LogGOPSim – Simulating Large-Scale Applications in the LogGOPS Model

Torsten Hoefler, Timo Schneider, Andrew Lumsdaine

Presented at the Workshop on Large-Scale System and Application Performance (LSAP'10) on June 21st 2010

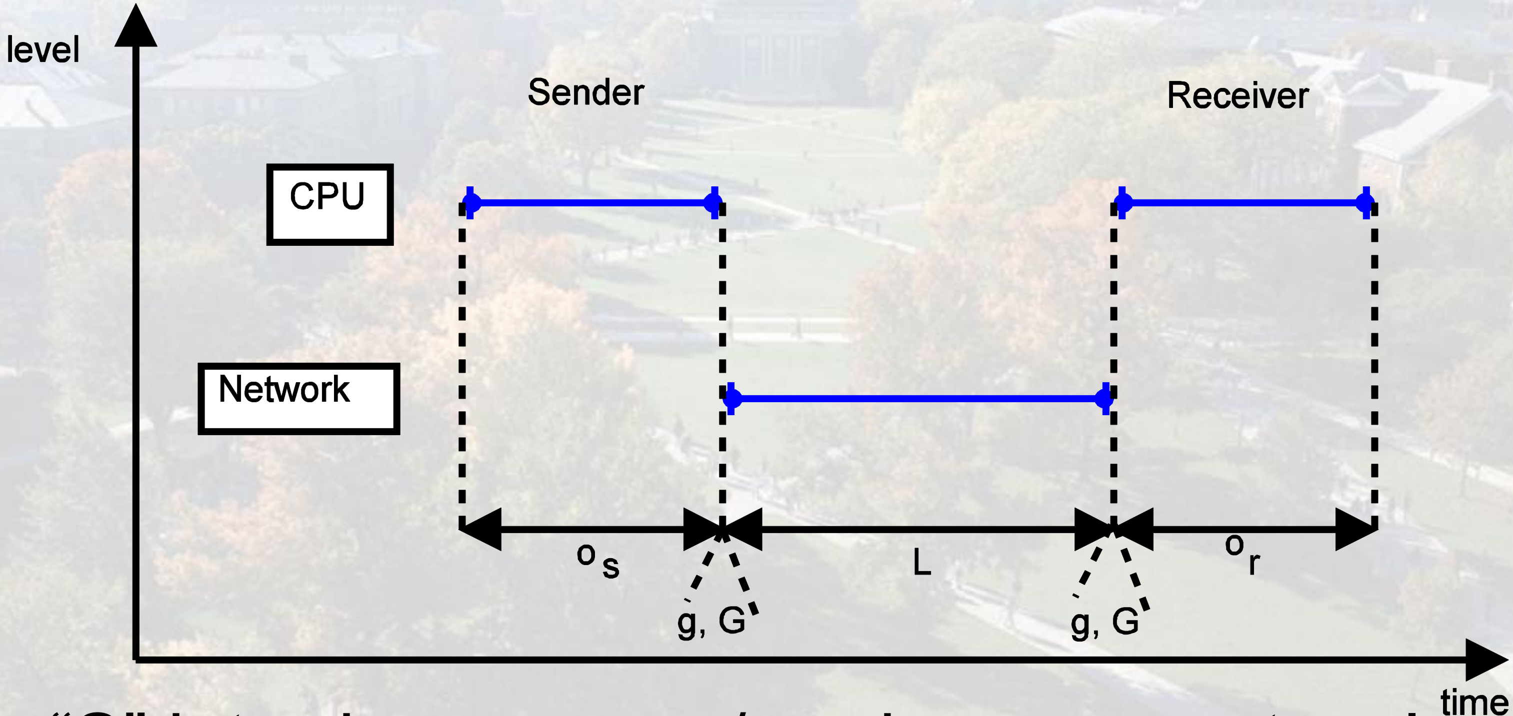
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Motivation – Why Simulation?

- Analytic methods can quickly become too complex and infeasible
- White-box analysis of application performance (count events, trace backwards)
- Understand complex phenomena in parallel programs (e.g., chained collectives)
- Save on expensive experiments or predict future systems (e.g., Blue Waters)

Why LogP, LogGP, LogGPS?

- The LogGPS model is well established



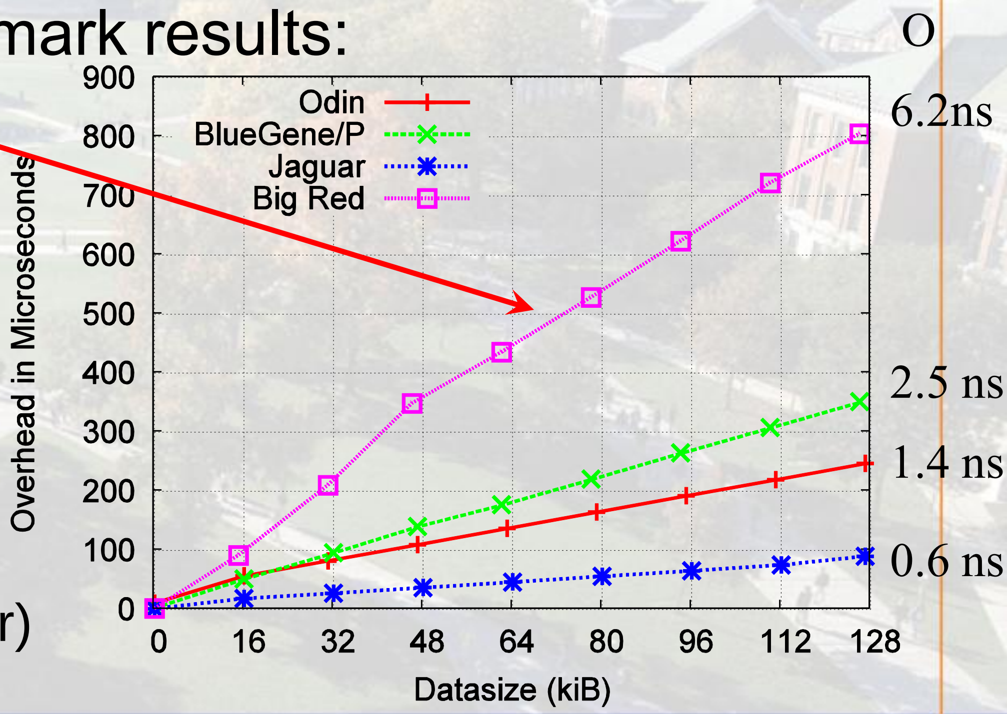
- “S” introduces eager/rendezvous protocols

And now LogGOPS?

- CPU overhead "o" is constant in the LogGPS model (independent of message size)
- Netgauge "loggp" benchmark results:

Overhead = o+s*O

- O = time per byte!
- Systems:
  - Odin @ IU (InfiniBand)
  - Big Red @ IU (Myrinet)
  - BlueGene/P @ ANL
  - Jaguar @ ORNL (Sea Star)



How to model message passing?

- Must support MPI but should be independent
- Used Global Operation Assembly Language

```
rank 0 {
    l1: calc 100 cpu 0
    l2: send 10b to 1 tag 0 cpu 0 nic 0
    l3: recv 10b from 1 tag 0 cpu 0 nic 0
    l2 requires l1
}
```

- Can easily be generated manually, by scripts, or from any MPI trace
- Is compiled into an efficient binary format for simulation

