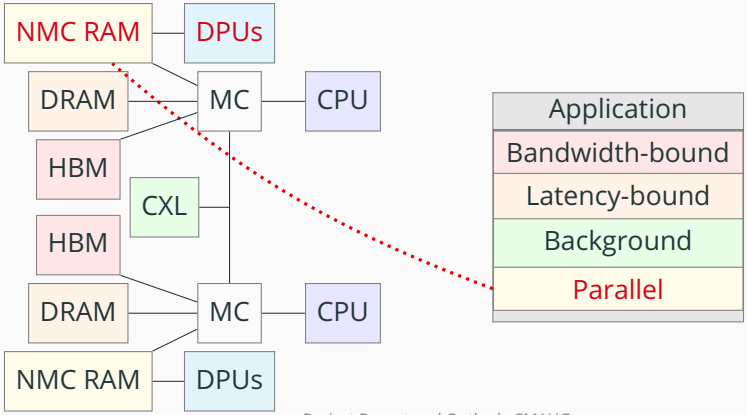
→ Place compute and data of given application on appropriate DMTs



Research Objectives

② Fine-grained resource usage optimization

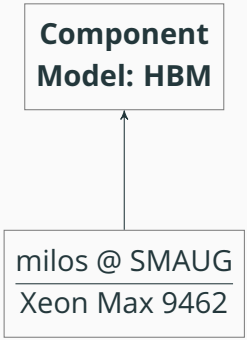
→ Place compute and data of given application on appropriate DMTs



Research Objectives

③ Model maintenance

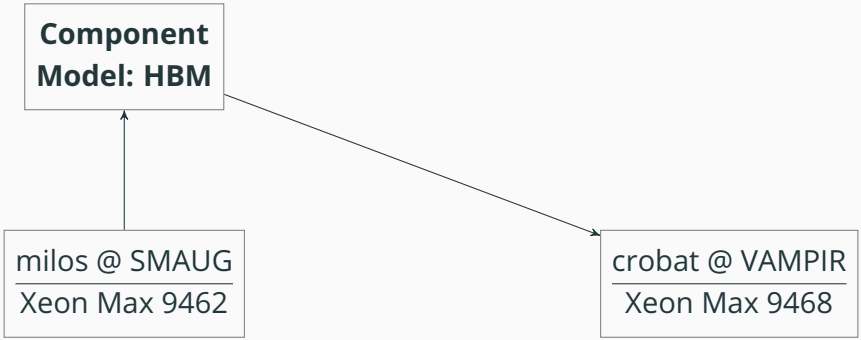
→ Avoid re-learning models from scratch after system changes



Research Objectives

③ Model maintenance

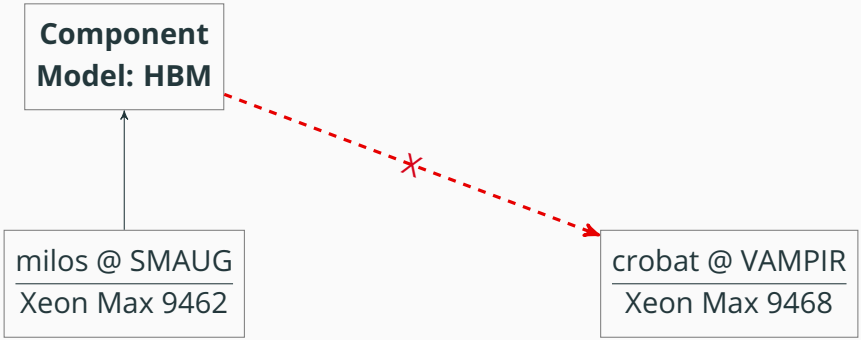
→ Avoid re-learning models from scratch after system changes



