A Multifaceted Approach to Automated I/O Bottleneck Detection for HPC Workloads

Izzet Yildirim[1], Hariharan Devarajan[2], Anthony Kougkas[1], Xian-He Sun[1], and Kathryn Mohror[2]
iyildirim@hawk.iit.edu, hariharandev1@llnl.gov, akougkas@iit.edu, sun@iit.edu, kathryn@llnl.gov
[1]Illinois Institute of Technology
[2]Lawrence Livermore National Laboratory

I. Extended Abstract

Real-world HPC workloads impose a lot of pressure on storage systems as they are highly data dependent. Such workloads may be in the form of individual applications such as simulations and machine learning (ML) applications, or in the form of workflows that consist of a collection of applications that work cooperatively towards a common goal, such as coupling of scientific simulation with data analytics and artificial intelligence (AI). On the other hand, as a result of recent developments in storage hardware, it is expected that the storage diversity in upcoming HPC systems will range from node-local storage in the form of solid-state disks (SSDs), and shared burst buffer layers, to parallel file systems with hard disk drives (HDDs). This growing complexity in the storage system presents challenges to users, and often results in I/O bottlenecks due to inefficient usage.

Reducing I/O bottlenecks has been the subject of several studies. Some of the earliest studies worked to explain I/O characteristics and their impact on a file system based on their evaluation [1], [2]. As the HPC systems grew more complex, this manual approach became impractical since it needed reconfiguring and executing applications to understand the implications of adjusted I/O characteristics. Following the development of I/O characterization tools (e.g., Darshan [3]), other studies captured I/O traces from running applications and analyzed them manually. Such analysis often needed to be coupled with domain expert insight to derive positive outcomes. This process may take several weeks, which is undesirable. Most analysis tools look at the performance data from one perspective. However, the problem is multifaceted with many metrics to consider. For instance, if we look at the CM1 [4] workload and only consider I/O request size as a metric, we observe that most I/O happens during read operations. However, if we consider the amount of time spent in I/O operations, we observe that CM1 spends the most of its I/O time on small writes. This example demonstrates how consideration of different metrics can identify different sources of I/O bottlenecks. Consequently, performing this multifaceted analysis manually is laborious and error-prone even for experts.

In this work, we develop a methodology that produces a *multifaceted view* of the I/O behavior of a workload to identify potential I/O bottlenecks. The facets of the view include time steps, individual processes, files, transfer sizes, and observed bandwidths. To demonstrate this methodology, we build a tool called *Diagnose I/O* (DigIO), which performs this task automatically on an HPC system. This tool can potentially reduce weeks of analysis by an I/O expert into a matter of minutes. The DigIO tool is created as a Python library that utilizes Dask [5] to benefit from distributed analysis on workload traces. To enable efficient distributed analysis, we first convert the collected I/O trace (from Recorder [6]) into the Parquet format [7]. Then our tool consumes the Parquet files to build a multifaceted view of the workload, that accounts for several key I/O metrics. This view represents a graph of I/O observations within the workload. Finally, we pass our observations through a rule engine which produces bottleneck diagnoses automatically. We analyzed six different workloads including two simulation applications, two AI applications, and two scientific workflows using Montage. These workloads were executed on the Lassen supercomputer at Lawrence Livermore National Laboratory (LLNL) [8]. The main contributions of this work are:

1) Illustration of a methodology to detect I/O bottlenecks in complex HPC workloads
2) Design of an automated tool, called DigIO, which automates the I/O bottleneck detection
3) Demonstration of I/O bottleneck detection for six diverse HPC workloads

II. Automated I/O Bottleneck Detection

A. The design of DigIO

DigIO uses Dask and runs the analysis on Lassen which uses the LSF workload manager. This allows us to complete the bottleneck detection in minutes even for large workloads with millions of I/O operations. Figure 1 shows the overview

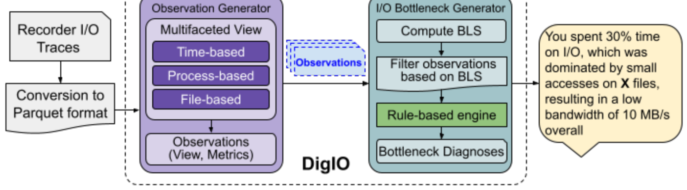
Recorder I/O Traces
Conversion to Parquet format
Observation Generator
Multifaceted View
Time-based
Process-based
File-based
Observations (View, Metrics)
Observations
I/O Bottleneck Generator
Compute BLS
Filter observations based on BLS
Rule-based engine
Bottleneck Diagnoses
DigIO
You spent 30% time on I/O, which was dominated by small accesses on X files, resulting in a low bandwidth of 10 MB/s overall

Fig. 1. Overview of DigIO. It takes Parquet-formatted I/O traces as input and generates observations though a multifaceted view. It then filters those observations according to their BLS and passes them through a rule engine. Finally, the rule engine produces bottleneck diagnoses.

of the tool, its components and how they interact with each other. The *Observation Generator* uses the traces to build a multifaceted view that the observations are derived from. The observations are then fed into the *I/O Bottleneck Generator*, which in turn filters them and run them through the rule-based engine to produce bottleneck diagnoses.

