Simulating Parallel Programs on Application and System Level

Julian M. Kunkel

julian.martin.kunkel@informatik.uni-hamburg.de

Scientific Computing
Department of Informatics
University of Hamburg

2012-06-18

informatik
die zukunft

informatik die zukunft

Introduction

Motivation

- What performance do we expect for running
  - an MPI-IO application
  - on an arbitrary (real) cluster?
- Where are the bottlenecks during execution?
- Is the observed performance of an MPI call suboptimal?
  - Is another algorithm better suited?

informatik die zukunft

Introduction

Goals of the simulator

- Foster understanding of performance factors
- Localization of bottlenecks and their causes
- Evaluation and optimization of the I/O path
- Extrapolation of system performance towards future systems
- Experimenting with MPI algorithms
- Evaluation of potential new MPI commands and MPI semantics
- Teaching of the aforementioned aspects

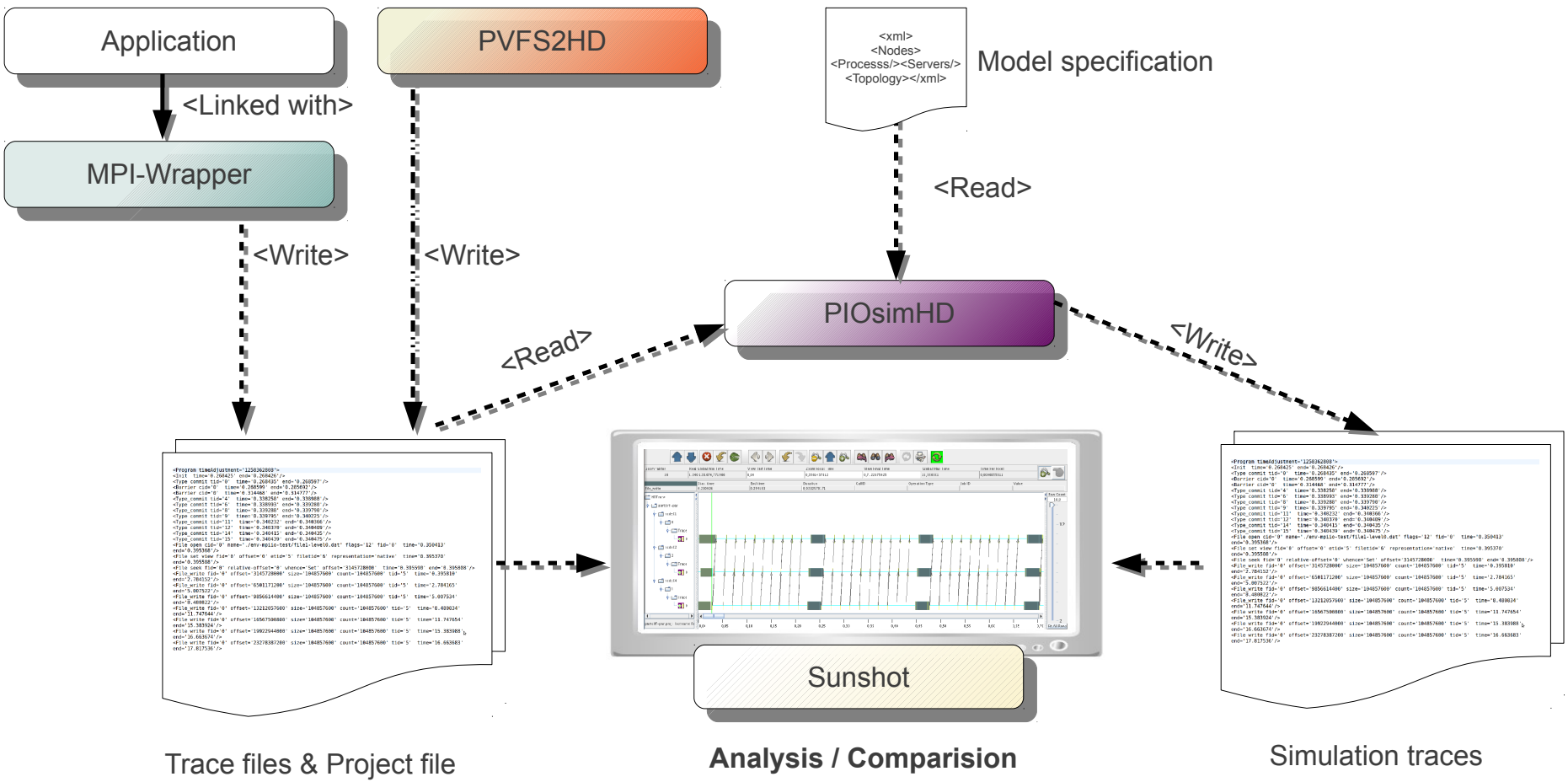
informatik die zukunft

PIOsimHD

Overview

- Sequential discrete-event simulator
- Hardware model captures most important characteristics for:
  - Nodes, network, block device, memory access (for IPC)
- Software model:
  - MPI: P2P & collective algorithms
  - Abstract model of parallel file system
- Modular – implementation / characteristics selectable
  - Easy selection / extension of provided models
- Support for trace/Replay of existing applications
- Visualization of simulation results with trace viewer
  - Allows comparison with real runs

