BLUE WATERS
SUSTAINED PETASCALE COMPUTING

On System Noise and Large-Scale Applications

Torsten Hoefler

With contributions fromTimo Schneider and Andrew Lumsdaine

Scientific talk at TU Dresden, May 26th 2011

GREAT LAKES CONSORTIUM
FOR PETASCALE COMPUTATION

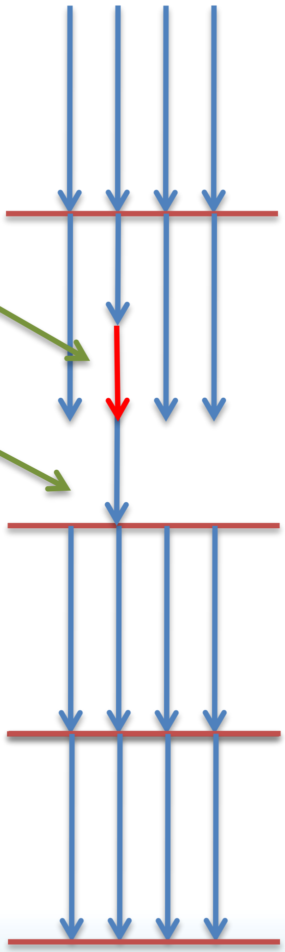
Vision: Performance-Centric Software Development

- Make performance a **first-class citizen**!
  - Equal to correctness, see type system
  - “Should be hard to write slow code”
- HPC-centric design (but not limited)
  - Must support distributed memory parallelism
  - Addresses static, runtime-static, dynamic applications
- This goal requires advances in:
  - Performance Modeling, Networks, Programming Model

Research Summary

System Noise – Introduction and History

- CPUs are time-shared
  - Daemons, interrupts, etc. steal cycles
    - Hardware noise will become a problem
  - No problem for single-core performance
    - Maximum seen: 0.26%, average: 0.05% overhead
  - "Resonance" at large scale (Petrini et al '03)
    - Slowdown of 2x on 8192 processes
- Numerous studies
  - Theoretical (Agarwal'05, Tsafrir'05, Seelam'10)
  - Injection (Beckman'06, Ferreira'08)



Why is Noise Interesting and Important?

- Practical large-scale computing
  - Noise is anticipated to be one of the biggest problems at large-scale (especially Blue Waters)
  - Performance losses up to 4x have been documented during production runs
- Academic interest
  - Fundamental interactions between local time-sharing and synchronization in parallel applications
  - Develop better understanding with analytical models

The main Observations

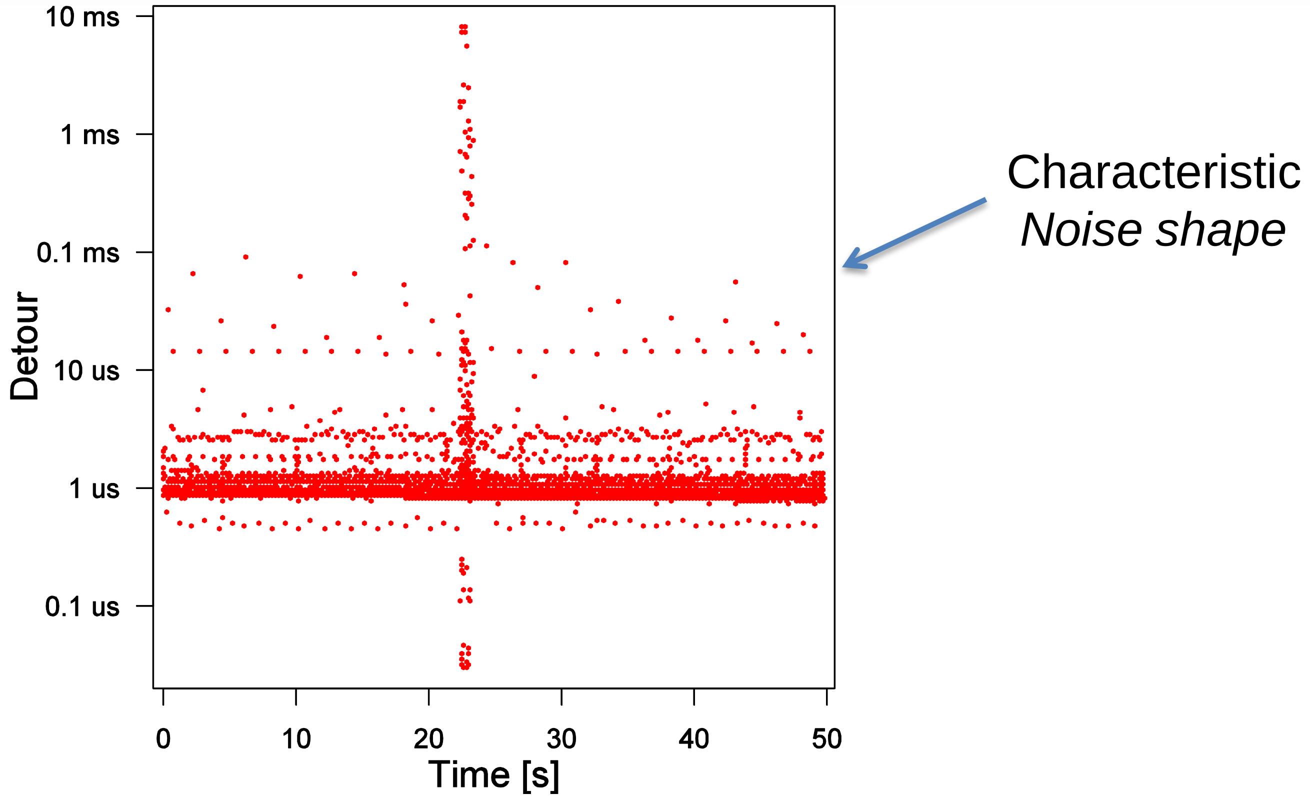
1. The effect of OS noise can be significant and at large scale (cf., Exascale)!
   - It needs to be considered as a main bottleneck!
   - Eliminating OS noise can be more important than speeding up the network
2. The slowdown is non-linear, it is very small until a certain threshold is reached
   - Using smaller systems (e.g., ½ of the size) for experiments may be misleading

Measuring Noise on a Single Core

- Selfish Detour Benchmark (Beckman et al.)
  - Tight execution loop, benchmark iteration time
  - Record each outlier in iteration time
  - Available at: http://www.unixer.de/Netgauge

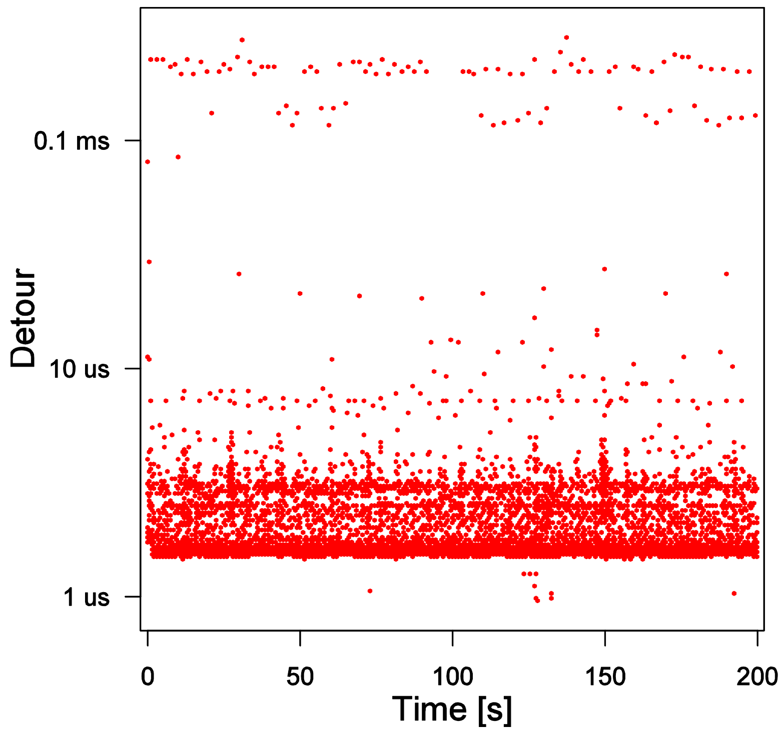
```
...
while(!abort) {
  tp = t;
  t = take_time();
  if(t-tp > 9*min_time) record_outlier();
}
```

Measurement Results – CHiC Linux (diskless)



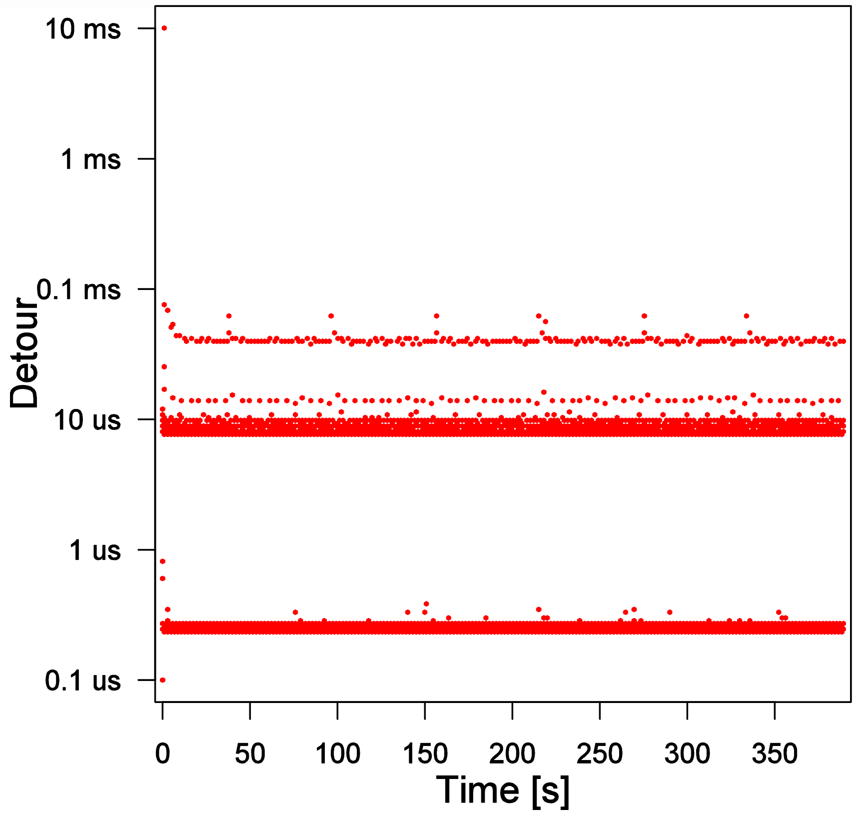
- 2152 Opteron cores, 11.2 Tflop/s Linux 2.6.18
- Resolution: 3.74 ns, noise overhead: 0.21%

