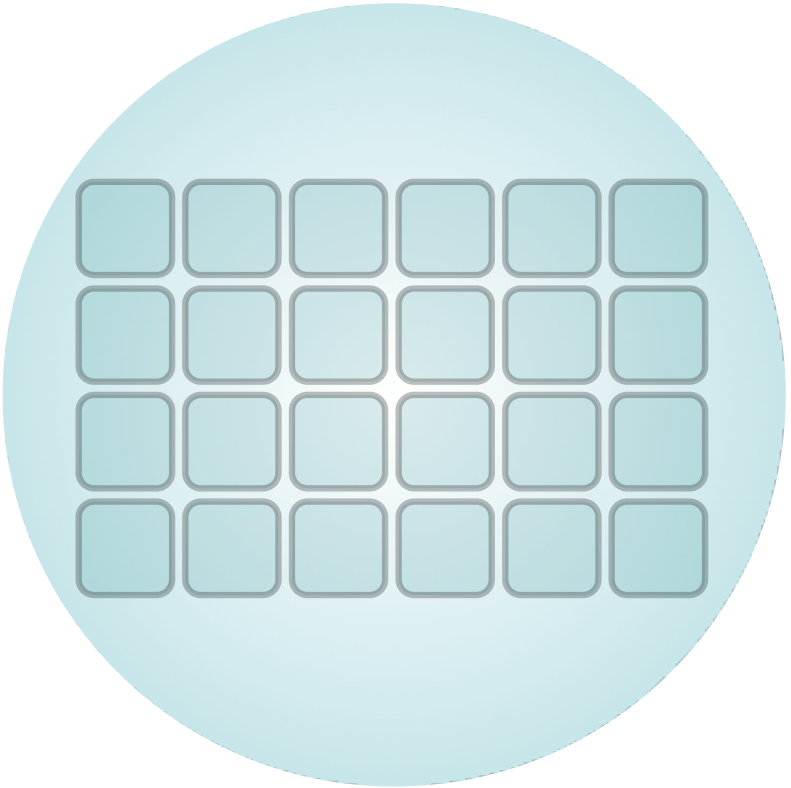
A Life without Cache Coherency – A triennial Experiment on Oneself

Dr. rer. nat. Stefan Lankes
Dr.-Ing. Carsten Clauß



Agenda

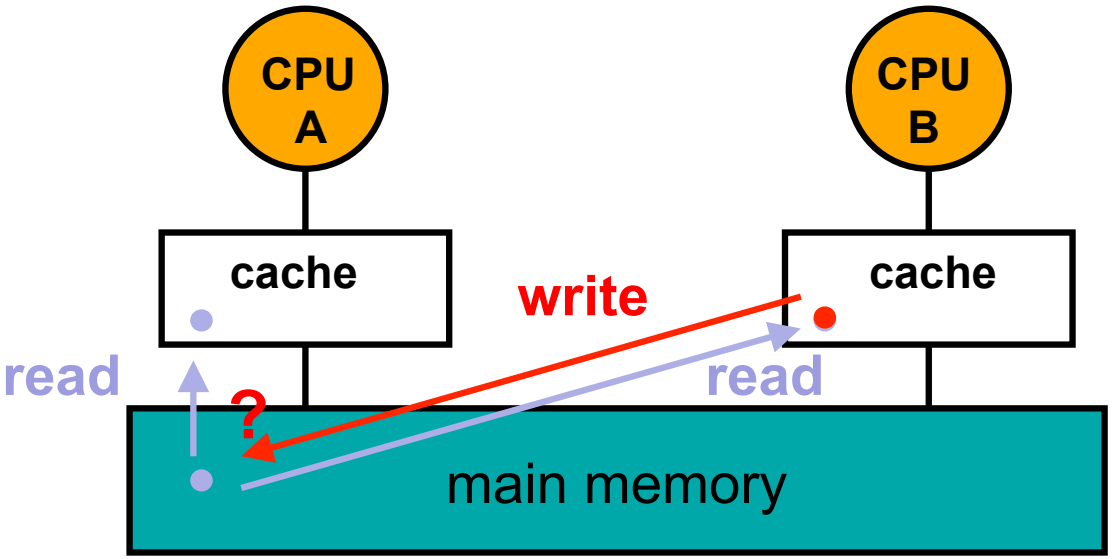
RWTH AACHEN UNIVERSITY

- Demand on Non-Coherent Memory-Coupled Cores
- Intel Single-Chip Cloud Computer (SCC)
- Message Passing vs. Shared Memory Paradigm
  - → iRCCE
  - → MetalSVM
- Performance Characteristics
- Comparison to current architectures
- Outlook and Conclusions

A Life without Cache Coherency

RWTH AACHEN UNIVERSITY

- The Cache Coherency Issue on Multi Processor Systems



Demand on Non-Coherent Memory-Coupled Cores

RWTH AACHEN UNIVERSITY

- **Ongoing assumption:** a cache coherent shared address space
  - → coherence incurs additional architectural overhead
    - » transistors **and** performance
  - → the raising core number leads to increasing chip complexity
    - » higher verification complexity

- Why do we not waive the cache coherency?
  - → e.g. message-passing scales and requires no cache coherency

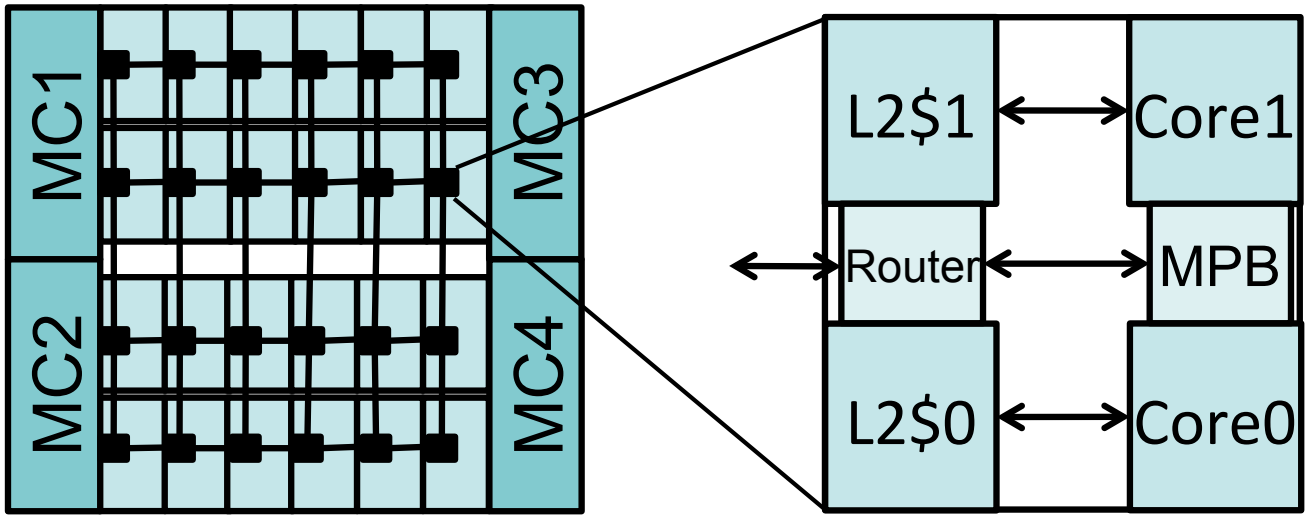
- Is message-passing the model of choice?
  - → using the best programming model for the specific algorithms

- Is there already such an architecture?
  - → Intel's research vehicle *Single-Chip Cloud Computer* (SCC)

The Intel SCC Many-core Processor

RWTH AACHEN UNIVERSITY

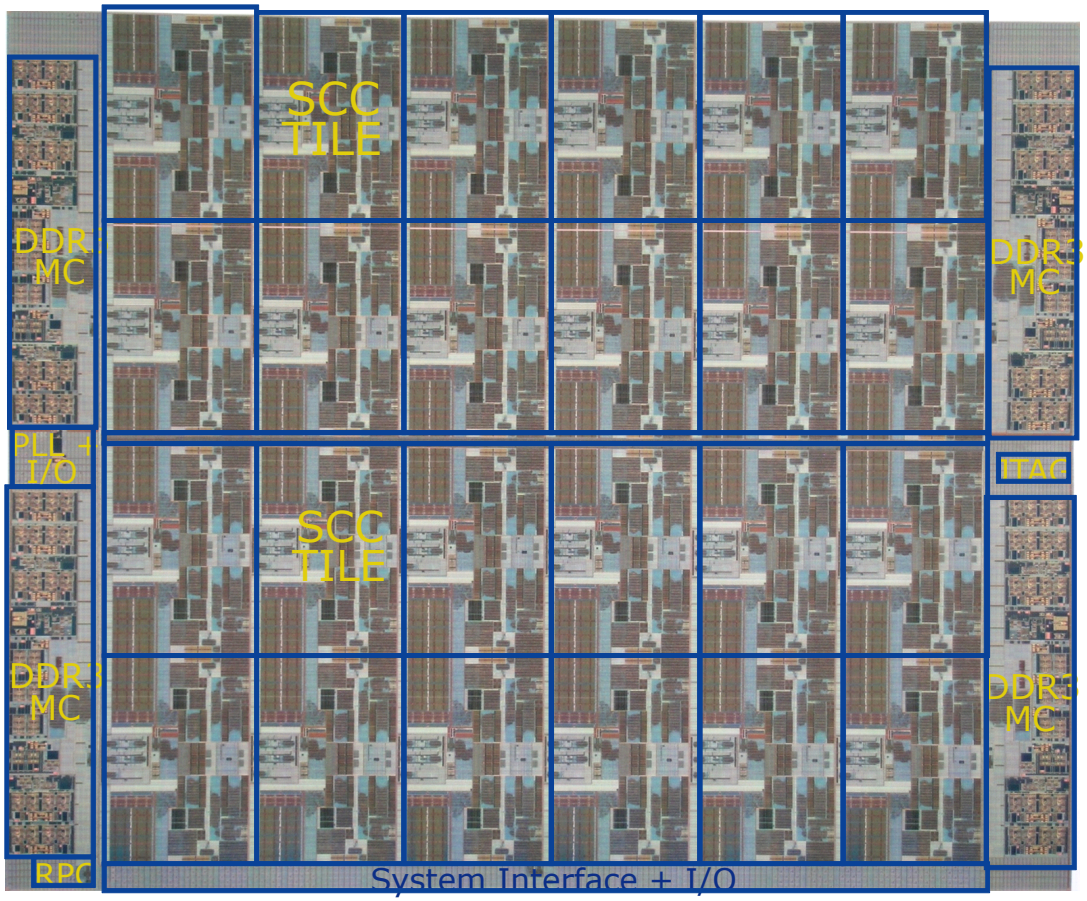
- 48 Pentium-I Cores (arranged in a 6x4 on-die Mesh)
  → 2 Cores and 1 Router per *Tile*
- On-die *Message-Passing Buffers* (MBP)
  16kByte per Tile (→ 8kByte per core)
  → accessible as distributed on-die *Shared-Memory*
- 4 on-die Memory Controllers (MC1-4)
  → max. 64GByte DDR3 off-die main memory

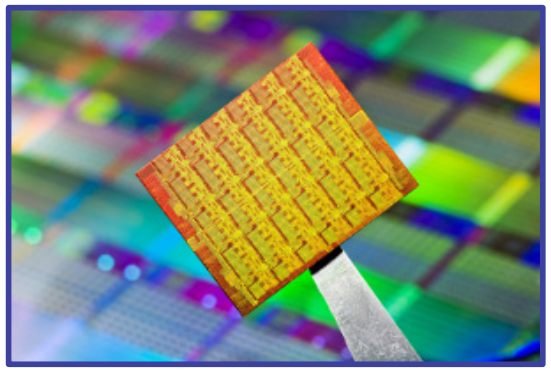


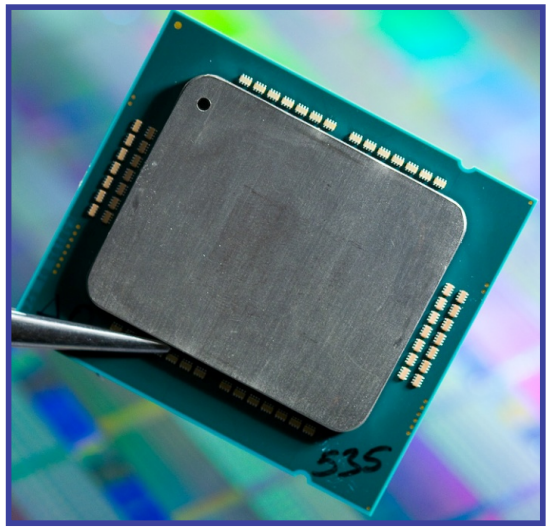
The Intel SCC Many-core Processor

RWTH AACHEN UNIVERSITY

- Images by Intel Labs

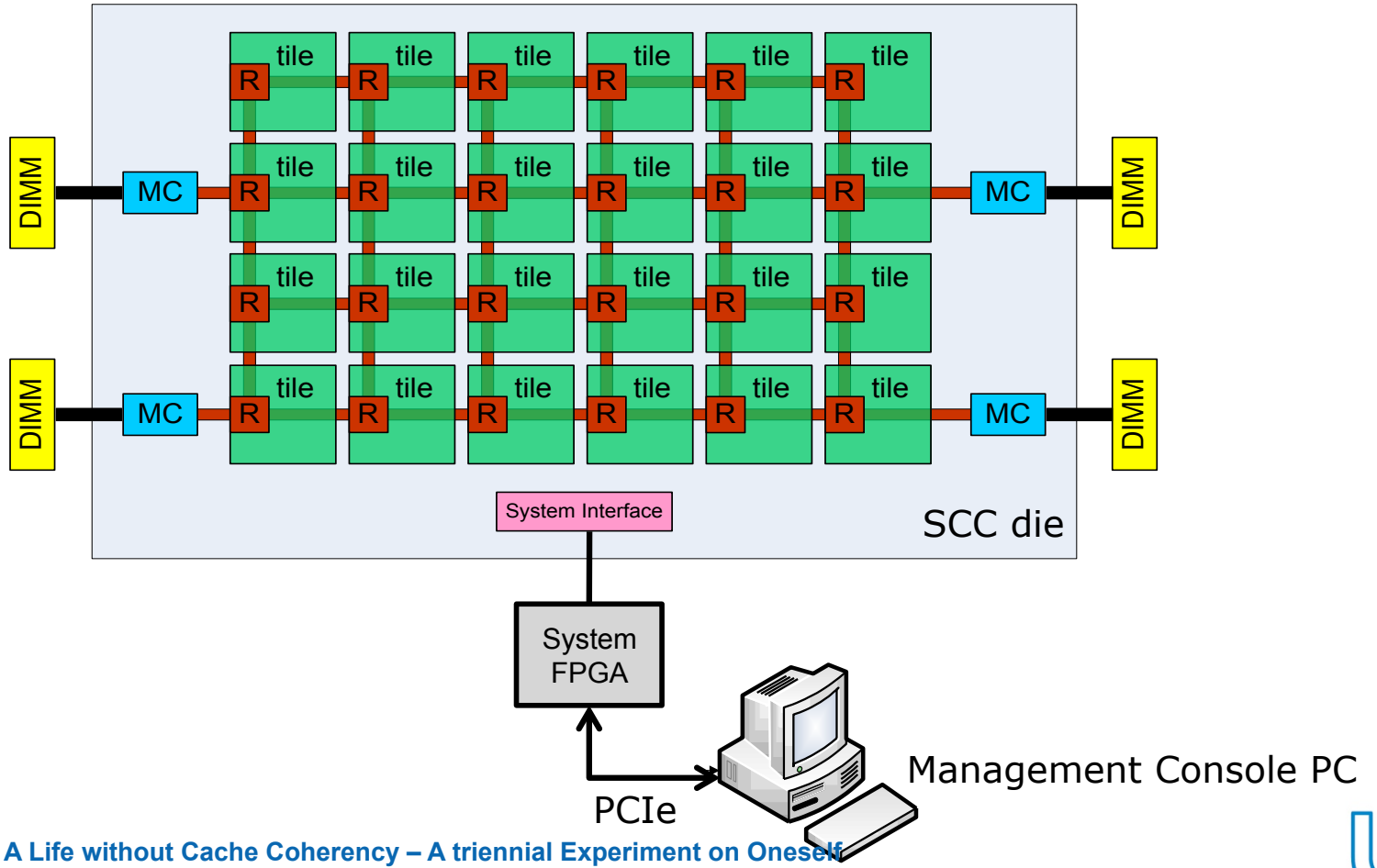






The Intel SCC Many-core Processor

- The SCC is not a standalone system
  - → connected via a FPGA to a frontend system