Simulation Workflow for Replaying Applications

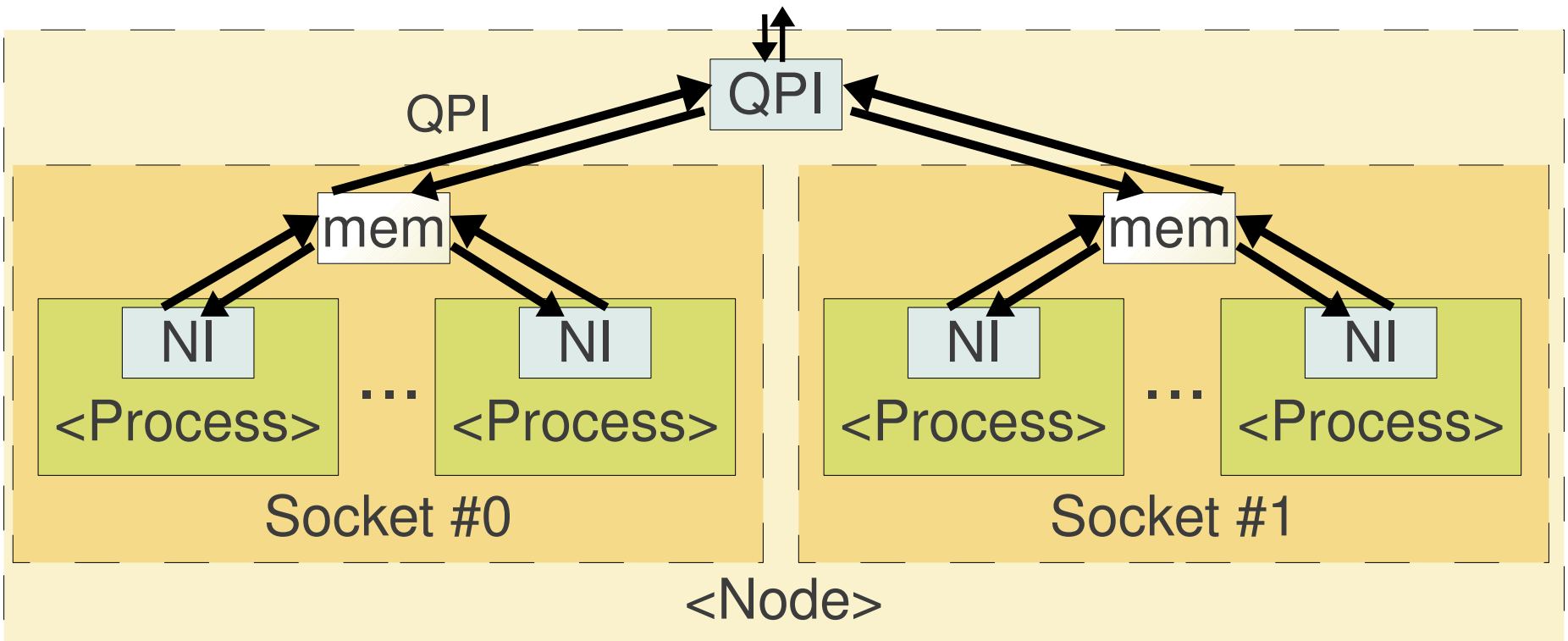


Cluster model

Components

- Node
  - Hosts processes
- Application process
  - Belongs to a parallel application
- Server process
  - Cache layer
  - Block device
- Network link & node
  - Supports arbitrary network graphs

Model for a dual-socket node



Model characteristics (of the default models)

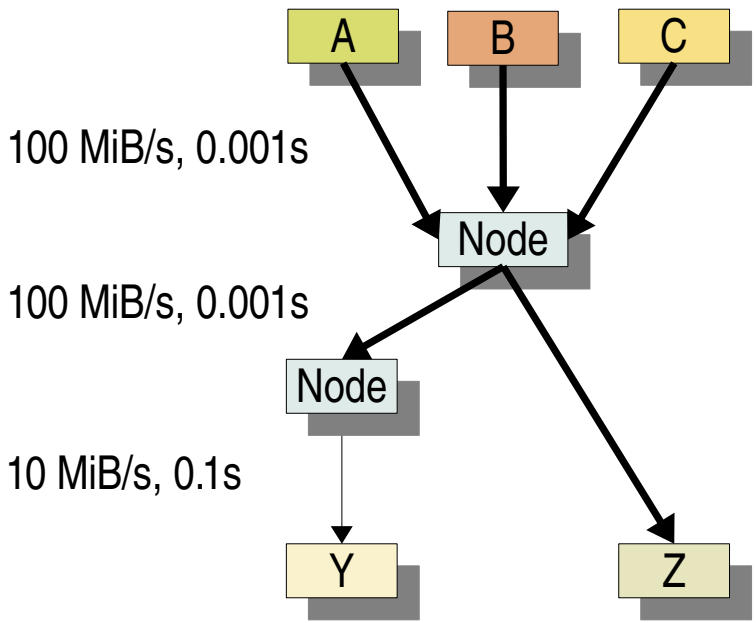
- Node
  - Number of CPUs, CPU speed, memory capacity
  - Resources are shared among all hosted processes
- Network nodes / links
  - Latency, throughput
  - Local throughput: faster intra-socket communication
- Block device (here: hard disk drive)
  - Average and track-to-track seek time, RPM, sequential troughput
  - Distance to the previous position determines seek time
  - Files are assumed to be “stored” consecutively

Network model

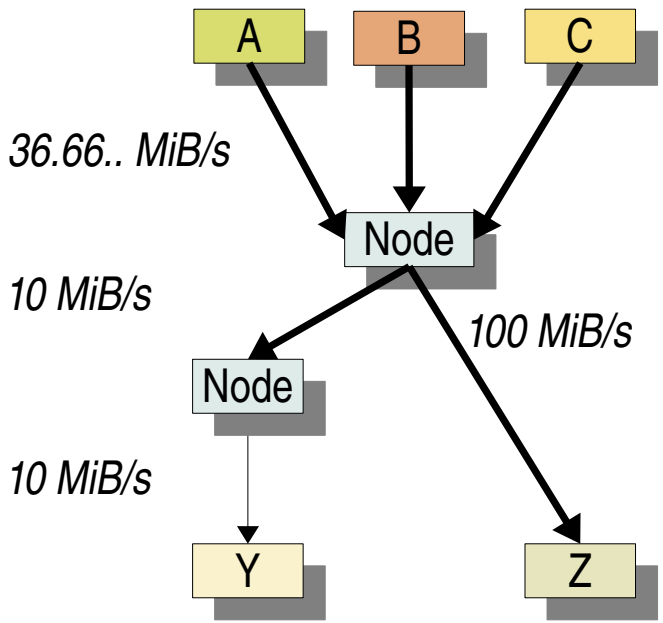
- No simulation of low-level communication protocols
- Store-and-forward switching
  - Messages are fragmented by network interface
  - Contention is explicitly modeled[1]
  - Routing
- Flow control
  - Virtual channel per source and destination
  - A source stops transfer if the channel is full.
    A node pulls packets from blocked sources.
  - Full utilization of available bandwidth-delay product
  - Transfers of slower links are prioritized

[1] Alternative network interface models are provided

Flow: Bottleneck Close to a Sink & Steady Flows

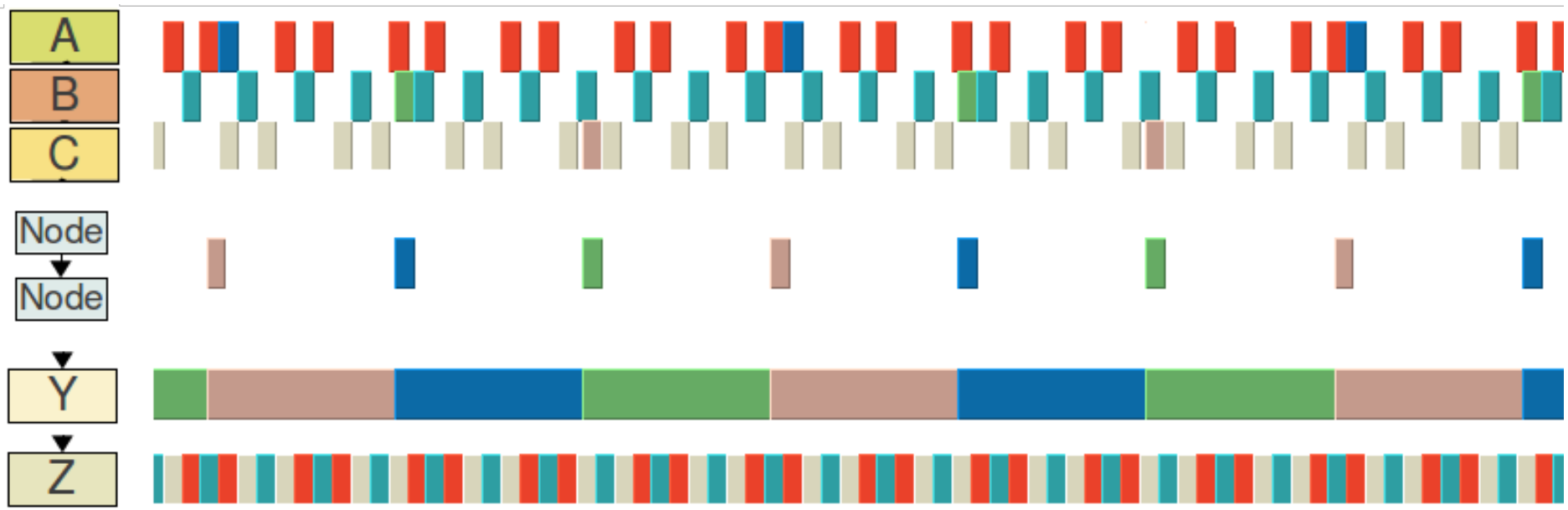


(a) Network topology



(b) Steady state of the network

Example processing scheme for a steady state



- Colors encode a pair of sender/receiver
  - Blue: Node A sends to Node Y
  - Red: Node A sends to Node Z
- Processing of a packet starts data transfer towards this node

