SIM GRID

ANR 08 SEGI 022

Single Node On-Line Simulation of MPI Aplications with SMPI

Pierre-Nicolas Clauss [1], Mark Stillwell[2], Stéphane Genaud[3]
Frédéric Suter[4], Henri Casanova[2] and Martin Quinson[1]

[1]Nancy University, LORIA, INRIA, Nancy, France,
[2]Department of Information and Computer Sciences,
University of Hawai'i at Mānoa, Honolulu, U.S.A.,
[3]University of Strasbourg, France
[4]IN2P3 Computing Center, CNRS, IN2P3, Lyon-Villeurbanne, France

May 18, 2011

Motivations: why use simulation?

- Performance Prediction ("what-if?" scenarios)
  - Platform dimensioning
  - Tuning of application parameters
- Teaching (parallel programming, HPC)
  - No need for real hardware
  - Handy environment

Challenges

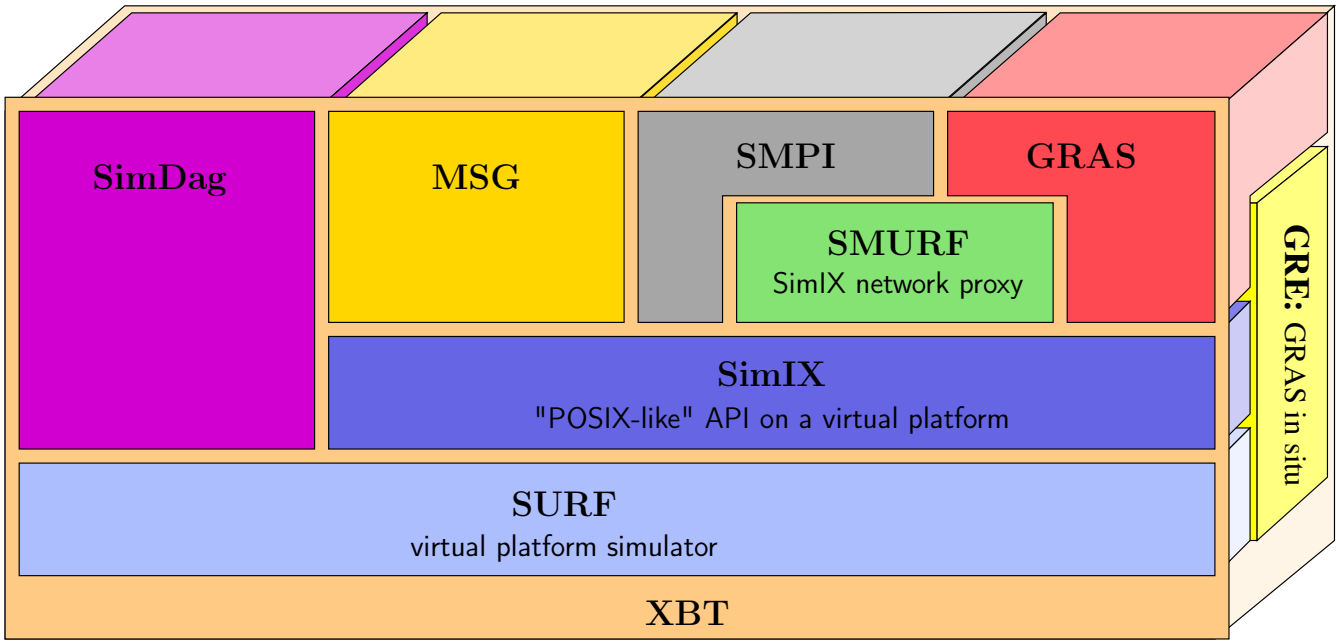
- Accuracy: How well does the simulation reflect reality?
- Scalability: Which problem size can we simulate? On which platforms?
- Speed: How fast is the simulation as compared to the real execution?
- Reproducibility: Are the simulation results stable and reusable?

Approaches to simulation

- *Off-line* simulation
  - Execute and record once the **events**. Replay events post-mortem
  - Requires a real platform comparable to the simulated system
  - Cannot simulate applications whose behavior is parameter-dependent
  - Can only extrapolate communications in homogeneous case
  - LogGOPSim, PSiNS
- *On-line* simulation
  - Real execution of the **code** each time
  - Simulation of communications
  - Only one computer required
  - MPI-NetSim, SMPI

On-line simulation in SMPI

- Partial implementation of MPI on top of SimGrid
  (SimGrid provides a *discrete event simulation* kernel)

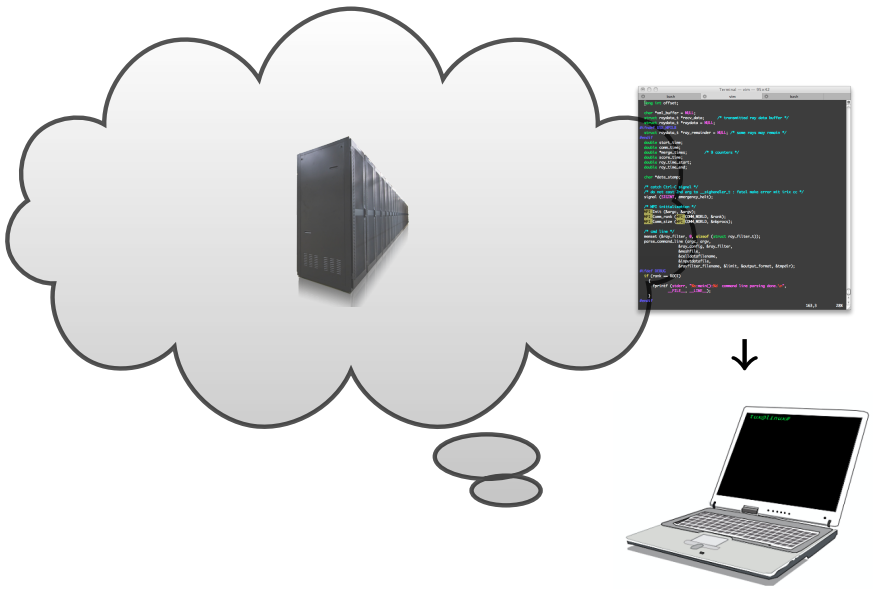


On-line simulation in SMPI

- Partial implementation of MPI on top of SimGrid
  (SimGrid provides a *discrete event simulation* kernel)
- No or few modifications to the source code (C or Fortran)

On-line simulation in SMPI

- Partial implementation of MPI on top of SimGrid (SimGrid provides a *discrete event simulation* kernel)
- No or few modifications to the source code (C or Fortran)



On-line simulation in SMPI

- Computations: real execution on the host computer
  - CPU bursts are benched
  - Scale linearly CPU time according to power ratios

On-line simulation in SMPI

- Computations: real execution on the host computer
  - CPU bursts are benched
  - Scale linearly CPU time according to power ratios
- Communications: simulated
  - Network models are flow-based models (TCP)
  - Validity of these models for MPI applications

