ISBN: 978-3-95450-231-8 ISSN: 2673-5490 doi: 10.18429/JACoW-IPAC2023-THPL008

UPGRADES TO LOGGING AND ML ANALYTICS ARCHITECTURE AT APS*

N. Kuklev†, R. Soliday, I. Lobach, H. Shang
Advanced Photon Source, Argonne National Laboratory, Lemont, IL, USA

Abstract

Several machine learning (ML) projects on anomaly detection and optimization were recently started at the Advanced Photon Source (APS). To improve training data quality, and accommodate the upcoming APS Upgrade changes, a large increase in the number and size of log files is expected. Recent studies found performance bottlenecks in the current log analysis architecture, especially for large ML analytics tasks. We explored several approaches to improve both data density and throughput. First, we swapped lzma compression algorithm for modern alternatives like zstd and lz4, scanning presets to find an optimal one that increased decompression throughput by 10x for a 20% file size increase. Several lossy compression schemes were attempted to take advantage of limited device resolution and ML quantization, yielding further size decreases with reasonable fidelity losses. Finally, we tested several analytics and time-series databases, finding them faster for both linear and random-access reads while maintaining good compression ratios. They also enabled offloading analytics computations to server nodes, reducing network load. Our results indicate that with some effort, it is possible to increase the amount of logged data significantly while improving ML analytics performance.

INTRODUCTION

Particle accelerators demand tight tolerances on accuracy and stability of beam parameters, which increases the time and cost of conventional expert-driven tuning, troubleshooting, and fault analysis. Efficiently automating these processes has motivated recent work on a variety of ML-aided methods for optimization and anomaly detection, which requires historical logs and/or real-time data for both pretraining the algorithm and for the immediate functionality. For example, an anomaly detection routine is trained on labelled historical malfunctions and then analyzes incoming data for similar signals. So far, all project at APS have relied on the standard logging infrastructure with reasonable results. However, several areas of improvement have been identified. In this paper we prototype solutions to increase the amount of stored data and its analysis speed, with the goal of smoothly integrating ML-specific tools into the upcoming APS-U logging infrastructure.

* The work is supported by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences, under Contract No. DE-AC02-06CH11357.
† nkuklev@anl.gov

CURRENT APS LOGGING SYSTEM

The APS accelerator complex is comprised of a large variety of physical devices with various interfaces and control protocols. These devices read and write through the EPICS control system, which presents a unified I/O interface to the users. A portion of the read data is stored in 'logs'. This enables debugging, fault analysis, and many other business-critical processes.

Conceptually, logs are time-series objects containing values (floats, ints, arrays, etc.) collection timestamps (floats, ints). Storing every read result in logs is not feasible due to storage requirements, and thus a manually configured subset is retained. APS logging system uses a collection of tools internally developed and maintained by the AOP group [1–3]. It is comprised of several stages:

- Configuration files are created with PV name, storage location, and acquisition parameters;
- PVs are read [4] using a server cluster, storing data into uncompressed SDDS files;
- Monthly, data is merged into compressed SDDS files;
- On schedule, old data is data subsampled or pruned.

ML-related limitations

Preliminary work on using logger data for anomaly detection and to initialize optimizers has recently been recently performed [5–7]. Several challenges have been identified in terms of read and search performance. Primary causes were determined to be the slow lzma decompression combined with overly broad queries (many devices) and lack of temporal indexing (beyond monthly chunking). Given small average file size of 0.1-10MB, and network file system latency, there are fundamental limits to how much improvement can be obtained without a new architecture.

Planned APS-U Apgrades

After extensive use of the current logging system over previous decades, it is quite robust to various breakdowns and has no fundamental issues that prevent its continued usage in APS-U, which is the current plan. In addition to standard devices, APS-U will introduce new high-speed data sources through 'DAQ' system for working with BPMs, power supplies, and other fast data sources [8]. Subsampled DAQ data will be exposed though DAQ plugins to the control system, and thus logged similarly to standard EPICS devices [9].

