Towards Efficient Collective Operations on the Intel SCC

Darko Petrović, Omid Shahmirzadi, Thomas Ropars, André Schiper

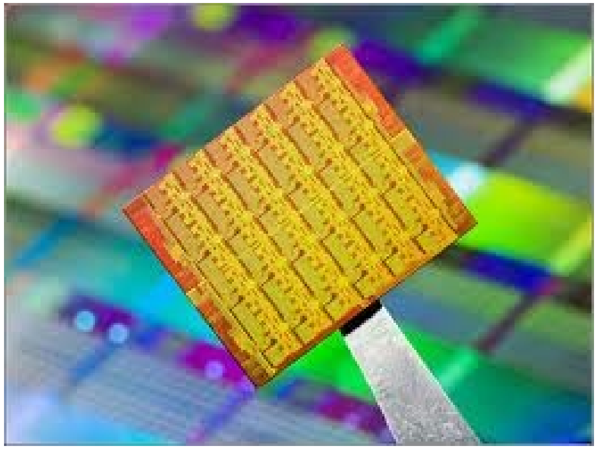
Distributed Systems Laboratory

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

A Trend: Many-Core Chips

- Integrates many loosely-coupled processors on a single chip
  - High-performance Network on Chip (NoC)
  - Hundreds to thousands of small cores

- The main solution to keep increasing the number of flops per watt provided by a single chip [Borkar, 2007]
  - Needed to reach ExaScale



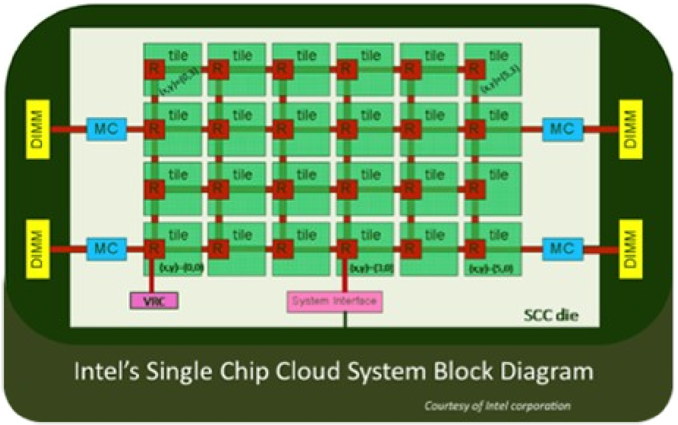
Scalability Issue in the Shared Memory Model

- Hardware cache coherence introduces high overhead at large scale [Mattson, 2010]

- 2 main alternatives:
  - Managing data coherence in software
  - **Adopting the message passing model**
    - The Intel SCC (Single-Chip Cloud Computing)

The Intel SCC: A Message-Passing Many-core

- Specification (features of interest for this talk):
  - 48 general-purpose x86 processors (Pentium-1)
    - 24 tiles (2 cores per tile)
  - A 2D-Mesh Network-on-Chip (6x4)
  - No cache coherence
  - Fast on-chip memory buffers for message passing between cores
- Can be viewed as a distributed system.
  - One process per core
  - Legacy SPMD codes can be run easily
    - A MPI-like interface is easy to provide



Intel's Single Chip Cloud System Block Diagram

Courtesy of Intel corporation

Goal

- **An efficient communication library for the Intel SCC**
  - Studying the communication mechanisms
  - Interface: MPI

- Leveraging the on-chip memory
  - Message Passing Buffers (distributed on the tiles)
  - Accessible by all cores
  - Remote Memory Access (RMA)
    - One-sided put/get interface

Related Work

- Point-to-point communications are well studied
  - RCCE
    - One-sided put/get interface using the MPBs.
    - Two-sided send/recv interface.
      - On top of the one-sided interface
      - Only accessible interface by default
  - iRCCE
    - Provides non blocking primitives
    - Improves performance using double-buffering
  - RCKMPI
    - Provides multiple solutions based on the message size