On-line simulation in SMPI

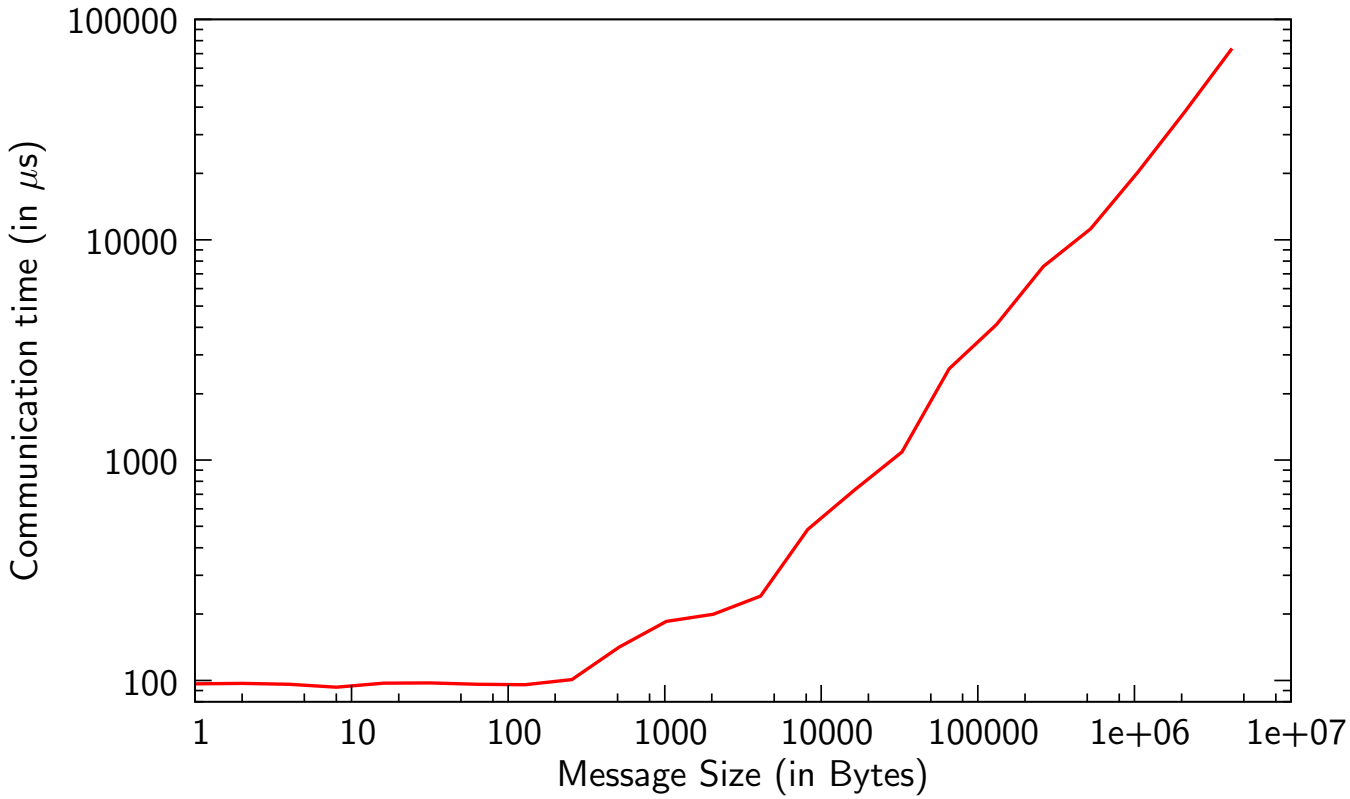
- Computations: real execution on the host computer
  - CPU bursts are benched
  - Scale linearly CPU time according to power ratios
- Communications: simulated
  - Network models are flow-based models (TCP)
  - Validity of these models for MPI applications
- Folding of the parallel program processes onto a single node
  - Serialization of computations
  - Single address space
  - Requires to reduce
    - Memory footprint (scalability)
    - Simulation time (speed)

Existing Network Models in SimGrid

- Arbitrary topology, endpoints connected through multi-hop paths
- Network link caracteristics: latency (L) and bandwidth (B)
- Simulation using flows
  - Simulation is fast ($\neq$ packet-level simulation)
  - Contention evaluation is simple
- Simple Model: $T(S) = L + \frac{S}{B}$
  - Shown to be valid for $S \geq 10$ MB
- Improved model: $T(S) = \alpha \cdot L + \frac{S}{min\left(\beta \cdot B, \frac{\gamma}{2 \cdot L}\right)}$
  - α accounts for TCP slow-start
  - β accounts for the overhead induced by TCP/IP headers (*e.g* 92%)
  - γ enables the modeling of the TCP window induced behavior
  - Model valid for $S \geq 100$ KiB, does not address a lot of message sizes found in MPI applications
- Need for a new, accurate network model when $S < 100$ KiB