Software model

- Application executes a sequence of (asynchronous) commands
  - Communicator-aware implementation inside the simulator
- Application commands are programmed as state machines
  - Wait for completion of initialized communication
  - Spawning of child commands to reuse existing commands
  - Computation occurs between states
- Abstract parallel file system
  - Logic is implemented in MPI-IO commands
    - MPI-IO functions access file system
    - Partition data among servers (default: round-robin)
    - I/O-forwarders are supported
  - Cache-layer controls block device
    - Strategies: FCFS, Write-behind, Disk-directed I/O, ...

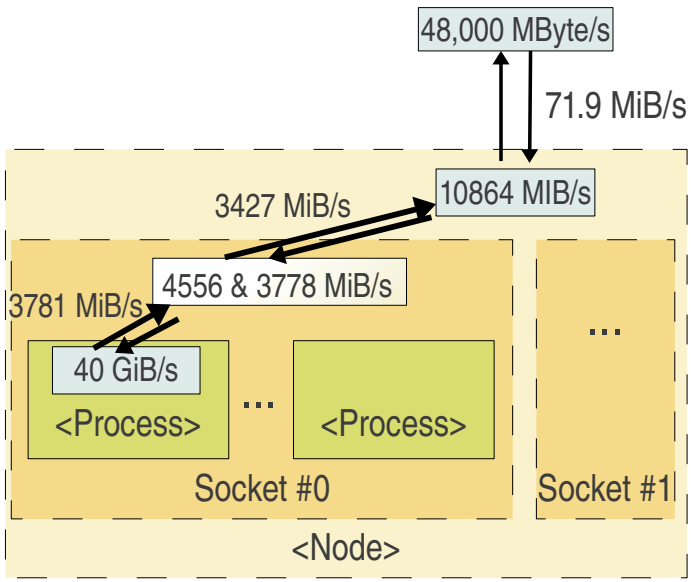
Validation methodology

1. Parameterization
   - Microbenchmarks determine component characteristics
   - Several benchmarks have been executed and compared.
     Components behave non-linear, especially the I/O-subsystem.
2. Qualification: validation of the implemented component models
   - Compare measurements with simple mathematical models
3. Validation of the simulator by using:
   (a) Basic communication and I/O operations
   (b) Simple network and I/O benchmarks
      - For collective operations: replaying internal MPICH2 operations
      - For I/O: different cache sizes, operation on tmpfs
   (c) Application traces (so far: Jacobi PDE solver)
      - Uses collective calls that are implemented in PIOsimHD