Measurement Results – SGI Altix



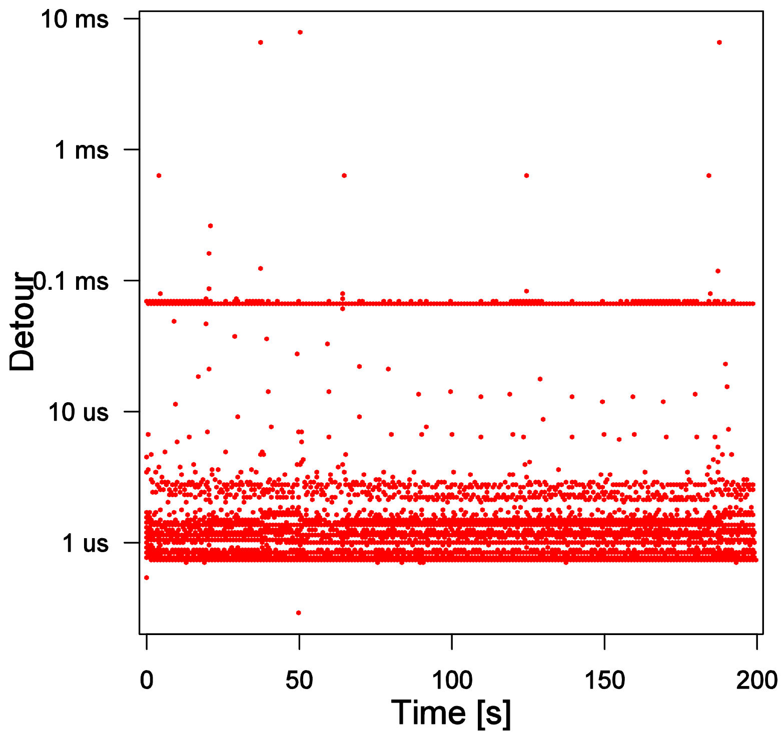
- Altix 4700, 2048 Itanium II cores, 13.1 Tflop/s, Linux 2.6.16
- Resolution: 25.1 ns, noise overhead: 0.05%

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%

First Observations from Scatterplots

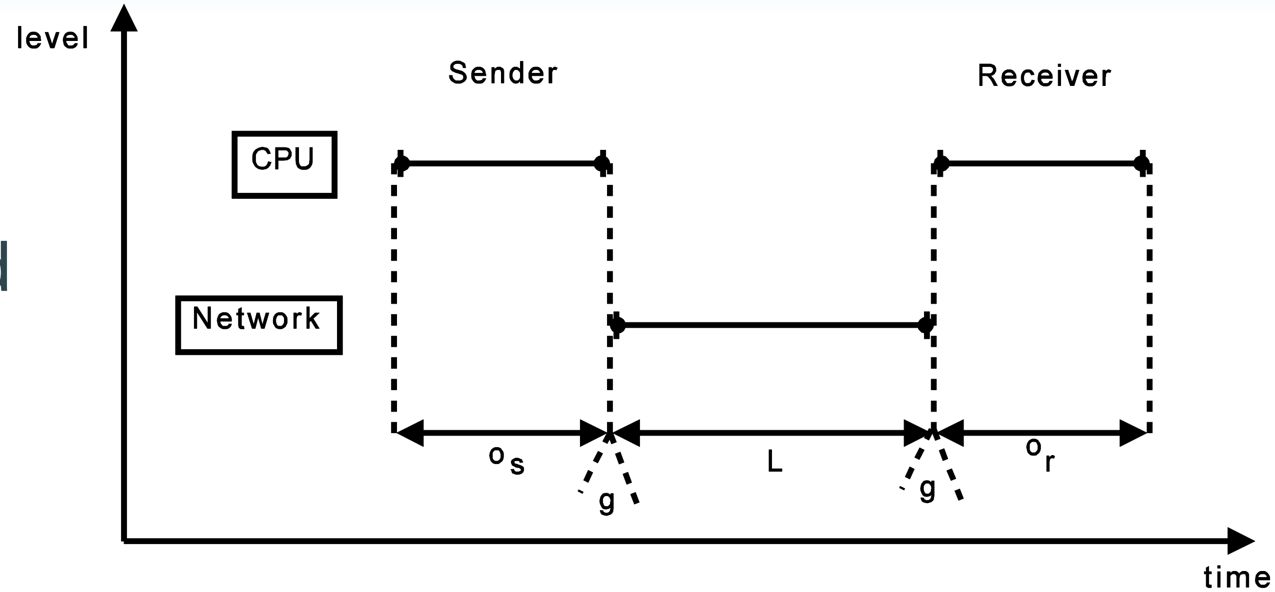
- Aggregate noise is generally low (<0.26%)
  - Systems have been well tuned
  - Cray has lowest aggregate noise level
- Noise distribution is very different
  - Specific noise shape for each OS
  - Altix and ZeptoOS are very “regular”
  - Chic and Jaguar are less regular

Modeling Message Passing Applications

- Message Passing Interface (MPI-1) is a widely used interface for parallel programming
  - Communication is explicit with messages
  - Send, recv, collectives also nonblocking
- Synchronization propagates or absorbs noise
  - Lamport's happens-before-relation for messages
  - Depends on relative time of send/recv (or wait)

An Analytical Model for Noise Propagation

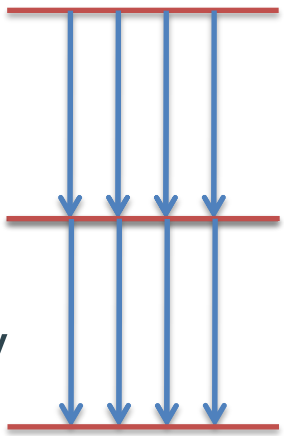
- LogP Model:
  - L – Latency
  - o – Overhead
  - g – Gap
  - P - # PEs



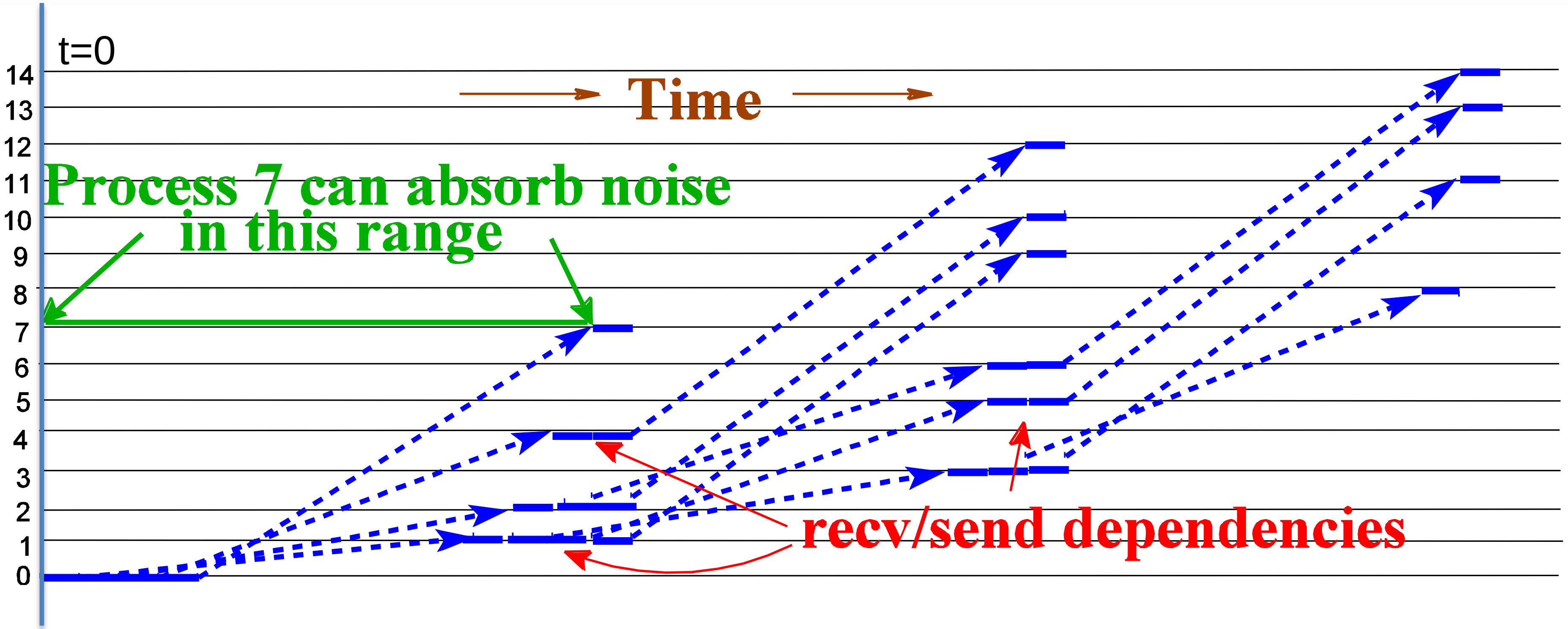
- Missing pieces:
  - Communication protocols, bandwidth, synchronization
  - LogGPS model (Ino et al.) captures most effects!
  - We added “O” to capture s/r overhead per byte
    - ➢ LogGOPS

MPI 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 collectives simultaneously
    - In fact, they leave as early as possible (when data is consistent)

