The Impact of Network Noise on Large-Scale Communication Performance

Torsten Hoefler, Timo Schneider, and Andrew Lumsdaine

Open Systems Lab
Indiana University
Bloomington, USA

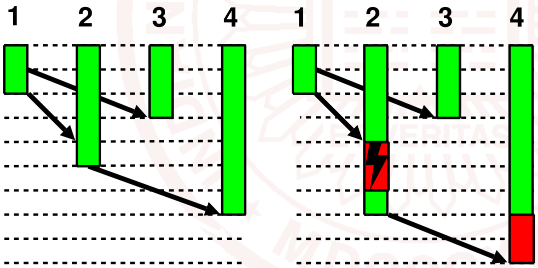
Workshop on Large-Scale Parallel Processing/IPDPS'09

Rome, Italy

May, 29th 2009

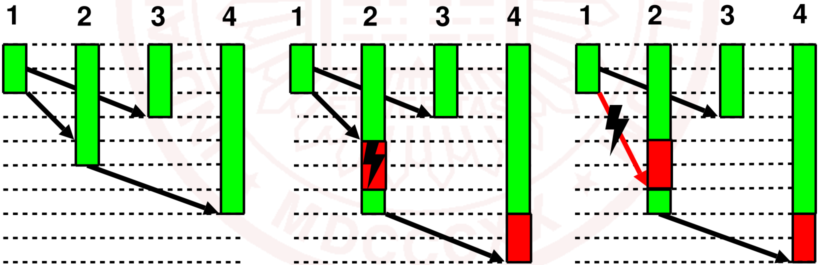
Motivation

- **operating system noise** is a known phenomenon
- local interruptions by daemons, interrupts, ...
- not problematic (<2%) for serial applications
- noise propagation is problematic
- can lower application performance significantly
- pure system issues, often “simple” to solve



Motivation

- effects in the network can cause similar behavior
- $\Rightarrow$ **network noise** (net noise)
- management, filesystem, other application, ... traffic
- such congestion causes delays
- delays optimized communication patterns (collectives)
- propagation can lead to delays
- applications interfering with themselves is **not** net noise!



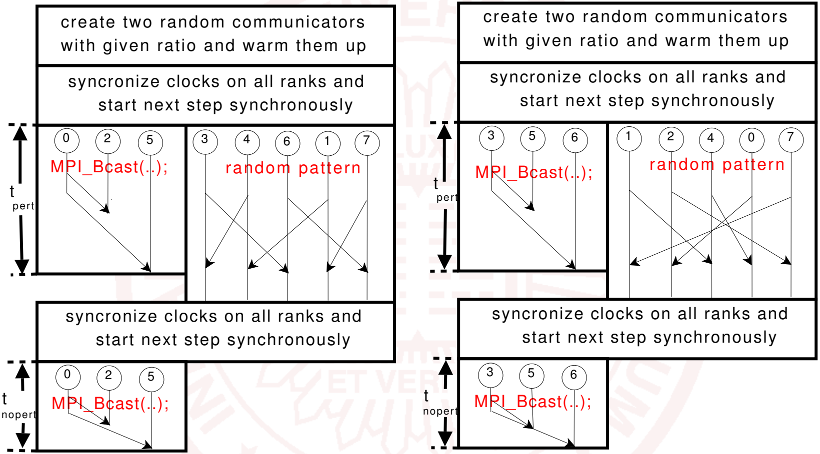
Our Approach

- OS noise is modelled with statistical or signal processing methods
- network noise depends on:
  - topology and routing
  - network technology, buffer policies, sizes etc.
  - number of PEs per endpoint (multicore)
  - communication pattern of all endpoints
- $\Rightarrow$ not as easy to model
- approach: benchmark + simulation

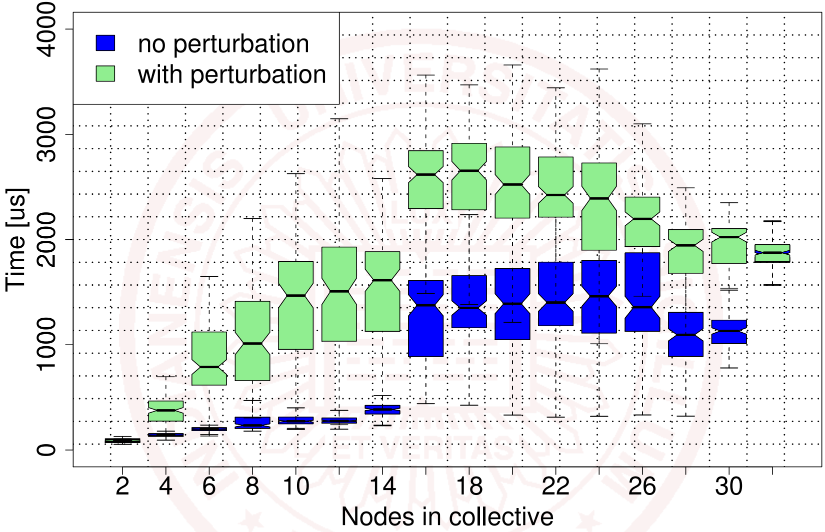
Target Architecture

- complex topologies
  - network noise can easily be avoided in tori/hypercubes
  - (make sure all allocations are convex sets)
  - other topologies (fat tree, Kautz) are not as simple
  - we focus on fat trees
- random application/application interaction
  - most common
  - also models random filesystem traffic
- collective communication patterns (including stencil)
  - most common in HPC scenarios