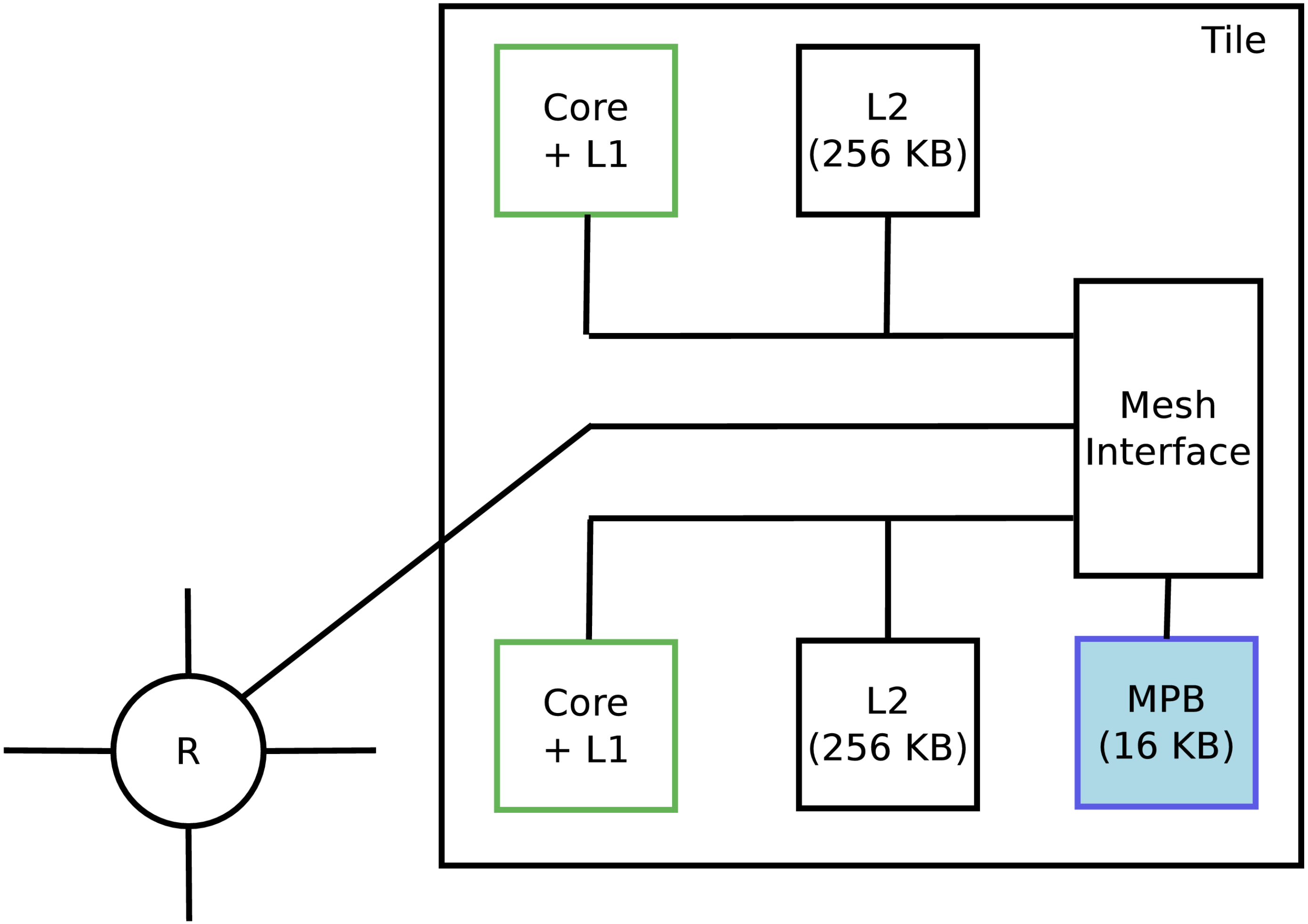
Related Work: Collectives

- 2 interfaces:
  - RCCE_comm (MPI-Like)
    - Built on top of RCCE send/recv interface
  - RCKMPI
    - Adding a SCC channel to MPICH2
    - Send/recv based on RMA to the MPBs
    - Collectives built on top of the send/recv interface

Outline

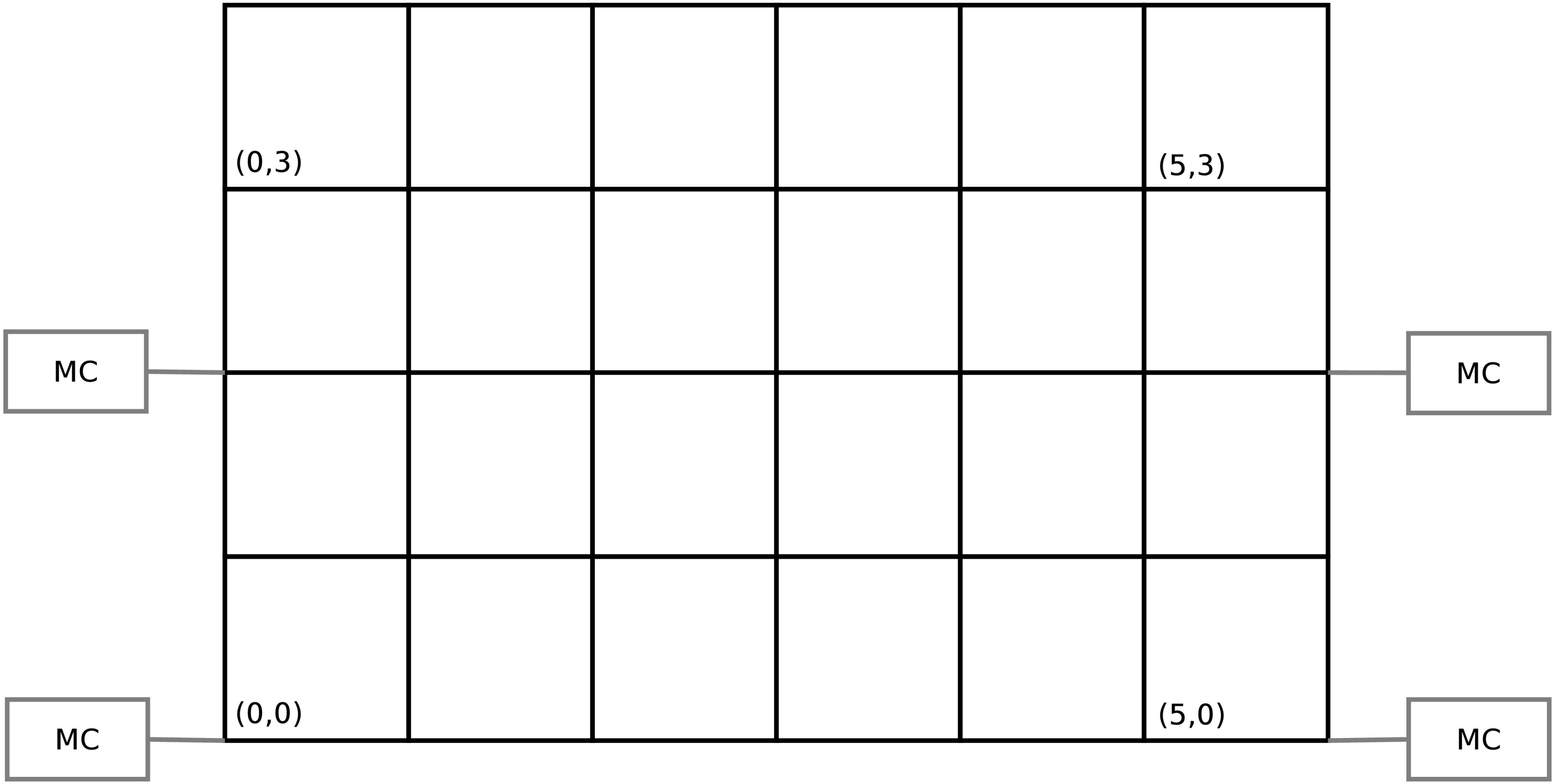
- Detailed description of the SCC
  - Opportunities to get more parallelism
- Description of OC-Bcast
  - RMA-based broascast algorithm
  - Pipelined *k*-ary tree
- Evaluation of the proposed solution
  - Analytical and experimental
  - 27% lower latency and 3 times higher peak throughput
- On-going work

