www.bsc.es

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

xSim – The Extreme-Scale Simulator

Janko Strassburg

With Material from ORNL
Oak Ridge National Lab

- Motivation
- Overview
- Network Models
- Examples
- Conclusion

Motivation

- Predict behaviour on different system
- Find bottlenecks, sweet spot, scaling problems
- Easier then running on several machines
- Reproducible

Overview

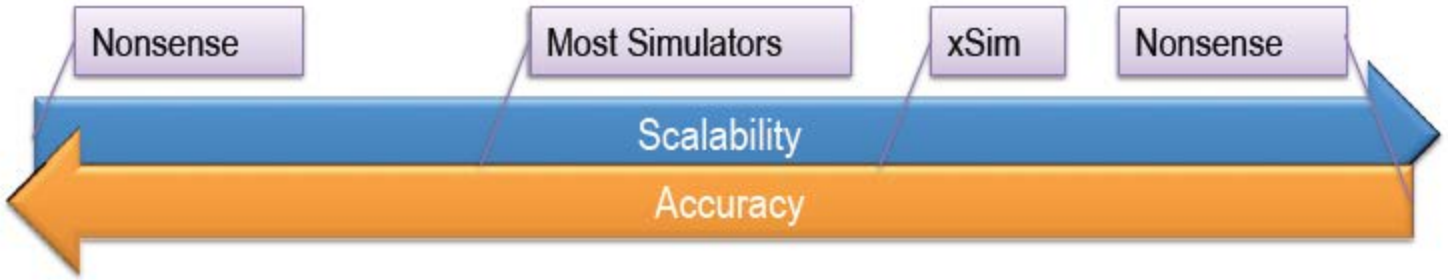
- Several existing simulators include JCAS, BigSim and MuPi
  - Limitations
- Highly scalable solution
  - trade off accuracy in exchange of node oversubscription simulation
- Execution of real applications, algorithms or their models atop a simulated HPC environment for
  - Performance evaluation
    - identification of resource contention
    - underutilization issues
  - Investigation at extreme scale, beyond the capabilities of existing simulation efforts

Facilitating HPC Hardware/Software Co- Design Through Simulation

- Parallel discrete event simulation (PDES) to emulate the behavior of future architecture choices

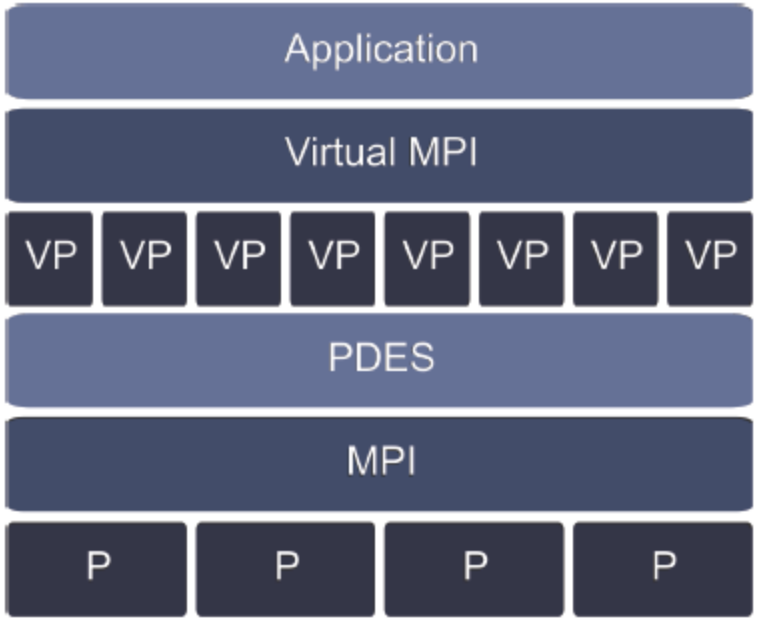
- Execution of real applications, algorithms or their models atop a simulated HPC environment for:
  - Performance evaluation, including identification of resource contention and underutilization issues
  - Investigation at extreme scale, beyond the capabilities of existing simulation efforts

- xSim: Highly scalable solution that trades off accuracy



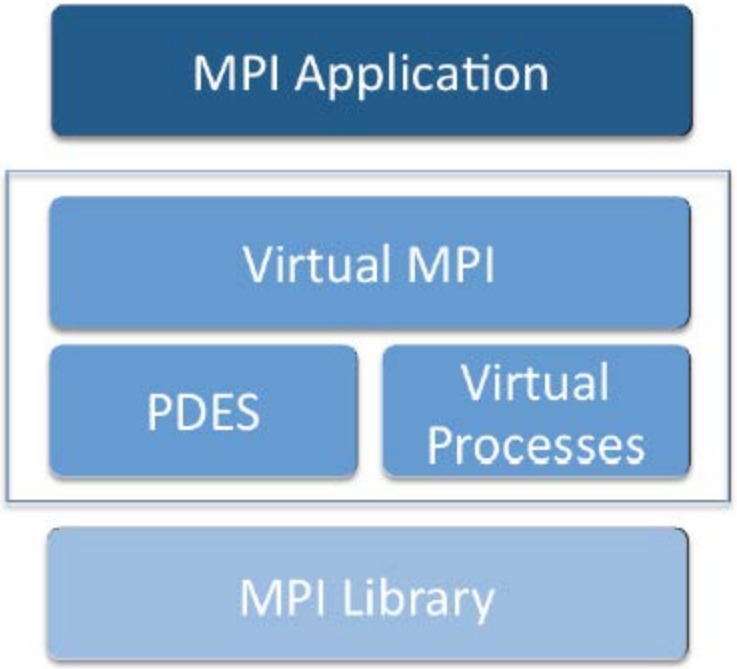
Overview

- Combining highly oversubscribed execution, a virtual MPI, and a time-accurate PDES (Parallel discrete event simulation)
- PDES uses the native MPI and simulates virtual processors
- The virtual processors expose a virtual MPI to applications



