Revisiting Co-Scheduling for Upcoming ExaScale Systems

Plenary talk at HPCS 2015

Stefan Lankes

FAST FOR EXASCALE

E.ON Energy Research Center

RWTH AACHEN UNIVERSITY

Agenda

- Motivation
- Migration in HPC Environments
  - Process-level
  - Virtualization
  - Container-based
- Evaluation of Migration Techniques
  - Overhead
  - Migration Time
- Co-Scheduling
- Conclusion

Motivation

Find a suitable Topology for Exascale Applications

Find a suitable Topology for Exascale Applications

- Funded by the Federal Ministry of Education and Research, Germany, 2014–2016
- Project partners
  - Johannes Gutenberg-Universität Mainz, Zentrum für Datenverarbeitung (JGU), Koordinator
  - Technische Universität München, Lehrstuhl für Rechnertechnik und Rechnerorganisation (TUM)
  - Universität zu Köln, Regionales Rechenzentrum (RRZK)
  - Fraunhofer-Institut für Algorithmen und wissenschaftliches Rechnen (SCAI)
  - RWTH Aachen University, Institute for Automation of Complex Power Systems (ACS), E.ON Energy Research Center
  - MEGWARE Computer Vertrieb und Service GmbH
  - ParTec Cluster Competence Center GmbH

Exascale Systems

- Characteristics/Assumptions
  - Increasing amount of cores per node
  - Increase of CPU performance will not be matched by I/O performance

→ Most applications cannot exploit this parallelism

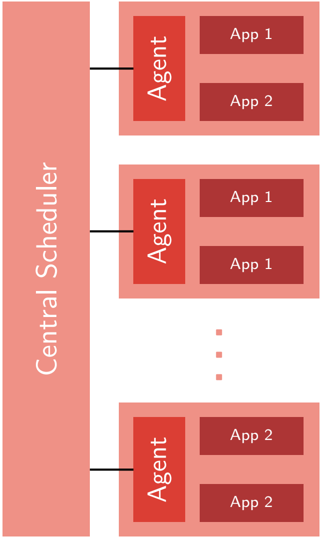
- Consequences
  - Exclusive node assignment has to be revoked (Co-Scheduling)
  - Dynamic scheduling during runtime will be necessary

→ Migration of processes between nodes indispensable

Find a Suitable Topology for Exascale Applications

A twofold scheduling approach

1. Initial placement of job by a global scheduler according to KPIs
   - Load of the individual nodes
   - Power consumption
   - Applications' resource requirements
2. Dynamic runtime adjustments
   - Migration of processes
   - Tight coupling with the applications
   - Feedback to the global scheduler



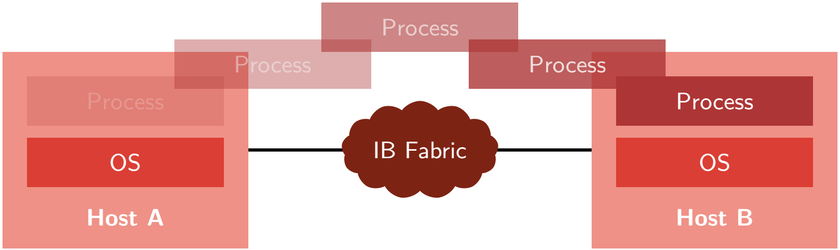
Find a Suitable Topology for Exascale Applications

Project partner – RWTH Aachen University

- Main task: Process migration for load balancing
- Close interaction between
  - Migration,
  - Scheduling,
  - Application
- Applications know their performance characteristic and resource requirements

Migration in HPC Environments

Process-level Migration



Process-level Migration

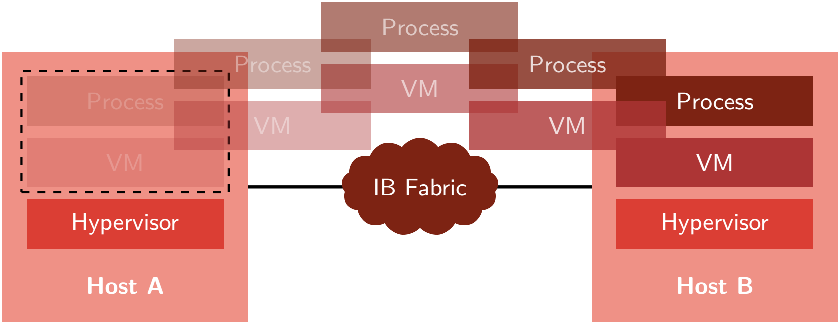
- Move a process from one node to another
  - Including its execution context (i. e., register state and physical memory)
- A special kind of Checkpoint/Restart (C/R) operation
- Several frameworks available
  - Condor's checkpoint library
  - libckpt
  - Berkley Lab Checkpoint/Restart (BLCR)
  - Distributed MultiThreaded Checkpointing (DMTCP)