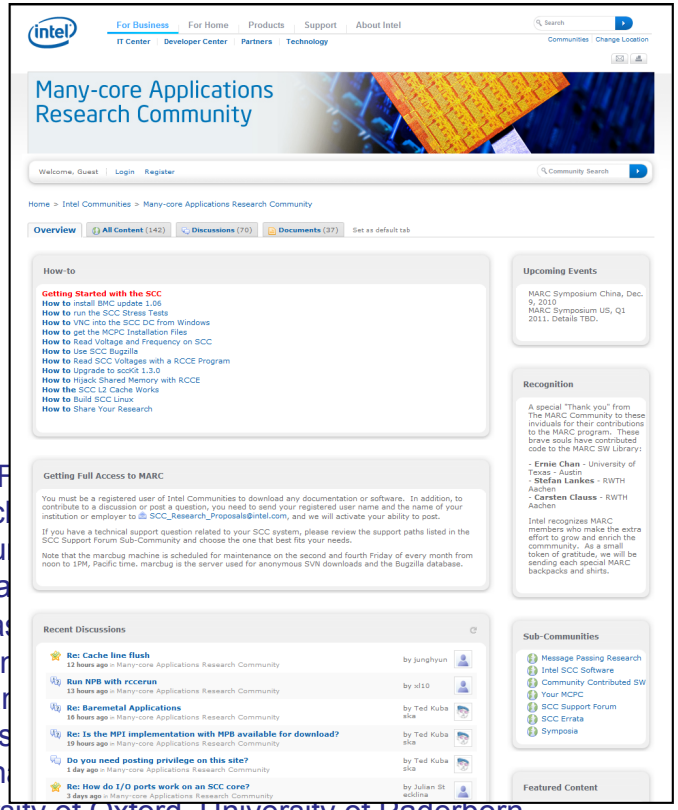


The Intel SCC Many-core Processor

RWTH AACHEN UNIVERSITY

- The MARC Community
  - → Intel Co-Traveler Program for Many-core SW research
- Program Timeline
  - → Stared in 2010
  - → Extended through Dec 2013
  - → Research Transition to Intel® Xeon Phi™ Product…
- Selected Members

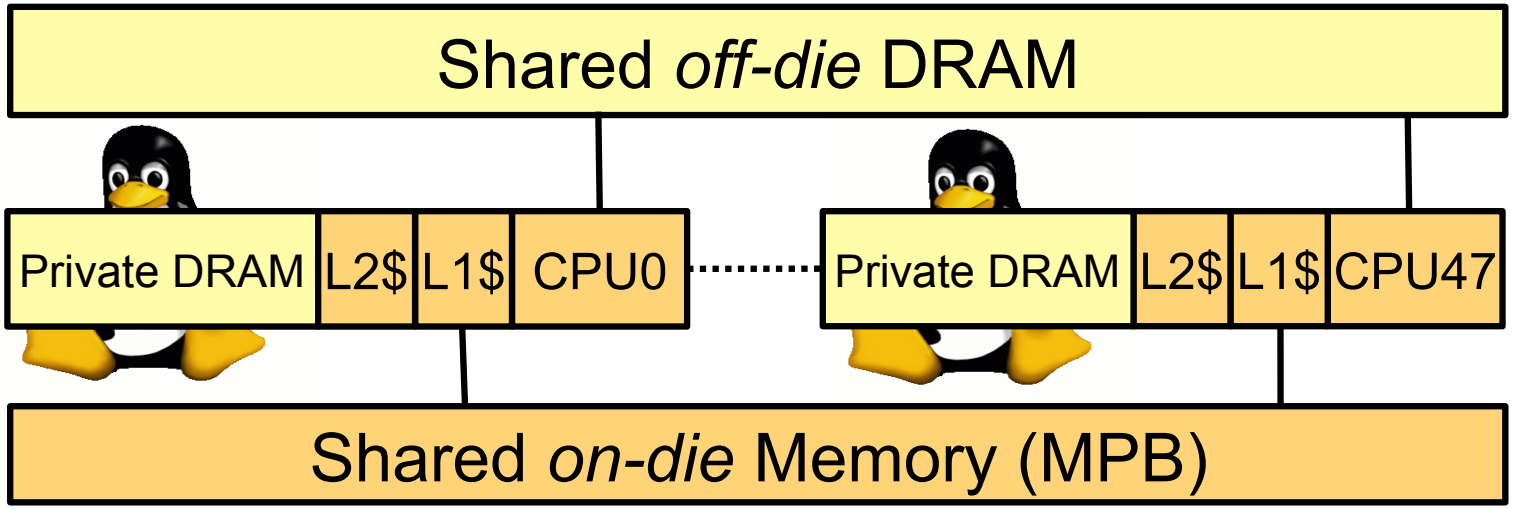
Altreonic, Barcelona Supercomputing Center, ET International, ETH Zurich, F
(University of Crete), Galicia Supercomputing Center (CESGA), Georgia Tech
Institute of Technology (KIT), Lessius-Mechelen - Campus De Nayer, Mercu
University, Reservoir Labs, Rutgers University, RWTH Aachen University, Sa
Stanford University, SUNY Binghamton, Swiss Federal Institute Zurich, Texas
Technical University Berlin, Technical University Braunschweig, Technical Ur
California – Irvine, UNICAMP, University of Amsterdam, University of Auckla
University of Bologna, University of Cambridge, University of Cyprus, Univers
University of Glasgow, University of Hertfordshire, University of Illinois Urban
University of Michigan Ann Arbor, University of Missouri Kansas City, University of Oxford, University of Paderborn,
University of Potsdam, University of Texas Austin, University of Toronto, University of Vienna, Vrije Universiteit,
and many others



Memory Model of the SCC

RWTH AACHEN UNIVERSITY

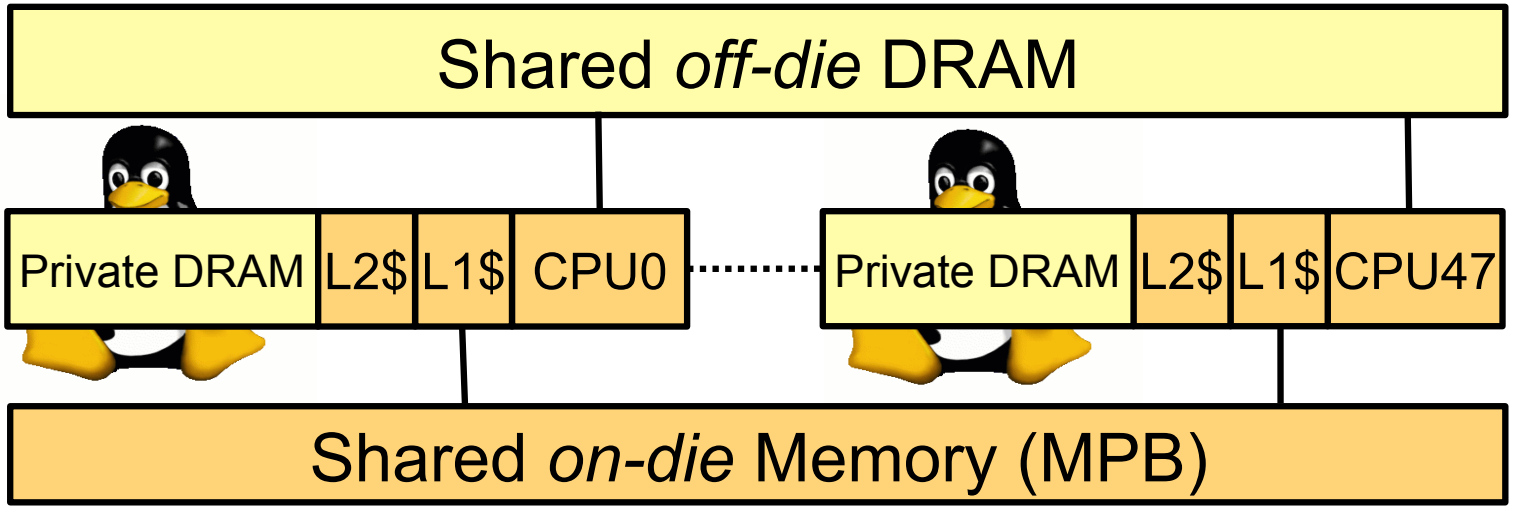
- Strictly No Cache Coherency
  - → Cluster-on-Chip Architecture
- Private off-die DRAM Regions (one per Core)
  - → Caches *enabled*! One *Linux* instance per Core!



Memory Model of the SCC

RWTH AACHEN UNIVERSITY

- Shared / Global off-die DRAM Region
  - → Caches *disabled* by default!
- Shared on-die MPB Regions (cached in L1 / bypass for L2 / fast L1 invalidation via new `CL1INVMB` op)
  - → for message-passing or shared data structures



Cache Characteristics of the SCC

RWTH AACHEN UNIVERSITY

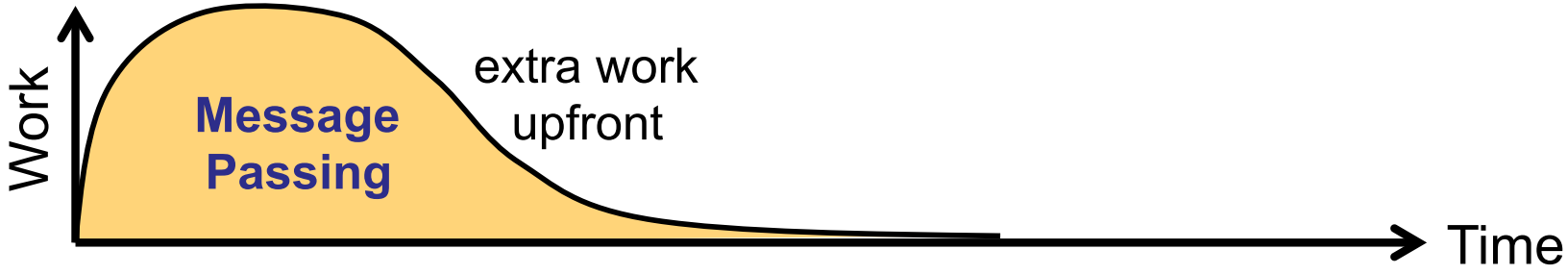
- 16kByte L1 Caches (separate data and instruction)
  - → Write-Back or Write-Through (configurable)
- 256kByte L2 Caches (2x 256kByte per Tile)
  - → 4-way Write-Back
- 32 Byte Cache Line Size
  - → Write-Combine Buffer for Write-Through
- Cache on Read (not on Write) Misses
  - → can lead to “strange” behavior
- Strictly NO Cache Coherency
  - → though cache lines have “MESI” states, no coherence protocol is implemented

Programming the SCC:
Message Passing vs. Shared Memory

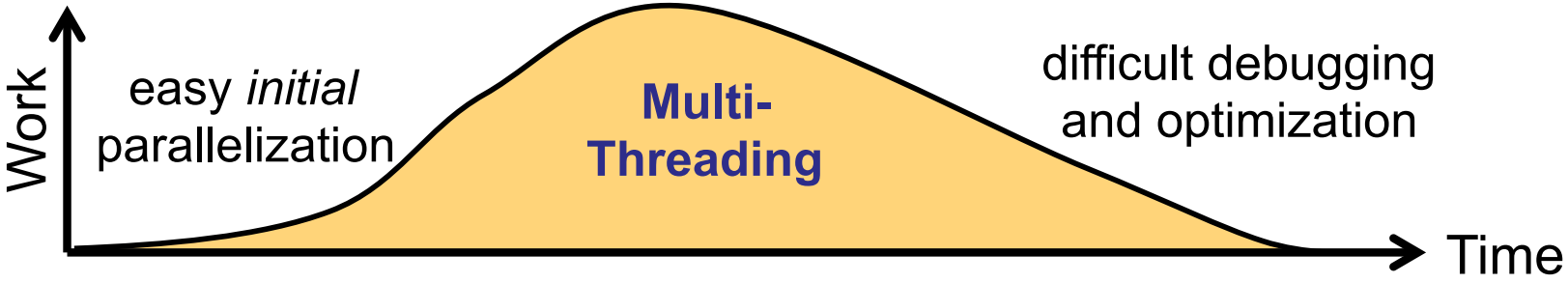
RWTH AACHEN UNIVERSITY

- Is Shared Memory programming easier? (*)

→ Effort for *Message Passing* based Applications:



→ Effort for *Shared Memory* based Applications:

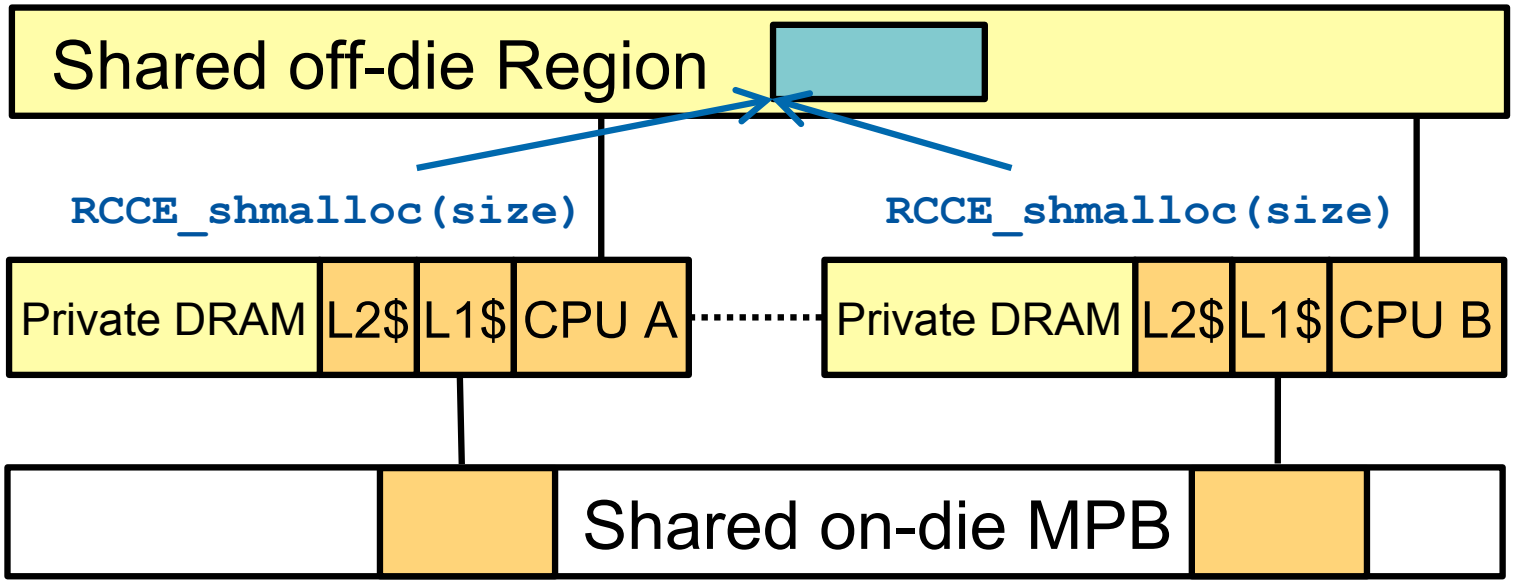


(*) from Tim Mattson “The Future of Many Core Computing”