Description of the Intel SCC: 2 Cores per Tile

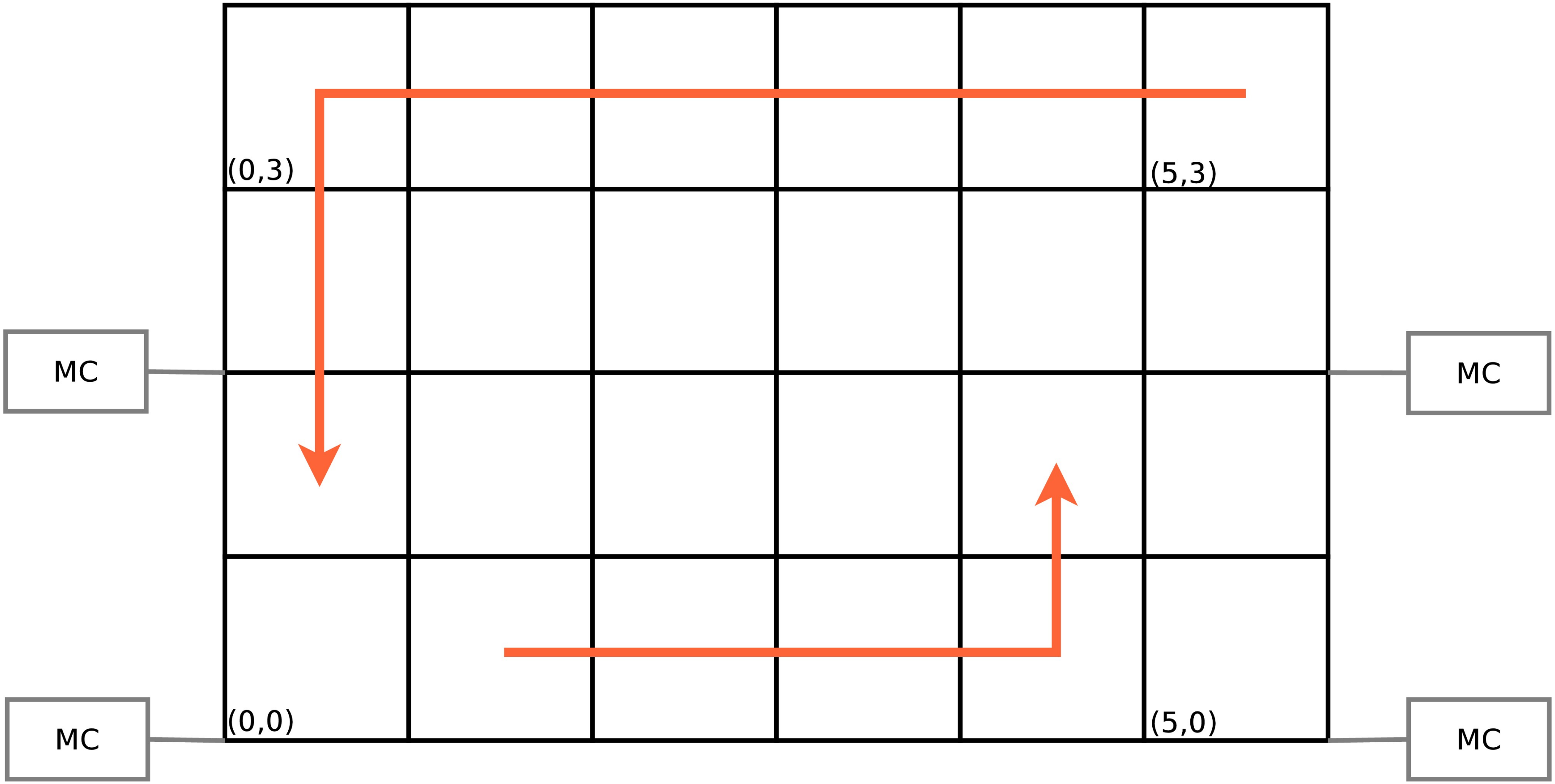


- MPB accessible by all cores

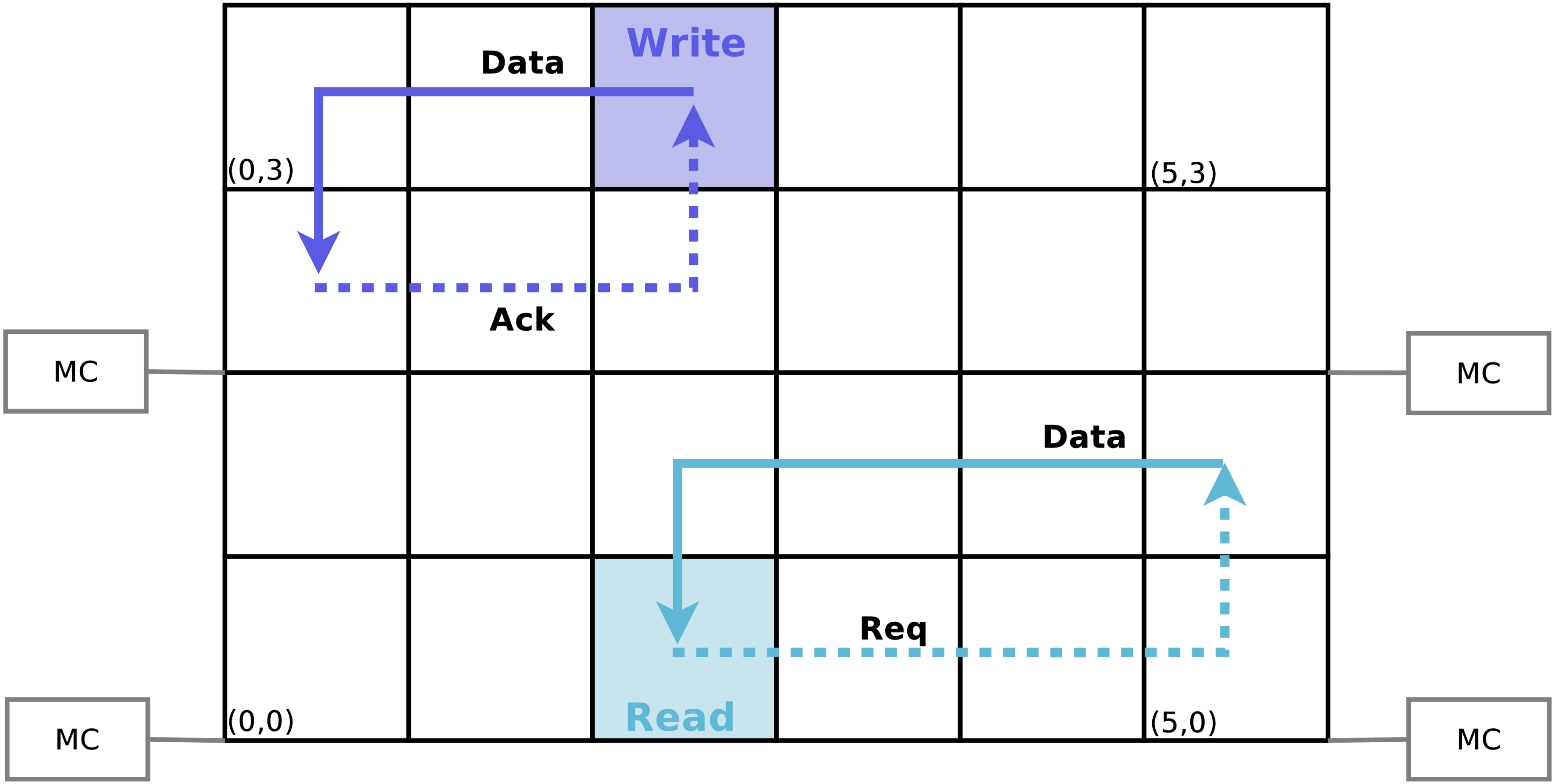
A 2D Mesh



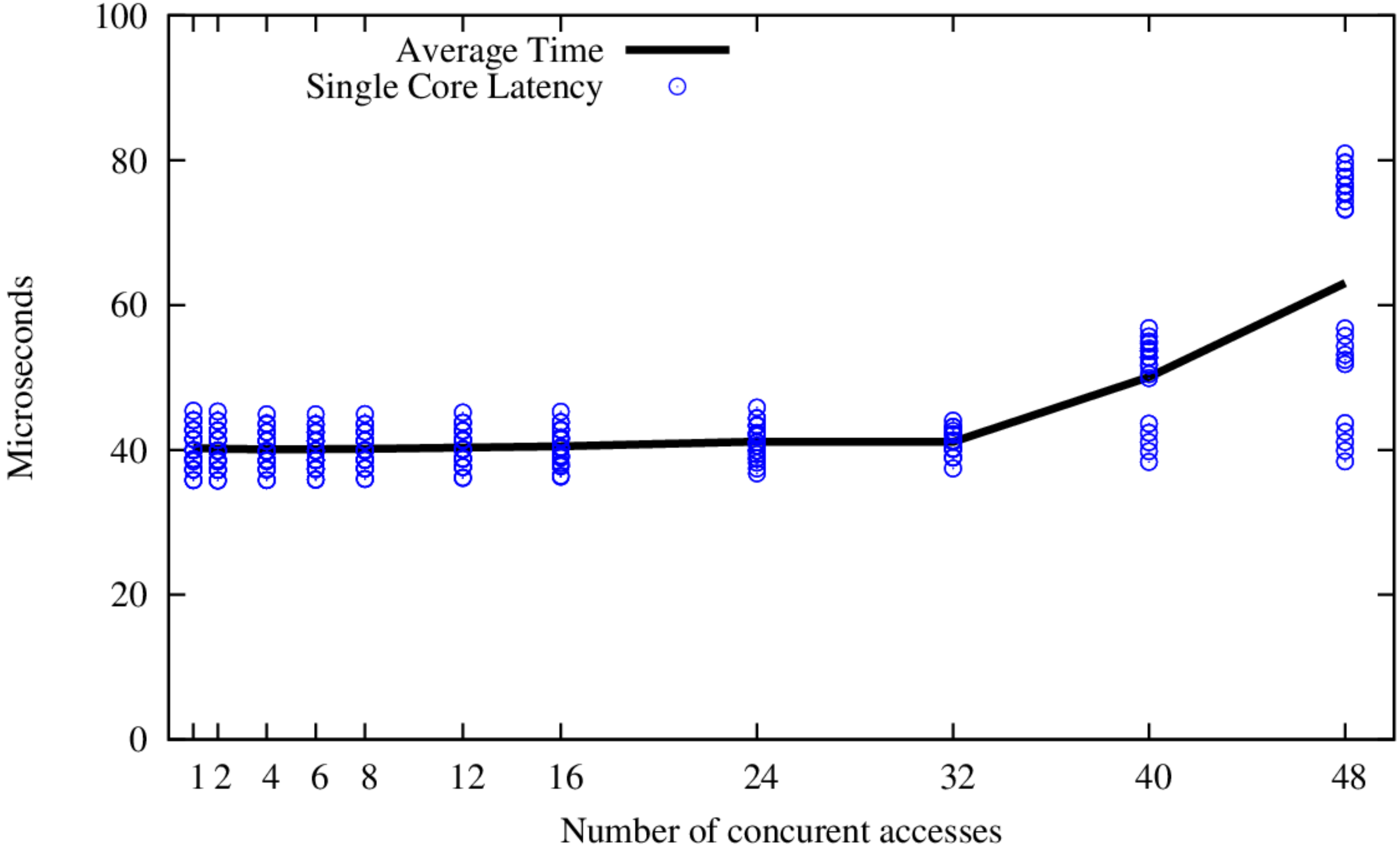
A 2D mesh: X-Y routing



Remote Read/Write to MPBs

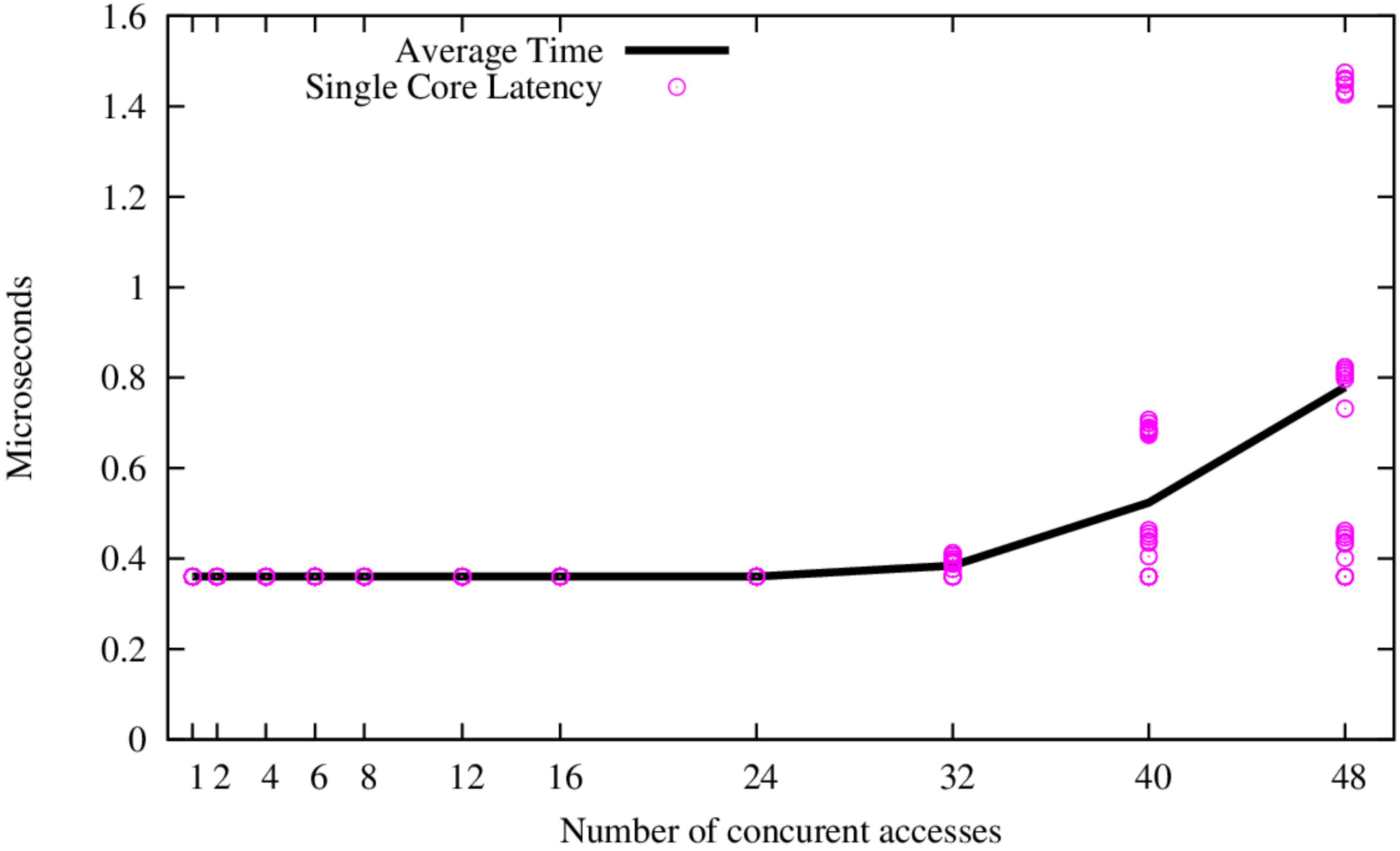


Contention On Concurrent Get to one MPB (128 CL)