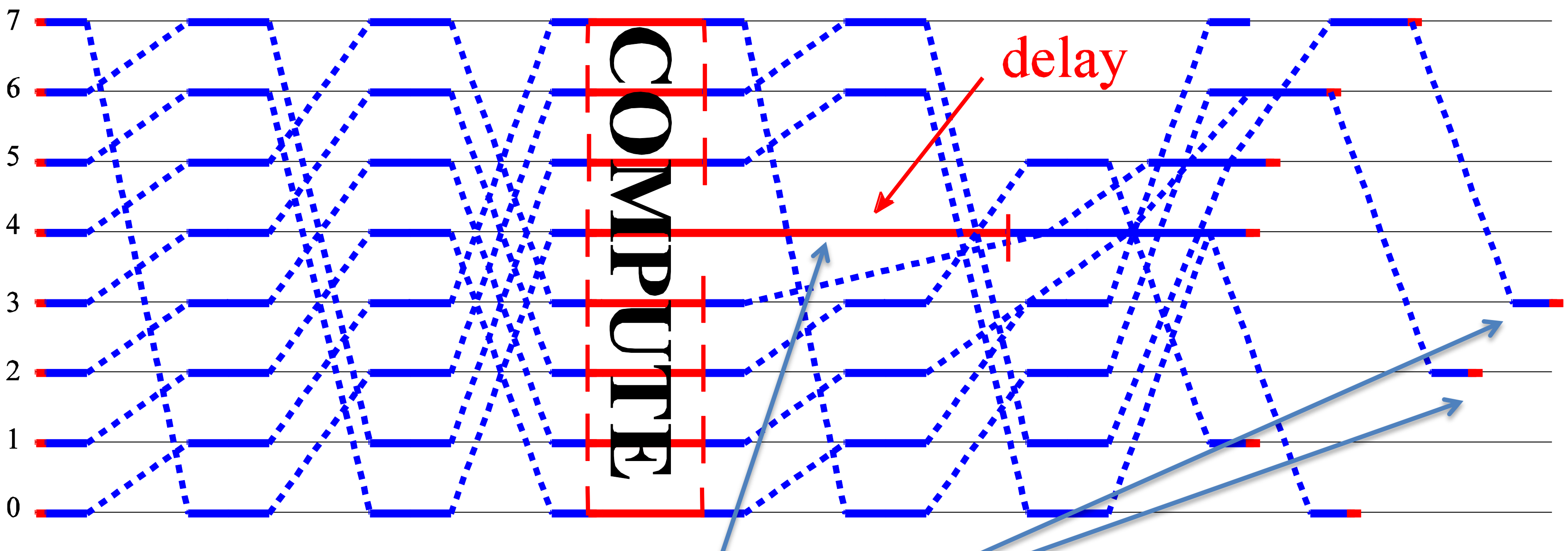


Example: Binomial Broadcast Tree



- 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`
  - Serial 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
  - Parameters are assessed by microbenchmark (Netgauge)

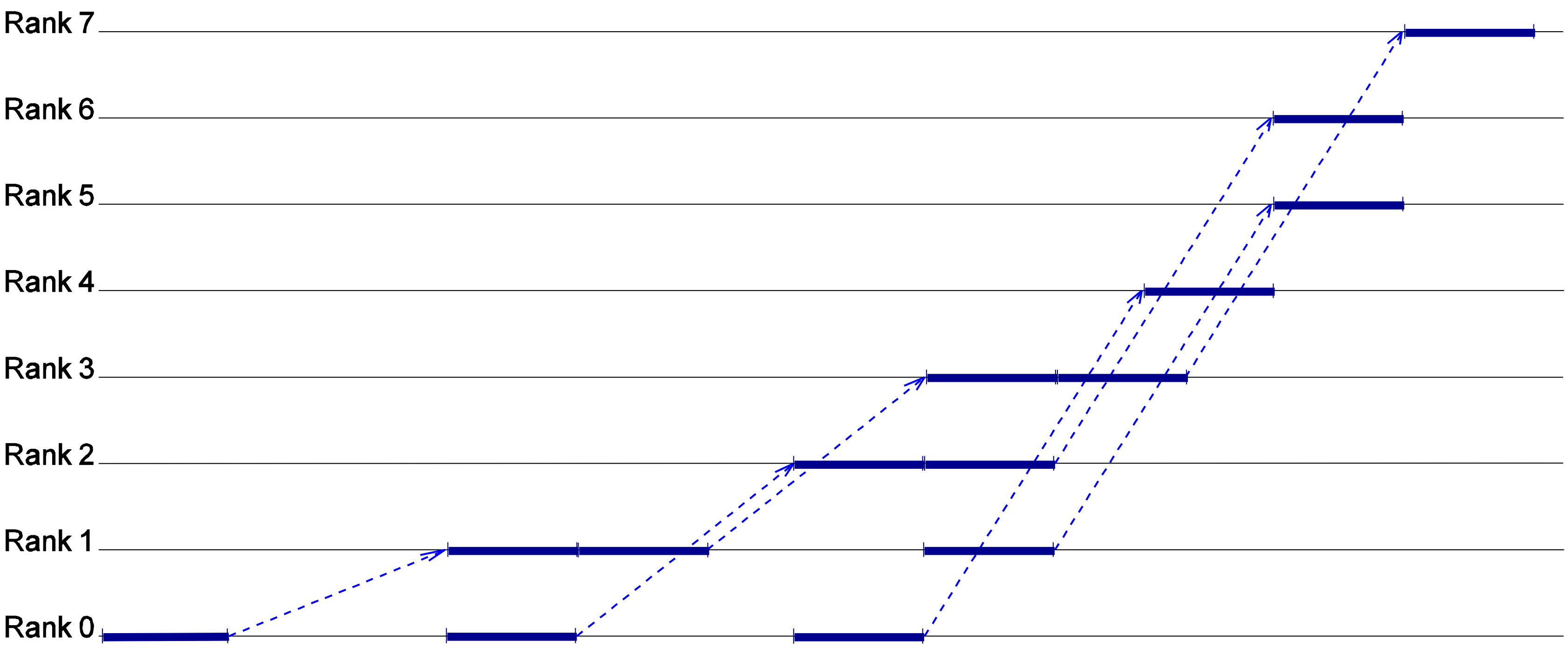
- Details: Hoefler et al. LogGOPSim – Simulating Large-Scale Applications in the LogGOPS Model (Workshop on Large-Scale System and Application Performance, Best Paper)

How to verify the simulator's correctness?

1. Analytic models of simple algorithms
   - Develop model and compare simulation result
   - We also visualized the output as LogP timeline
2. Simulate single collective operations
   - Compare to measurements
3. Simulate full application traces
   - Compare to measurements
4. Compare performance measurements with noise
   - Compare to related work (Beckman, Ferreira)

1) LogGOPS Verification – Binomial Tree

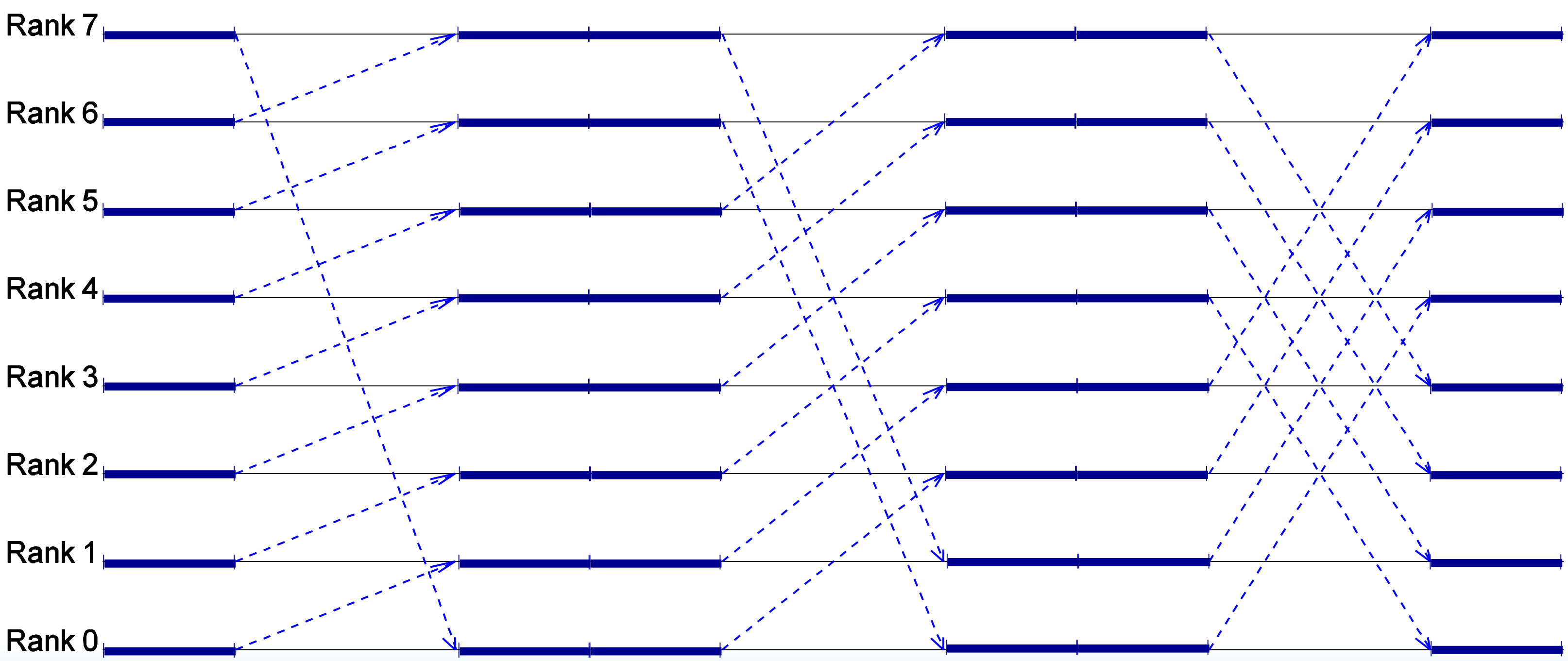
$$T_{bino} = (2o + L + \max\{sO, sG\})\lceil \log_2 P \rceil$$