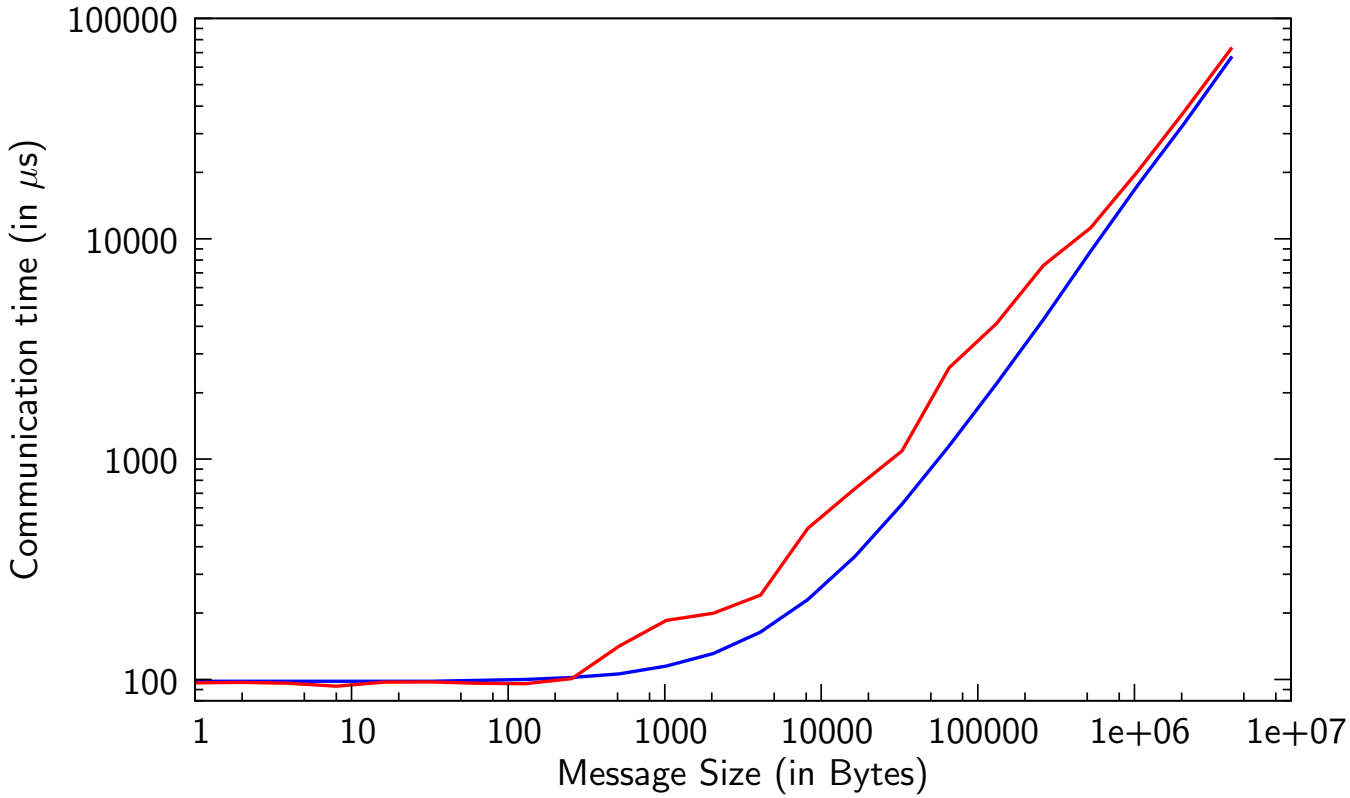
Outline

Point-to-point Communication



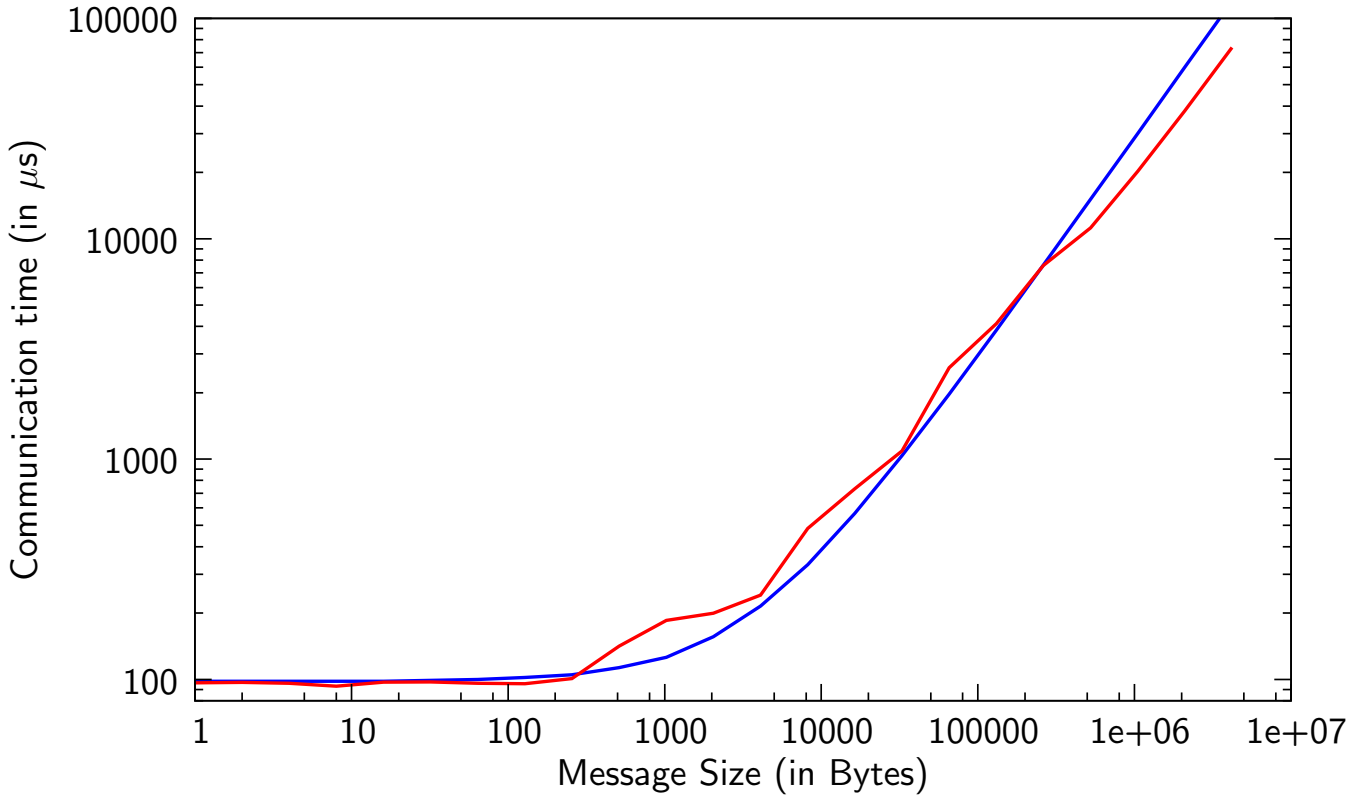
Experimental measurement using SKaMPI

Point-to-point Communication



Experimental measurement using SKaMPI
Default linear model, error: 32.1%
Ok with asymptotic message sizes,
but wrong for 1KiB-1MiB messages