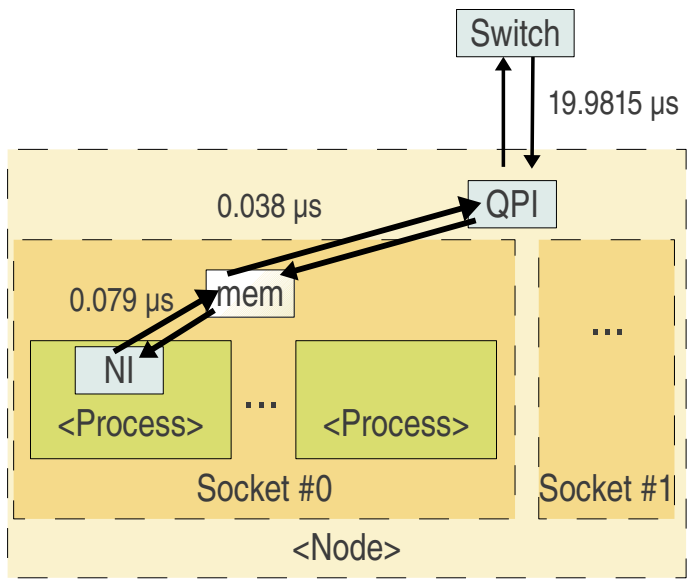
Reference system

The working groups 10 node / 120 core Westmere GiGE cluster

Parameterization: communication characteristics



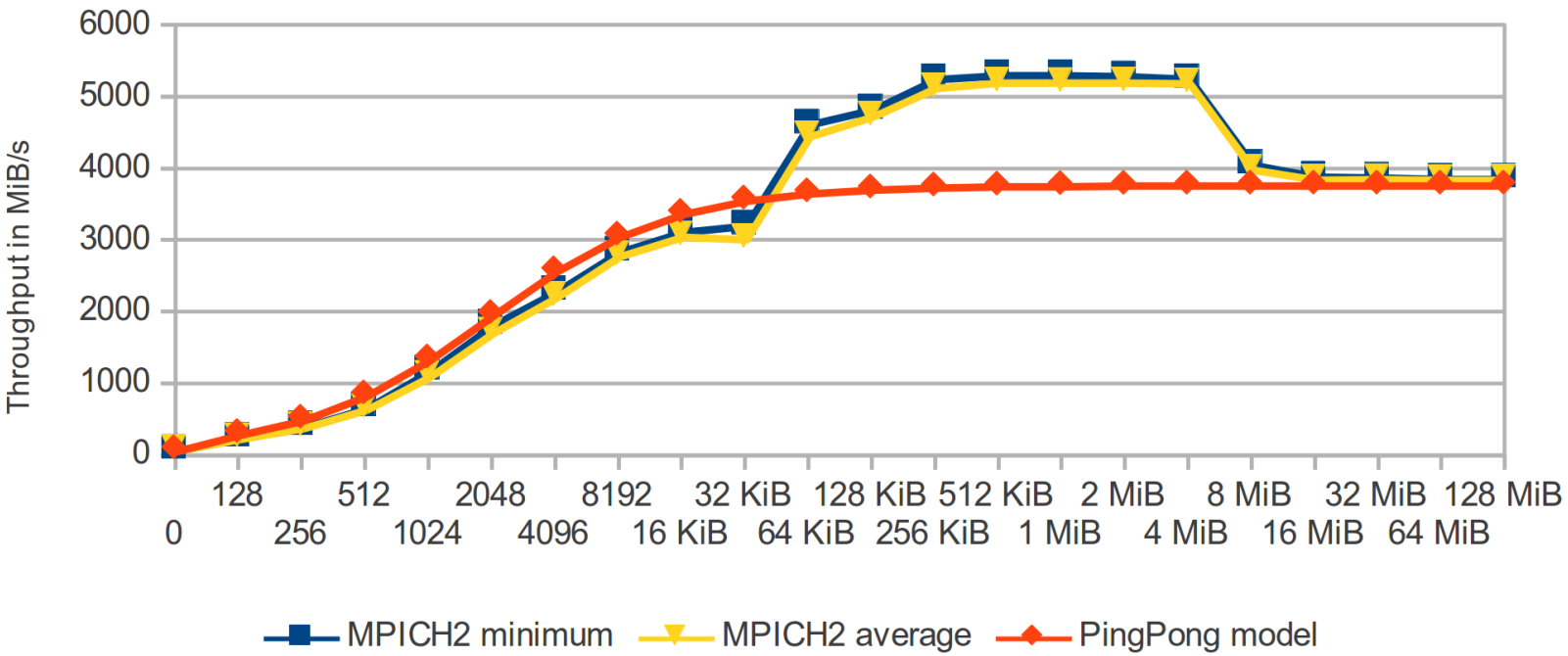
(c) Component throughput



(d) Latency of edges

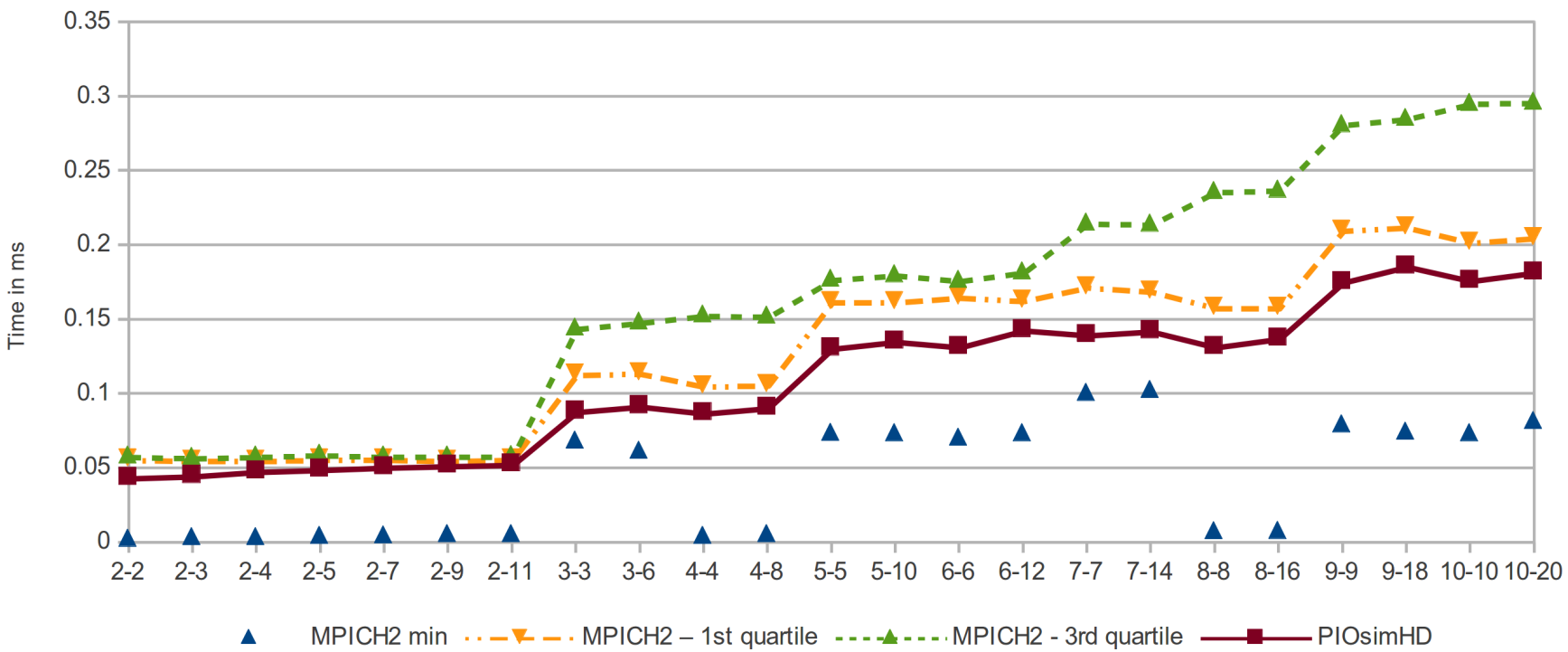
- Average values are determined with MPICH2
- Uni & Bi-directional communication
- QPI value is from the datasheet

Qualification: Model communication throughput



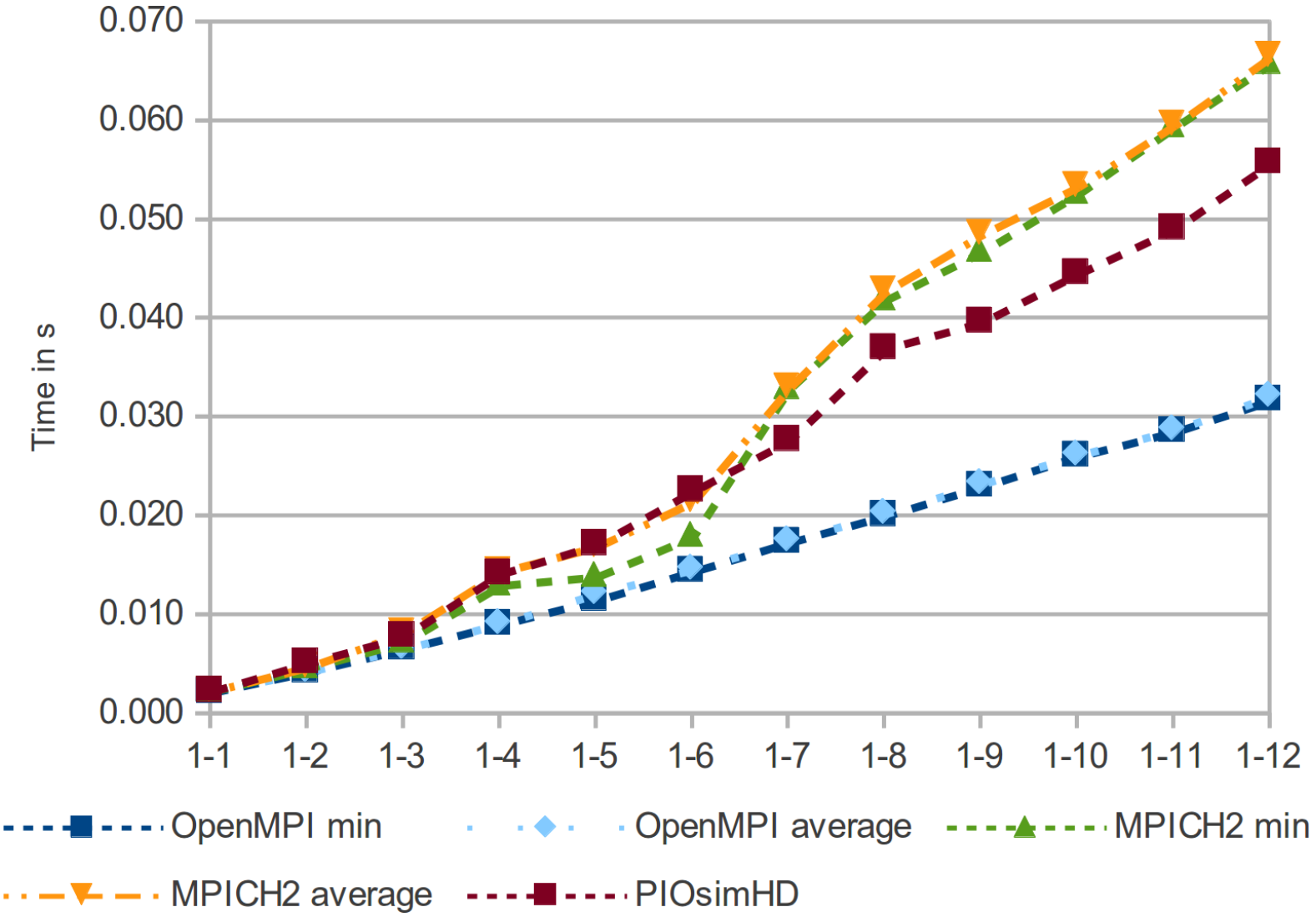
- Inter-socket throughput for the PingPong communication kernel
- Underestimation of performance if it fits into L2/L3 cache

