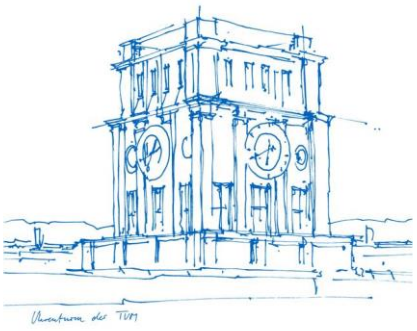
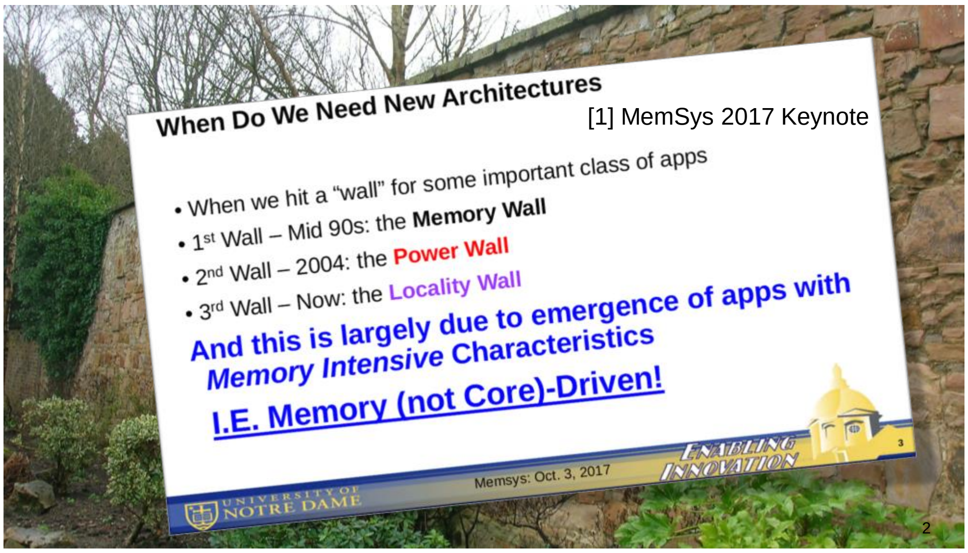
Near Memory Accelerators for Efficient Inter-Tile Communication in Distributed-Shared-Memory Architectures

Andreas Herkersdorf, Sven Rheindt,
Akshay Srivatsa, Thomas Wild
Technical University of Munich

MPSoC Forum 2019



Big Picture: Hitting yet another wall!



[1] MemSys 2017 Keynote

When Do We Need New Architectures

- When we hit a "wall" for some important class of apps
- 1st Wall – Mid 90s: the **Memory Wall**
- 2nd Wall – 2004: the **Power Wall**
- 3rd Wall – Now: the **Locality Wall**

And this is largely due to emergence of apps with *Memory Intensive* Characteristics

I.E. Memory (not Core)-Driven!

Memsys: Oct. 3, 2017

UNIVERSITY OF NOTRE DAME
ENABLING INNOVATION
3

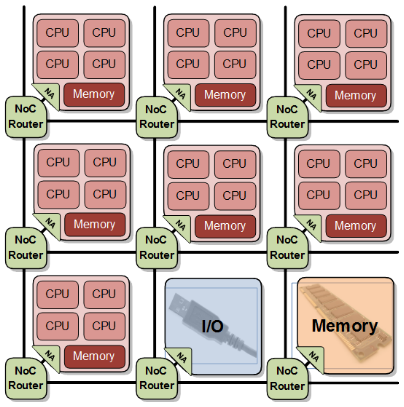
2

The Walls of Computer Architecture

- 1st Wall – Mid 90s: the Memory Wall
- 2nd Wall – 2004: the Power Wall

Steps taken:
- Multi-/Many Core, Cache Hierarchies
- NoCs and Distributed Computing with application-specific accelerators

3rd Wall – now: the Locality Wall [1]
- Memory intensive, but cache-unfriendly characteristics
- Dominated by data access & movement



How to break the locality wall?