Point-to-point Communication



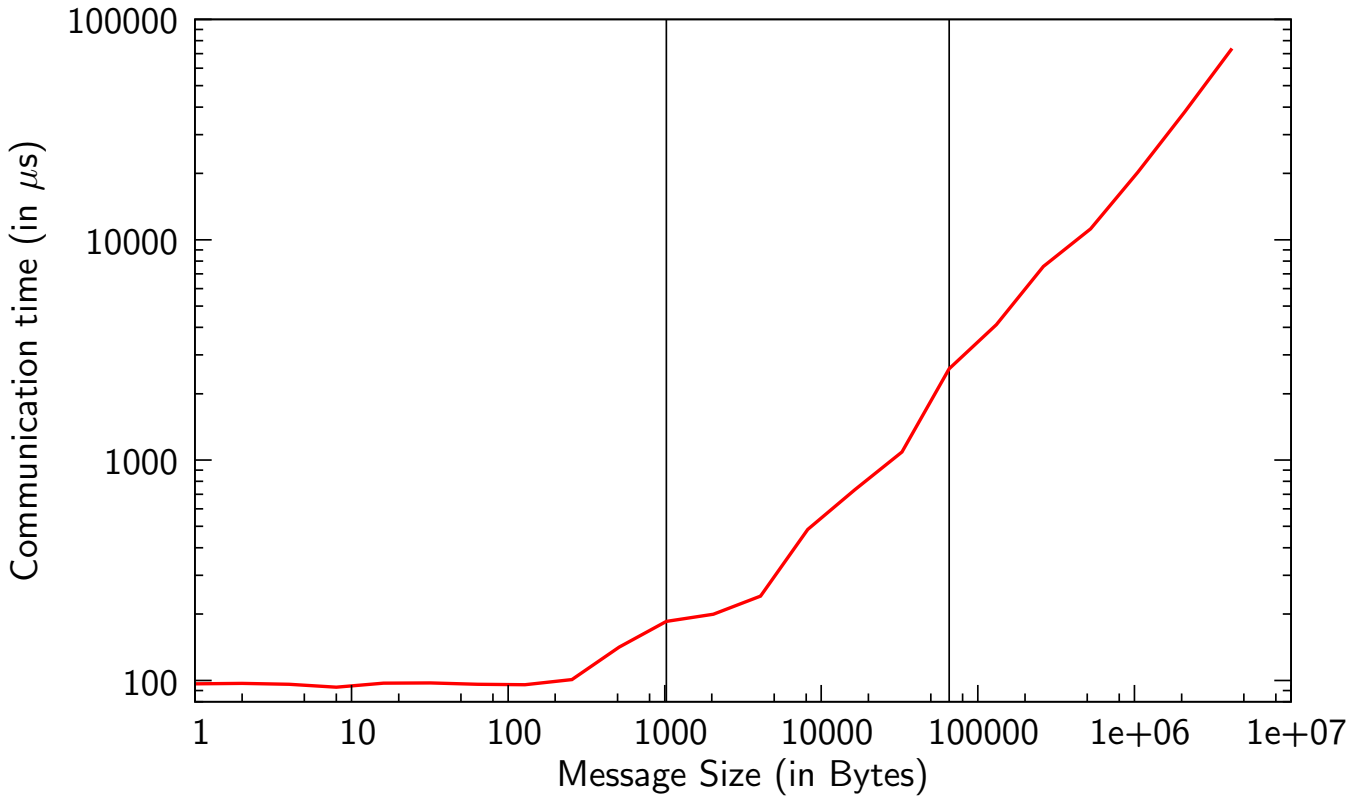
Experimental measurement using SKaMPI

Best-fitted linear model (α, β, γ), error: 18.5%

Better for a lot of sizes,

but cannot fit all real values

Point-to-point Communication

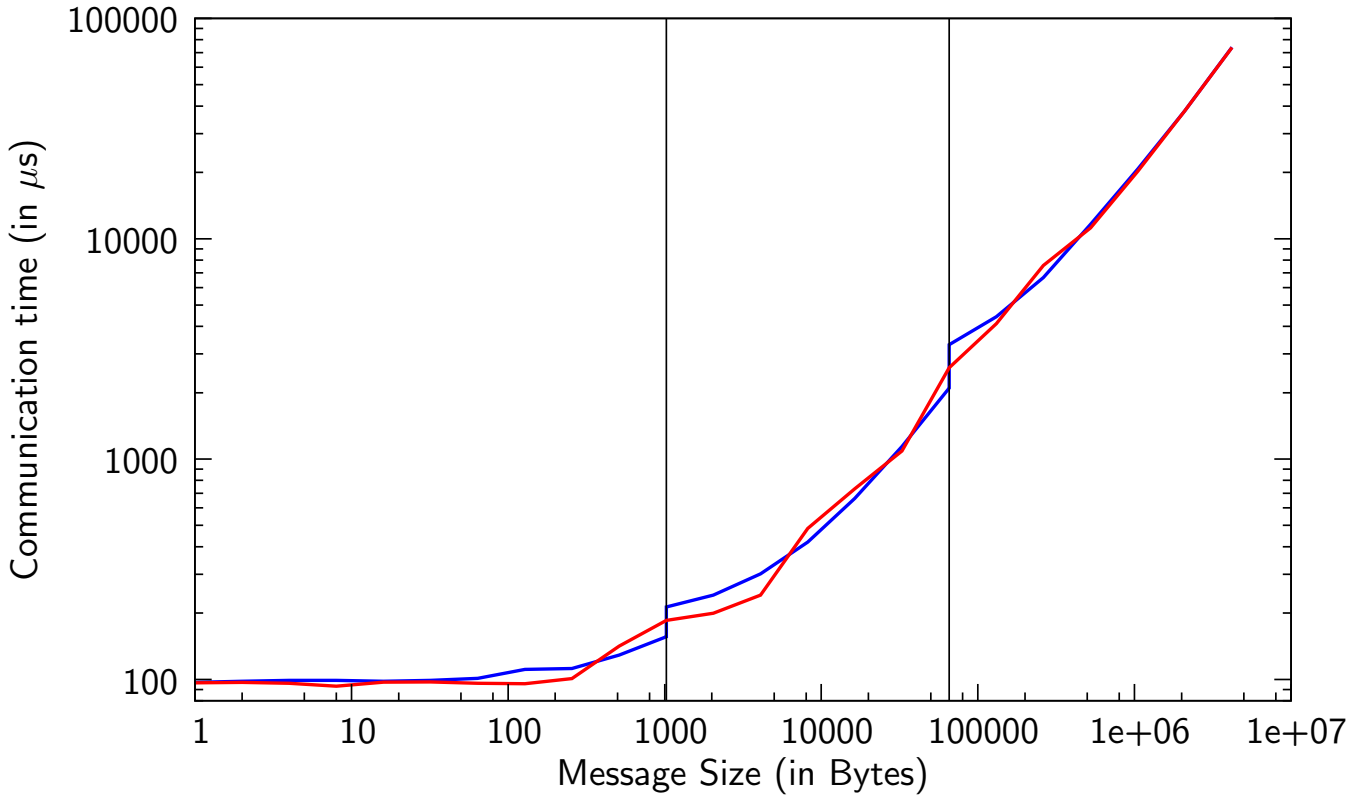


Experimental measurement using SKaMPI
Breakdown depending on message size

– packet size $<$ MTU,

– eager/rendezvous switch limit

Point-to-point Communication



Experimental measurement using SKaMPI
New piece-wise linear model, error: 8.63%
Correctly adjust linear segments

Calibration of the piece-wise linear model

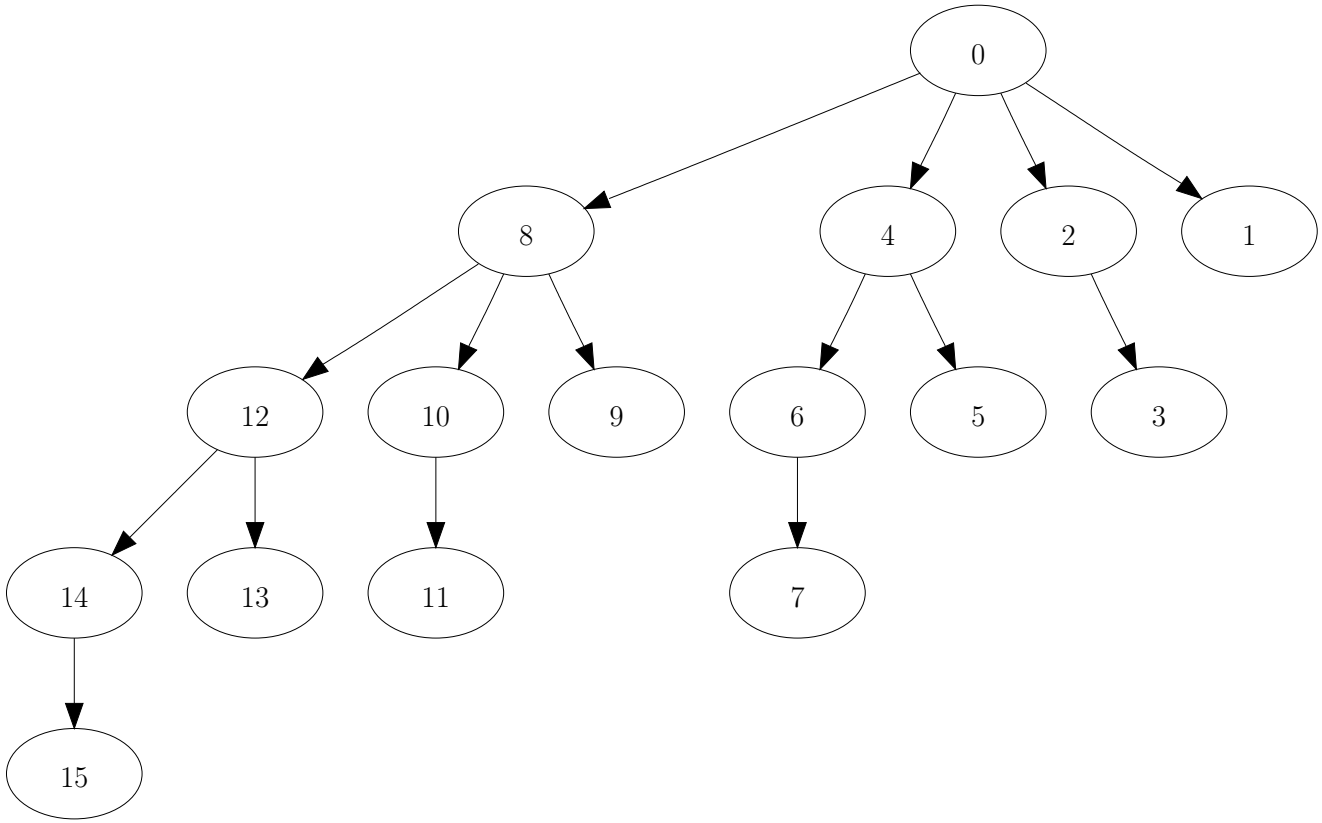
- Instantiate 9 parameters instead of 3
  - 2 segment frontiers
  - 2 factors α and β per segment
  - 1 global factor γ
- A calibration script comes with SMPI. Computes parameters given:
  - 1 SKaMPI-formatted datafile of a ping-pong performance measurement
  - The number of physical links crossed by packets in the ping-pong
  - L and B values for the links
  - segment bounds (computed by another script)

