Impacts of Shared Filesystem Performance on Real-Time Data Acquisition and Analysis

Justin M. Wozniak
woz@anl.gov
Argonne National Laboratory
Lemont, Illinois, USA

Sushil Regmi
Tong Shu
SushilRegmi@my.unt.edu
Tong.Shu@unt.edu
University of North Texas
Denton, Texas, USA

Ian Foster
foster@anl.gov
Argonne National Laboratory
Lemont, Illinois, USA

ACM Reference Format:
Justin M. Wozniak, Sushil Regmi, Tong Shu, and Ian Foster. 2025. Impacts of Shared Filesystem Performance on Real-Time Data Acquisition and Analysis. In *Proceedings of The 2nd International Workshop on Near Real-time Data Processing for Interconnected Scientific Instruments (NRDPISI 2025).* ACM, New York, NY, USA, **??** pages. https://doi.org/XXXXXXX.XXXXXXX

1 OVERVIEW

Large scientific instruments are shared resources that are often coupled with advanced computing resources, such as HPC clusters, powerful storage systems, and, increasingly, machine learning-oriented hardware. Each of these resources may be a multi-user system shared among disparate teams. In the context of real-time data acquisition and analysis, sharing can have a negative impact on the ability of the computing system to satisfy scientific objectives. In this study, we focus on the impact of a shared **FS!** (**FS!**), such as NFS or GPFS, on the scientific workflows that move and analysis variable-sized data sets. We then present a preliminary study on the use of predictive methods to capture and potentially feed back information about filesystem usage to the scientific goal level.

2 MOTIVATION

The overarching goal of the work presented here is to support automated, near-real-time scientific data acquisition and analysis by enhancing reliability through performance prediction and anomaly detection. This will be delivered as a service to workflow-level services that are capable of responding to dynamic resource availabilities and capabilities, automatically steering data collection in support of scientific goals. Communication within this workflow, including among monitoring and predictive components, will be performed over a reliable Kafka-like service [**?** **?**].

The first step in such an effort is to select or develop a predictive model that can make reasonable predictions. To do this, we need to collect representative data. In this paper, we present a representative system for collecting data about shared **FS!** performance at the application level for user facilities like the **APS!** (**APS!**).

NRDPISI 2025, March 10, 2025, Singapore

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/XXXXXXX.XXXXXXX

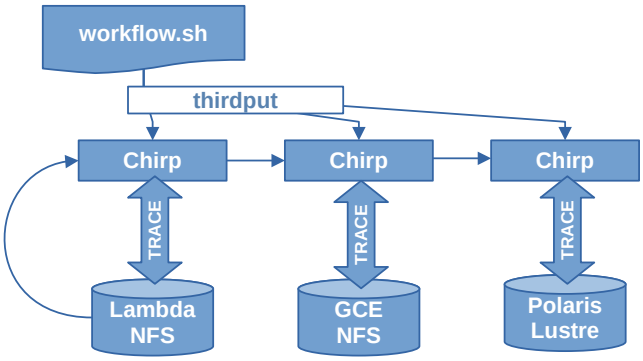

Figure 1: Synthetic version of Laue workflow. Data is copied across 3 filesystems.

Many other efforts have focused on particular aspects of performance prediction, such as modeling the use of HPC computing resources or networks. The novelty of this work is our focus on whole-system ad hoc scientific facility infrastructures, their topologies, and the integration of scientific goals with traditional systems management.

3 BACKGROUND

We selected a Laue diffraction workflow from the **APS!** for this study [**?**]. As described in the original paper, this experiment is computationally expensive, and a typical 12-hour scane would run for 250 days on a conventional computer. The scientific objective, described more fully in the reference, is to produce 3D models of crystal structures such as that of Aluminum metals, by reconstructing thousands of image frames from the x-ray diffraction pattern. The detector and standard **APS!** tools produce HDF5 files consisting of a collection of 300-400 images, each 2048 by 2048 pixels.

Data is collected by the APS Data Management system, copied from there to a central storage system, and from there to the **ALCF!** (**ALCF!**). Scans arrive every 72 seconds over a experiment run of 12 hours, leading to an aggregate bandwidth of 1.4 Mbps. The storage required for a full run including reconstructions was 4-8 TB. While this data rate is low for modern storage and network hardware, it is representative of other workflows at the **APS!**. The **APS!** upgrade [**?**], currently nearing completion, will dramatically increase data rates for all **APS!** experiments.