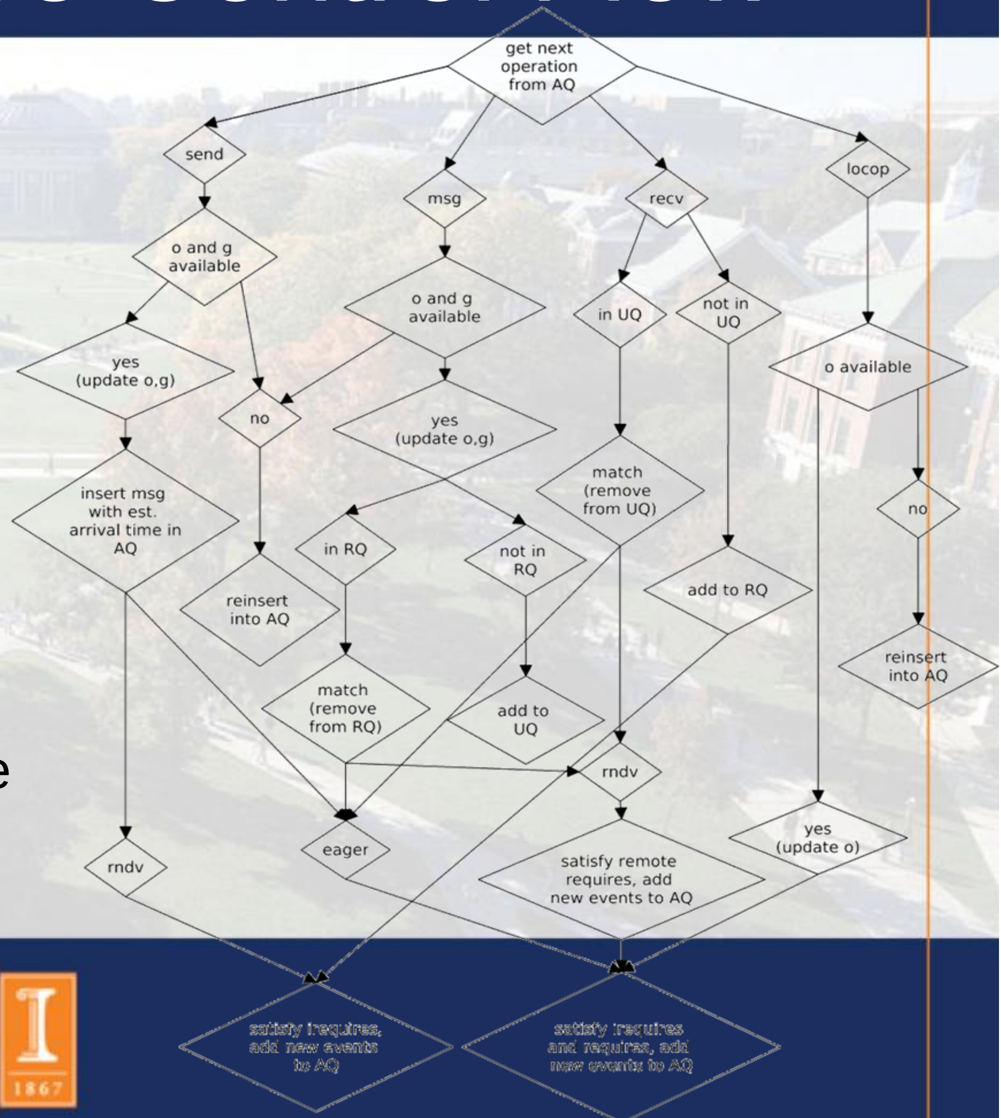
Design for Speed and Scalability

- Support MPI message semantics
  - Matching: source, tag + any_source, any_tag
  - Nonblocking send/recv (keyword irequires)
- Simulate eager/rendezvous protocols
  - eager: recv depends on send only
  - rndvz: send depends on recv and vice versa
- Semantics require two queues per process:
  - Unexpected queue (UQ): received eager msgs
  - Receive queue (RQ): posted receives
- Each proc has virtual time for o and g
  - Supports multiple CPUs and multiple NICs per process

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



Limitations and Assumptions

- LogGOPSim ignores congestion
  - assumed full bisection bandwidth by definition
  - High effective bisection topologies (e.g., Fat Tree, Clos, Kautz) are accurately simulated
    - Often have >70% effective bisection bandwidth
  - Congestion simulation is implemented
    - comes at the cost of speed
- Messages are delayed until o, g are available at receiver (this is undefined in LogGPS)
- I/O is not considered