Overview

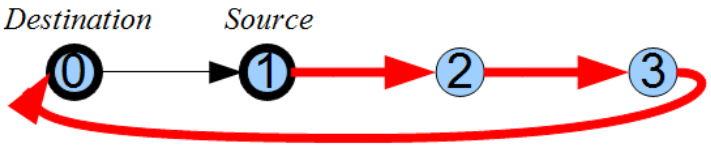
- The simulator is a library
- Utilizes PMPI to intercept MPI calls and to hide the PDES
- Easy to use:
  - Replace MPI header for xSim
  - Compile and link with the simulator library
  - Run the MPI program mpirun – np <np> ./prog –xsim-np <vp>
- Support for C and Fortran MPI applications



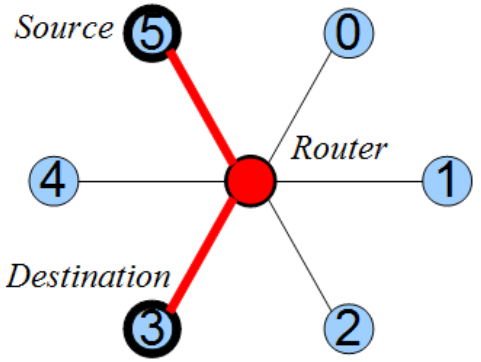
Network Models

- Support for various networking models
  - Analyze existing hardware conditions
  - Test for differing architectures
- No accounting for traffic, congestion and any subsequent re-routing of messages

