



The Impact of System Design Parameters on Application Noise Sensitivity

Kurt B. Ferreira, Ron Brightwell, and Kevin Pedretti
Scalable System Software
Sandia National Laboratories
`{kbferre, rbbrigh, ktpedre}@sandia.gov`

Patrick Bridges
University of New Mexico
Dept. of Computer Science
`bridges@cs.unm.edu`





OS “Jitter” Critical to Performance of HPC Systems

- ▶ Studied for nearly 20 years
- ▶ Key limiter of application performance on large-scale systems
- ▶ Previous work focused on noise mitigation specific an architecture, an application, certain communication patterns, etc.
- ▶ Conflicting results for certain apps/ubench on different architectures





OS “Jitter” Critical to Performance of HPC Systems

- ▶ Studied for nearly 20 years
- ▶ Key limiter of application performance on large-scale systems
- ▶ Previous work focused on noise mitigation specific an architecture, an application, certain communication patterns, etc.
- ▶ Conflicting results for certain apps/ubench on different architectures

Goal: Unify current results with a study of the influence of system features on HPC application noise sensitivity

Our Contributions

- ▶ Systematic, empirical study on the impact system design parameters have on an HPC applications sensitivity to OS noise
- ▶ Examine the impact the following system features have on an applications noise sensitivity:
 - ▶ System Balance. The ratio of peak network bandwidth to peak compute performance (bytes/FLOP)
 - ▶ Percent of “noisy” nodes. Simulate nodes running full-featured operating systems
 - ▶ Non-blocking collectives
- ▶ Show noise impact influenced by:
 - ▶ System balance ratio
 - ▶ Total number and placement of noise generating nodes
 - ▶ Non-blocking collective algorithms



Big Picture

Will Linux scale for next generation systems and if not, what needs to be changed?



A Brief Introduction to OS Noise

- ▶ Defined as any OS activity that asynchronously interrupts the progress of an application
- ▶ Examples:
 - ▶ Timer “tick”
 - ▶ Periodic kernel daemon
 - ▶ Hardware interrupt
- ▶ Duration varies from a few microseconds to a few milliseconds
- ▶ Global performance cost due to variance in time for process to participate in a collective operation





Overview of Approach

Using a previously developed, kernel-level noise injection framework, study the influence of the following on application noise sensitivity:

- ▶ System Balance
- ▶ Percent of total and placement of “noisy” nodes
- ▶ Noise resistant (non-blocking) collectives



Three Important Modeling and Simulation Workloads

- ▶ Representative Applications:
 - ▶ CTH – Shock physics code from SNL
 - ▶ SAGE – Hydrodynamics code from LANL
 - ▶ POP – Ocean circulation model from LANL
- ▶ Key applications to both DOE and DOD
- ▶ Represent a range of different computation techniques
- ▶ Frequently run on tens of thousands of nodes for weeks at a time





Hardware Platform

- ▶ Cray Red Storm located at SNL
- ▶ Cray XT3/4 Series
- ▶ Nearly 13,000 Nodes
 - ▶ ~9000+ 2.2GHz quad-core Opterons
 - ▶ ~3000+ 2.4GHz dual-core Opterons
 - ▶ 2GB per core
 - ▶ SeaStar NIC: 3GB+ peak unidirectional link BW
- ▶ Ideal for study as one of the more balanced modern systems



Our Metric: The Accumulation of Noise

- ▶ Measure the global *accumulation* of noise
 - ▶ Measured percent slowdown minus percent local noise injected
 - ▶ If negative, absolute value is injected noise *absorbed*