Berkley Lab Checkpoint/Restart

- Specifically designed for HPC applications
- Two components
  - Kernel module for performing the C/R operation
  - Shared-library enabling user-space access
- Cooperation with the C/R procedure via callback interface
  - Close/open file descriptors
  - Tear down/reestablish communication channels
  - ...

→ Residual dependencies that have to be resolved!

Virtual Machine Migration



E.ON Energy Research Center RWTHAACHEN UNIVERSITY

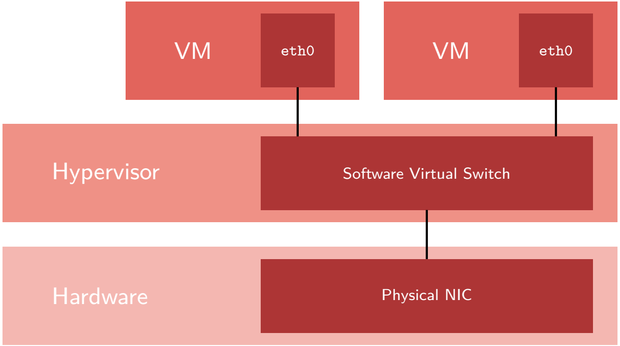
Virtual Machine Migration

- Migration of a virtualized execution environment
- Reduction of the residual dependencies
  - File descriptors still valid after the migration
  - Communication channels are automatically restored (e. g., TCP)
- Performance degradation of I/O devices can be compensated by
  - Pass-through (e. g., Intel VT-d)
  - Single Root I/O virtualization (SR-IOV)
- Various hypervisors available
  - Xen
  - Kernel Based Virtual Machine (KVM)
  - ...

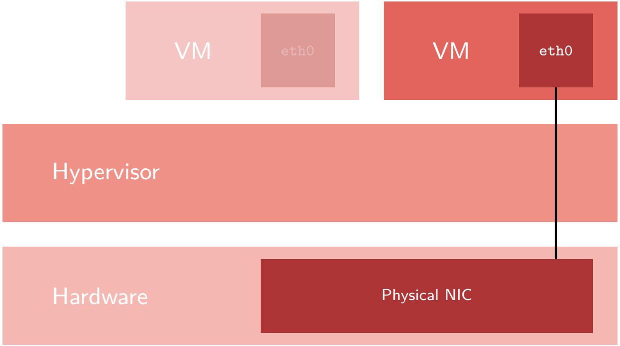
Kernel Based Virtual Machine

- A Linux kernel module that benefits from existing resources
  - Scheduler
  - Memory management
  - ...
- Implements full-virtualization requiring hardware support
- VMs are scheduled by the host like any other process
- Already provides a migration framework
  - Live-migration
  - Cold-migration

