Improving I/O Performance of HPC Applications Using Intra-Job Scheduling

Arnab K. Paul*, Olaf Faaland†, Adam Moody†, Elsa Gonsiorowski†,
Kathryn Mohror†, Ali R. Butt*
*Virginia Tech, †Lawrence Livermore National Laboratory
{akpaul, butta}@vt.edu, {faaland1, moody20, gonsiorowski1, mohror1}@llnl.gov

I. Introduction

The current trend for high performance computing (HPC) systems is that processor performance improves at a rate of 20% per year, while disk access time improves by only 10% every year [2]. As a result, massively parallel HPC applications can suffer from imbalance in computation and I/O performance, with I/O operations becoming a limiting factor in application efficiency [3]. To mitigate this problem, much effort has been dedicated to implementing high performance parallel file systems to support the I/O needs of HPC applications. The Lustre file system [6] is one of the most widely-used parallel file systems, supporting ∼60% of the top supercomputers in the latest Top-500 list (June, 2019) [1].

A lot of promising approaches have been made to improve I/O behavior of HPC applications. There are recent works on improving I/O scheduling for burst buffer systems [4]. Also, with the increase in machine learning, models are being built to predict runtime I/O behavior of HPC applications using the job scripts [7].

An analysis of the I/O patterns for HPC applications has shown that they have bursty write patterns. Also, the I/O behavior of such applications is predictive and can be modeled. Our work makes use of such predictive nature of the I/O requests of HPC applications. We have data for per-minute I/O statistics for over 1.4 million jobs that ran on Quartz cluster in Lawrence Livermore National Laboratory (LLNL). All of these jobs can be modeled using time-series prediction techniques. In this preliminary study, we focus on write requests. Once jobs are modeled and the runtime write behavior predicted, the requests can be manipulated to lower HPC job I/O contention, which will increase the overall I/O performance.

II. Methodology

The dataset for the job-statistics is divided into training and validation data. Jobs from the traning dataset are modeled. The validation dataset serves two purposes. First, the modeling of job write pattern is verified using the validation dataset. Second, the validation dataset serves as the test case for reducing I/O contention. Whenever a new job arrives at the job scheduler, the scheduler will look at the other jobs that are currently running on the system. The other jobs have already been modeled and the pattern of those jobs can be predicted based on the model. After the initial few write requests of the newly submitted job, that job can be matched to a model of another job which was previously run on the system, thus getting access to its future write requests. The write requests of the present running job can then be scheduled in such a way that the overall I/O contention of the system decreases. This will ensure increase in the I/O performance of the system. The machine learning model will continuously evolve based on the results of the jobs from the validation dataset.

III. Preliminary Results and Next Steps

We have built a simulator for the Lustre file system to accurately model the system in Quartz cluster at LLNL. The simulator has four key components closely mirroring those of Lustre's OST, OSS, MDT, and MDS, which implement the various Lustre operations and enable us to collect data about the system behavior. All the network components in the simulator are modeled using Network Simulator (NS-3) [5]. This simulator will measure the I/O performance of the system. The initial results from the time-series modeling have been promising. The jobs that have been validated show an accuracy of 95% in predicting job write bursts.

The next steps will be to build a system that reduces I/O contention by matching the jobs with their model and scheduling their write requests. We need to measure the I/O performance of the jobs as well as the overall performance of the system.

References

[1] T. 500. Top 500 list - june 2019. https://www.top500.org/lists/2019/06/. Accessed: November 2 2019.
[2] J. L. Hennessy and D. A. Patterson. *Computer architecture: a quantitative approach*. Elsevier, 2011.
[3] S. Lang, P. Carns, R. Latham, R. Ross, K. Harms, and W. Allcock. I/o performance challenges at leadership scale. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, pages 1–12. IEEE, 2009.
[4] W. Liang, Y. Chen, J. Liu, and H. An. Contention-aware resource scheduling for burst buffer systems. In *Proceedings of the 47th International Conference on Parallel Processing Companion*, page 32. ACM, 2018.
[5] Nsnam. Network simulator - ns3. https://www.nsnam.org. Accessed: November 2 2019.
[6] OpenSFS and EOFS. Lustre file system. http://lustre.org/. Accessed: November 2 2019.
[7] M. R. Wyatt II, S. Herbein, T. Gamblin, A. Moody, D. H. Ahn, and M. Taufer. Prionn: Predicting runtime and io using neural networks. In *Proceedings of the 47th International Conference on Parallel Processing*, page 46. ACM, 2018.