Increase Data-to-Task Locality

- Data Migration
- Task Migration
- Near Memory Computing

Research Questions:

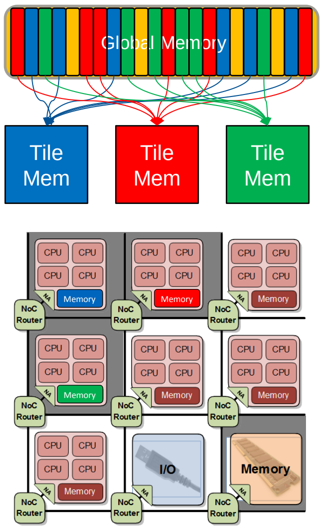
- Architecture of the future
- Best approach/combination
- Programming paradigm

MPSoC 2018: Region-based Coherence [2,3]

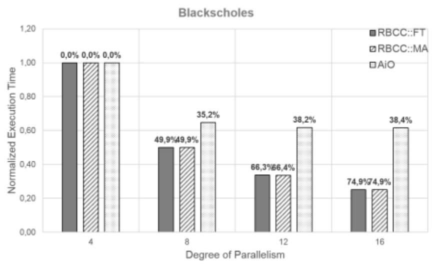
Non-uniform memory accesses (NUMA)

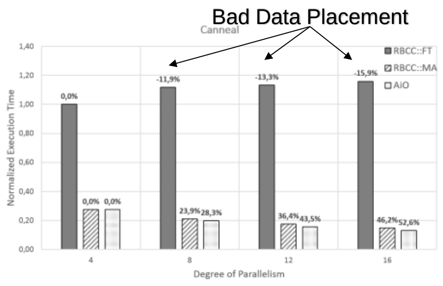
Data placement influences performance
→ Analyze the impact of data placement

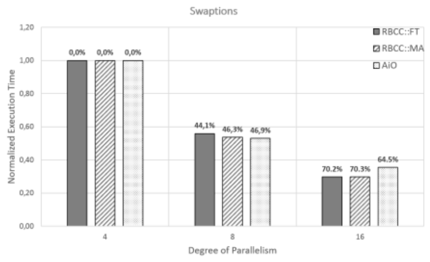
Placement Algorithms
- First Touch Policy
  - Place data into tile of first access
- Most Accessed Policy
  - Place data to preferred tile
  - Known after task execution, but exploitable for periodic task invocation
  - Maximize local accesses

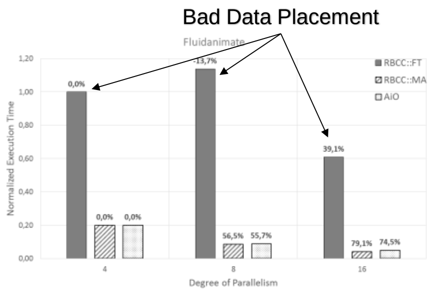


Impact of Data Placement





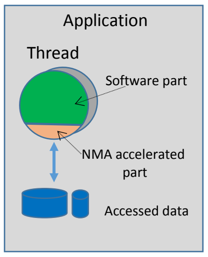


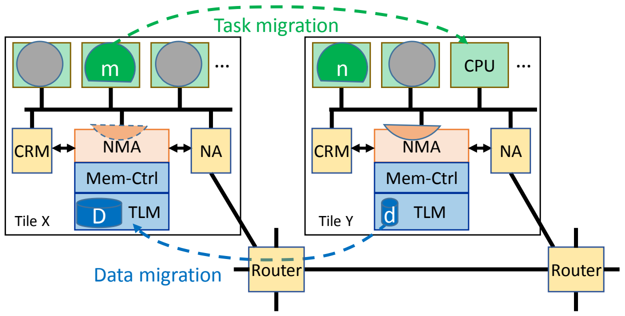


Near Memory Acceleration (NMA)

Topical research subject
- Eliminating / reducing CPU-to-memory accesses via local interconnect for memory-intense sub-functions
- Mostly addressing accelerator-centric 3D-stacked memory systems [4] [5]

Our scope
- Role of NMA in distributed-shared memory architectures
- **Implies necessity for considering task and data placement / migration**
- **SHARQ & Graphcopy**





SHARQ – An Overview [6]