Benchmark Method

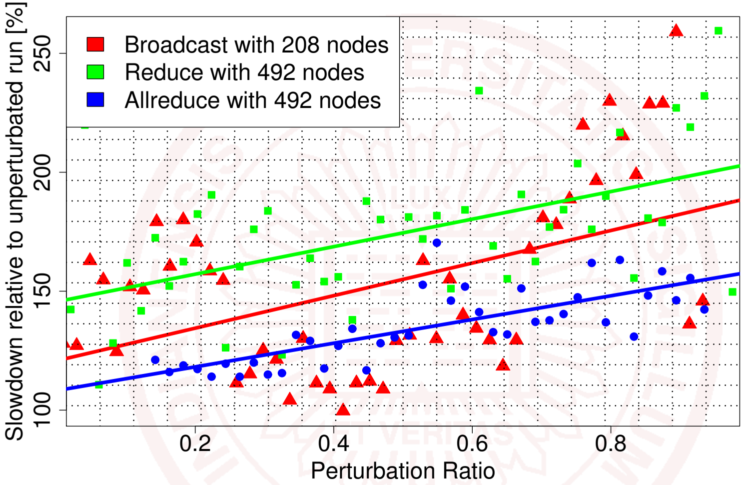


Benchmark Results



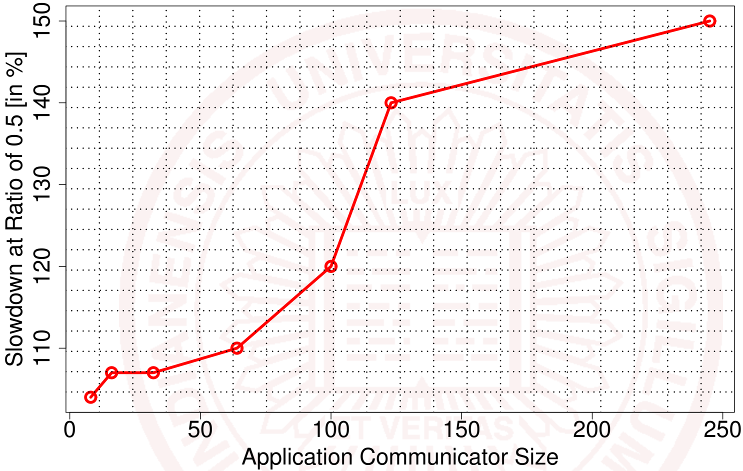
- boxplot, 32 nodes, MPI_Allreduce (single MPI_DOUBLE)
- Open MPI 1.2.8, SDR/IB, 566 node fat tree, FBB

Benchmark Results



- different collectives, 128 measurements, average plotted
- very high variance (only 128 samples, background load)

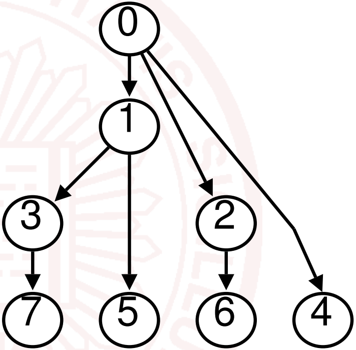
Benchmark Results



- fixed perturbation ratio (0.5)
- slowdown with increasing communicator size

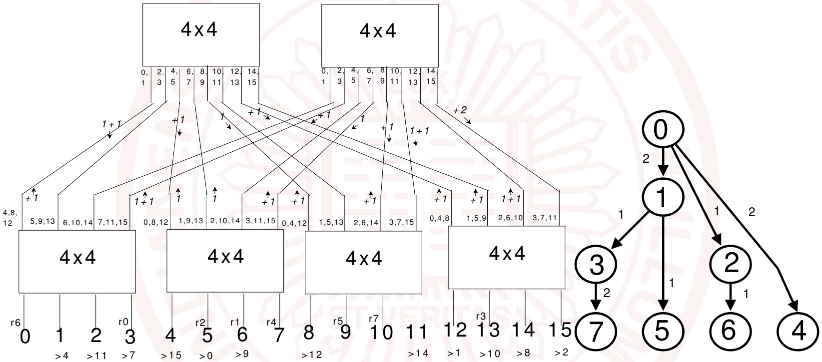
Simulation Methodology

- needs to consider topology and routing
- use IB as a model
- simple linear congestion model
- we model collective operations as a set of dependencies
- (collective) level-wise simulation



