ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-WEMG015

UPGRADING THE ATLAS TUNE ARCHIVING SYSTEM*

K. Bunnell†, M. Torres, A. Ramaswamy, B. Blomberg, C. Dickerson,
G. Dunn, D. Novak, D. Stanton
Argonne National Laboratory, Lemont, IL, USA

Abstract

The Argonne Tandem Linear Accelerating System (ATLAS) is a U.S. DOE national user facility that delivers stable and radioactive ion beams from hydrogen to uranium for low-energy nuclear physics research [1]. Operators routinely expedite setup by restoring previously optimized machine parameter sets ("tunes"). The legacy tune archiving system, implemented in Corel Paradox (1999), has become a maintenance and operational bottleneck due to recurrent table corruption, single-user access, limited integration, and proprietary language.

We present the ATLAS Time Machine (ATM), a modern replacement comprised of PySide6 for the UI, FastAPI for backend services, MariaDB for experiment metadata, and InfluxDB v2 for time-series device data. ATM supports multi-user access, direct integration with the ATLAS control system, and automated beamline-aware data collection based on a dynamically generated beam path view. Initial results from beta operations indicate improved reliability, streamline operator workflows, and more convenient operator access. We conclude with lessons learned and a roadmap toward full production deployment.

INTRODUCTION

Experiments at ATLAS typically start every 3-5 days with beam being created at one of three ion sources and being delivered to one of nine target areas. Beams are on target 24 hours a day, seven days a week excluding tuning and down time. At the start of an experiment, the accelerator requires operators to tune the accelerator for the specific ion species used. During this process operators must find the optimized value for all beam influencing devices to deliver beam to the target efficiently. The faster the beam can be tuned onto the target, the more hours that are able to go towards the experiment.

One way to improve the speed of the tuning process is to load values from a previous experiment that is similar to the current one and attempt to improve the tune from there. The tunes still require minor tweaks to improve transmission which takes 24 hours on average.

The current solution to this is the Tune Archiving System (TAS) [2]. TAS utilizes legacy software, is difficult to maintain, and has limitations due to the software and programming practices used. Due to these limitations, a replacement for the old TAS is being developed that uses Python to make the solution modern, simpler to maintain, and lower the learning curve.

* This work was supported by the U.S. Department of Energy, Office of Nuclear Physics, under Contract No. DE-AC02-06CH11357. This research used resources of ANL's ATLAS facility, which is a DOE Office of Science User Facility.
† kbunnell@anl.gov

ATLAS CONTROL SYSTEM

Real-time control of the accelerator is actively being upgraded from a legacy system to an upgraded system both using Vista Controls VSystem software as shown in Fig. 1. The legacy system consists of a Digital Equipment Corporation AlphaServer that relies on a CAMAC (Computer Automated Measurement and Control) serial highway to control and monitor devices. This system supports an Oracle Relational Database (RDB) that allows systems outside of VSystem to read control system data including TAS and websites [3].

The upgraded modern hardware consists of 14 Linux servers. Figure 1 shows that each computer includes VSystem and the drivers required to monitor and control devices using Ethernet [4], USB, or VME. ATM makes use of MariaDB to hold many VSystem channels commonly used amongst the programs at ATLAS. Device data is stored in InfluxDB so that historical trends can be displayed in Grafana.

This upgrade approach allows for a slow migration of beamline devices and software systems to move from the old hardware to more modern hardware.

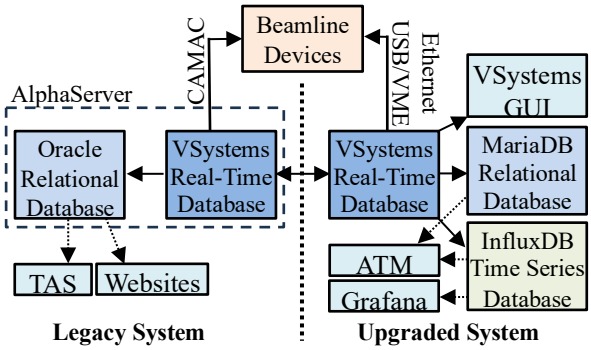

Figure 1: ATLAS Control System configuration. Upgraded system summarizes network of 14 Linux-based computers.