Validation: Assessing MPI_Barrier()

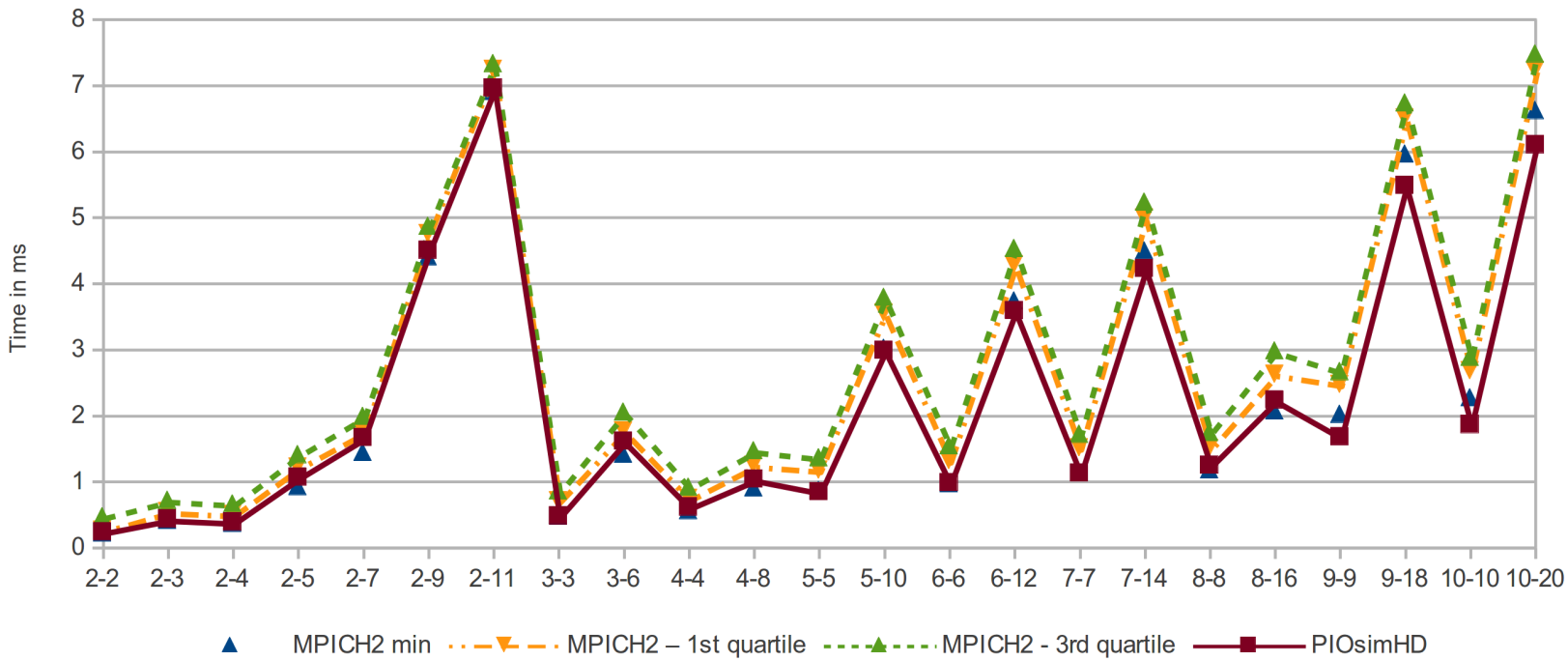


- Time has been measured on the root node only

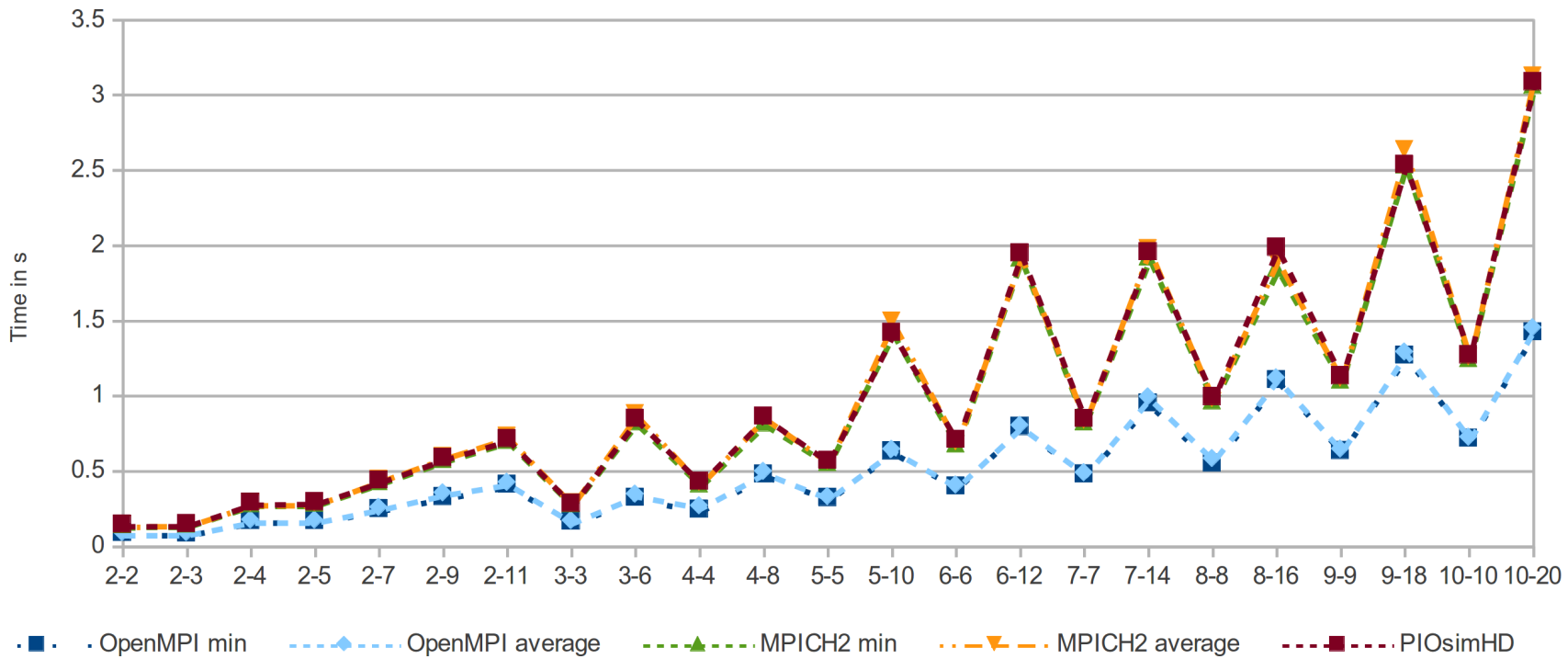
Assessing MPI_Scatter() for 10 MiB of data – intra-node communication