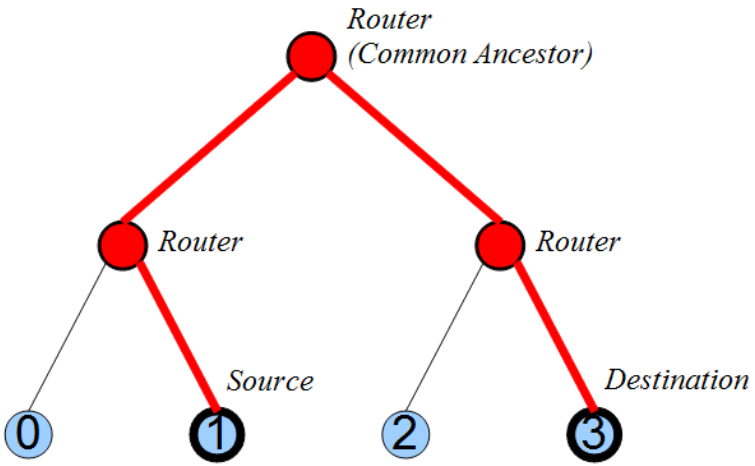
Network Models



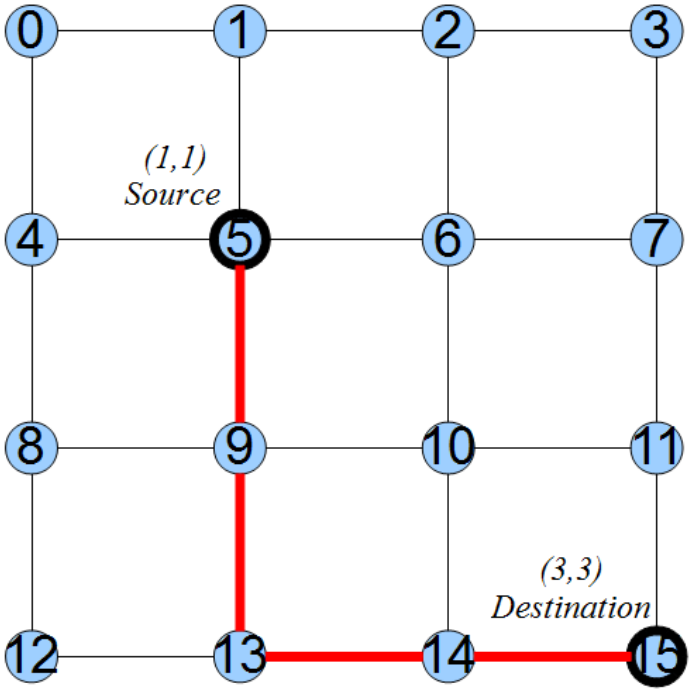
Unidirectional Ring



Star

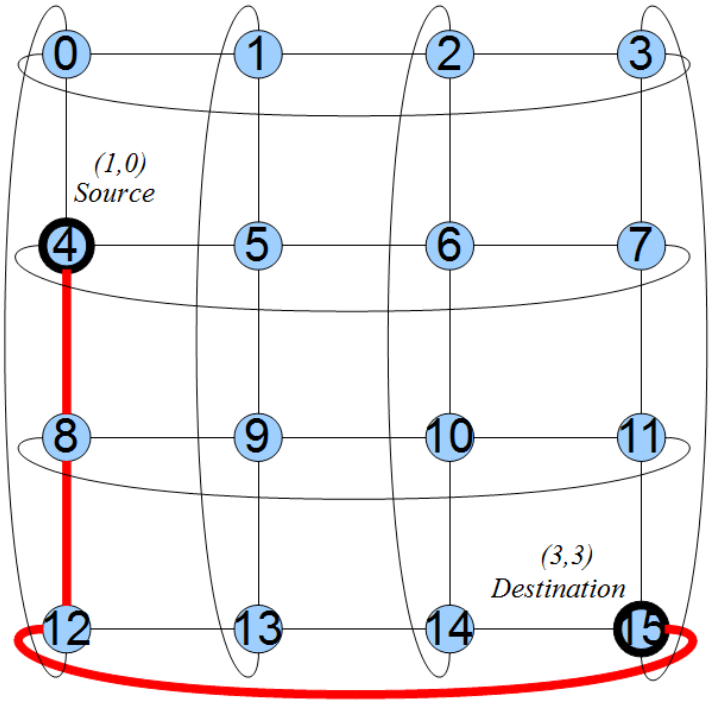


Tree

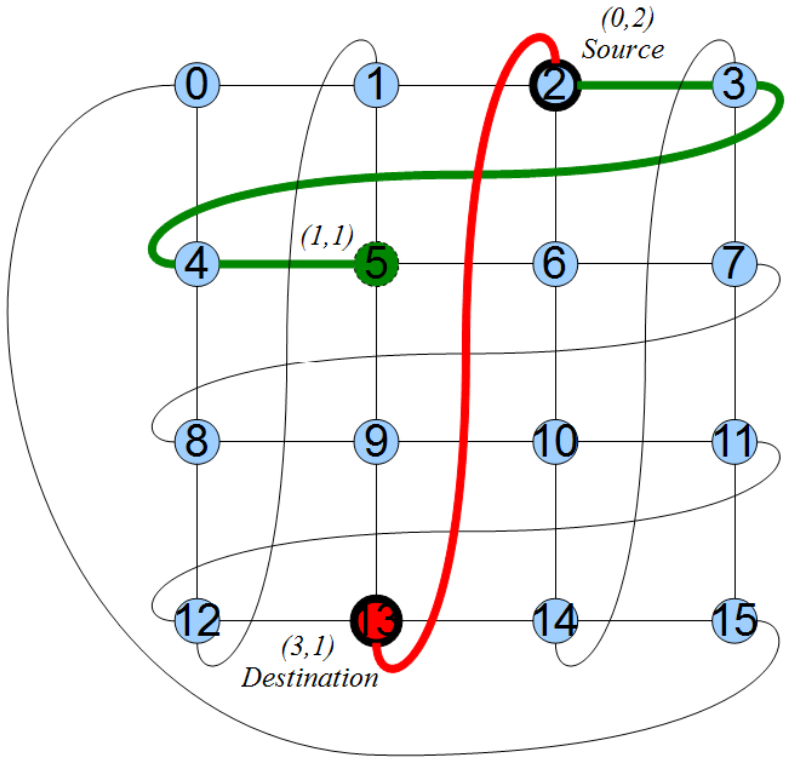


Mesh

Network Models

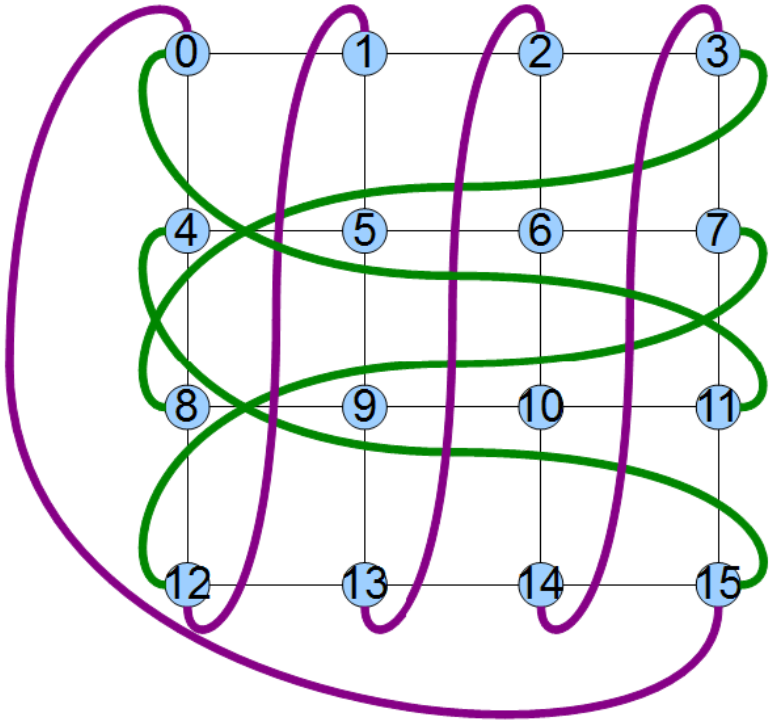


Torus



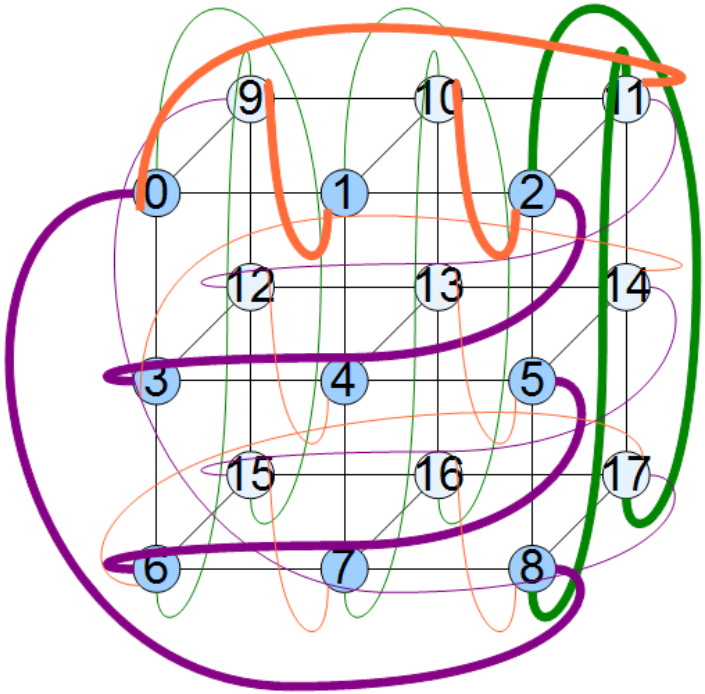
Twisted Torus

Network Models



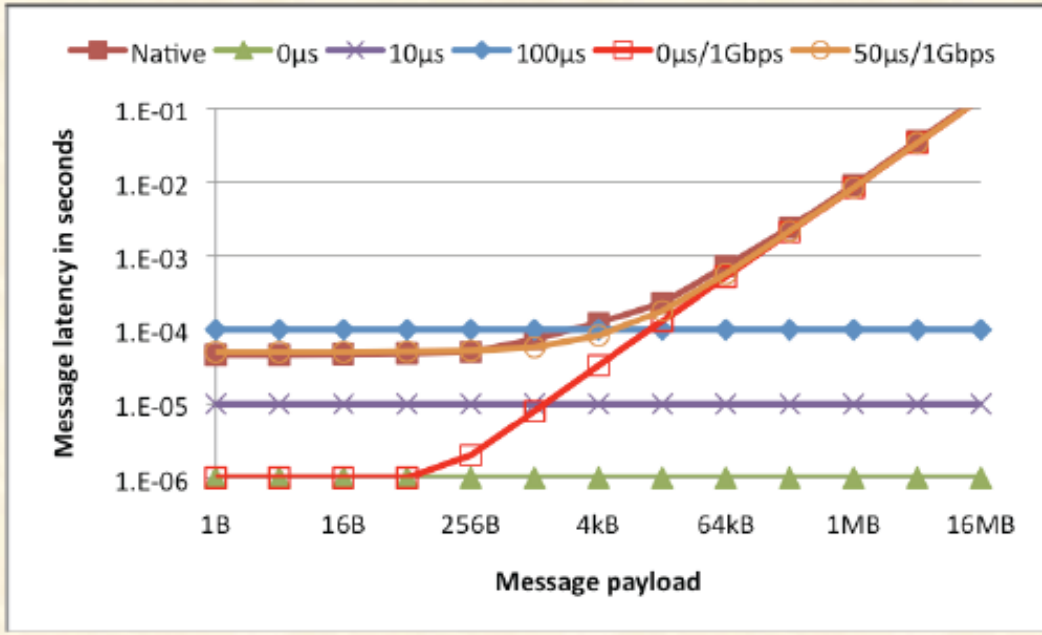
Twisted Torus with Toroidal Jump

Twisted Torus with Toroidal Degree



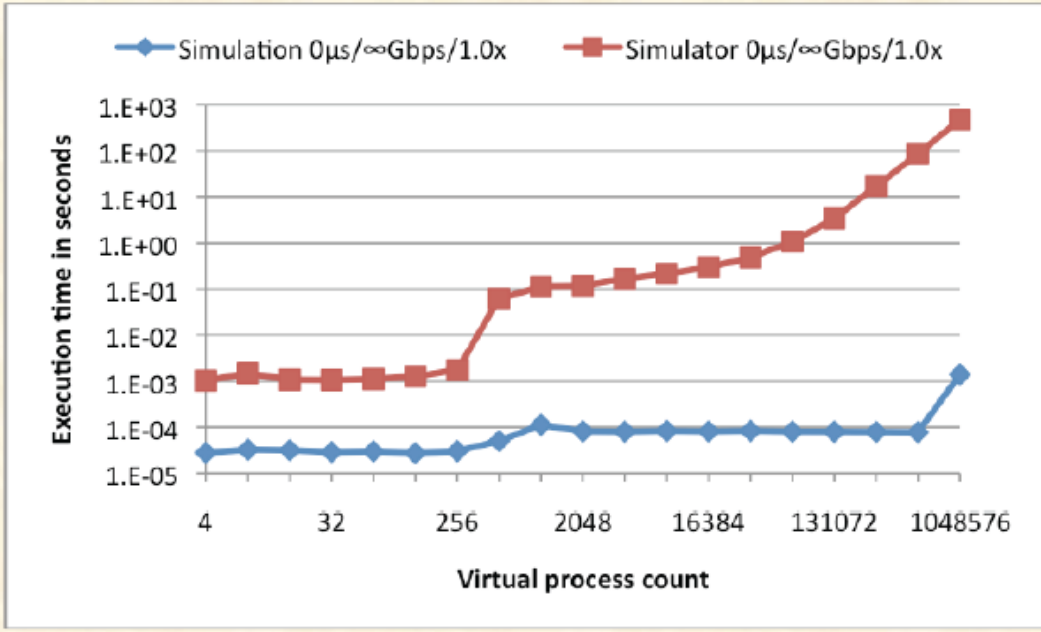
Experimental Results: Network Model

- Model allows to define network architecture, latency and bandwidth
- Basic star network at the time of writing this paper
- Model can be set to 0µs and ∞Gbps as baseline
- 50µs and 1Gbps roughly represented the native test environment
  - 4 Intel dual-core 2.13GHz nodes with 2GB of memory each
  - Ubuntu 8.04 64-bit Linux
  - Open MPI 1.4.2 with multi-threading support



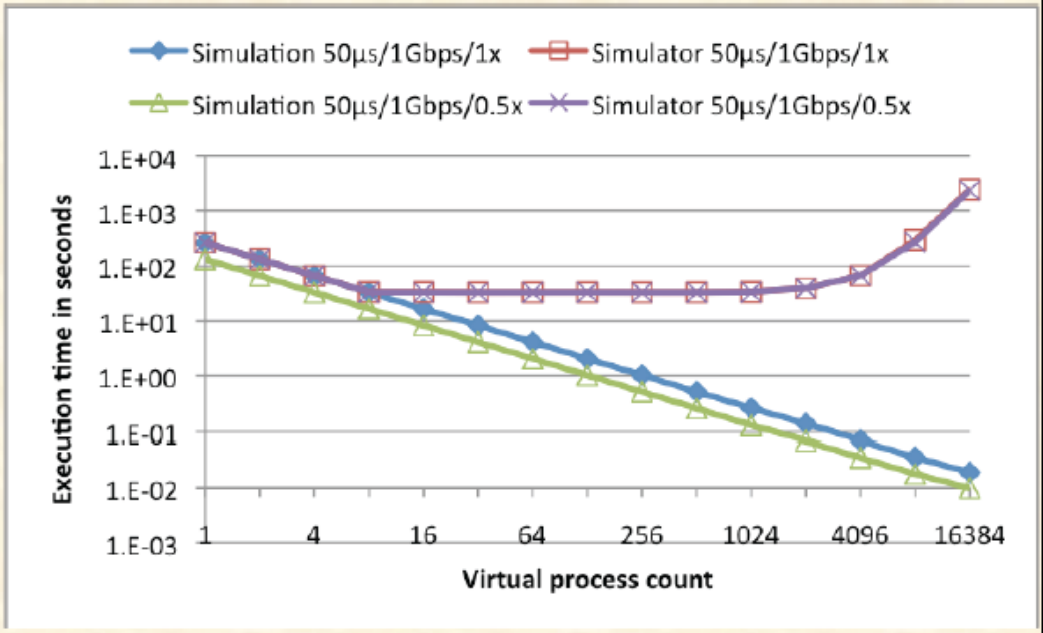
Experimental Results: Processor Model

- Model allows to set relative speed to a future processor
- Basic scaling model
- Model can be set to 1.0x for baseline numbers
- MPI `hello world` scales to 1M+ VPs on 4 nodes with 4GB total stack (4kB/VP)
- Simulation (application)
  - Constant execution time
  - <1024 VPs: Noisy clock
- Simulator
  - >256 VPs: Output buffer issues



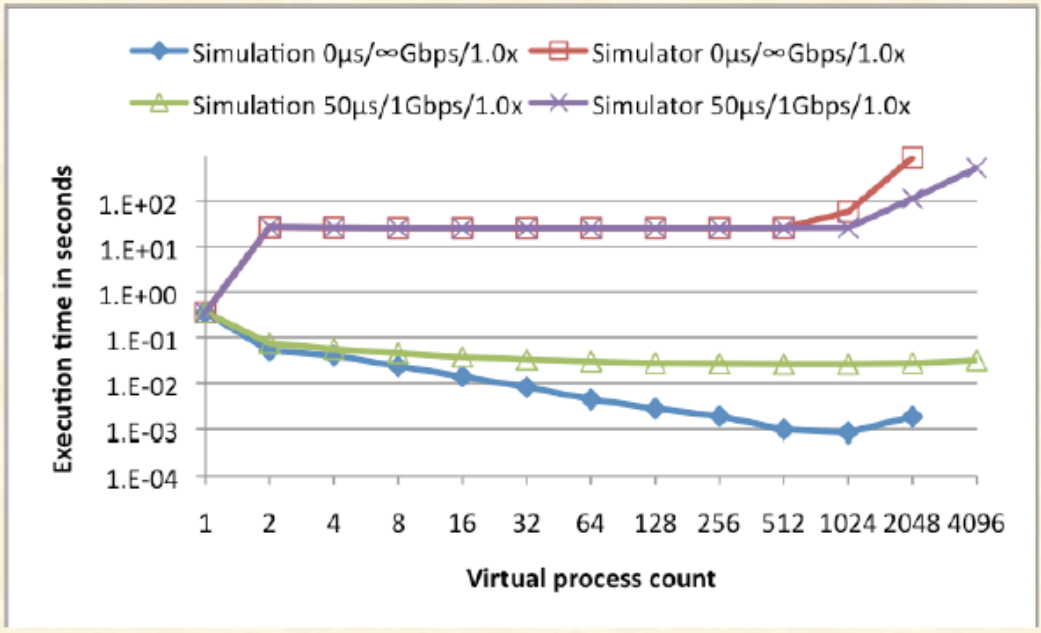
Experimental Results: Scaling up a computation-intensive application

- Basic PI Monte Carlo solver
- Network model:
  - Star, 50µs and 1Gbps
- Processor model
  - 1x (32kB stack/VP)
  - 0.5x (32kB stack/VP)
- Simulation (application)
  - Perfect scaling
- Simulator
  - <= 8 VPs: 0% overhead on the 8 processor cores
  - >= 4096 VPs: comm. load dominates

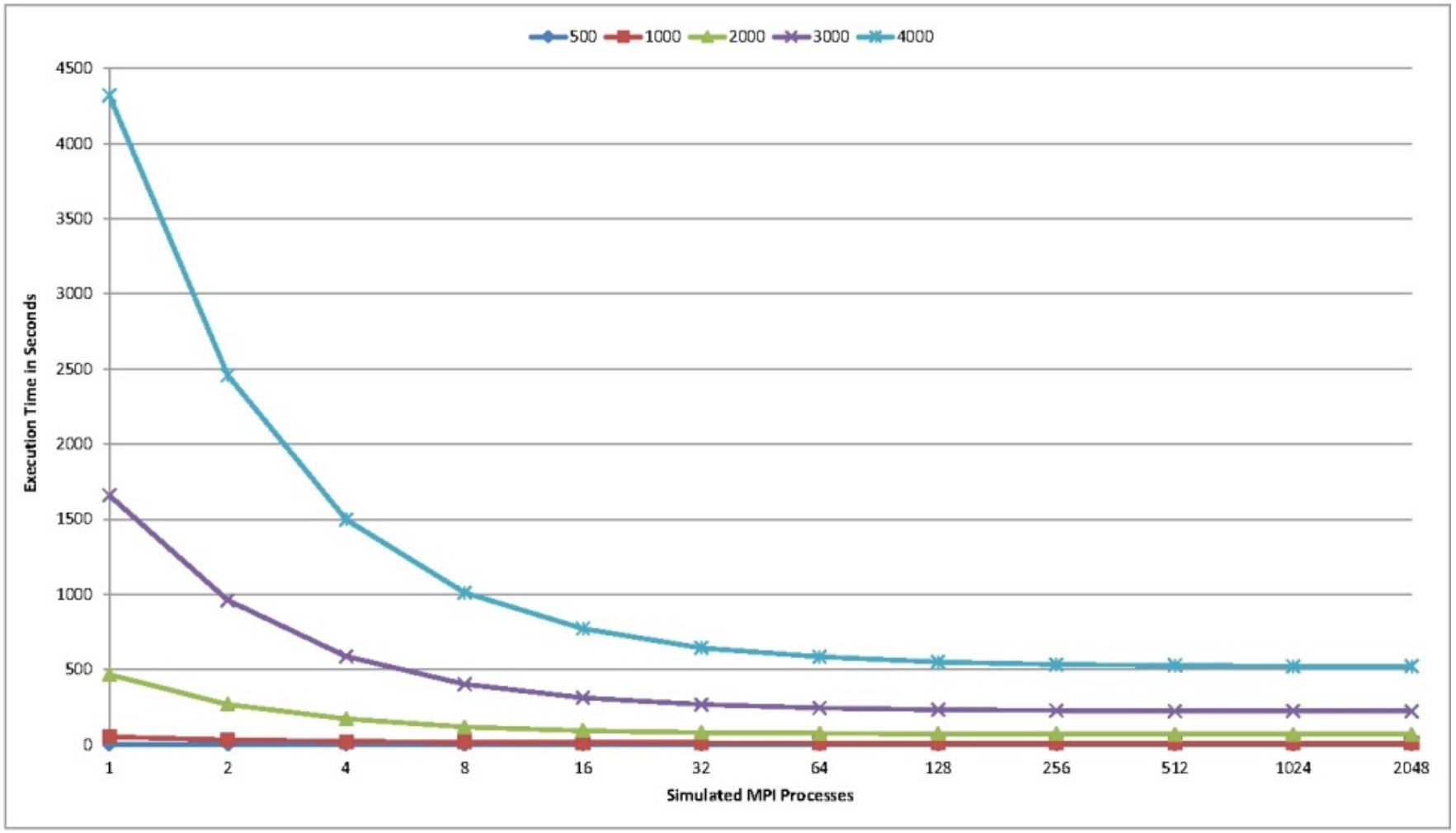


Experimental Results: Scaling up a commmunication-intensive application

- Basic 1-D Heat Eq. solver
- Network model:
  - Star, 50µs and 1Gbps
  - Star, 0µs and ∞Gbps
- Processor model
  - 1x (32kB stack/VP)
- Simulation (application)
  - Limited scaling
- Simulator
  - 1 VP: no communication, therefore no overhead
  - >= 1024 VPs: comm. load dominates

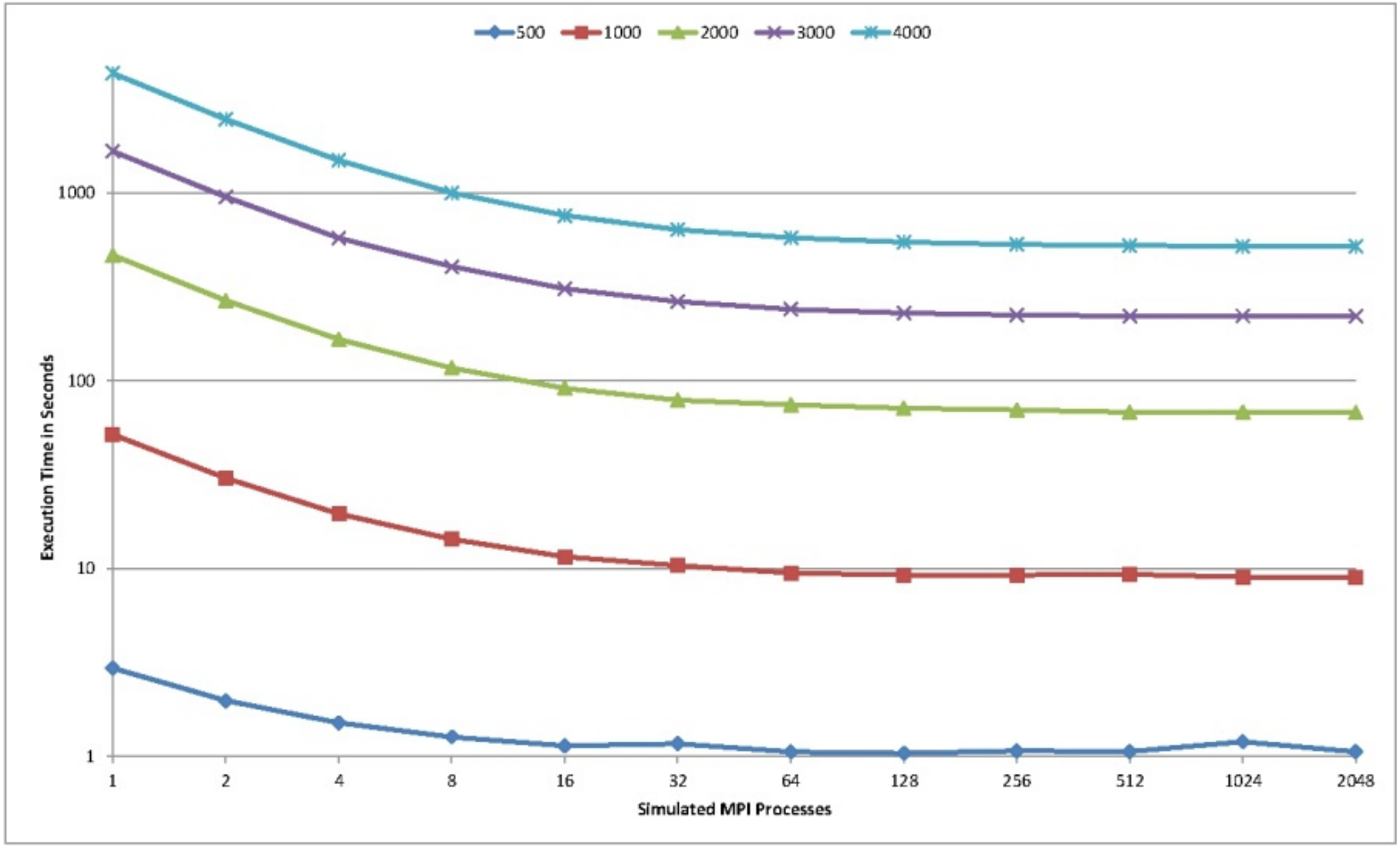


Core scaling – Monte Carlo MI



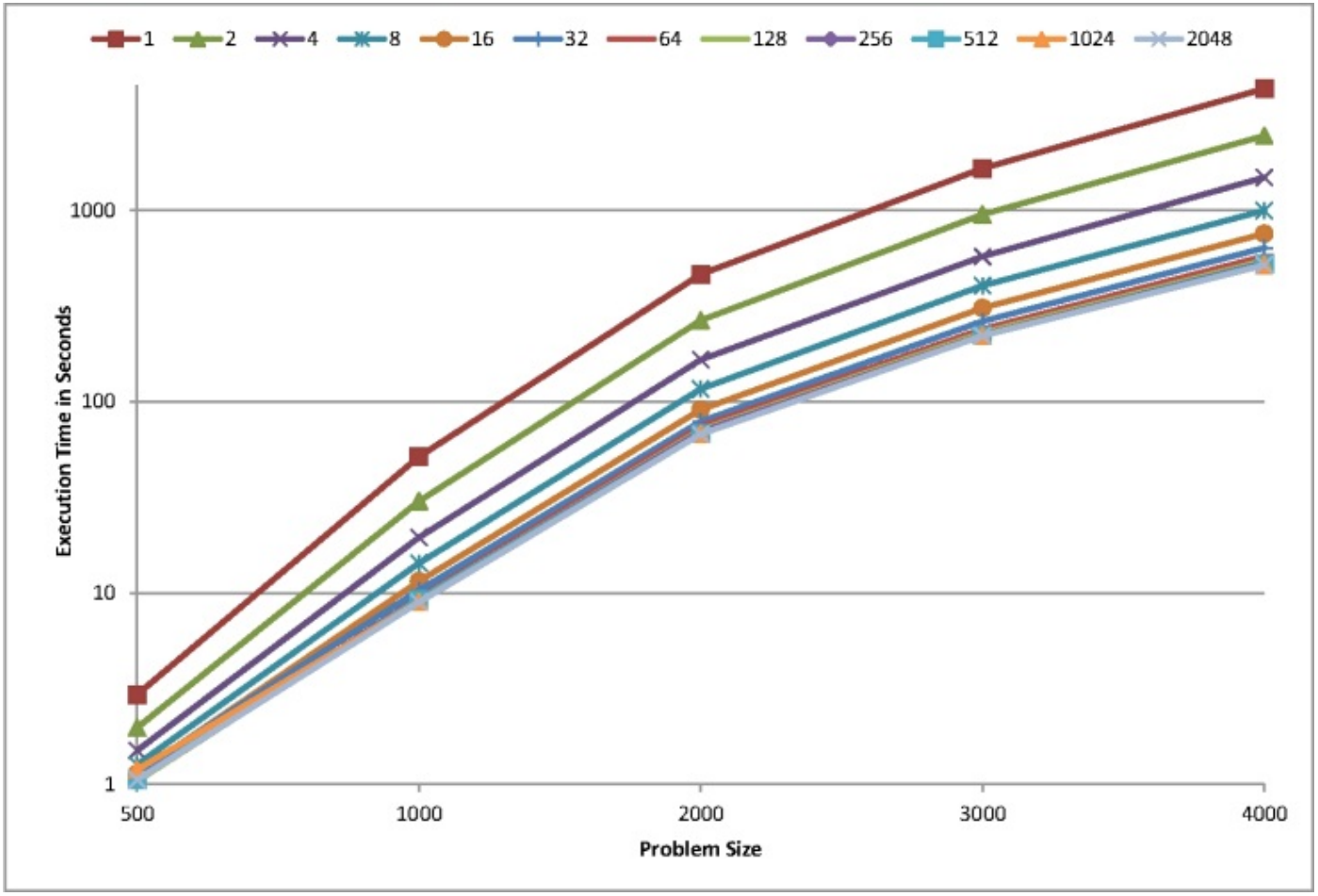
- 960 Core system
- 240 cores for simulation due to memory bandwidth restrictions

Core scaling

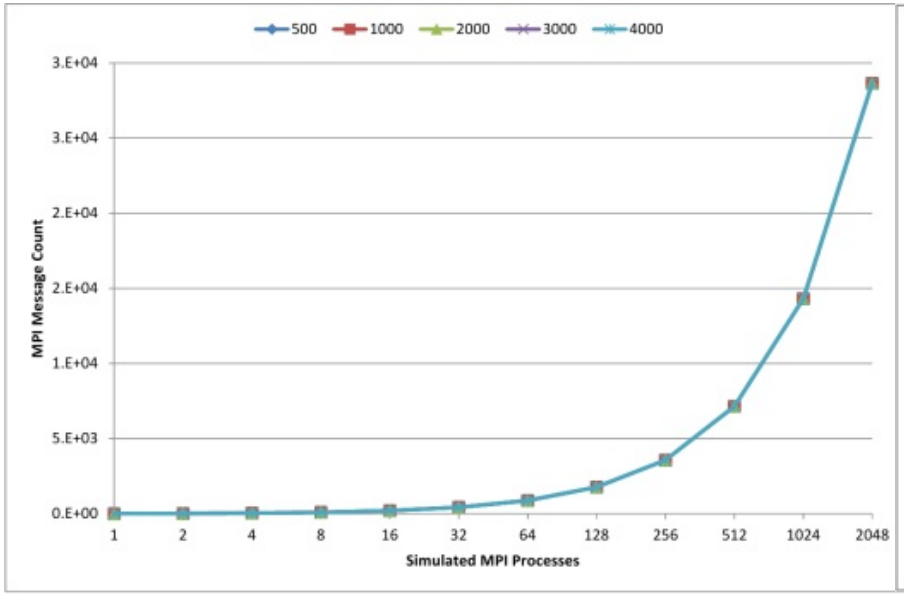


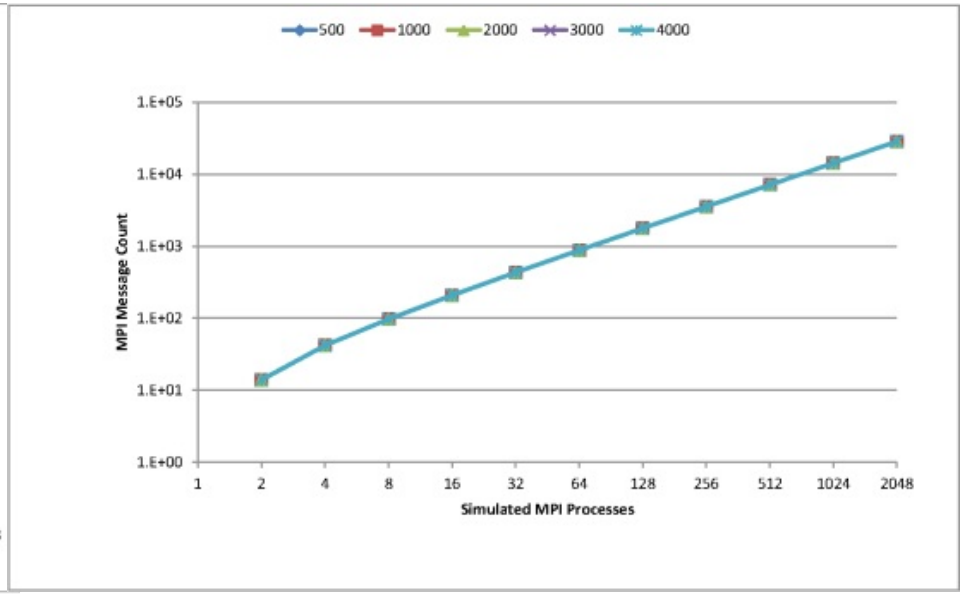
- Recurring behaviour for increasing MPI process sizes
- Scales well, then plateaus

Problem scaling



MPI message count scaling





- Simulator also gathers MPI statistics
- Linear increase of exchanged messages, as expected

Outcome & Conclusions

- Behaviour of code is predictable
- Simulation provides valuable information
- Forecast behaviour on varying systems possible
- Time and resource saving via simulation