The MariaDB contains several tables that include data about each device on a small section of beamline. At the start of each experiment, the tables determined to be on the beampath are automatically combined into an SQL view that describes the entire beamline from source to target. This view simplifies the code that determines the devices on the beamline and their order. The beampath table encodes a globally unique, order-preserving device id and several other tags to enable efficient sorting and filtering.

MOTIVATION

TAS was developed in 1999 and uses Corel's Paradox, a relational database management software which includes a UI designer and integrated programming language.

ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-WEMG015

Paradox is a part of the WordPerfect package from Corel and has not seen a software patch since 2009 [5].

ATLAS operators experience Paradox table corruption weekly. When table corruption occurs, the operators are forced to stop TAS and rebuild the tables before continuing. Typically, corruption is seen on multiple tables simultaneously, accounts for a minor loss of data, and takes 5 minutes on average to recover from with the help of 3rd party software that speeds up this process.

Paradox also relies on a local database architecture that prevents users from accessing the database tables simultaneously. This locks Paradox into a setup where the data can only be viewed by one user at a time which creates a bottleneck during tune days where operators might wait up to 30 minutes for access when other operators need to analyze data in TAS.

The TAS local database structure also complicates the data upload process. Since the TAS database is not accessible while in use, it means that TAS must pull data. This is accomplished using a process that copies VSystems data into Oracle RDB. TAS then reads the data from Oracle RDB as shown in Fig. 1. Different parts of control system data are updated to the RDB table at different frequencies between 1-3 minutes. Saved data in Paradox is often 3 minutes old. Data is allowed to be 15 minutes old before operators receive a warning message.

TAS has grown difficult to maintain due to extensive duplication, deeply nested conditionals, and code organization practices that make code comprehension difficult. In addition, skills required to maintain TAS do not transfer well to other applications in use by ATLAS. The integrated programming language, ObjectPAL, is specific to Paradox and documentation is difficult since it predates the internet. ObjectPAL code can also be organized into libraries, forms, or elements within a form. This can create a complex hierarchy that impedes maintainability and traceability. Additionally, updates of TAS are only made a few times a year making it difficult for programmers to remember the important details to make updates.

TOOLS USED

ATM consists of 4 publicly available tools that can be upgraded independently should any piece become obsolete. InfluxDB V2.7 is the time-series database used to store all the historical data within ATM. MariaDB V10.3 stores a list of experiments and saves. The server uses FastAPI V0.116.1 to communicate with clients. PySide6 V1.2.4 is the UI. Each part was analyzed and determined to have a good chance of long-term support at the time of publishing.

Client/server communication uses a RESTful API rather than gRPC to prioritize simplicity, transparency, and language-agnostic interoperability via JSON. This makes messages easy to inspect and debug, which is valued over the potential throughput gains of gRPC for small payloads. We implement the API with FastAPI for its strong ecosystem, automatic documentation, and solid performance.

The GUI is built with PySide6 because it is well supported, documented in open sources, and backed by the Qt project, offering good long-term prospects. Its close API parity with PyQt provides an easy migration path if needed, and QML support helps separate presentation from application logic.

DESIGN OVERVIEW

ATM consists of 4 main pieces as shown in Fig. 2. The data collection process, the new experiment process, ATM server, and clients (UI).

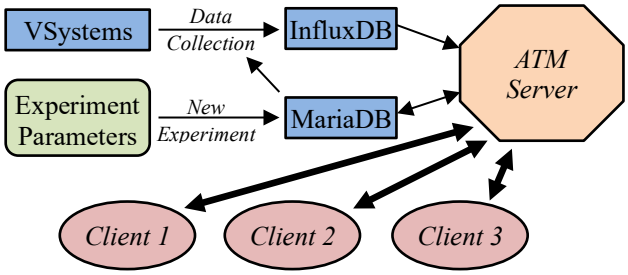

Figure 2: Flowchart of ATM. Pieces of ATM are shown with italics.

New Experiment

At the start of each experiment, operators enter details in the control system and submit them to MariaDB. Because comments reference each entry via a primary–foreign key, a unique ID is required. The tool auto-generates this ID on submission and pre-fills fields with the most recently inputted parameters, saving operators over 10 minutes on tune days when rerunning repeated setups.

