Evaluating the Usage of Python Libraries on a Production Supercomputer

Thomas M. Papka
Loyola University Chicago
Chicago, Illinois, USA
tpapka@luc.edu

Venkatram Vishwanath*
Argonne National Laboratory
Lemont, Illinois, USA
venkat@anl.gov

Filippo Simini*
Argonne National Laboratory
Lemont, Illinois, USA
fsimini@anl.gov

ABSTRACT

Advances in artificial intelligence (AI) and machine learning (ML) are reshaping scientific computing and influencing programming practices on high-performance computing (HPC) systems. We analyze Python library usage on the Polaris supercomputer to understand adoption patterns in modeling, simulation, data analysis, and ML. Using XALT, a runtime monitoring tool, and PySnooper, a lightweight tracer, we correlate library imports with job scheduler data and scientific domains. Results are presented through visualizations and an interactive dashboard, enabling scientists to track usage trends, identify performance impacts from non-optimized environments, and inform improvements to Argonne's default Python stack. This work provides actionable guidance for software provisioning, user support, and infrastructure planning in the era of AI-driven science.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; • **Computer systems organization** → **High-performance computing**.

KEYWORDS

Python, HPC, library usage, Conda, machine learning

ACM Reference Format:
Thomas M. Papka, Venkatram Vishwanath, and Filippo Simini. 2025. Evaluating the Usage of Python Libraries on a Production Supercomputer. In *Proceedings of ACM Student Research Competition (SC '25).* ACM, New York, NY, USA, 2 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

The Argonne Leadership Computing Facility's Polaris system delivers over 44 petaflops of peak performance with 560 NVIDIA A100 GPUs and 2,240 AMD EPYC Milan CPUs. Understanding Python library usage on such platforms is essential for optimizing software environments and supporting increasingly AI-driven workloads.

Our study leverages two complementary tools. **XALT** [1] captures system-wide runtime information, including library paths, executables, and job metadata, providing detailed coverage of nearly every Python job. **PySnooper** [2] offers a lightweight but less comprehensive view of library imports and execution paths. Together, they enable analysis of library adoption across scientific domains.

Argonne's default Python distribution, based on Conda, includes common scientific libraries, but usage logs reveal that many popular packages are missing. By distinguishing users who run directly in the base environment from those who customize it, we highlight opportunities to refine software provisioning and improve efficiency. Ultimately, our goal is to identify widely used libraries and usage patterns that can guide future HPC support and planning.

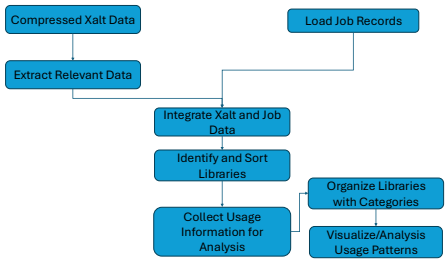

Figure 1: Overview of the workflow used for processing and analyzing XALT data.

2 METHODS

We analyzed XALT logs stored in structured JSON format, ranging from a few hundred megabytes to over 40 GB. To process these efficiently, we parsed the JSON directly rather than loading full dataframes, extracting Python libraries, dynamically loaded libraries, and non-Python objects. We also identified whether users modified the default Conda environment and aggregated results to produce distributions of usage across domains.

XALT collects data for nearly all Python jobs on Polaris, while PySnooper provides a lightweight but less comprehensive view of library imports. Together, they allow detailed tracking of library adoption patterns.

To quantify runtime overhead, we compared library imports with and without XALT enabled (Table 1). The increase was 16% for a single import and 47% for 95 imports.

Table 1: XALT overhead during Python library imports.

Test	Without XALT (s)	With XALT (s)
Single import	1.50 ± 0.12	1.74 ± 0.24
95 imports	12.95 ± 0.89	19.08 ± 9.11

*Advisor

SC '25, November 16 – 21, 2025, St. Louis, MO

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

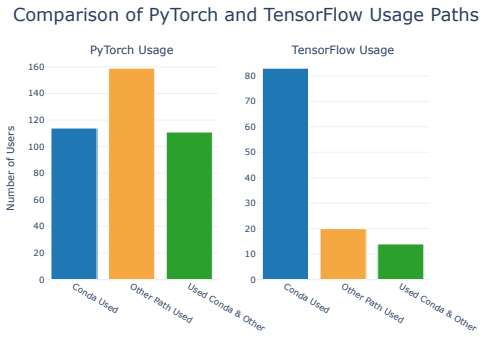

Figure 2: Conda environment usage by unique users. PyTorch users prefer custom environments, while TensorFlow users rely more on the base.