1) LogGOPS Verification - Dissemination

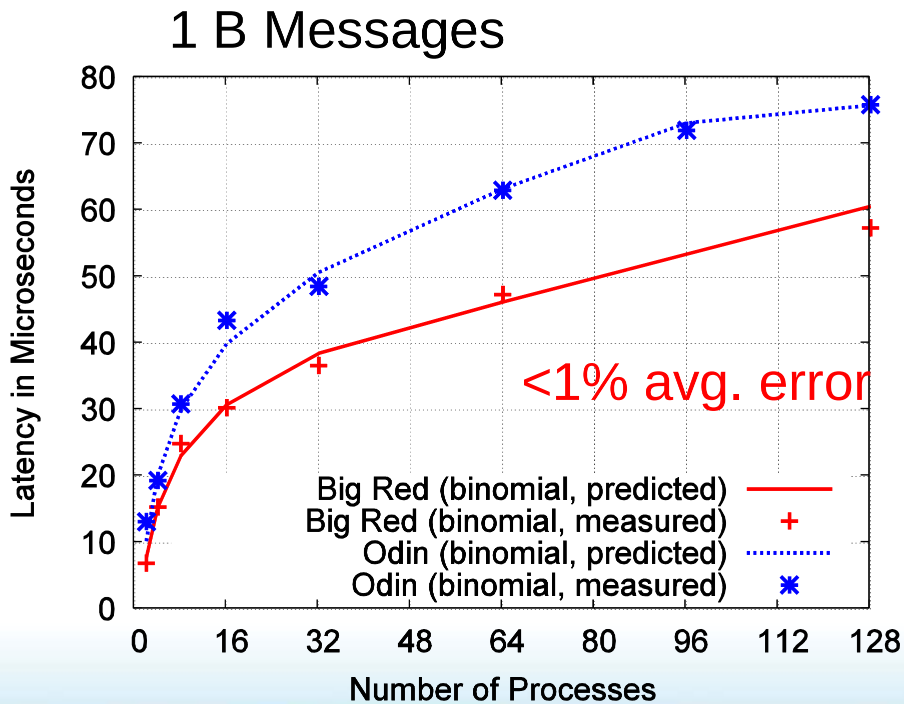
$$\delta = \begin{cases} (s-1)O - L : (s-1)O - L > 0 \\ 0 : \text{otherwise.} \end{cases} \quad (1)$$

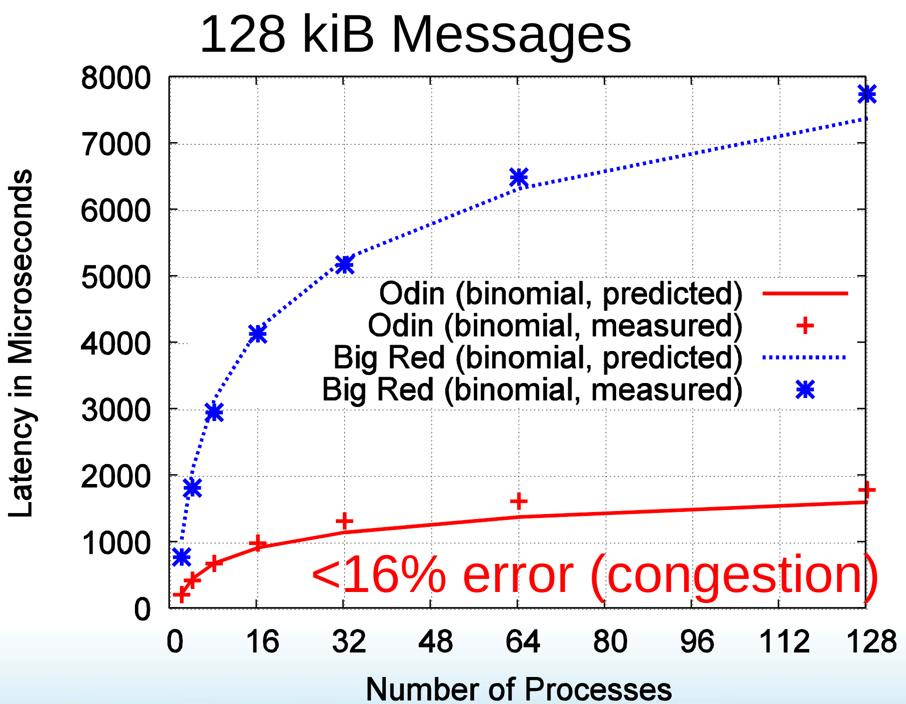
$$T_{diss} = (\delta + 2o + L + \max\{sO, sG\})\lceil \log_2 P \rceil \quad (2)$$



2) Collective Operation Verification - Experiments

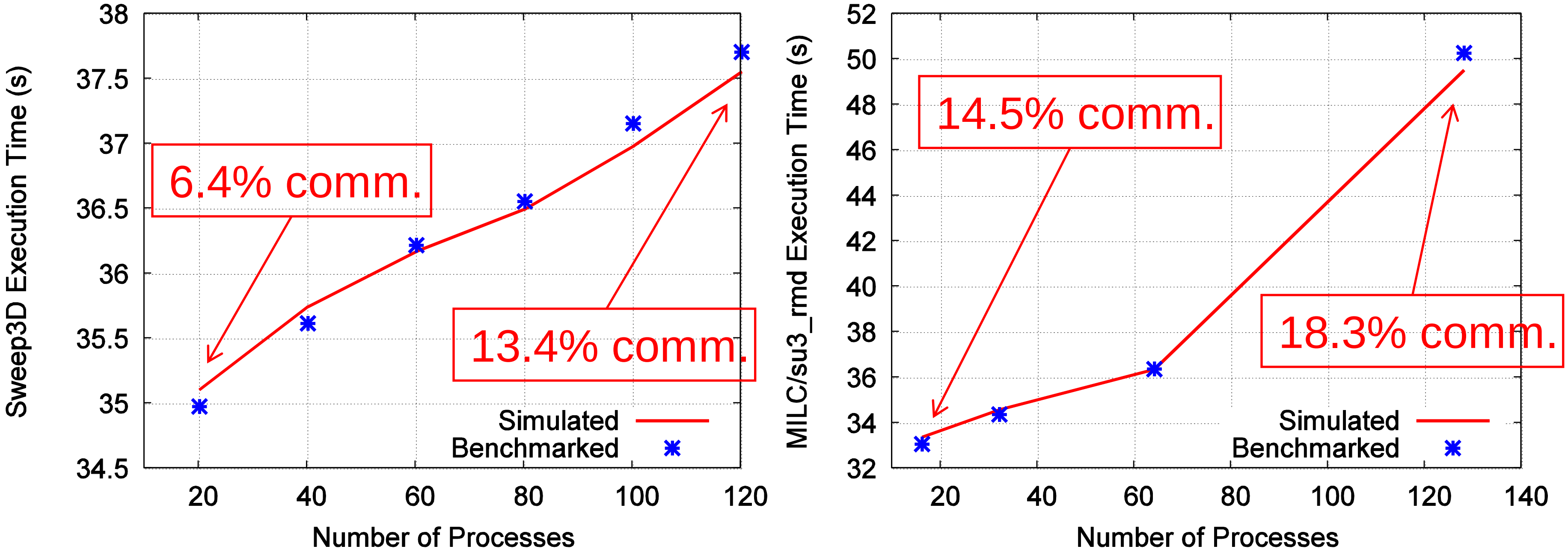
- Odin: $L=5.3\mu s$, $o=2.3\mu s$, $g=2\mu s$, $G=2.5ns$, $O=1ns$
- Big Red: $L=2.9\mu s$, $o=2.4\mu s$, $g=1.7\mu s$, $G=5ns$, $O=2ns$





3) Application Verification - Experiments

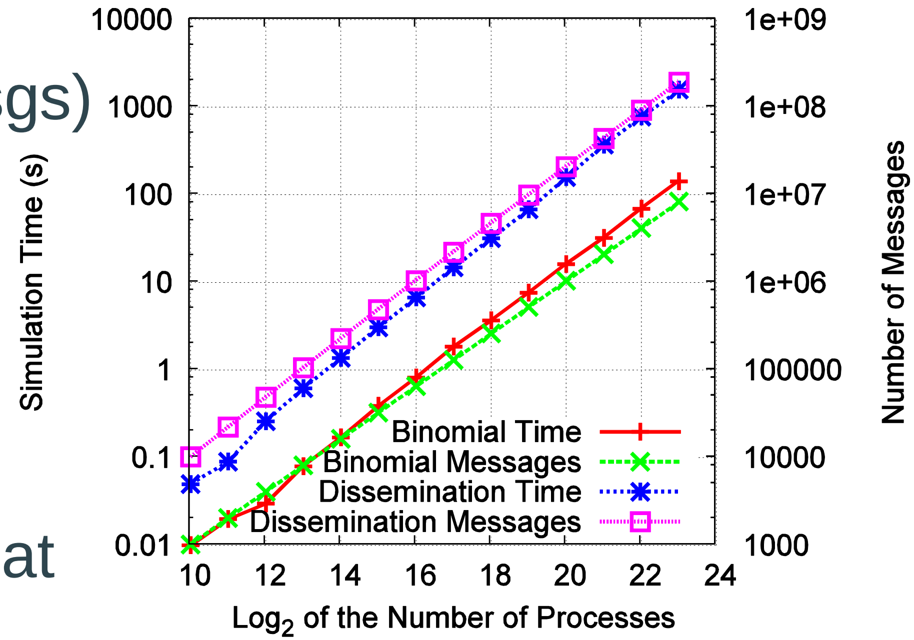
- Sweep3D and MILC weak scaling on Odin



- <2% average error

How fast is the simulation of collectives?

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



How fast is the simulation of applications?

