Viability of Virtual Machines in HPC

A State of the Art Analysis

22nd August 2016

Jens Breitbart[1], Simon Pickartz[2], Josef Weidendorfer[1], Antonelli Monti[2]

[1] Computer Architecture, Technische Universität München
[2] Automation of Complex Power Systems, E.ON ERC, RWTH Aachen

j.breitbart@tum.de

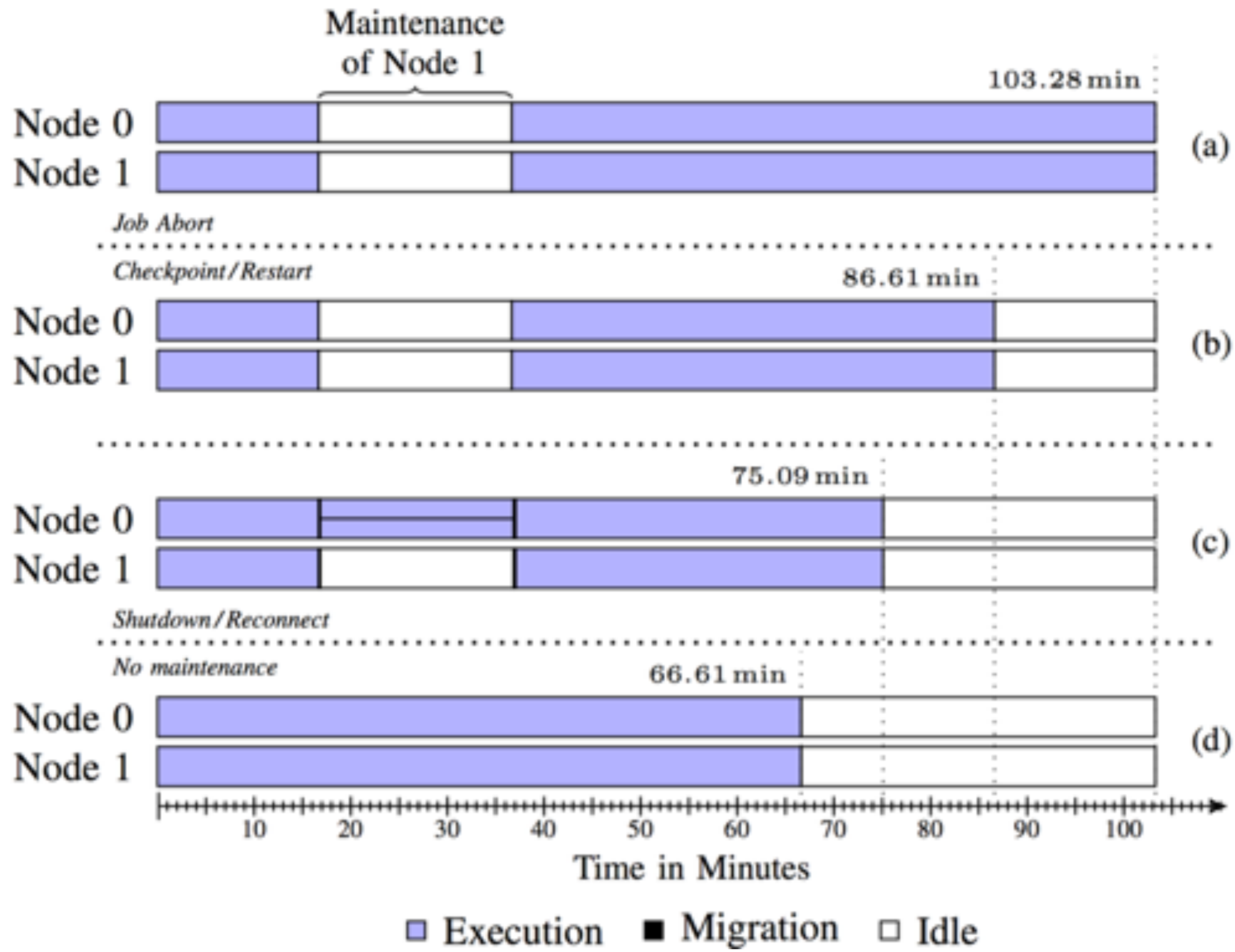
Why bother?