Programming the SCC:
Shared Memory

RWTH AACHEN UNIVERSITY

- RCCE: The SCC's Native Communication Library
  - → Application Programming Interface (API) for the SCC provided by Intel
- Using *Shared off-die* Memory: `RCCE_shmalloc()`
  - → returns pointers to a new off-die shared region

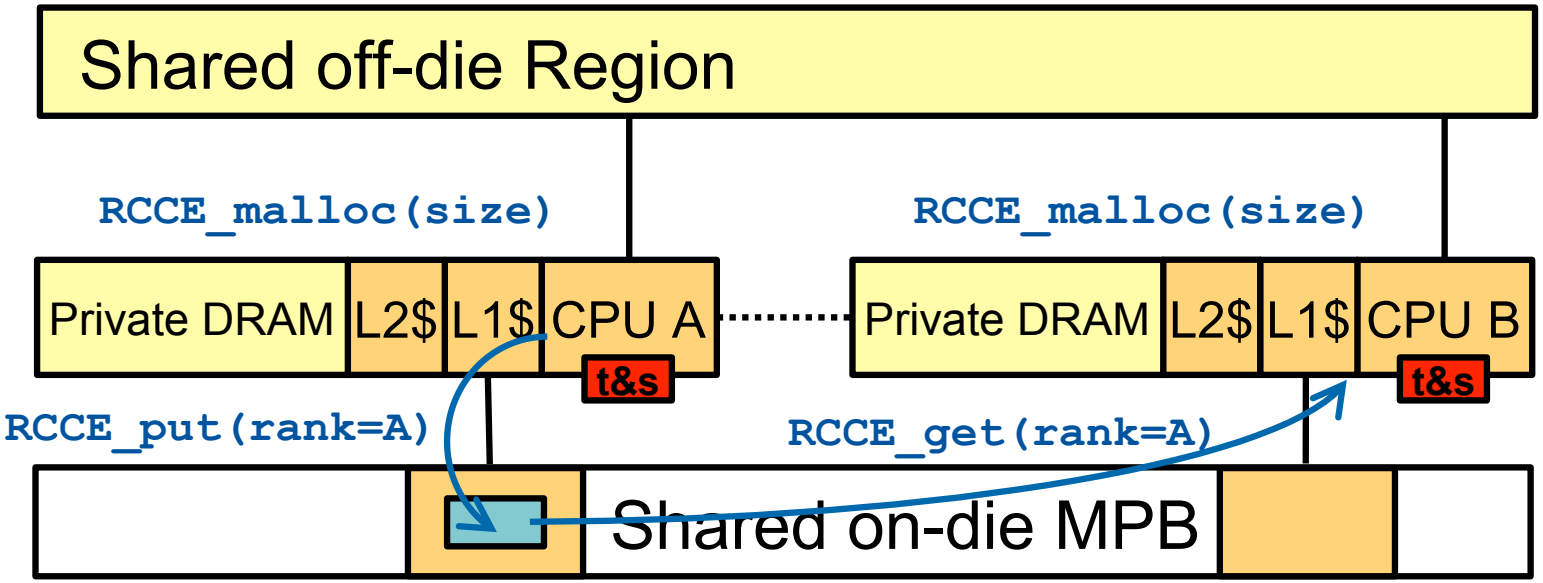


LFBS

Programming the SCC:
Shared Memory

RWTH AACHEN UNIVERSITY

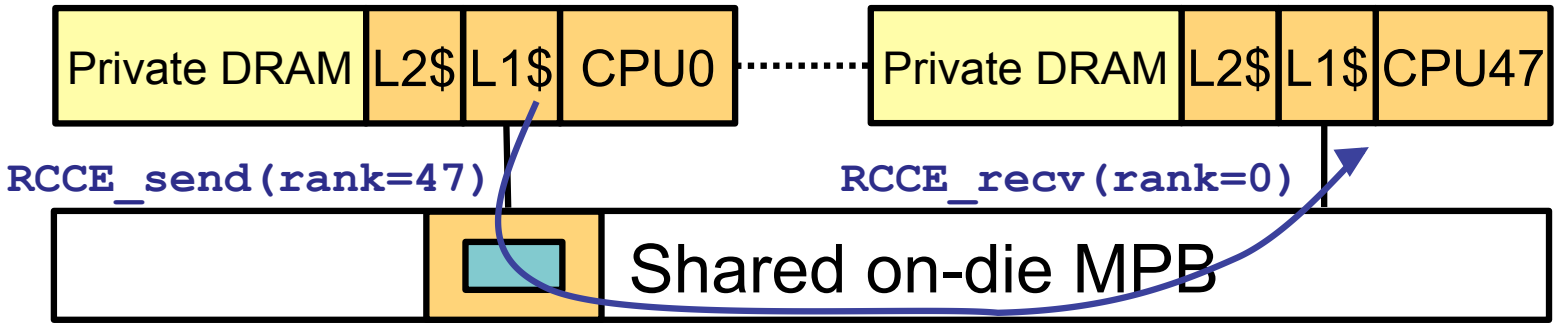
- Using Shared *on-die* MPB Regions: `RCCE_put/get()`
  - → core IDs and pointers are used to address certain MPB regions (→ *Symmetric Memory Model*)
- Additional *Synchronization Registers* are provided
  - → 48x *Test-and-Set* / 96x *Atomic Increment*



Programming the SCC:
Message Passing

RWTH AACHEN UNIVERSITY

- RCCE's Two Sided Communication Interface:
  - → common send/receive functions: `RCCE_send/recv()`

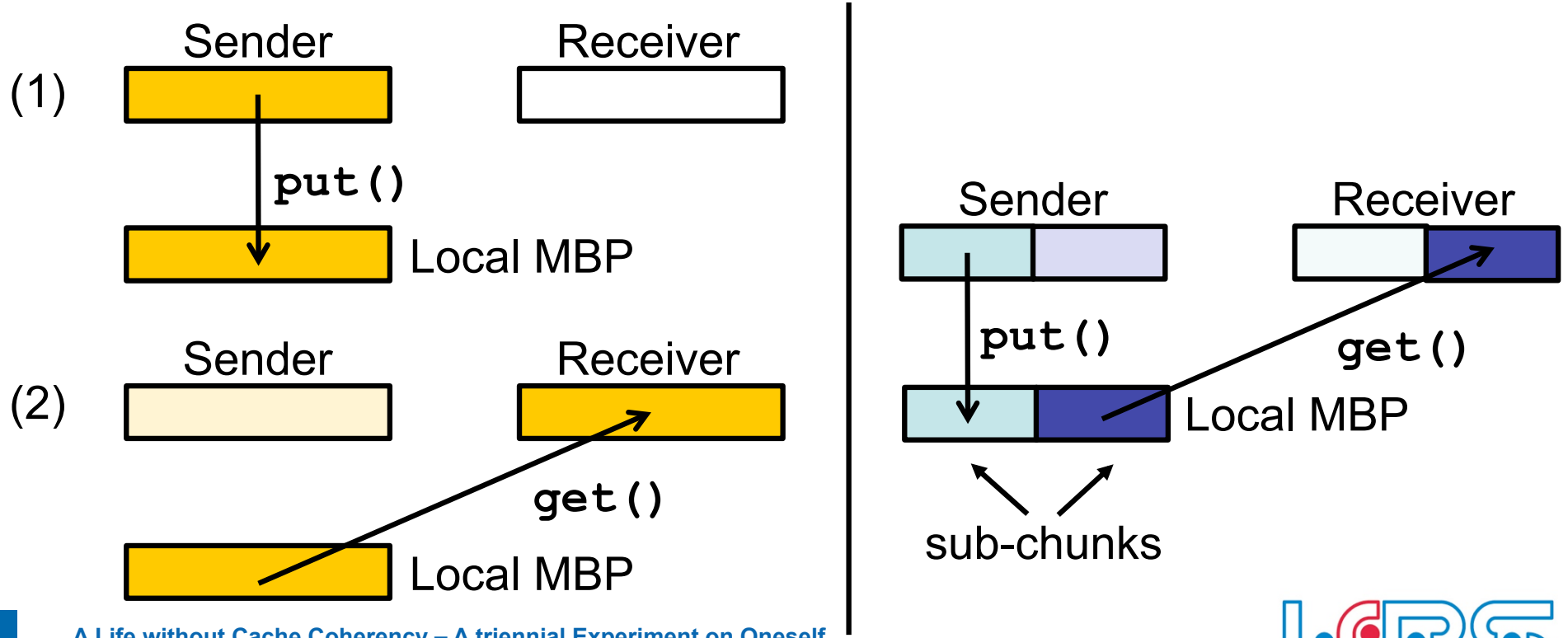


- Communication via on-die MPB
  - → *"local put / remote get"* Protocol
  - → *blocking* functions / *synchronizing* communication
- Extension Library: iRCCE
  - → e.g. *non-blocking* communication functions

    `iRCCE_isend/irecv()`

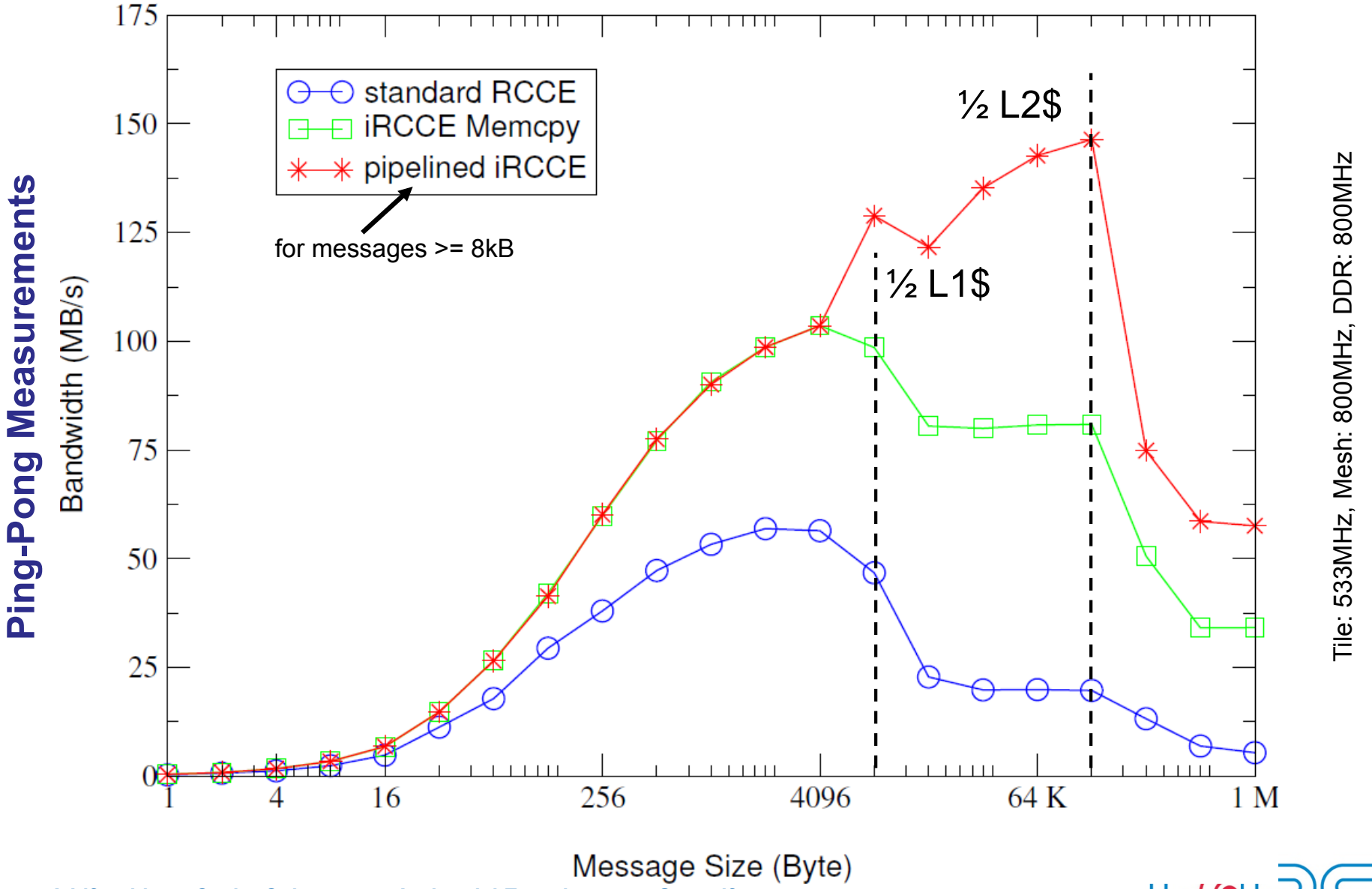
Programming the SCC:
Message Passing

- Pentium-optimized `memcpy()` routines
  - → Prefetching and Loop Unrolling
- Pipelining for large messages
  - → divide local MPB and process in parallel



Programming the SCC:
Message Passing

RWTH AACHEN UNIVERSITY



Programming the SCC:
Message Passing

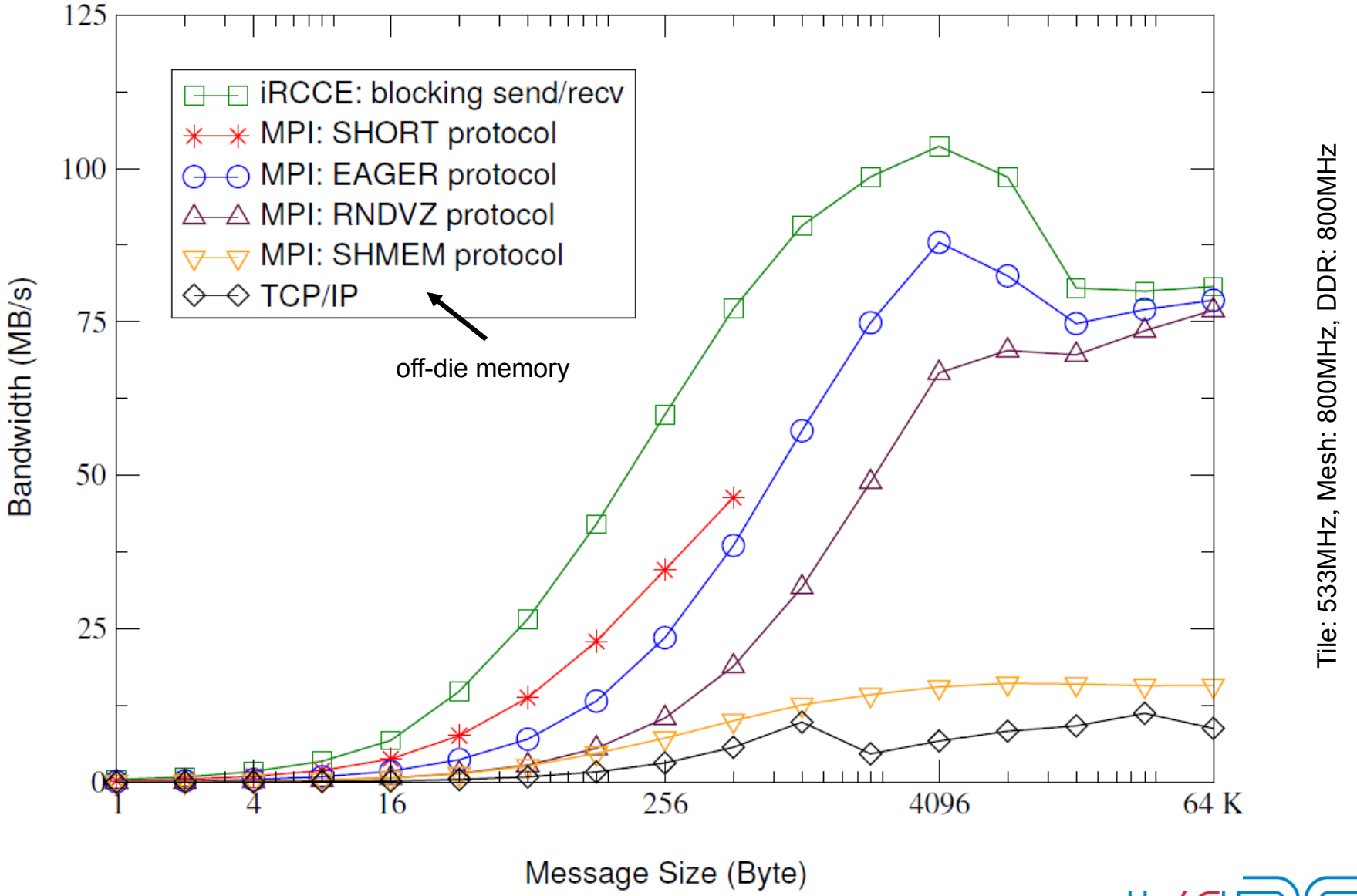
RWTH AACHEN UNIVERSITY

- SCC-MPICH: an MPI for the Intel SCC
  - → support for different communication protocols:
- SHORT Protocol
  - → for short payload and signaling messages
- EAGER Protocol
  - → for mid-size messages (expected and unexpected)
- RENDEZVOUS Protocol
  - → for synchronous as well as for large messages
- SHMEM Protocol
  - → utilizes the shared off-die memory
- TCP/IP based Protocol
  - → for communication with the “outside world”