1) Multifaceted Analysis: To perform a multifaceted analysis, DigIO creates filter groups and metrics. We have five filter groups, that split the I/O trace into different views of the same performance data, as follows: a) *Time-based:* splits the I/O trace into time-steps according to the job time; b) *Process-based:* splits the I/O trace into processes for a given process (or app); c) *File-based:* splits the I/O trace into files for a given file (or directory); d) *Transfer size-based:* splits the I/O trace into transfer size groups (e.g., 4KB, 1MB); e) *Bandwidth-based:* splits the I/O trace into bandwidth groups (e.g., 100MB/s, 1GB/s). We currently employ the following metrics: *total I/O size*, *I/O time per process*, *number of I/O operations*, *number of files*, *average bandwidth*, *transfer size*, and *parallelism*. DigIO uses the metrics to make observations and calculate the *Bottleneck Likelihood Score* (BLS) that gives an indication of the importance of the observation. We score each metric by its observed value in the filter group, based on global min-max of the metric. For instance, in a time-based filter group, if the average bandwidth of a particular time-step is significantly lower than the ideal bandwidth, we regard such bandwidth as an indicator of a bottleneck.

Using the filter groups and metrics, DigIO creates a *multifaceted view* of the I/O behavior of the workload, resulting in more accurate bottleneck identification. One of the advantages of this approach is that it helps us eliminate false-positives efficiently. For instance, a time-step with densely concentrated I/O operations may be falsely identified as a bottleneck in the time-based view, but if the corresponding processes exhibit high parallelism in the process-based view with a satisfactory bandwidth, that observation gets discarded. Finally, to produce user-friendly text-based results, DigIO utilizes a heuristically designed rule-based engine to identify bottlenecks from candidate observations.

B. I/O Bottlenecks in HPC Workloads

We select HPC workloads with different I/O behaviors that are representatives of real-world scenarios. These include CM1 (an atmospheric simulation), HACC [9] (a cosmological simulation), CosmoFlow [10] (a cosmological simulation using DL), JAG ICF model [11], and two workflows [12], [13] using Montage (a mosaic engine). Due to lack of space, we will showcase the results from a single workload (CM1) and from only the time-based analysis (The rest of the results will be showcased on the poster).

1) Application Description: CM1 is an atmospheric simulation that models phenomena such as thunderstorms and tornadoes. It proceeds by iterations and produces more than 750 files (each around 128MB) during 193 simulation steps.

2) Observations detected by DigIO: Our time-based analysis shows that the application takes 668 seconds to finish, reads 20.03GB, and writes 1.14GB data in total. The I/O time per process is around 0.08 seconds and the number of I/O operations is around 26k. The initial observation that DigIO made was the fact that among 26k I/O operations, 86.9% of them was actually metadata operations. This indicates that there might be a large amount of unnecessary metadata operations that reduces the overall I/O performance of the application. DigIO also observed that 94.6% of I/O time is spent during the first 20 seconds and on only read operations. However, the BLS calculated for the time-step was only 48.6% as the tool found no significant issue on the other metrics such as the average bandwidth and the transfer size. This illustrates the efficiency of the multifaceted approach.

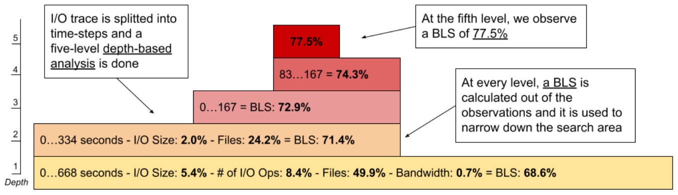
I/O trace is splitted into time-steps and a five-level depth-based analysis is done
77.5%
At the fifth level, we observe a BLS of 77.5%
83…167 = 74.3%
0…167 = BLS: 72.9%
At every level, a BLS is calculated out of the observations and it is used to narrow down the search area
0…334 seconds - I/O Size: 2.0% - Files: 24.2% = BLS: 71.4%
0…668 seconds - I/O Size: 5.4% - # of I/O Ops: 8.4% - Files: 49.9% - Bandwidth: 0.7% = BLS: 68.6%
5
4
3
2
1
Depth

Fig. 2. Time-based analysis on the I/O write operations of CM1. We run a five-level depth-based analysis and calculate BLS at every level for each time-step. As the search area narrows, we observe an increased BLS, indicating a likely bottleneck within the time-step being analyzed.

3) Bottleneck detected by DigIO: Figure 2 shows a potential bottleneck detected on the I/O write operations with a BLS of 77.5%. DigIO finds that in this particular time-step, the application does around 300 small (<500KB) I/O operations with a low bandwidth (~400MB/s) from a single process (no parallelism).