- Observable contention from 40 parallel Gets
  - All cores are not equally impacted
  - The slowest core is 2 times slower

Contention On Concurrent Put to one MPB (1 CL)



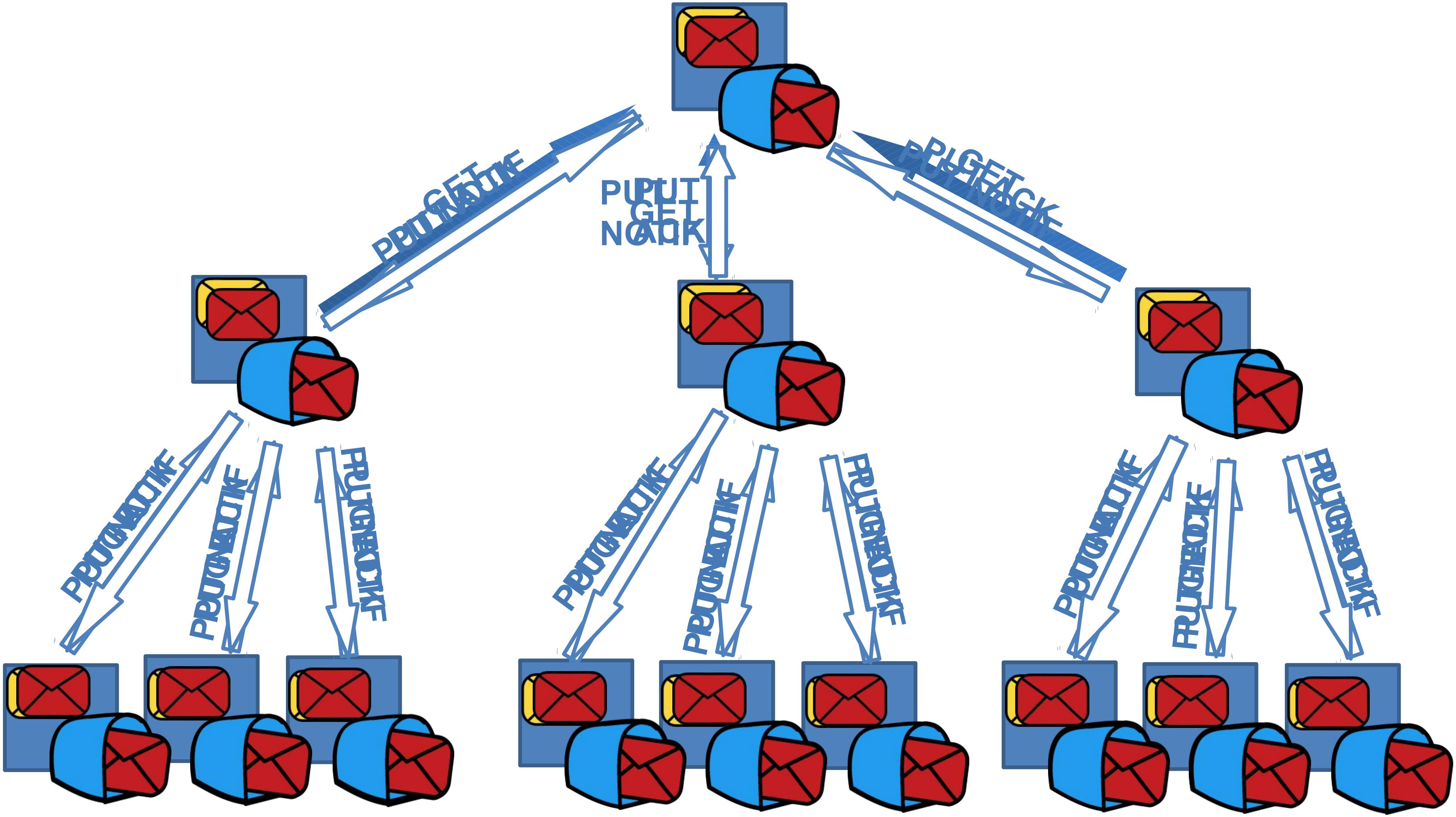
- Observable contention from 32 parallel Puts
  - All cores are not equally impacted
  - The slowest core is more than 4 times slower

Our Broadcast Algorithm

- Goal: Taking advantage of RMA to the MPBs
  - Fast data movements (but limited size)
  - Parallel accesses

- OC-Bcast
  - K-ary tree
  - Pipelining
  - Double-Buffering
  - Based on RCCE Put/Get interface

OC-Bcast (3-ary tree)



Additional Optimizations

- Binary notification tree:
    - K children are notified of a new chunk
        - Put is used: sequential
        - A binary tree is build between the k children to increase the parallelism
- Double buffering (proposed in iRCCE):
    - The MPB is divided into two buffer.
    - The sender can put a new chunk in its MPB while the receivers are getting the previous one.