- Virtual Machines are widely used in various fields.

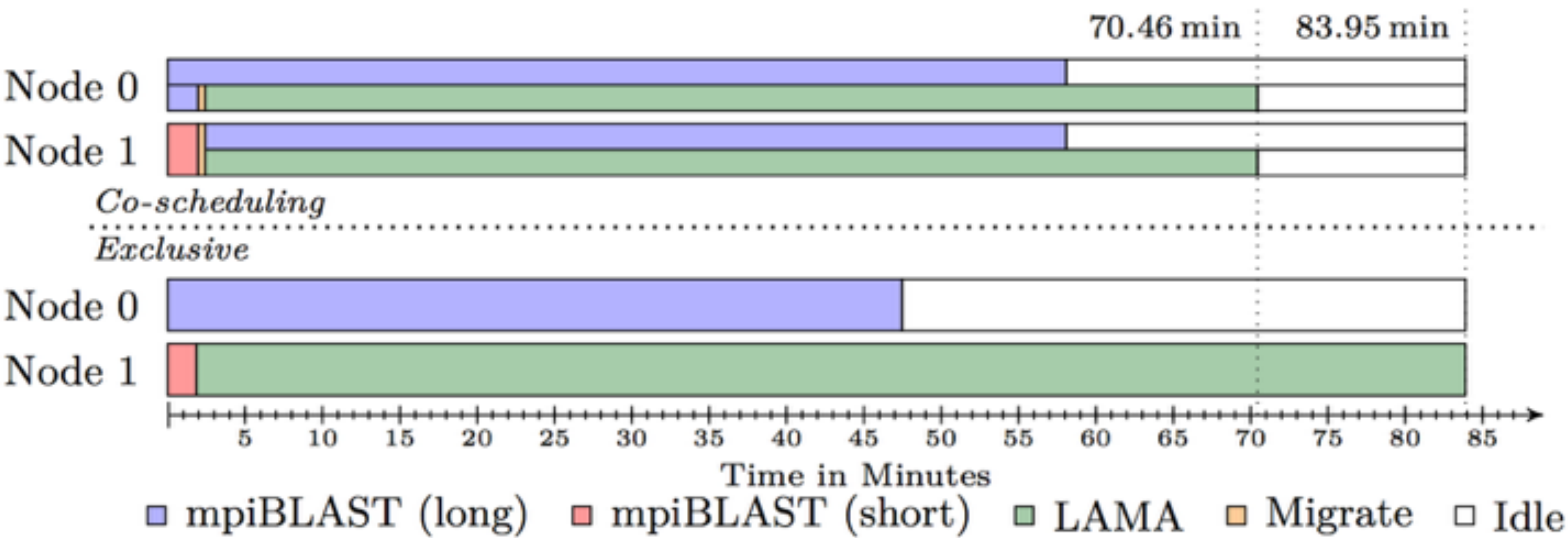
- Isolation
  - HPC systems typically isolate jobs using dedicated nodes.
  - Multiple jobs on one node can increase overall throughput.

- Transparent start, stop and migration of jobs
  - Enables hardware maintenance without loosing job progress.
  - Reorchestrate job placement at runtime.