Programming the SCC:
Message Passing

RWTH AACHEN UNIVERSITY

Ping-Pong Measurements



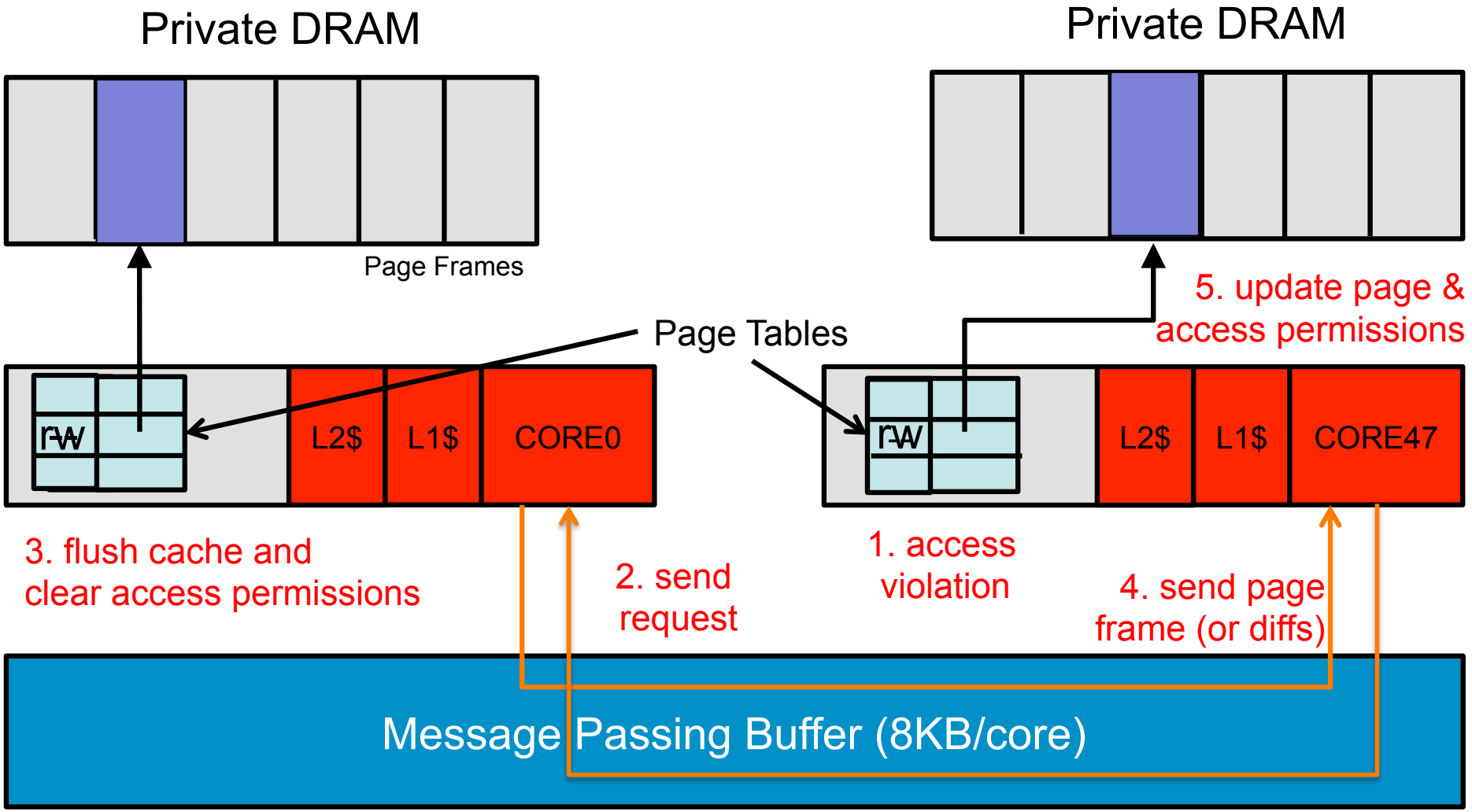
Tile: 533MHz, Mesh: 800MHz, DDR: 800MHz

Motivation in Revisiting of Shared Virtual Memory Systems

RWTH AACHEN UNIVERSITY

- Established concept
  - → shared memory parallelization on distributed memory systems
- Problems of SVM systems
  - → missing of a standard programming model
    - » most systems define their own extensions to C/C++
  - → uniform view of the memory, but non-uniform view of IO devices
  - → overhead by managing the modifications to the shared pages
    - » e.g. calculation and transferring of changes
- Current interconnects support transparent *remote memory access* (RMA)
  - → data are already shared
  - → usage of SVM systems to realize the consistency model

Common Shared Virtual Memory Systems

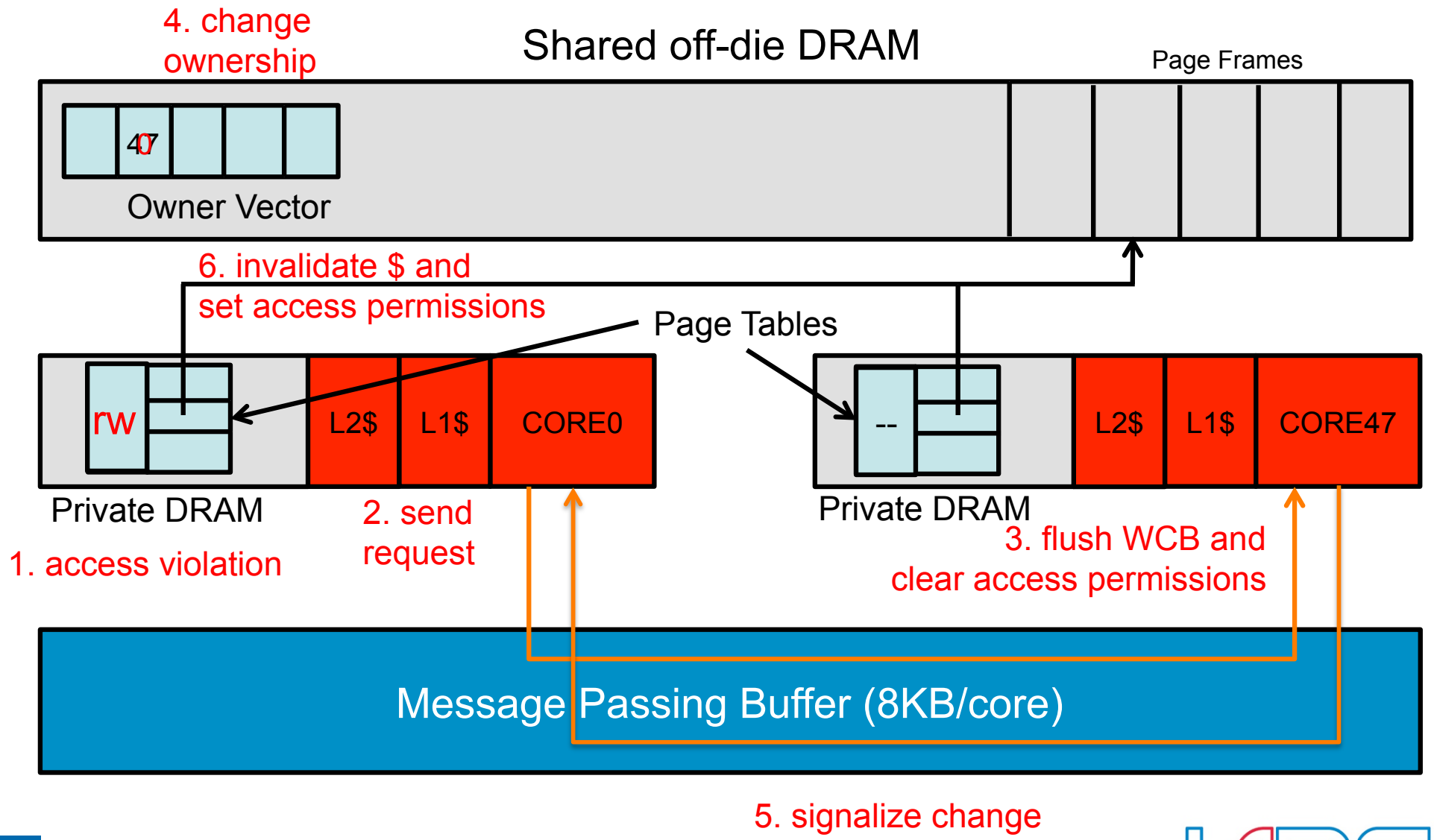


SVM Subsystem

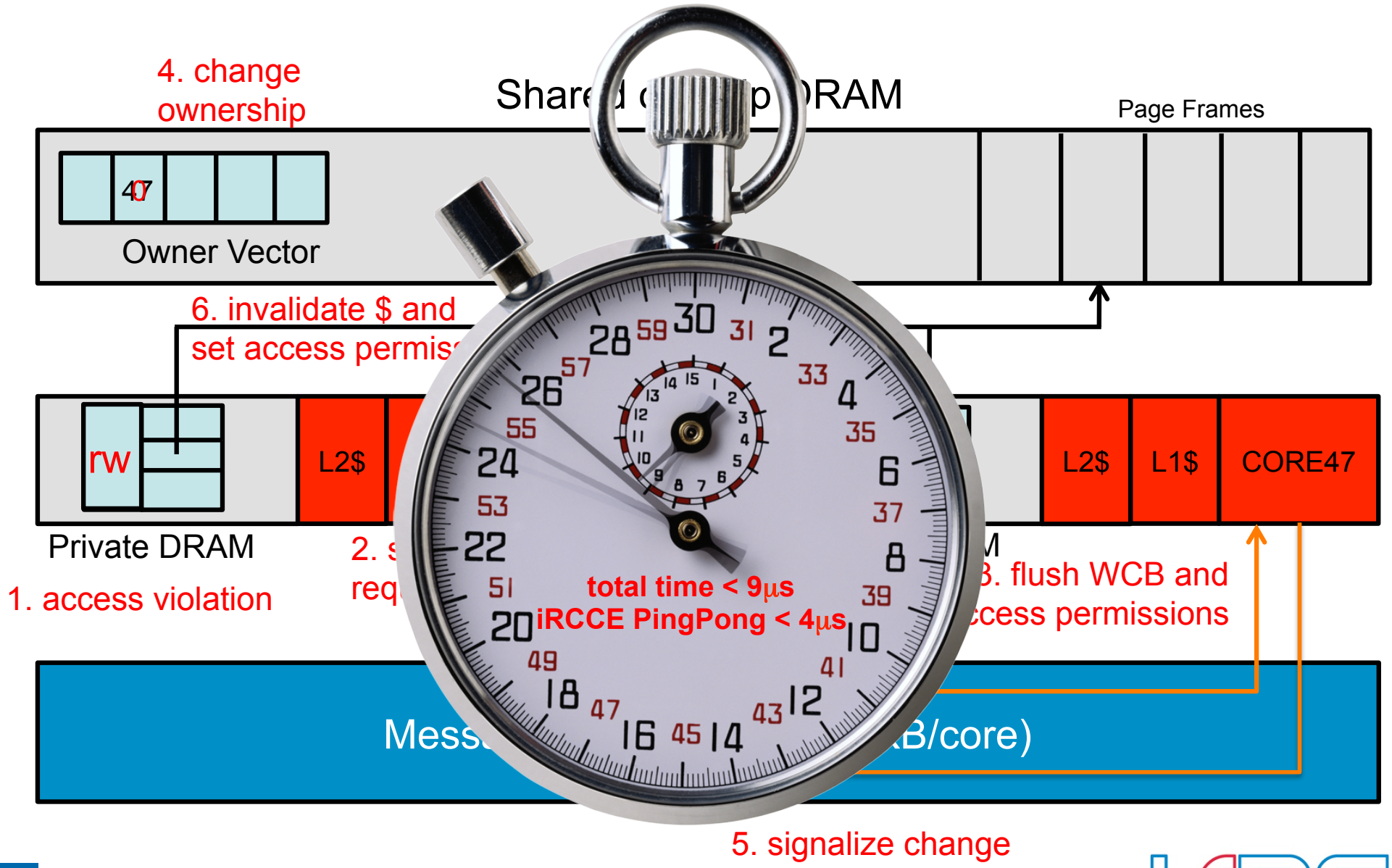
RWTH AACHEN UNIVERSITY

- Boundary conditions for shared data
  - → using the write through caching strategy
  - → only L1$ enabled
  - → tagging of related cache lines
- Consequences
  - \+ support of write combining
  - \+ hardware support for invalidation of L1 entries
  - – no support of L2$
- Requirements for the OS
  - → full asynchronous inter-kernel communication layer
    - » realized as mailbox system
    - » extension to (i)RCCE