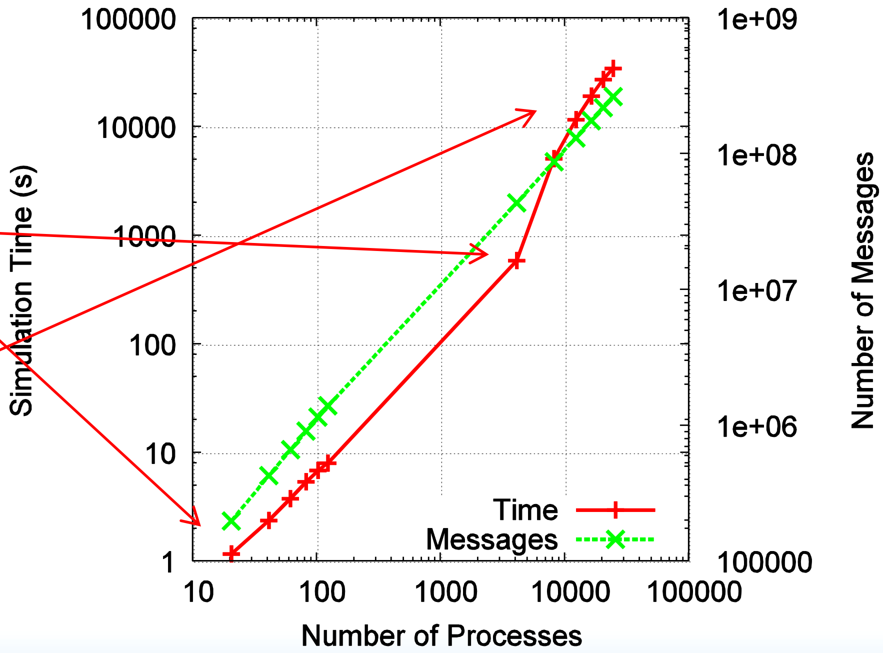
- 37.7 s Sweep3D extrapolated from 40-28k CPUs
  - 0.4 M messages → 313 M messages

40 CPUs – 2.43 s

4k CPUs – 10 min

28k CPUs – 9.7h (swap)

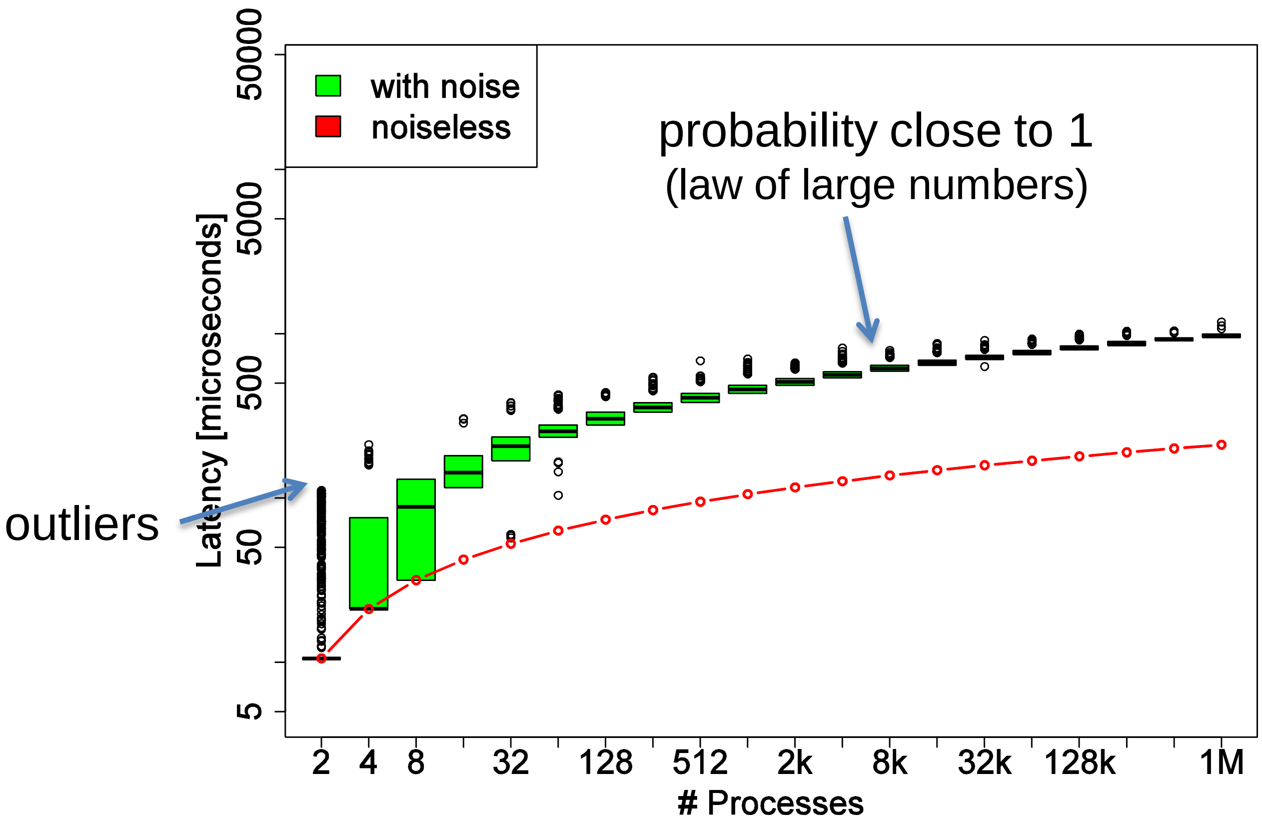
Main memory is an issue! hits swap at 8k CPUs

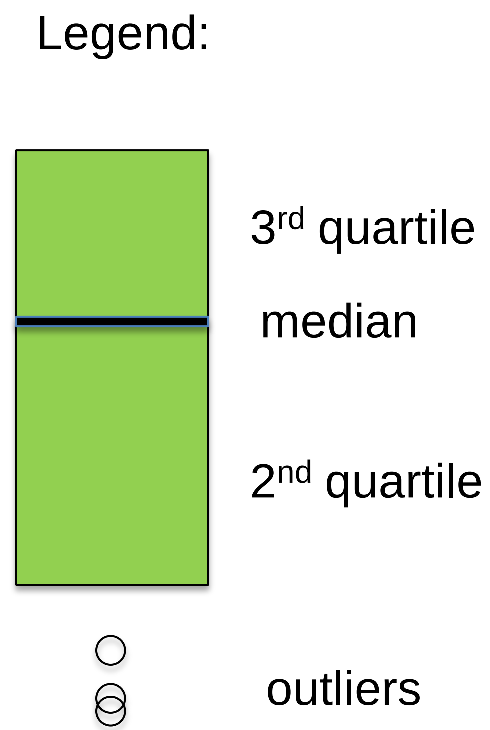


Excursion: More Use-Cases for LogGOPSim

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 System Noise at very large scale**

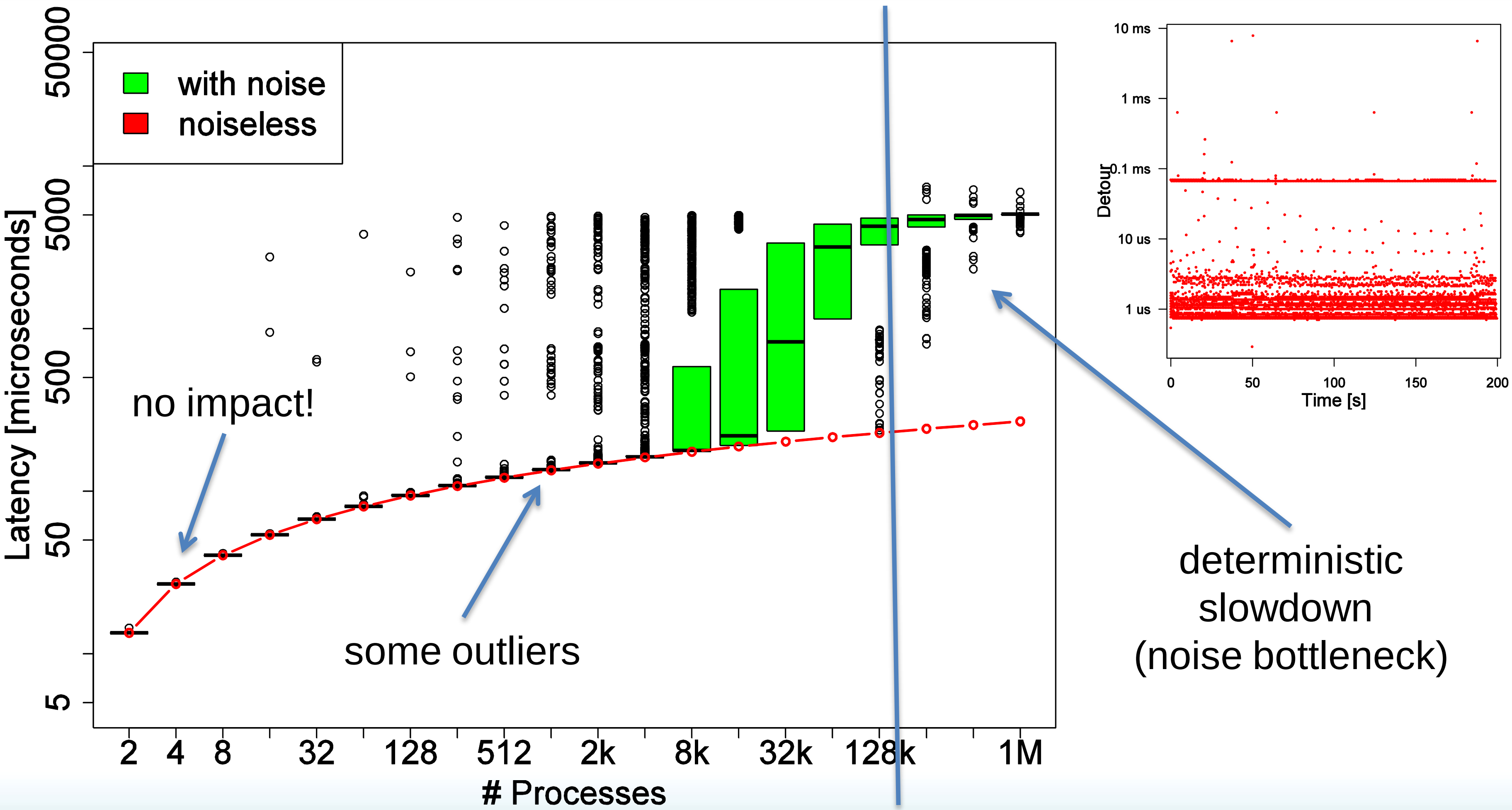
Single Byte Collective Operations and Noise