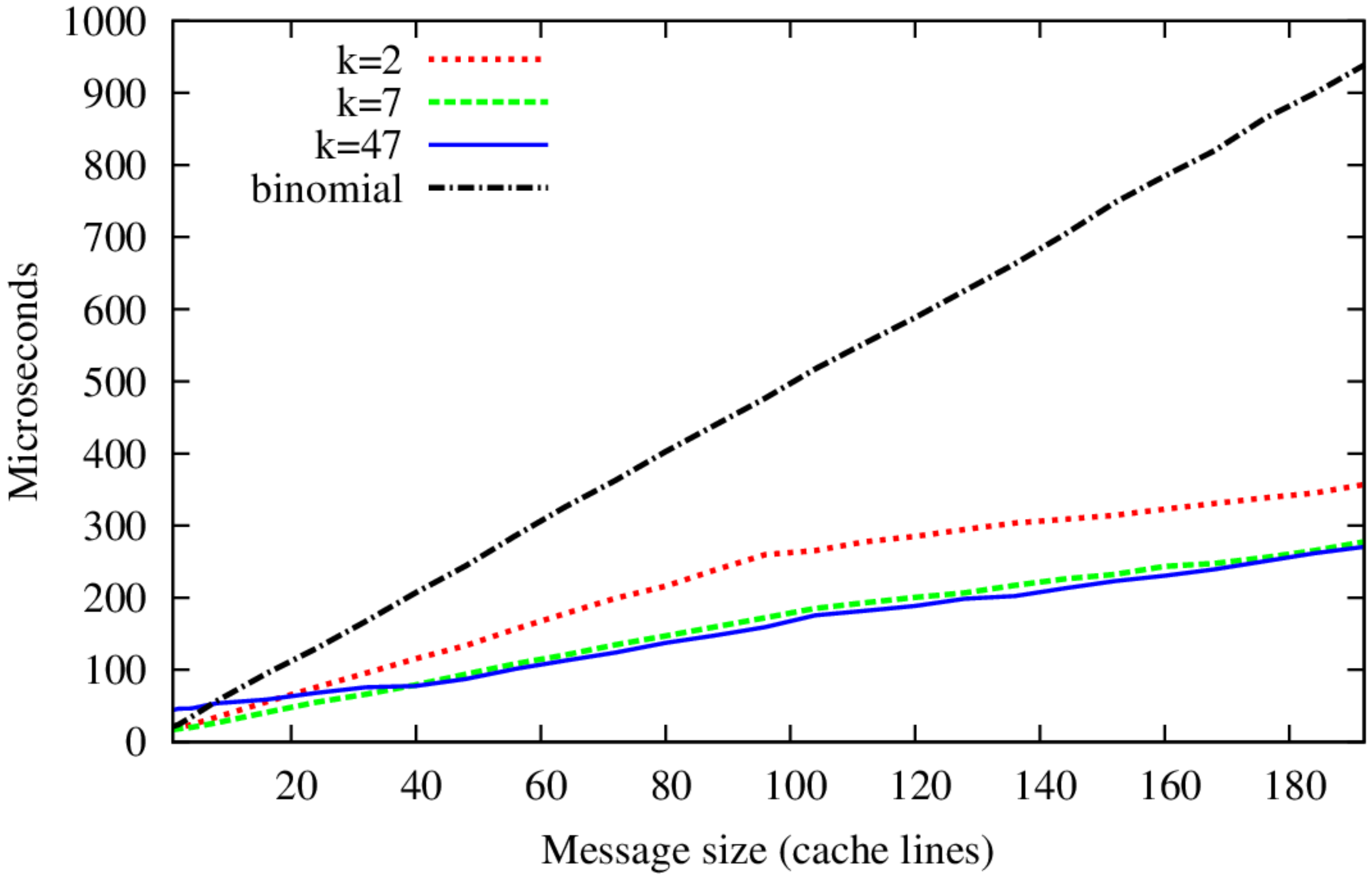
Evaluation

- Comparison with RCCE_comm
  - Provides better performance than RCKMPI
  - Broadcast algorithms based on the send/recv interface

- Small messages
  - Binomial Tree
- Large messages
  - Scatter-Allgather
    - Allgather based on Bruck,1997