C. Conclusion

In this work, we demonstrate our methodology and our tool DigIO that produce multifaceted views of I/O data to identify I/O bottlenecks. We showcase that applying an automated multifaceted analysis is a complex task but can reveal several new insights on our performance data as shown in the evaluation for CM1. Specifically, we identify that, even though CM1 performs most I/O on read operations initially, it spends majority of I/O time on small write operations followed by metadata operations on these files. Additionally, we demonstrate the tool can perform this multifaceted analysis in seconds or minutes (as opposed to weeks by an I/O expert) and can produce user-friendly outputs for describing the bottleneck.

Acknowledgments

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract LLNL-POST-838770. Also, the material is based upon work supported by the National Science Foundation under Grant no. NSF OAC-2104013, OCI-1835764, and CSR-1814872.

References

[1] A. L. N. Reddy and P. Banerjee, “A study of i/o behavior of perfect benchmarks on a multiprocessor,” in *Proceedings of the 17th Annual International Symposium on Computer Architecture*, ser. ISCA ’90. New York, NY, USA: Association for Computing Machinery, 1990, p. 312–321. [Online]. Available: https://doi.org/10.1145/325164.325157

[2] P. Crandall, R. Aydt, A. Chien, and D. Reed, “Input/output characteristics of scalable parallel applications,” in *Supercomputing ’95:Proceedings of the 1995 ACM/IEEE Conference on Supercomputing*, 1995, pp. 59–59.

[3] P. Carns, R. Latham, R. Ross, K. Iskra, S. Lang, and K. Riley, “24/7 characterization of petascale i/o workloads,” in *2009 IEEE International Conference on Cluster Computing and Workshops (CLUSTER)*. Los Alamitos, CA, USA: IEEE Computer Society, sep 2009, pp. 1–10. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CLUSTR.2009.5289150

[4] H. Rahman, M. M. Verstraete, and B. Pinty, “Coupled surface-atmosphere reflectance (csar) model: 1. model description and inversion on synthetic data,” *Journal of Geophysical Research: Atmospheres*, vol. 98, no. D11, pp. 20 779–20 789, 1993. [Online]. Available: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/93JD02071

[5] M. Rocklin, “Dask: Parallel computation with blocked algorithms and task scheduling,” in *Proceedings of the 14th python in science conference*, vol. 130. Citeseer, 2015, p. 136.

[6] C. Wang, J. Sun, M. Snir, K. Mohror, and E. Gonsiorowski, “Recorder 2.0: Efficient parallel i/o tracing and analysis,” in *2020 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE, 2020, pp. 1–8.

[7] D. Vohra, *Apache Parquet*. Berkeley, CA: Apress, 2016, pp. 325–335. [Online]. Available: https://doi.org/10.1007/978-1-4842-2199-0_8

[8] “Lassen — High Performance Computing.” [Online]. Available: https://hpc.llnl.gov/hardware/platforms/lassen

[9] K. Heitmann, T. D. Uram, H. Finkel, N. Frontiere, S. Habib, A. Pope, E. Rangel, J. Hollowed, D. Korytov, P. Larsen, B. S. Allen, K. Chard, and I. Foster, “HACC cosmological simulations: First data release,” *The Astrophysical Journal Supplement Series*, vol. 244, no. 1, p. 17, sep 2019. [Online]. Available: https://doi.org/10.3847/1538-4365/ab3724

[10] Y. Oyama, N. Maruyama, N. Dryden, E. McCarthy, P. Harrington, J. Balewski, S. Matsuoka, P. Nugent, and B. V. Essen, “The case for strong scaling in deep learning: Training large 3d cnns with hybrid parallelism,” *IEEE Transactions on Parallel amp; Distributed Systems*, vol. 32, no. 07, pp. 1641–1652, 2021.

[11] J. L. Peterson, B. Bay, J. Koning, P. Robinson, J. Semler, J. White, R. Anirudh, K. Athey, P.-T. Bremer, F. Di Natale, D. Fox, J. A. Gaffney, S. A. Jacobs, B. Kailkhura, B. Kustowski, S. Langer, B. Spears, J. Thiagarajan, B. Van Essen, and J.-S. Yeom, “Enabling machine learning-ready hpc ensembles with merlin,” *Future Gener. Comput. Syst.*, vol. 131, no. C, p. 255–268, jun 2022. [Online]. Available: https://doi.org/10.1016/j.future.2022.01.024

[12] M. Reagan, *The Hand of God: Thoughts and Images Reflecting the Spirit of the Universe*. Templeton Press, 2011. [Online]. Available: https://books.google.com/books?id=L7BmzgAACAAJ

[13] M. Rynge, G. Juve, J. Kinney, J. Good, B. Berriman, A. Merrihew, and E. Deelman, “Producing an Infrared Multiwavelength Galactic Plane Atlas Using Montage, Pegasus, and Amazon Web Services,” in *Astronomical Data Analysis Software and Systems XXIII*, ser. Astronomical Society of the Pacific Conference Series, N. Manset and P. Forshay, Eds., vol. 485, May 2014, p. 211.