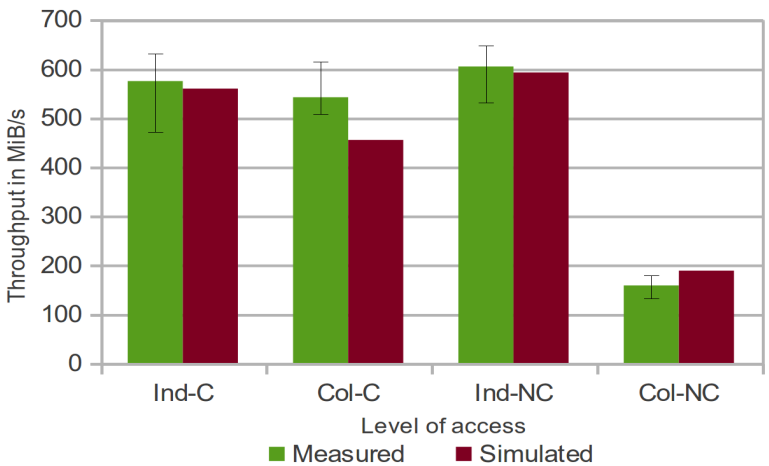
MPI_Allgather() with 10 KiB of data



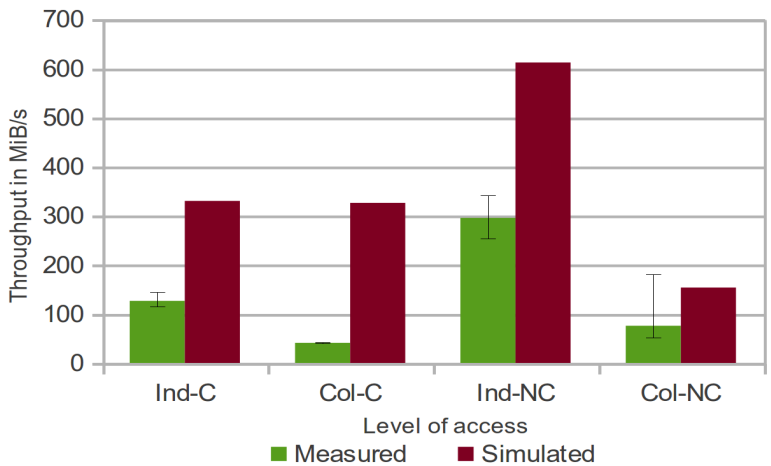
Assessing MPI_Scatter() for 10 MiB of data – inter-node communication



Example for MPI-IO



(e) Reading 100 MiB records



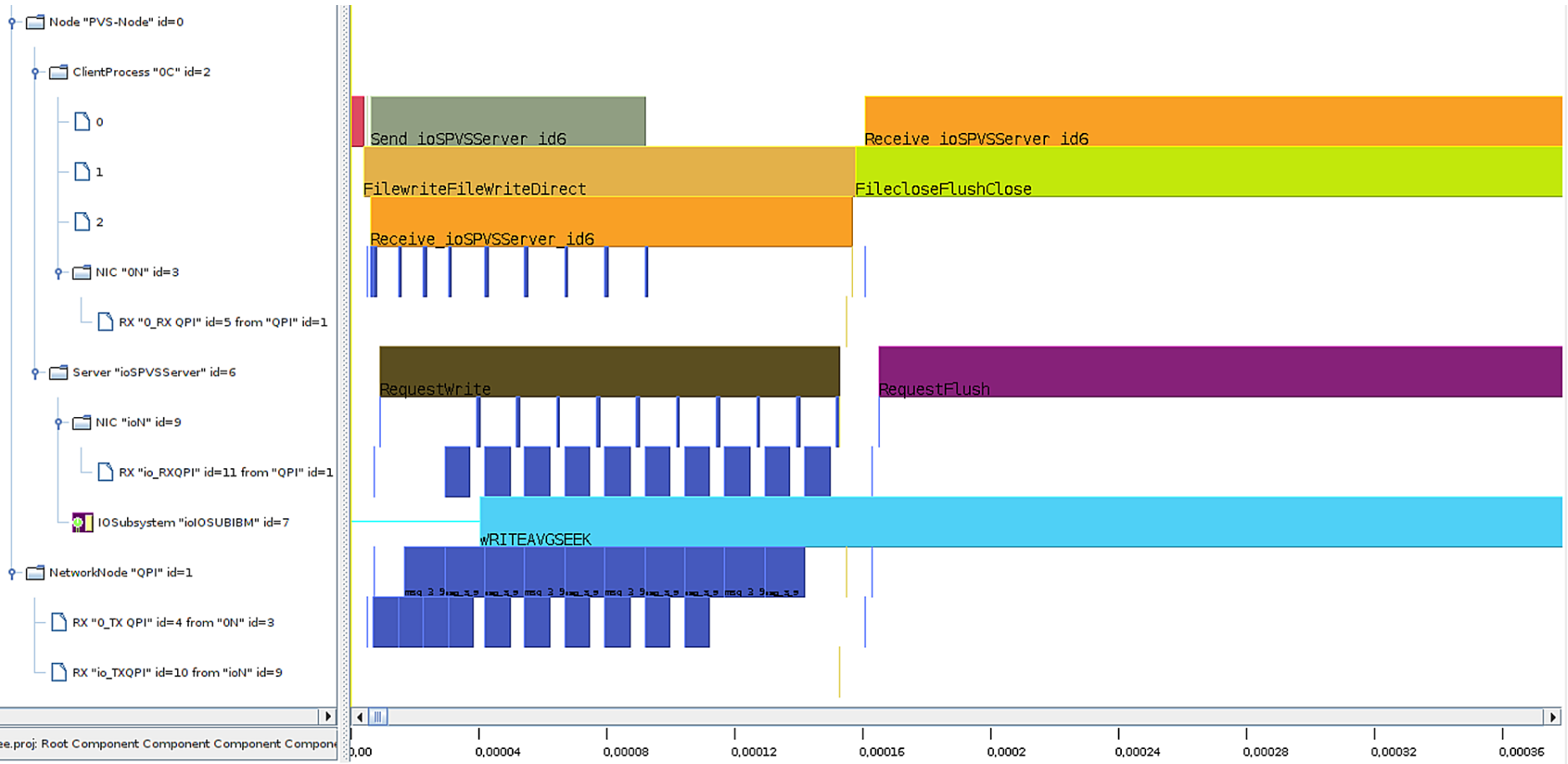
(f) Writing 100 KiB records

- 8 clients transferring 1 GiB of data per process
- Servers are placed on the same nodes
  - 2 GiB of memory for caching writes
- Access of smaller records is better in the simulator
  - Improved I/O scheduling scheme in PIOsimHD