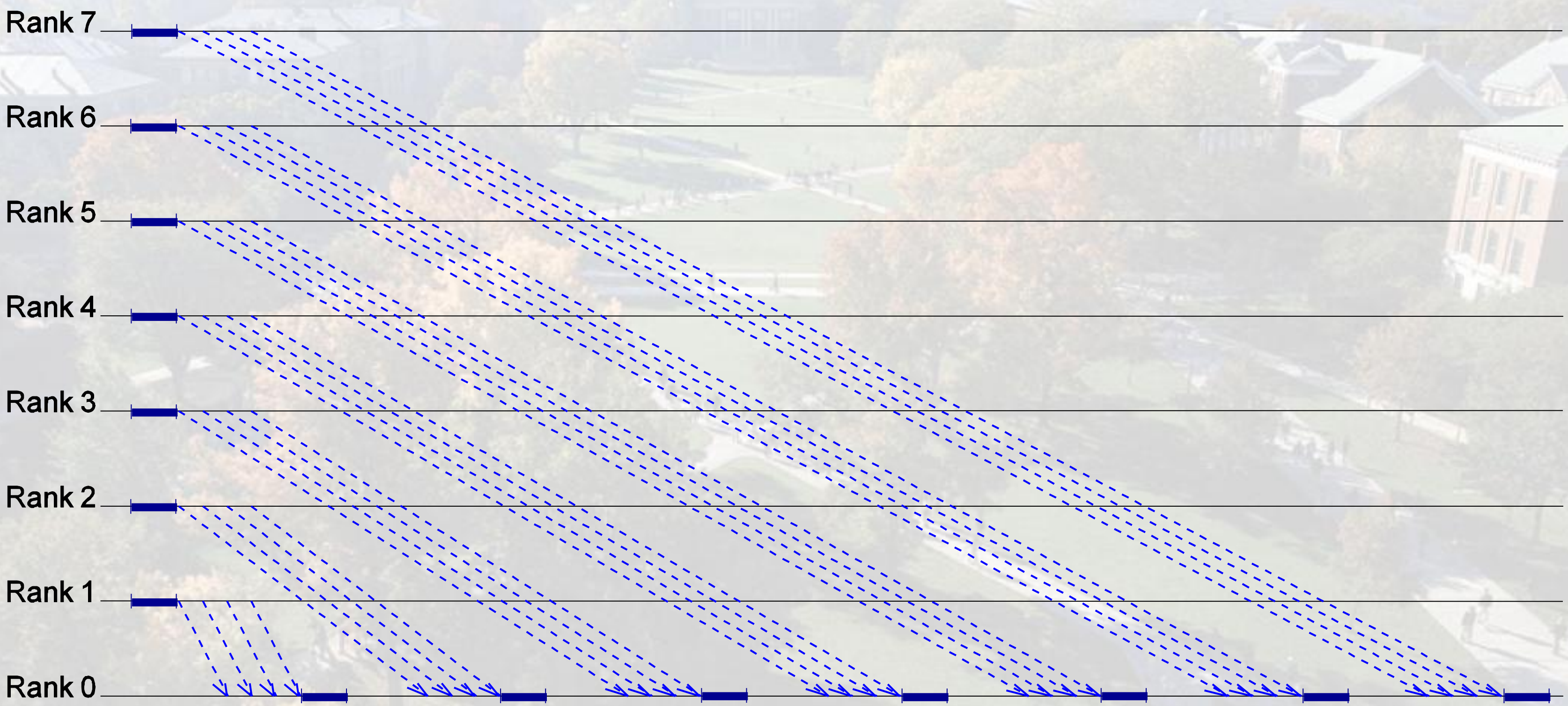
Verification – Linear Scatter

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

- LogGOPS makes verification simple

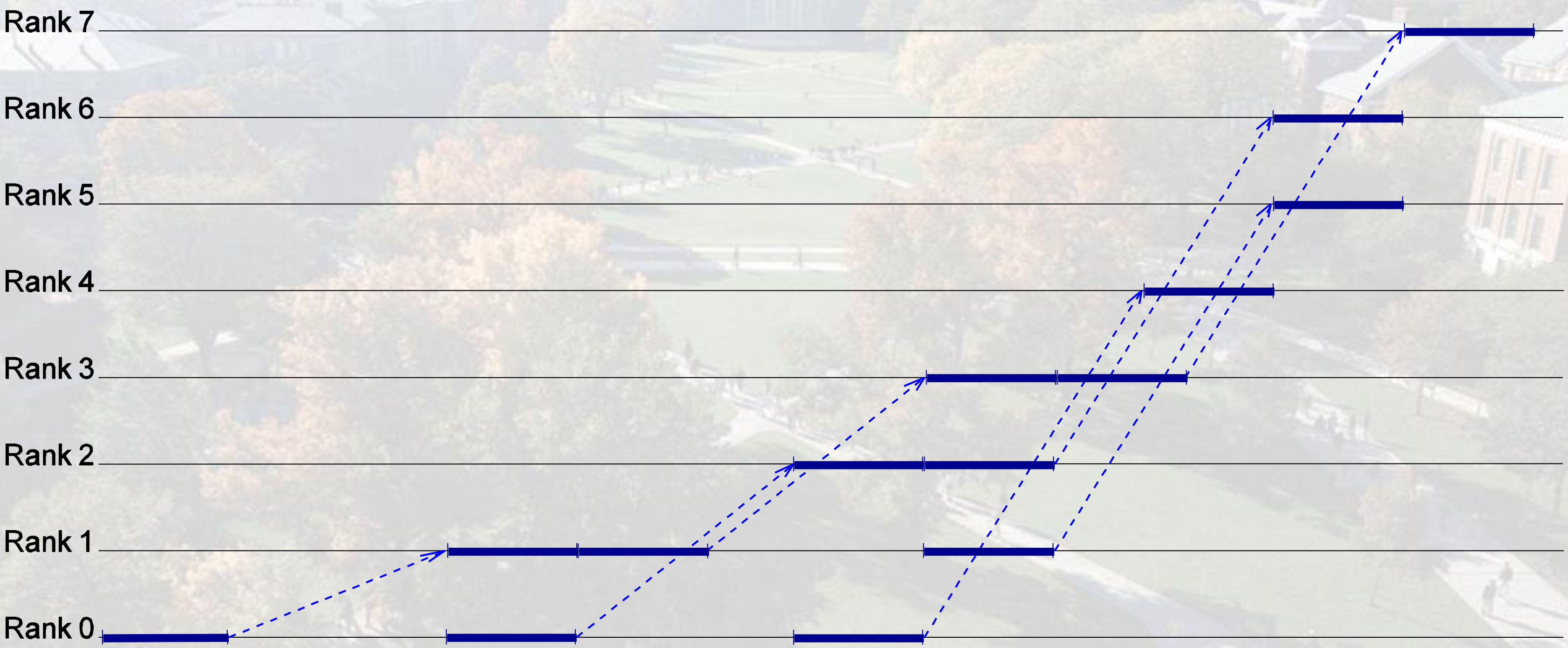
Verification - Gather

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



Verification – Binomial Tree

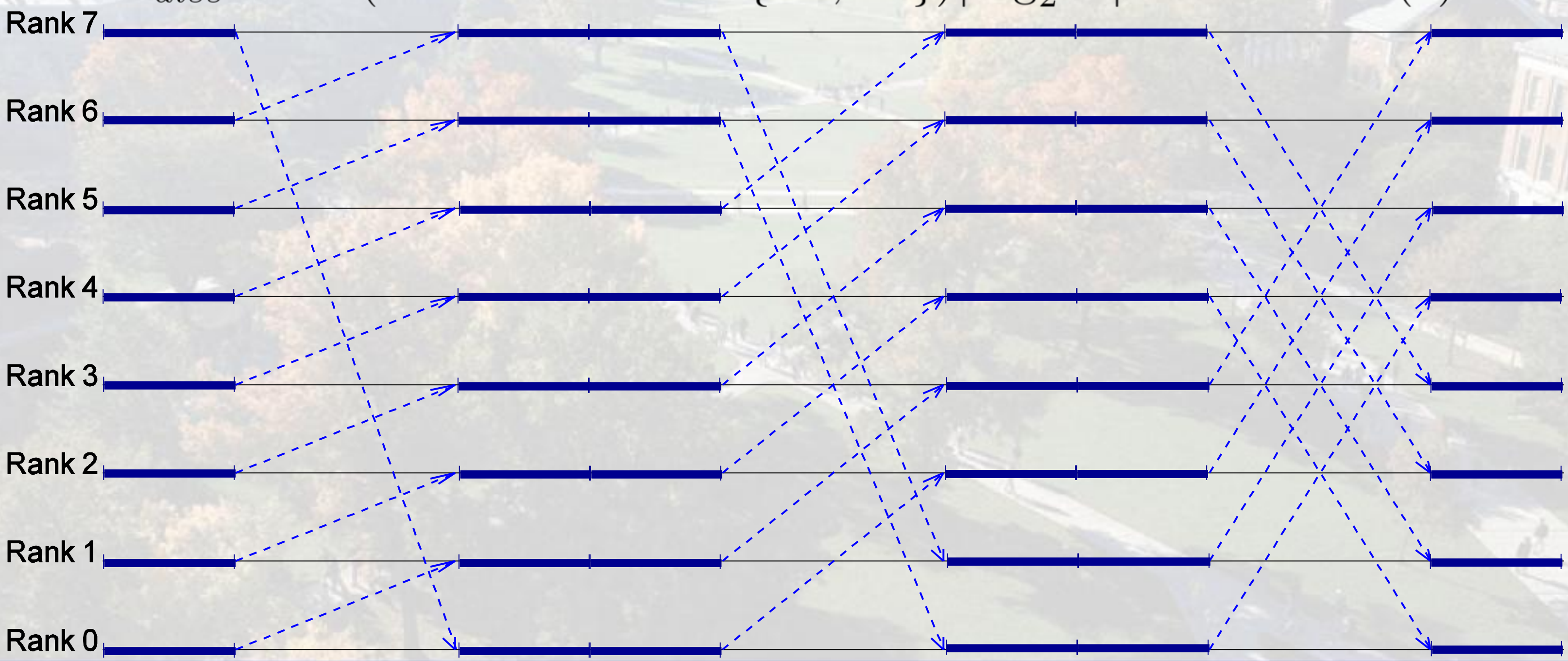
$$T_{bino} = (2o + L + \max\{sO, sG\})\lceil \log_2 P \rceil$$



Verification - Dissemination

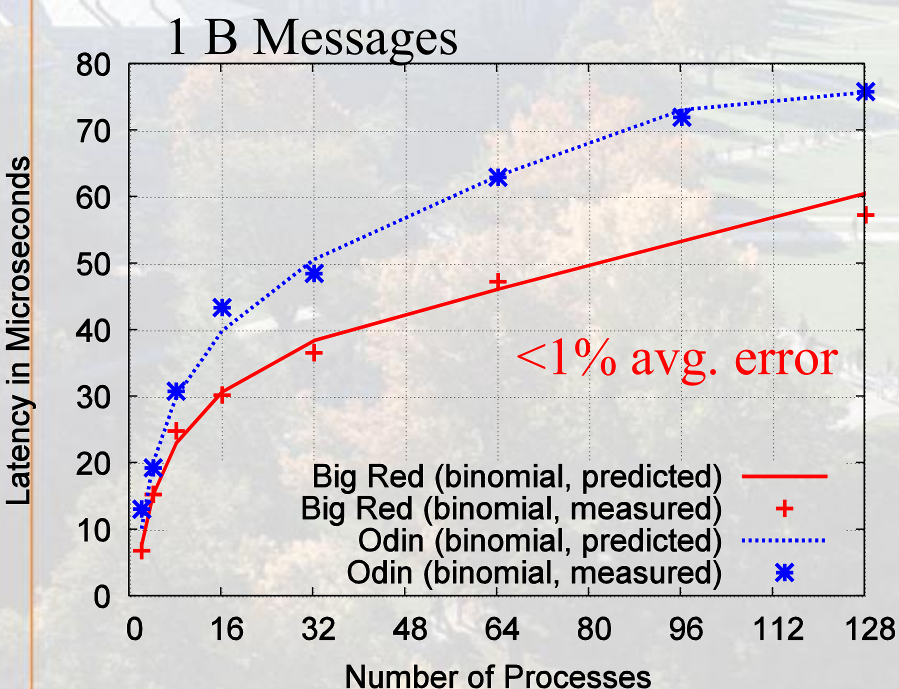
$$\delta = \begin{cases} (s-1)O - L : (s-1)O - L > 0 \\ 0 : \text{otherwise.} \end{cases} \quad (1)$$

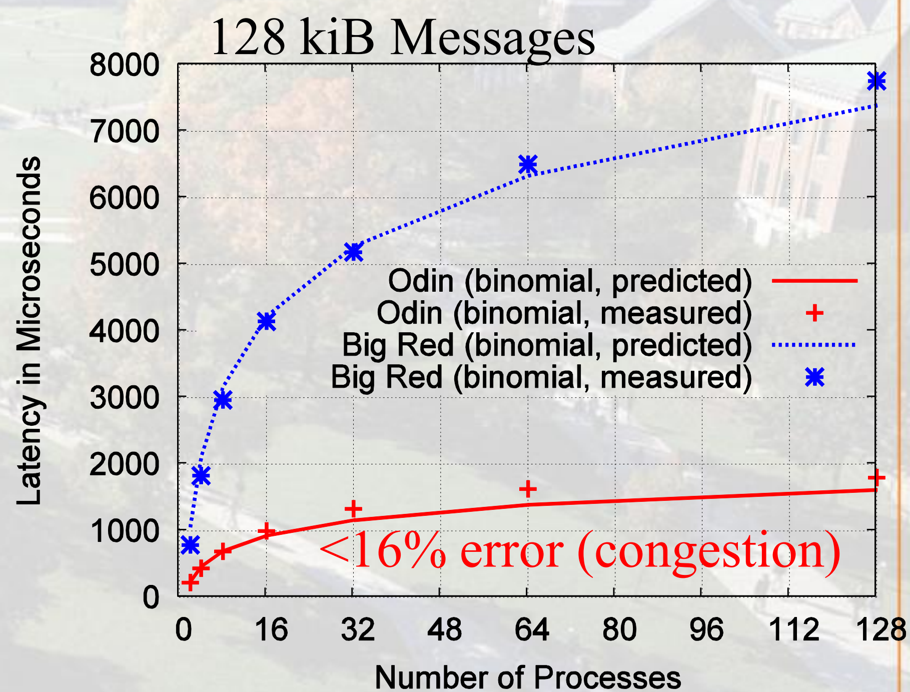
$$T_{diss} = (\delta + 2o + L + \max\{sO, sG\})\lceil \log_2 P \rceil \quad (2)$$