Strong Memory Consistency

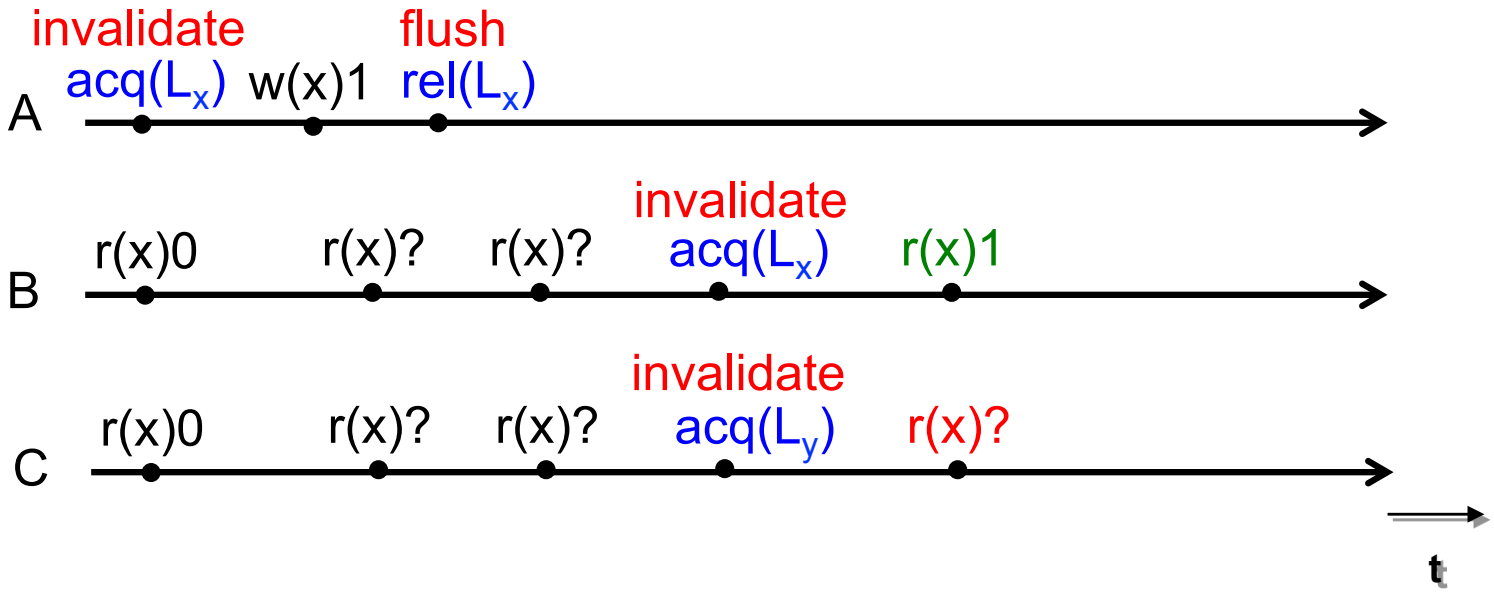


Strong Memory Consistency



Lazy Release Consistency

- Weaker memory models increase performance
  - → reduce the number of inter-core communications
  - → e.g. Lazy Release Consistency
    - » guarantee that the lock owner receives the correct value
    - » reading without holding the lock returns an unspecified value



Increasing Impact of Data Affinity

RWTH AACHEN UNIVERSITY

- On which memory bank does the allocated memory reside?
  - → support of the strategy *Affinity-On-First-Touch*
  - → using NUMA-aware algorithms avoids the memory wall

- Example:

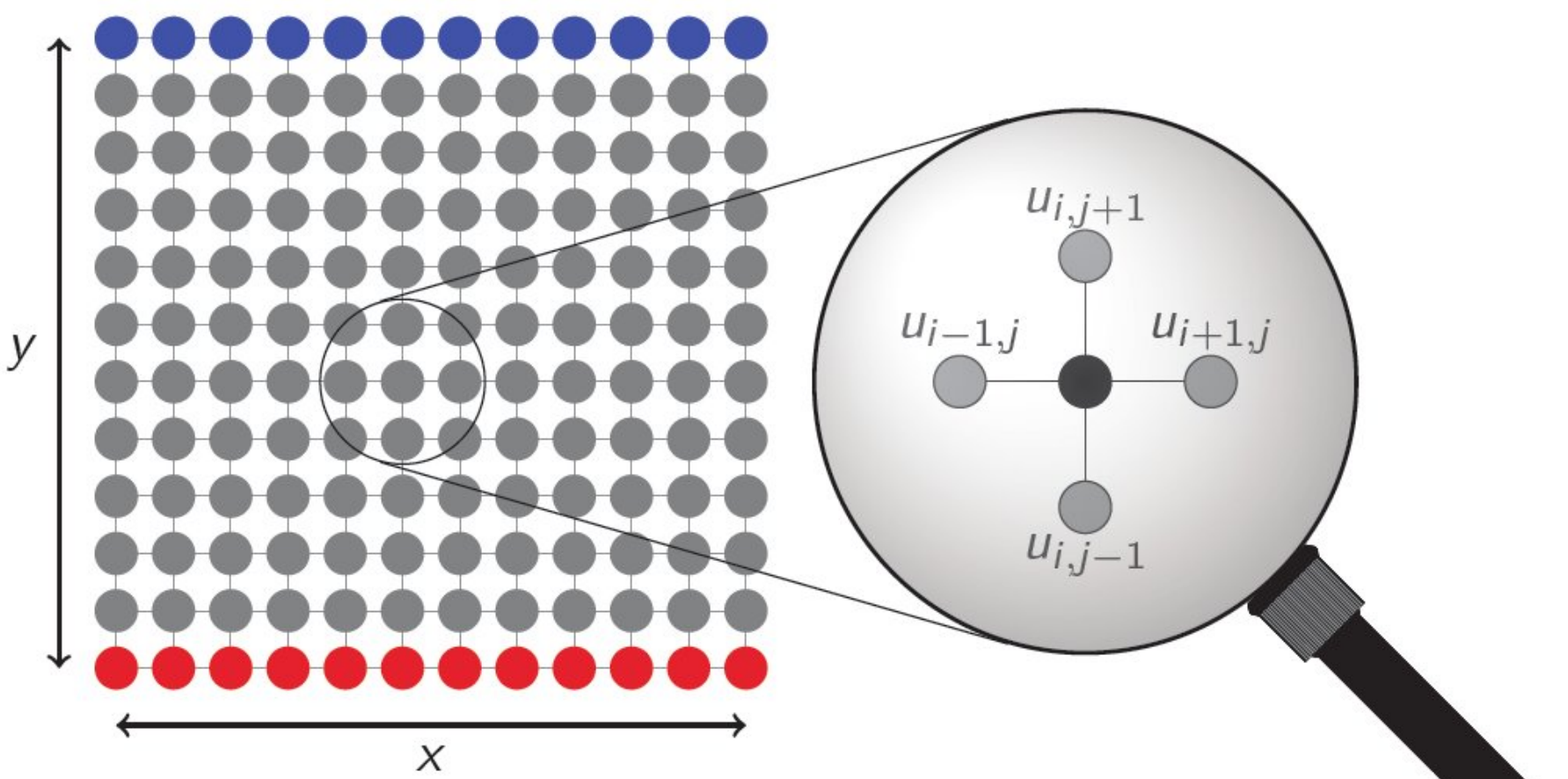
```
int* iArray = (int*) svmalloc(sizeof(int)*SIZE);
for(i=0; i<SIZE; i++)
  iArray[i] = …;
```

- At which moment will the array be physically allocated?
  - → `svmalloc` reserves a place in the virtual address space
  - → The array initialization physically allocates the memory!

- Where is the data located?
  - → in the memory bank nearest to the (initialization) core

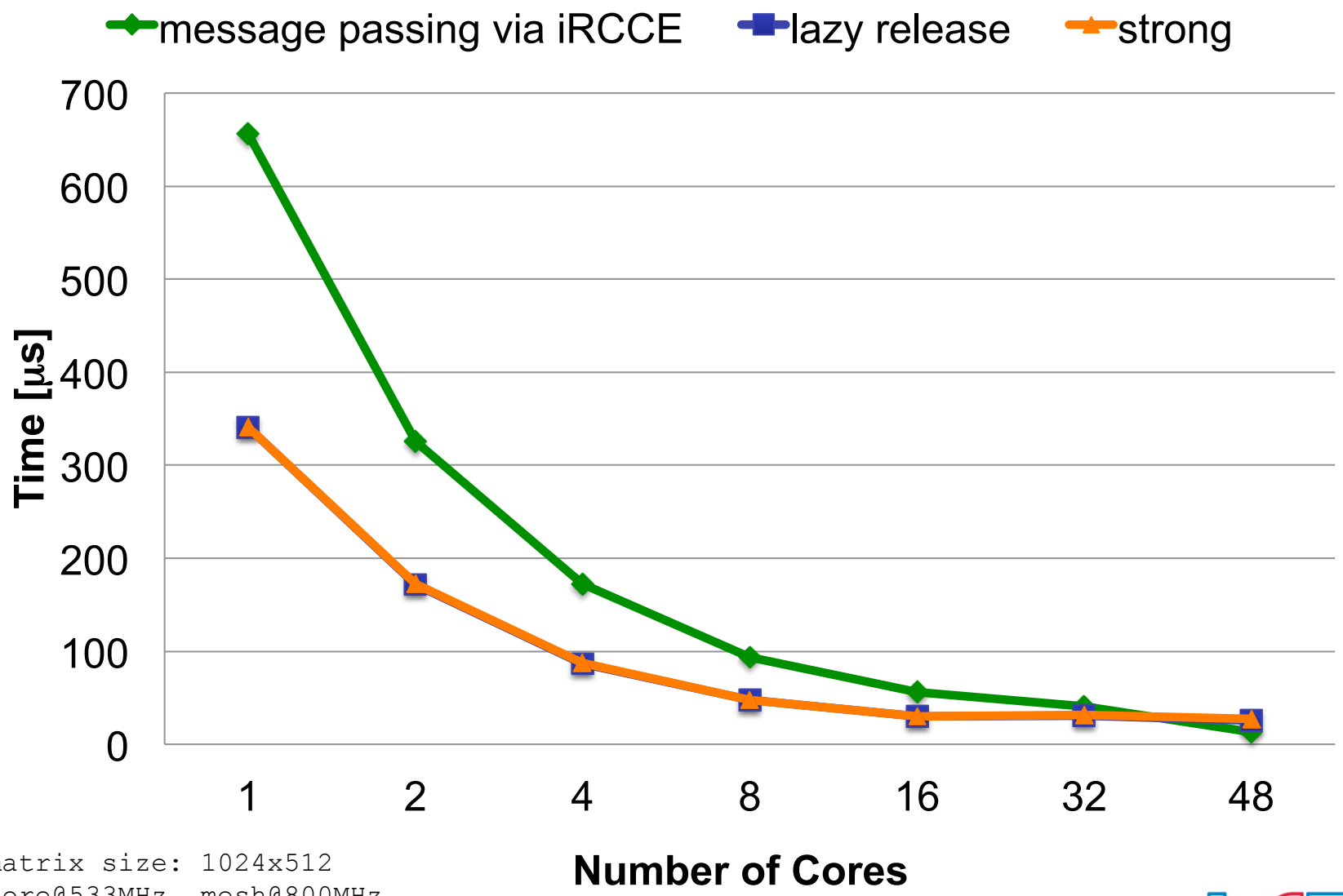
The Two-Dimensional Laplace Problem

RWTH AACHEN UNIVERSITY



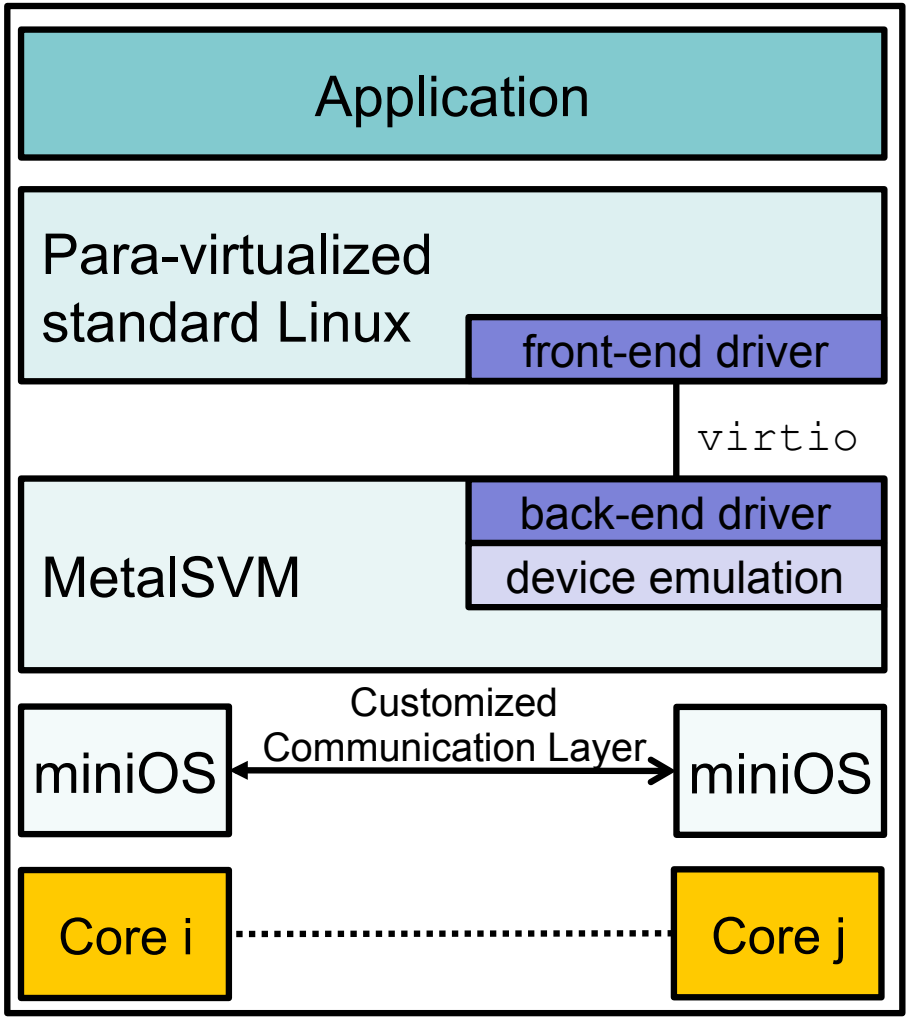
$$u_{i,j}^{k+1} = \frac{1}{4}\left[u_{i-1,j}^{k} + u_{i+1,j}^{k} + u_{i,j-1}^{k} + u_{i,j+1}^{k}\right]$$

Runtime of the Laplace Benchmark



matrix size: 1024x512
core@533MHz, mesh@800MHz

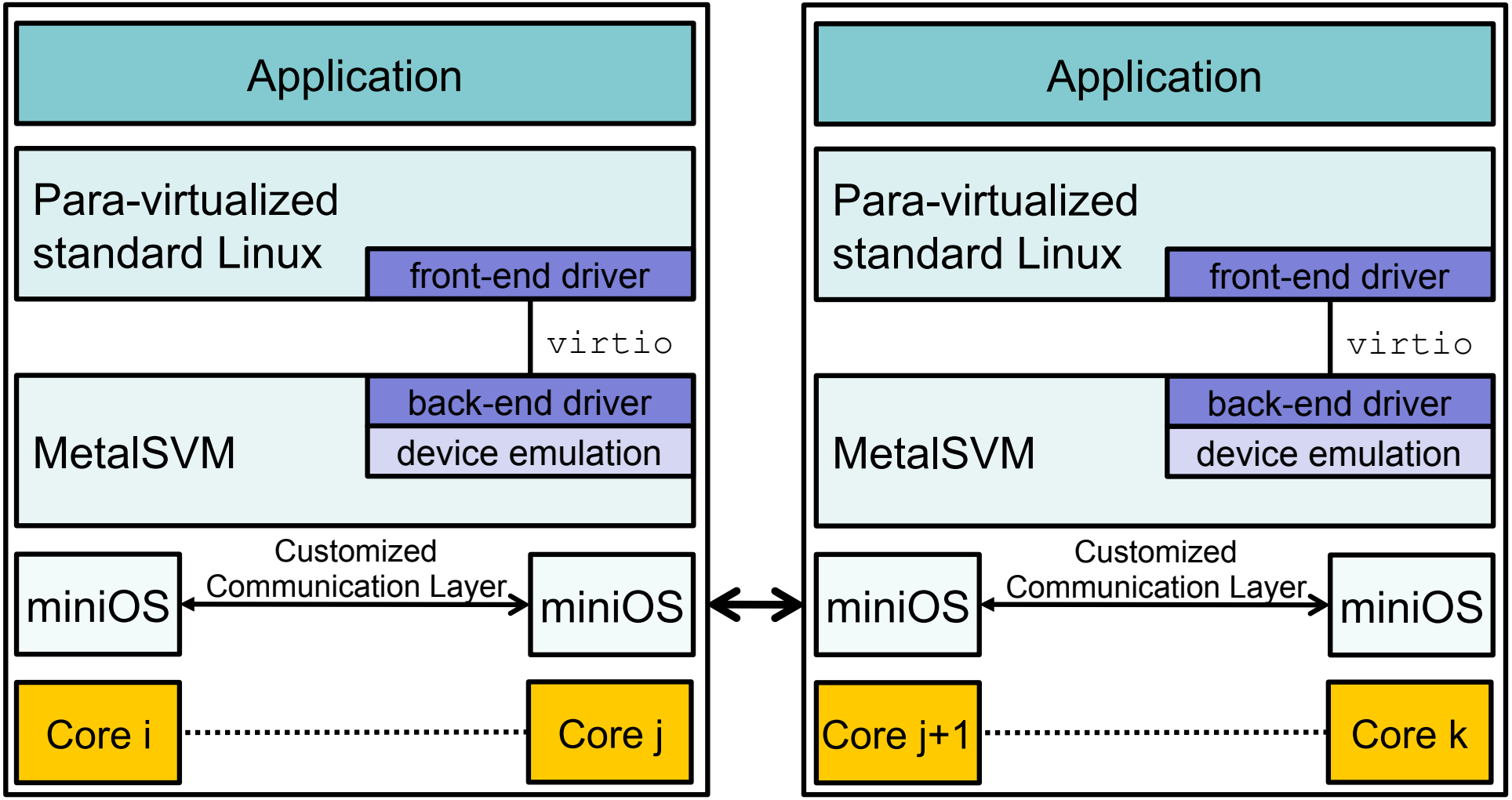
The Project Goals of MetalSVM



- MetalSVM is a bare-metal hypervisor
  - → para-virtualization
  - → inspired by lguest
- Support of a SVM system
  - → transparent (and cacheable) view of the memory
  - → similar to *vNUMA* and *vSMP*
    - » based on traditional clusters
- The virtualization framework virtio provides a common front end for IO devices.