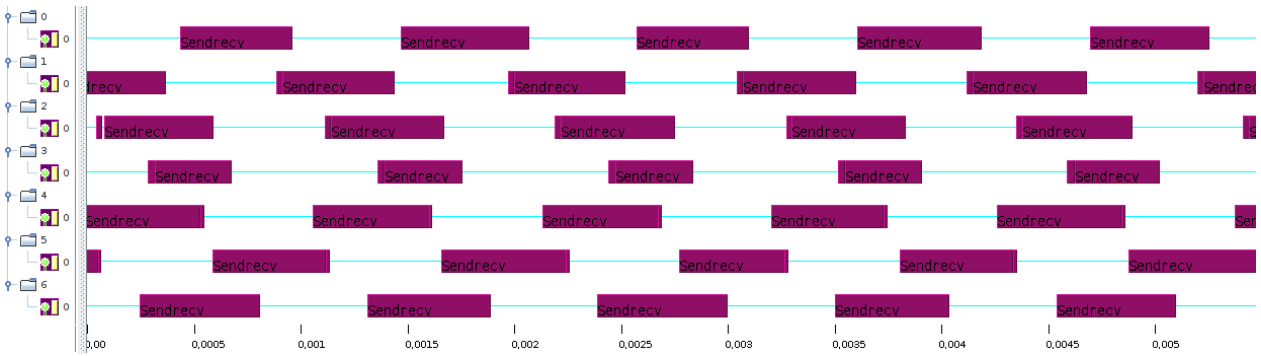
In-depth analysis of the simulation results

- Statistics for hardware devices are output
- Recording of client & internal activity
  - Detailed analysis of internal processing
  - Timeline per component (client, server, ...)

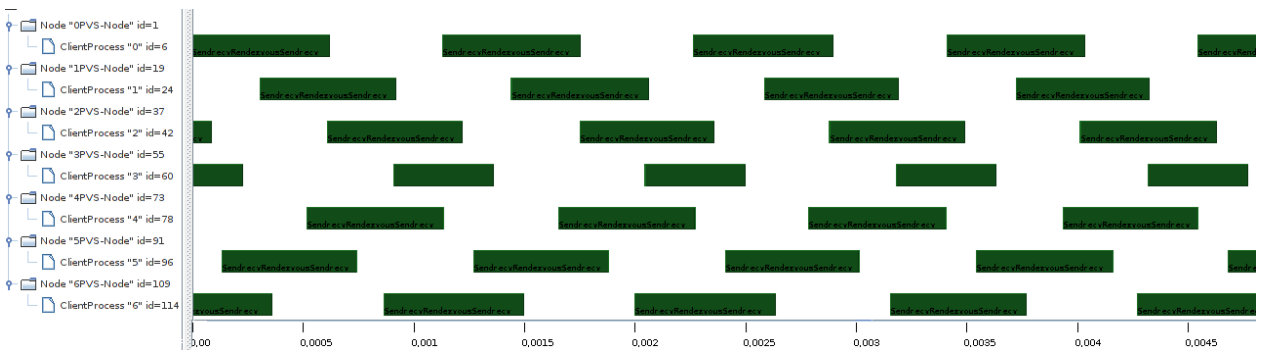
Screenshot of one simulated I/O operation



Trace excerpts for a PDE experiment



(g) Observed behavior

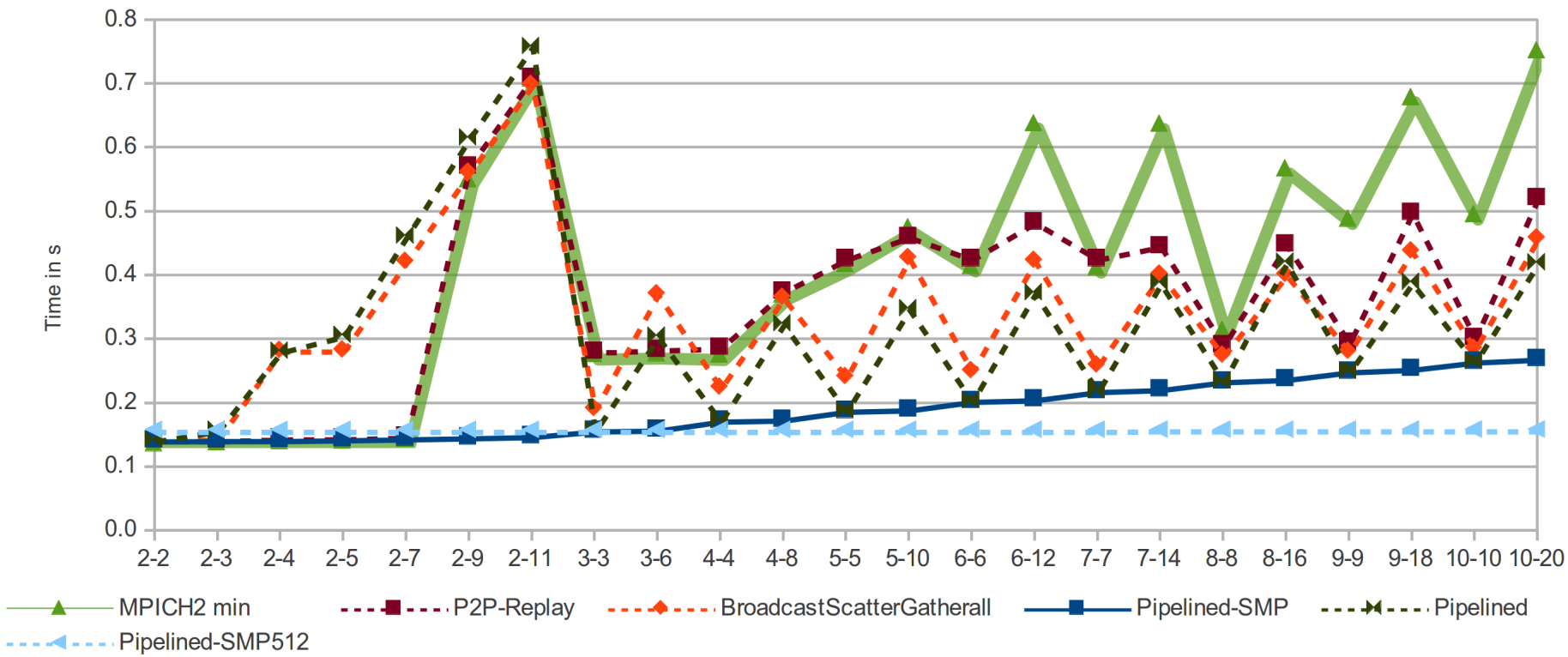


(h) Simulated behavior

Evaluating MPI algorithms

- Simulation of alternative MPI_Bcast() implementations
- Validate performance of existing implementation
  - Just replay recorded P2P access patterns
  - Localize inefficiencies in existing algorithms
- Evaluated implementations:
  - Measured runtime
  - Pipelined (SMP-aware) communication
  - Broadcast-Scatter-Gatherall