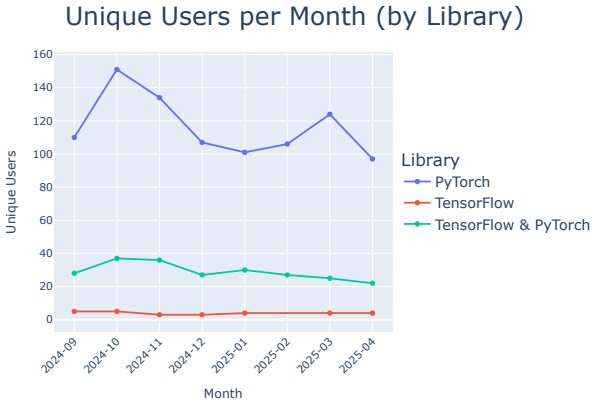

Figure 3: Unique users of PyTorch and TensorFlow. Most users rely on PyTorch exclusively, with few using TensorFlow alone.

Although XALT adds overhead during imports, the effect is concentrated at startup and remains small relative to the runtime of large HPC workloads, making it practical for system-wide monitoring.

3 RESULTS

Figure 2 compares Conda environment usage for PyTorch and TensorFlow users. PyTorch users mostly rely on custom environments (160), with 110 using the base or both. TensorFlow users are more dependent on the base environment (80), with fewer adopting custom setups.

We also examined the origin of library paths. Most libraries were loaded from home directories, a practice that can reduce performance.

Figure 3 shows unique users of PyTorch and TensorFlow. PyTorch dominates overall usage, with a smaller but consistent group running both, and few using TensorFlow alone.

Finally, we compared usage of five widely adopted libraries (NumPy, Zstandard, Cryptography, PIL, and YAML) with total unique Python users (Figure 4). Their adoption closely follows

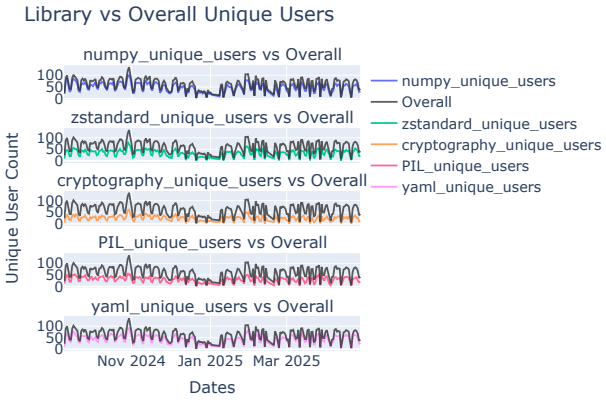

Figure 4: Unique users of five popular Python libraries compared with total Python users on Polaris. Trends closely track overall system usage.

overall Python usage, indicating that they are broadly important across scientific domains.

4 CONCLUSION

Our analysis reveals clear patterns in Python library usage on the Polaris supercomputer. Core scientific and AI libraries dominate across domains, with PyTorch showing broader adoption than TensorFlow. Monitoring these trends provides guidance for refining the default Conda environment and supporting more efficient scientific workflows. XALT's system-wide coverage makes it a practical tool for capturing usage while introducing only modest overhead. By connecting usage trends to system planning, our study underscores the role of software analytics as a key component of HPC operations.

5 FUTURE WORK

We plan to extend this study by linking library usage to runtime and job sizes, tracking emerging packages over time, and examining libraries loaded outside the base Conda environment. Additional goals include identifying factors driving library popularity and developing an interactive dashboard for daily usage exploration on Polaris.

6 ACKNOWLEDGMENTS

This research used resources of the Argonne Leadership Computing Facility, a U.S. Department of Energy (DOE) Office of Science user facility at Argonne National Laboratory, and is based on research supported by the U.S. DOE Office of Science-Advanced Scientific Computing Research Program, under Contract No. DE-AC02-06CH11357.

REFERENCES

[1] Kapil Agrawal, Mark R. Fahey, Robert McLay, and Doug James. 2014. User Environment Tracking and Problem Detection with XALT. In *2014 First International Workshop on HPC User Support Tools*. 32–40. https://doi.org/10.1109/HUST.2014.6

[2] Misha Salim and Kyle Felker. 2025. PyModuleSnooper. https://github.com/Argonne-lcf/PyModuleSnooper/commits?author=felker. GitHub repository.