Experimental Evaluation

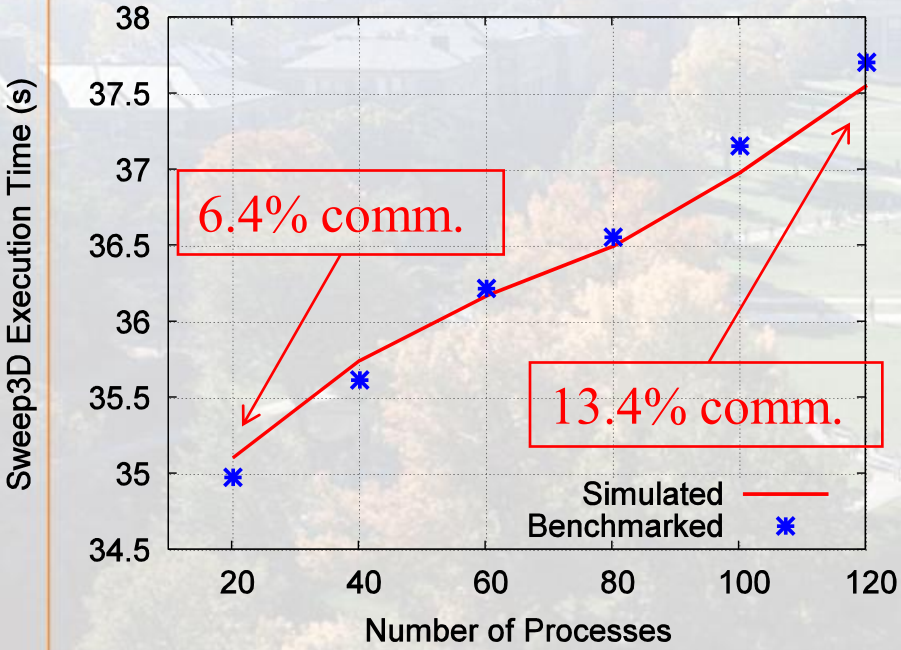
- Odin: $L=5.3\mu s$, $o=2.3\mu s$, $g=2\mu s$, $G=2.5ns$, $O=1ns$
- Big Red: $L=2.9\mu s$, $o=2.4\mu s$, $g=1.7\mu s$, $G=5ns$, $O=2ns$