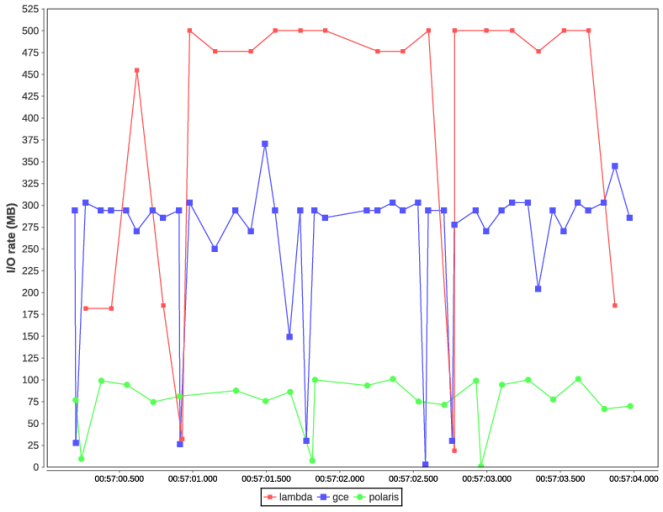

Figure 2: Sample of I/O duration data across filesystems.

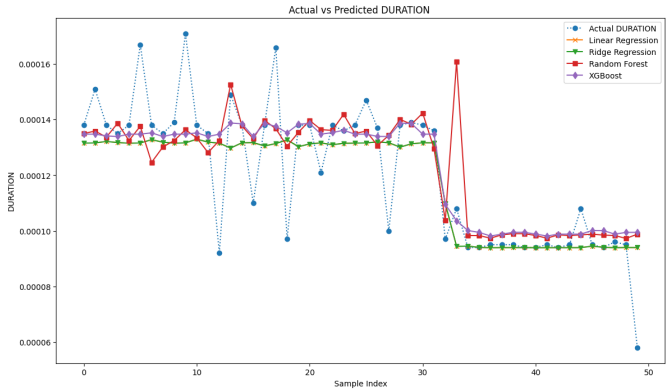

Figure 3: Actual and predicted IOP durations using multiple algorithms.

4 WORKFLOW

The Laue workflow is our reference point, but it is not always runnable due to limitations on **APS!** availability and the administration of **APS!** computing systems. Thus, we reproduced this workflow on generally available computing systems at **ANL!** (**ANL!**). Thus, we developed the synthetic workflow shown in Figure **??**. This consists of a network of 3 Chirp [**?**] fileservers running on 3 **FS!**s that are comparable to the **FS!**s used by the real Laue workflow running at the **APS!**. The Lambda **FS!** is a small shared NFS service supporting a small, focused group, representing the data acquisition computer at the **APS!**. The GCE **FS!** is a large NFS service supporting a whole directorate, with hundreds of diverse users, representing the larger Data Manager service at the **APS!**. The Polaris FS is a GPFS service that is the same HPC **FS!** targeted by the real **APS!** workflow.

The workflow is scripted to trigger a file transfer randomly at intervals of 0-60 minutes. Two such workflows run simultaneously, one moving an 11 KB file 100 times, and one moving a 126 MB file 10 times. Each run continuously, 24 hours a day, looping 1000 times each, which takes about 22 days total. The file is copied from an initial location into the shared filesystem, from there to the intermediate filesystem, and from there to the final HPC filesystem. Chirp-to-Chirp copies are performed using the Chirp `thirdput` feature, which performs a third-party copy from server to server.

Fine-grained logs are captured using slight modifications to the normal Chirp logs. These capture time-of-day, hour-of-day, day-of-week, type of I/O operation (read or write), size of operation in bytes, and the time in microseconds for the operation. This allows for periodic behavior to be easily captured. The Chirp server was modified to use `O_DIRECT` for user file operations so that the impact of caching would not have an effect.

5 ANALYSIS

Results using this workflow have been captured for several months on an ongoing basis to the present, starting in mid-October 2024. The goal of this is to collect enough data so that anomalies may be detected, at present or retrospectively, and so that user-level filesystem performance may be predicted over some short future interval.

A illustrative sample of representative data with `O_DIRECT` disabled is shown in Figure **??**. This shows the effect of caching, as most operations take run at a high data rate (short I/O operation durations), but some are slower, presumably due to cache flushing. This effect is not seen with `O_DIRECT` enabled.

A preliminary prediction study is shown in Figure **??**. As shown, the machine learning-based methods do not currently capture the variability in the underlying data very well, but do capture its average. More methods are currently being applied, including LSTMs, which will have a better chance to pick up on the **FS!** congestion patterns imposed by other users. We hypothesize that these will be predictable using a sufficiently advanced LSTM.

6 CONCLUSION

In this paper, we described an experimental testbed and preliminary investigation into the impacts of shared **FS!**s on data acquisition and analysis workflows. Our scripts [**?**] and data [**?**] are already available to other teams for collaboration or extension.

7 ACKNOWLEDGMENTS

This work was supported by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research, under contract number DE-AC02-06CH11357. This research is also sponsored by U.S. National Science Foundation under Grant No. OAC-2306184 and Grant No. OAC-2443633 with the University of North Texas.

Temporary page!

LaTeX was unable to guess the total number of pages correctly. As there was some unprocessed data that should have been added to the final page this extra page has been added to receive it.

If you rerun the document (without altering it) this surplus page will go away, because LaTeX now knows how many pages to expect for this document.