Research Objectives

③ Model maintenance

→ Avoid re-learning models from scratch after system changes



Research Objectives

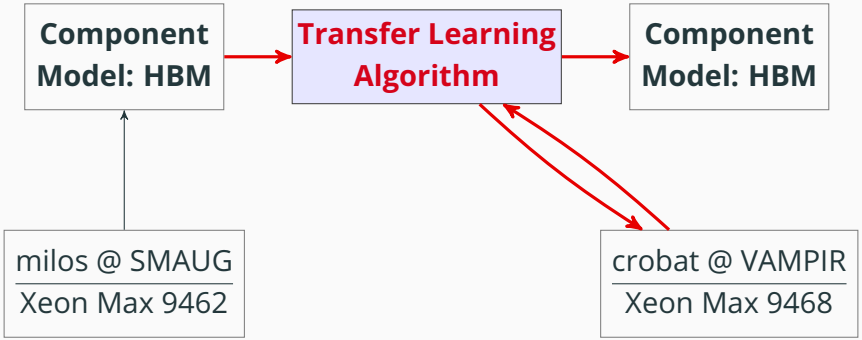
③ Model maintenance

→ Avoid re-learning models from scratch after system changes

Research Objectives

③ Model maintenance

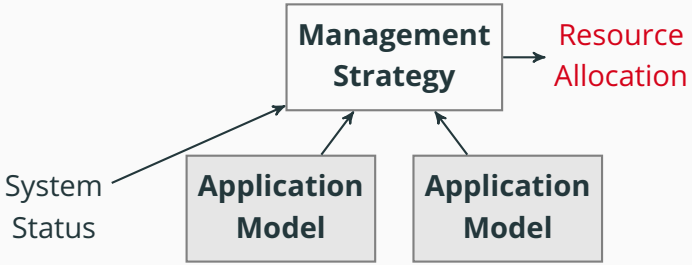
→ Avoid re-learning models from scratch after system changes [Jam+18; Zhu+21]



Research Objectives

④ Runtime efficiency

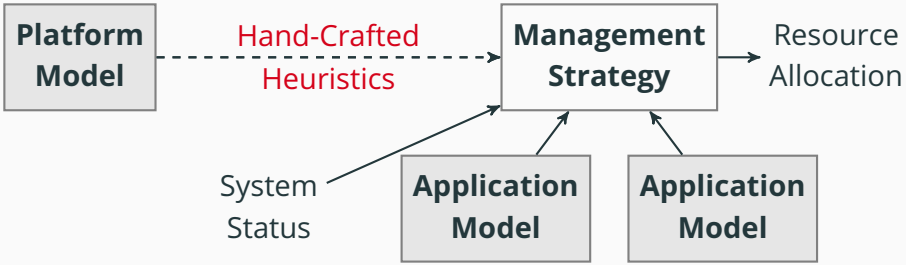
→ Microsecond-scale decisions to minimize tail latency



Research Objectives

④ Runtime efficiency

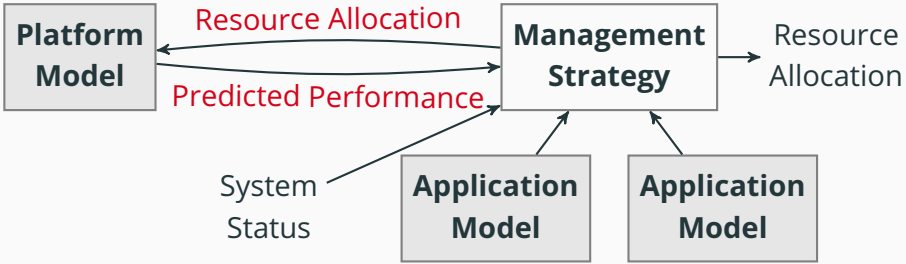
→ Microsecond-scale decisions to minimize tail latency