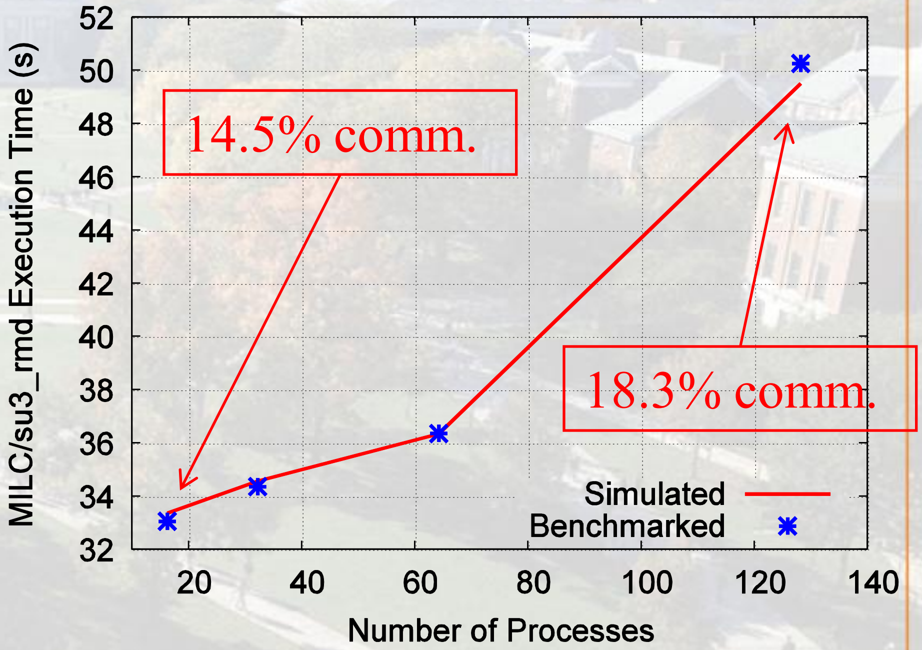




Application Simulation Accuracy

- Sweep3D and MILC weak scaling on Odin

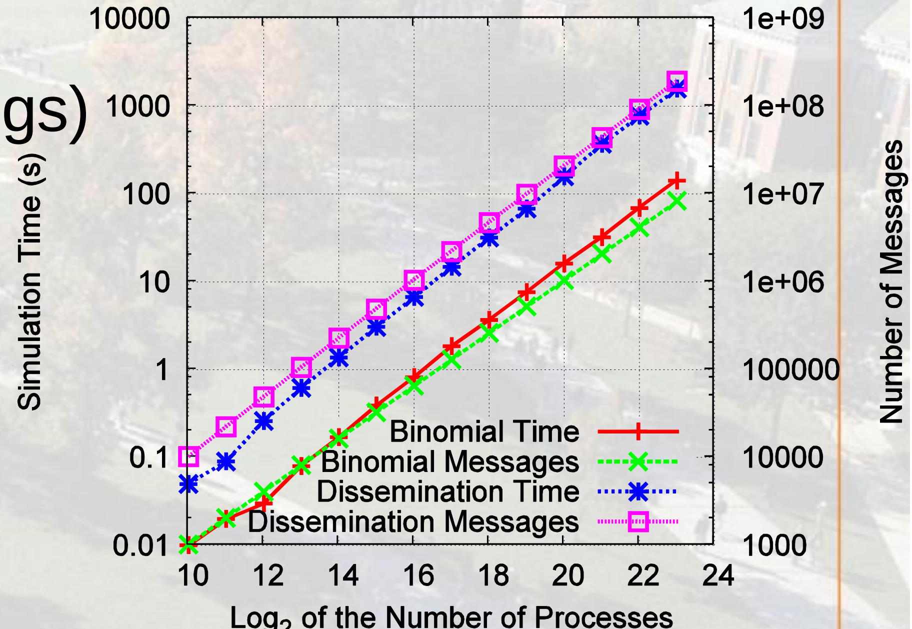




- <2% average error

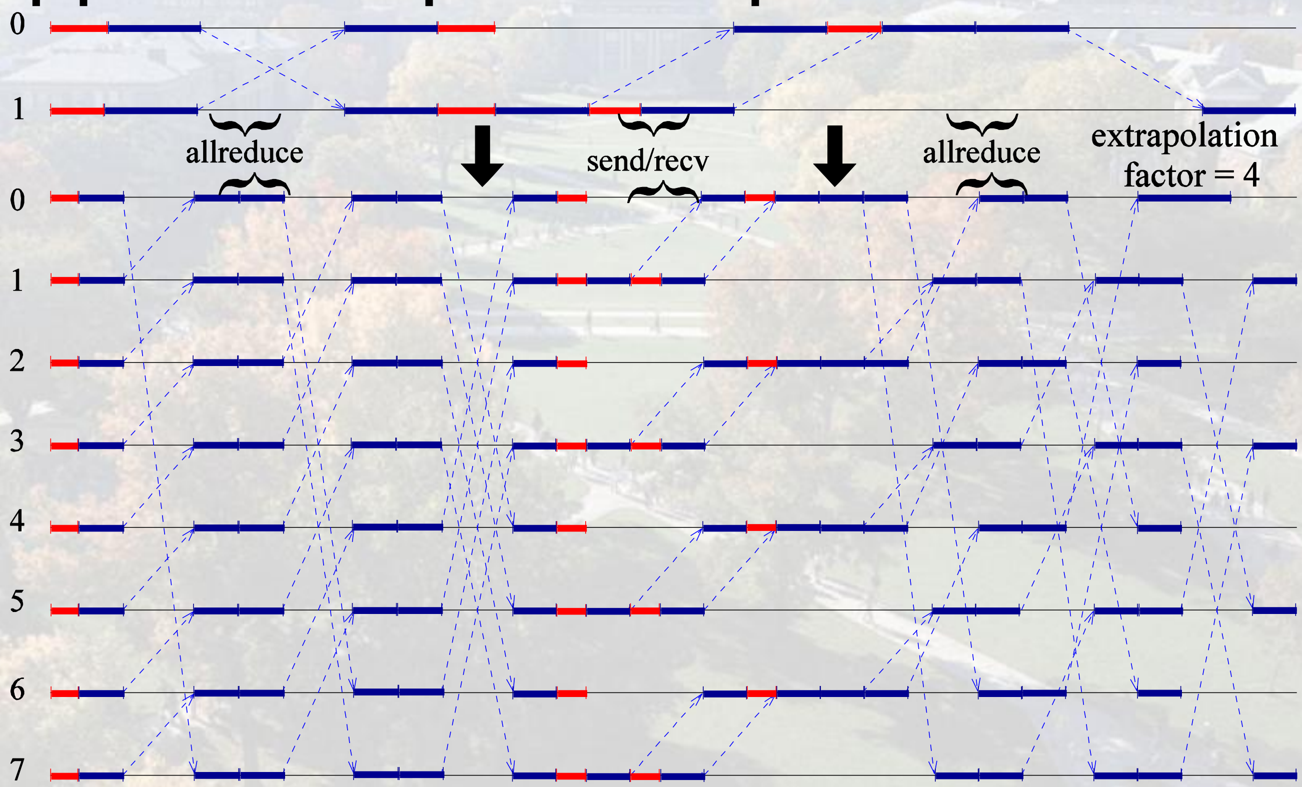
Simulation Speed

- 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

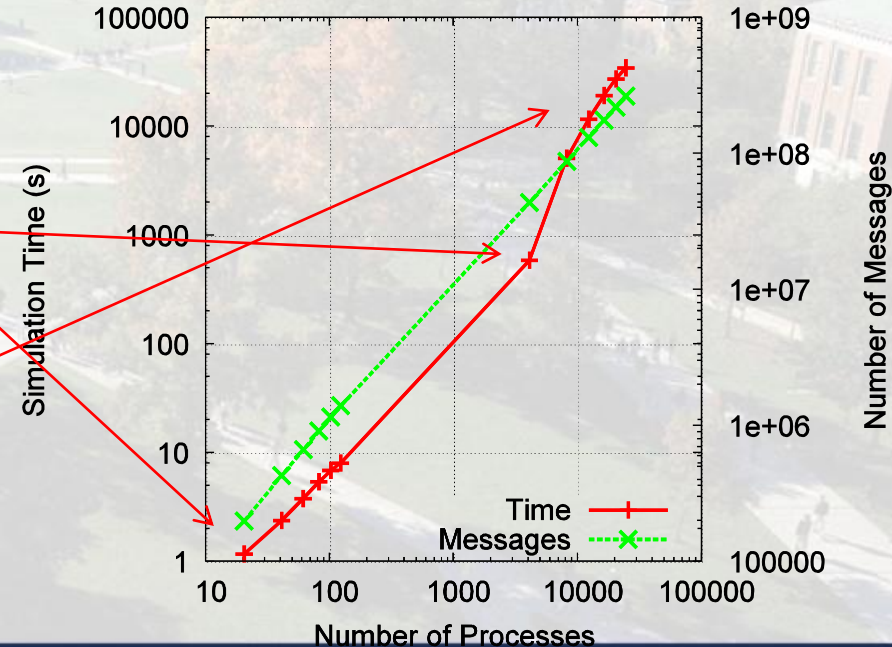
- 37.7 s Sweep3D extrapolated from 40-28k CPUs
  - 0.4 Mio msgs → 313 Mio msgs

40 CPUs – 2.43 s

4k CPUs – 10 min

28k CPUs – 9.7h (swap)

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

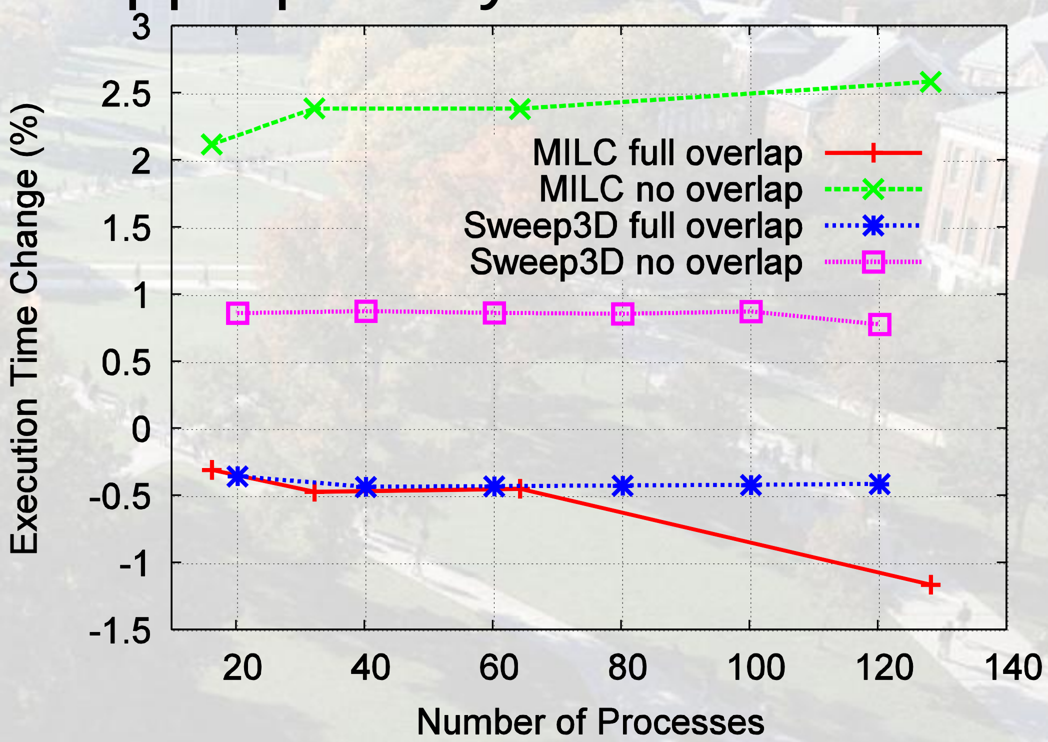


Some More Use-Cases

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

Application Overlap Potential

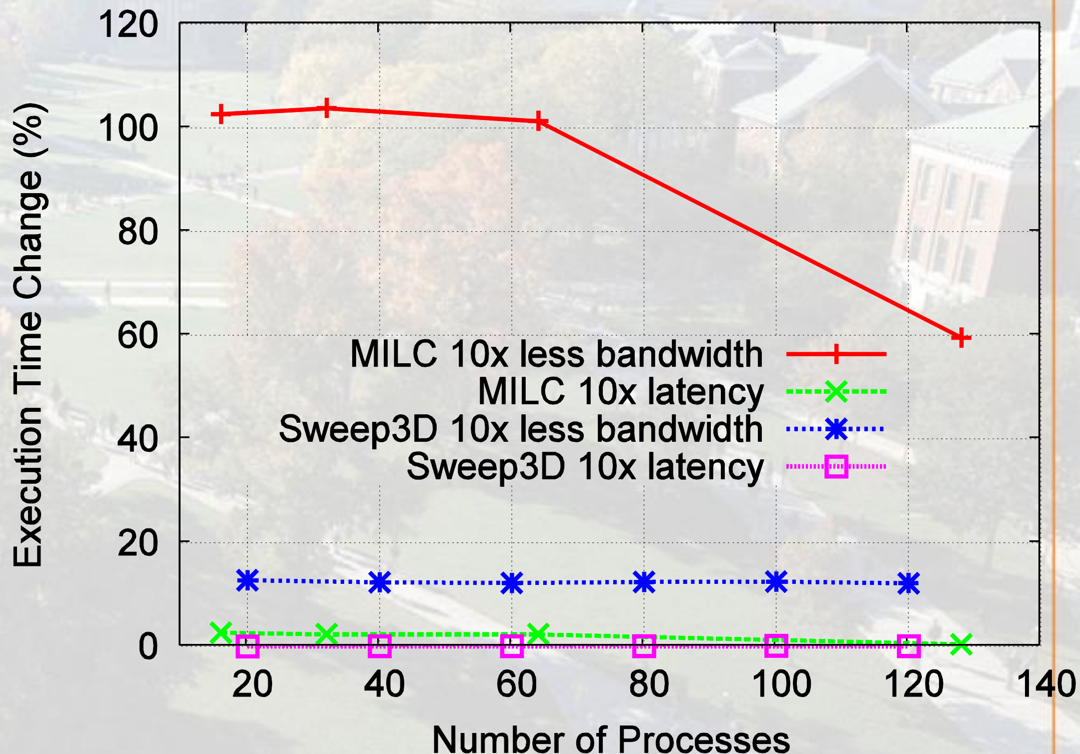
- Choose overhead appropriately:
  - full overlap:
    - o=0
    - O=0
  - no overlap:
    - o=g
    - O=G



Influence of Network Parameters

- Adjust L (latency) and G (bandwidth)

Both are much more sensitive to bandwidth than to latency!