Collectives

Assess contention

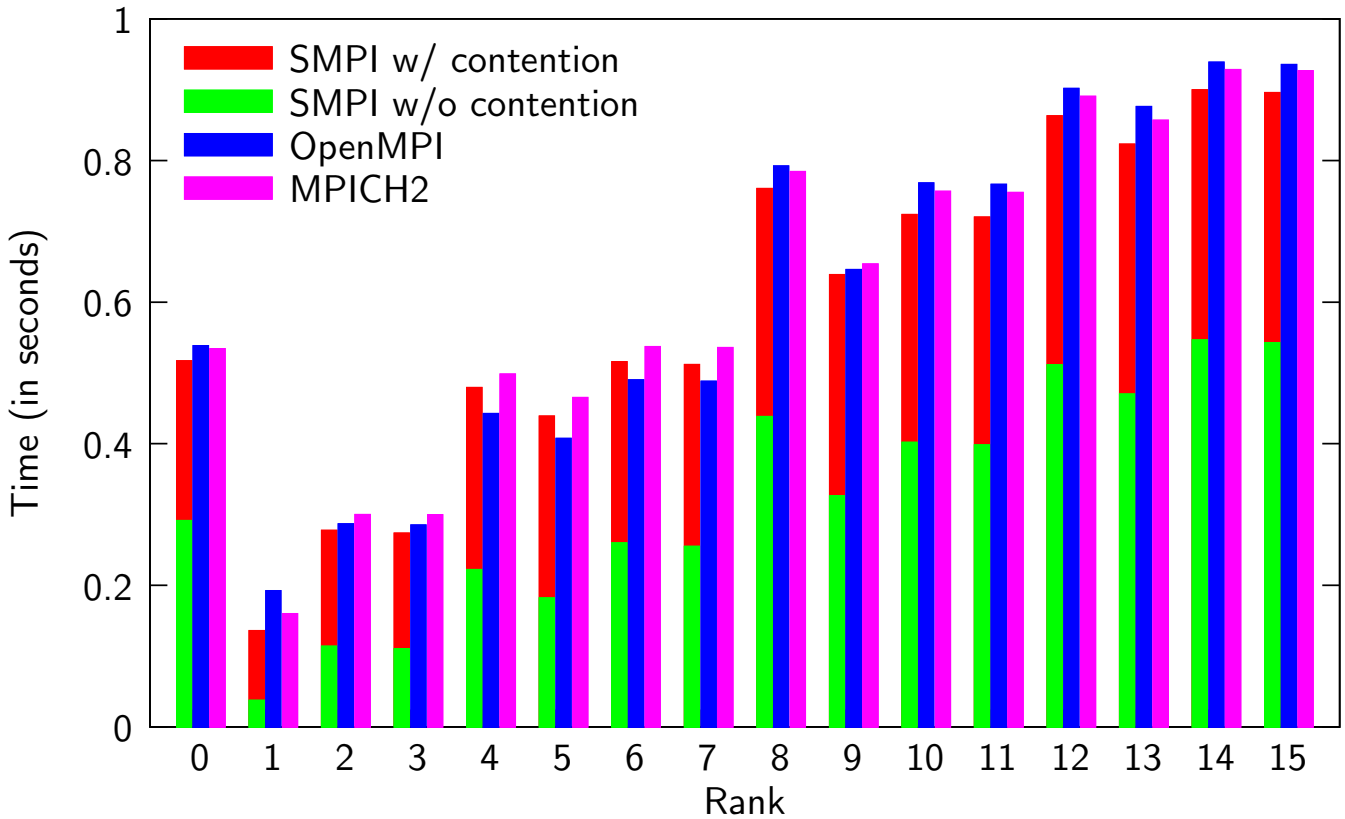
- Real world (OpenMPI, MPICH2)
  - Dynamic selection of tuned algorithms
  - Depends on the number of processes and message size
- Simulated world (SMPI)
  - Smaller variety of algorithms
- For a sake of comparison: use a manual implementation for real and in simulation

One-to-many: MPI_Scatter



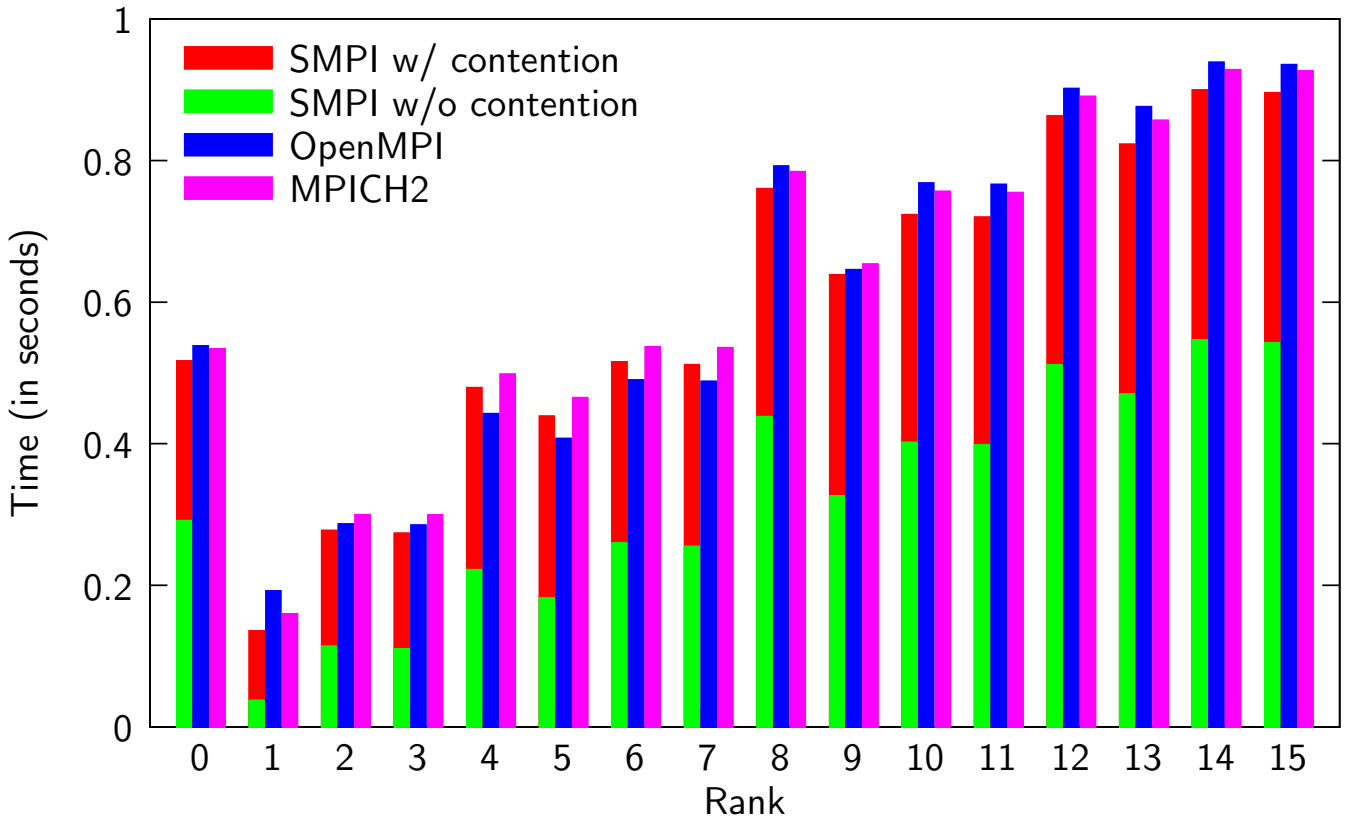
- Algorithm: A binomial tree
- 64 MiB at the root, 4 MiB per process