The Project Goals of MetalSVM

RWTH AACHEN UNIVERSITY

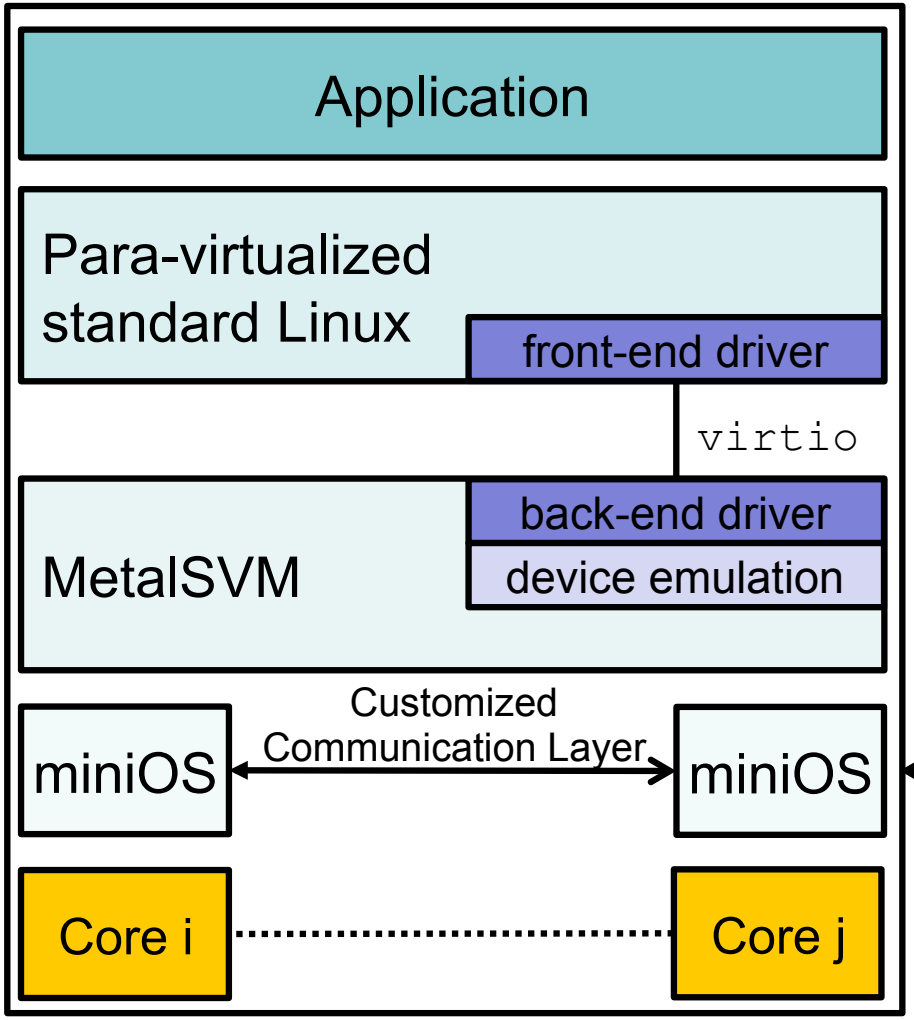


Building of „Coherence Domains“

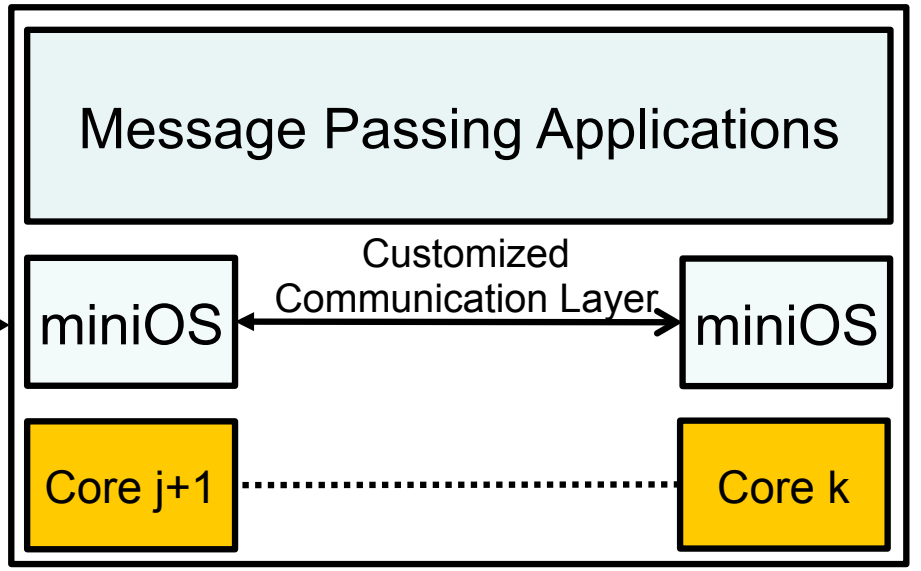
LFBS

The Project Goals of MetalSVM

RWTH AACHEN UNIVERSITY



- Message passing applications as service for the para-virtualized standard Linux

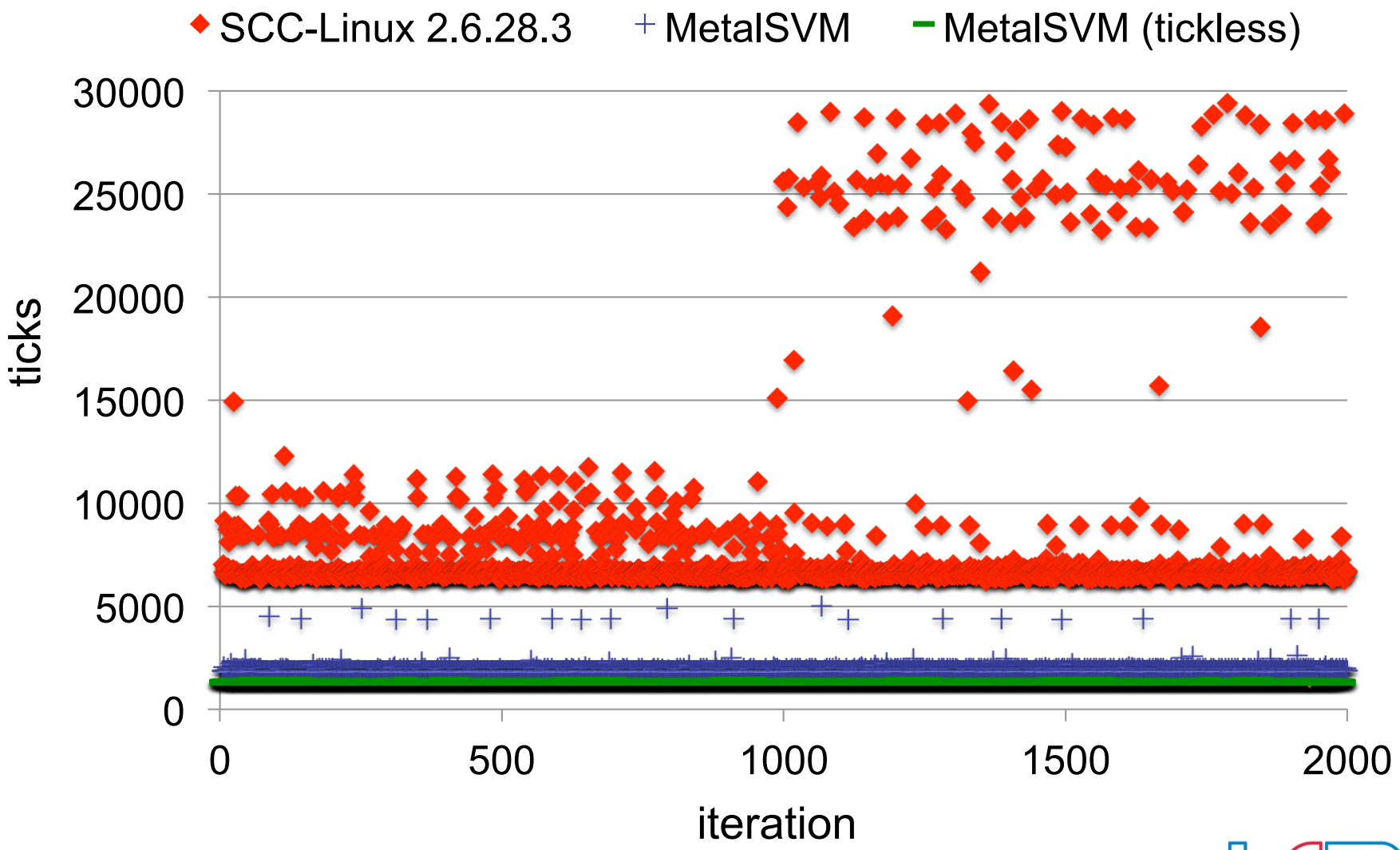


Why do we use a bare-metal environment?

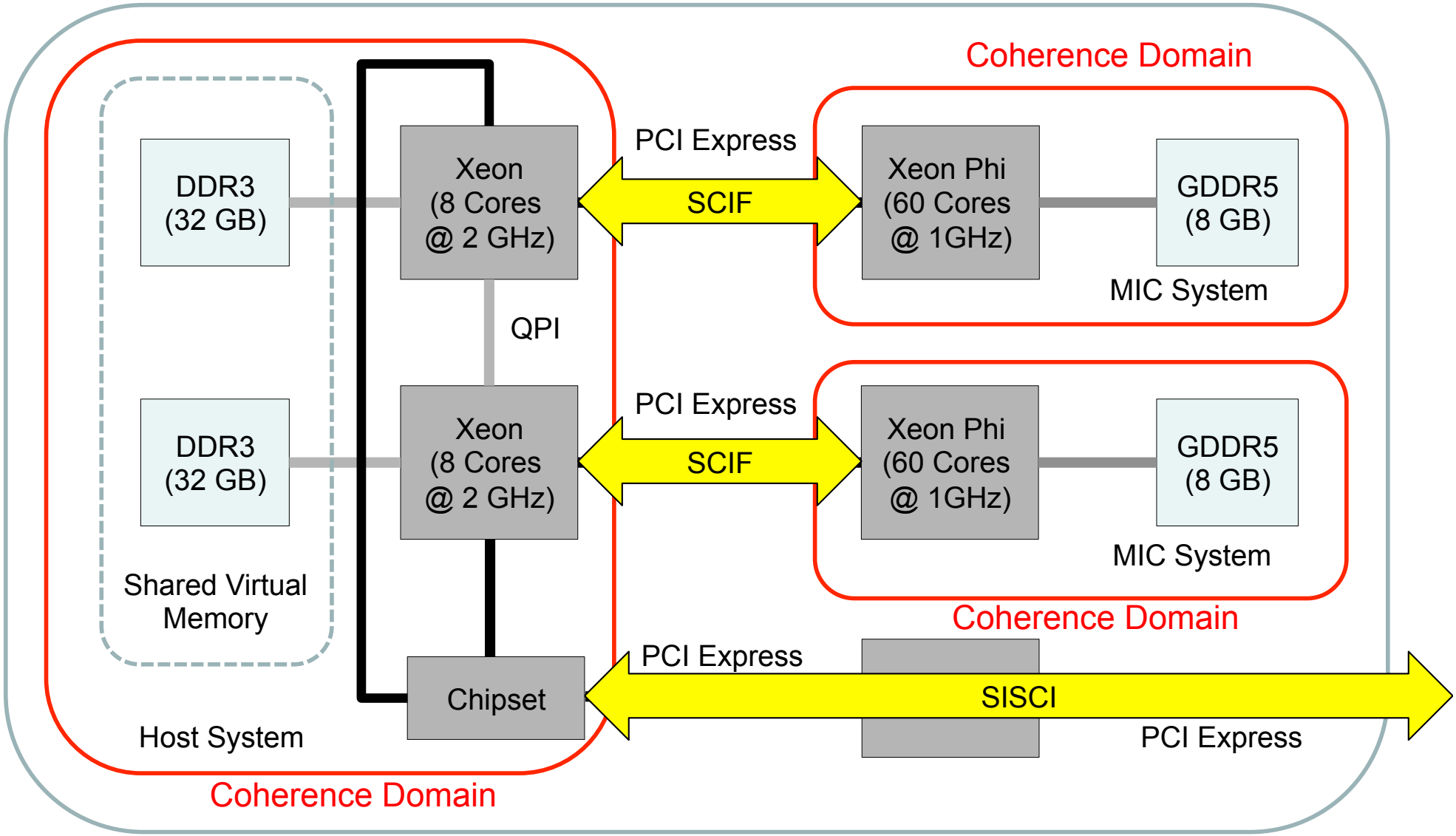
RWTH AACHEN UNIVERSITY

- Advantage
  - → easier to manage / to enhance than common operating systems
  - → lower footprint
    - » Linux: ~3470 Kbytes (image size)
    - » MetalSVM: ~486 Kbytes (image size)
  - → lower overhead
  - → possibility to disable timer interrupts completely
    - » real tickless
    - » no/cooperative multitasking
- Disadvantage
  - → tool support
  - → hard to debug