Explaining Benchmark Problems

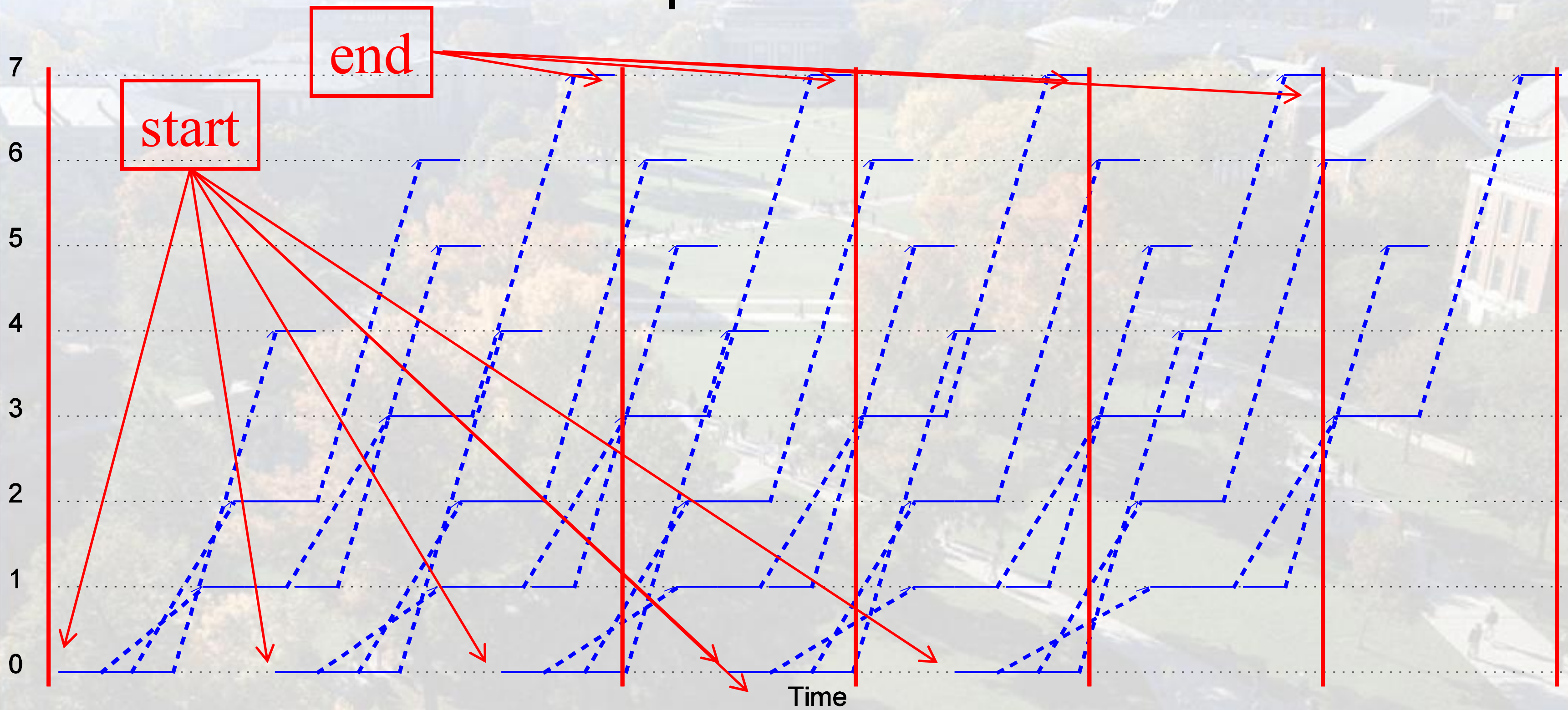
- Collective operations are often benchmarked in loops:

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

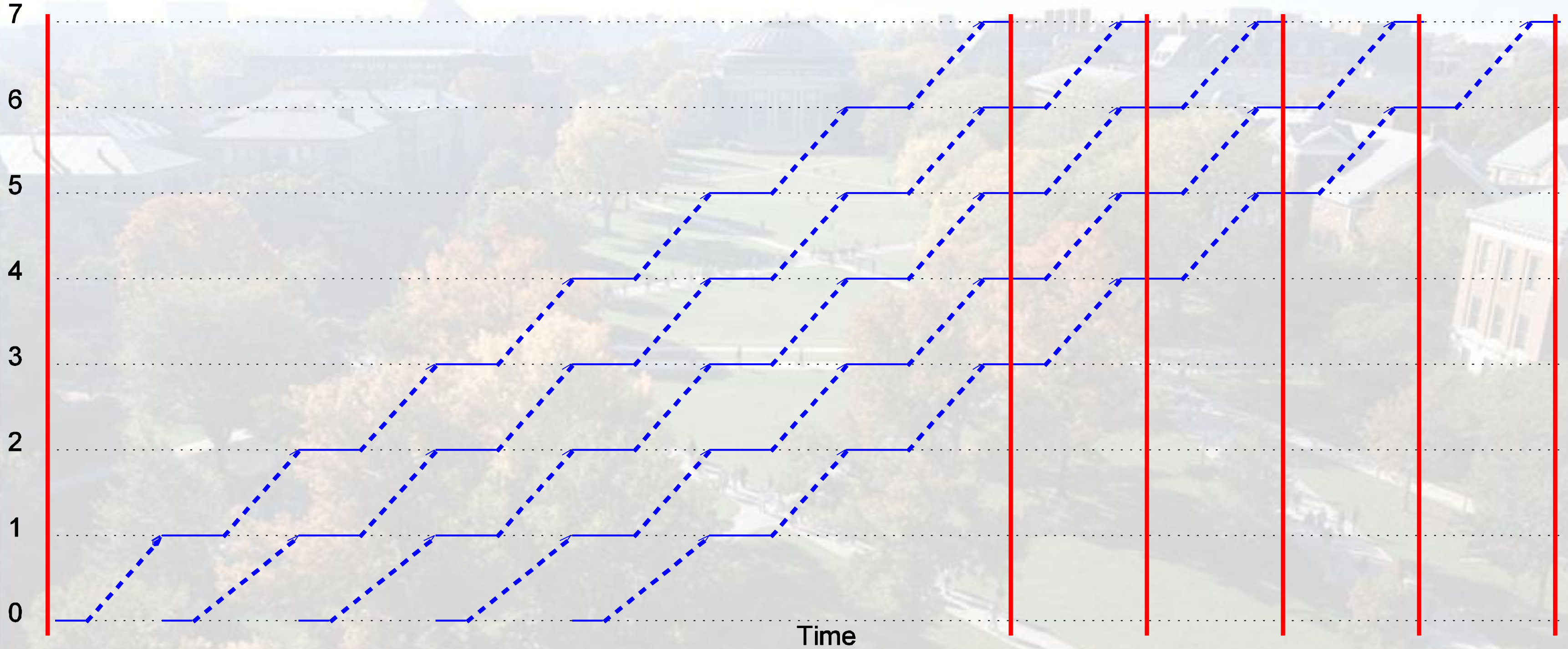
- This leads to pipelining and thus wrong benchmark results!

Pipelining? What?

Binomial tree with 8 processes and 5 bcasts:



Linear broadcast algorithm!