Scatter: 16-processes test



- Comparison SMPI/MPICH2 $\Leftrightarrow$ OpenMPI/MPICH2: error 5.3%

Scatter: 16-processes test



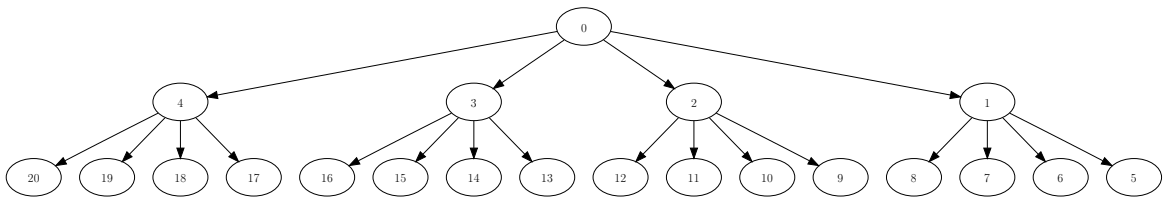
- Comparison SMPI/MPICH2 $\Leftrightarrow$ OpenMPI/MPICH2: error 5.3%
- Taking contention into account is important

Outline

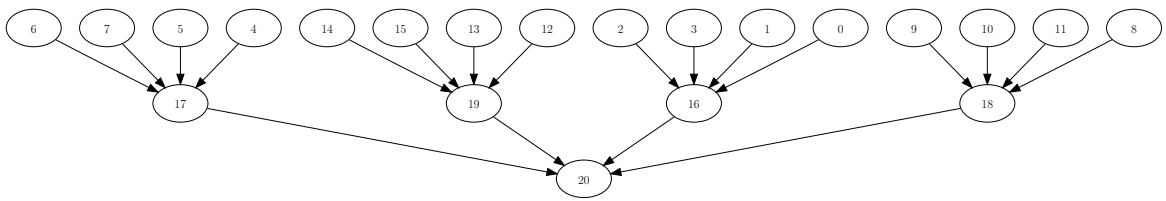
- Introduction
- Accuracy
- Scalability
  - DT NAS Benchmark
  - Reducing the Memory Footprint
  - Reducing the Memory Footprint – Results on DT
- Speed
- Conclusions and Future Work

Data Traffic (DT)

- Not many computations (excepted the graph construction)
- Possible communication schemes:
  - WhiteHole (WH)



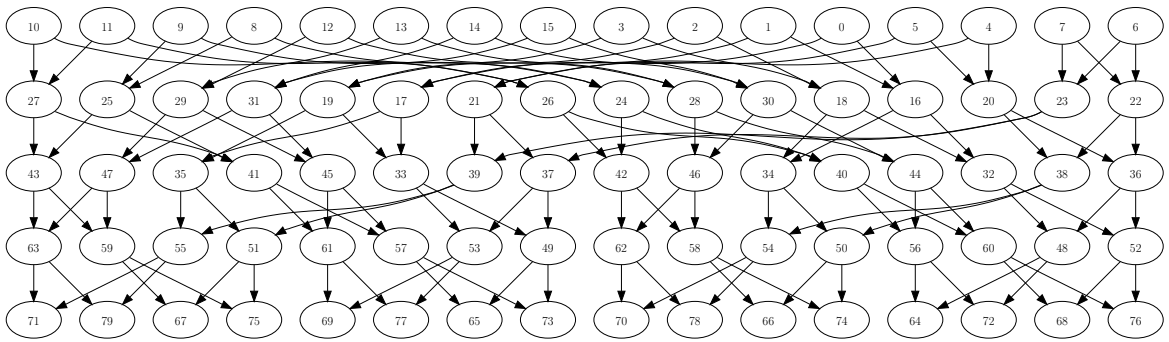
  - BlackHole (BH)



- The problem size determines the number of processes
  - Classe A: 21 processes
  - Classe B: 43 processes
  - Classe C: 85 processes

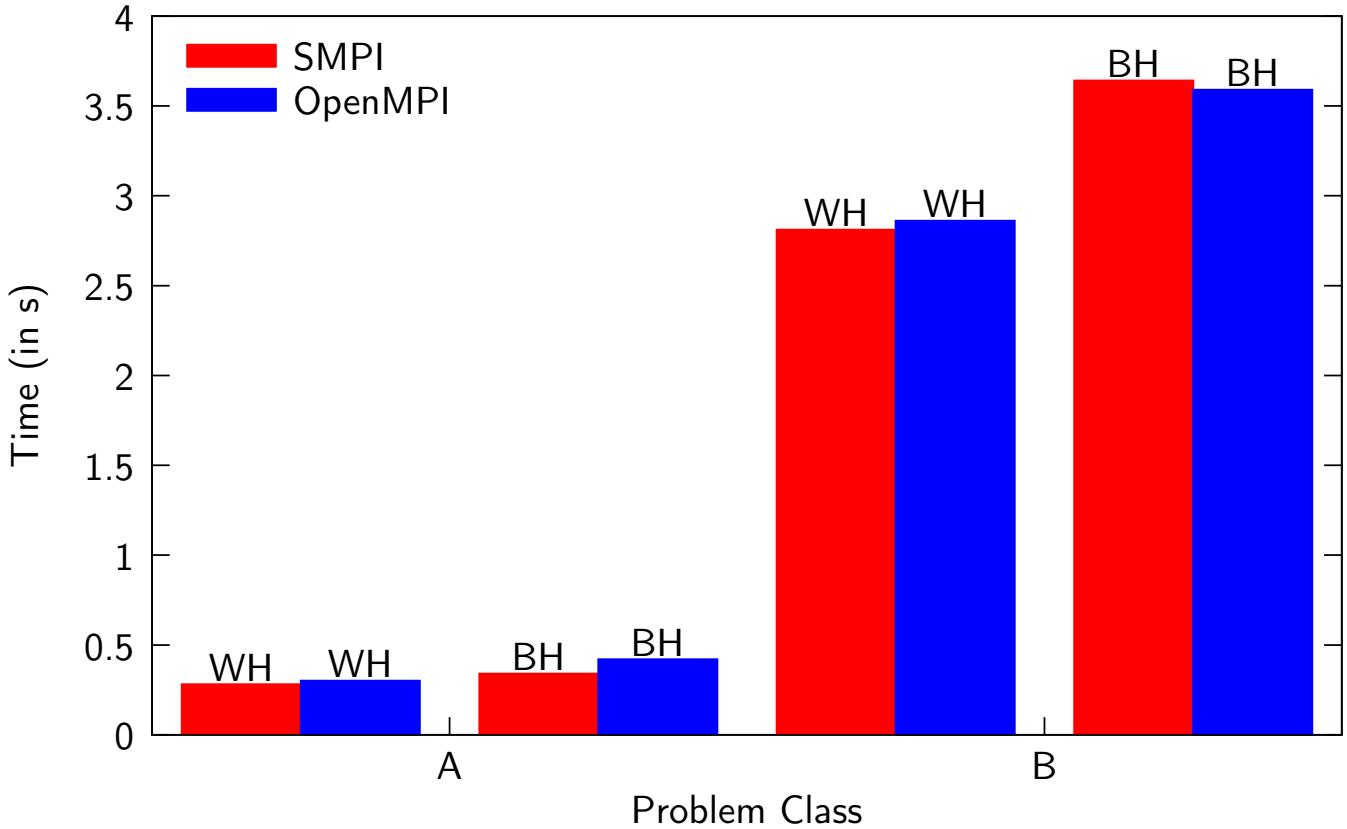
Data Traffic (DT)