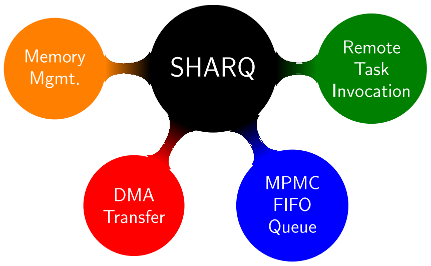
Software-Defined Hardware-Managed Queues

Combine **flexibility of software** queues with **performance of hardware** acceleration

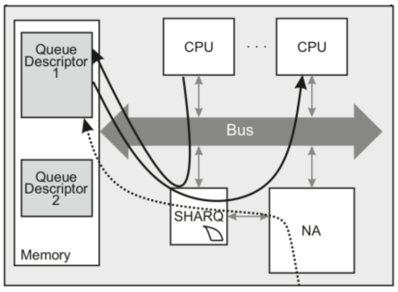
Software-Defined

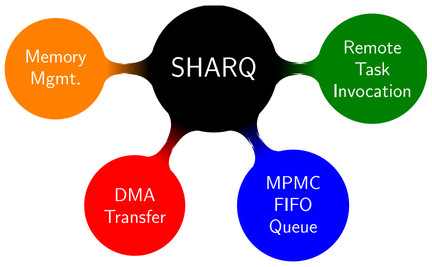
Dynamic allocation and definition of arbitrary sized queues

Hardware-Managed



Selected SHARQ Features





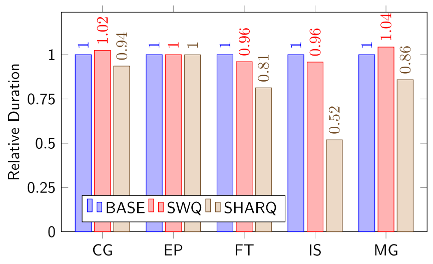
Queue & Memory Management

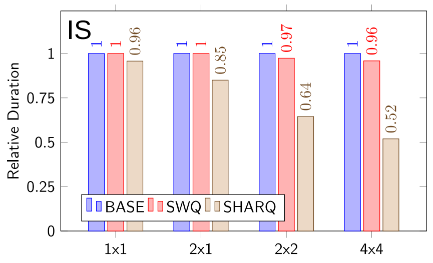
- SHARQ access to queue descriptor
- Separation into allocator stack & bounded buffer
- No software involvement / up-calls

Remote Task Invocation

- Ensure processing of elements
- Scheduled by SHARQ on demand
- Specify max. number of handler tasks
- Contract between hardware & software

Evaluation: NAS-Benchmarks & Scalability





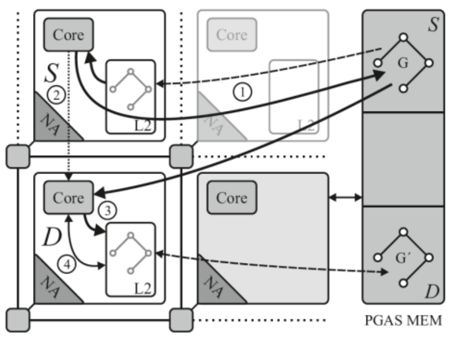
Benchmark Results

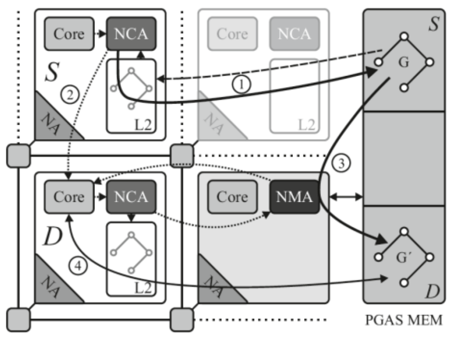
- Communication intensive kernels highly benefit from SHARQ
- Only EP (embarassingly parallel) does not profit
- SHARQ has good scaling behaviour
- Especially multi-tile systems benefit

