A Case for Co-scheduling for Hybrid Memory Based Systems

Eishi Arima[1,2], and Carsten Trinitis[2]
[1]The University of Tokyo, [2]Technical University of Munich

THE UNIVERSITY OF TOKYO

Technical University of Munich

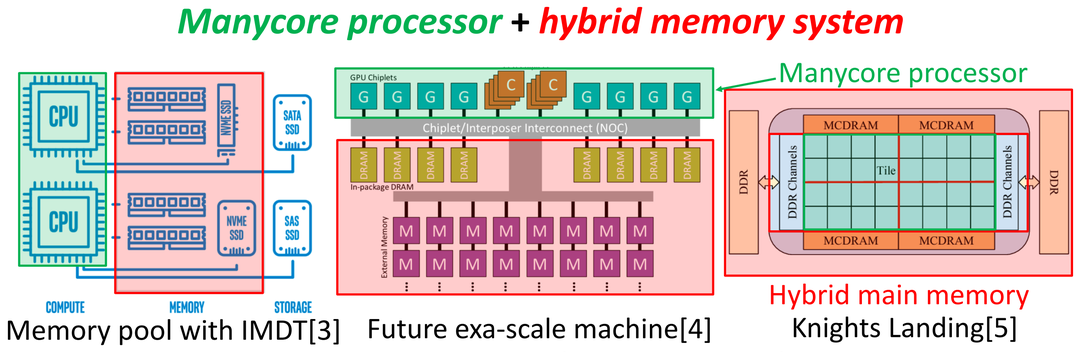
1. Background

Recent technology trends:

- ✓ Manycore processors (e.g., A64FX, KNL, GPUs) for ***higher arithmetic throughput***
- ✓ 3D stacked DRAMs (e.g., wide-I/O, HMC, and HBM[1]) for ***higher memory bandwidth***
- ✓ Emerging NVRAMs (e.g., Optane DIMM/SSD [2,3]) for ***larger memory capacity***

Promising node architecture (our target):

Manycore processor **+** ***hybrid memory system***



Memory pool with IMDT[3] | Future exa-scale machine[4] | Knights Landing[5]

Fig. 1: Target systems (extracted figures)

2. Motivation and Goal

Problem: **increasing resource wastes within a node**

1. Waste of ***processor core resources*** when memory intensive
2. Waste of ***memory bandwidth resources*** when CPU intensive
   - ✓ The wastes become ***even worse*** when these ***resources are scaled***

Promising approach: ***co-scheduling*** **[6]**

- ✓ Co-running multiple jobs on one single node at the same time
- ✓ ***Mixing CPU/memory intensive jobs is effective***

Goal: ***to understand the impact of co-scheduling on the recently emerging hybrid memory based systems***

- ✓ Note: the combination of co-scheduling and hybrid memory system ***is not well studied in prior studies***
- ✓ To do so, we utilized the system summarized in the table below

TABLE1: Evaluation settings

CPU Package	XeonPhi 7210, 64cores, 1.3GHz, quadrant mode, x1 socket
Memory System	**MCDRAM(1st memory):** 16GB 450GB/s, **DDR4(2nd memory):** 96GB 90GB/s, **Data management:** hardware cache mode
OS	Cent OS 7
Compiler	Intel C++/Fortran Compiler 19, **Options:** -O3, -qopenmp, -xMIC-AVX512
Workloads	Streaming + CORAL benchmark[7] (AMG, LULESH, MCB, miniFE, SNAP)

3. Summary

Our new insight: ***problem size awareness is quite important when co-scheduling multiple applications on the hybrid memory based node***

- ✓ The followings highly depend on the problem sizes:
  1) Optimal selections of co-run application pairs
  2) Optimal resource allocations to them

4. Experiment

CPU resource requirement analysis (solo-run)

- ✓ ***The CPU resource requirement (= thread-level scalability) depends on the problem size***
  – The 2nd memory (***large but slow***) is more frequently accessed when the ***problem size*** is scaled, which ***changes the bottleneck***

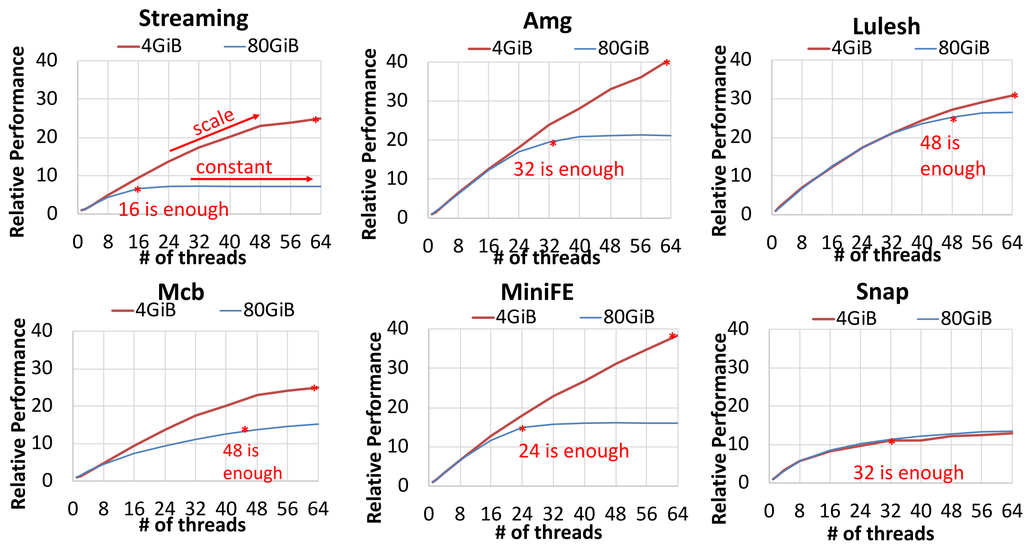


Fig. 2: Thread-level scalability vs data size

Optimal† core allocations when co-running

- ✓ ***The optimal settings also depend on the problem sizes due to the scalability changes***

