A Simulation-Based Framework to Reduce I/O Contention in HPC

Simone Pernice [1], Ahmad Tarraf [2], Jean-Baptiste Besnard [3], Barbara Cantalupo [1], Alberto Cascajo [4], David E. Singh [4], Felix Wolf [2], Jesús Carretero [4], Sameer Shende [5], and Marco Aldinucci [1]
[1]*Department of Computer Science, University of Turin, Turin, Italy*
[2]*Department of Computer Science, Technical University of Darmstadt, Darmstadt, Germany*
[3]*DataDirect Networks*
[4]*Computer Science and Engineering Department, University Carlos III of Madrid, Madrid, Spain*
[5]*ParaTools, Inc.*

***Abstract*—Emerging hardware constraints are pushing workloads to become more composite. This transition involves new jobs where the HPC I/O systems are shared among multiple and concurrent jobs. This can generate load imbalances and contention in the end-to-end I/O paths, thus degrading the I/O system performance and the workloads. Recognizing this context, we define a simulation-based framework that alleviates resource contention in applications and ultimately allows us to design contention avoidance strategies. Specifically, by capturing behavior system-wide and extracting phases and characteristics of various performance metrics, we can mitigate contention by delaying the launch of applications. This framework leverages frequency domain analysis of performance metrics alongside clustering methods and is coupled with a comprehensive model of an HPC system implemented using Extended Stochastic Symmetric Nets.**

***Index Terms*—HPC, Monitoring, Performance Modeling, Markov Process, I/O.**

I. Introduction

One of the most significant factors limiting performance in High-Performance Computing (HPC) is *contention* which manifests itself among different components, including memory, input/output (I/O) operations, and networking. Contended resources typically exhibit a non-linear response in terms of their overall performance [1]. Therefore, avoiding such configurations can significantly boost the performance of both system and applications [2]. Moreover, HPC applications are known to exhibit alternating behavior [3], as they often involve phases featuring computation, communication, and I/O outlining iterations, and consequently, resource usage usually follows a similar periodic performance pattern. Given these observations, we propose a simulation-based framework to monitor applications running on a supercomputer, identifying and optimizing jobs with periodic resource needs. Our methodology combines system-wide measurement capabilities with modeling of applications across a wide range of metrics, with a particular focus on I/O. These models (representing the different applications) are then employed to predict future machine configurations. Leveraging our model in a simulation, we can identify likely contending jobs and apply a small temporal shift to avoid potential performance degradation.

II. Framework Overview

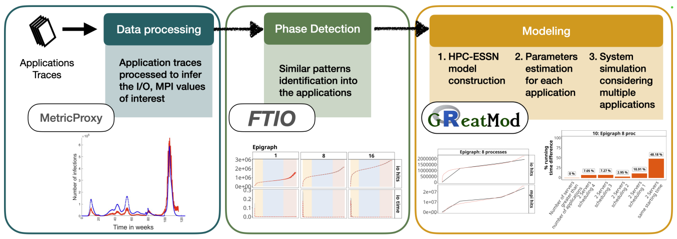
Applications Traces
Data processing
Application traces processed to infer the I/O, MPI values of interest
MetricProxy
Number of infections
Time in weeks
Phase Detection
Similar patterns identification into the applications
FTIO
Epigraph
1
8
16
io hits
io time
Modeling
1. HPC-ESSN model construction
2. Parameters estimation for each application
3. System simulation considering multiple applications
GReatMod
Epigraph: 8 processes
io hits
mpi hits
10: Epigraph 8 proc
% running time difference
0 %
7.05 %
7.27 %
2.95 %
10.91 %
48.18 %
Number of servers greater than number of applications
2 Servers scheduling 4
2 Servers scheduling 3
2 Servers scheduling 2
2 Servers scheduling 1
2 Servers same starting time

Fig. 1: Overview of the modeling toolchain.

We developed a modeling chain combining various components, as illustrated in Fig. 1, to consistently monitor and model past and running applications. Specifically, we used and improved the following tools:

1) *Metric Proxy* for collecting applications and system data. The Metric Proxy [4] is a hierarchical monitoring infrastructure that leverages a Tree-Based Overlay Network to provide a near-real-time view of the entire system.

2) *FTIO* for capturing and labeling periodic I/O phases. The tool "Frequency Techniques for I/O" (FTIO) [5], uses frequency techniques coupled with outlier detection methods to identify and label the I/O phases of an application through a single metric, namely the frequency.