Research Objectives

④ Runtime efficiency

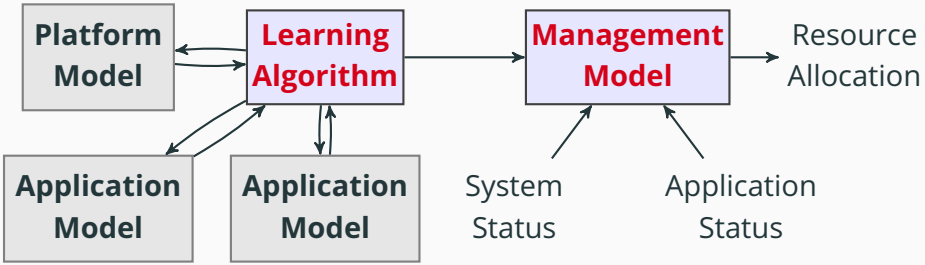
→ Microsecond-scale decisions to minimize tail latency



Research Objectives

④ Runtime efficiency

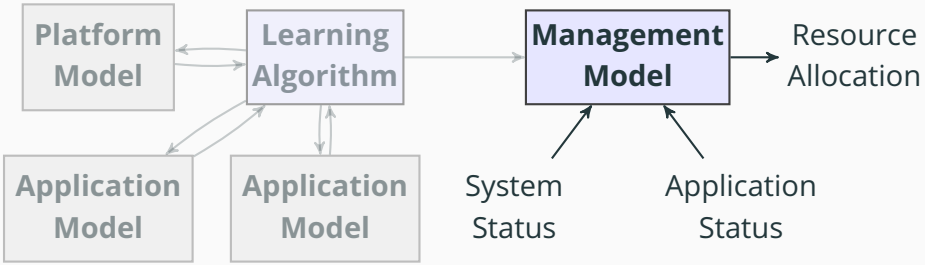
→ Microsecond-scale decisions to minimize tail latency



Research Objectives

④ Runtime efficiency

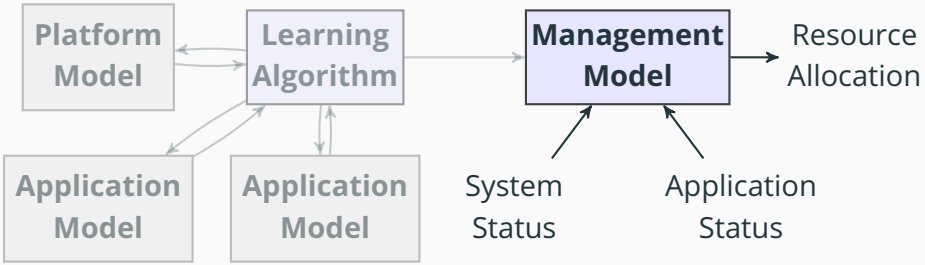
→ Microsecond-scale decisions to minimize tail latency



Research Objectives

④ Runtime efficiency

→ Microsecond-scale decisions to minimize tail latency



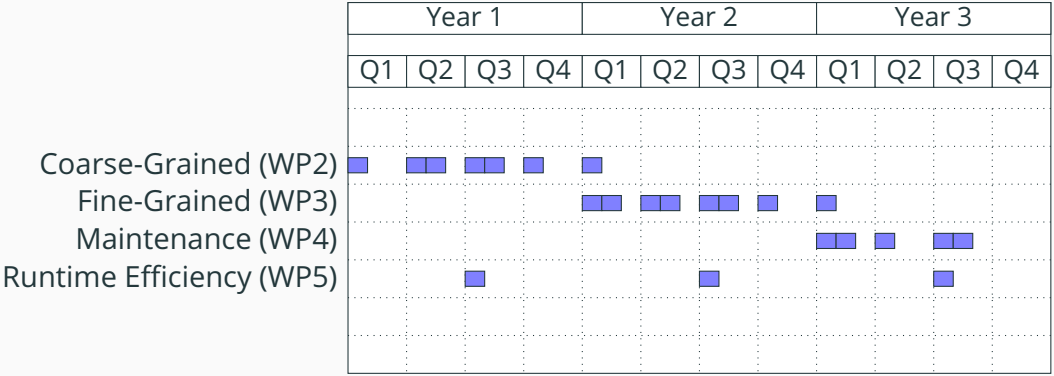
Idea: Adjust RMTs for multi-dimensional output