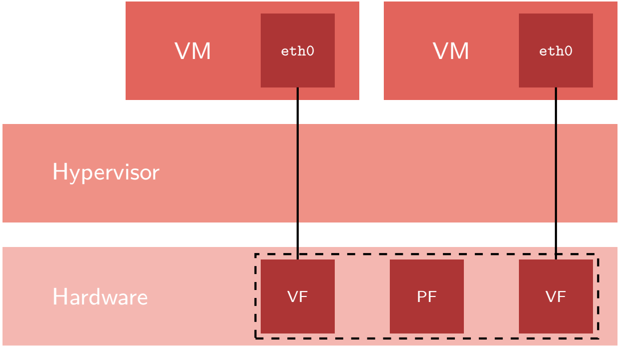
Software-based Sharing



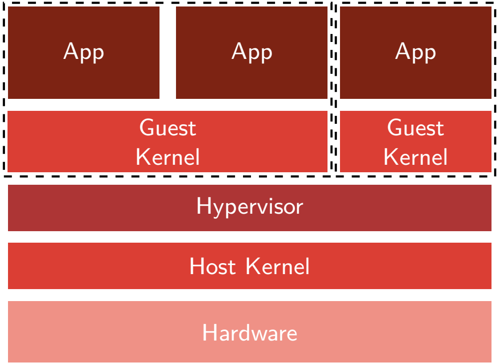
Pass-Through



Single Root I/O Virtualization

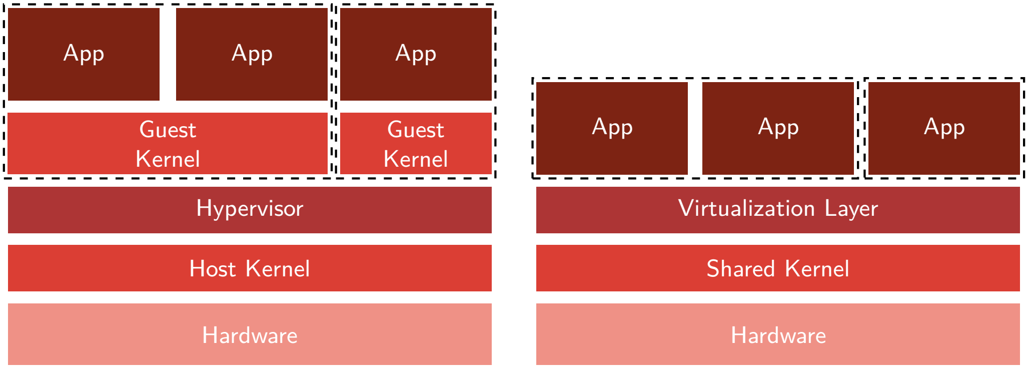


Container-based Virtualization



(a) Virtual Machines

Container-based Virtualization



(a) Virtual Machines (b) Containers

Container-based Migration

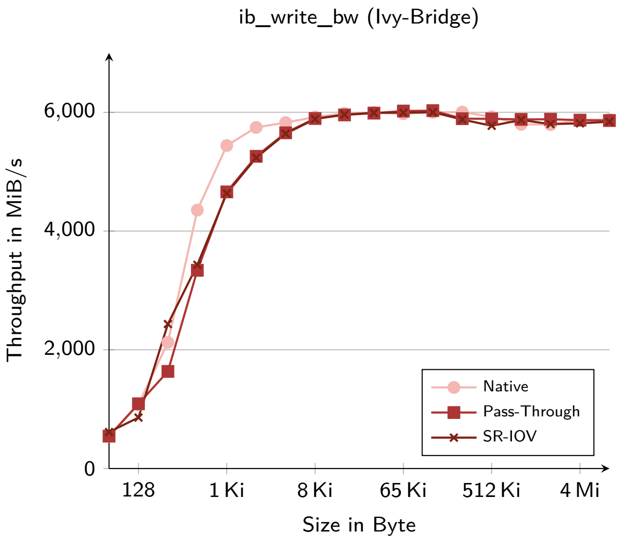
- Full-virtualization results in *multiple* kernels on one node
- Idea of container-based virtualization
  - → Reuse the existing host kernel for the management of multiple user-space instances
- Host and guest have to use the same operating system
- Common representatives
  - OpenVZ
  - LinuX Containers (LXC)

Evaluation of Migration Techniques

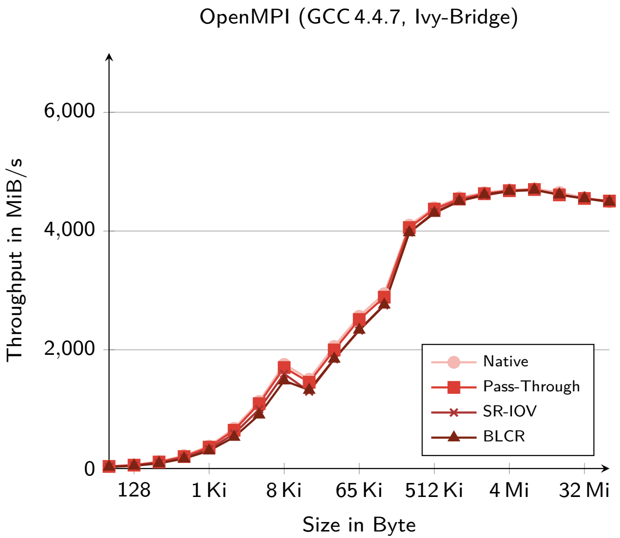
Test Environment

- 4-node Cluster
  - 2 Sandy-bridge Systems
  - 2 Ivy-bridge Systems
- InfiniBand FDR Mellanox Fabric
  - Up to 56 GiB/s
  - Support for SR-IOV
- OpenMPI 1.7 with BLCR support (except for the LXC results)

