Understanding Simultaneous Impact of Network QoS and Power on HPC Application Performance

Tapasya Patki*, Emre Ates†, Ayse Coskun†, Jayaraman J. Thiagarajan*
*Lawrence Livermore National Laboratory
†Boston University

***Abstract*—With the growing complexity of high performance computing (HPC) systems, application performance variation has increased enough to disrupt the overall throughput of the systems. Such performance variation is expected to worsen in the future, when job schedulers will have to manage *flow* resources such as power, network and I/O in addition to traditional resources such as nodes and memory. In this work, we study the simultaneous impact of inter-job interference, Infiniband service levels, and power capping on different applications in a controlled experimental setup, with the goal of understanding the range of performance variation as well as potential mitigation strategies.**

I. Introduction

Performance variation and unpredictability in modern supercomputers are growing concerns. On current HPC systems, user applications already experience about 20% run-to-run variation with the exact same input configuration [3]. Such variation is typically attributed to *interference from neighboring jobs* or to manufacturing differences on power-limited systems [5]. Limited understanding of the sources of run-to-run variation and the range of variation can reduce the overall efficiency of the system. As we venture toward exascale, scientific reproducibility will worsen if system software (such as job scheduler) does not adapt to the changing landscape of managing *flow resources* such as network bandwidth and power *simultaneously*.

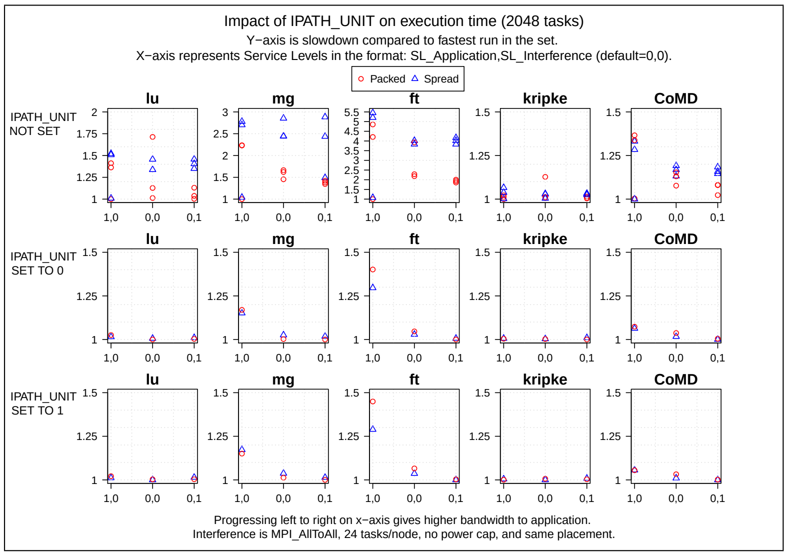
Most modern schedulers do not consider flow resources when making allocation decisions, even though it has been shown that there is tremendous scope for improving throughput and utilization [9], [10]. One of the primary reasons for this is the lack of understanding of application performance variation and its sources. For system schedulers to be able to boost throughput, both application performance and the associated range of run-to-run variation need to be predictable to some degree. Our goal is to construct a dataset of application performance subject to changing parameters such as power, network bandwidth, and rank-to-node mapping/placement. Such a dataset can be used to create effective performance prediction models, which can then be used for future scheduling research. Creating this dataset is extremely challenging because scheduling environments are dynamic in nature. Therefore, in this work, we run simpler *control jobs* that can be well understood when subject to different parameters instead of complex applications.

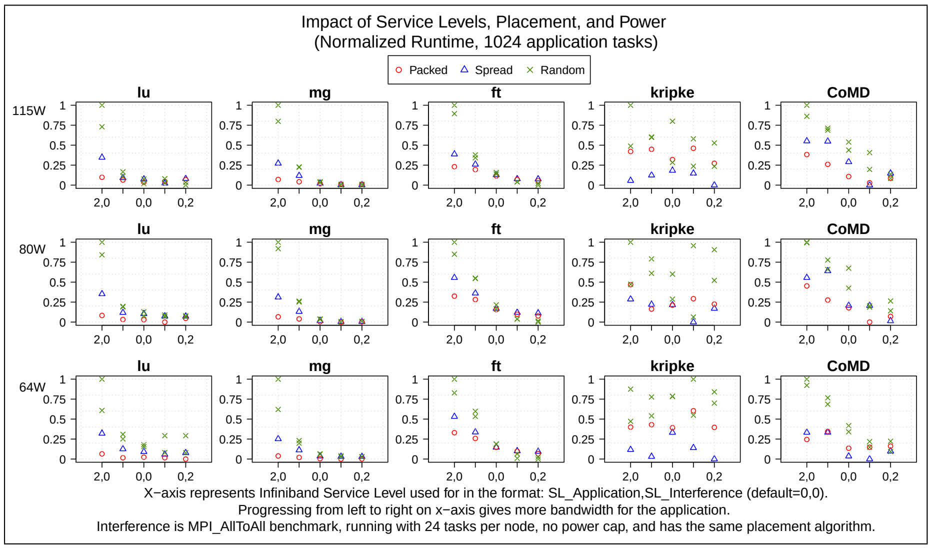
II. Experimental Methodology and Results