- Not many computations (excepted the graph construction)
- Possible communication schemes:
  - Shuffle (SH)



- The problem size determines the number of processes
  - Classe A: 80 processes
  - Classe B: 192 processes
  - Classe C: 448 processes

DT: Comparison with OpenMPI



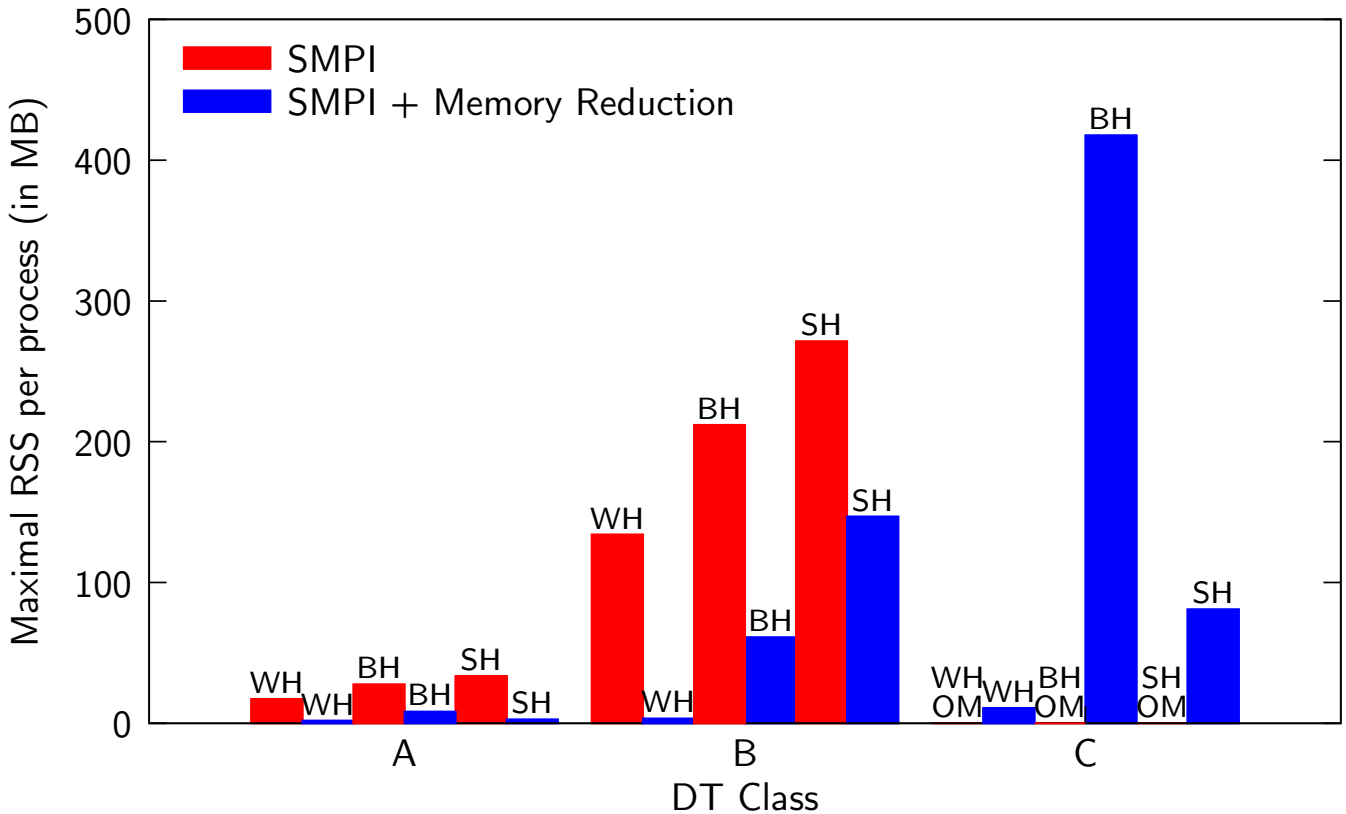
- Average error: 8.11%
- Not enough memory to simulate class C (and above)

Reducing the Memory Footprint

- Idea: Share arrays between processes
  - Pros: Allocate once, use plenty
  - Pros: Simulated times stay valid
  - Cons: Computed results become erroneous
- Implemented as (optional) macros

```
double* data = (double*)SMPI_SHARED_MALLOC(...);
...
SMPI_SHARED_FREE(data);
```

Reducing the Memory Footprint – Results on DT



- Average reduction by factor of 11.9 (maximum 40.5x)
- Class C can now be simulated

Outline

Embarassingly Parallel (EP) Benchmark

- No communication
  - Only result agregation in the end
- Computation shared among processes

- Ideal parallel execution but simulation worst case
  - Process simulation is fully serialized
- Simulation takes more time than actual execution
  - But on less resources

Reducing the Simulation Time

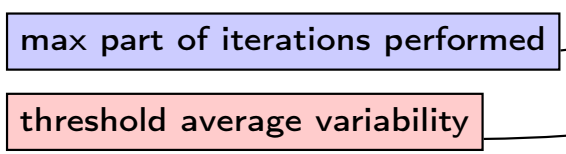
- Idea: Do not execute all the iterations
- Use sampling instead
  - LOCAL: each process executes a specified number of iterations
  - GLOBAL: a specified number of samples is produced by all processors
- Remaining iterations are replaced by average of measured values
- Implemented as (optional) macros

```
for(i = 0; i < n; i++) SMPI_SAMPLE_LOCAL( 0.75*n , 0.01 )
}
...
for(j = 0; j < k; j++) SMPI_SAMPLE_GLOBAL(0.5*k,0.01) {
    ...
}
```

