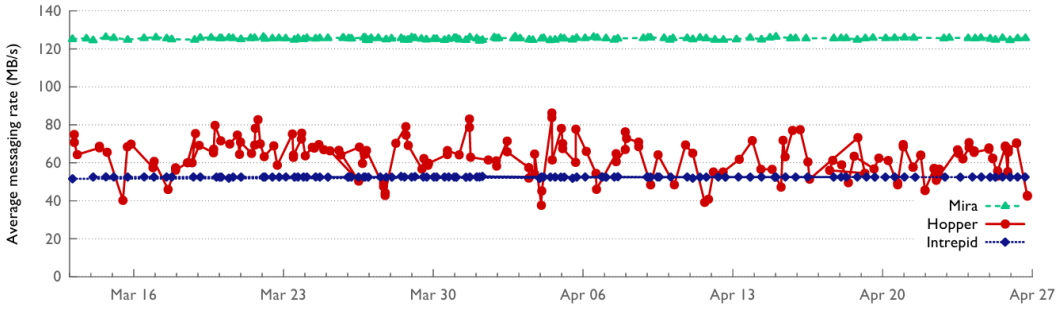
There Goes the Neighborhood: Performance Degradation due to Nearby Jobs

Sanika Prashant Deshmukh

CS 575 – Parallel Programming

Spring 2025

General Theme:



The authors explore how a parallel application's performance can fluctuate significantly, not due to issues within the code or hardware faults, but because of external interference from other jobs running simultaneously on the same system.

Using a communication-heavy scientific simulation code called pF3D, the researchers ran experiments on several large supercomputing systems. They observed that on Cray XE6 systems, the performance of identical runs varied considerably sometimes running up to 41% slower or 28% faster than average. In contrast, IBM Blue Gene systems showed consistent performance with minimal variation.

The paper examines three potential causes for this variability:

1. Operating system (OS) noise or jitter – background tasks affecting core availability.
2. Shape of the job allocation – how scattered or compact the assigned nodes are on the network.
3. Contention from other nearby jobs – jobs that share network links and cause congestion.

Their analysis shows that network contention from neighbouring jobs is the dominant factor affecting performance, particularly when the surrounding jobs are also communication-intensive.

Meaning of the Title

“There Goes the Neighbourhood” is a metaphor referring to the idea that a job’s performance can suffer when "bad neighbors" other jobs using shared resources are placed nearby. Just like a quiet, well-maintained neighborhood may decline if disruptive neighbors move in, a well-optimized HPC job can experience degraded performance if nearby jobs overload shared network infrastructure. Thus, the title reflects how the presence of surrounding jobs can negatively affect performance, even when nothing in the job itself has changed.

Authors and Backgrounds

1. Dr. Abhinav Bhatele

- **Affiliation**: Lawrence Livermore National Laboratory (LLNL), Livermore, California, USA
- **Current Position**: Computer Scientist
- **Background**:
  Dr. Bhatele is an accomplished computer scientist with a strong focus on high-performance computing (HPC), parallel applications, and communication modeling. He holds a Ph.D. in Computer Science from the University of Illinois Urbana-Champaign. His work includes optimizing application performance on supercomputers, developing visualization tools, and studying network and system-level behaviors that impact application efficiency. He has contributed extensively to research on MPI-based communication, job placement, and performance bottlenecks in HPC systems.

2. Dr. Kathryn Mohror

- **Affiliation**: Lawrence Livermore National Laboratory (LLNL), Livermore, California, USA

- **Current Position**: Director, Center for Applied Scientific Computing (CASC) at LLNL
- **Background**:
  Dr. Mohror is an expert in scalable computing and fault tolerance in large-scale systems. She completed her Ph.D. in Computer Science at Portland State University and has been at LLNL since 2012. Her research involves improving the reliability and efficiency of parallel applications, with key contributions in parallel I/O and checkpoint/restart libraries such as SCR. She currently leads efforts in advanced computing systems and is recognized for her leadership in enabling high-performing, resilient software systems for scientific applications.

3. Dr. Steven H. Langer

- **Affiliation**: Lawrence Livermore National Laboratory (LLNL), Livermore, California, USA
- **Current Position**: Physicist and Scientific Software Developer
- **Background**:
  Dr. Langer is a senior physicist at LLNL with deep expertise in laser-plasma interactions and simulation software. He has been heavily involved in developing the pF3D code used in the paper for experiments which is a tool for modeling laser-plasma behavior in support of fusion research at the National Ignition Facility (NIF). His work bridges scientific computing and physics, enabling more accurate and scalable simulations of complex physical systems.

Experiments Conducted

In this paper, the researchers aimed to investigate performance variability in high-performance computing systems using a parallel application called pF3D. This program is designed to simulate laser-plasma interactions and is ideal for performance analysis due to its heavy communication demands and balanced computational workload. The study investigated how external conditions such as hardware architecture and resource distribution impact the repeatability of application performance. Experiments were carried

out on three high-performance computing architectures: IBM Blue Gene/P, IBM Blue Gene/Q, and Cray XE6. These systems were deployed across four U.S. Department of Energy laboratories: Los Alamos, Lawrence Livermore, Lawrence Berkeley, and Argonne. Five supercomputers Cielo, Dawn, Hopper, Intrepid, and Mira were tested during typical usage hours, meaning the systems were running other jobs simultaneously, providing a realistic multi-user environment for performance analysis. This setup provided realistic and practical insight into performance fluctuations.