Simulation Methodology

- route every logical link through the network
- record congestion on edges

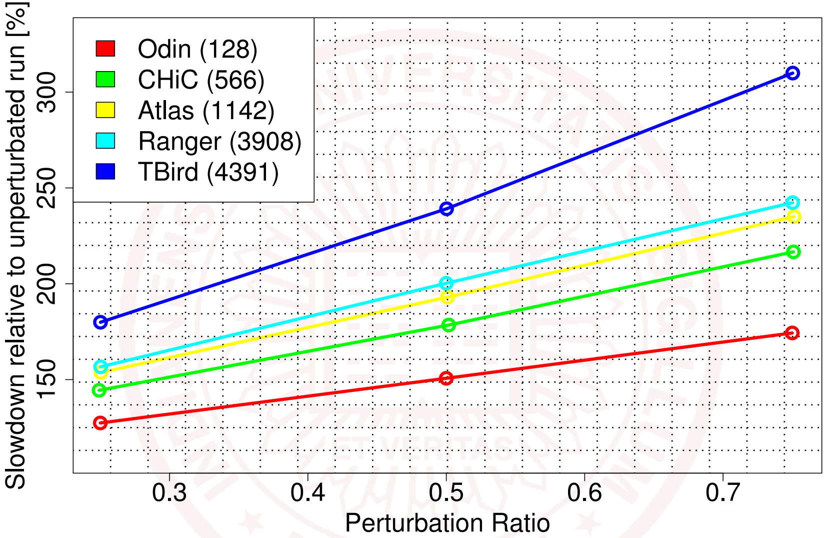


- annotate collective graph with maximum path congestion
- longest path from any root node is reported

Simulation Systems

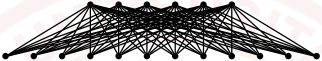
- real-world system inputs (IB network maps)
- Odin @ IU (128 nodes, FBB fat tree)
- CHiC @ TUC (566 nodes, FBB fat tree)
- Atlas @ LLNL (1142 nodes, FBB fat tree)
- Ranger @ TACC (3908 nodes, FBB fat tree)
- TBird @ SNL (4391 nodes, 1/2 BB fat tree)
- $\Rightarrow$ your system? Please give us the maps!

Simulation Results

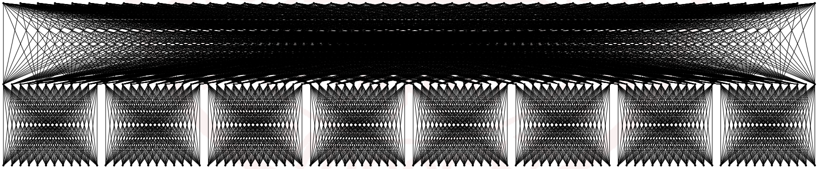


- binomial tree pattern (small message Bcast, Reduce)
- CHiC results reflect microbenchmark accurately!

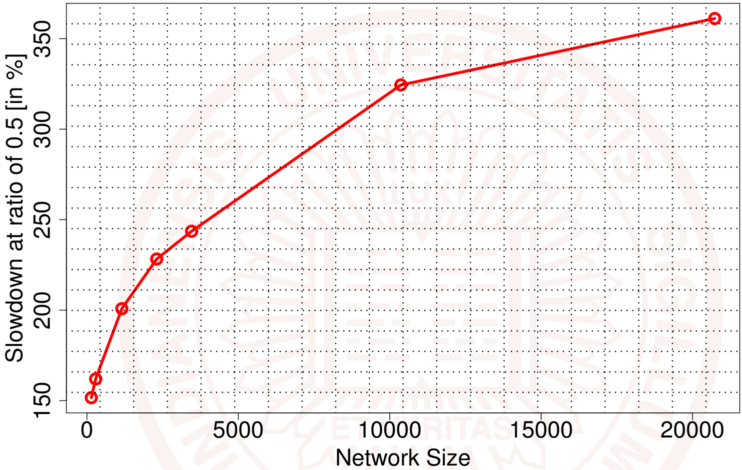
Real Large-Scale Simulation Results



- simulated large exteded generalized fat trees (XGFT)
- 24 port crossbars, full bisection bandwidth
- fat tree optimized routing (OpenSM)
- 144 nodes (one level) to 20,736 nodes (three levels)
- above: 144 nodes, below: 1152 nodes



Simulation Results



- perturbation ratio 0.5, tree pattern
- logarithmic shape reflects CHiC benchmarks!

Conclusions and Future Works

Conclusions

- network noise must be considered
- significant impact, similar to OS noise
- no known real-world analyses yet
- network topology and routing are very important

Future Work

- good process-to-node mapping could reduce problems
- topology-aware communication algorithms
- extend analysis to real applications (profiling, tracing)
- analyze several network topologies and workarounds

Thanks for your attention!

Questions?

Download the (research-quality) ORCS simulator at:

`http://www.unixer.de/ORCS`