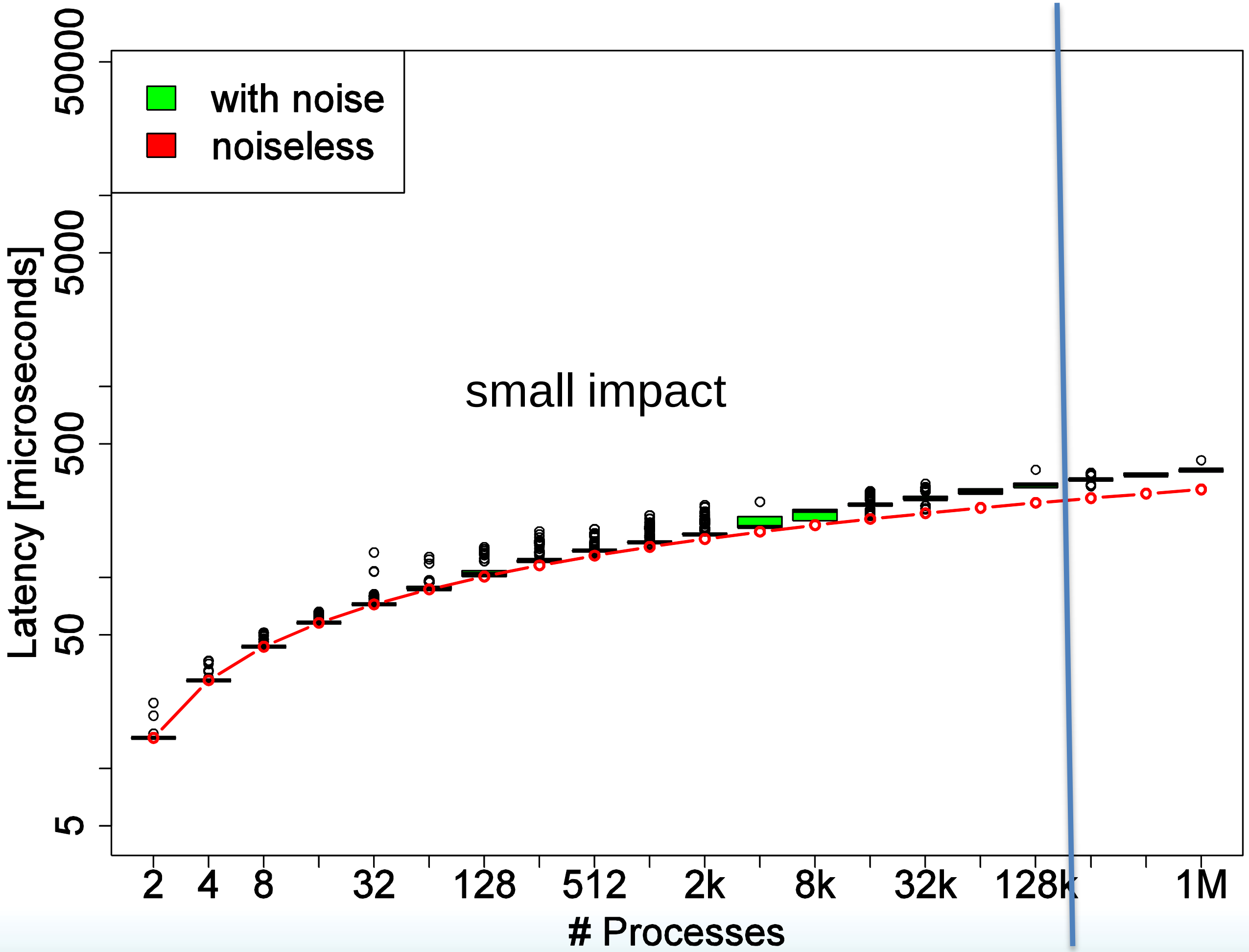


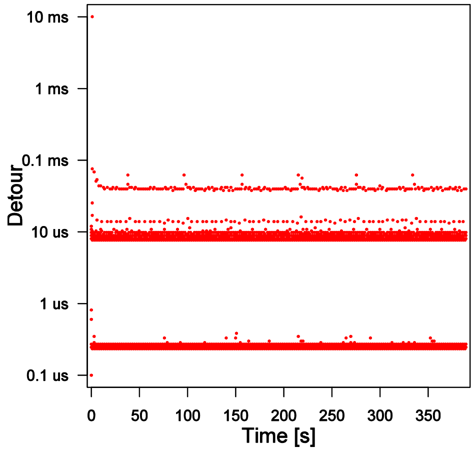
- 1 Byte, Dissemination, regular noise, 1000 Hz, 100 µs