3) *GreatMod* for modeling the target system. GreatMod [6] uses high-level graph formalisms, including Petri Nets and their extensions Extended Stochastic Symmetric Nets (ESSNs) [7] to represent system dynamics.

This work was supported by Spoke 1 "FutureHPC & BigData" of the Italian Research Center on High-Performance Computing, Big Data and Quantum Computing (ICSC) funded by MUR Missione 4 Componente 2 Investimento 1.4: Potenziamento strutture di ricerca e creazione di "campioni nazionali di R&S (M4C2-19)" - Next Generation EU (NGEU). We acknowledge the support of the European Commission and the German Federal Ministry of Education and Research (BMBF) under the EuroHPC Programme ADMIRE (GA No. 956748, BMBF funding No. 16HPC006K). As part of the NHR funding, we further acknowledge the support from BMBF and HMWK. Finally, this publication is part of the I+D+I project PID2022-138050NB-I00, financed by MICIU/AEI/10.13039/501100011033/"FEDER Una manera de hacer Europa".

Corresponding author: Simone Pernice (email: simone.pernice@unito.it).

We propose an intermediate-level abstraction of the system as a Markovian model, denoted HPC-ESSN, which simulates the execution of a variable mix of applications competing for storage resources on the same system. The model's parameters are derived from the system's architecture and calibrated on actual measurements using data collected by the Metric Proxy and the phases detected by FTIO. Subsequently, using this information for defining the parameters in the model (a set of parameters of each phase), the calibrated model can be used to predict global system's behavior when a different set of applications are running, for example, evaluating the behavior of arbitrary combinations of known applications.

III. Use-Cases and Results

We demonstrated the value of our approach using two different examples: 1) EpiGraph, an agent-based parallel simulator that models the propagation of influenza and COVID-19 [8]; 2) malleable benchmarks, developed as an evolution of [9] to allow the generation of diverse malleable applications with distinct usage patterns. We conducted experiments on the HPC4AI (https://hpc4ai.unito.it) system, which has 68 nodes with Omnipath 100 GB interconnect and relies on BeeGFS as a back-end file system. Each node features 2 sockets with 18 Intel Xeon E5-2697 v4 cores. We validate our approach demonstrating the effectiveness of scheduling multiple applications to mitigate contention through a process we refer to as phase shift. This approach utilizes our machine model HPC-ESSN to test hypotheses regarding contention by introducing phase shifts between applications that are likely to contend. The likelihood of contention is characterized by both the intensity of events (in our case, I/O operations) and the Lowest Common Multiplier (LCM) of their periods, with lower LCM values indicating a higher frequency of interactions between programs. The study analyzes a system with twelve configurations combining the EpiGraph application and malleable benchmarks with varying process counts. Two primary scenarios are considered: the Optimal Resource Allocation Scenario, where ample servers prevent queuing and allow independent execution, and the Resource-Constrained Scenario, where limited servers cause contention and delays. Although we focus on this specific combination of applications for demonstration purposes, it is worth noting that our model can be easily extended to incorporate additional calibrated applications from the previous section.
By evaluating different scheduling strategies under these conditions, our study explores how staggered application start times can mitigate performance bottlenecks. Fig. 2 shows the percentage increase in application running time compared to the best-case scenario, where sufficient servers are available for all applications. Lower percentages indicate better performance with minimal delays. The figure compares six scenarios, highlighting the impact of resource constraints and different scheduling strategies on execution efficiency. As expected, the worst-case situation (*2 Servers same starting time*) displays the most substantial difference, indicating a significant increase in the average total running time of each

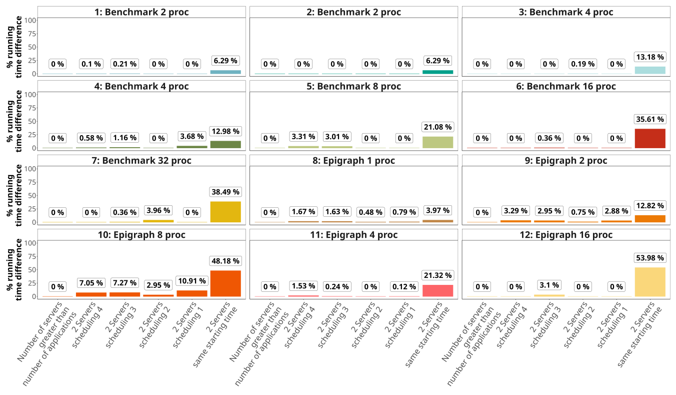

Fig. 2: Running time increase (in percentage) of each application relative to the best-case scenario across six scenarios.