Why bother? — Maintenance



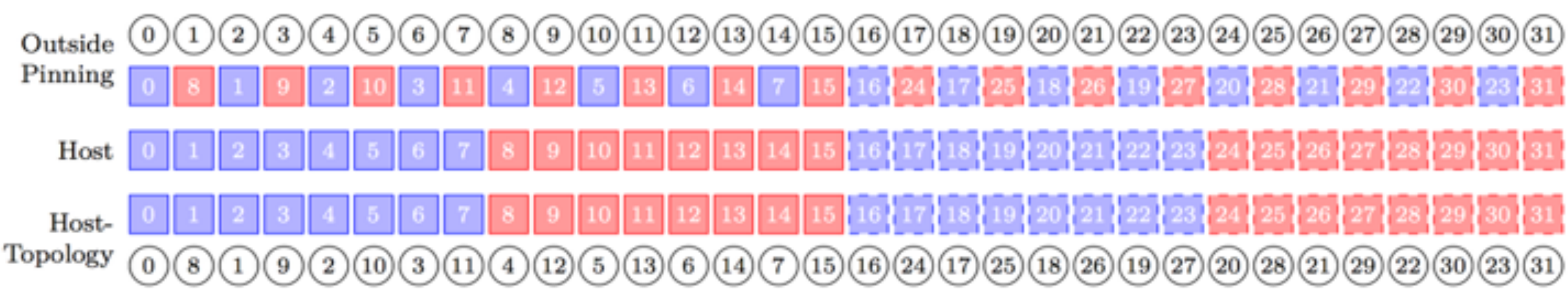
Why bother? — Reorchestration



Virtual Machines

- PCIe devices may be passed-through directly to the VM and Single Root I/O Virtualization (SRIOV) can be used
  - See our previous paper for details

- Virtual CPUs
  - => is thread-to-core mapping still effective?



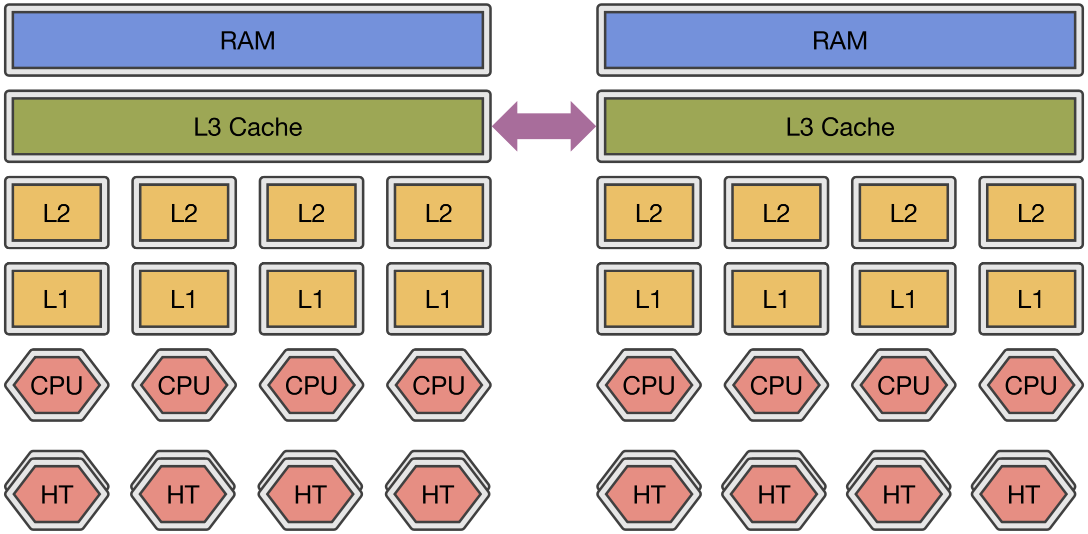
Virtual Machines

- PCIe devices may be passed-through directly to the VM or use Single Root I/O Virtualization (SRIOV)
  - See our previous paper for details
- Virtual CPUs
  - => is thread to core mapping still effective?
- Nested page tables with two level page walk
  - => is main memory bandwidth affected negatively?