→ fast and interpretable management decisions

Evaluation Platforms

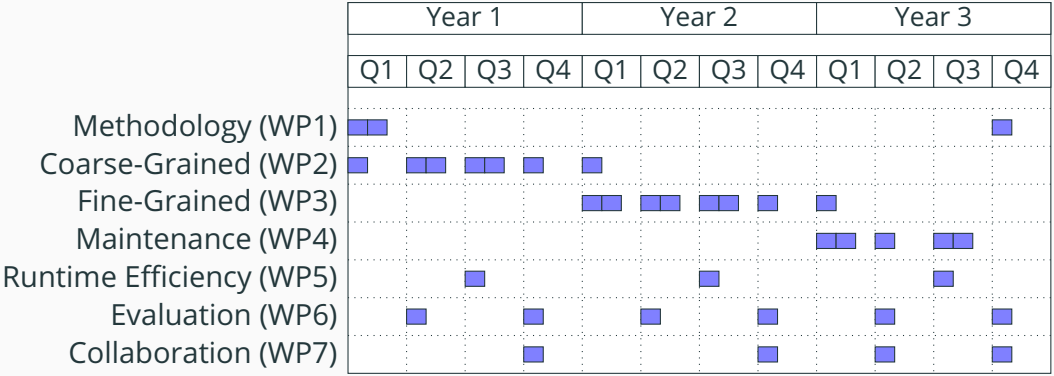
- HetSim: heterogeneous scheduling simulator [LFS24]
  - Supports CPU, GPU, FPGA, UPMEM NMC, (DRAM / HBM / CXL)
  - Already used and verified in first phase [FLS24]
- EalánOS / MxKernel: bare-metal OS for heterogeneous servers [MS19]
  - Supports CPU, GPU, and NMC execution
  - Evaluation of coarse- and fine-grained management strategies

Work Packages



WP2 . . . WP5 ≙ research objectives ① . . . ④

Work Packages



WP2 . . . WP5 ≙ research objectives ① . . . ④

Collaboration Opportunities

- HetCIM: behaviour models + kernel performance prediction [Sil+25]
- HetCIM, Memento, PIMPDM: performance-aware placement algorithms
- PIMPDM, VAMPIR: evaluation of results in DBMS context
- VAMPIR: model maintenance (slightly different HBM servers)

Collaboration Opportunities

- HetCIM: behaviour models + kernel performance prediction [Sil+25]
- HetCIM, Memento, PIMPDM: performance-aware placement algorithms
- PIMPDM, VAMPIR: evaluation of results in DBMS context
- VAMPIR: model maintenance (slightly different HBM servers)
- *Your Project Here*
  - SMAUG: Evaluation on HetSim and EalánOS
  - We are always interested in more use cases and evaluation targets

Conclusion

- Goal: examine benefits of
  - holistic, model-guided resource management strategies
  - for systems with disruptive memory technologies

Conclusion

- Goal: examine benefits of
  - holistic, model-guided resource management strategies
  - for systems with disruptive memory technologies
- Research objectives:
  - Coarse-grained partitioning and fine-grained placement
  - Model maintenance / transfer after software / hardware changes
  - Efficient (microsecond-scale) runtime decisions

Conclusion

- Goal: examine benefits of
  - holistic, model-guided resource management strategies
  - for systems with disruptive memory technologies
- Research objectives:
  - Coarse-grained partitioning and fine-grained placement
  - Model maintenance / transfer after software / hardware changes
  - Efficient (microsecond-scale) runtime decisions
- Implementation and evaluation: HetSim, EalánOS
- Lots of potential for collaboration – please reach out

References i

[Bre+84] Leo Breiman et al. **Classification and Regression Trees.** 1st ed. Routledge, 1984. ISBN: 978-1-3151-3947-0. DOI: 10.1201/9781315139470.