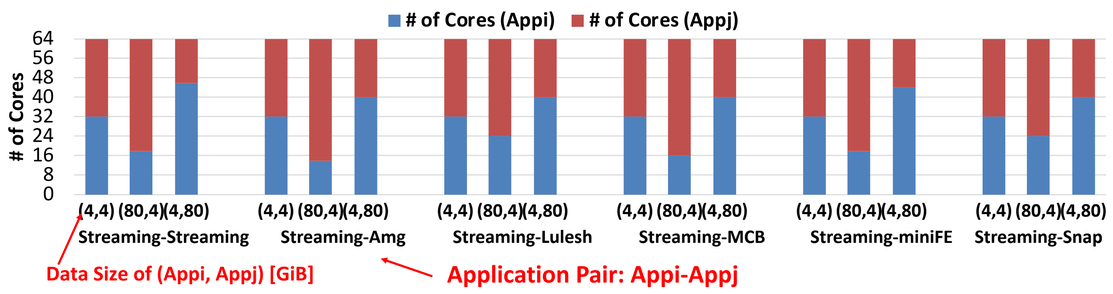


Fig. 3: Core allocation breakdowns

Performance improvement (co-run opportunities)

- ✓ ***Speed-up highly depends on the problem sizes*** * Core settings are optimal†
  – ***We should revisit the pair selection policy in co-scheduling to consider this***

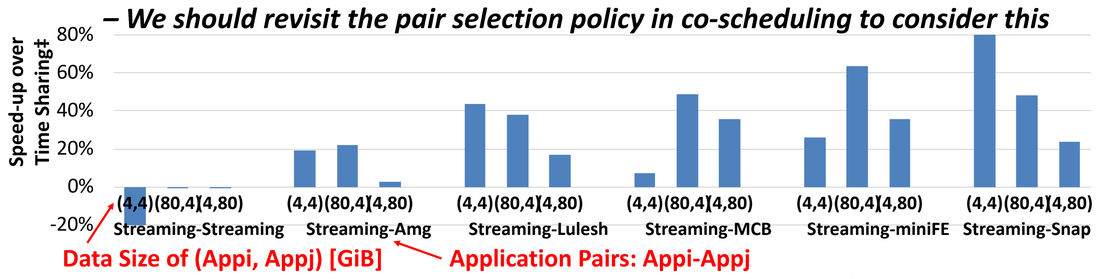


Fig. 4: Performance improvement over time sharing‡

Impact of problem size awareness on co-run performance

- ✓ ***A non-problem-size-aware resource allocation policy will not work well***
  – ***We should rethink the policy too***

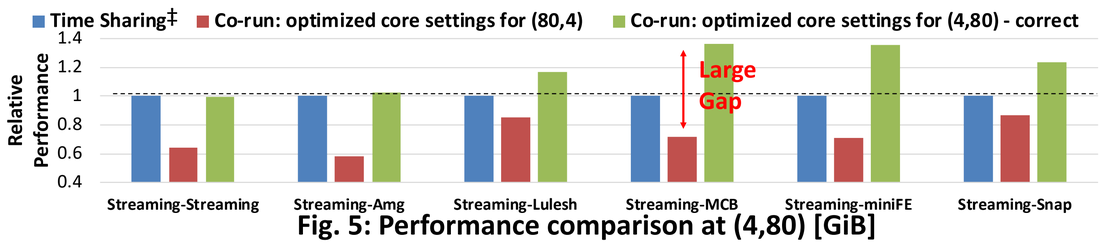


Fig. 5: Performance comparison at (4,80) [GiB]

†***Optimal***: total execution time is minimized ‡***Time sharing:*** conventional exclusive solo-runs

5. Future Challenges

1) Developing problem size aware co-scheduling methodology/algorithm
2) Extending a conventional co-scheduling framework to support our approach
3) Evaluating our strategy using various hybrid memory systems(e.g.,DRAM DIMM + Optane DIMM/SSD[2,3])

This Poster & Extended Abst.

Contact Info:

Eishi Arima, The University of Tokyo
Email: arima@cc.u-tokyo.ac.jp

Acknowledgement:

We thank Prof. Martin Schulz and the members of his research group at TU Munich for valuable discussions. This work is partly supported by JSPS Grant-in-Aid for Early-Career Scientists (JP18K18021), and Research on Processor Architecture, Power Management, System Software and Numerical Libraries for the Post K Computer System of RIKEN.

References:

1. JEDEC Standard. “High Bandwidth Memory (HBM) DRAM,” JESD235 (2013).
2. J. Izraelevitz et al. “Basic Performance Measurements of the Intel Optane DC Persistent Memory Module,” arXiv preprint arXiv:1903.05714 (2019).
3. Intel. “Intel® Memory Drive Technology, Set Up and Configuration Guide,” 2019.
4. T. Vijayaraghavan et al. “Design and Analysis of an APU for Exascale Computing,” In HPCA, pp. 85–96, 2017.
5. J. Jeffers et al. “Intel Xeon Phi Processor High Performance Programming: Knights Landing Edition,” Morgan Kaufmann Publishers Inc. 2016.
6. M. Bhadauria et al. “An Approach to Resource-aware Co-scheduling for CMPs,” In ICS, 189–199, 2010.
7. CORAL Benchmark Codes. https://asc.llnl.gov/CORAL-benchmarks/.