ISBN: 978-3-95450-231-8 ISSN: 2673-5490 doi: 10.18429/JACoW-IPAC2023-THPL008

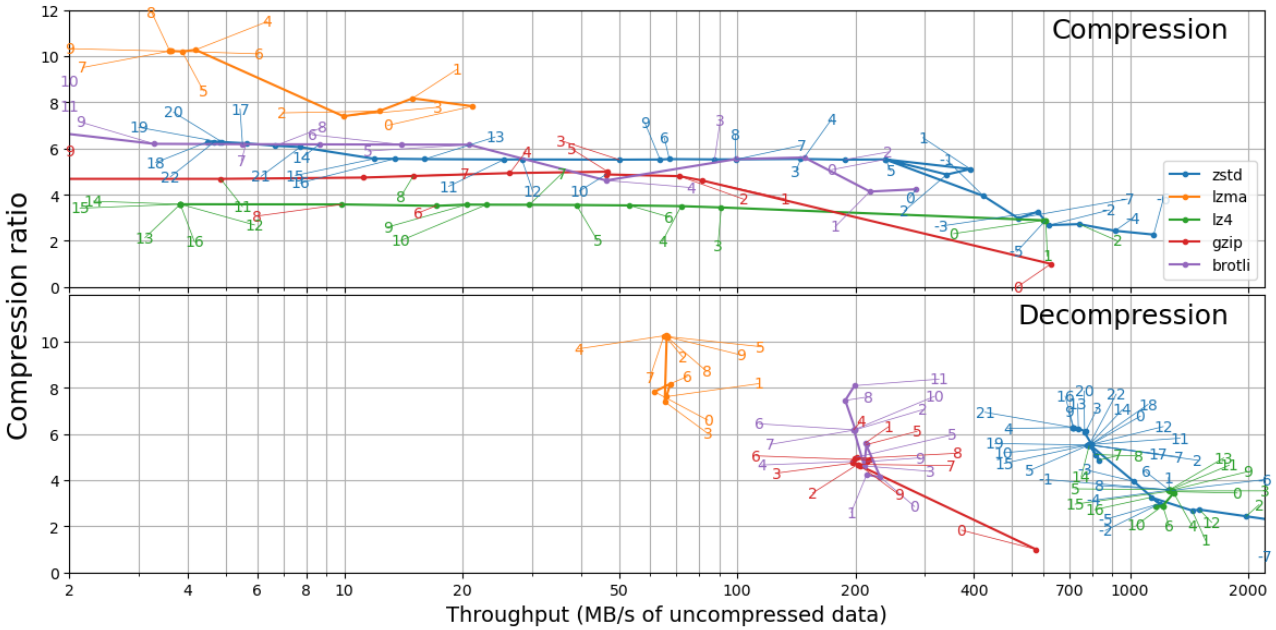

Figure 1: Performance of lossless algorithms applied to 2022 APS logger data. Numbers indicate preset (level). Latest library version as of April 2023 were used with default parameters, except lgwin=24 for brotli.

STORAGE DENSITY IMPROVEMENTS

The need for a new ML data architecture is at odds with maintaining interoperability. We decided to approach this in two ways, first of which was improving standard logging by increasing storage density and decompression performance through use of new lossless and lossy formats.

Lossless Compression

Compression algorithms have seen significant progress in the last decade, driven by data streaming and processing requirements. One such advance is the asymmetric numerical systems (ANS) entropy encoding [10], upon which formats like ZStandard (zstd) [11] rely for their performance gains.

Currently, APS logs are compressed with the lzma algorithm (.xz container) that is implemented by directly integrating liblzma sources into SDDSlib at a hardcoded compression level of 2 [12]. We reviewed various compression formats for those with high performance and good project activity, with final candidates being gzip, lzma, lz4, zstd, and brotli. We also considered using FLAC (used in lossless audio and for generic time series), but found 32-bit depth limit too small. Data containers like HDF5 and Parquet, which are quite popular in scientific applications, rely on above formats for compression (along with some data shuffling/delta encoding) and so were not tested explicitly. Results for a selection of 80,000 (~ 41GB compressed) randomly selected 2022 APS log files are shown in Fig. 1.

Several interesting trends are observed. It is clear that lzma is still the best available option in terms of compression ratio. Consistent with design expectations, lzma decompression speed is not affected by preset. This suggest a trivial change - use compression level 4+ instead of 2 within SDDSlib for logged data only, improving compression ratio from 8x to 10x with no impact on read performance.

As for other algorithms, a wide distribution is observed with different performance to compression tradeoffs. Overall, several codecs end up Pareto-optimal (lzma - brotli - zstd - lz4) as one exchanges compression ratio for decompression speed. Recalling the performance limitations outlined previously, for read-dominated operations zstd at compression levels 0-3 strikes the best balance, with ~500MB/s of uncompressed data at 6x ratio corresponding nicely to ~120MB/s network throughput of 1Gb/s connection.

Lossy compression

Above methods tried to preserve the full float64 precision of both the value and the timestamp. However, this format is a result of software conversion during logging and does not represent the (significantly lower) underlying device precision. Moreover, for many analyses float64 precision is either excessive or will get lost anyways during neural network processing. Thus, lossy compression can be harmless if fidelity loss can be limited. We first tried downsampling to float32 format, reducing storage requirements for both raw and compressed forms. However, issues were encountered with timestamps - the machine precision of float32 (relative spacing between two closest numbers) is ϵ=1e-7, corresponding to a step of ~ 100s at the current UNIX timestamp of ~1.6B, an unacceptable error. Thus, we only converted value columns. A better way to deal with large offsets is to use specialized scientific lossy compression codecs, like FZP (later discarded as authors recommend against 1D use) and SZ [13]. Lossy compression results on 2022 data are shown in Fig. 2, and indicate that there is little

ISBN: 978-3-95450-231-8 ISSN: 2673-5490 doi: 10.18429/JACoW-IPAC2023-THPL008

to be gained from float32 conversion. SZ however is promising, with 50% size reduction at quite reasonable 1e-5/1e-7 relative/absolute tolerance thresholds. By tuning the exact parameters based on device type, further improvements can likely be achieved.