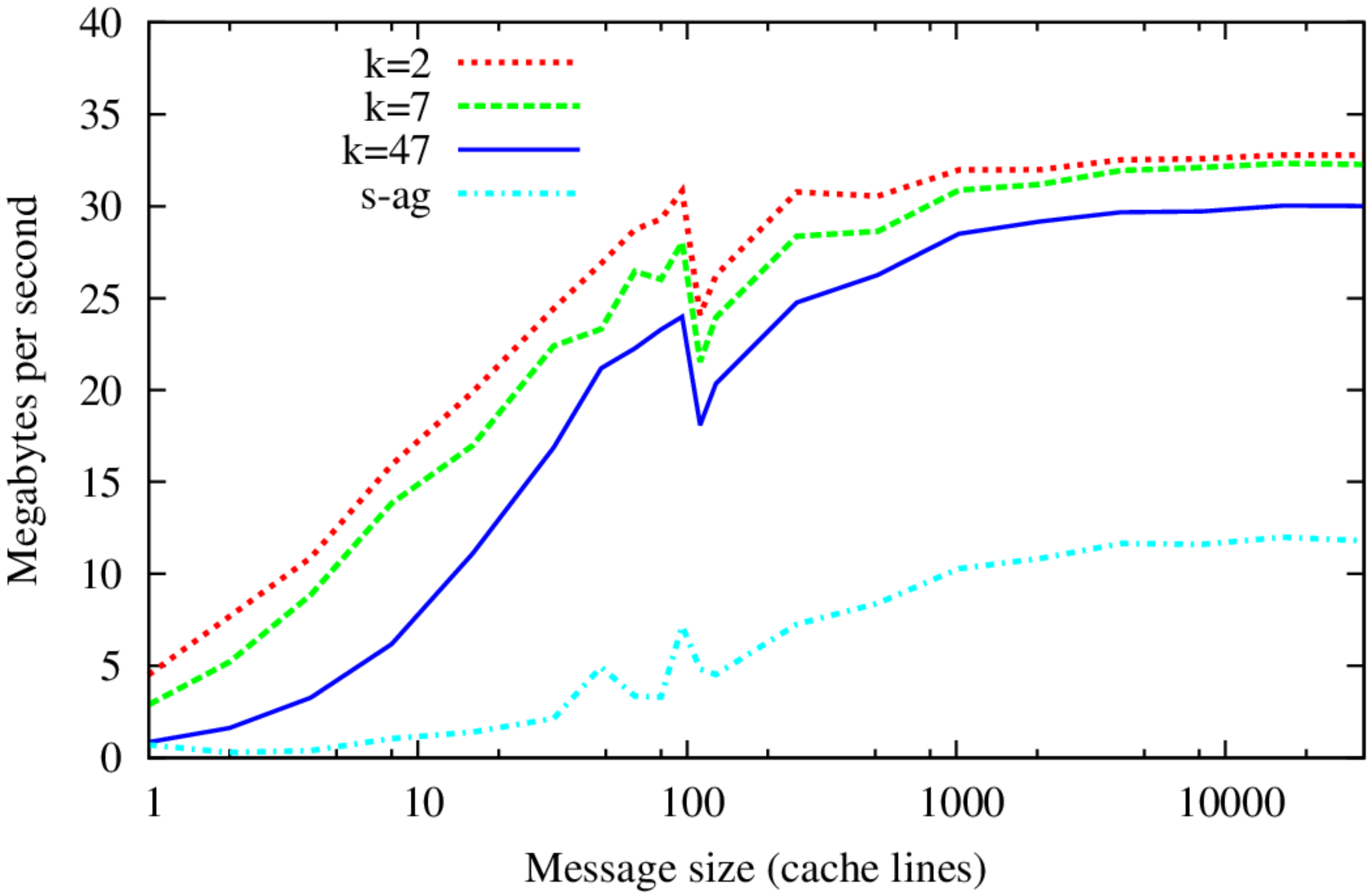
- Trees built based on process ids

Latency (Memory to Memory)



- OC-Bcast outperforms the binomial tree
- Contention is visible for *k*=47

Throughput (Memory to Memory)



- 3 times better performance (61% of max theoretical throughput)
- Lower values of *k* provide better pipelining

Analytical Evaluation

- SCC communication model (LogP based model)

- Performance improvements are a direct consequence of implementing collectives on top of one-sided operations:
  - Latency (one chunk)
    - OC-Bcast: 2 off-chip memory accesses on the critical path
    - Binomial: 3.$\log_2$P off-chip memory accesses
  - Throughput (one chunk)
    - 3 times less write accesses to off-chip memory with OC-Bcast

Summary

- The Intel SCC provides efficient RMA to on-chip MPB for fast communication
- Study of collective operations
  - OC-Bcast
    - Pipelined k-ary tree
      - Takes into account contention issues
    - At least 27% lower latency
    - 3 times better throughput
- Building collectives on top of on-chip one-sided operations can help improving performance of collective operations on many-core chips
  - Confirmed by the analytical study
- *High-Performance RMA-Based Broadcast on the Intel SCC,* SPAA, 2012.

New results: Asynchronous broadcast

- Leveraging parallel interrupts for collective operations
  - System level services
    - Many-core OS

- Broadcast based on parallel interrupts
  - A userspace library to manipulate parallel interrupts
  - An asynchronous version of OC-Bcast
  - Low single broadcast latency
  - Efficient handling of concurrent broadcasts
    - 68% of maximum theoretical bandwidth

Ongoing Work

- Efficient communication library for the Intel SCC
  - Study of other collectives

- Implementation of concurrent data structure on many-cores processors
  - Shared vs replicated data structure
    - Each process has a copy of the data structure in its private memory
    - The data structure is stored in shared memory
  - Implementation of atomic broadcast

Towards Efficient Collective Operations on the Intel SCC

Darko Petrović, Omid Shahmirzadi, Thomas Ropars, André Schiper

Distributed Systems Laboratory

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE