BLUE WATERS

SUSTAINED PETASCALE COMPUTING

Characterizing the Influence of System Noise on Large-Scale Parallel Applications

Torsten Hoefler

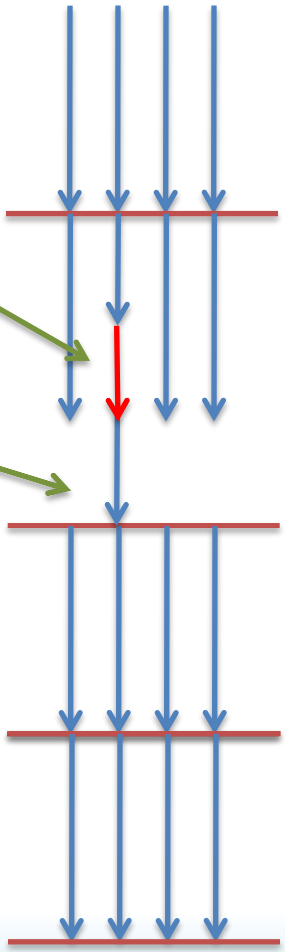
With contributions fromTimo Schneider and Andrew Lumsdaine

Scientific talk at RWTH Aachen, April 14th 2011

NCSA NSF IBM GREAT LAKES CONSORTIUM FOR PETASCALE COMPUTATION

System Noise – Introduction and History

- CPUs are time-shared
  - Deamons, interrupts, etc. steal cycles
  - No problem for single-core performance
    - Maximum seen: 0.26%, average: 0.05% overhead
  - “Resonance” at large scale (Petrini et al ’03)
- Numerous studies
  - Theoretical (Agarwal’05, Tsafrir’05, Seelam’10)
  - Injection (Beckman’06, Ferreira’08)
  - Simulation (Sottile’04)

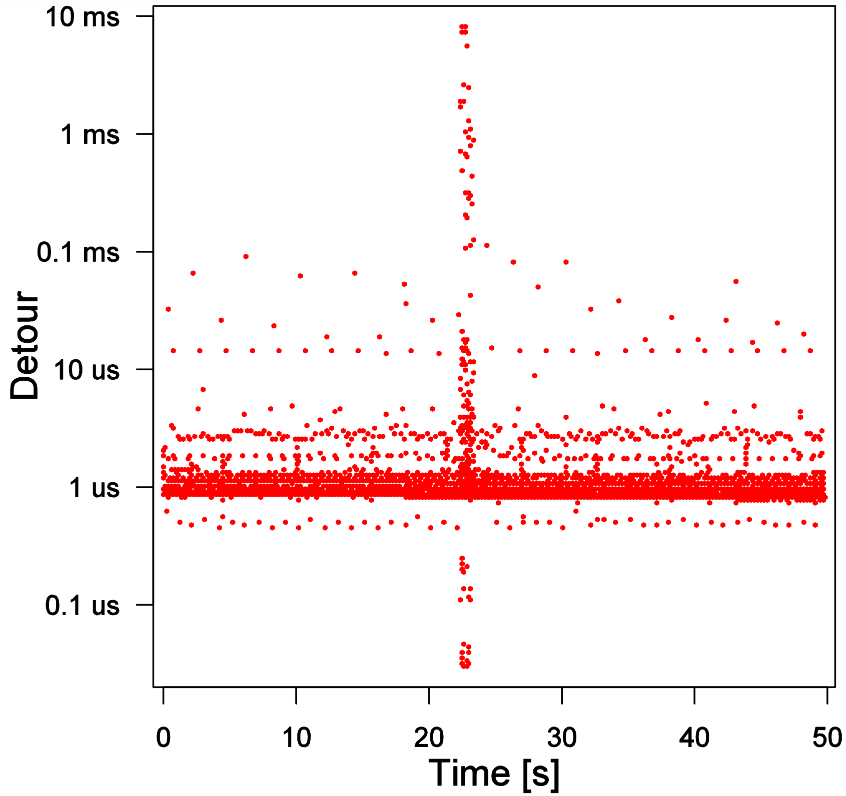


Measuring OS Noise on a Single Core

- Selfish Detour Benchmark (Beckman et al.)
  - Tight execution loop, benchmark iteration time
  - Record each outlier in iteration time
  - Improved detour (~30% better resolution)

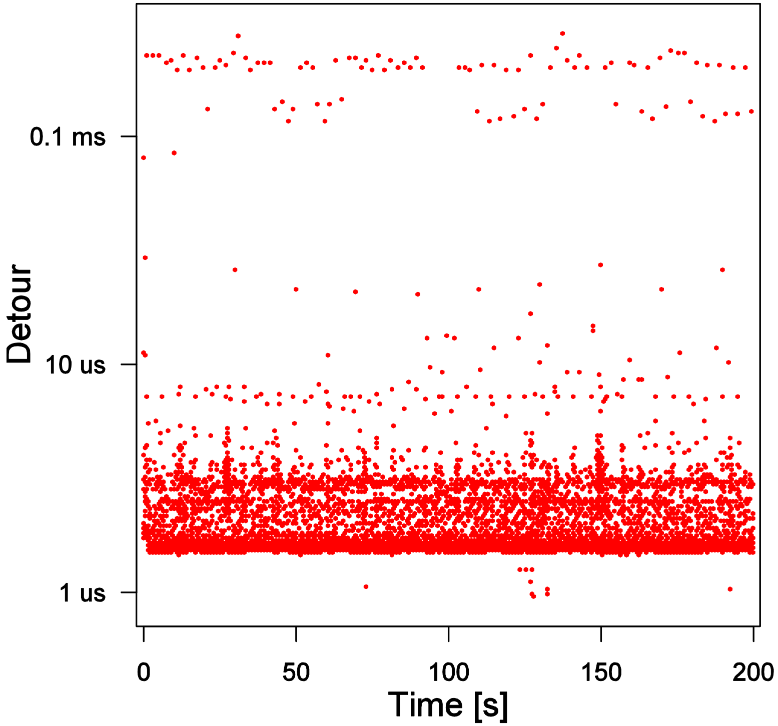
- Detour implemented in Netgauge benchmark tool
  - Also FWQ, FTQ (not used in this work)
  - Available at: `http://www.unixer.de/Netgauge`

Measurement Results – CHiC Linux (diskless)



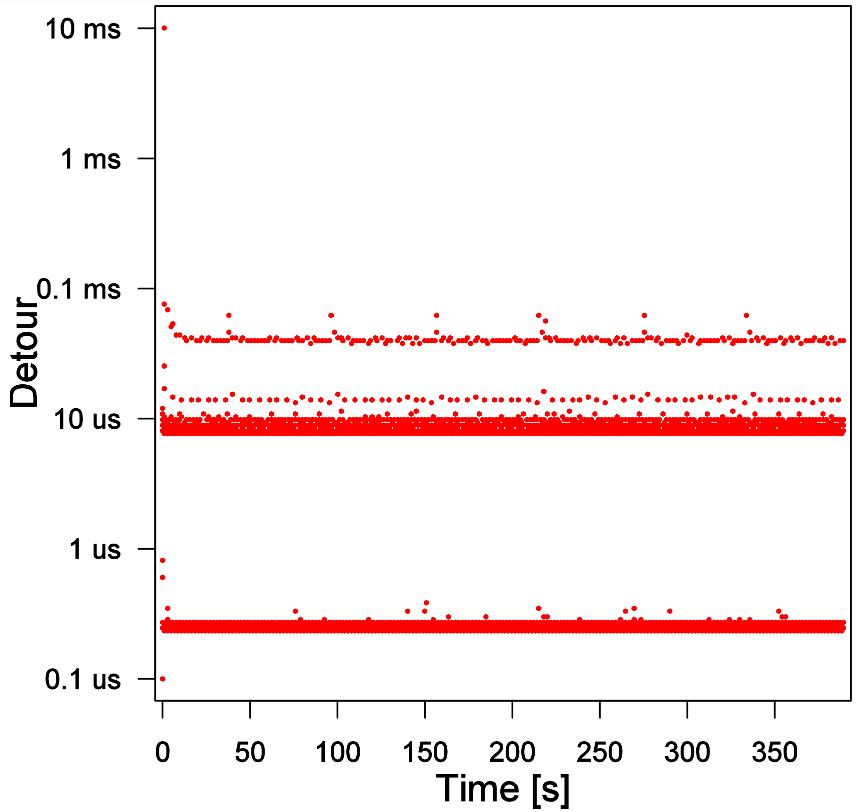
- 2152 Opteron cores, 11.2 Tflop/s Linux 2.6.18
- Resolution: 3.74 ns, noise overhead: 0.21%

Measurement Results – SGI Altix



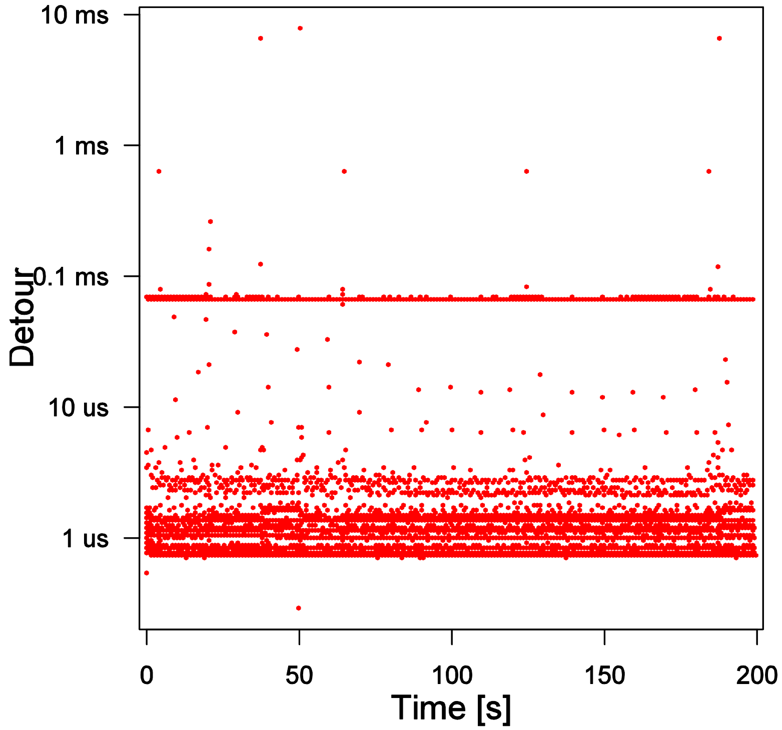
- Altix 4700, 2048 Itanium II cores, 13.1 Tflop/s, Linux 2.6.16
- Resolution: 25.1 ns, noise overhead: 0.05%

Measurement Results – BG/P ZeptoOS



- 164k PPC 450 cores, 485.6 Tflop/s, ZeptoOS 2.6.19.2
- Resolution: 29.1 ns, noise overhead: 0.08%

Measurement Results – Cray XT-4 (Jaguar)



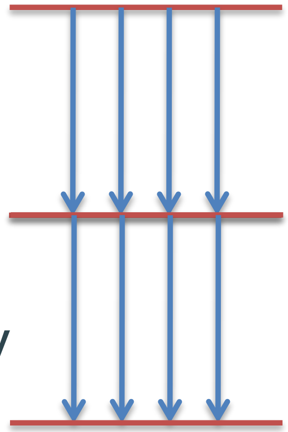
- 150k Opteron cores, 1.38 Pflop/s, Linux 2.6.16 CNL
- Resolution: 32.9 ns, noise overhead: 0.02%

An Analytical Model for Noise Propagation

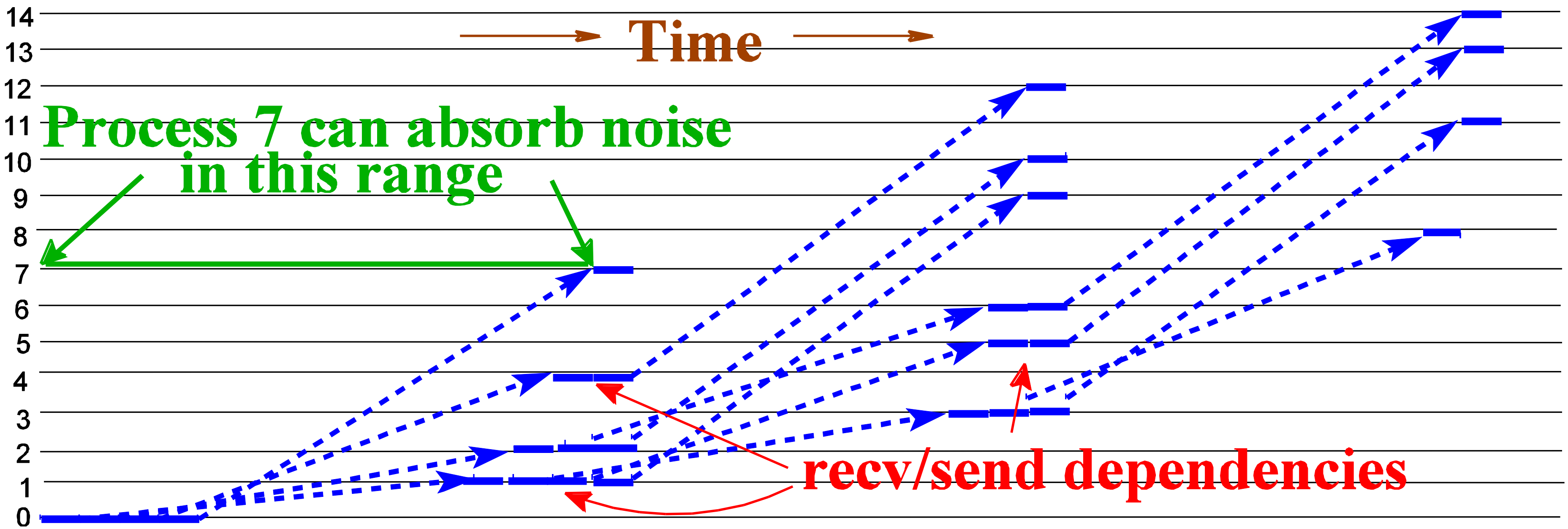
- Synchronization propagates or absorbs noise
  - Lamport's happens-before-relation for messages
  - Depends on relative time of send/recv (or wait)
- Several protocol-specific details
  - Small (eager), large (rendezvous), and nonblocking
- LogP model to express communication
  - Several missing pieces
  - LogGPS model (Ino et al.) captures most effects!
  - We added "O" to capture s/r overhead per byte

Collective Operations

- MPI-2.2: *"[...] a collective communication call may, or may not, have the effect of synchronizing all calling processes. This statement excludes, of course, the barrier function."*
- Main weaknesses in theoretical models:
  - Assumption 1: All collective operations synchronize
    - In fact, many do not (e.g., Bcast, Scan, Reduce, …)
  - Assumption 2: Collectives synchronize instantaneously
    - In fact, they (most likely) communicate with messages
  - Assumption 3: All processes leave collective simultaneously
    - In fact, they leave as early as possible (when data is consistent)

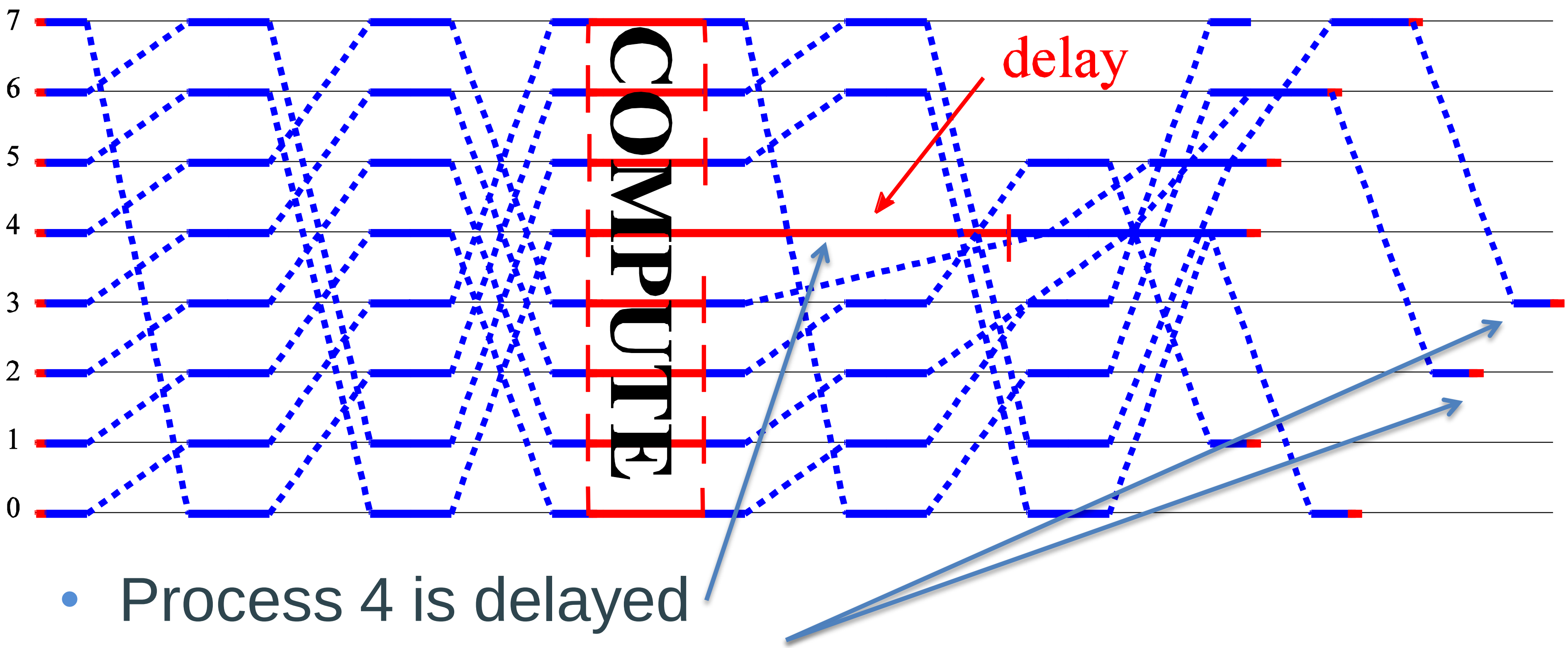


Example: Binomial Broadcast Tree



- Violates all three assumptions:
  - No global or instant synchronization, asynchronous exit

A Noisy Example – Dissemination Barrier



- Process 4 is delayed
  - Noise propagates "*wildly*" (of course deterministic)

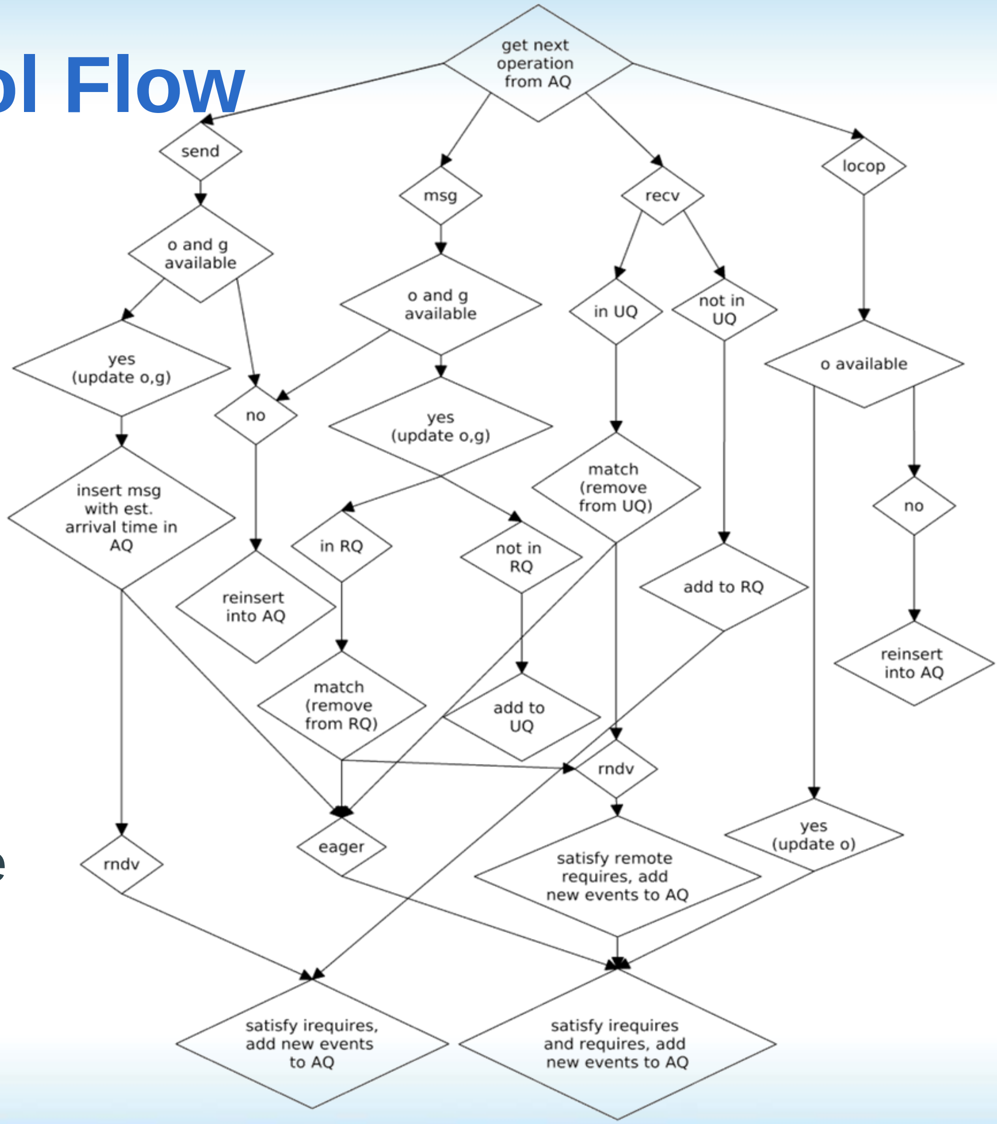
LogGOPS Simulation Framework

- Detailed analytical modeling is hard!
- Model-based (LogGOPS) simulator
  - Available at: `http://www.unixer.de/LogGOPSim`
  - Discrete-event simulation of MPI traces (<2% error) or collective operations (<1% error)
  - > 10^6 events per second!
- Allows for trace-based noise injection
  - In o_s, o_r, O, local reduction, and application time
- Validation
  - Simulations reproduce measurements by Beckman and Ferreira well!

- Details: Hoefler et al. LogGOPSim – Simulating Large-Scale Applications in the LogGOPS Model (Workshop on Large-Scale System and Application Performance, Best Paper)

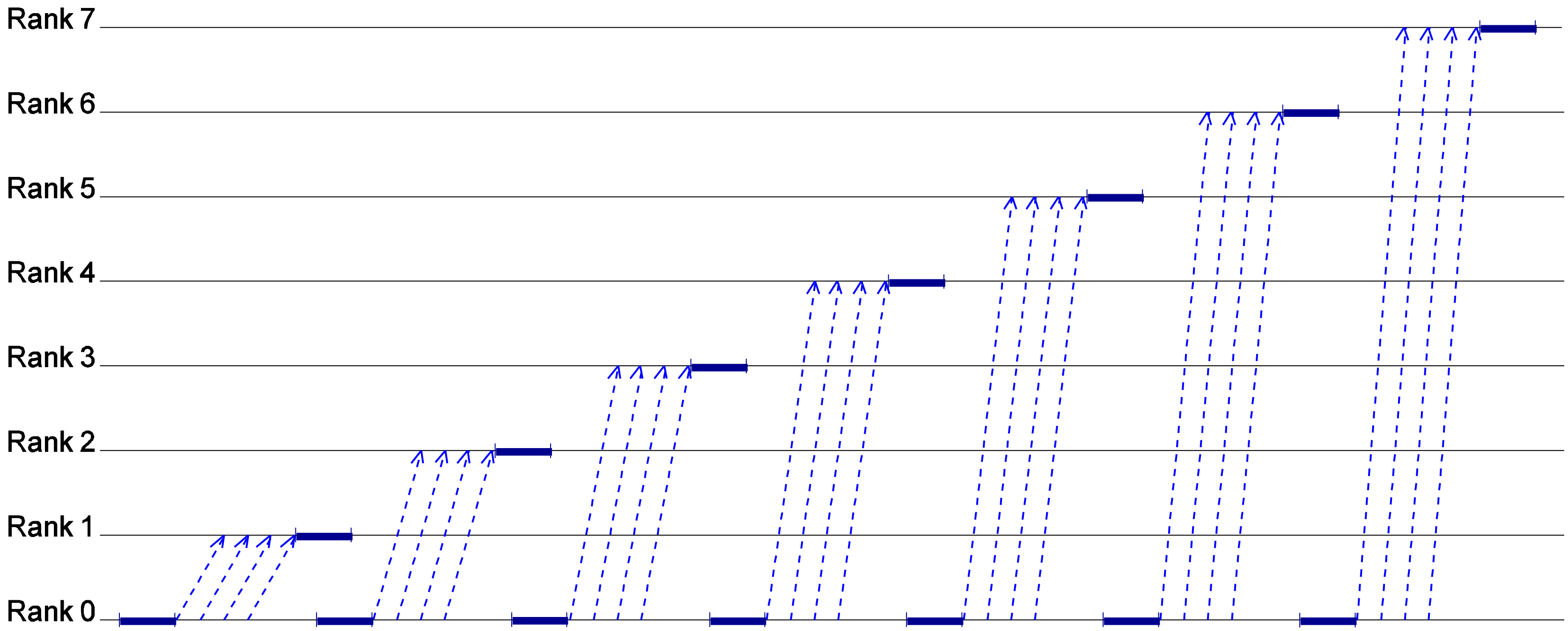
Simulator Core Control Flow

- Single queue design
  - Fast priority queue
1. Find executable ops
   - send, recv, msg, or loclop
2. Insert with current time
3. Fetch (globally) next op
   - check if it can be executed
   - match send/recv
   - re-insert if o, g not available
4. Lather, rinse, repeat



Verification – Linear Scatter

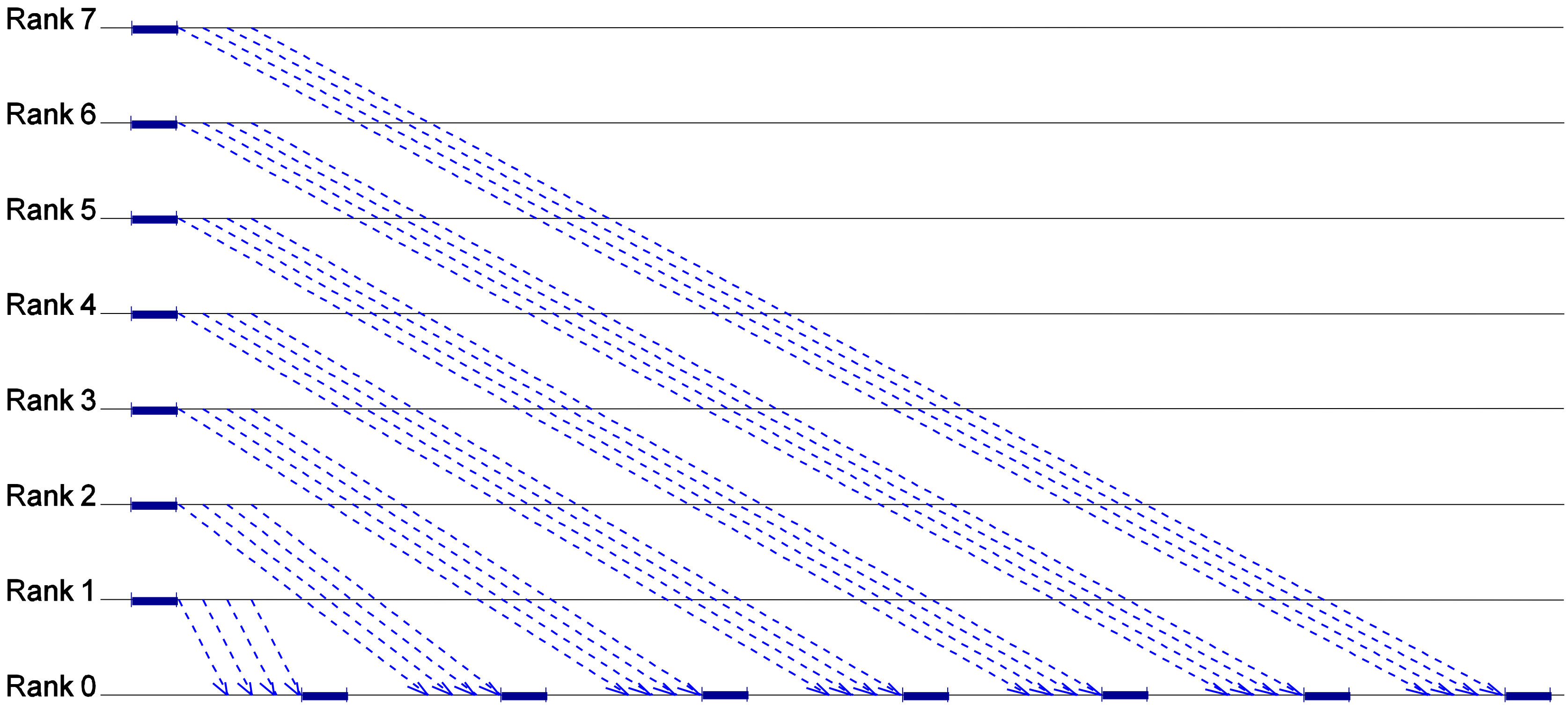
$$T_{scat} = 2o + L + \max\{(P-2)o + (P-1)sO), (P-2)g + (P-1)sG\}$$



- LogGOPS makes verification simple

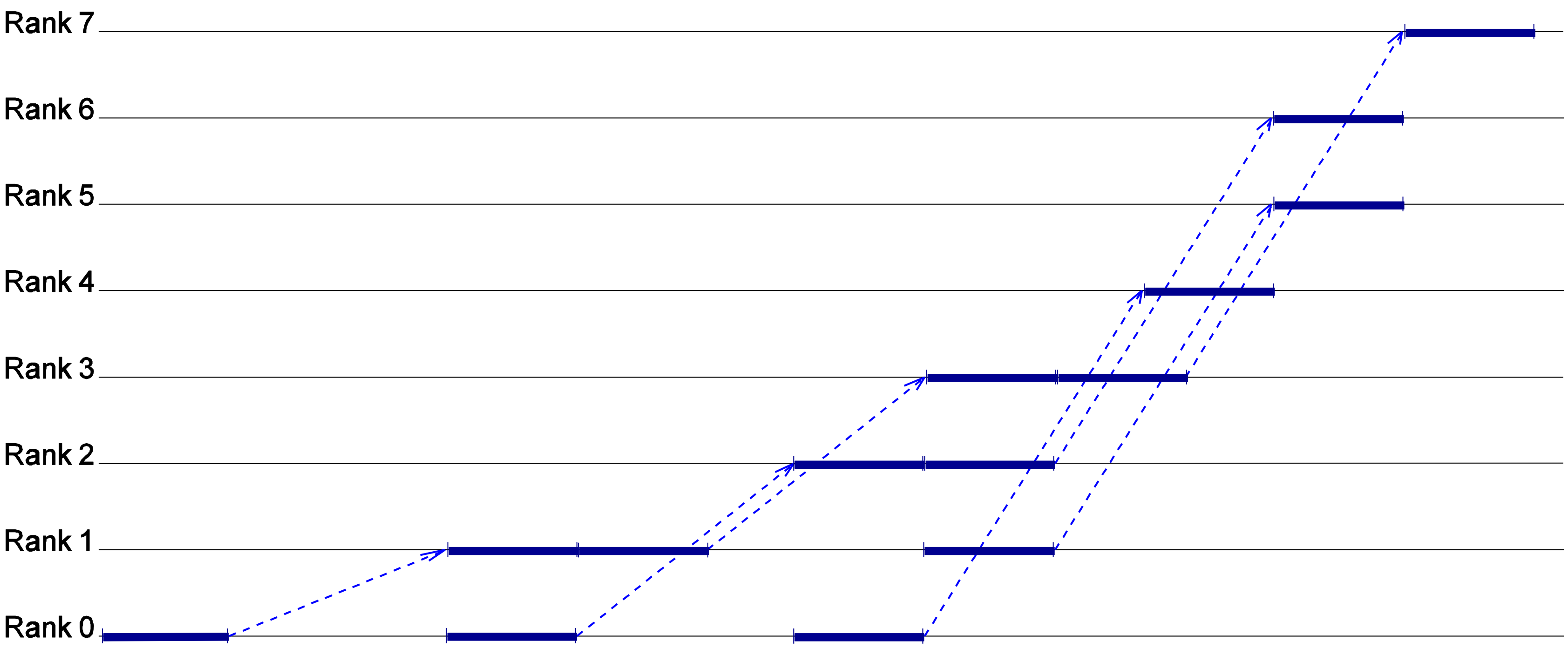
Verification - Gather

$$T_{gat} = 2o + L + \max\{(P-2)o + (P-1)sO), (P-2)g + (P-1)sG\}$$



Verification – Binomial Tree

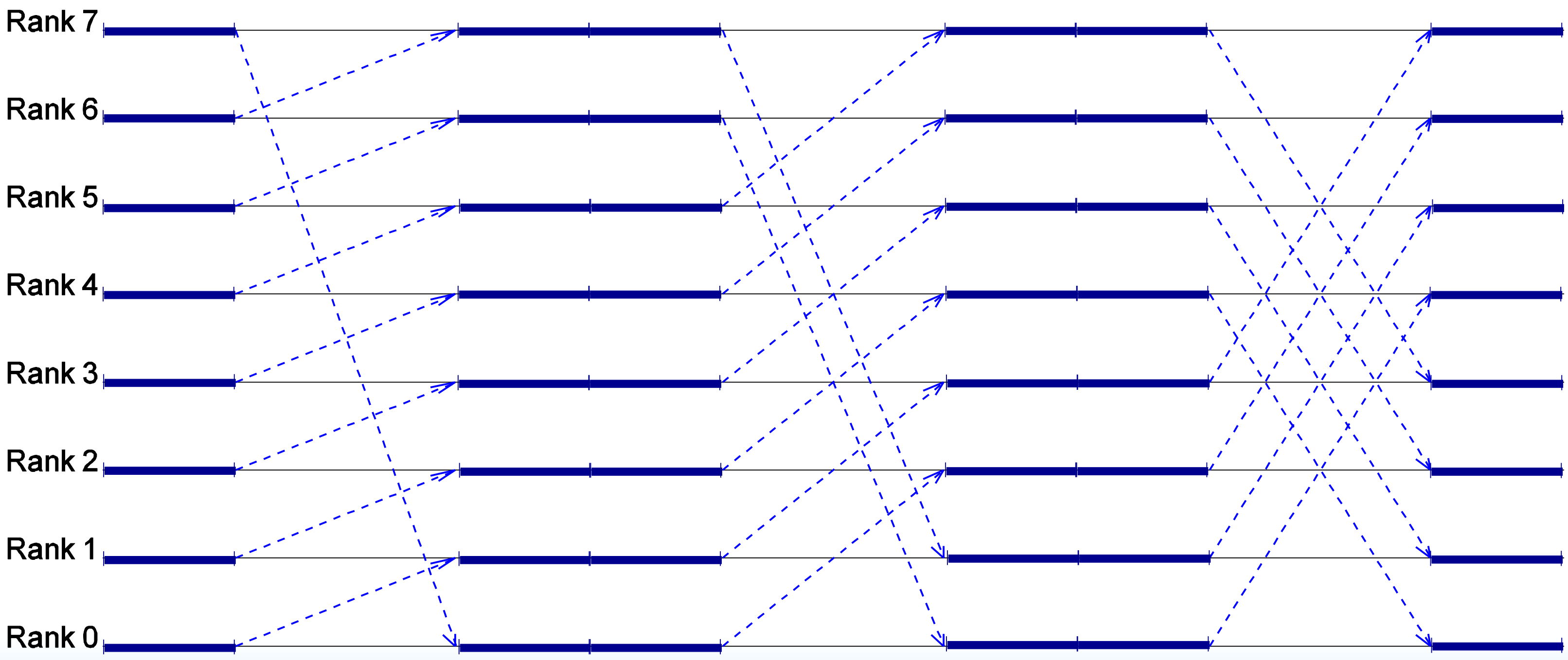
$$T_{bino} = (2o + L + \max\{sO, sG\})\lceil \log_2 P \rceil$$



Verification - Dissemination

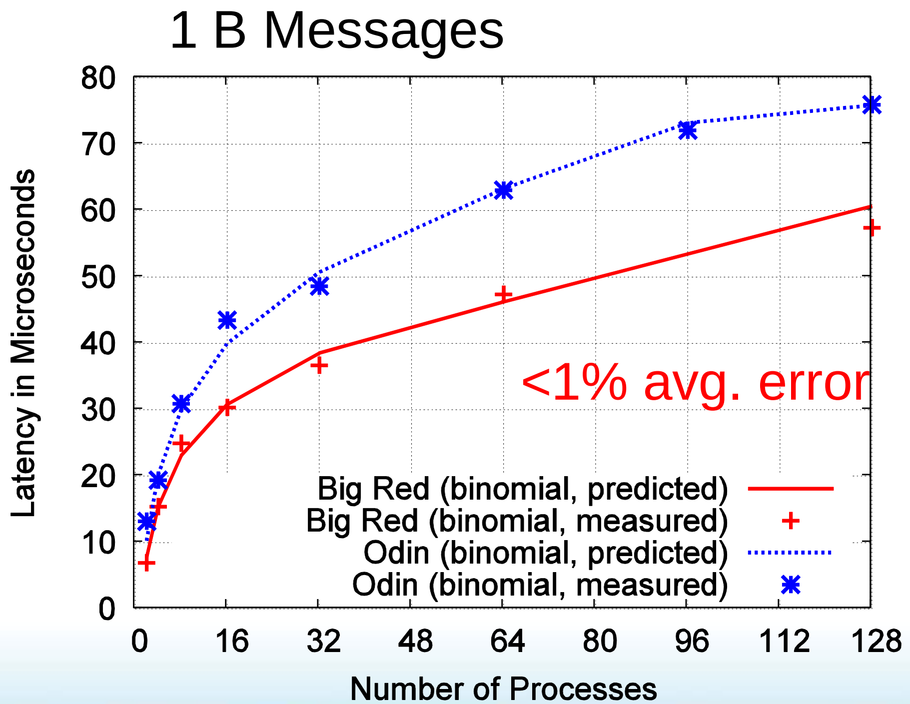
$$\delta = \begin{cases} (s-1)O - L : (s-1)O - L > 0 \\ 0 : \text{otherwise.} \end{cases} \quad (1)$$

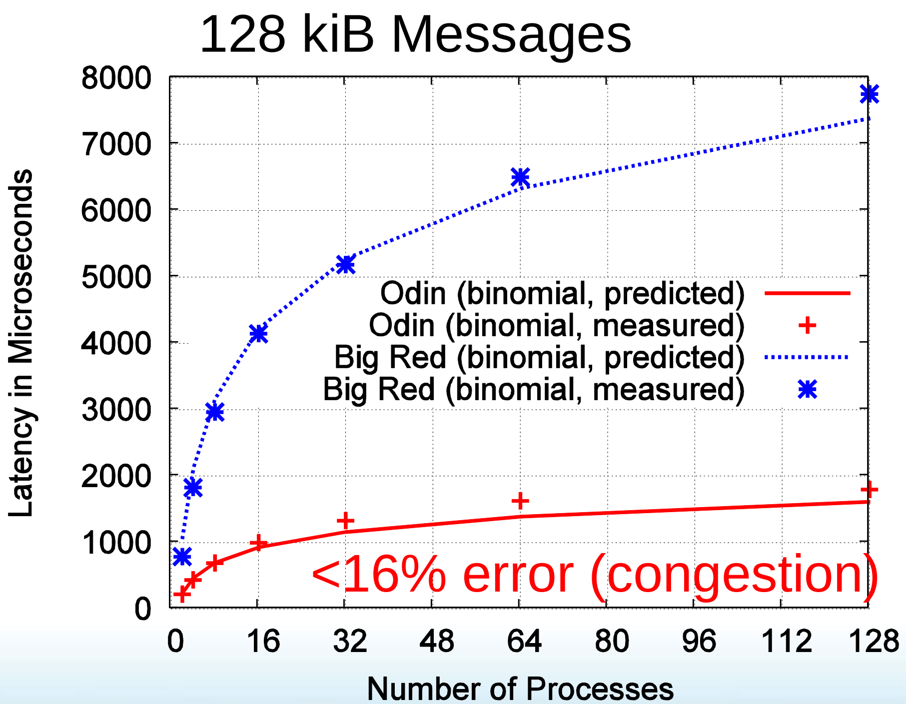
$$T_{diss} = (\delta + 2o + L + \max\{sO, sG\})\lceil \log_2 P \rceil \quad (2)$$



Experimental Evaluation

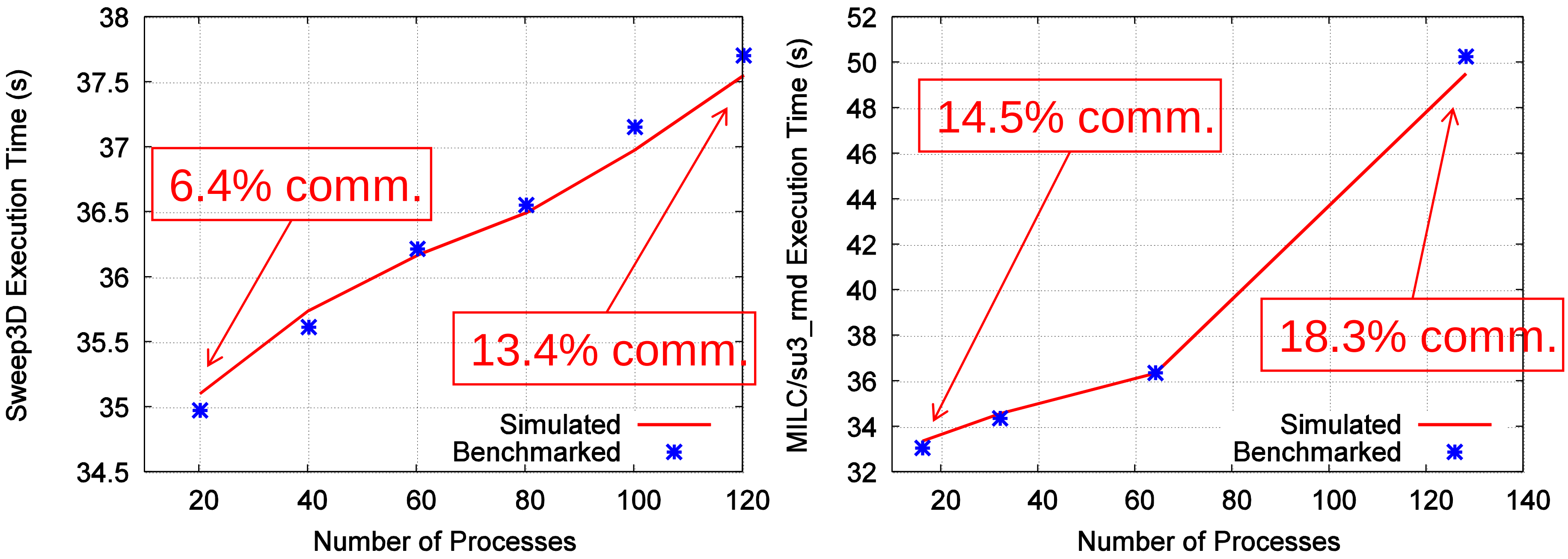
- Odin: $L=5.3\mu s$, $o=2.3\mu s$, $g=2\mu s$, $G=2.5ns$, $O=1ns$
- Big Red: $L=2.9\mu s$, $o=2.4\mu s$, $g=1.7\mu s$, $G=5ns$, $O=2ns$





Application Simulation Accuracy

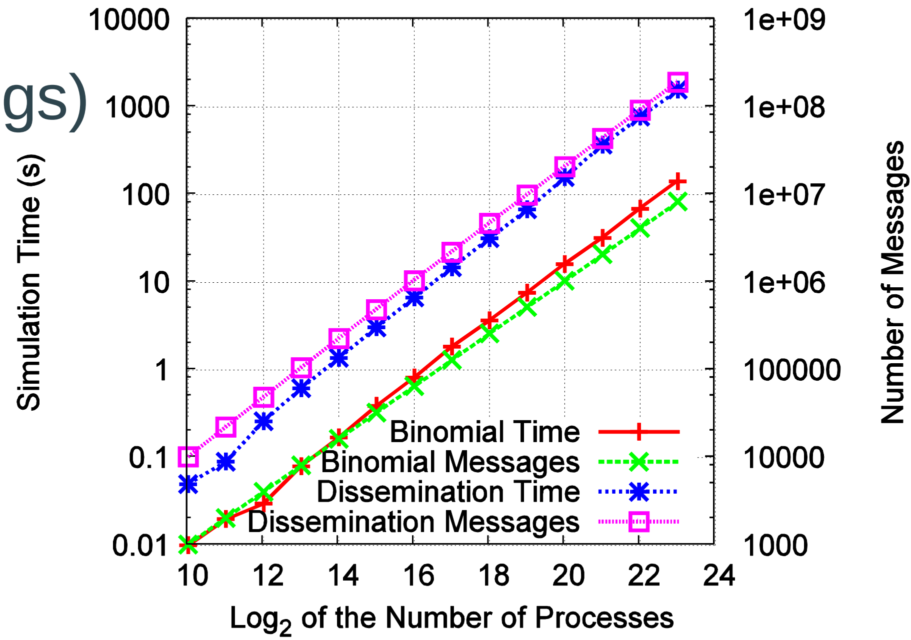
- Sweep3D and MILC weak scaling on Odin



- <2% average error

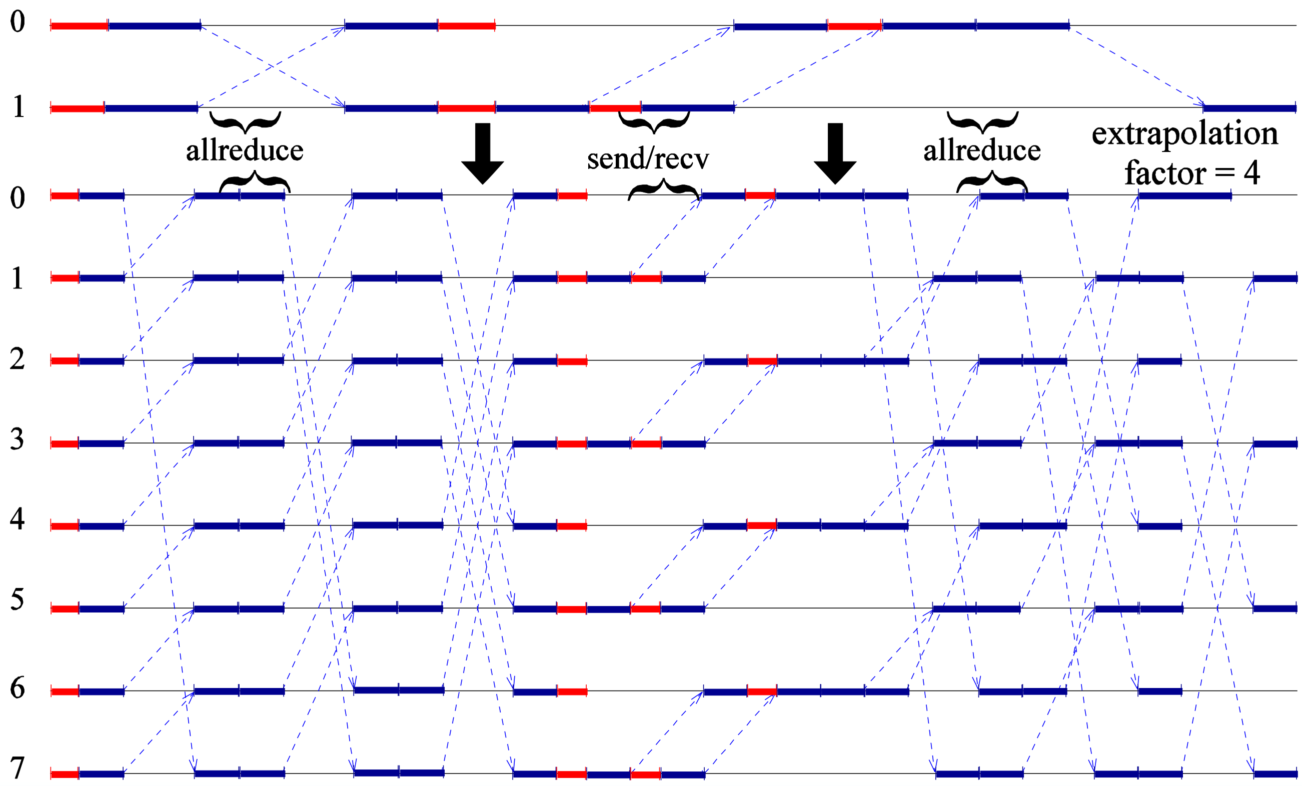
Simulation Speed

- Tested on 1.15 GHz Opteron (slow!)
  - 1024 – 8 million processes
  - Binomial (P msgs)
  - Dissemination ($P\log(P)$ msgs)
- > 1 million events per second
  - Can demo it on my laptop later ☺



Application Trace Extrapolation

- Supports simple extrapolation scheme:



Application Simulation Performance

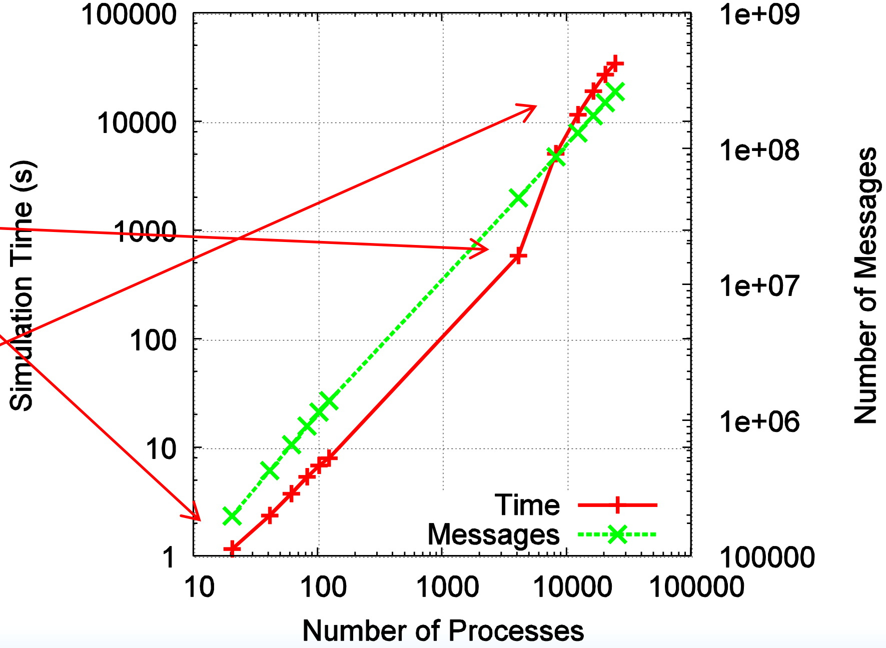
- 37.7 s Sweep3D extrapolated from 40-28k CPUs
  - 0.4 Mio msgs → 313 Mio msgs

40 CPUs – 2.43 s

4k CPUs – 10 min

28k CPUs – 9.7h (swap)

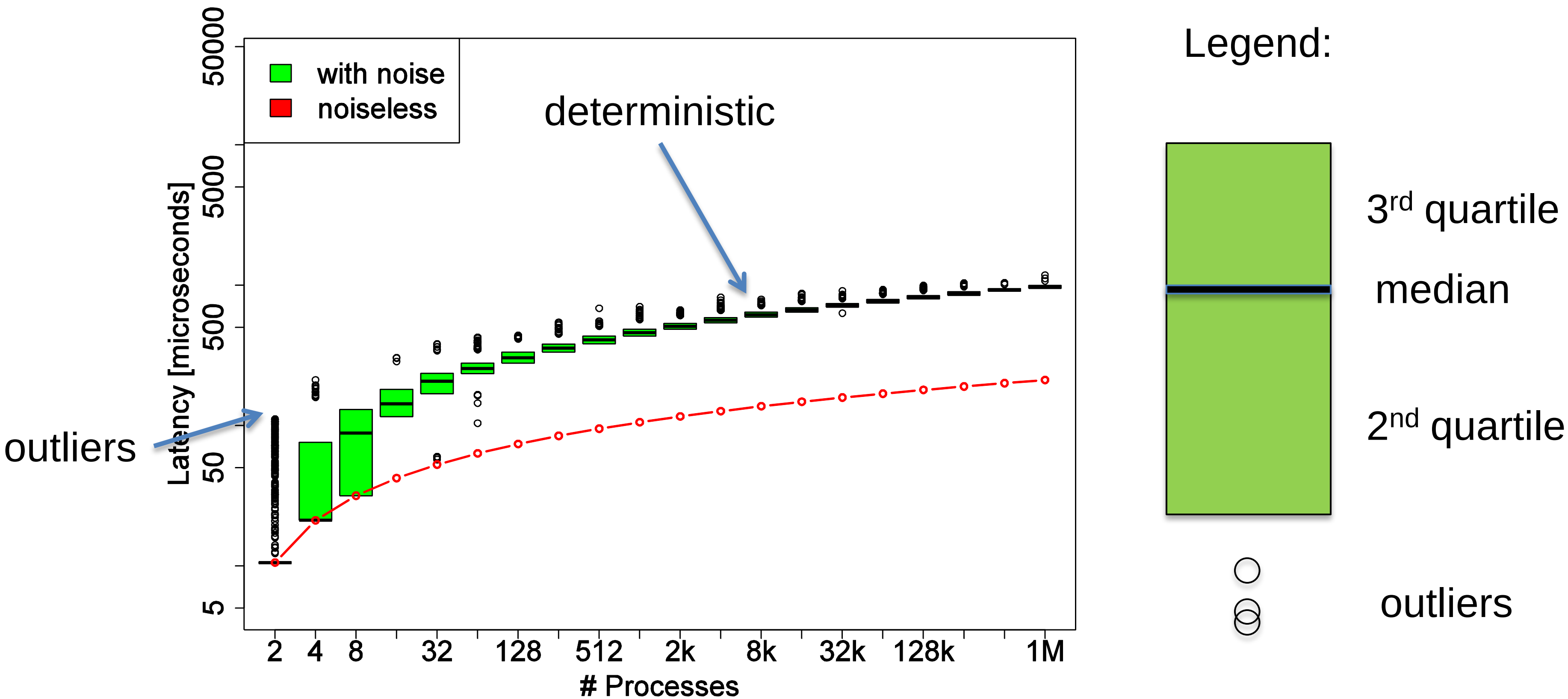
Main memory is an issue! hits swap at 8k CPUs



Some More Use-Cases

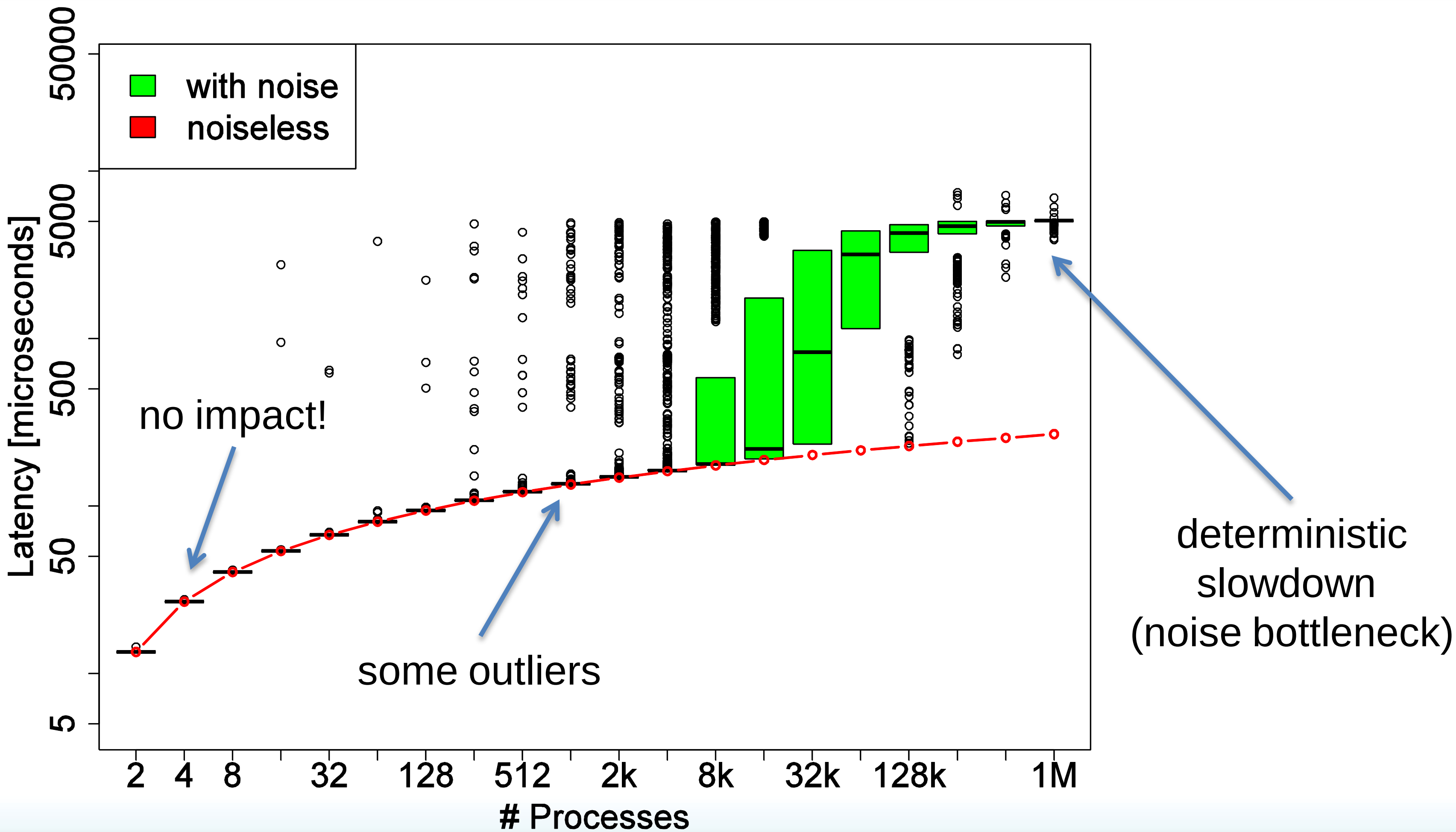
1. Estimating an application's potential for overlapping communication/computation
2. Estimating the effect of a faster/slower network on application performance
3. Demonstrating the effects of pipelining in current benchmarks for collectives
4. **Estimating the effect of Operating System Noise at very large scale**

Single Collective Operations and Noise

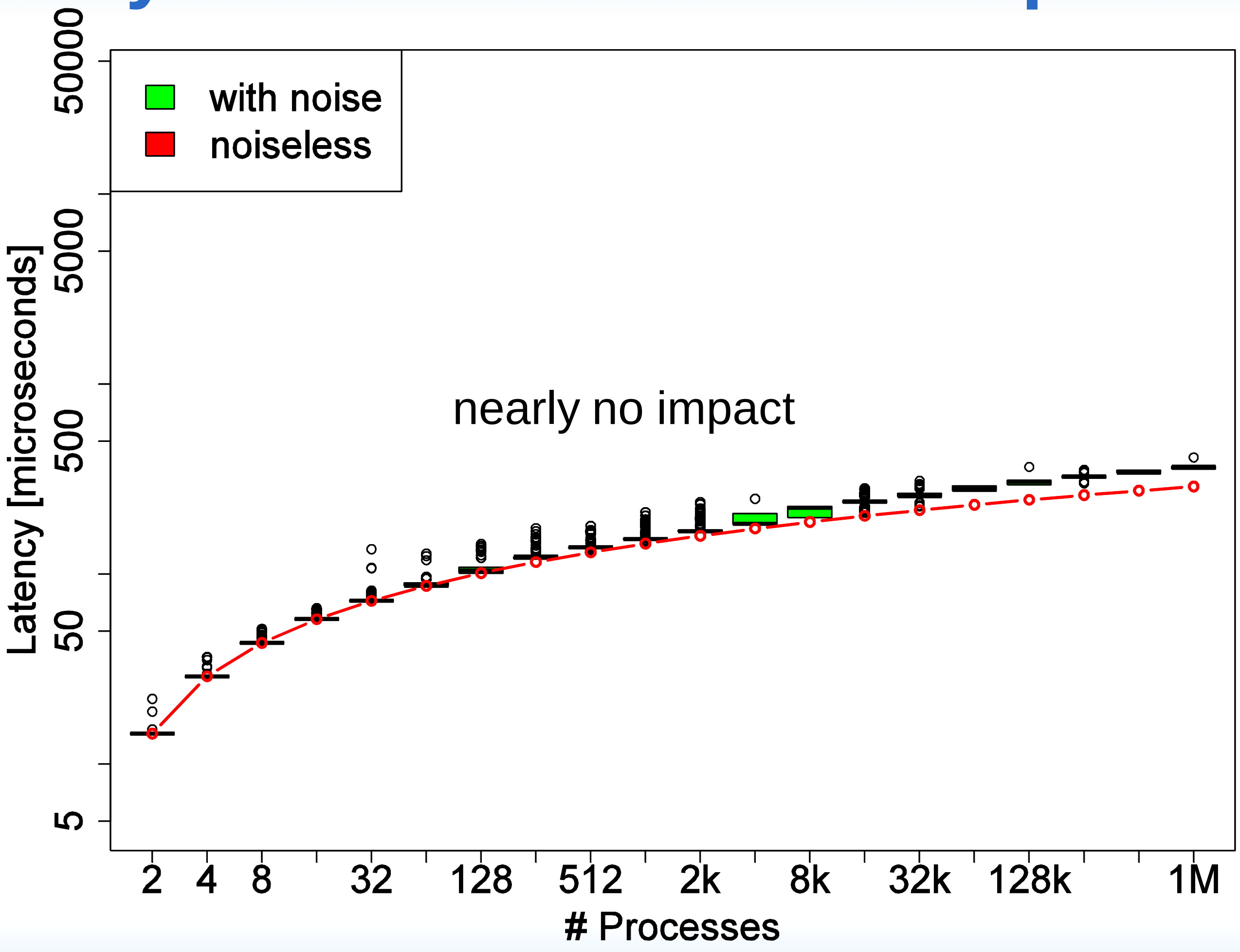


- 1 Byte, Dissemination, regular noise, 1000 Hz, 100 μs

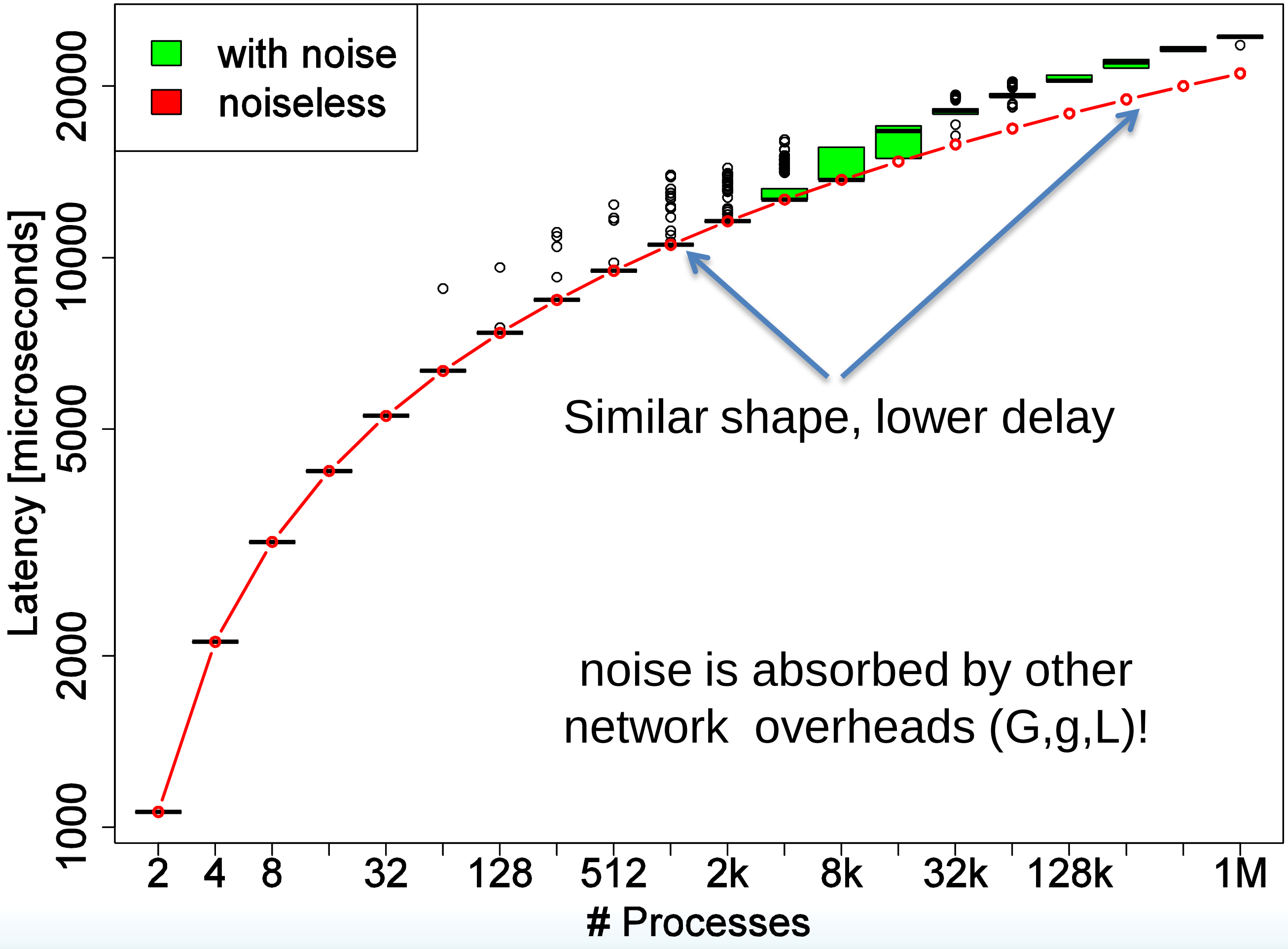
Single Byte Dissemination on Jaguar



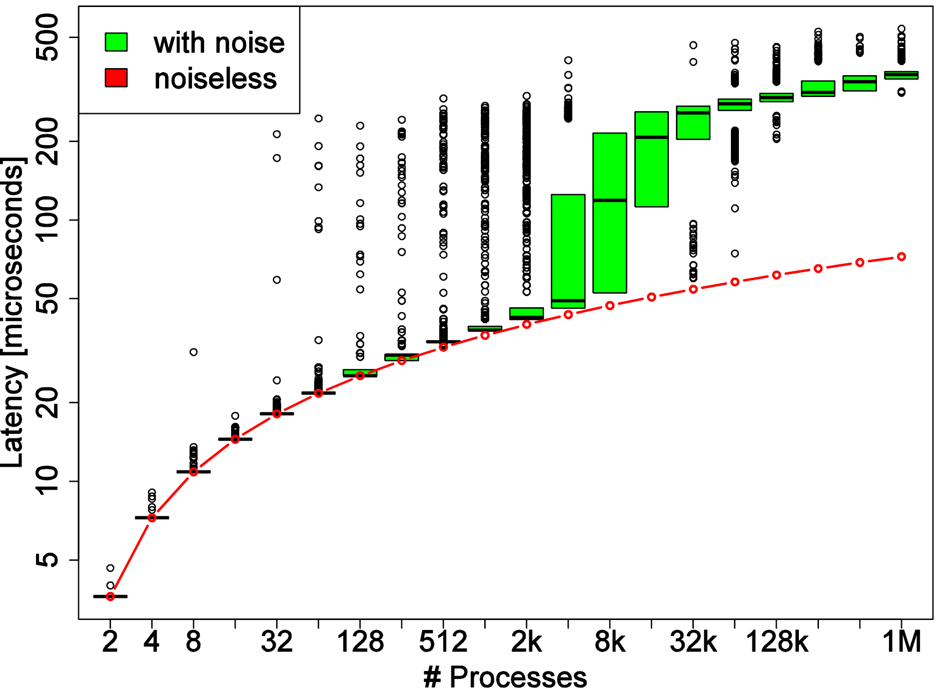
Single Byte Dissemination on ZeptoOS



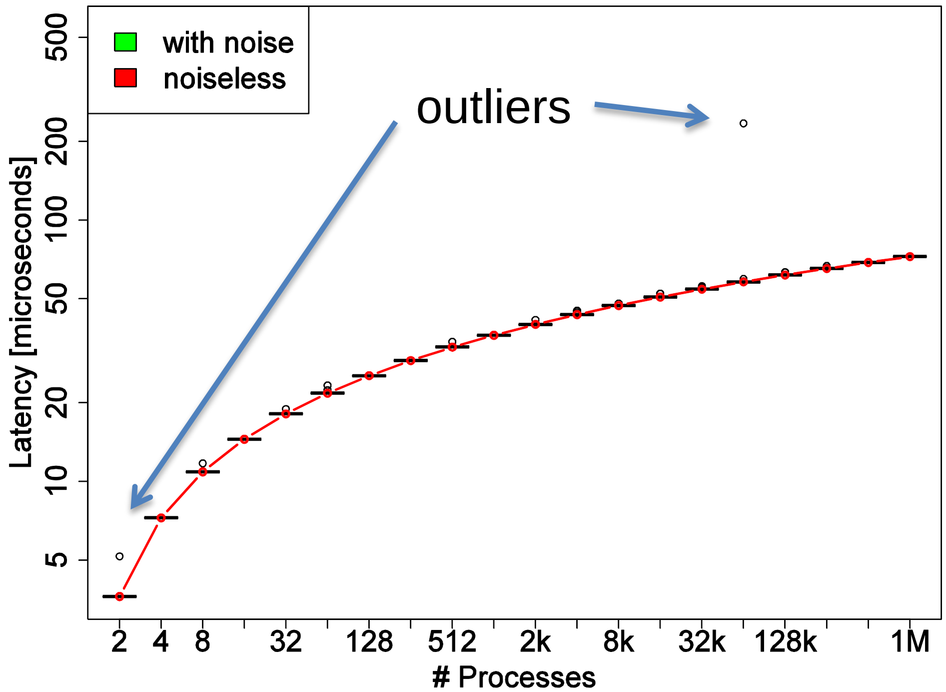
1MiB Messages on Jaguar



Effect of Co-Scheduling Noise (Altix)

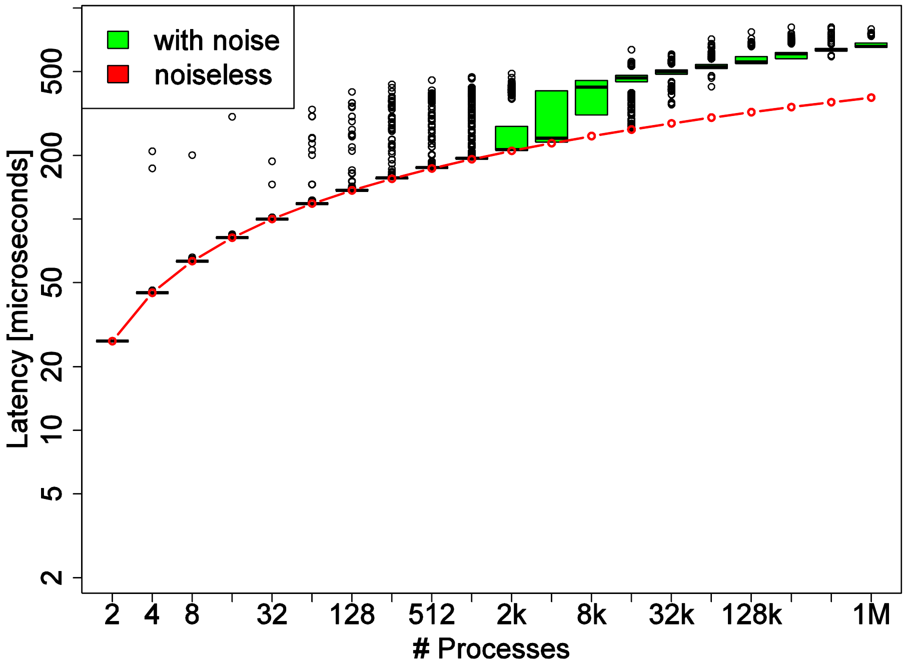


Normal

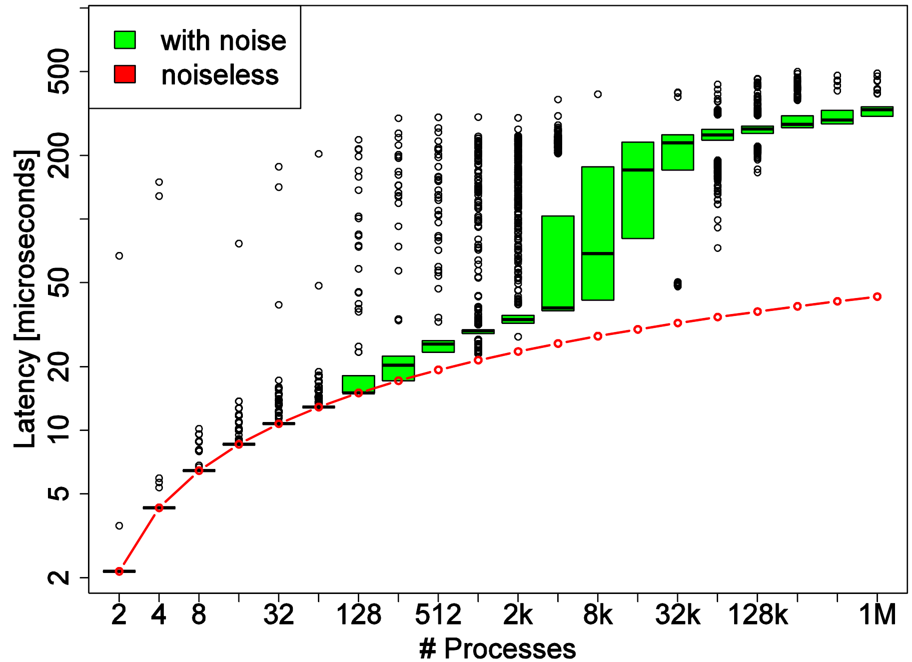


Co-Scheduled

Does the Network Speed Matter?



0.1x network speed

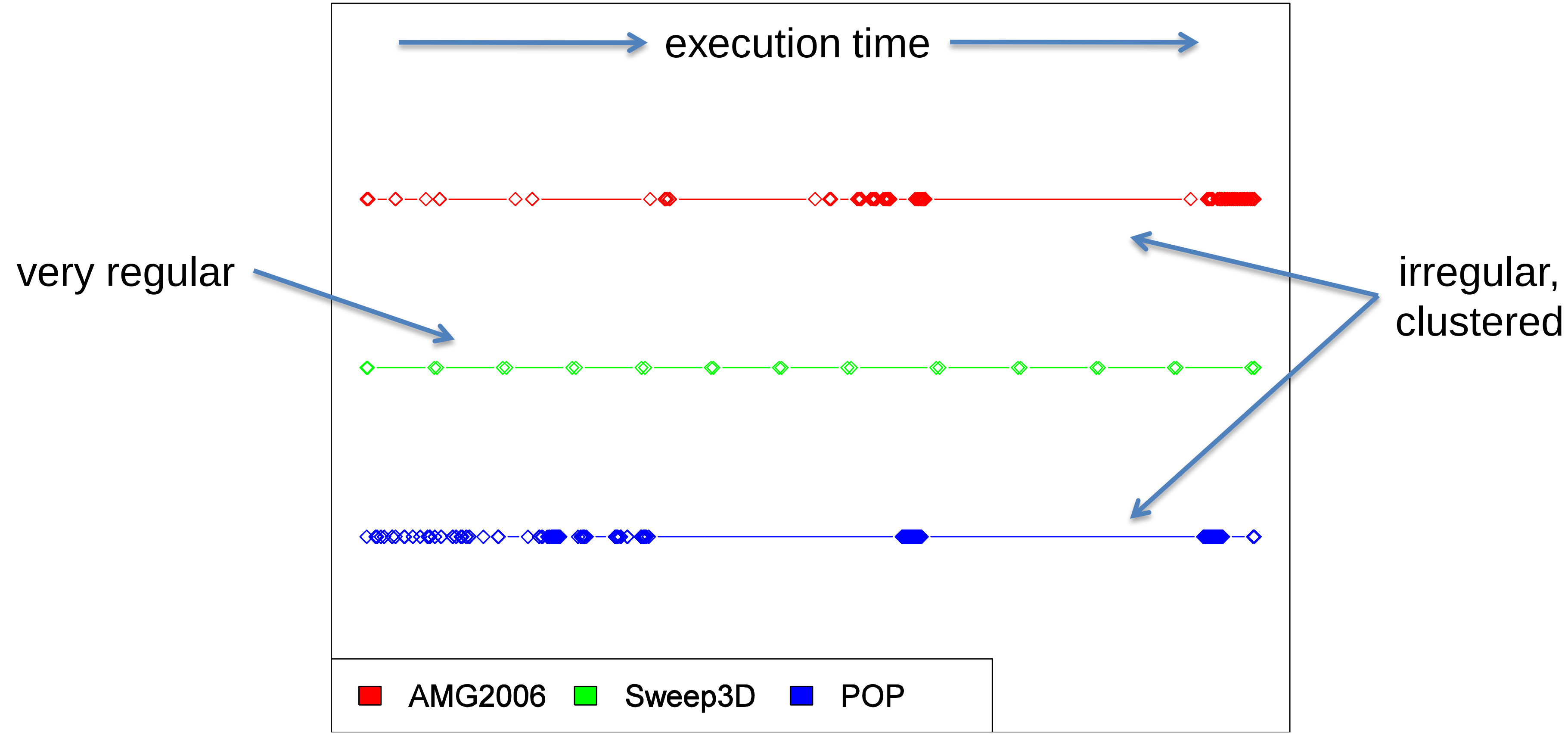


10x network speed

Method: increase/decrease L,G,g

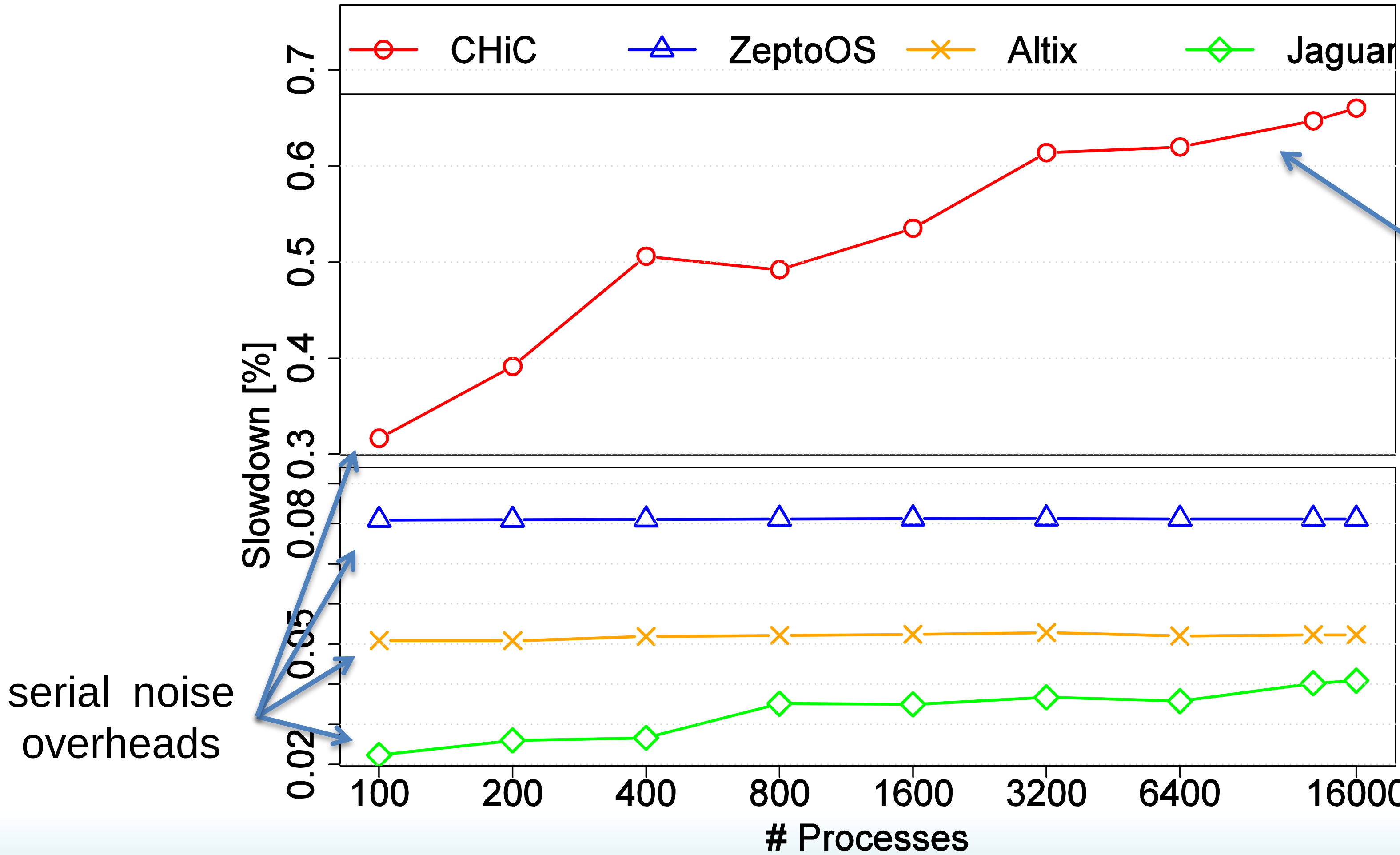
Observation: noise bottleneck independent of network speed

Real Applications



Distribution of Collective Operations

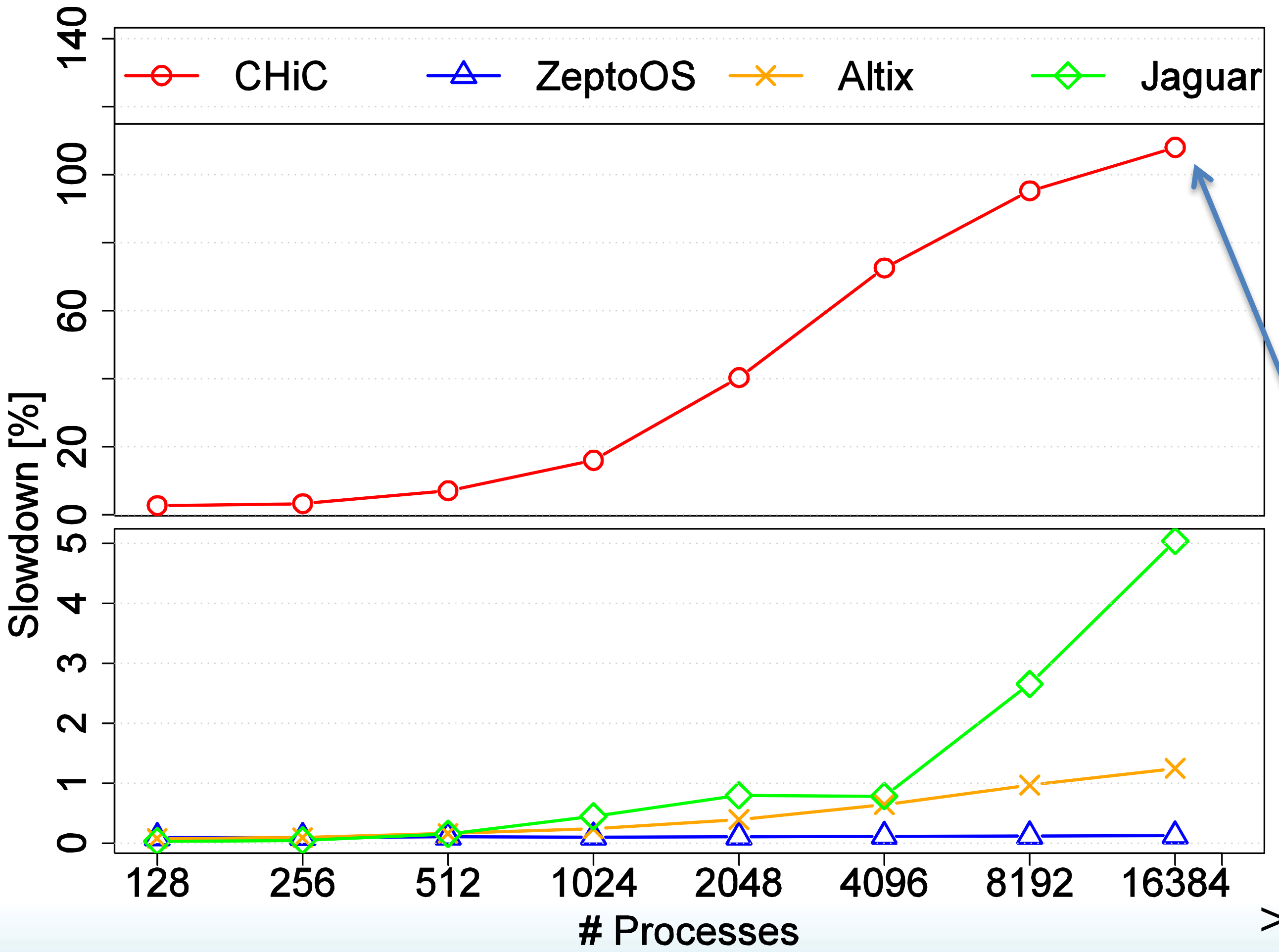
Sweep3D (Collective and Point-to-Point)



92.4% bl p2p
4 bcast
3 barrier
32 allreduce

>289 million messages!

POP (Collective and Point-to-Point)



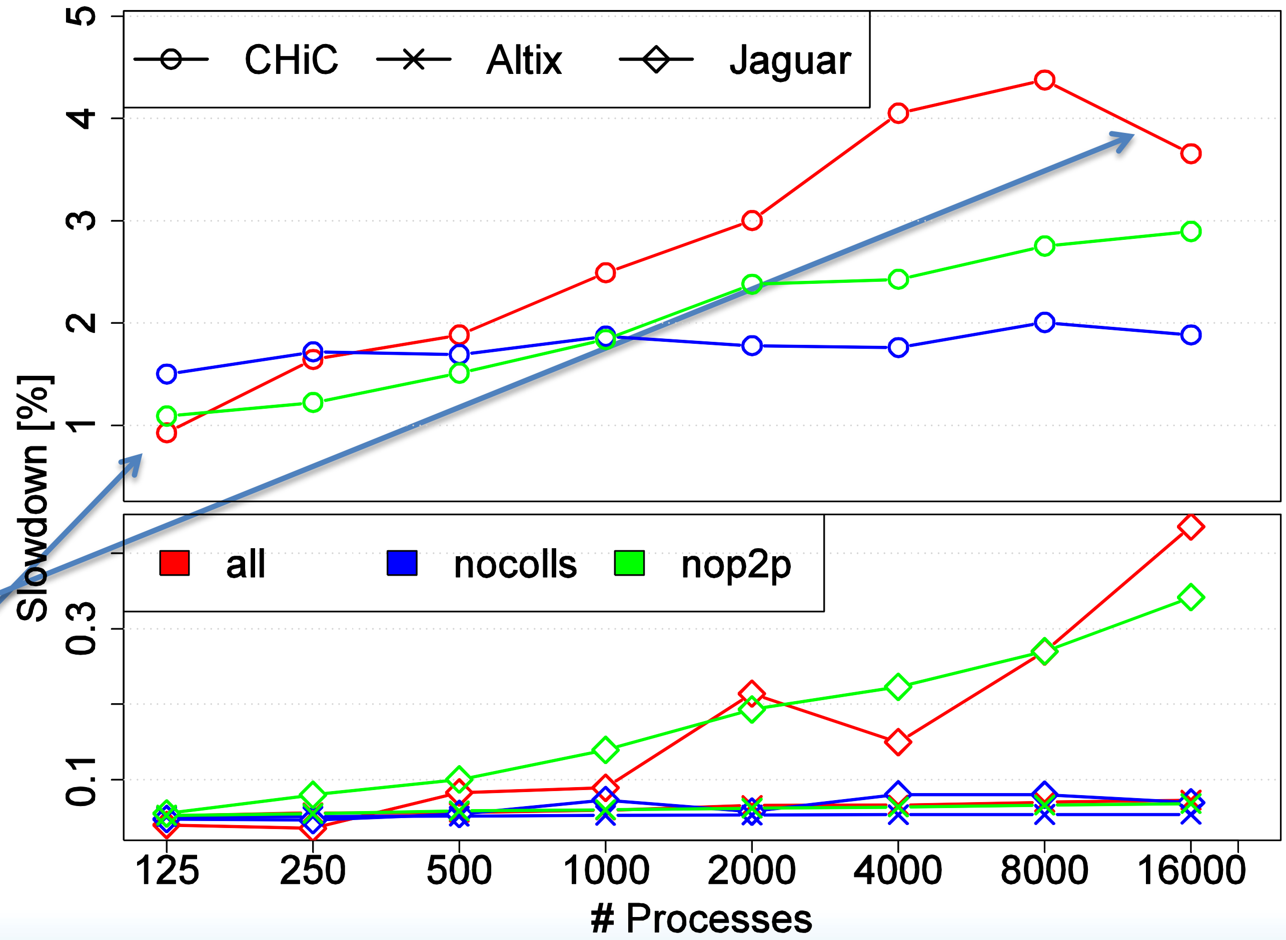
0.2% nb p2p
703 bcast
575 barrier
608 allreduce

>2x slowdown!

>625 million messages!

Does Point-to-Point Communication Matter?

AMG 2006

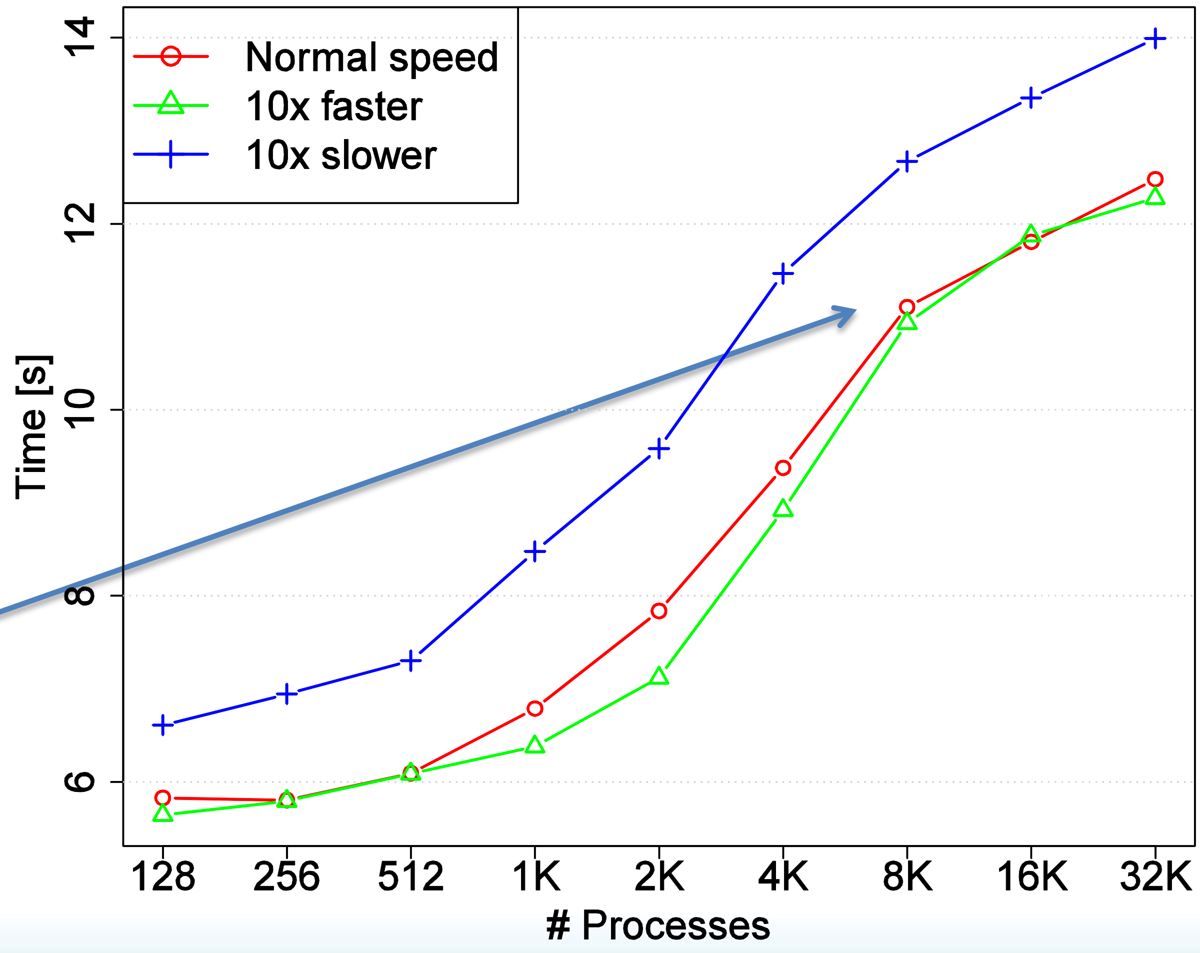


p2p propagates

p2p absorbs

Influence of Network Speed on Applications

POP @ CHiC



Noise bottleneck: faster network does not increase performance

Conclusions & Future Work

- Modeling OS noise is not that simple
  - Will validate used models with simulation
- Model-based simulation approach scales well
  - Results match previous benchmark studies (<6% error)
- Overhead depends on noise *shape* rather than *intensity*
  - ZeptoOS shows nearly no propagation! (0.08% overhead)
  - Cray XT is severely impacted! (0.02% overhead)
- Noise bottleneck is serious at scale!
  - Faster network or CPU cannot help, noise will dominate!
- We developed a tool-chain to adjust the bottleneck
  - Available online: `http://www.unixer.de/LogGOPSim`

Collaborators, Acknowledgments & Support

- Collaborators:
  - Timo Schneider, Andrew Lumsdaine — INDIANA UNIVERSITY PERVASIVE TECHNOLOGY INSTITUTE
- Thanks to (alphabetically)
  - Franck Cappello, Steven Gottlieb, William Gropp, William Kramer, and Marc Snir
- Sponsored by

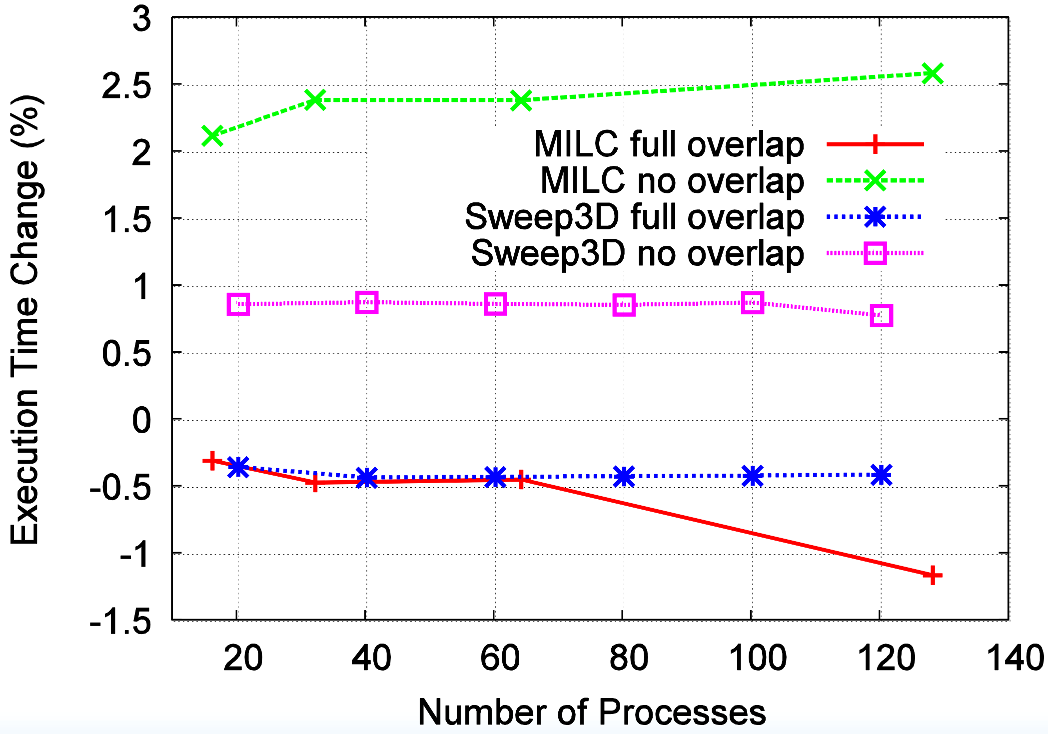
Thanks and try it at Home!

- LogGOPSim (the simulation framework)
  http://www.unixer.de/LogGOPSim
- Netgauge (measure LogGP parameters + OS Noise)
  http://www.unixer.de/Netgauge
- References:
  - Hoefler et al.: “Characterizing the Influence of System Noise on Large-Scale Applications by Simulation” (Best Paper at SC10)
  - Hoefler et al.: “LogGOPSim - Simulating Large-Scale Applications in the LogGOPS Model” (Best Paper at LSAP’10)

Backup

Application Overlap Potential

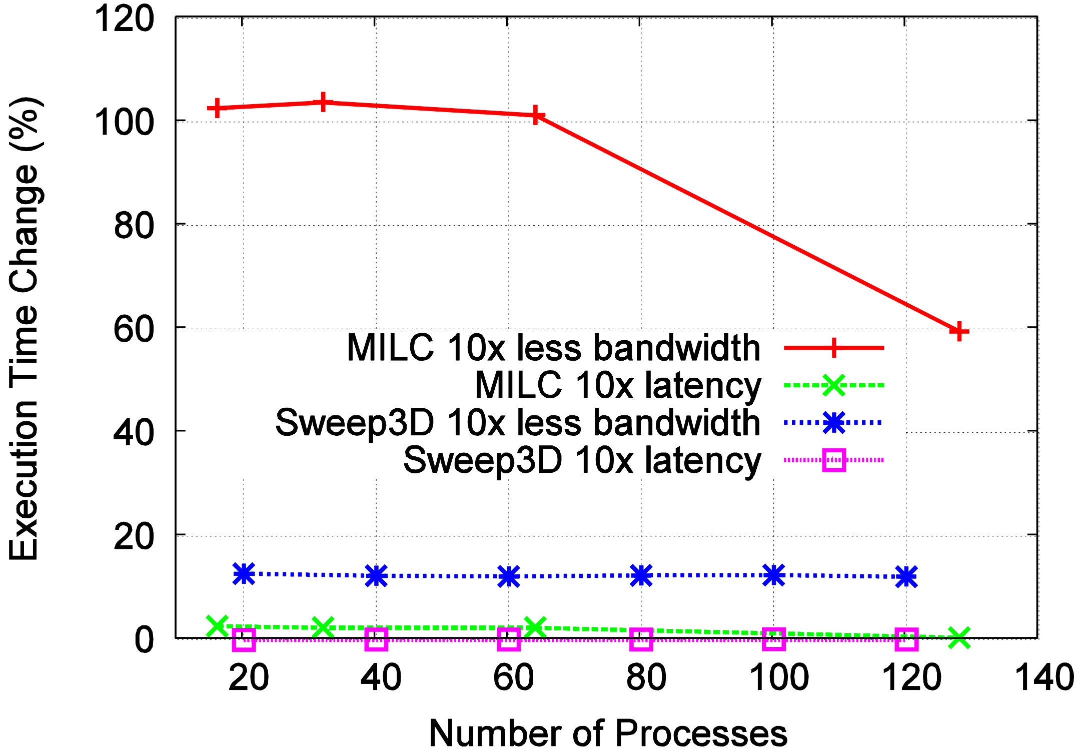
- Choose overhead appropriately:
  - full overlap:
    - o=0
    - O=0
  - no overlap:
    - o=g
    - O=G



Influence of Network Parameters

- Adjust L (latency) and G (bandwidth)

Both are much more sensitive to bandwidth than to latency!



Explaining Benchmark Problems

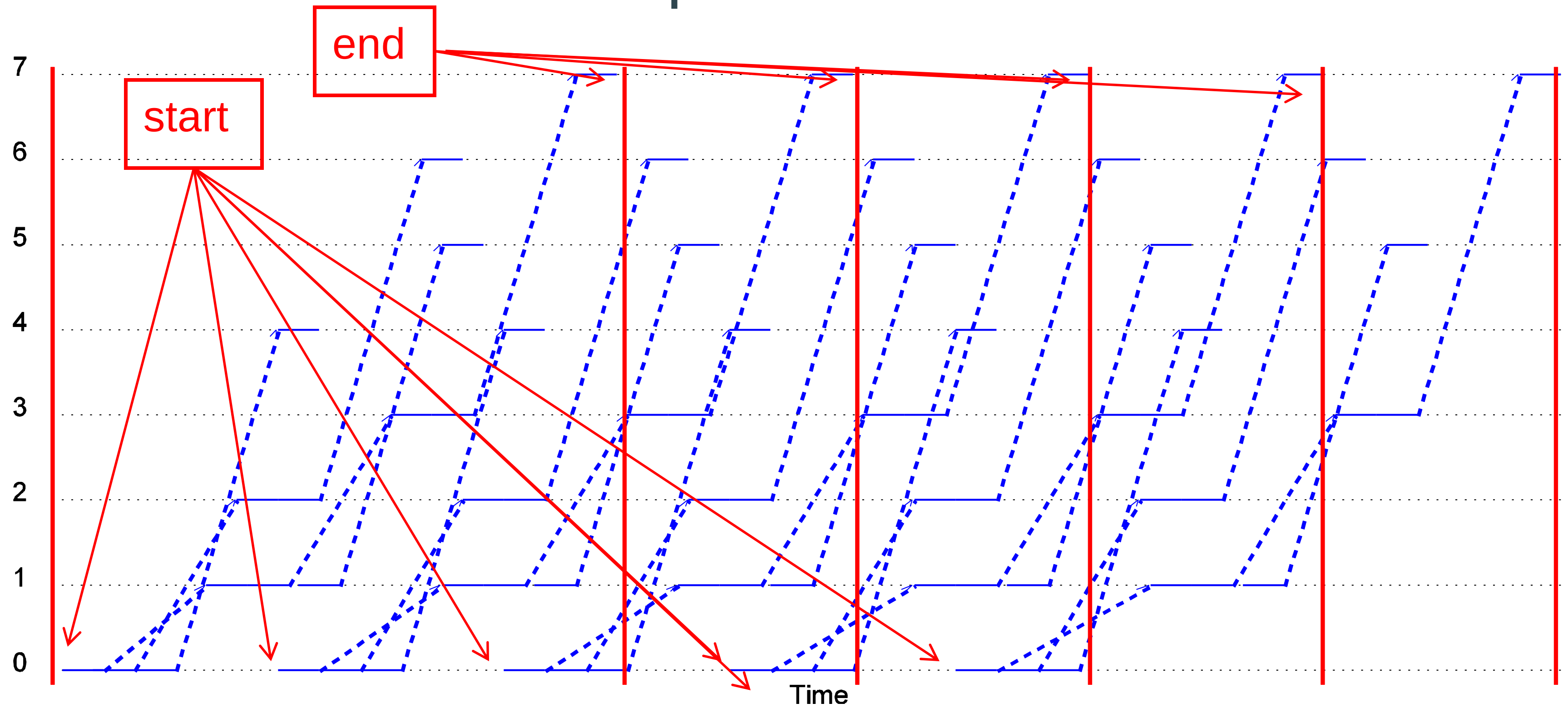
- Collective operations are often benchmarked in loops:

```
start= time();
for(int i=0; i<samples; ++i) MPI_Bcast(…);
end=time();
return (end-start)/samples
```

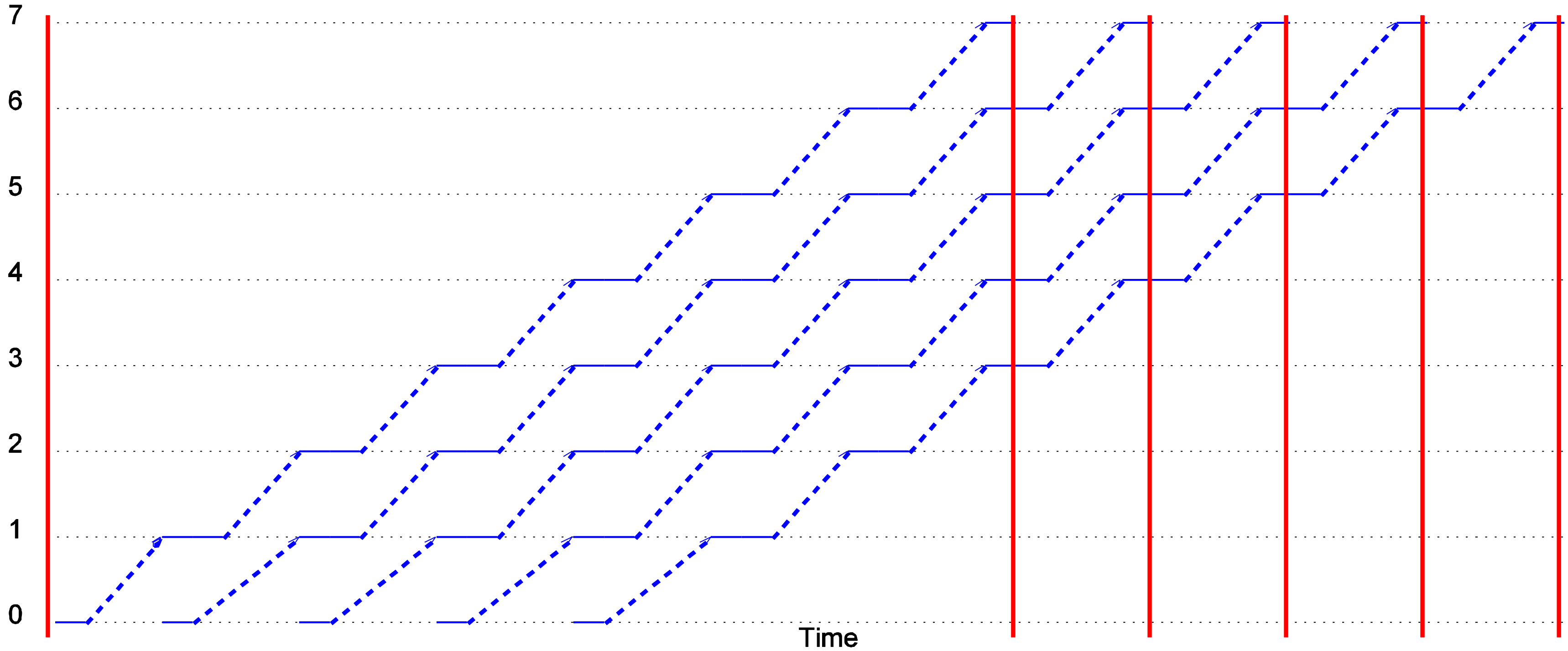
- This leads to pipelining and thus wrong benchmark results!

Pipelining? What?

Binomial tree with 8 processes and 5 bcasts:



Linear broadcast algorithm!



This bcast must be really fast, our benchmark says so!

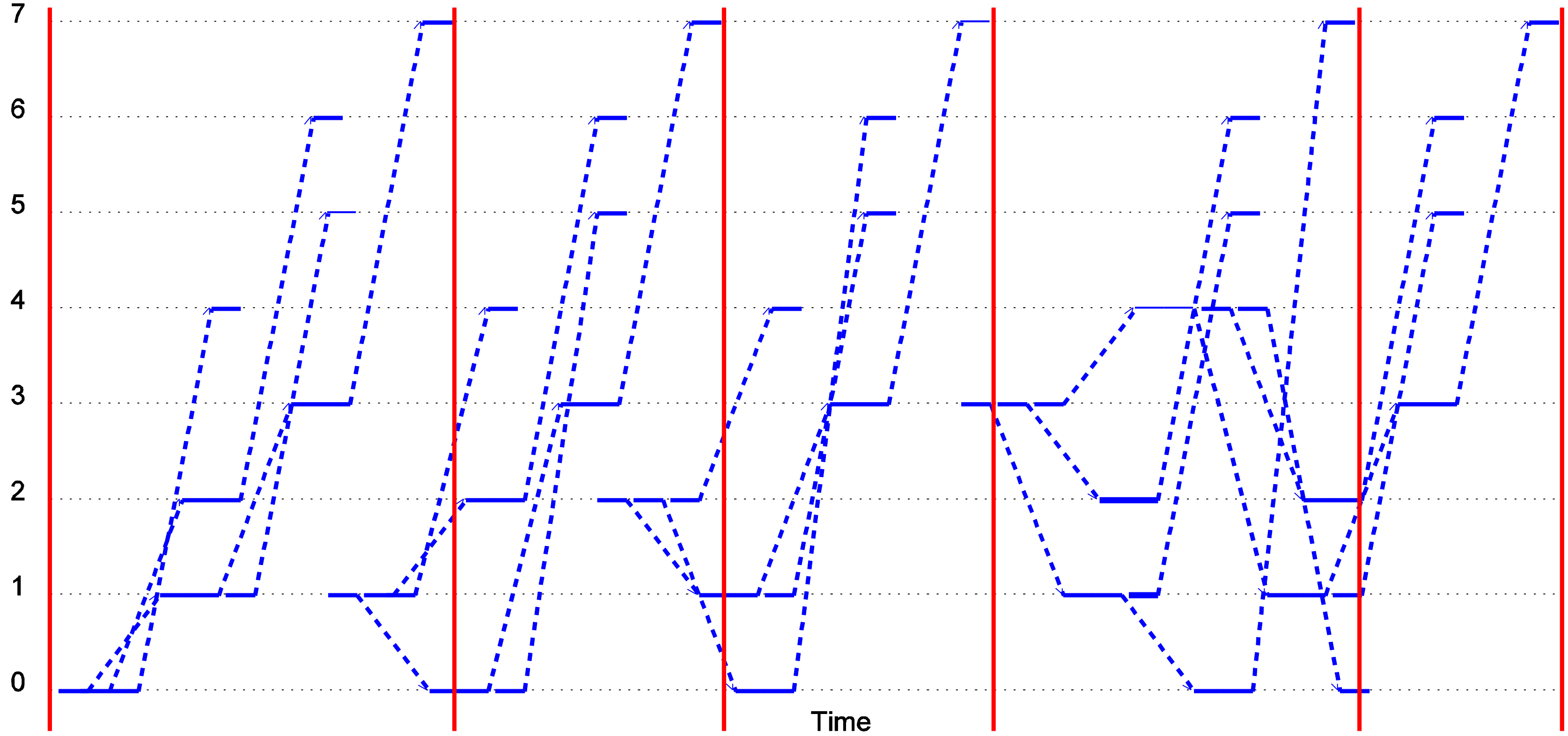
Root-rotation! The solution!

- Do the following (e.g., IMB)

```
start= time();
for(int i=0; i<samples; ++i)
  MPI_Bcast(…,root= i % np, …);
end=time();
return (end-start)/samples
```

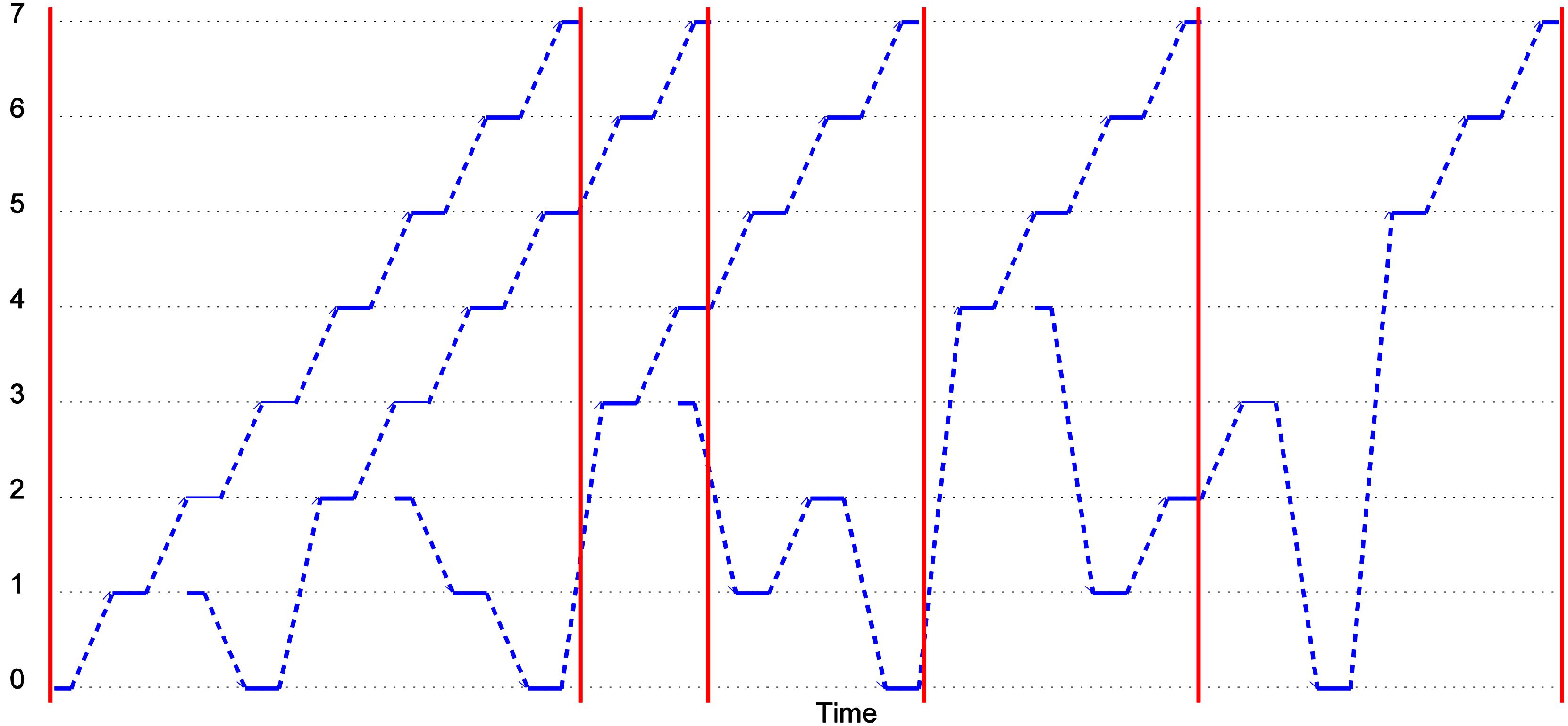
- Let’s simulate …

D’oh!



- But the linear bcast will work for sure!

Well … not so much.



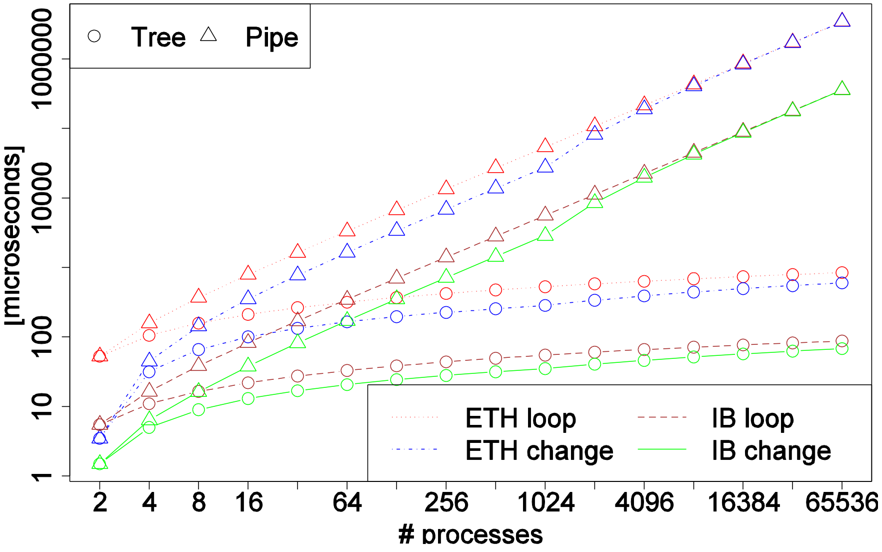
But how bad is it really? Simulation can show it!

Absolute Pipelining Error

- Error grows with the number of processes!
- Details in:

Hoefler et al.: “*LogGP in Theory and Practice*”

In: Journal of Simulation Modelling Practice and Theory (SIMPAT). Vol 17, Nr. 9

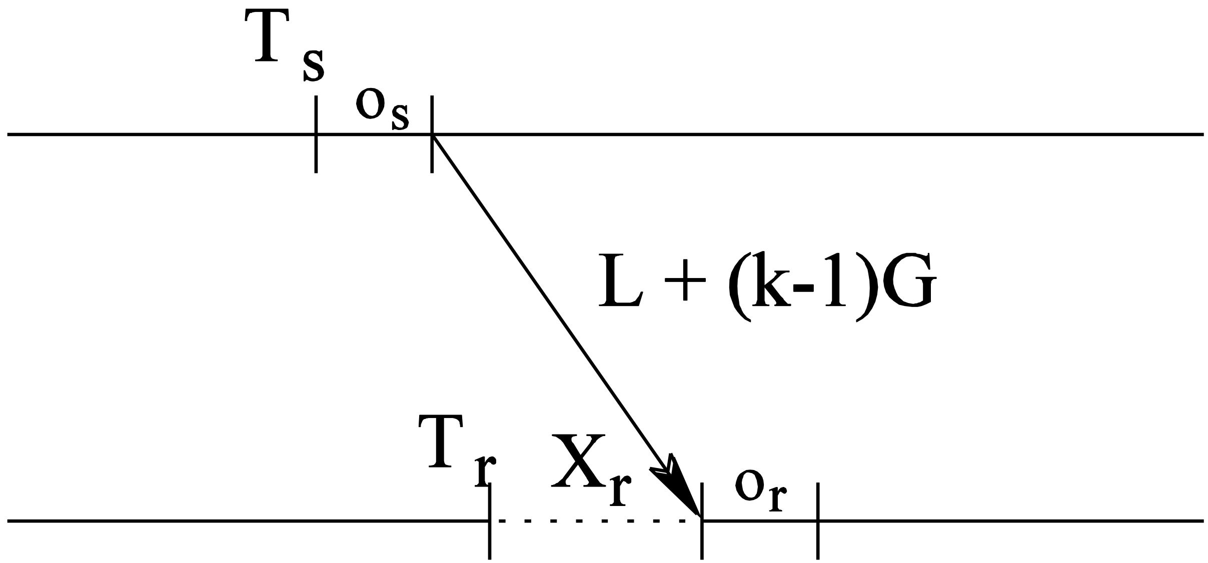


Comparison of Simulations and Experiments

- Beckman et al. - Allreduce on BG/L
  - 1000, 100, 10 Hz detours of 16, 50, 100, 200 μs
  - Reproduced linear scaling with noise
  - 16x slowdown in 32k processes, simulation: 13x
    - We used (better) LogGOPS parameters from BG/P

- Ferreira et al. Allreduce on Cray XT
  - 10 Hz, 2.5 μs detours
  - Experiment: 32x slowdown, simulation: 30x

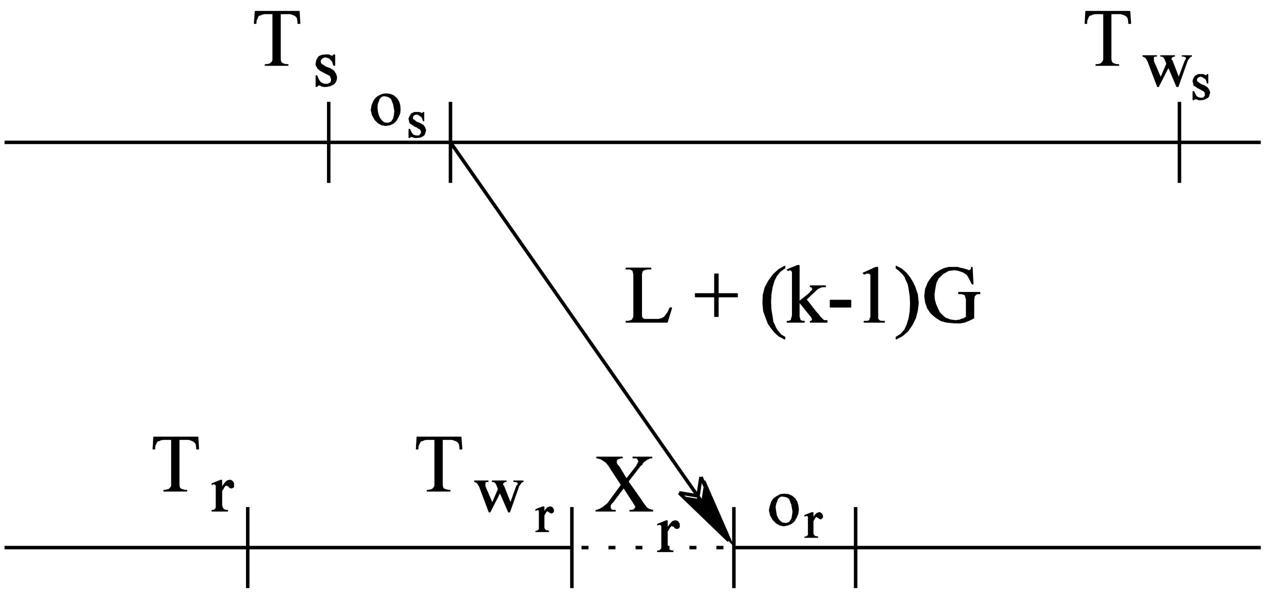
Blocking Point-to-Point Communication



- Synchronization overhead X_r can absorb noise at receiver
  - Sender noise can be absorbed if receive is posted late
- X_r and X_s (rendezvous) can be amplified by noise
- Synchronization overheads can only be avoided if

$$T_r = T_s + o_s + L + (k-1)G + o_r$$

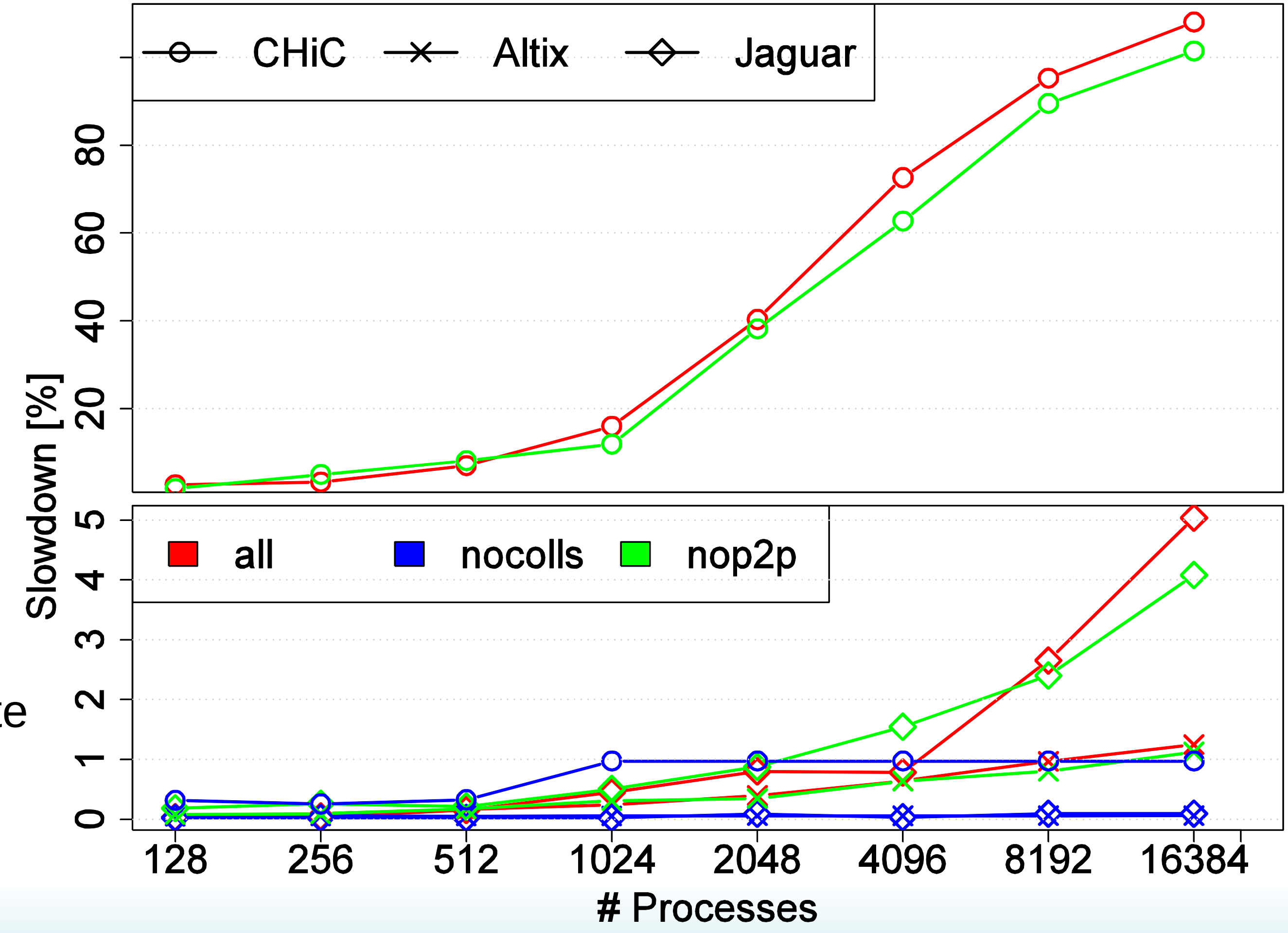
Nonblocking Point-to-Point Communication



- Time between Wait and Send/Recv Init acts as “buffer”
- Can absorb OS noise and synchronization overheads
  - Exact details are discussed in the full paper

Does Point-to-Point Communication Matter?

POP



collectives dominate noise overhead

AMG 2006 (Collective and Point-to-Point)