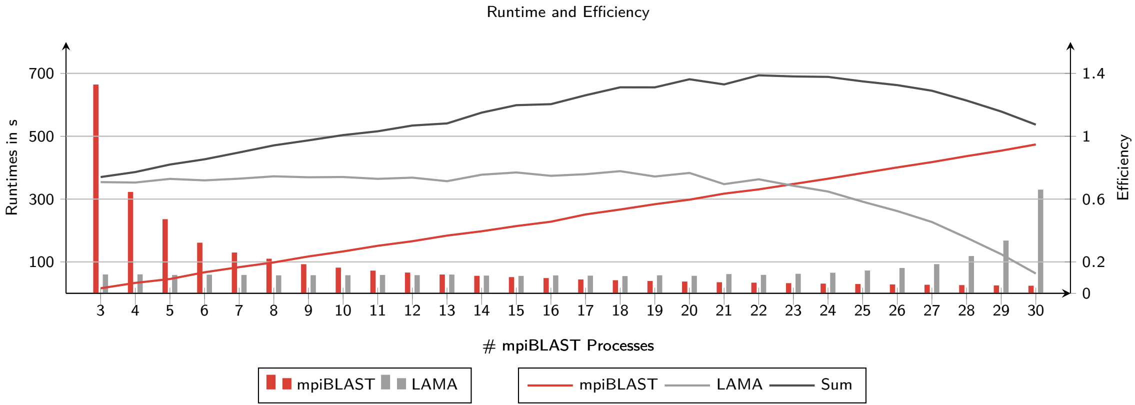
Co-scheduling on Upcoming Many-Core Architectures

2nd Workshop on Co-Scheduling of HPC Applications

Simon Pickartz[1], **Jens Breitbart**[2], and ***Stefan Lankes***[1]

[1]**RWTH Aachen University** [2]**Bosch Chassis Systems Control**

Co-scheduling on Multi-Core Architectures



Do These Results Hold for Many-core Architectures?

- A very large number of cores per chip with a simple micro-architecture
  - → Can a subset of the cores still saturate the main memory bandwidth?
- More complex memory hierarchies
  - → How map these onto the applications?
- Wider SIMD instructions
  - → We need highly vectorizable code

Agenda

- Test Setup
- Application Scalability
- Co-scheduling on the KNL
- Conclusion

The Intel KNL

- 72 cores on 36 connected via a 2D on-die mesh
- 4 hardware thread contexts per core
- 16 GiB on-die MCDRAM + 96 GiB DDR4 RAM
- Directory-based cache coherency
- Clustermodes

  All-to-all Memory addresses are uniformly hashed across the distributed tag directories

  Cluster Affinity between the directories and the memory via four virtual quadrants

  Hybrid A mix of both
- Memory Configurations

  Cache The MCDRAM acts as last-level cache

  Flat Extension of the physical adress space via additional NUMA domain(s)

  SNC Exposes the quadrants as NUMA domains to the system

The NAS Parallel Benchmarks

- 5 computing kernels and 3 pseudo applications of computational fluid dynamics
- Performance evaluation of high-performance systems
- Kernels used for our evaluation (Class C)

CG	Irregular memory access and communication
EP	Embarrassingly parallel

Application Scalability

- Exclusive execution of a single application
- Pinning strategies (on the core-level respectively)