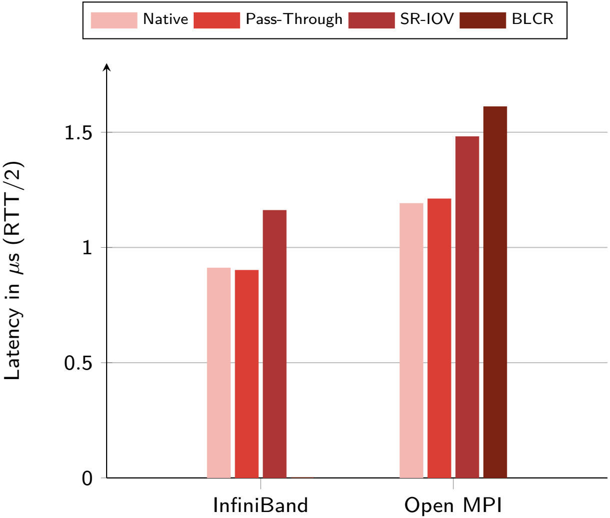
Throughput


ib_write_bw (Ivy-Bridge)
Throughput in MiB/s
6,000
4,000
2,000
0
128
1 Ki
8 Ki
65 Ki
512 Ki
4 Mi
Size in Byte
Native
Pass-Through
SR-IOV

Throughput

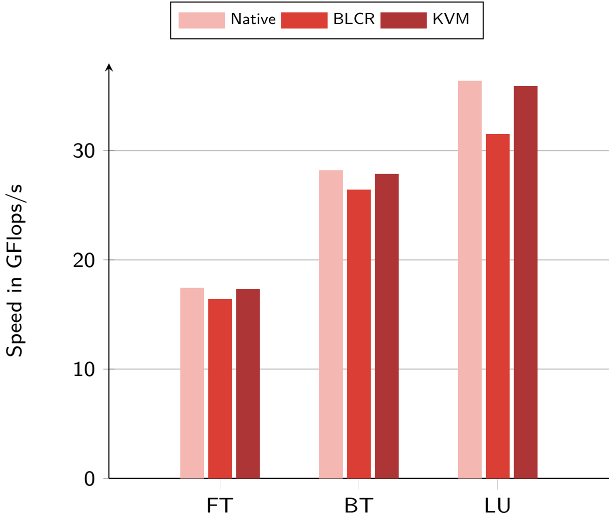

OpenMPI (GCC 4.4.7, Ivy-Bridge)
Throughput in MiB/s
6,000
4,000
2,000
0
128
1 Ki
8 Ki
65 Ki
512 Ki
4 Mi
32 Mi
Size in Byte
Native
Pass-Through
SR-IOV
BLCR

Latency

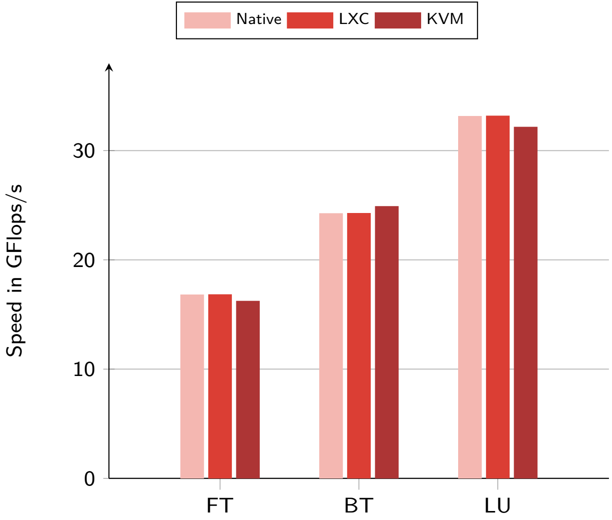


Native
Pass-Through
SR-IOV
BLCR
Latency in μs (RTT/2)
1.5
1
0.5
0
InfiniBand
Open MPI