Near-Memory Accelerated Graph Copy

Pegasus
- Avoids (de)serialization
- Graph copy algorithm in software

Near-Memory Graph Copy
- Reduces NoC traffic
- Graph copy algorithm in hardware





11

Near-Memory Graph Copy Results

IMSuite Graph Algorithm Kernels

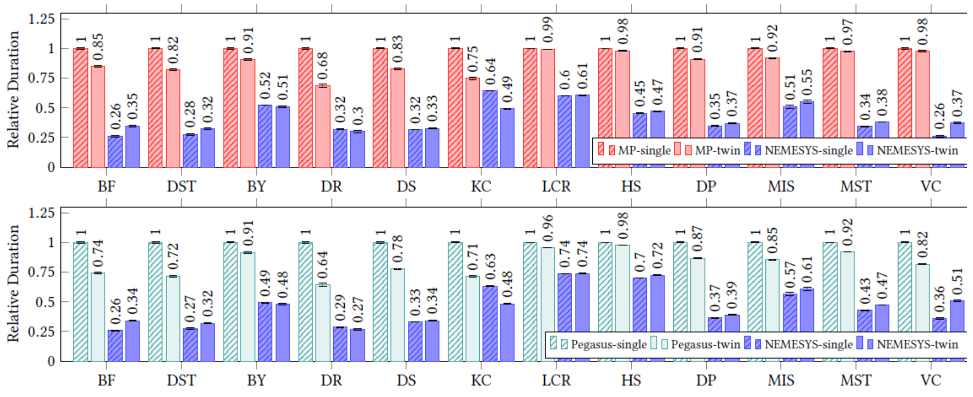


Figure 9: Runtime measurements of the IMSuite benchmarks in the 4x4 configuration. Top: NEMESYS vs. Message-passin
MP) normalized to MP-single. Bottom: NEMESYS vs. Pegasus normalized to Pegasus-single.

12

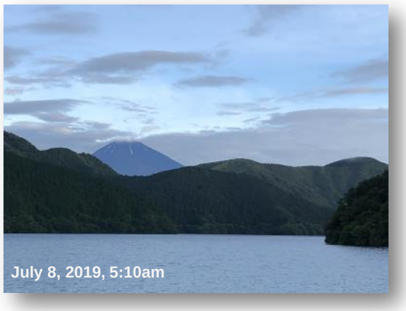
Summary

Near Memory Acceleration in Distributed-Shared-Memory architectures is an effective means to tackle the locality wall

- NMA primarily means bringing the processing closer to the data
  - … with all its implications / dependencies on data / task placement
  - … or bring / keep the data closer to the processor
    - with Region-Based Cache Coherence

This presentation: Near Memory Acceleration applied to Multicore OS or runtime middleware support functions

13



Thanks for your attention!

Acknowledgements:
This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – project number 146371743 – TRR 89: Invasive Computing. (2010 – 2022).
We like to thank our collaborators at FAU Erlangen (groups of Prof. Schröder-Preikschat, Prof. Teich) and KIT Karlsruhe (Prof. Becker, Prof. Snelting).

14

References

[1] Peter Kogge. 2017. Memory Intensive Computing, the 3rdWall, and the Need for Innovation in Architecture. https://memsys.io/wp-content/uploads/2017/12/The_Wall.pdf
[2] Akshay Srivatsa, Sven Rheindt, Thomas Wild, Andreas Herkersdorf. “Region Based Cache Coherence for Tiled MPSoCs”. 2017 30th IEEE International System-on-Chip Conference (SOCC), 2017
[3] Akshay Srivatsa, Sven Rheindt, Dirk Gabriel, Thomas Wild, Andreas Herkersdorf. CoD: Coherence-on-Demand – Runtime Adaptable Working Set Coherence for DSM-based Manycore Architectures. (SAMOS 2019)
[4] S. F. Yitbarek, T. Yang et al, “Exploring specialized near-memory processing for data intensive operations”, in DATE, pp. 1449-1452, 2016.
[5] F. Schuiki, M. Schaffner et al, “A Scalable Near-Memory Architecture for Training Deep Neural Networks on Large In-Memory Datasets”, 2018.
[6] Sven Rheindt, Sebastian Maier, Florian Schmaus, Thomas Wild, Wolfgang Schröder-Preikschat, Andreas Herkersdorf. SHARQ: Software-Defined Hardware-Managed Queues for Tile-Based Manycore Architectures. (SAMOS 2019)
[7] Manuel Mohr and Carsten Tradowsky. 2017. Pegasus: Efficient Data Transfers for PGAS Languages on non-cache-coherent many-cores. DATE.