Data Collection

The data collection process gets the beam path devices from MariaDB, then reads the data for those channels from the ATLAS Control System and uploads the data to InfluxDB with various tags. To monitor and prevent any missing data, the data collection process sends metadata to InfluxDB that includes the total time took to read all data from each database, the number of channels read from each source, and the number of channels that failed to be read. This can assist in troubleshooting and allow operators to self-identify and diagnose issues with ATM. Data is saved every 5 minutes, or by operator request. The data is retained indefinitely.

After two months of operation with metadata enabled, the collector averages 3.87 seconds to read 779 channels across 165 devices from the control system and write the values to InfluxDB. The metadata indicates that VSystem is the most sensitive to interruptions, with delays frequently coinciding with control system maintenance windows. To reduce manual intervention, the process automatically resets database connections when they drop.

Server

The server was designed to handle the bulk of the logic in ATM. It is responsible for querying InfluxDB and MariaDB, organizing the data, and sending the data to the clients using FastAPI. The server runs continuously but only serves the ATM clients with data they request. The server

ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-WEMG015

allows multiple clients to connect simultaneously and requires lab credentials when accessing the server from outside the lab. For more information about the server, see [6].

User Interface

The UI is designed for simplicity and low maintenance. The client retrieves data from the server, while most sorting and filtering is done server-side; the server returns ready-to-render results. The client runs on Windows, Linux, and macOS. Because installation is not yet streamlined, a Controls Group member must perform the install.

Figure 3 shows the layout: operators select experiments in the left sidebar, and the right pane displays parameters, comments, and data. The device list is a dynamic, horizontal scrollable list of all devices on the beamline, showing values at time posted on the selected comment. Devices are ordered from entries in MariaDB, so adding or reordering beamline devices requires only a server-side change; clients pick up the change automatically. For focused data viewing, the experiment selector can be collapsed. For more on the UI, see [7].

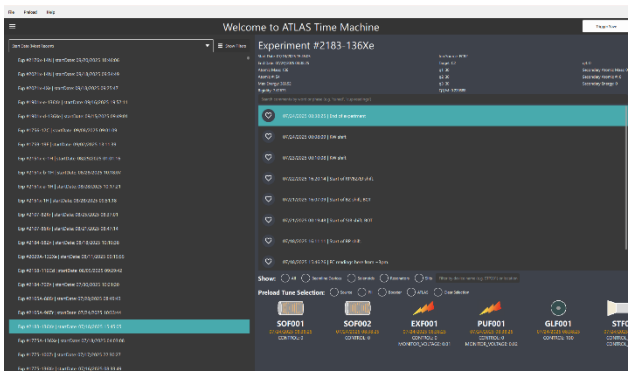

Figure 3: ATM client: left pane selects experiments; right pane shows parameters, data, and comments.

DEVELOPMENT TEAM

ATM was executed as a student-led effort over the course of two summers with clear goals, guided autonomy, and regular mentorship. A returning student built the server, evaluated REST vs gRPC, helped select FastAPI, and implemented the API and streaming features based on MariaDB/InfluxDB schemas and expected client requests, coordinating on a shared protocol. A second student with strong Python experience developed the client in PySide6 using the server protocol and operator-informed UI mockups, engaging operators directly for rapid iteration. Starting progress with the server led to fast client development, though concurrent timelines might have improved efficiency. The two students collaborated closely, with check-ins about three times per week to maintain momentum and keep meetings focused, yielding a working product by the end of the second summer. Code reviews and AI-generated manuals supported the handover, easing the transition of maintenance to the Controls Group.

IMPROVEMENTS

User Experience

ATM offers an advanced sorting and filtering tool to help operators find desired experiments. Operators can sort based on any experiment parameters. Paradox offered sorting but lacked any filtering ability. For more on filtering, see [7].

ATM features rich-text comments that can bookmark preferred tunes. To promote good lab practice, the comment editor is append only: the original comment cannot be altered; instead, you add a timestamped note to it. The original and every appended note carry their own timestamps, creating a clear, chronological audit trail of the reasoning behind each comment.