One of the main findings was that Cray XE6 systems showed considerable variability in execution time for the same job. The performance ranged from 28% faster to 41% slower than the average. In contrast, Blue Gene systems demonstrated consistent runtime with minimal variation. This difference was primarily attributed to how the systems managed resource allocation. Blue Gene machines assigned each job a dedicated and contiguous section of the network, isolating it from other jobs and reducing interference. Cray systems, however, allocated nodes in a fragmented manner, leading to potential overlap and contention on network links shared between jobs.

To identify the reasons for performance variability, the researchers focused on three potential sources: operating system jitter, irregular node allocation shapes, and inter-job interference via shared network links. Their aim was not only to understand the causes but also to find ways to mitigate them in order to enhance overall system throughput and reduce energy costs.

The hardware and interconnect structures of the machines varied notably. While all systems employed a form of 3D torus topology, the internal configuration differed. For example, Blue Gene/P provided each node with a dedicated router and multiple directional links. Cray XE6, on the other hand, paired two nodes per router and had fewer links in some directions. Blue Gene/Q implemented an even more complex 5D torus. Additionally, Cray systems had a higher total link bandwidth but a lower injection bandwidth, which could act as a bottleneck in real-world applications.

Another major distinction was in node allocation policies. Blue Gene systems ensured job isolation by assigning exclusive partitions, whereas Cray systems used arbitrary node placements, sometimes resulting in gaps or fragmented patterns. This fragmentation increased the likelihood of messages crossing through busy areas of the network, amplifying contention and variability.

The researchers also compared systems with similar hardware but different usage models. For instance, Cielo and Hopper had almost identical hardware, but their operational goals differed. Cielo primarily handled large-scale jobs and was weekly regularly rebooted, resulting in cleaner node assignments. Hopper, designed for mixed workloads, experienced more fragmentation due to less frequent reboots and smaller job scheduling.

The application used in the study, pF3D, simulates wave propagation and energy a 3D Cartesian grid and extensive MPI-based communication. The application relies heavily on MPI_Alltoall,MPI_Send, and MPI_Recv routines for moving data across nodes, particularly in solving 2D FFTs in the x-y planes and sending data in the z-direction. Because of its intensive and balanced workload, pF3D was a suitable candidate for detecting even small amounts of external interference.

In summary, the experimental setup revealed that network topology, node allocation strategy, and system usage patterns significantly impact the reproducibility of performance in HPC systems. Cray systems, while powerful, are more prone to variability due to their allocation and communication patterns, whereas Blue Gene systems offered more predictable performance due to their structured and isolated node assignments.

Conclusions

This study explored the causes of performance variability in HPC systems by analyzing pF3D runs across IBM Blue Gene and Cray XE6 architectures. While Blue Gene systems showed highly consistent runtimes due to their dedicated, contiguous node allocations, Cray XE6 systems exhibited performance swings ranging from 28% faster to 41% slower than the average. The key driver of this variability was not OS jitter or hop count, but

interference from concurrently running jobs amplified by Cray's fragmented node allocations and shared network links.

Detailed instrumentation and visual analysis revealed that even jobs with identical allocations suffered performance drops when surrounded by communication-heavy neighbors. Interestingly, systems with similar hardware like Cielo and Hopper showed different variability levels due to differing operational policies and scheduling behavior. Cray's higher injection bandwidth was not enough to compensate for topological contention.

This study's unique contribution lies in correlating message-passing rates with actual network placements and job interference. The findings emphasize the need for topology-aware, interference-minimizing schedulers to improve performance predictability and efficiency in modern HPC environments.

New Insights Gained

This paper gave me new perspective on performance issues in supercomputers. I used to think that if a program is optimized and the system is powerful, it should perform consistently. But the study showed that's not always the case. A job's speed can vary a lot depending on what else is running on the system at the same time even if the job itself hasn't changed.

One of the biggest surprises was learning that things like operating system noise or message distance aren't the main reasons for slowdowns. Instead, the biggest factor was network congestion caused by nearby jobs, especially when those jobs were also heavy on communication. It's like trying to get somewhere quickly but getting stuck because others are using the same road.

I also didn't realize how much node allocation policies could affect performance. On Blue Gene systems, where jobs are isolated in clean, compact blocks, performance was steady. But on Cray systems, where jobs are scattered, there was a lot more variability due to

overlapping network use. That made me understand that how jobs are placed on a machine matters just as much as how they're written**.**

In the end, the paper made it clear that in shared computing environments, job performance depends on the larger system activity. It's not just about your code it's also about your neighbors. This really highlighted the importance of smarter scheduling and system-level awareness to keep things efficient and fair.

Flaws or short sightedness

Upon careful review, I did not identify any significant flaws or short-sightedness in the paper's methodology. The experiments were planned carefully, the data was analyzed properly, and the results made sense based on the evidence. Overall, the paper was clear and well-supported.

Future Work

If I were one of the researchers, the next step would be to develop an automated monitoring system that collects real-time data about job placement, network traffic, and system performance across all running applications. This would make it easier to identify when and why slowdowns happen.

I would also explore smarter job scheduling methods that try to group jobs in ways that reduce network interference. For example, assigning jobs that use the network heavily to separate areas could avoid bottlenecks.

Additionally, it would be useful to repeat this study using newer HPC systems or cloud-based HPC platforms to see if the same problems still exist with modern technologies. Finally, I would consider testing with more applications like pF3D to confirm that the patterns found in this study are common across different types of workloads.