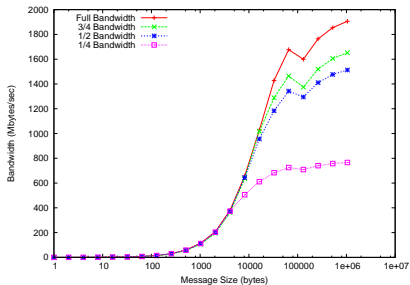




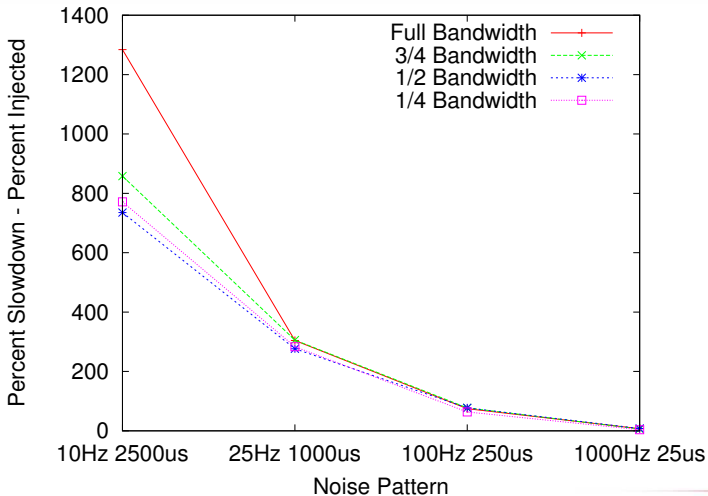
System Balance

Approach – System Balance

- ▶ Use noise signatures that represent 2.5% net processor interference
- ▶ Modify balance by using existing hardware network degradation modes.
- ▶ Available modes:
 - ▶ Full Bandwidth
 - ▶ $\frac{3}{4}$ bandwidth – ASC Purple
 - ▶ $\frac{1}{2}$ bandwidth – single-core IB cluster
 - ▶ $\frac{1}{4}$ bandwidth – Thunderbird @ SNL

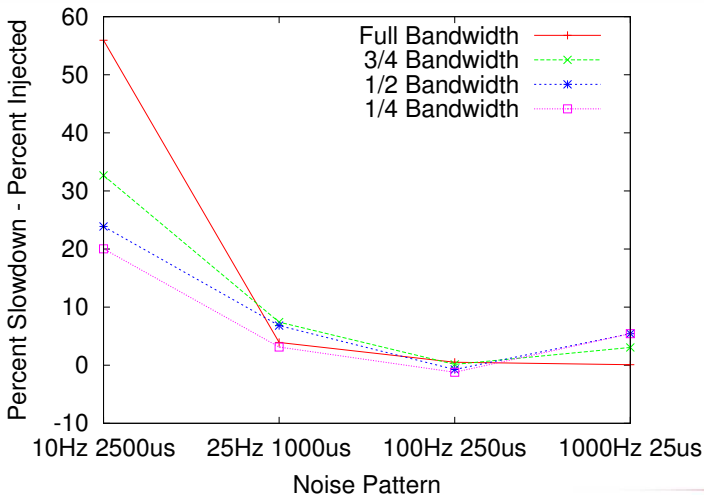


More Compute Performance == Less Noise Impact




POP on 2500 nodes

More Compute Performance == Less Noise Impact



SAGE on 3360 nodes

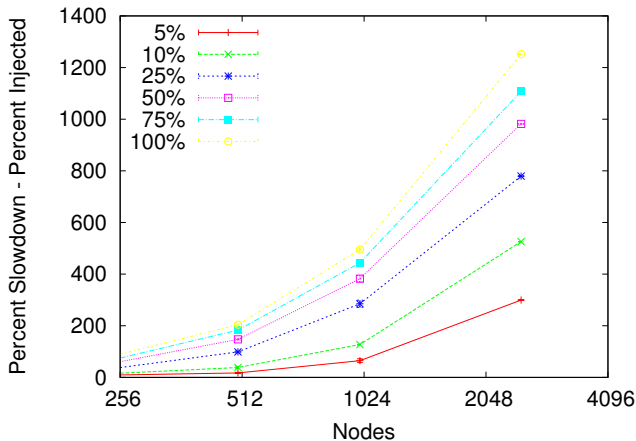


Isolating Noise to a Subset of Nodes

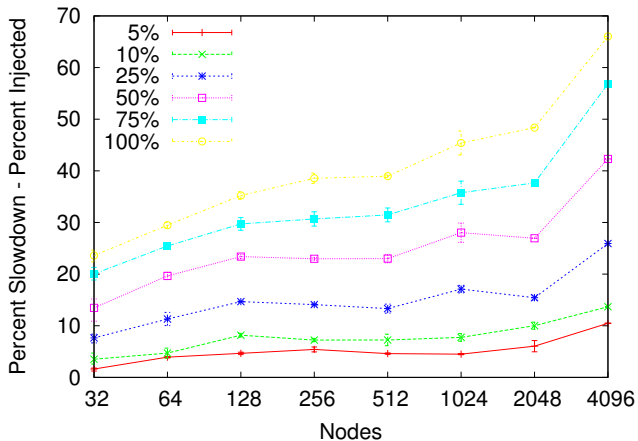
Approach – Noisy nodes

- ▶ Use a previously developed, kernel-level noise injection framework
- ▶ Use noise signatures that represent 2.5% net processor interference
- ▶ Noise parameters: frequency, duration, which nodes generating noise, and randomization
- ▶ Location of noise nodes
 - ▶ Randomly using Fisher-Yates permutation
 - ▶ Sequentially based on MPI rank

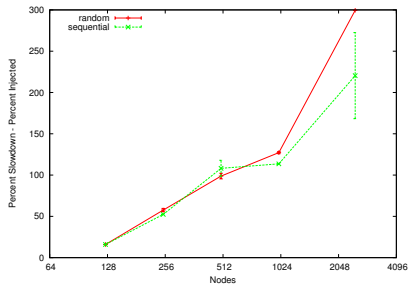
POP – Few “Noisy” Nodes Still Slowdown the System