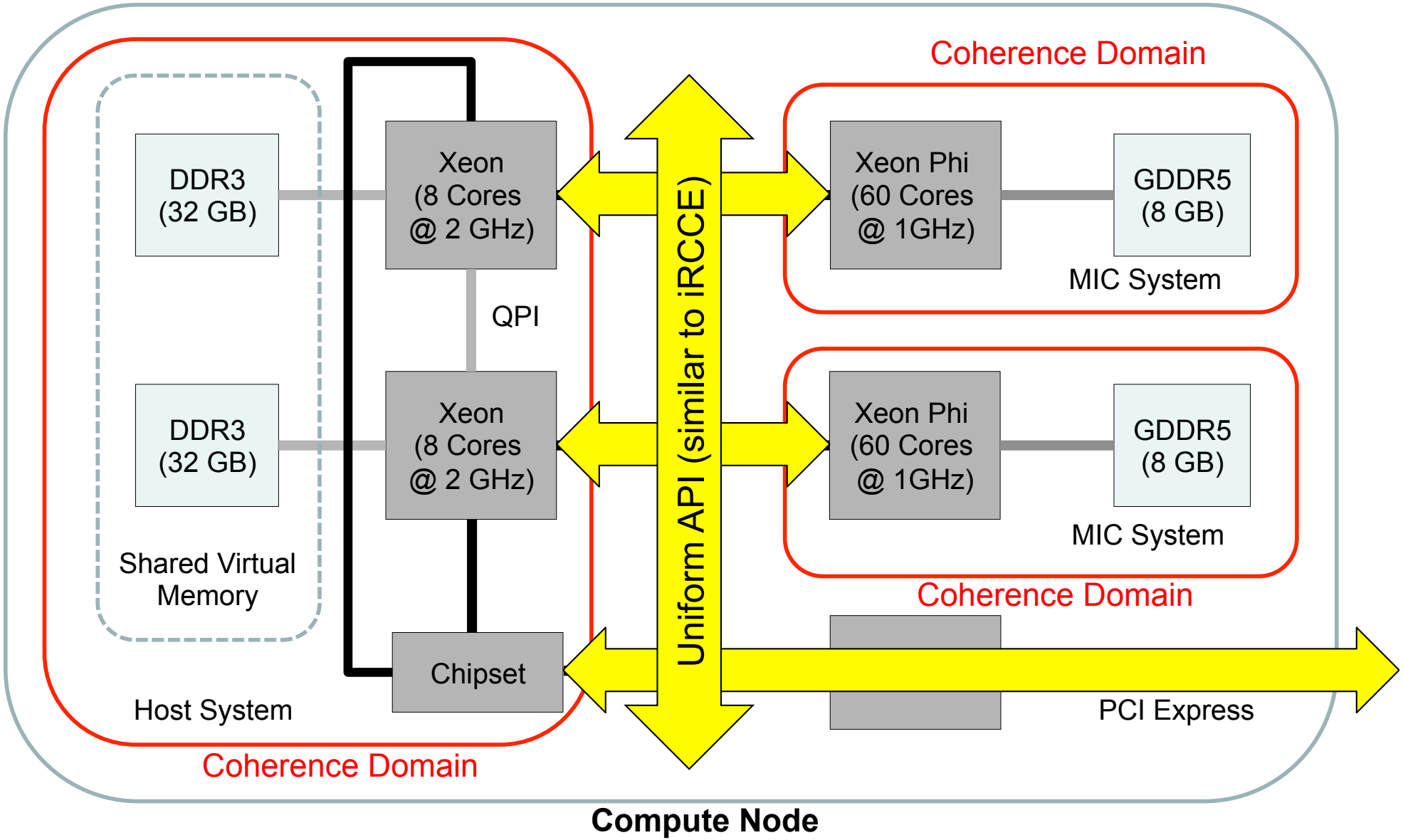
Scheduling Overhead



Xeon Phi Nodes at LfBS



Xeon Phi Nodes at LfBS



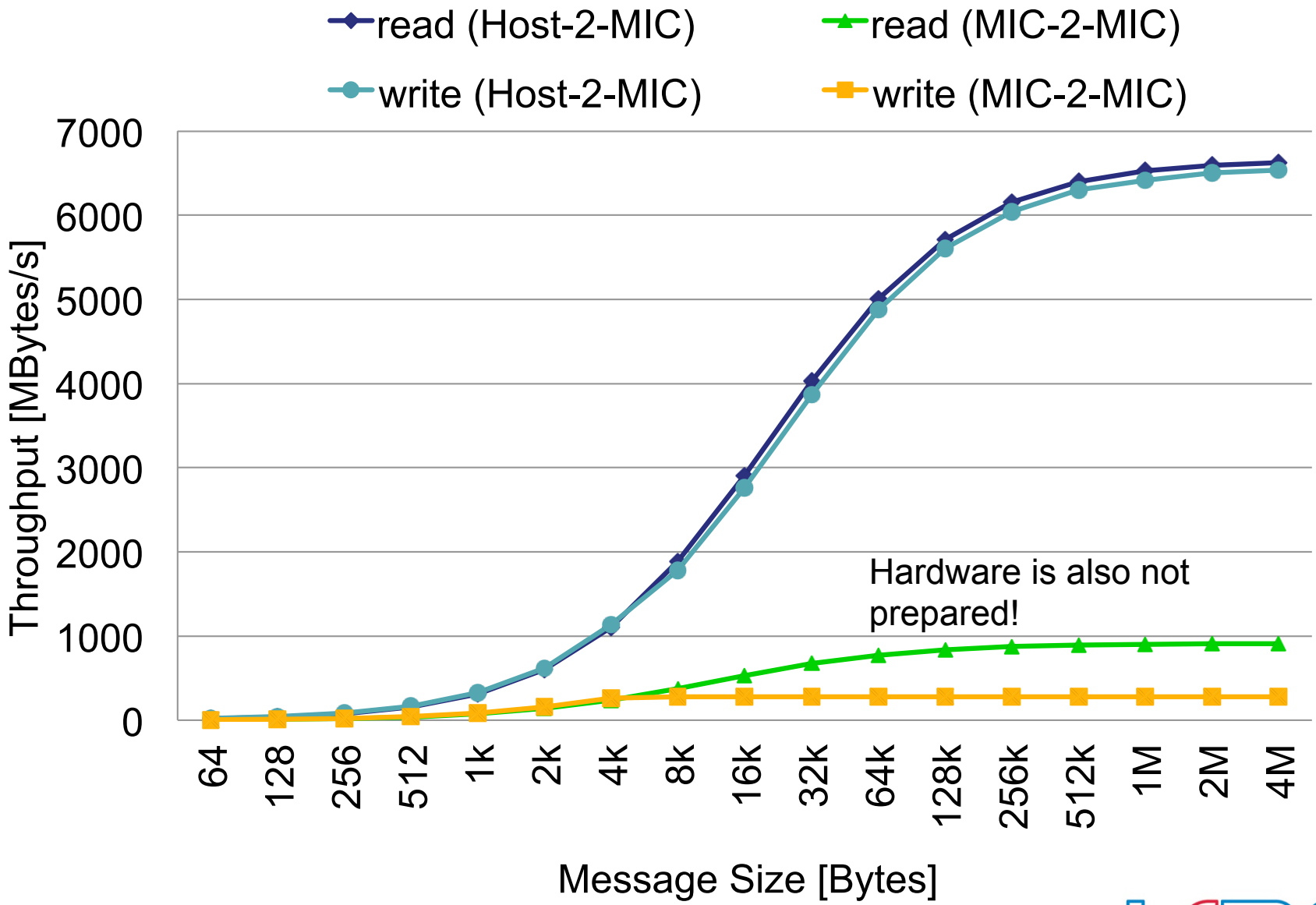
Compute Node

Breaking Boundaries of PCIe-based Clusters

RWTH AACHEN UNIVERSITY

- PCIe Devices are not longer limited to IO
  - → Xeon Phi
  - → GPUs
  - → Clustering via PCIe-Switches
- Device-to-device communication becomes more and more important
- A system wide software interface is missing
  - → Xeon Phi
    - » Symmetric Communication Interface (SCIF)
    - » Mine Yours Ours (MYO)
      - – “similar” to a SVM System
  - → PCIe-based cluster
    - » SISCI, Dolphin’s SuperSockets, etc.
  - → (iRCCE)

One-sided Communication



Conclusions and Outlook

RWTH AACHEN UNIVERSITY

- SCC is a great research vehicle
  - → a life without cache coherency is possible
  - → But
    - » P54C-Pentium is clearly outdated
      - – single core performance is bad
      - – prefer 16 “huskies” instead of 48 “terriers”
    - » the behavior of the write-combining buffers is “suboptimal”
      - – in German: *interessant!*
- Whish list
  - → on-die memory or free configurable caches
  - → More influence on the cache behavior
    - » faster message passing
    - » new possibilities for the system software

References

- T. Mattson, R. van der Wijngaart, M. Riepen, T. Lehnig, P. Brett, W. Haas, P. Kennedy, J. Howard, S. Vangal, N. Borkar, G. Ruhl, and S. Dighe. *The 48-core SCC Processor: The Programmer's View*. In Proceedings of the 2010 ACM/IEEE Conference on Supercomputing (SC10), pages 1–11, New Orleans, LA, USA, November 2010
- T. Mattson and R. van der Wijngaart. *RCCE: a Small Library for Many-Core Communication*. Intel Corporation, Santa Clara, CA, USA, May 2010. Software 1.0-release
- Intel Corporation, Santa Clara, CA, USA. *SCC External Architecture Specification (EAS)*, July 2010. Revision 0.98.
- C. Clauss, S. Lankes, P. Reble, and T. Bemmerl. *Evaluation and Improvements of Programming Models for the Intel SCC Many-core Processor*. In Proceedings of the International Conference on High Performance Computing and Simulation (HPCS2011), Workshop on New Algorithms and Programming Models for the Manycore Era (APMM), pages 525–532, Istanbul, Turkey, July 2011.
- S. Lankes, P. Reble, C. Clauss, and O. Sinnen. *Revisiting Shared Virtual Memory Systems for Non-Coherent Memory-Coupled Cores*. In Proceedings of the 2012 International Workshop on Programming Models and Applications for Multicores and Manycores (PMAM 2012) in conjunction with the 17th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP 2012), pages 45–54, New Orleans, LA, USA, February 2012.
- MARC Community: http://communities.intel.com/community/marc

Thank you for your kind attention!

Dr. rer. nat. Stefan Lankes
Dr.-Ing. Carsten Clauß

Lehrstuhl für Betriebssysteme / Chair for Operating Systems
RWTH Aachen University
Kopernikusstr. 16
52056 Aachen, Germany

www.lfbs.rwth-aachen.de
contact@lfbs.rwth-aachen.de

Backup slides

Kernel Features

RWTH AACHEN UNIVERSITY

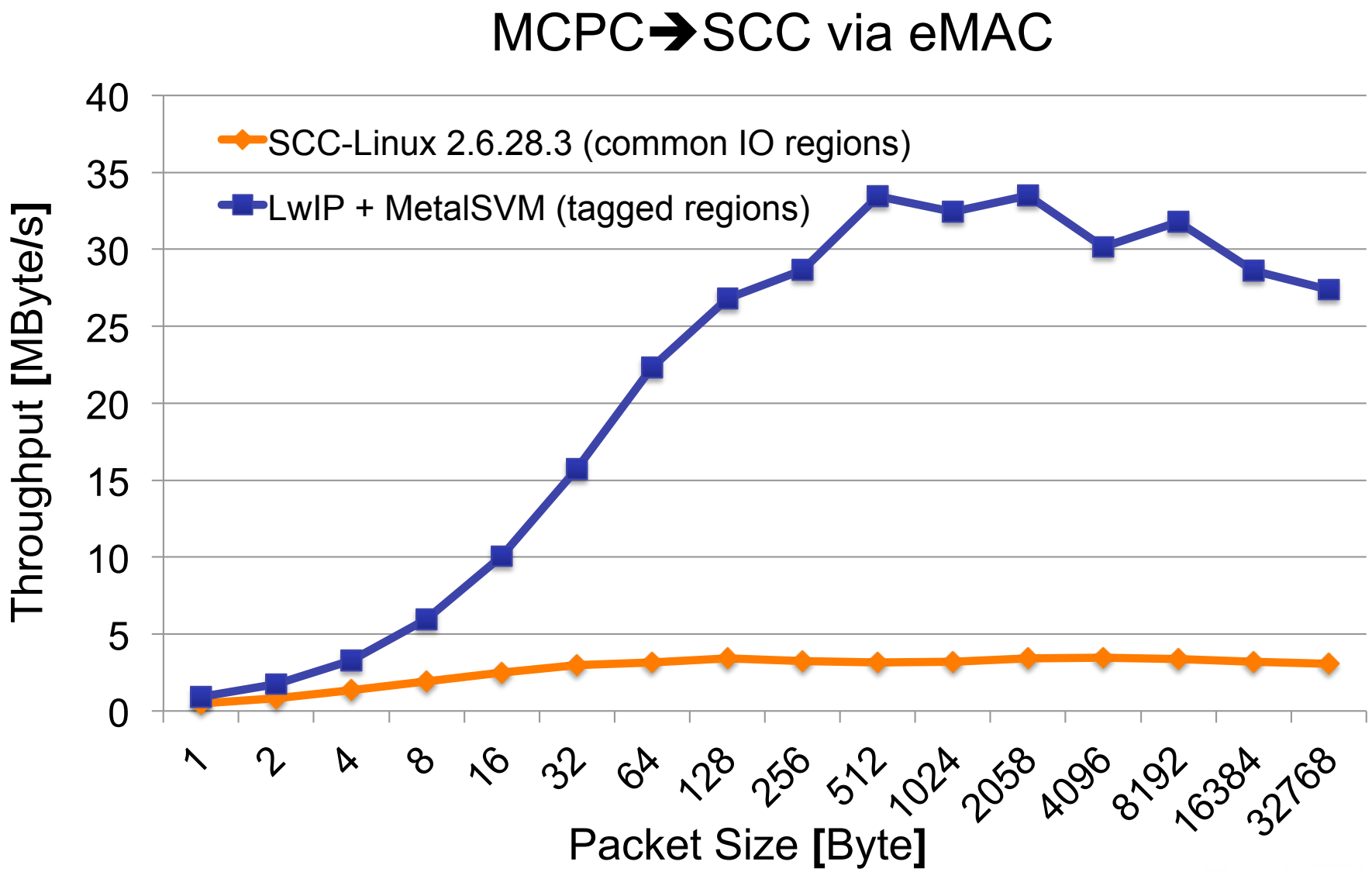
- Using LwIP as lightweight TCP/IP stack
  - → integrated in the kernel
  - → support of the eMAC device, Loopback device, RealTek's NIC RTL8139 and Intel's E1000
  - → support of BSD sockets for user-level task
- Support of SMP-Systems
- Using `newlib` as C Library
  - → easy to support for a self-developed kernel
  - → open-source license
- Full asynchronous inter-kernel communication layer
  - → based on iRCCE
- Use the bare-metal environment!
  - → http://www.metalsvm.org

Advantages of a Fine Grain Cache Control

RWTH AACHEN UNIVERSITY

- New possibilities for device drivers
- Memory mapped IO is a common way to communicate with IO devices
  - → cache is disabled for these regions
    - » guarantees to get the latest results
    - » slow
- On the SCC, tagged cache lines could be invalidated via a new instruction
  - → for the whole tagged region
  - → in one processor cycle
- Device drivers should explicitly invalidate the cache
  - → increase the performance