[FBS18] Birte Friesel, Markus Buschhoff, and Olaf Spinczyk. **"Parameter-Aware Energy Models for Embedded-System Peripherals".** In: Proceedings of the 13th International Symposium on Industrial Embedded Systems. SIES '18. Graz, Austria: IEEE, June 2018. DOI: 10.1109/SIES.2018.8442096.

References ii

[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, Oct. 2023, pp. 1–7. ISBN: 979-8-4007-0300-3. DOI: 10.1145/3609308.3625266.

[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, pp. 1–8. ISBN: 979-8-4007-1303-3. DOI: 10.1145/3698783.3699376.

References iii

[FLS25] Birte Friesel, Marcel Lütke Dreimann, and Olaf Spinczyk. **“Lightning Talk: Feasibility Analysis of Semi-Permanent Database Offloading to UPMEM Near-Memory Computing Modules”.** In: Datenbanksysteme für Business, Technologie und Web – Workshopband. BTW ’25. Bamberg, Germany: GI, Mar. 2025, pp. 355–366. DOI: 10.18420/BTW2025-140.

[FS22] Birte Friesel and Olaf Spinczyk. **“Regression Model Trees: Compact Energy Models for Complex IoT Devices”.** In: Proceedings of the Workshop on Benchmarking Cyber-Physical Systems and Internet of Things. CPS-IoTBench ’22. Milan, Italy: IEEE, May 2022, pp. 1–6. DOI: 10.1109/CPS-IoTBench56135.2022.00007.

References iv

[FS25a] Birte Friesel and Olaf Spinczyk. **“Overhead Prediction for PIM-Enabled Applications with Performance-Aware Behaviour Models”.** In: Proceedings of the 1st IEEE Cross-disciplinary Conference on Memory-Centric Computing. CCMCC ’25. to appear. Dresden, Germany: IEEE, Oct. 2025.

[FS25b] Birte Friesel and Olaf Spinczyk. **“Performance-Aware Behaviour Models for Feature-Dependent Runtime Attributes in Product Lines”.** In: Proceedings of the 19th International Working Conference on Variability Modelling of Software-Intensive Systems. VaMoS ’25. Rennes, France: ACM, Feb. 2025, pp. 131–135. ISBN: 9798400714412. DOI: 10.1145/3715340.3715435.

References v

[FS25c] Birte Friesel and Olaf Spinczyk. **“Understanding Product Line Runtime Performance with Behaviour Models and Regression Model Trees”.** In: Proceedings of the 29th ACM International Systems and Software Product Line Conference - Volume A. SPLC-A ’25. A Coruña, Spain: ACM, Sept. 2025. DOI: 10.1145/3744915.3748472.

[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 vi

[Jam+18] Pooyan Jamshidi et al. **“Learning to Sample: Exploiting Similarities across Environments to Learn Performance Models for Configurable Systems”.** In: Proceedings of the 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ESEC/FSE ’18. Lake Buena Vista, FL, USA: Association for Computing Machinery, Nov. 2018, pp. 71–82. ISBN: 978-1-4503-5573-5. DOI: 10.1145/3236024.3236074.

[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.

References vii

[MS19] Michael Müller and Olaf Spinczyk. **“MxKernel: Rethinking Operating System Architecture for Many-core Hardware”.** In: 9th Workshop on Systems for Multi-core and Heterogenous Architectures. Dresden, Germany, Mar. 2019.

[Sil+25] Anderson Faustino da Silva et al. **“LearnCNM2Predict: Transfer Learning-based Performance Model for CNM Systems”.** In: Proceedings of the 25st IEEE International Conference on Embedded Computer Systems: Architectures Modeling and Simulation. SAMOS ’25. IEEE. Berlin, Heidelberg: Springer-Verlag, July 2025.

[Zhu+21] Fuzhen Zhuang et al. **“A Comprehensive Survey on Transfer Learning”.** In: Proceedings of the IEEE 109.1 (2021), pp. 43–76. DOI: 10.1109/JPROC.2020.3004555.