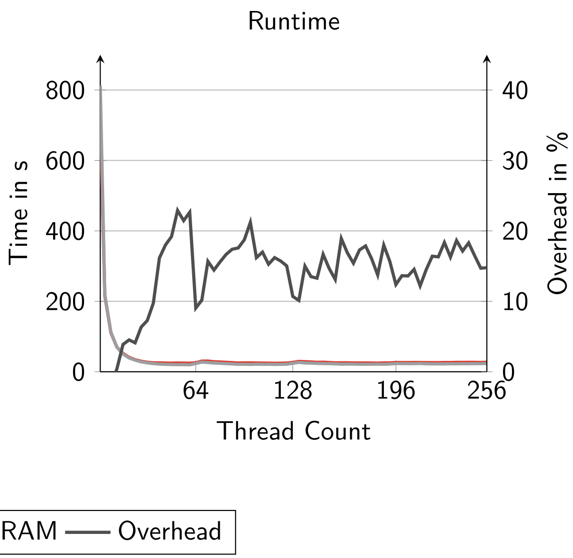
  CG Scatter

  EP Compact
- The speedup is computed based on the most efficient sequential run

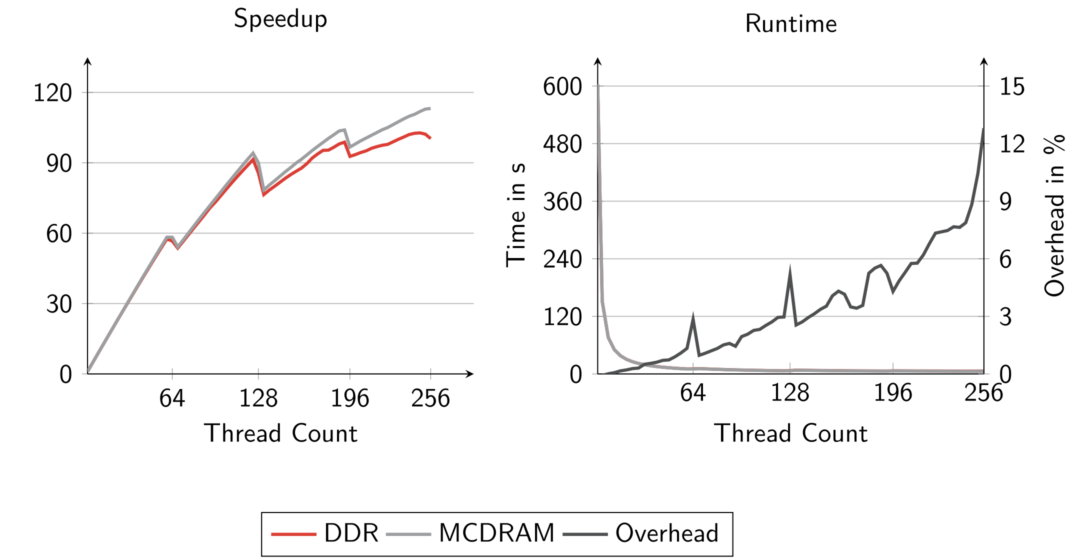
  → This was on DDR4 for both applications
- Power measurement
  - Overall consumption via Clustsafe PDU[1]
  - RAPL counters provided by the KNL (i. e., package and RAM)