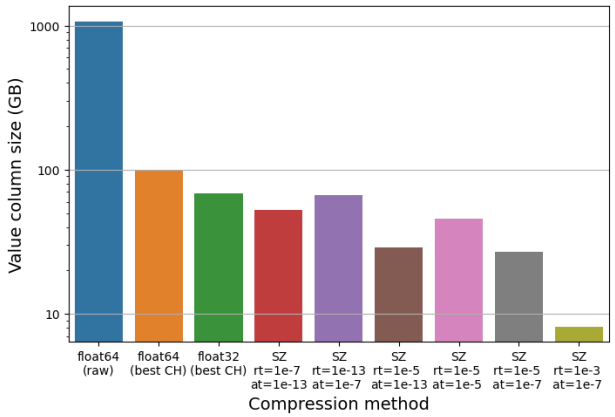

Figure 2: Lossy compression results. Relative and absolute tolerances (rt/at) were used in ABS_OR_REL mode, meaning one but not both had to be satisfied.

ANALYTICS AND DATA PROCESSING

Our second approach was to mirror logger data into a standalone analytics database where it can be analyzed internally. We considered using Archiver Appliance [14], but its' architecture is also not optimized for ML tasks and would put extra load on the IOCs.

With the explosion of 'big data', last decades saw development of specialized analytics databases, both open-source ones like ClickHouse (CH), InfluxDB, and TimescaleDB, as well as commercial ones like kdb+. These databases combine column-oriented storage with compression and advanced query languages. Their use in particle physics was evaluated in 2020 [15], but given our data format and new software features, we did an independent evaluation.

For the 3 open-source databases, we inserted 2022 APS logs while following the suggested format for data layout. InfluxDB demonstrated poor compression efficiency (>500 GB per 10000 devices), and thus was quickly excluded. TimescaleDB insertion performance was unacceptably slow with multiple concurrent writes, and it was also excluded. ClickHouse performed well, and uniquely had per-column storage tuning [16]. We scanned compression parameters on a set of 8000 devices, with results shown in Fig. 3.

Based on the above benchmark, overall configuration was selected as ZSTD(3) for all columns, with DoubleDelta encoding applied for time column. With these settings, all 2022 data was inserted at a total size of 400 GB. Several realistic benchmarks were implemented, mimicking previously discussed ML workloads. Results for a set of 50 random devices are summarized in Table 1. In both cases, execution time included any parsing required to get data into NumPy arrays, and caching was left on.

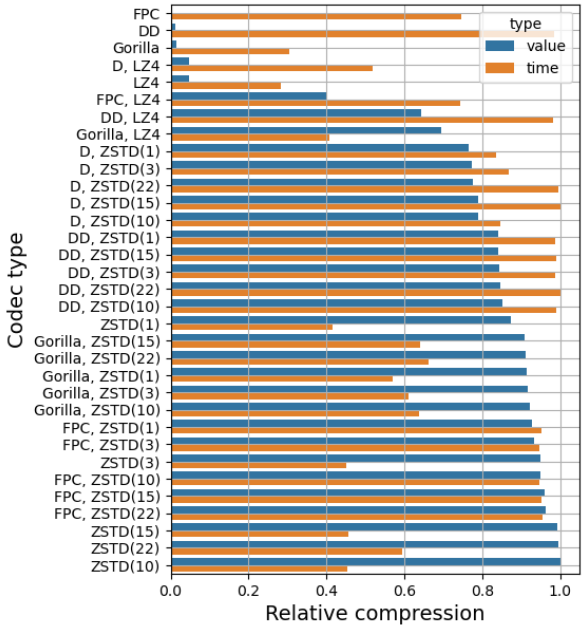

Figure 3: CH relative compression ratio (higher=better) of a representative APS log dataset. Abbreviations: D=Delta, DD=DoubleDelta, rest as defined in CH manual.

Table 1: Benchmark Results for Typical ML Tasks

Task	Direct read	CH read
Read a year of data	91.9s	75.3s
Read specific timestamp	7.0s	9.4s
Yearly average	90.8s	9.3s
Mean absolute deviation	91.7s	9.2s
Overall median (all devices)	26.8s	0.2s

Overall, CH achieves comparable linear and random read performance but wins on tasks where indices/statistics can be leveraged, making it highly suited for analytics.