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

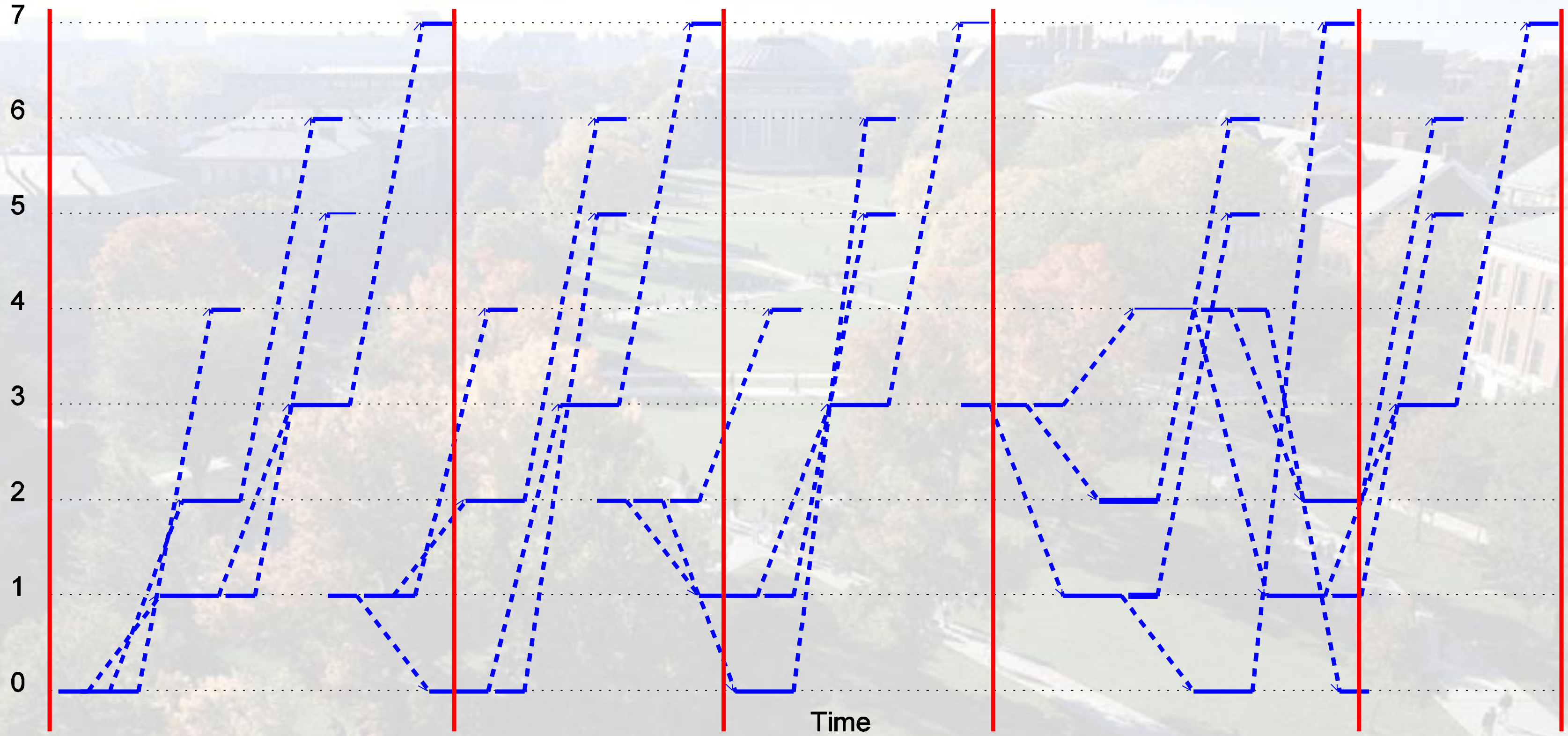
Root-rotation! The solution!