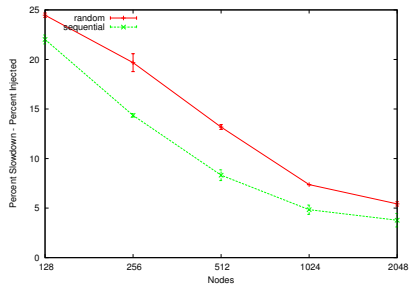
Yet Isolation Effective for SAGE and CTH



Placement of Noisy Nodes Matters

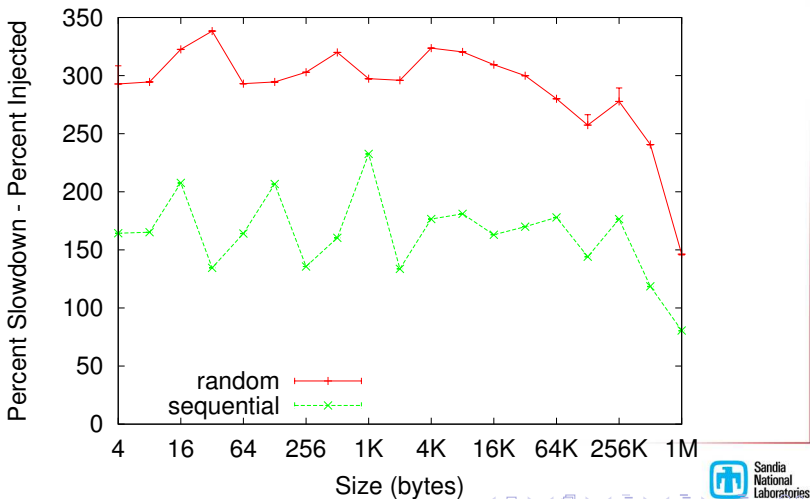


POP with 125 noisy nodes



SAGE with 128 noisy nodes

Difference due to Location of Noisy Nodes in Allreduce Tree



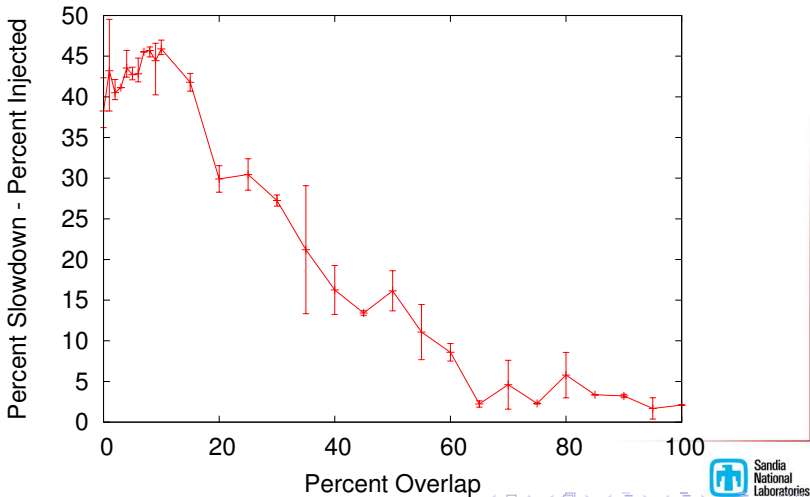


Noise Resilient Collectives

Approach – Noise Resilient Collectives

- ▶ Currently exists in a few machines (BlueGene) and under consideration for inclusion for MPI 3
- ▶ Implemented a non-blocking `MPI_Allreduce()` for Red Storm
- ▶ Modifying our applications to use NBC complex so implemented a bulk-synchronous microbenchmark which allows you to specify compute phase time before allreduce step and overlap of compute and communication
- ▶ Synchronization rate set to that measured for SAGE and POP

Nonblocking Techniques Effective in Mitigating Impact of Jitter





Summary

- ▶ We examine the impact system balance, number of noisy nodes and their placement, non-blocking collectives have on HPC application noise sensitivity
- ▶ Show balance ratios shifted towards excessive computation, noise isolation, and non-blocking collective are effective in mitigating OS noise for many HPC applications
- ▶ Yet for some applications (POP), noise on a relatively small percentage of nodes – as little as 5% – can have a significant impact on performance
- ▶ This knowledge is important in designing future-generation, extreme-scale systems



Acknowledgements

- ▶ Sandia: Courtenay Vaughan, Sue Kelly, Bob Ballance, the Cray/Red Storm support team

This work was supported by the US Department of Energy's (DOE) Office of Advanced Scientific Computing Research and the FASTOS program