Hardware - Specification

- 2 Intel Xeon E5-2670 (Sandy Bridge) with 8 cores / 16 HTs each
- 2.6—3.3 GHz
- 115 W TDP for each CPU
- 2 * 64 GB memory
- QDR Infiniband, 1 GBit/s Ethernet, SSD

Hardware - Energy Measurements



- RAPL - Running Average Power Limit
  - Cores: CPU cores and L1/2 cache
  - Package: whole package
  - DRAM: main memory

- MEGWARE Clustsafe PDU: whole system incl. power supply

Applications — MPIBlast

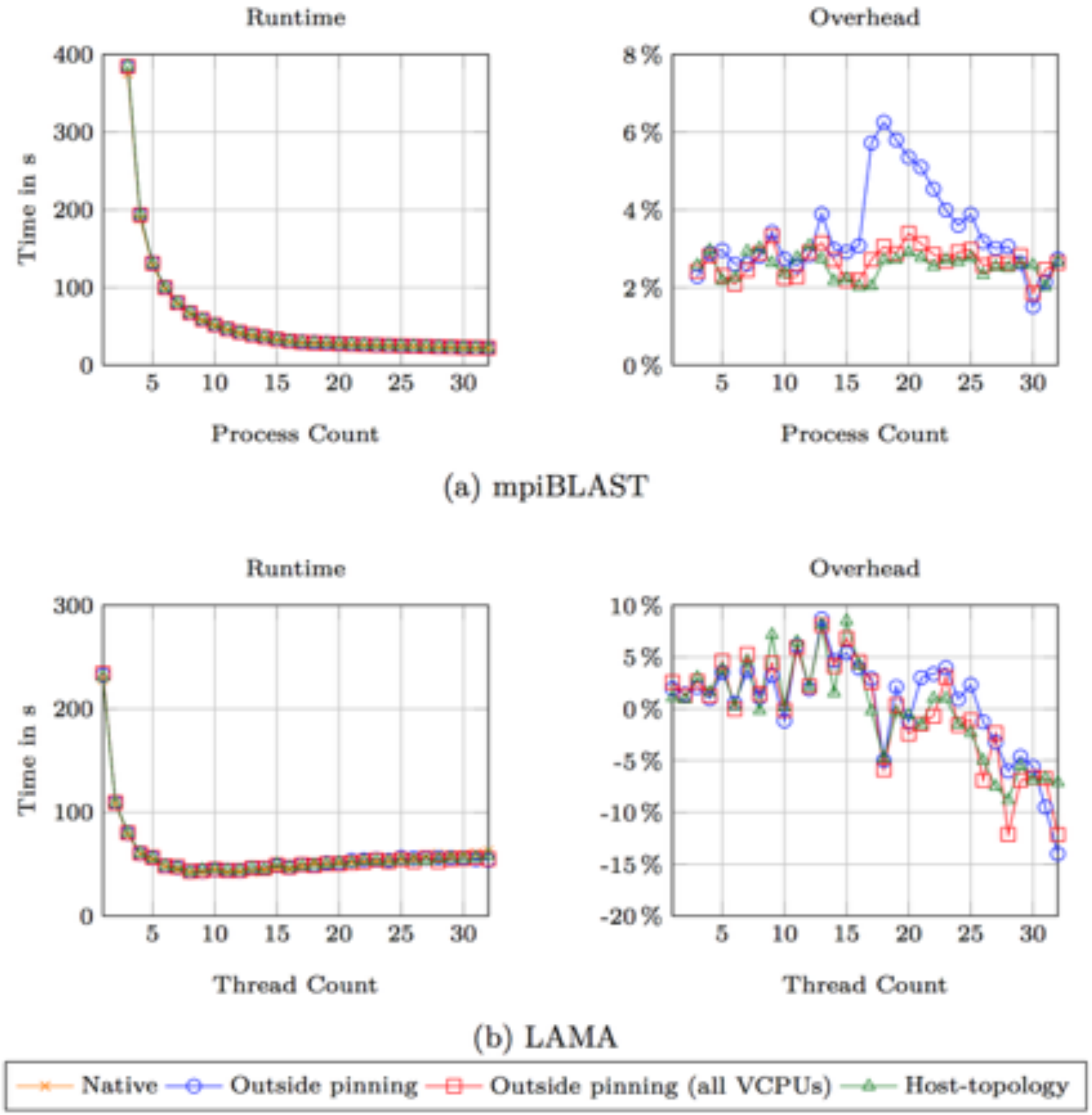
- We used a slightly modified version of MPIBlast 1.6.0
- It is a computational bioinformatics application
- "embarrassing parallel"
- Data fits into L1 cache
- A lot of instruction dependencies within the main kernel
- **A compute bound application**