- 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
```

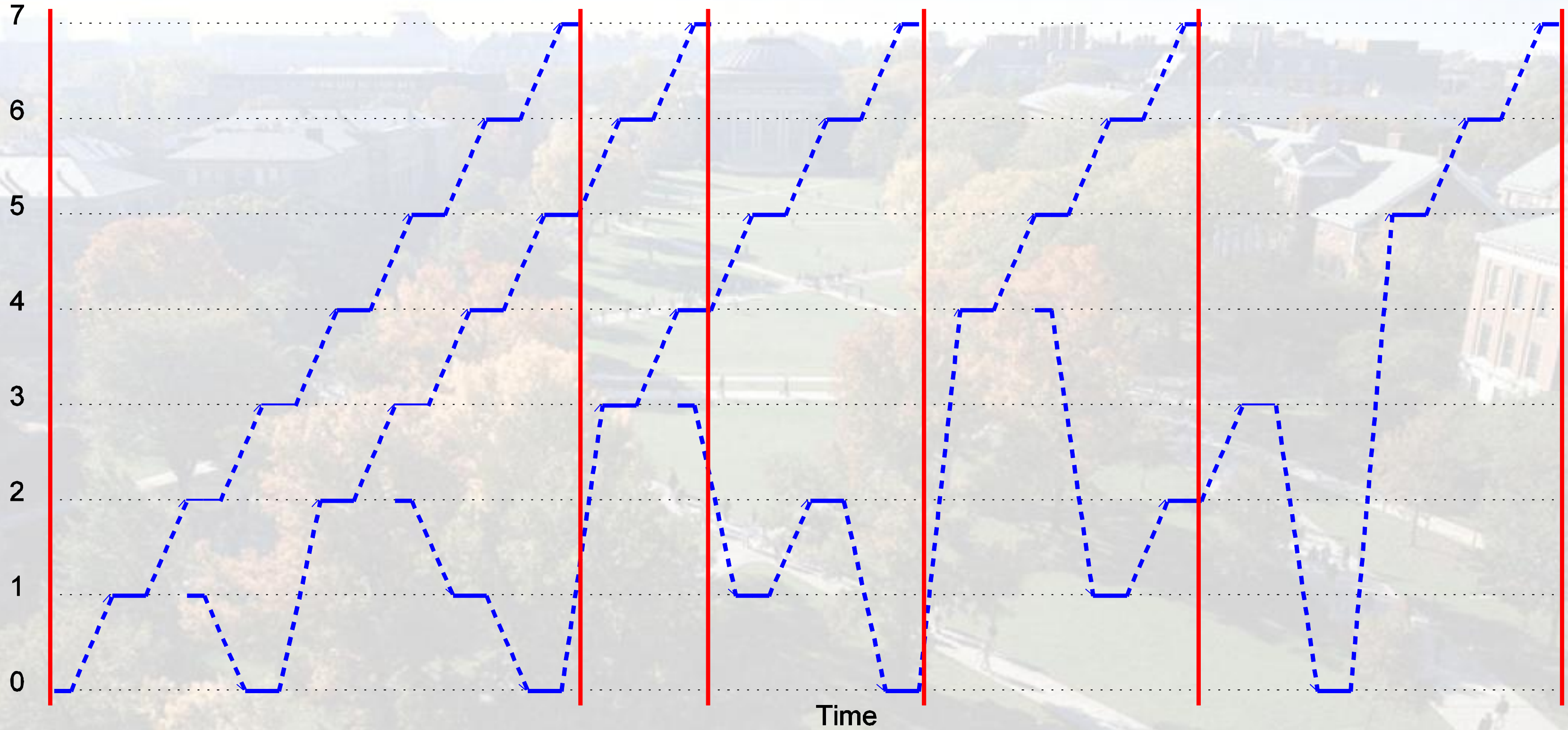
- Let's simulate …

D’oh!



- But the linear bcast will work for sure!

Well … not so much.



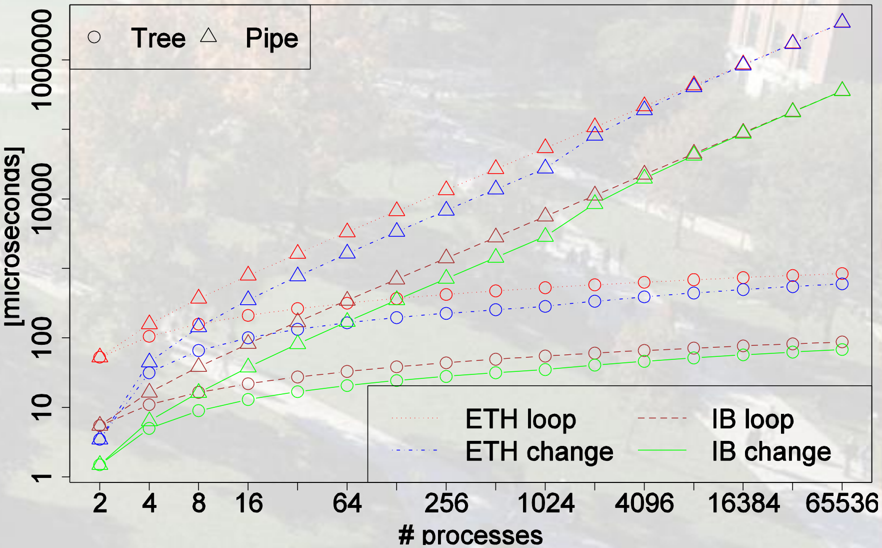
But how bad is it really? Simulation can show it!

Absolute Pipelining Error

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

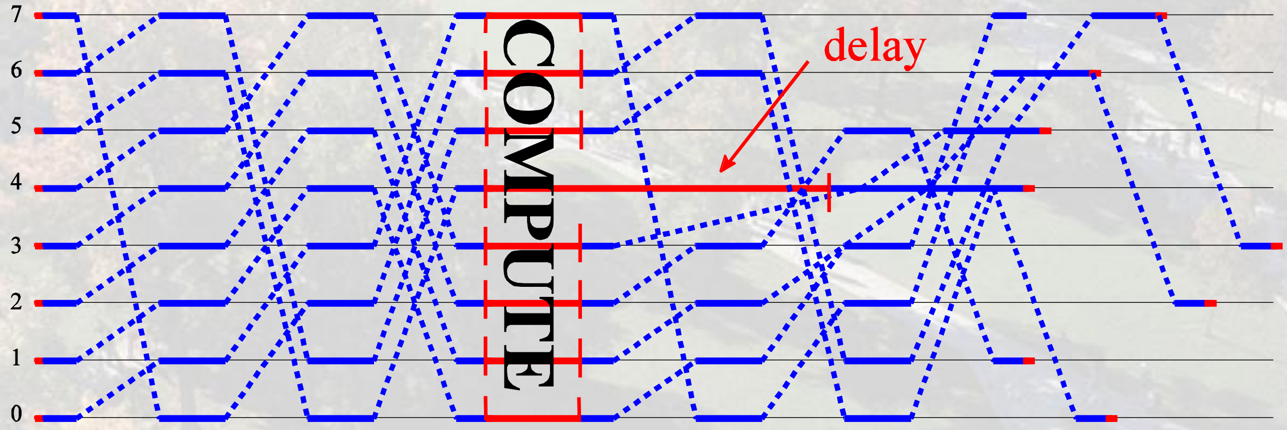
Hoefler et al.: "*LogGP in Theory and Practice*"

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

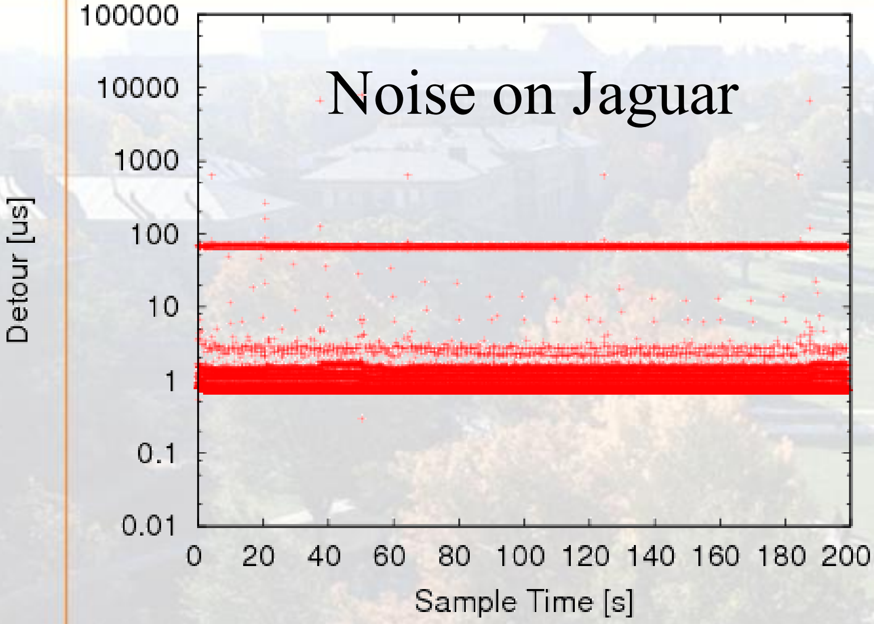


Assessing the Influence of OS Noise

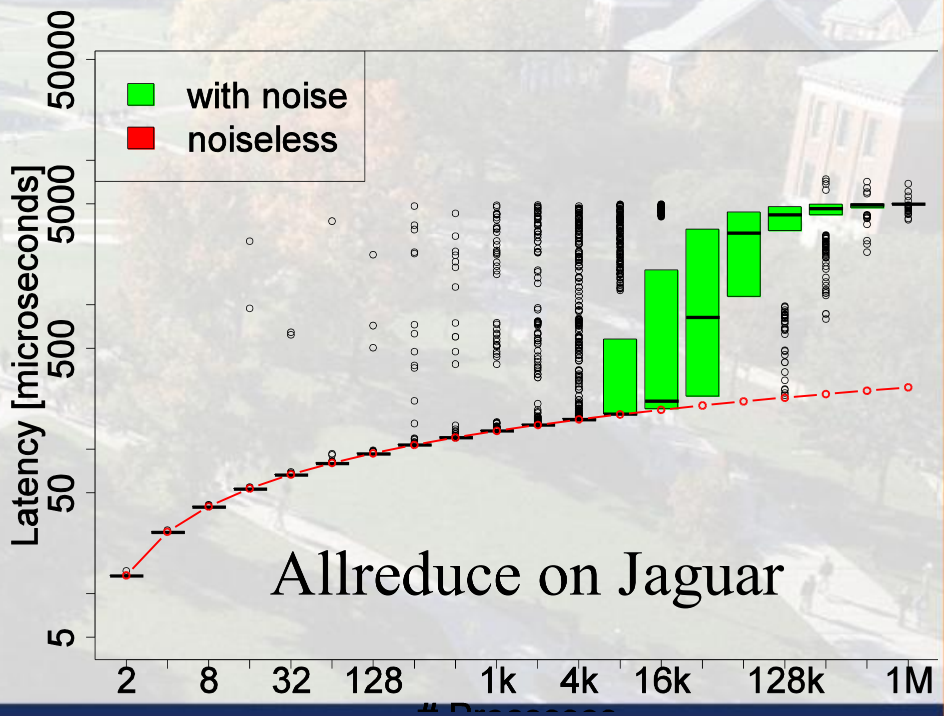
- OS Noise or Jitter is “the influence of the OS on large parallel applications”
- The noise-bottleneck limits scaling
- Consequences are non-trivial:



Influence on Collectives



Netgauge noise trace + LogGOPSim =

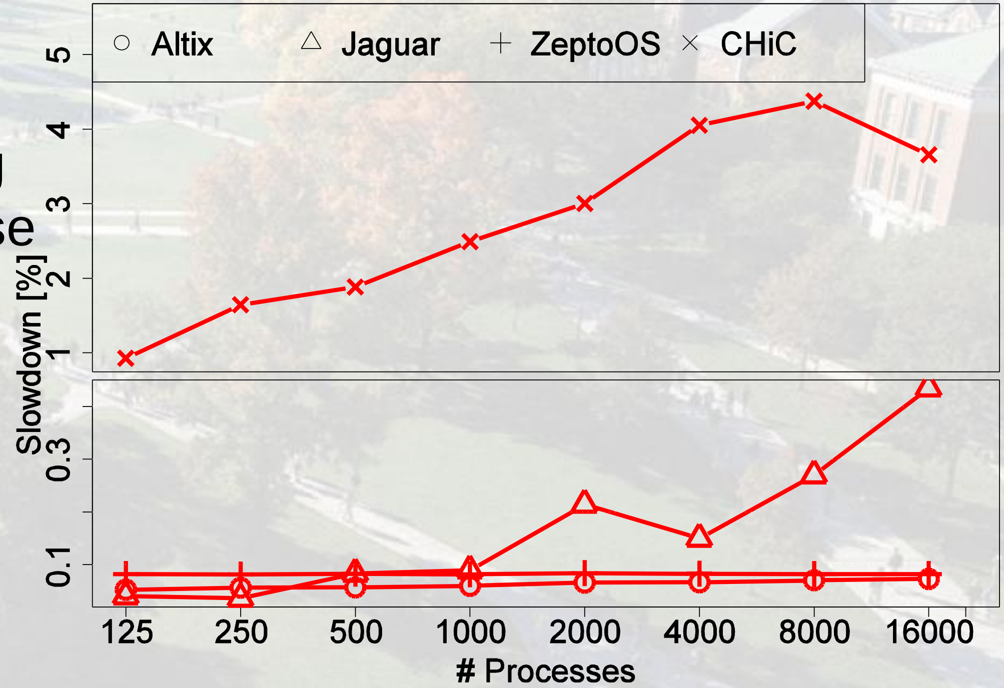


LogGOPSim supports noise injection.

OS Noise and full Applications

- AMG2006 slowed down by >4% on 8k CPUs
- Details in:

Hoefler et al. “Characterizing the Influence of System Noise to Large-Scale Applications by Simulation” Accepted at IEEE/ACM Supercomputing (SC10). Best Paper finalist.



Summary and Outlook

- LogGOPSim is a fast and scalable message passing simulator
  - supports MPI semantics but is not limited
- Simulates single collectives up to 16 Mio and application kernels up to 32k processes
  - >1 Mio events/sec
- We showed different interesting use-cases
- Future work:
  - Experience with congestion models
  - Parallelization (?)

Thanks and try it!!!

- LogGOPSim (the simulation framework)
  http://www.unixer.de/LogGOPSim
- Netgauge (measure LogGP parameters + OS Noise)
  http://www.unixer.de/Netgauge

Questions?