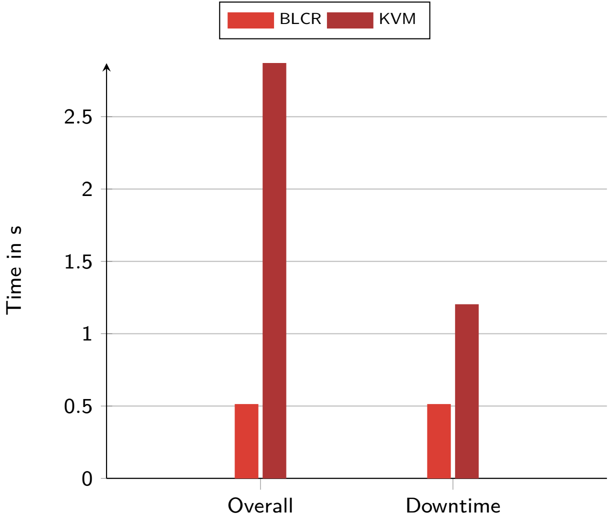
NAS Parallel Benchmarks (Open MPI)



NAS Parallel Benchmarks (Parastation)



Migration Time



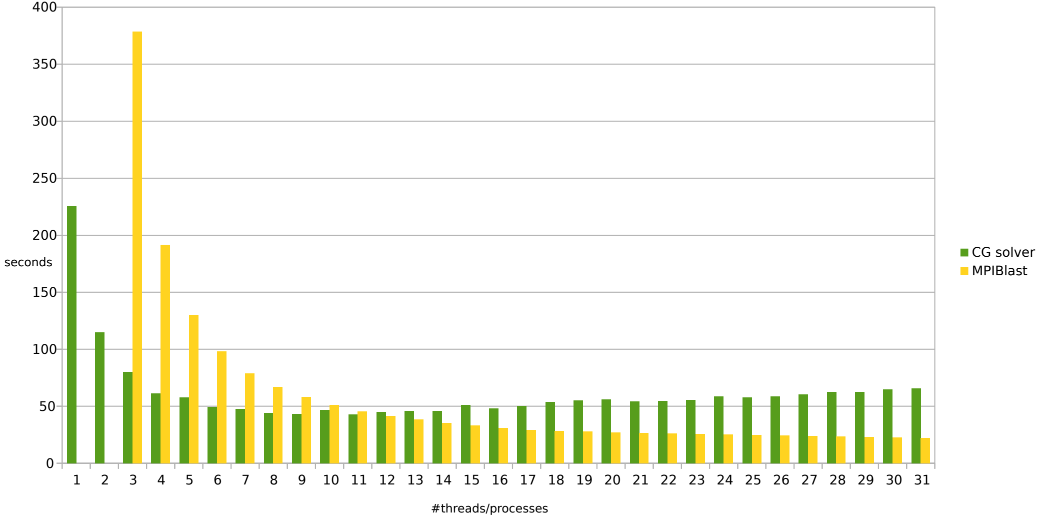
Co-Scheduling

Find a Suitable Topology for Exascale Applications

Project partner – Technischen Universität München (TUM)

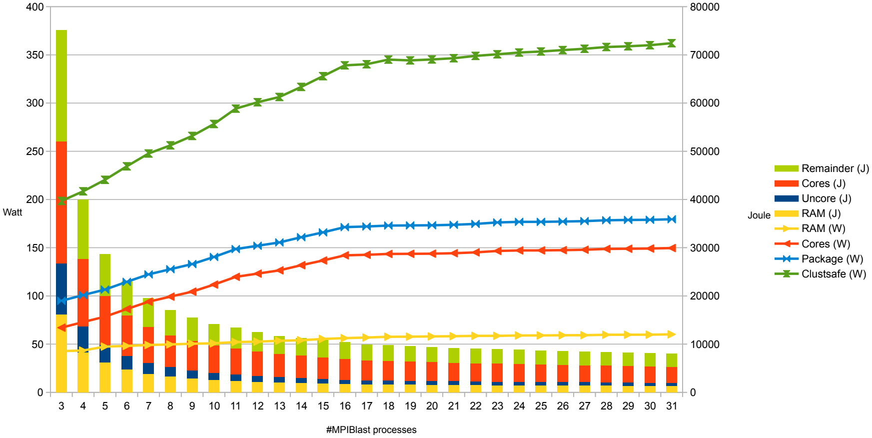
- Detailed analysis of reference applications
  - LAMA (shared memory, memory bound) und BLAST (message passing, compute bound) are analyzed with existing and for the project developed tools.
  - LAMA and BLAST are accelerated up to a factor of two
- In cooperation with JGU and RWTH, integration into the agent system
  - Further development of `autopin+`