ATM saves data automatically at a higher frequency than TAS since corruption is no longer a limitation. This captures states without operator intervention, so they no longer need to babysit saves or worry about missing a snapshot. Both ATM and TAS allow comments to be added to past timestamps, but the key improvement is ATM's more frequent, reliable saves, reducing operator workload and letting them focus on tuning rather than managing saves.

ATM and TAS are not directly comparable on latency because of their different architectures, so comparisons should be made by task. At initial launch, TAS starts in about 3 seconds, whereas ATM often takes more than 20 seconds depending on the machine. For similar pages, TAS is typically less than one second faster than ATM. Despite ATM's heavier startup and slightly slower per-page interactions, operators can complete key tasks faster in ATM because its search utility makes it quicker to find suitable experiments and load the corresponding beamline configurations, improving overall time to result.

Maintenance

ATM leverages existing tools at ATLAS including InfluxDB [8], MariaDB, Python and C. Upgrading to more common tools lowers the threshold for maintaining TAS.

Hardcoded VSystem channels in TAS have been replaced by channels pulled from the MariaDB, so only a single source needs to be updated when changing VSystem channels. Likewise, the beamline device list is also sourced from MariaDB, so adding or removing devices requires only a database change, no UI changes.

CURRENT PROGRESS

ATM is in beta use by operators. Core features, structure, sorting, and filtering are stable; however, tune loading into the accelerator is not yet enabled. The plan is to keep features critical to operation while removing features that were poorly implemented in TAS.

FUTURE GOALS

Near-term priorities to improve usability include simplifying client installation, implementing client versioning with enforced updates to prevent legacy clients, and adding role-based permissions to limit or moderate comments.

ISBN: 978-3-95450-255-4 ISSN: 2226-0358 doi: 10.18429/JACoW-ICALEPCS2025-WEMG015

Long-term features that are desired include storage and retrieval of beam current and beam shape from faraday cups and beam profile monitors respectively.

CONCLUSION

ATM replaces the obsolete, single-user TAS system with a modular, easy to maintain platform integrated with the upgraded ATLAS control system. Early beta results show improved reliability, multi-user access, automated snapshots, and richer search/comment workflows that streamline tune selection, with acceptable latency trade-offs.

ACKNOWLEDGEMENTS

We thank Chis Roderick and Stephane Deghaye for meeting with us to share their expertise in Python and Python-based user interfaces and for offering thoughtful suggestions on the design of ATM. Their insights materially improved this work.

REFERENCES

[1] Argonne Tandem LINAC Accelerator System Available Beams, https://www.anl.gov/atlas/available-beams

[2] F. Munson and D. Quock, "Argonne National Laboratory ATLAS Accelerator tune archiving system", in *Proc. PCaPAC'00*, Hamburg, Germany, Oct. 2000.

[3] F. H. Munson, D. E. R. Quock, S. L. Dean, and K. J. Eder, "The Relational Database Aspects of Argonne's ATLAS Control System", in *Proc. ICALEPCS'01*, San Jose, CA, USA, Nov. 2001, paper WEAP066, pp. 401-403.

[4] K. J. Bunnell, C. Dickerson, D. J. Novak, and D. Stanton, "Exploring Ethernet-Based CAMAC Replacements at ATLAS", in *Proc. ICALEPCS'23*, Cape Town, South Africa, Oct. 2023, pp. 1542-1544. doi:10.18429/JACoW-ICALEPCS2023-THPDP081

[5] WordPerfect Office Patches and Updates, https://kb.corel.com/en/127502

[6] M. Torres *et al.*, "A server for the ATLAS time machine", presented at ICALEPCS'25, Chicago, IL, USA, Sep. 2025, paper THPD097, this conference.

[7] A. Ramaswamy *et al*., "A client for the ATLAS time machine", presented at ICALEPCS'25, Chicago, IL, USA, Sep. 2025, paper THPD110, this conference.

[8] D. J. Novak, K. J. Bunnell, C. Dickerson, and D. Stanton, "Teaching an Old Accelerator New Tricks", in *Proc. ICALEPCS'23*, Cape Town, South Africa, Oct. 2023, pp. 1545-1546. doi:10.18429/JACoW-ICALEPCS2023-THPDP082