application due to queuing in I/O operations. In contrast, the scheduled scenarios exhibit applications with average total times comparable to those in the best-case scenario. However, applications that experience longer durations due to competition for available servers demonstrate a more pronounced increase in their running time. These results highlight the impact of resource contention on application performance. Notably, Fig. 2 reveals that "scheduling 2" is the most effective strategy, as it minimizes the increase in running time for applications and thus optimizes overall system performance. This outcome underscores the importance of carefully designed scheduling strategies in mitigating the effects of resource contention and optimizing system efficiency.

IV. Conclusion and Future Work

Contention can arise at various layers within HPC systems, ranging from hardware components like networks and storage devices to the application level itself, including MPI. Due to its complex and non-linear nature, accurately modeling contention in advance poses a significant challenge. To address this issue, our work introduces a novel contention avoidance methodology that combines frequency analysis with predictive application modeling. Our approach first captures the behavior of applications across the entire system, extracting characteristic phases and frequencies of key performance metrics. Then, we leverage a Petri Net-based machine model, namely HPC-ESSN, to predict potential contention points, enabling the effective contention mitigation through proactive measures. More practically, our approach enables us to either delay the launch of a given application or introduce artificial delays into currently running applications, thereby reducing the likelihood of contention. We have successfully applied our modeling methodology to representative applications and demonstrated its effectiveness in enhancing overall system performance. Future work will focus on expanding our analysis on more benchmarks and varying our model against the real system. Also, we aim to integrate our simulation-based approach with scheduling algorithms to create a comprehensive simulation-based framework for optimizing system performance.

References

[1] L. A. Steffenel, M. Martinasso, and D. Trystram, “Assessing contention effects on MPI_alltoall communications,” in *Advances in Grid and Pervasive Computing: Second International Conference, GPC 2007, Paris, France, May 2-4, 2007. Proceedings 2*. Springer, 2007, pp. 424–435.

[2] D. Skinner and W. Kramer, “Understanding the causes of performance variability in HPC workloads,” in *IEEE International. 2005 Proceedings of the IEEE Workload Characterization Symposium, 2005.* IEEE, 2005, pp. 137–149.

[3] A. Gainaru, G. Aupy, A. Benoit, F. Cappello, Y. Robert, and M. Snir, “Scheduling the I/O of HPC applications under congestion,” in *2015 IEEE International Parallel and Distributed Processing Symposium*. IEEE, 2015, pp. 1013–1022.

[4] J.-B. Besnard, A. Tarraf, A. Cascajo, and S. Shende, “Introducing the metric proxy for holistic i/o measurements,” in *High Performance Computing. ISC High Performance 2024 International Workshops*, M. Weiland, S. Neuwirth, C. Kruse, and T. Weinzierl, Eds. Cham: Springer Nature Switzerland, 2025, pp. 213–226.

[5] A. Tarraf, A. Bandet, F. Boito, G. Pallez, and F. Wolf, “Capturing periodic I/O using frequency techniques,” in *2024 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2024, pp. 465–478.

[6] P. Castagno, S. Pernice, G. Ghetti, M. Povero, L. Pradelli, D. Paolotti, G. Balbo, M. Sereno, and M. Beccuti, “A computational framework for modeling and studying pertussis epidemiology and vaccination,” *BMC bioinformatics*, vol. 21, 2020.

[7] S. Pernice, M. Pennisi, G. Romano, A. Maglione, S. Cutrupi, F. Pappalardo, G. Balbo, M. Beccuti, F. Cordero, and R. A. Calogero, “A computational approach based on the colored Petri Net formalism for studying multiple sclerosis.” *BMC bioinformatics*, vol. 20, no. 6, pp. 1–17, 2019.

[8] G. Martín, D. E. Singh, M.-C. Marinescu, and J. Carretero, “Towards efficient large scale epidemiological simulations in EpiGraph,” *Parallel Computing*, vol. 42, pp. 88–102, 2015, parallelism in Bioinformatics. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016781911400115X

[9] A. Cascajo, D. E. Singh, and J. Carretero, “Detecting interference between applications and improving the scheduling using malleable application clones,” *The International Journal of High Performance Computing Applications*, vol. 38, no. 2, pp. 108–133, 2024. [Online]. Available: https://doi.org/10.1177/10943420231220898