Scalability without Co-Scheduling



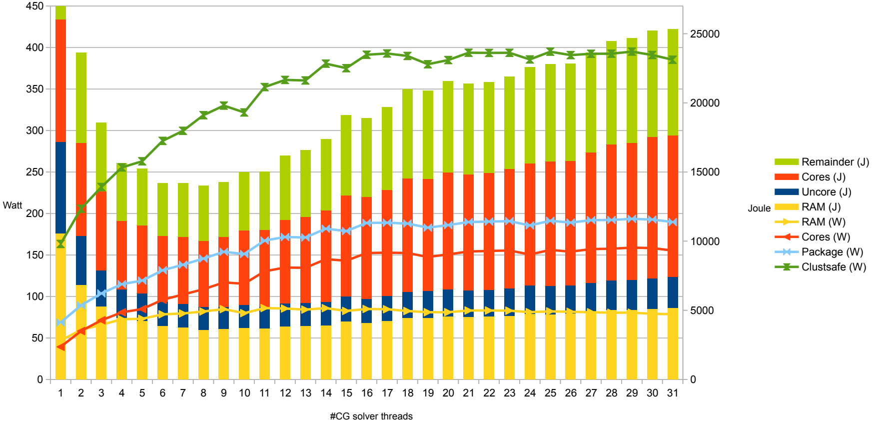
Enegry / Power Demand without Co-Scheduling

MPIBlast

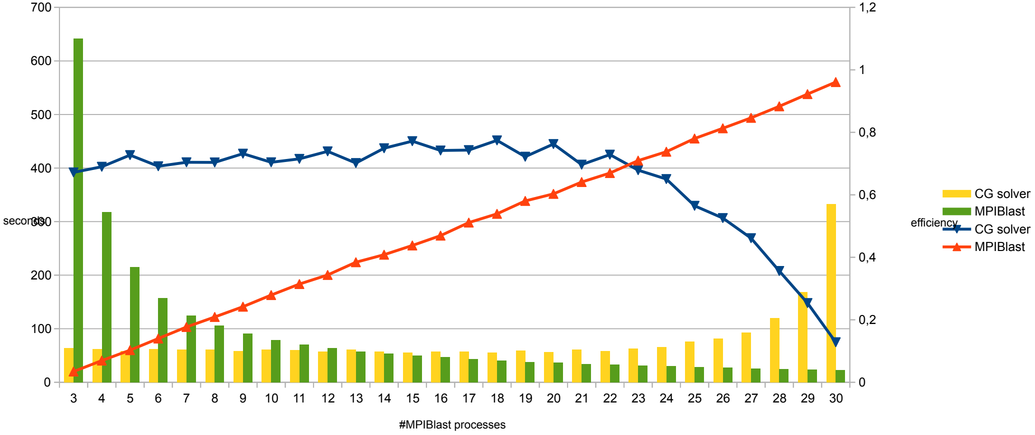


Enegry / Power Demand without Co-Scheduling

LAMA



Run time / Efficiency with Co-Scheduling



Conclusion

Conclusion and Outlook – Co-Scheduling

- Co-Scheduling quite effective in the case examined
- Improvements over sequential execution
  - Up to 12 % improvement in energy consumption (in joules).
  - Up to 28 % improvement in run time.
- Next steps: classifying application characteristics to find suitable candidates for co-scheduling
  - Survey started at german compute centers (LRZ, RZG, RRZE, RWTH, . . .)