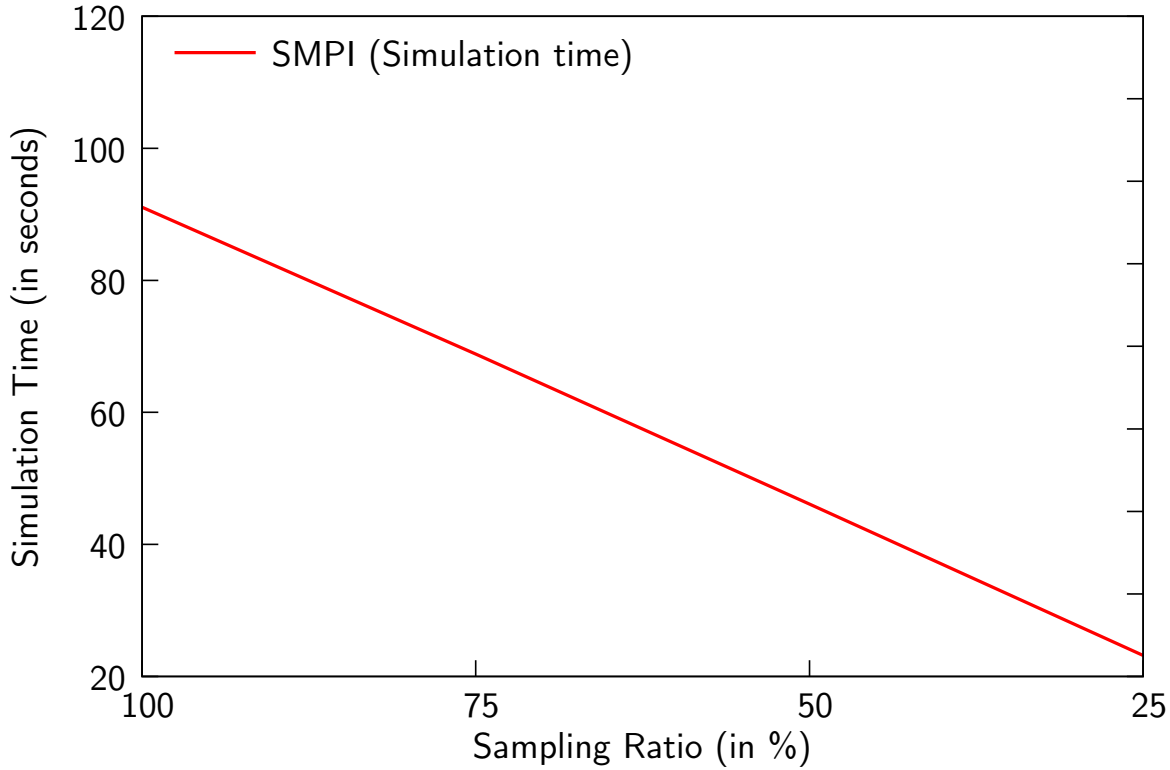
Reducing the Simulation Time

- Idea: Do not execute all the iterations
- Use sampling instead
  - LOCAL: each process executes a specified number of iterations
  - GLOBAL: a specified number of samples is produced by all processors
- Remaining iterations are replaced by average of measured values
- Implemented as (optional) macros

```
for(i = 0; i < n; i++) SMPI_SAMPLE_LOCAL( 0.75*n , 0.01 )
}
...
for(j = 0; j < k; j++) SMPI_SAMPLE_GLOBAL(0.5*k,0.01) {
    ...
}
```

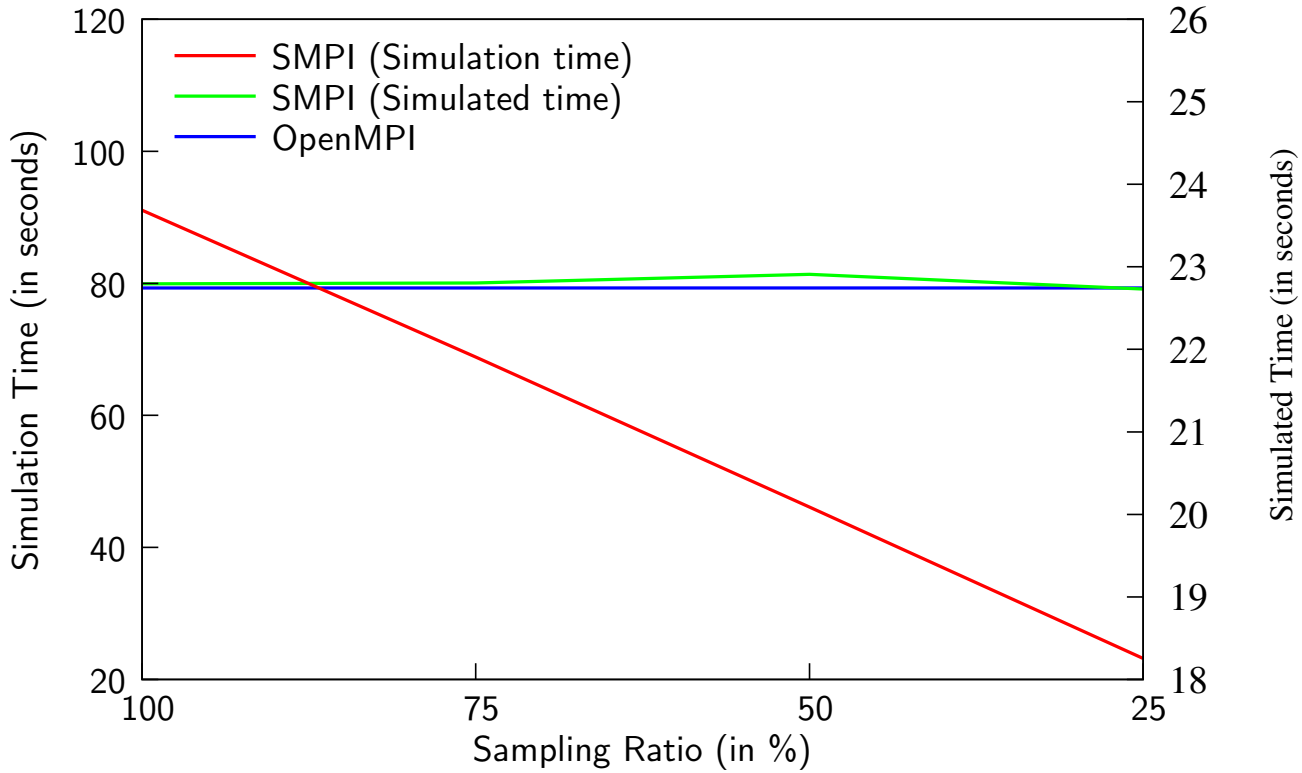


Reducing the Simulation Time – Results on EP



- Reduction of the simulation time linear with the sampling ratio

Reducing the Simulation Time – Results on EP



- Reduction of the simulation time linear with the sampling ratio
- No impact on simulated time accuracy

Outline

- Introduction
- Accuracy
- Scalability
- Speed
- Conclusions and Future Work

Conclusions

- SMPI is a functional simulation tool
  - Reproducible simulation of unmodified MPI application
  - On a single node
  - Open Source and freely available

    http://simgrid.gforge.inria.fr
- New network model for more accuracy
- Optional techniques to reduce
  - Memory footprint ⇒ Scalability
  - Simulation time ⇒ Speed
- Validation of main ideas
  - Identification of strengths and limitations

Future Work

- Short term
  - Automate the privatization process that enables unmodified code to compile
  - Handle packet serializing phenomenons (Ongoing)
- Mid term
  - Model other network interconnects: Myrinet, Infiniband
- Long term
  - I/O simulation
  - Automatic memory factoring and loop sampling
  - Simulation of a full implementation (OpenMPI or MPICH2)