Single Byte Dissemination on Jaguar

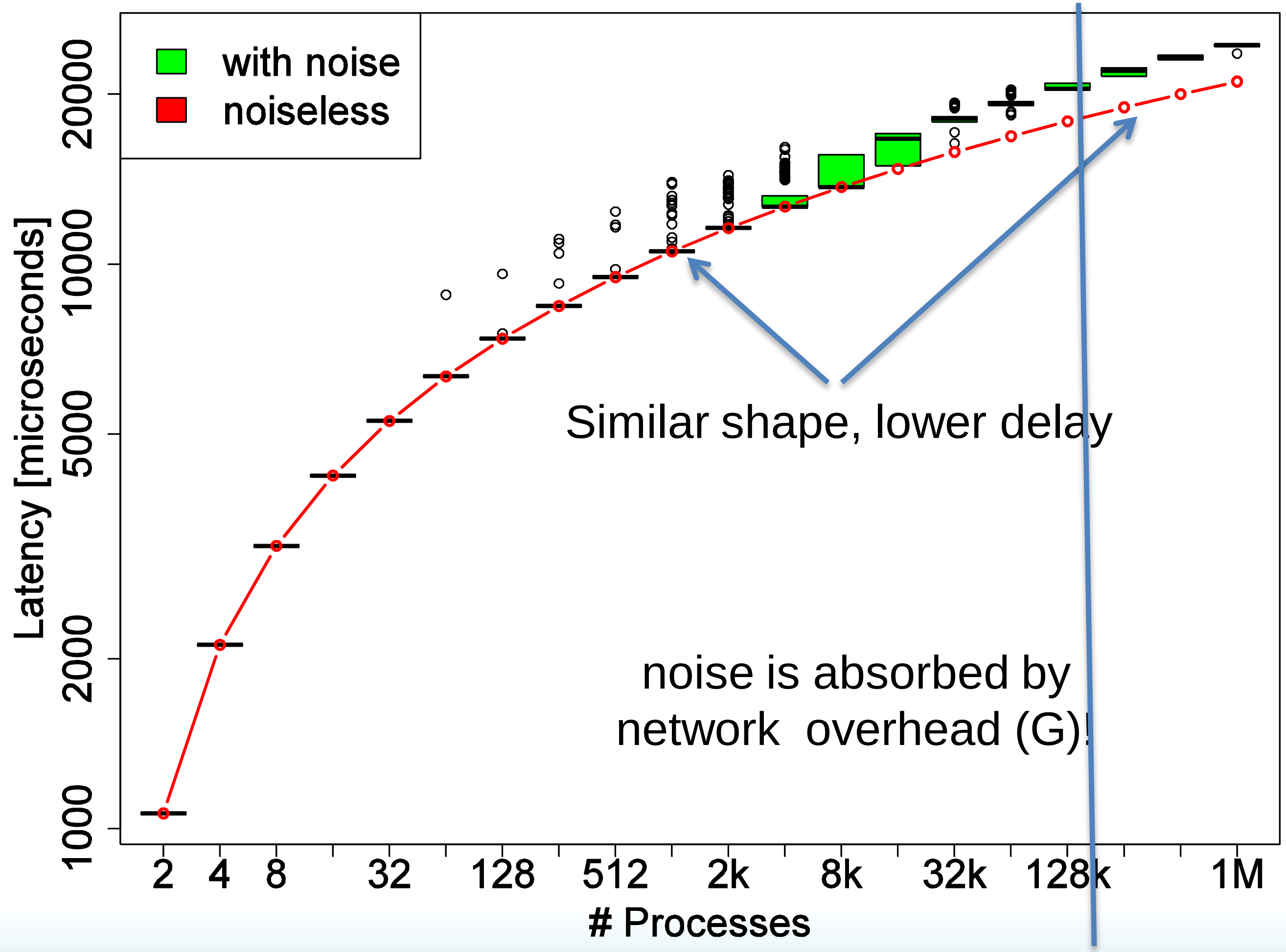


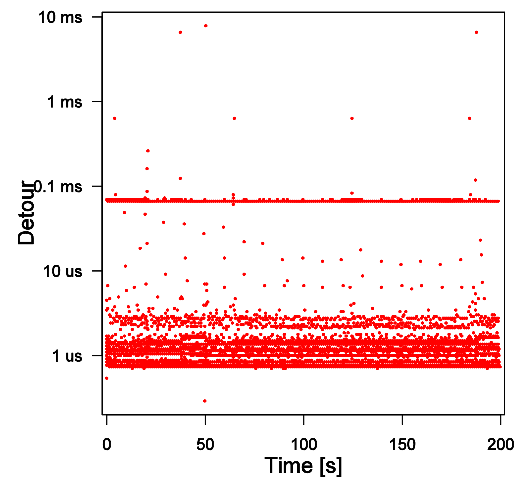
Single Byte Dissemination on ZeptoOS



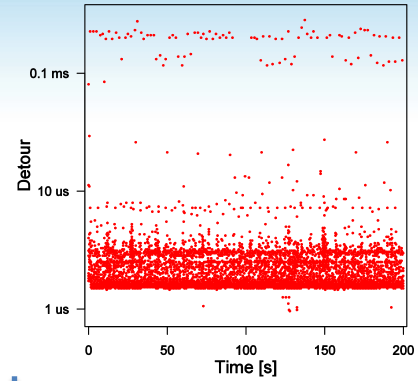


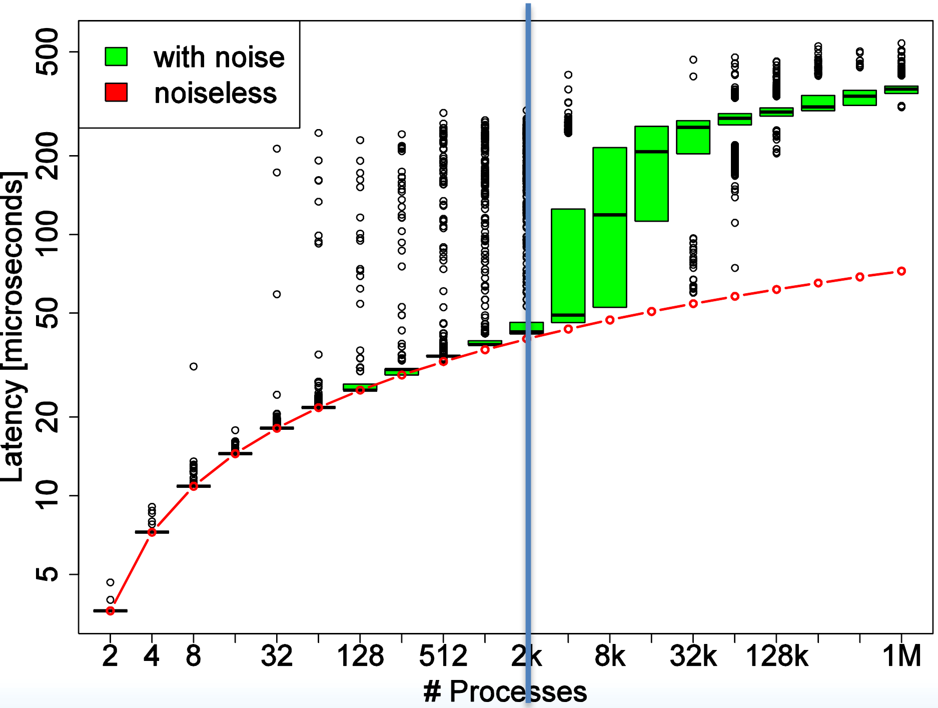
1MiB Dissemination on Jaguar



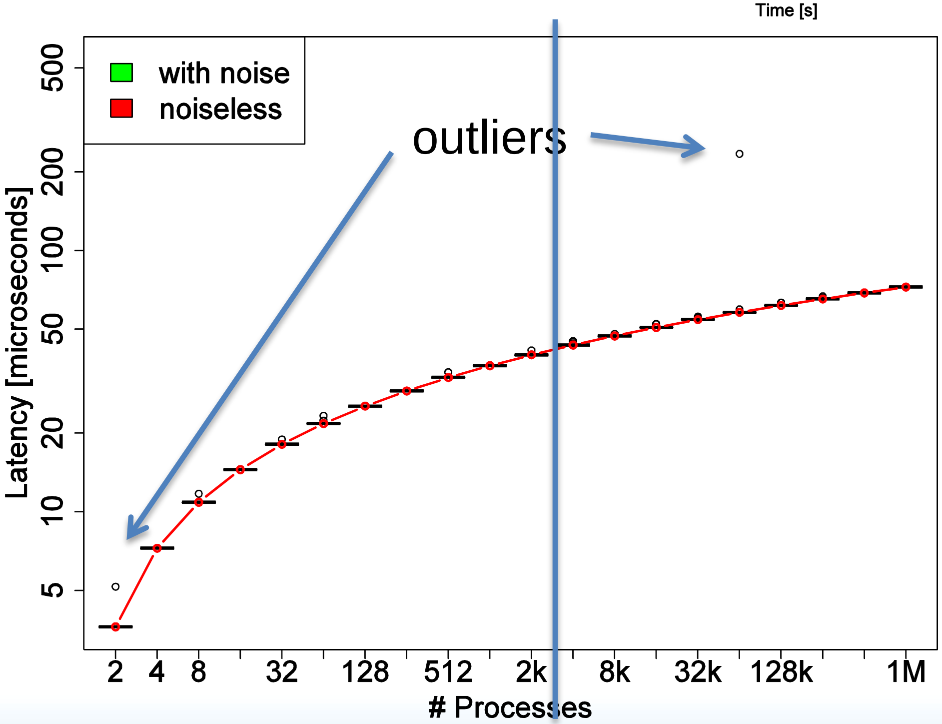


Effect of Co-Scheduling Noise (Altix)



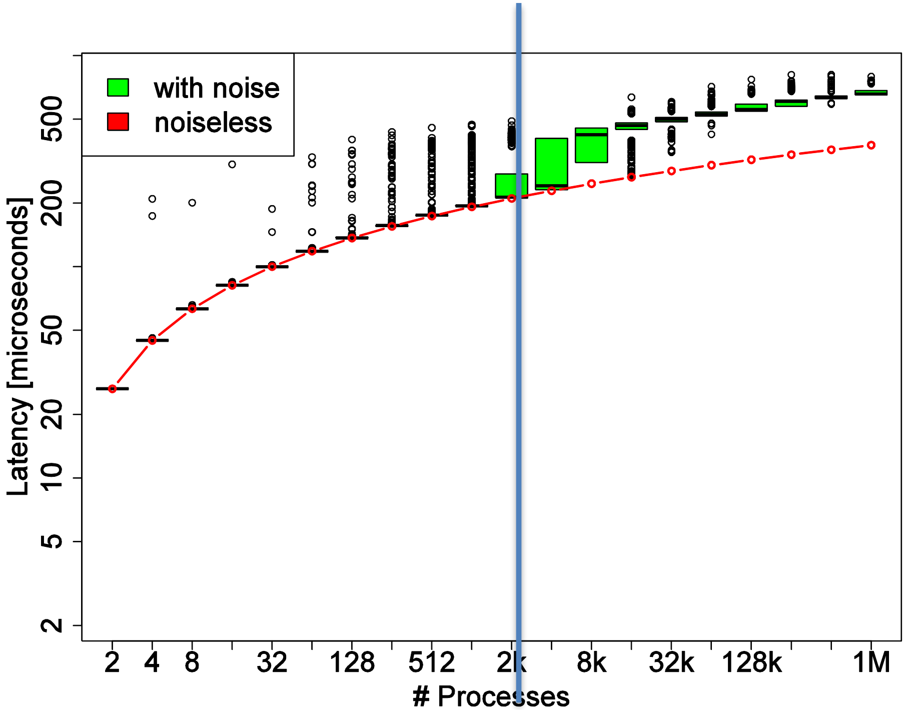


Normal

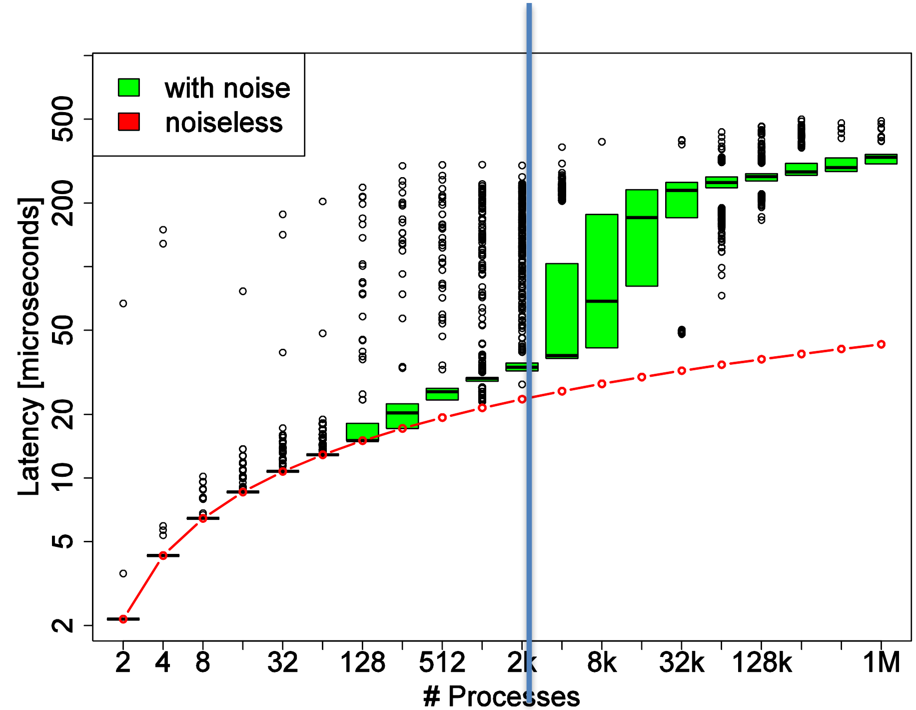


Co-Scheduled

Does the Network Speed Matter (Altix)?



0.1x network speed

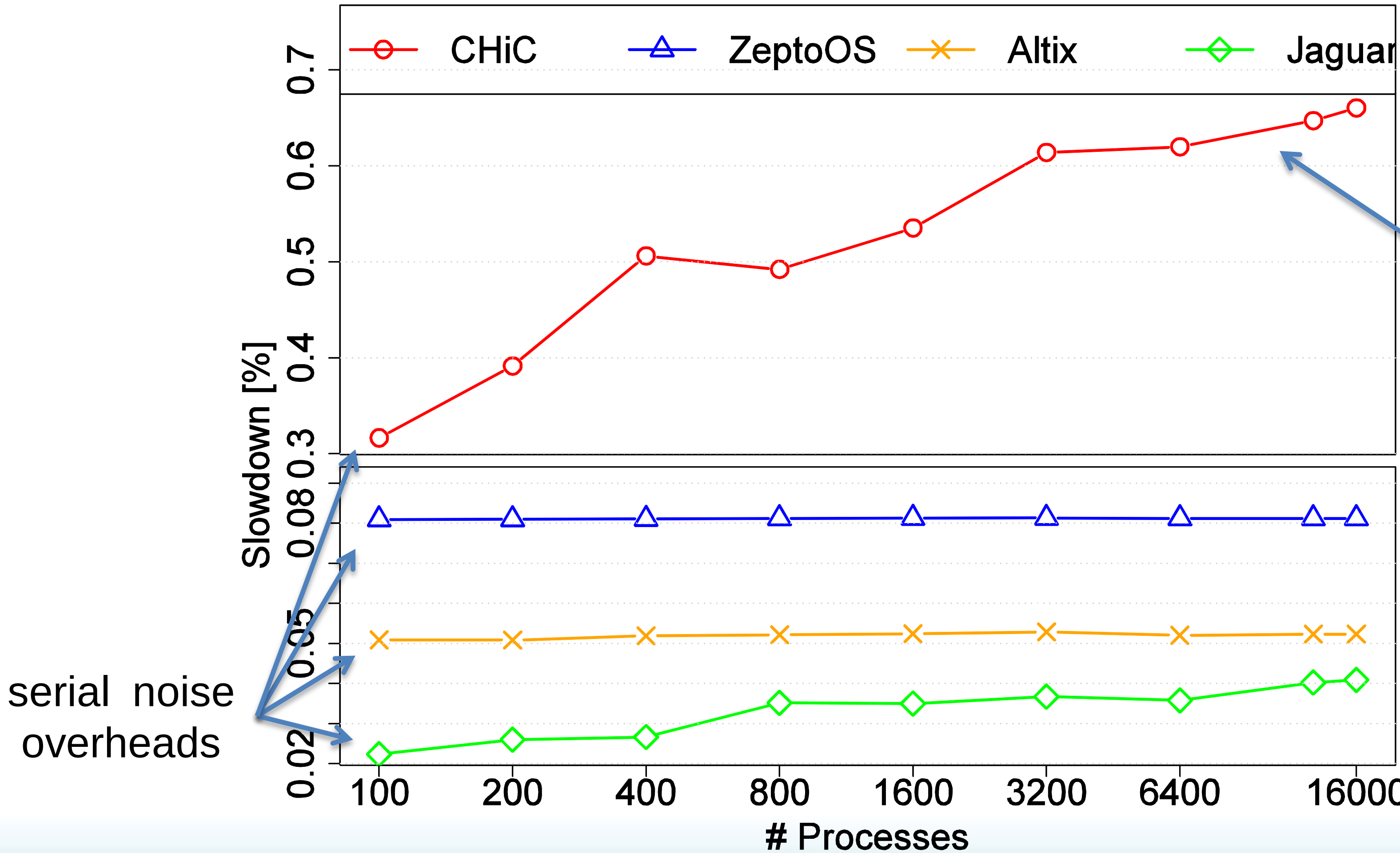


10x network speed

Method: increase/decrease L,G,g

Observation: faster network doesn't help (noise bottleneck)