Conclusion and Outlook – Virtualization

- Inconclusive microbenchmarks analysis
- Overhead is highly application dependent
- Migration
  - Significant overhead of KVM concerning overall migration time
  - Qualified by the downtime test
  - KVM already supports live-migration
  - BLCR requires *all* processes to be stopped during migration
- Flexibility
  - Process-level migration generates residual dependencies
    - → A non-transparent approach would be required
  - VM/Container-based migration reduces these dependencies

Conclusion and Outlook – Virtualization

- Firstly, we will focus on VM migration within FAST
- Containers may provide even better performance
  - Not as flexible as VMs (e. g., same OS)
  - More isolation than process-level migration
- Investigation of the application dependency observed during our studies
- Enable VM migration with attached pass-through devices
  - IB migration nearly finalized

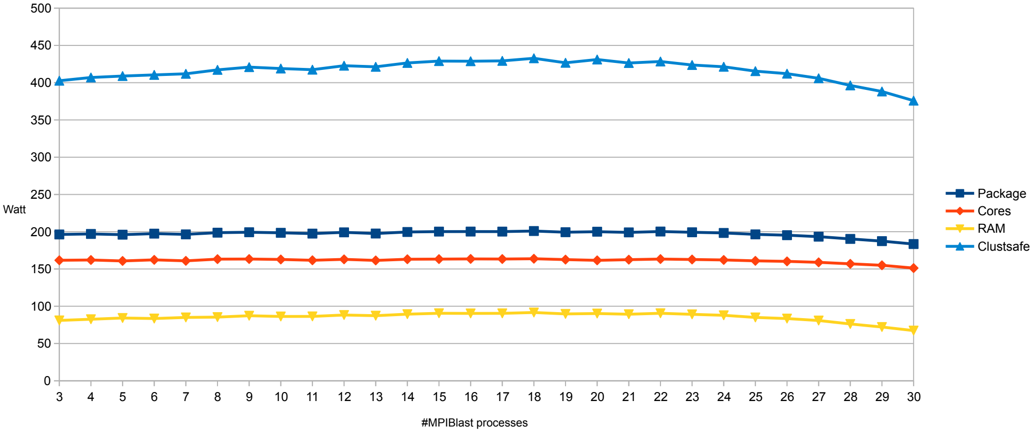
→ More about FAST on `http://www.en.fast-project.de`

Thanksgiving

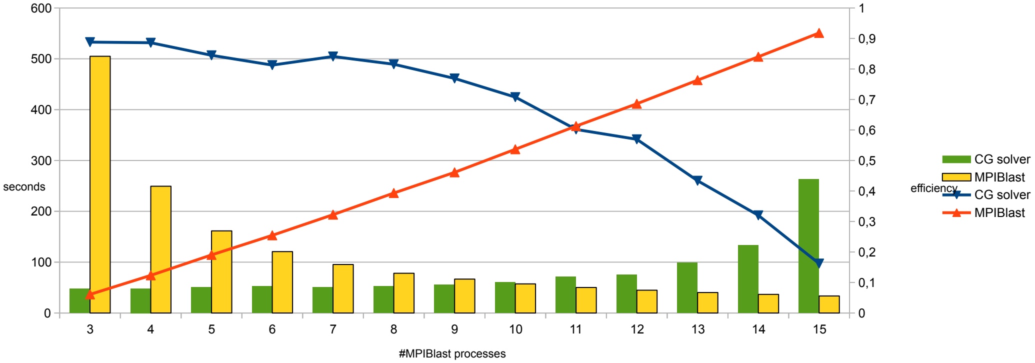
- The results[1] were obtained in collaboration with the following persons:
  - Carsten Trinitis (TUM)
  - Josef Weidendorfer (TUM)
  - Jens Breitbart (TUM)
  - André Brinkmann (JGU)
  - Ramy Gad (JGU)
  - Lars Nagel (JGU)
  - Tim Süß (JGU)
  - Simon Pickartz (RWTH)
  - Carsten Clauss (ParTec)
  - Stephan Krempel (ParTec)
  - Robert Hommel (Megware)

[1] Jens Breitbart, Carsten Trinitis, and Josef Weidendorfer. "Case Study on Co-Scheduling for HPC Applications". In: *Proceedings of the International Workshop on Scheduling and Resource Management for Parallel and Distributed Systems (SRMPDS 2015)*. Beijing, China, 2015, Simon Pickartz et al. "Migration Techniques in HPC Environments". English. In: *Euro-Par 2014: Parallel Processing Workshops*. Vol. 8806. Lecture Notes in Computer Science. Springer International Publishing, 2014, pp. 486–497. DOI: 10.1007/978-3-319-14313-2_41.

Power Demand with Co-Scheduling



Power Demand with Co-Scheduling



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
Germany

www.acs.eonerc.rwth-aachen.de