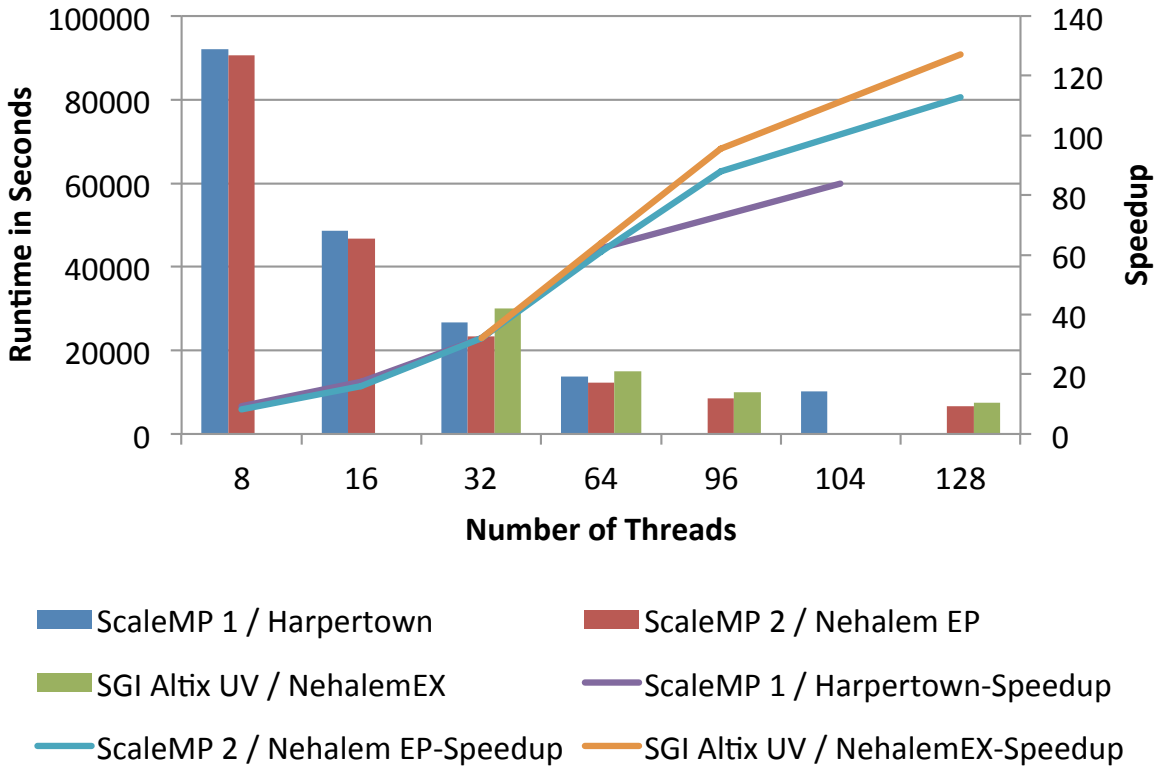
TCP/IP Bandwidth



Analysis of 3D Passive Scalar Field Transported by a Turbulent Flow

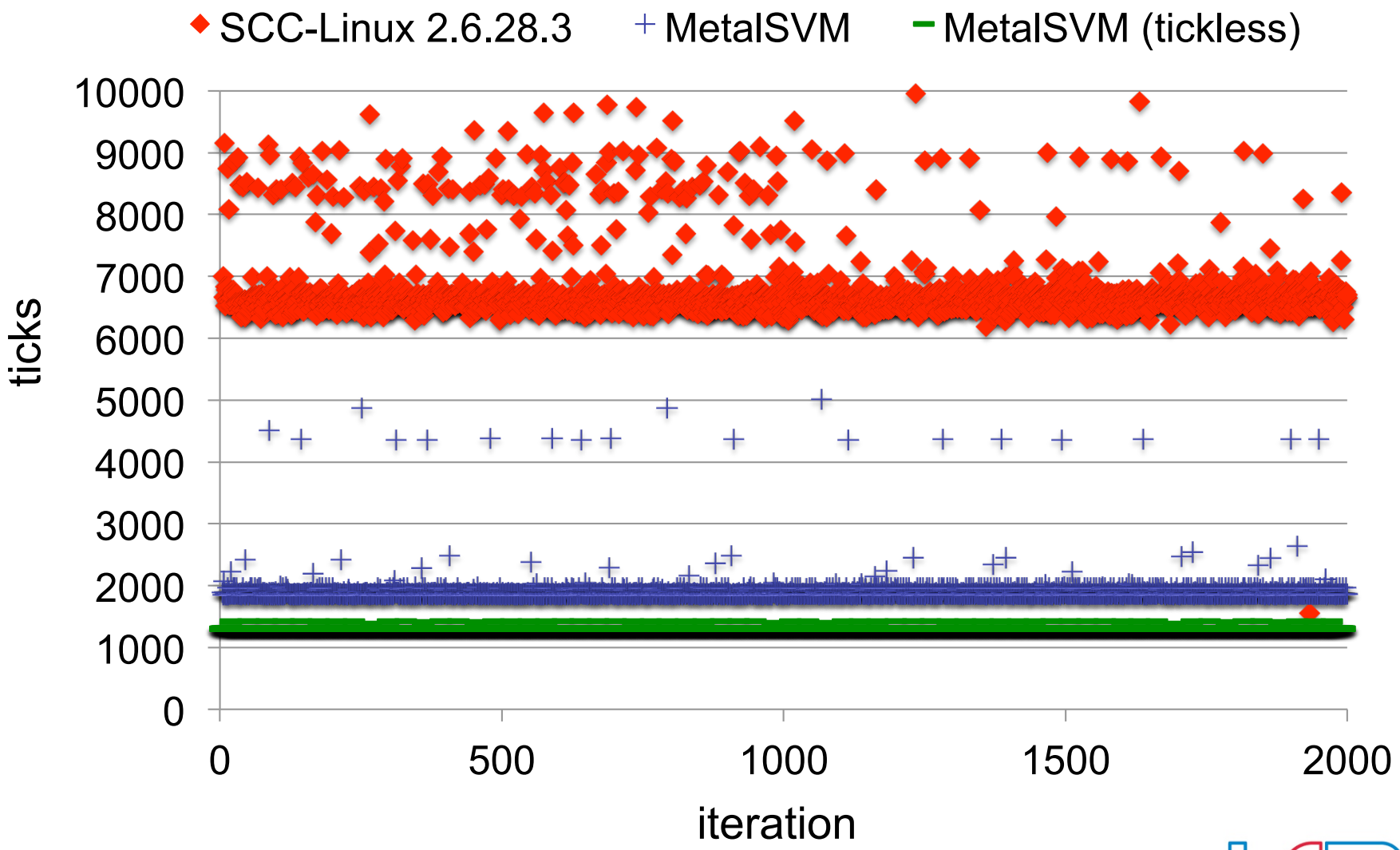
RWTH AACHEN UNIVERSITY

- ~5500 lines of Fortran Code
- Irregular data access pattern
  - → highly imbalanced
  - → not well suited for MPI
- Parallelization for shared memory with OpenMP
- Demand for large memory
  - → target dataset 320 GB
  - → tested 40GB data set
- ccNUMA optimization
- serial runtime > 1 week



Machine	Number of Boards	Processors per Board	Cores per Processor	Total Cores	Processor	Interconnect	Memoy [GB]
ScaleMP 1 / Harpertown	13	2	4	104	Xeon E5420 @ 2,5 GHz	4 X DDR Infiniband	170
ScaleMP 2 / Nehalem EP	16	2	4	128	Xeon E5550 @ 2,67 GHz	4 X QDR Infiniband	320
SGI Altix UV / Nehalem EX	8 (out of 16)	2	8	128 (out of 256)	Xeon X7550 @ 2,00 GHz	NUMAlink	512

Scheduling Overhead

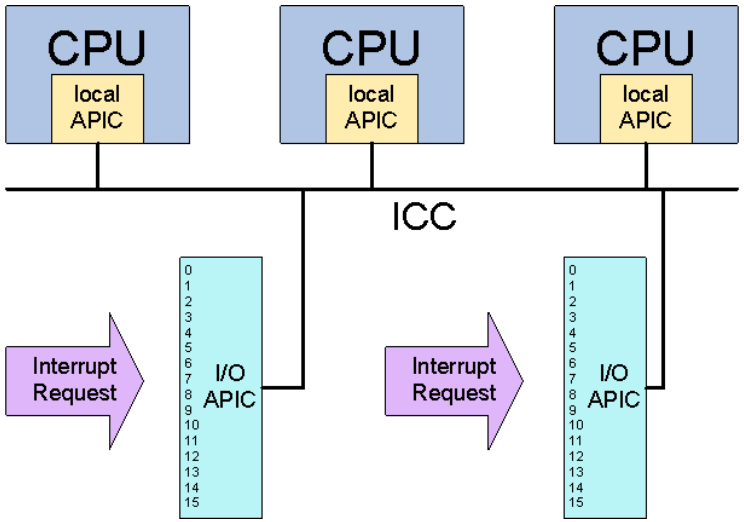


core@533MHz, mesh@800MHz

SMP support on traditional x86

RWTH AACHEN UNIVERSITY

- Each core possesses its own interrupt logic (local APIC)
  - → dealing traps and timer interrupts
- I/O-APICs handle “extern” interrupts
  - → routing interrupts to a specific APIC



- At boot time, only one (boot) processor (BP) is active.
- BP activates the other (application) processor (AP) via an inter-processor interrupt (IPI).
  - → How many processors do exist?
    - » parsing of the multi-processor table
- TLB modifications have to broadcast via IPIs

Possibilities to enable SMP support

RWTH AACHEN UNIVERSITY

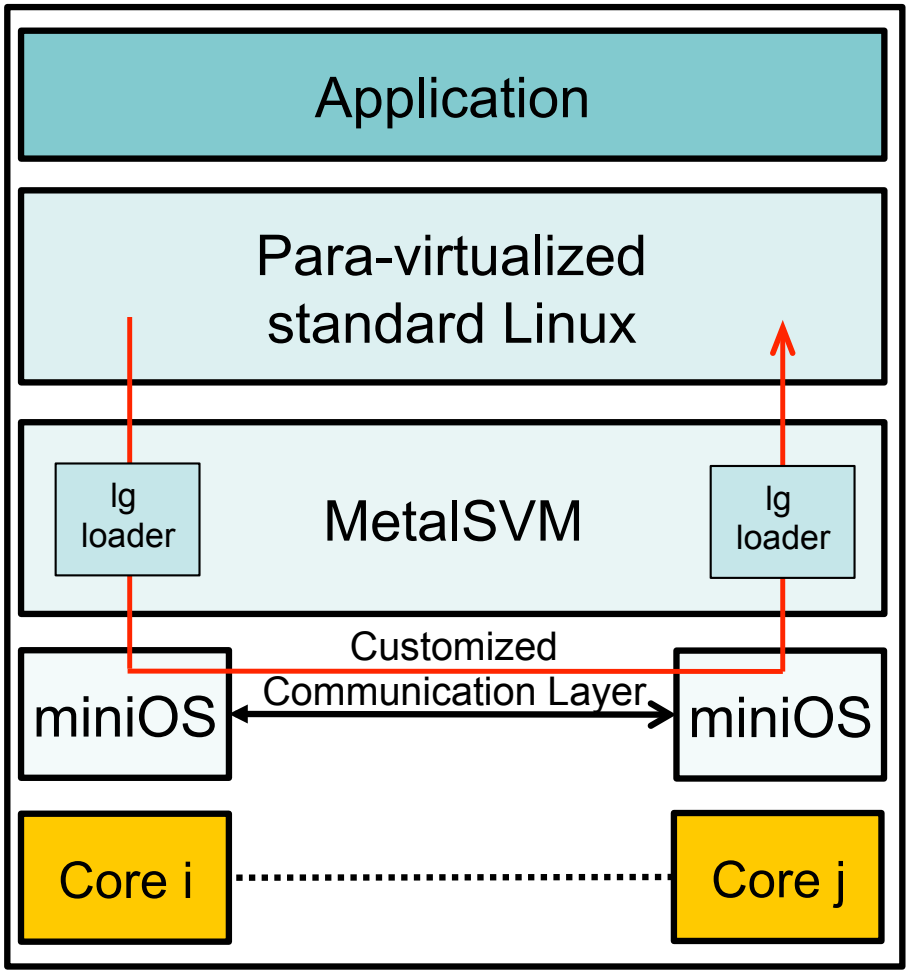
- Emulation of an (local and I/O) APIC
  - → creation of a “virtual” MP table and “virtual” APIC
  - → all-purpose approach
  - ➢ low-level and fault-prone approach
- Overloading of SMP operations (`smp ops`)
  - → Linux embedded all “higher-level” SMP operations in overload-able functions
    - » trigger functions on specific processors
    - » boot up a specific processor
    - » etc.
  - → our selected approach

How many Cores will be activated?

RWTH AACHEN UNIVERSITY

- MP table doesn't exist
- Linux has some nice hooks / kernel parameters
  - → no kernel modification required
  - → `maxcpus=1`
    - » number of activated processors
    - » in this case, the system always comes up
      - – nice for debugging
  - → `possible_cpus=4`
    - » number of possible processors
- Activation of additional application processor
  - → Linux supports hot-pluggable processors
    - » echo 1 > /sys/devices/system/cpu/cpu1/online
    - » triggers an SMP-Op to bring up a CPU

Wake Up of an Application Processor



- Linux triggers a corresponding hypercall
- MetalSVM sends a message to the new processor
- New processor enters idle loop
- Similar techniques to trigger a TLB flush

Avoiding False Sharing

RWTH AACHEN UNIVERSITY

- At boot time, a strong memory model is used
  - → for instance only one processor has access to a page
  - → bad behavior if often used data is located in one page
    - » false sharing hazard
    - » e.g. scheduling information
- vSMP has the same problem
  - → fix it with small patches
    - » e.g. defines a cache size of 4Kbyte

```
--- a/arch/x86/Kconfig.cpu
+++ b/arch/x86/Kconfig.cpu
@@ -303,6 +303,7 @@ config X86_GENERIC
 config X86_INTERNODE_CACHE_SHIFT
        int
        default "12" if X86_VSMP
+       default "12" if LGUEST_GUEST_SCC
        default X86_L1_CACHE_SHIFT
```

  - → part of the vanilla kernel

Reducing the Overhead by use of Read-Only Regions

RWTH AACHEN UNIVERSITY

```
[    0.148355] Device at 31 has size 21
Resetting device console
[    0.157647] Switching to clocksource lguest
[    0.190944] NET: Registered protocol family 2
[    0.192342] IP route cache hash table entries: 1024 (order: 0, 4096 bytes)
[    0.195632] TCP established hash table entries: 1024 (order: 1, 8192 bytes)
[    0.196733] TCP bind hash table entries: 1024 (order: 1, 8192 bytes)
[    0.197621] TCP: Hash tables configured (established 1024 bind 1024)
[    0.198046] TCP reno registered
[    0.198513] UDP hash table entries: 128 (order: 0, 4096 bytes)
[    0.199370] UDP-Lite hash table entries: 128 (order: 0, 4096 bytes)
[    0.199996] Unpacking initramfs...
[    1.414026] Freeing initrd memory: 4096k freed
[    1.480066] platform rtc_cmos: registered platform RTC device (no PNP device found)
[    1.497802] HugeTLB registered 4 MB page size, pre-allocated 0 pages
[    1.710826] Mapping virtqueue 0 addr 1e07000
[    1.712307] Mapping virtqueue 1 addr 1e0a000
Device console OK: offered, accepted
[    1.780059] Mapping virtqueue 0 addr 1e01000
[    1.822759] Mapping virtqueue 1 addr 1e04000
Device net OK: offered 20, accepted 20
[    1.890064] TCP cubic registered
[    1.930286] Registering the dns_resolver key type
[    2.000504] Using IPI No-Shortcut mode
[    2.050404] registered taskstats version 1
[    2.102456] Freeing unused kernel memory: 200k freed
[    2.154218] Write protecting the kernel text: 1684k
[    2.211069] Write protecting the kernel read-only data: 632k
[    2.782374] Switching to clocksource tsc
Starting logging: OK
Initializing random number generator... done.
Starting network...
Starting dropbear sshd: OK

Welcome to MetalSVM's guest
mike login:
```