CONCLUSION

Efficient collection, storage, and analysis of logs is important for both standard and ML-aided accelerator tools. We prototyped potential improvements to the APS logging architecture that enable more data collection and faster analysis. For existing logging, compression ratio could either be increased from 8x to 10x at same speed, or decompression speed increased by 10x for a 20% increase in file size. For standalone tools, by using a modern analytics database ClickHouse we were able to offload and speed up data processing for a number of typical queries. Overall, we believe there is a clear path forward to collect and process an order of magnitude more data in the APS-U era.

ACKNOWLEDGMENTS

Authors acknowledge ANL Laboratory Computing Resource Center (LCRC) for providing the compute resources.

ISBN: 978-3-95450-231-8 ISSN: 2673-5490 doi: 10.18429/JACoW-IPAC2023-THPL008

REFERENCES

[1] M. Borland, "Applications Toolkit for Accelerator Control and Analysis," in *Proc. PAC'97*, Vancouver, Canada, May 1997.

[2] R. Soliday, M. Borland, L. Emery, and H. Shang, "New Features in the SDDS Toolkit," in *Proc. PAC'03*, Portland, OR, USA, May 2003. https://jacow.org/p03/papers/FPAG005.pdf

[3] H. Shang, M. Borland, L. Emery, and R. Soliday, "New Features in the SDDS-Compliant EPICS Toolkit," in *Proc. PAC'03*, Portland, OR, USA, May 2003. https://jacow.org/p03/papers/FPAG004.pdf

[4] M. Borland, "A Self-Describing File Protocol for Simulation Integration and Shared Post-Processors," in *Proc. PAC'95*, Dallas, TX, USA, May 1995.

[5] I. Lobach, M. Borland, K. C. Harkay, N. Kuklev, A. Sannibale, and Y. Sun, "Machine Learning for Anomaly Detection and Classification in Particle Accelerators," in *Proc. NA-PAC'22*, Albuquerque, NM, USA, 2022, pp. 311–314. doi:10.18429/JACoW-NAPAC2022-TUYE4

[6] N. Kuklev, M. Borland, G. I. Fystro, H. Shang, and Y. Sun, "Online Accelerator Tuning with Adaptive Bayesian Optimization," in *Proc. NAPAC'22*, Albuquerque, NM, USA, 2022, pp. 842–845. doi:10.18429/JACoW-NAPAC2022-THXD4

[7] N. Kuklev, Y. Sun, H. Shang, M. Borland, and G. Fystro, "Robust adaptive Bayesian optimization," presented at IPAC'23, Venice, Italy, May 2023, paper THPL007, this conference.

[8] S. Veseli *et al.*, "Data Acquisition System for the APS Upgrade," in *Proc. ICALEPCS'19*, New York, NY, USA, 2020, p. 842. doi:10.18429/JACoW-ICALEPCS2019-TUDPP02

[9] I. Lobach, E. Chandler, H. Shang, N. Arnold, and R. Soliday, "Application for Anomaly Detection in the Storage Ring Power Supplies of APS-U," presented at IPAC'23, Venice, Italy, May 2023, paper THPL006, this conference.

[10] J. Duda, "Asymmetric numeral systems as close to capacity low state entropy coders," *CoRR*, vol. abs/1311.2540, 2013. http://arxiv.org/abs/1311.2540

[11] https://github.com/facebook/zstd.

[12] H. Shang, M. Borland, L. Emery, R. Soliday, and Y. Wang, "Parallel SDDS: A Scientific High-Performance I/O Interface," in *Proc. ICAP'09*, San Francisco, CA, USA, Aug.-Sep. 2009, pp. 347–350. https://jacow.org/ICAP2009/papers/THPSC050.pdf

[13] X. Liang *et al.*, "Sz3: A modular framework for composing prediction-based error-bounded lossy compressors," *IEEE Transactions on Big Data*, vol. 9, no. 2, pp. 485–498, 2023. doi:10.1109/TBDATA.2022.3201176

[14] M. V. Shankar, L. F. Li, M. A. Davidsaver, and M. G. Konrad, "The EPICS Archiver Appliance," in *Proc. ICALEPCS'15*, Melbourne, Australia, Oct. 2015, pp. 761–764. doi:10.18429/JACoW-ICALEPCS2015-WEPGF030

[15] M.-E. Vasile, G. Avolio, and I. Soloviev, "Evaluating influxdb and clickhouse database technologies for improvements of the atlas operational monitoring data archiving," *Journal of Physics: Conference Series*, vol. 1525, no. 1, p. 012 027, 2020. doi:10.1088/1742-6596/1525/1/012027

[16] https://clickhouse.com/docs/en/sql-reference/statements/create/table.