Performance comparison of several implementations for MPI_Bcast() – 10 MiB of data



Summary & Conclusion

- PIOsimHD is a tool for simulating MPI-IO activity
  - Suitable for simulating medium sized clusters
  - Simulation of parallel I/O
- Validation of the cluster (and file system) model has been done
  - Bottlenecks can be identified with trace-replay
  - Microscopic real-world effects might become visible
- Alternative MPI-IO implementations can be evaluated

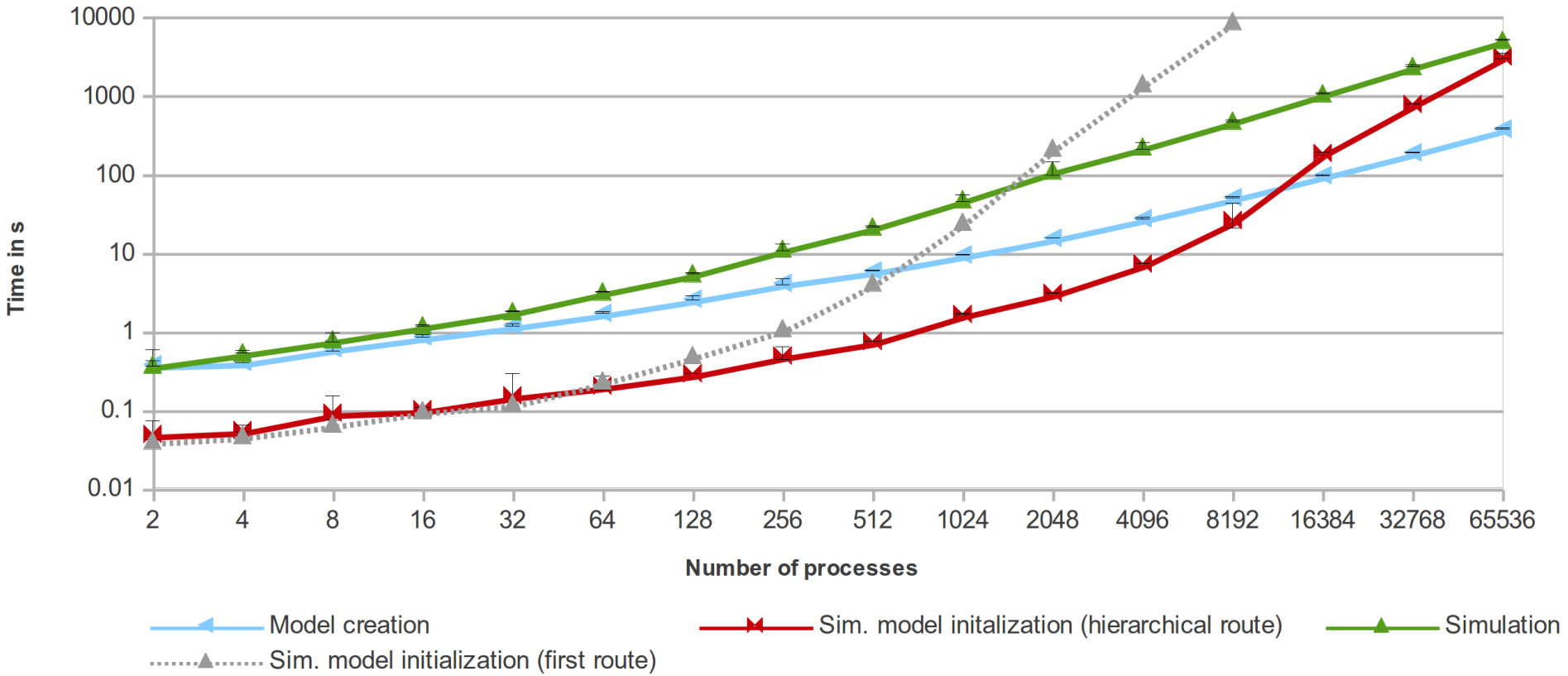
Future work

- Conduct more experiments with HDTrace/PIOsimHD e.g:
  - Evaluate relaxed MPI collective calls (e.g. `MPI_Reduce()`)
  - Evaluate and implement improved collective calls for clusters

Simulator performance

- Evaluation of the three steps required for a simulation run:
  - Model creation
  - Model initialization
  - Simulation execution (event processing)
- Raw event processing speed: 1.5 M events/s
  - The number of events created depends on:
    - Topology (number of links between source/destination)
    - Message size & network granularity
- `Bcast()` scalability up to 65536 processes has been measured

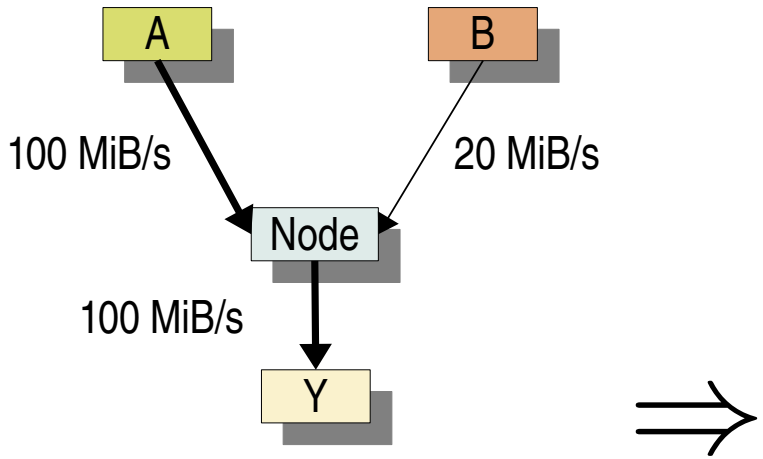
Scalability of a binary `MPI_Bcast()` implementation[2]



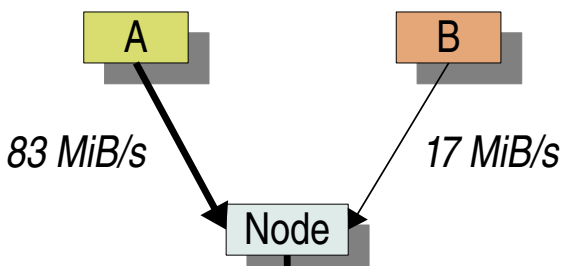
- Almost linear scaling for the model initialization and simulation
- Simpler network models increase performance by orders of magnitute
- Static routing algorithms can be improved

[2] Broadcast 100 MiB of data with 100 KiB packets

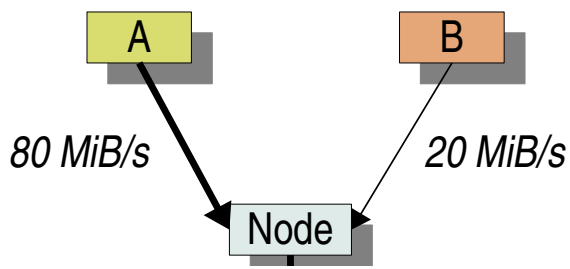
Flow: Bottleneck Close to a Source & Steady Flows



(i) Network topology & maximum throughput



(j) Fair flow control



(k) Prefer slow transfers

Screenshot of the pipelined implementation