[1] http://www.megware.com

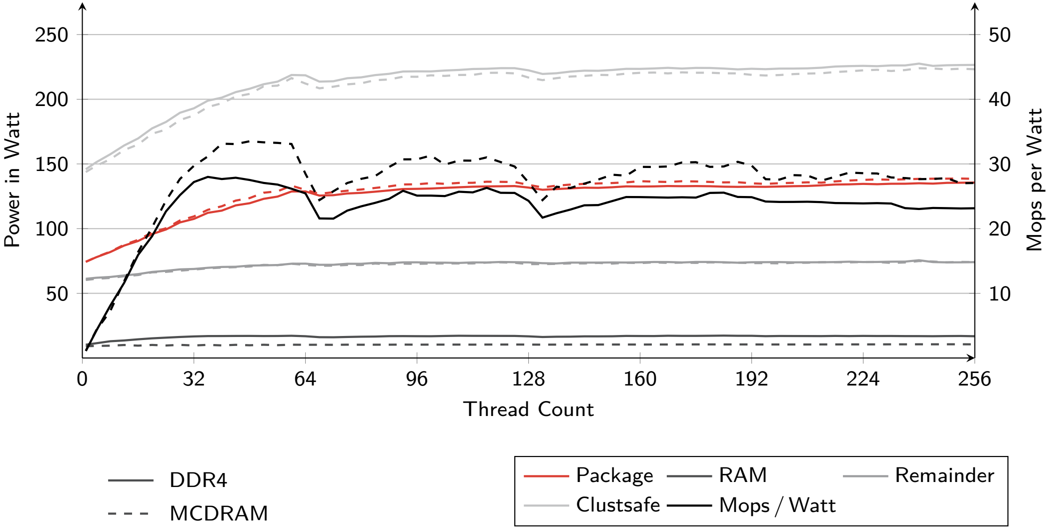
Application Scalability – CG



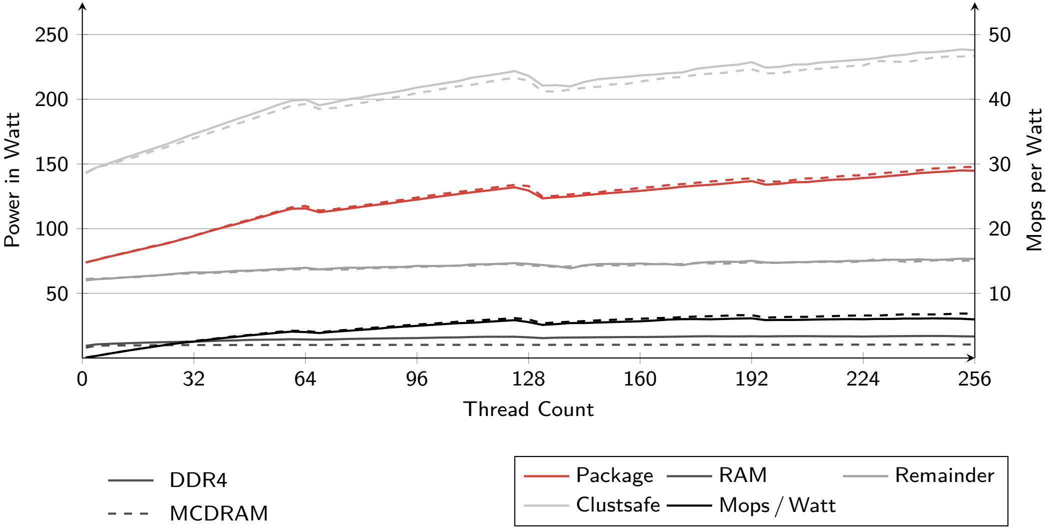
Application Scalability – EP



Power Consumption – CG



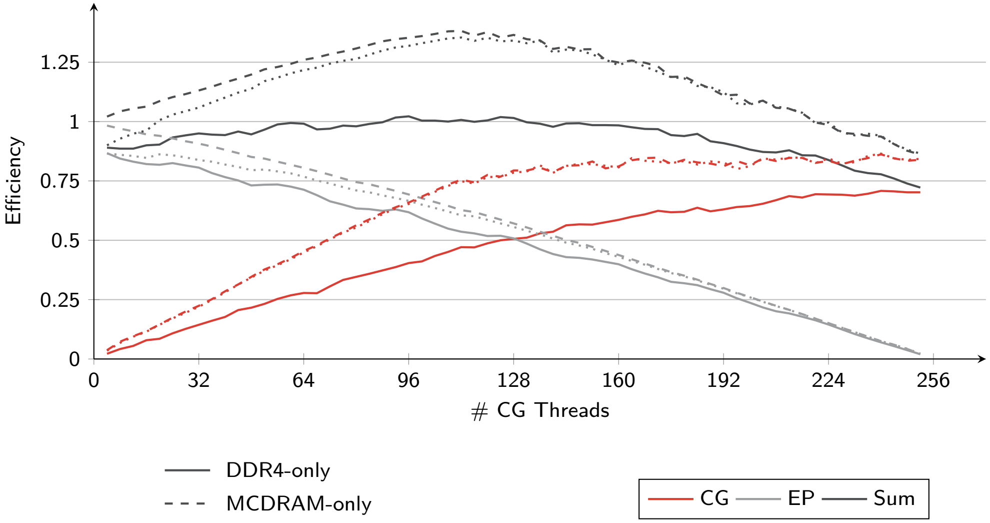
Power Consumption – EP



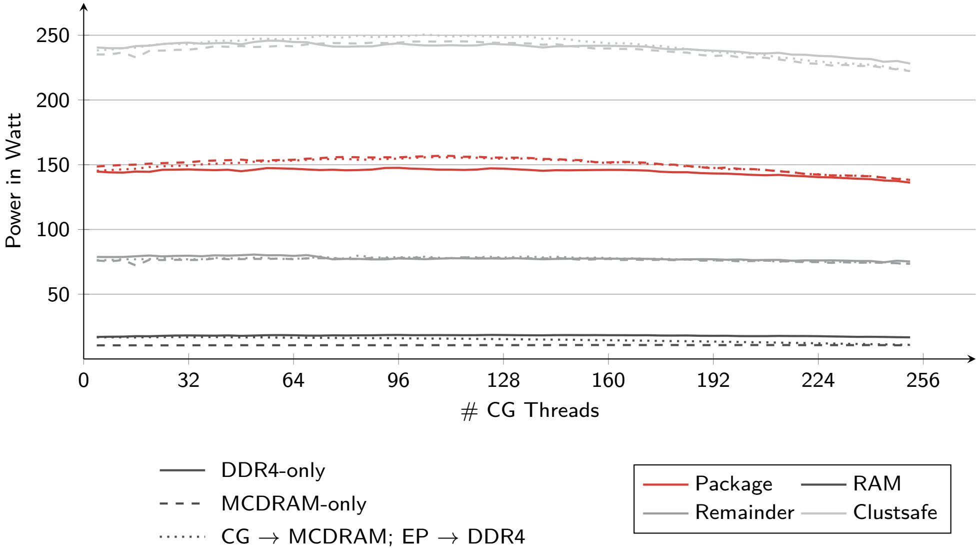
Co-scheduling

- Use *n* CG threads (scatter pinning) and fill up the remaining cores with EP
- The efficiency is computed based on the fastes exclusive application run
- Three different scenarios
  - Both kernels on DDR4 memory
  - Both kernels on the MCDRAM
  - Mixed: CG on MCDRAM; EP on DDR4

Co-scheduling – Efficiency



Co-scheduling – Power



Conclusion

- Co-scheduling is still viable for architectures such as the KNL
  - → We could increase the efficiency by up to 38 %
- The usage of MCDRAM results in a reduced power consumption
  - → The energy efficiency is increased by up to 20 %
- The assignment of different (memory) resources could not lead to an additional performance gain
  - → We expect the on-die mesh to be the bottleneck in that case

Thank you for your kind attention!

Stefan Lankes – slankes@eonerc.rwth-aachen.de

Institute for Automation of Complex Power Systems
E.ON Energy Research Center, RWTH Aachen University
Mathieustraße 10
52074 Aachen

`www.eonerc.rwth-aachen.de`