We run our experiments on the `catalyst` cluster, which is located at Lawrence Livermore National Laboratory. Catalyst is a 150 TeraFLOP/s, 324-node cluster with two 12-core Intel Xeon CPUs per node. Catalyst has an InfiniBand QDR network with a two-level fat-tree topology. Each node has two HCA network adapters, and there are 18 nodes per switch (total 18 switches). For power-capping, we use Intel's RAPL technology which is supported through `libmsr` [8] and `msr-safe`. The range for CPU power capping is between 65W and 115W. For controlling the network QoS, we use the `IPATH_SL` environment variable to select the virtual lanes [4]. 4 service levels associated with 4 virtual lanes are available (0 being the highest bandwidth, and 3 being the lowest bandwidth). The default setting is to run at service level 0 and the maximum CPU power of 115W. The *control jobs* we use are LU, FT and MG from NAS [2], and the Kripke and CoMD proxy applications [6], [7]. We generate inter-job interference by using the OSU `MPI_Alltoall` benchmark [1] using 32KB messages. We use 256-node allocations for our experiments, which is split between the control job and the interfering job. We pick three placement algorithms: `packed`, `spread` and `random`. Packed places all application ranks as close together as possible, so that the fewest number of switches are used from the fat tree. Spread distributes the ranks in a round-robin manner across the switches.

Figures 1 and 2 show our data[1]. Figure 1 shows the impact of using both the network adapters as opposed to picking one adapter across 5 benchmarks (2048 application ranks/86 nodes). Setting `IPATH_UNIT` selects one of the two adapters, and more bandwidth is allocated to the application (and less to interference) as we progress from left to right on the x-axis. Our main counter-intuitive observation here is the significant amount of variation that we see when both adapters are used, which is the default configuration on most HPC clusters. Note the scale of the y-axis, over 5x run-to-run variation was observed in our simple control benchmarks (e.g. FT). This can be mitigated by choosing one of the two adapters (see rows 2 and 3). In Figure 2, we set the IPATH_UNIT to 0, and further explore power capping and additional service levels. Here, we make three key observations: (1) contrary to popular belief of topology-aware scheduling, applications benefit from being spread across the switches (e.g. Kripke), (2) the default service

[1]More details and data can be made available in a full paper





level of zero can result in poor performance, and there are scenarios where running the application at a lower service level (with low-bandwidth) but with no interference turns out to be better; and (3) different placement algorithms may be needed based on the power cap under consideration. Our next step is to explore prediction models and algorithms for advanced research of HPC system software with such data.

Acknowledgment

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

References

[1] OSU Benchmarks. http://mvapich.cse.ohio-state.edu/benchmarks/, 2016.

[2] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, L. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber, H. D. Simon, V. Venkatakrishnan, and S. K. Weeratunga. The nas parallel benchmarks. In *Proceedings of the 1991 ACM/IEEE Conference on Supercomputing*, Supercomputing '91.

[3] A. Bhatele, K. Mohror, S. H. Langer, and K. E. Isaacs. There goes the neighborhood: Performance degradation due to nearby jobs. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '13.

[4] D. Crupnicoff, S. Das, and E. Zahavi. Deploying quality of service and congestion control in infiniband-based data center networks. Technical report, Mellanox Technologies, 2005.

[5] Y. Inadomi, T. Patki, K. Inoue, M. Aoyagi, B. Rountree, M. Schulz, D. Lowenthal, Y. Wada, K. Fukazawa, M. Ueda, M. Kondo, and I. Miyoshi. Analyzing and mitigating the impact of manufacturing variability in power-constrained supercomputing. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '15.

[6] A. J. Kunen, T. S. Bailey, and P. N. Brown. KRIPKE - A Massively Parallel Transport Mini-App. In *American Nuclear Society M&C 2015*.

[7] O. Pearce, H. Ahmed, R. W. Larsen, P. Pirkelbauer, and D. F. Richards. Exploring dynamic load imbalance solutions with the comd proxy application. *Future Generation Computer Systems*, 2017.

[8] B. Rountree and S. Labasan. Libmsr. https://github.com/LLNL/libmsr.

[9] L. Savoie, D. K. Lowenthal, B. R. d. Supinski, T. Islam, K. Mohror, B. Rountree, and M. Schulz. I/o aware power shifting. In *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2016.

[10] H. Subramoni, P. Lai, S. Sur, and D. K. D. Panda. Improving application performance and predictability using multiple virtual lanes in modern multi-core infiniband clusters. In *Proceedings of the 2010 39th International Conference on Parallel Processing*, ICPP '10.