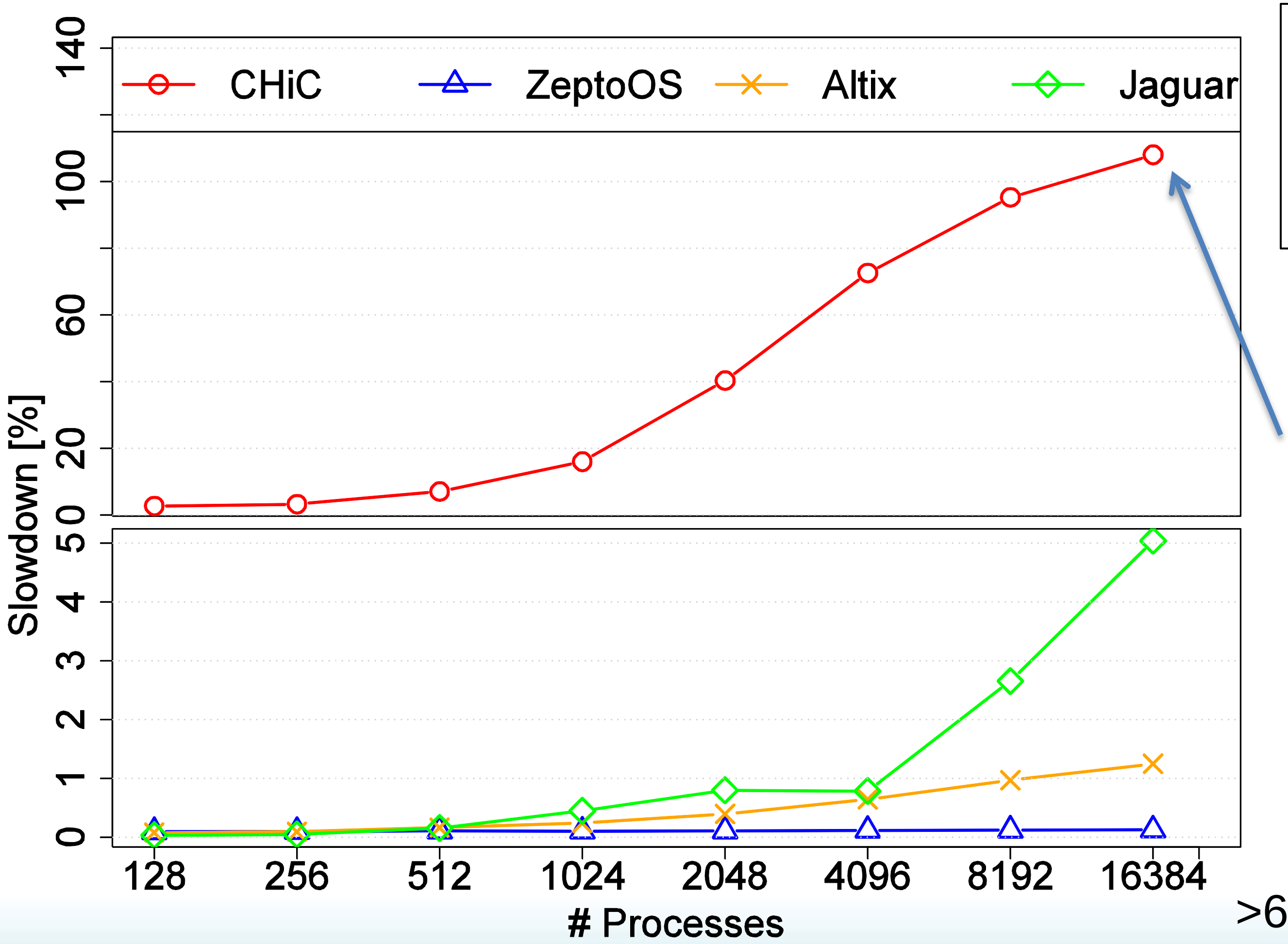
Sweep3D (Collective and Point-to-Point)



92.4% bl p2p
4 bcast
3 barrier
32 allreduce

>289 million messages!

POP (Collective and Point-to-Point)



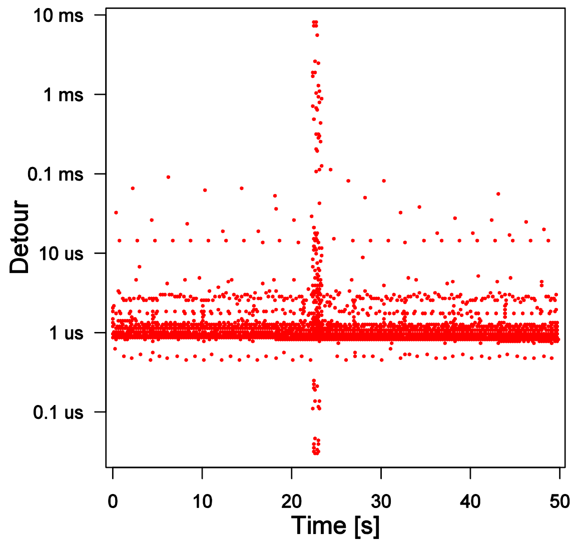
0.2% nb p2p
703 bcast
575 barrier
608 allreduce

>2x slowdown!

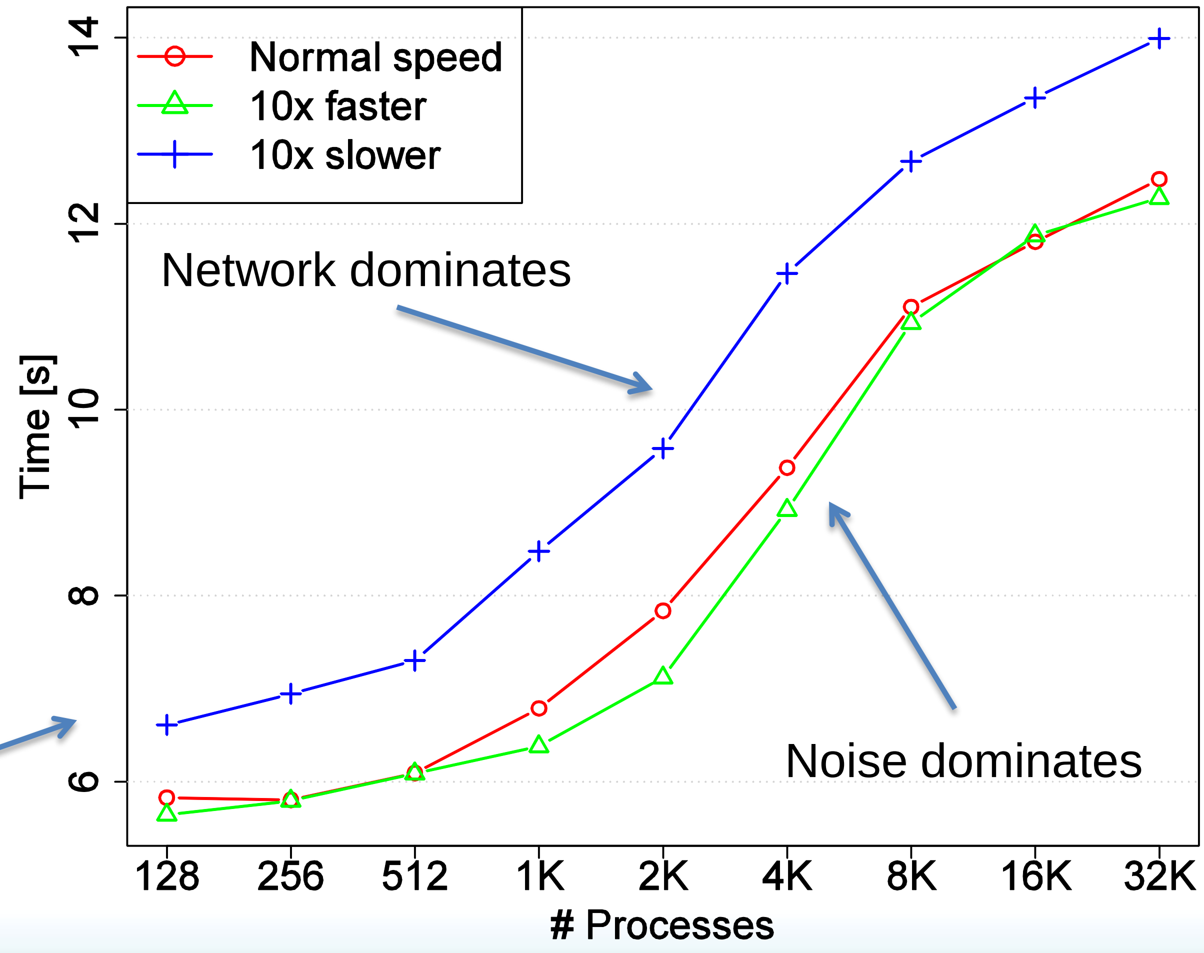
>625 million messages!

Influence of Network Speed on Applications

POP @ CHiC



"Noise bottleneck": faster network does not increase performance



Conclusions and Impact

- 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 cannot help, noise will dominate!
- We developed a tool-chain to quantify the noise bottleneck
  - Available online: `http://www.unixer.de/LogGOPSim`

Future and Ongoing Work on Noise

- Develop analytical model
  - Already showed inaccuracy of previous models
- Influence of “fuzziness” in co-scheduling
  - There is no ideal co-scheduling as we simulated
- Influence of “slack” in synchronizations
  - Nonblocking messages and collective operations
- Noise-resilient algorithm design
  - Collective operations, CG method

Research Outline

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*)

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