Applications — CG solver

- Part of the LAMA library
- Conjugate gradient solver used with randomly created matrices
- Uses OpenMP for shared memory parallisation
- About 70% of the runtime is spent in Intels MKL

- **A main memory bandwidth limited application**

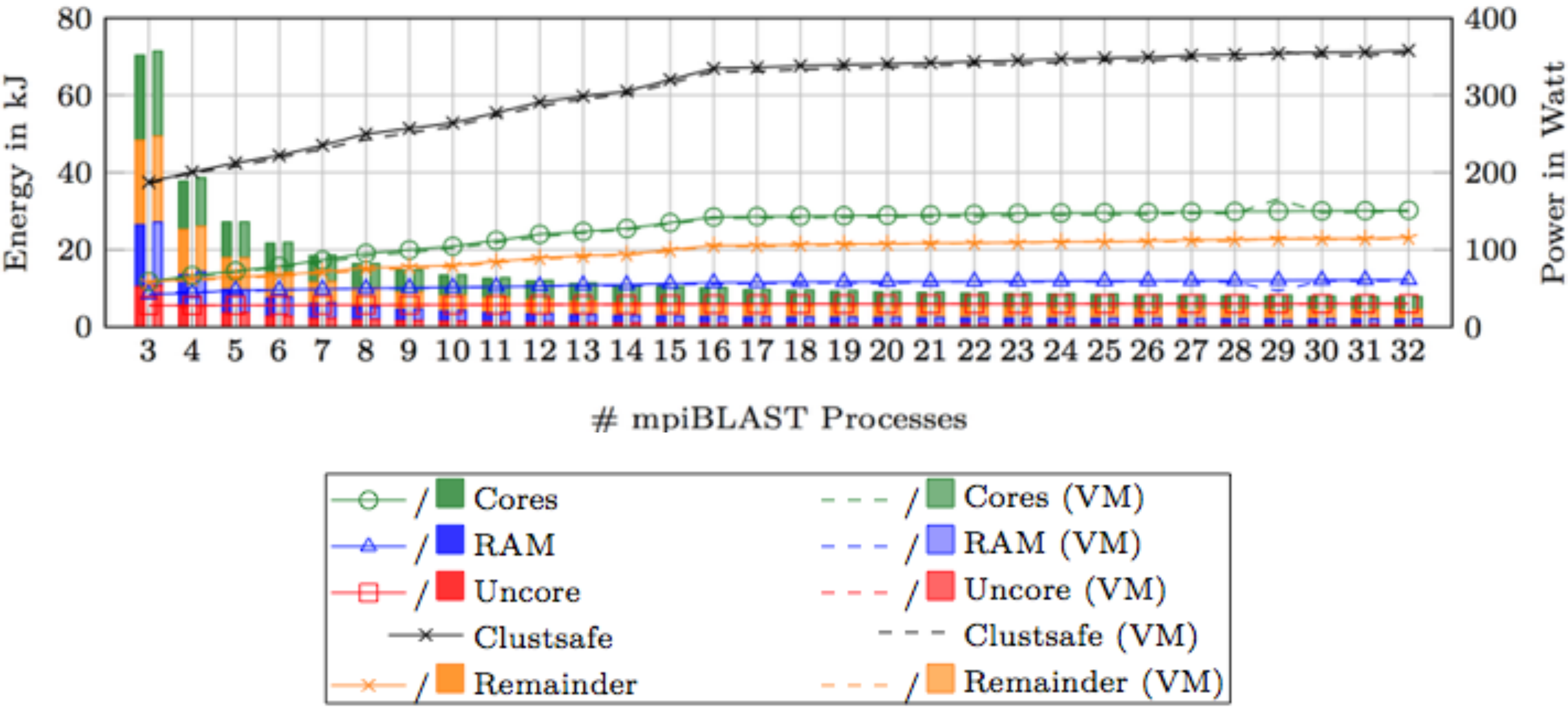
Thread pinning with VMs



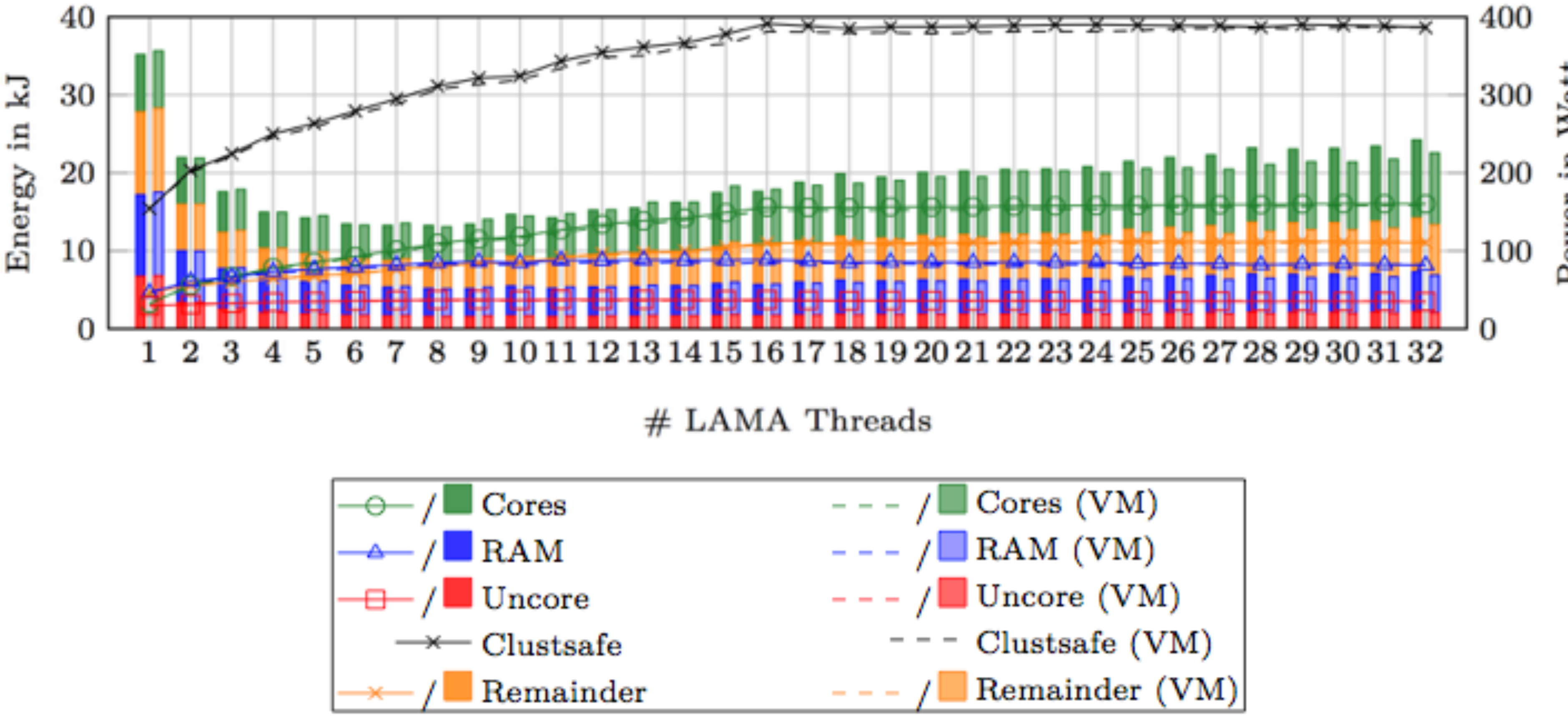
(a) mpiBLAST

(b) LAMA

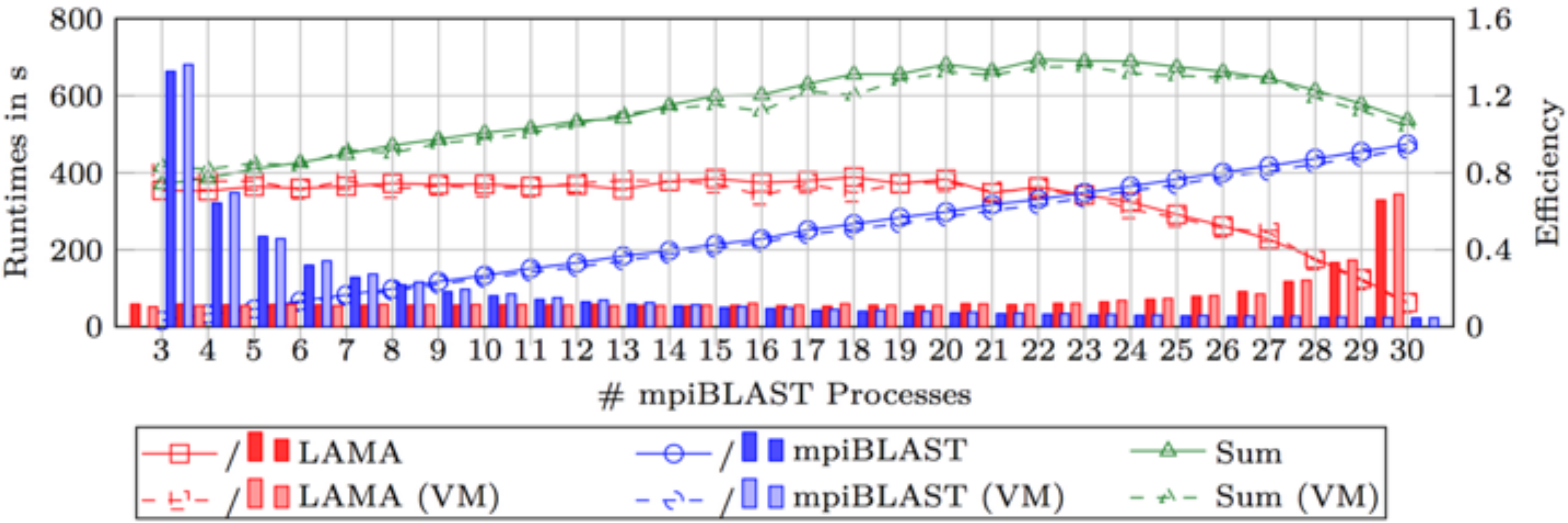
Energy consumption within VMs — MPIBlast



Energy consumption within VMs — LAMA



Co-scheduling with VMs



VMs in HPC

- Overall performance is fine…
  - … besides a small drop only noticeable in STREAM
- Energy consumption is fine as well
- But…

VMs in HPC

- Increase in complexity
  - We could not identify the reason for the performance increases when running LAMA within a VM.
  - Thread pinning gets more complicated and most runtimes don't get it right.
- Start, stop, or migrate is not possible with a VM that has an attached PCIe device (such as Infiniband).
  - MPI support is required!
    - We have a prototype.
- Inter-VM intra host communication is slow => VM granularity is important.

Conclusion

- Most benefits cannot be achieved with the default HPC software stack.
- But there are various possibilities that should be analyzed further.

- Please take a look at www.en.fast-project.de for related research.