Modernizing ATLAS PanDA for a sustainable multi-experiment future

Tatiana Korchuganova[3,*], *Fernando Harald* Barreiro Megino[2], *Kaushik* De[2], *Wen* Guan[1], *Edward* Karavakis[1], *Alexei* Klimentov[1], *Fa-Hui* Lin[2], *Tadashi* Maeno[1], *Paul* Nilsson[1], *Torre* Wenaus[1], *Zhaoyu* Yang[1], and *Xin* Zhao[1] on behalf of the ATLAS Computing Activity

[1]Brookhaven National Laboratory, Upton, NY, USA
[2]University of Texas at Arlington, Arlington, TX, USA
[3]University of Pittsburgh, Pittsburgh, PA, USA

Abstract. In early 2024, ATLAS undertook an architectural review to evaluate the functionalities of its current components within the workflow and workload management ecosystem. Pivotal to the review was the assessment of the Production and Distributed Analysis (PanDA) system, which plays a vital role in the overall infrastructure. The review findings indicated that while the current system shows no apparent signs of scalability limitations or critical defects, several issues still require attention. These include areas for improvement, such as cleaning the historical accumulation of code over nearly two decades of continuous operation in ATLAS, further organizing development activities, maximizing the utilization of continuous integration and testing frameworks, bolstering efforts toward cross-experimental outreach, spreading greater awareness of workflows at the core level, expanding support for complex workflows, implementing a more advanced algorithm for workload distribution, optimizing tape and network resource usage, refining interface design, enhancing transparency to showcase system dynamism, ensuring allocation of key developers to R&D projects with clear long-term visions for integration and operation, and accommodating the growing diversity of resources. In this paper, we first highlight the issues identified in the review, exploring their historical and cultural roots. Then, we outline the recommendations derived from the review, and present the solutions developed to address these challenges and pave the way to sustainably support multiple experiments.

ATL-SOFT-PROC-2025-021
31 January 2025

1 Introduction

The Production ANd Distributed Analysis (PanDA) system [1] has been the primary workflow management system (WMS) for the ATLAS experiment [2] since 2007, continuously evolving to meet the demands of large-scale data processing and distributed computing. Designed to manage millions of computational jobs across a diverse set of resources, PanDA has adapted to operate seamlessly on various platforms, including WLCG Grid resources [3], commercial cloud services [4], and high-performance computing (HPC) facilities [5]. Also,

*e-mail: tatiana.korchuganova@cern.ch

PanDA accommodates various job requirements such as multiprocessing, multithreading, and jobs with varying memory needs. In recent years, the system has expanded to support a wide range of processors, including ARM architectures, and has integrated the use of accelerators like GPUs to optimize performance across heterogeneous computing environments.

In early 2024, the ATLAS experiment conducted an architectural review to evaluate the functionality and performance of its workflow and workload management system. While no critical defects or immediate scalability limitations were identified, several areas for improvement were highlighted:

- Project Management: adopt a more structured, strategic approach to project management, ensuring efficient resource allocation and goal tracking.
- Code Modernization: streamline and update legacy code accumulated over nearly two decades to enhance maintainability and performance.
- Continuous Integration: expand the use of continuous integration (CI) and testing frameworks to improve system reliability and accelerate development cycles.
- Workflow Awareness: integrate advanced workflow awareness into the system core to enhance decision-making and resource allocation.
- User Interface and Dynamic Monitoring: develop a more intuitive and accessible GUI to improve usability across diverse user groups and to better visibility into system operations with the near real-time monitoring and enhanced responsiveness.
- Sustainable R&D: prioritize R&D projects with clearly defined long-term integration and operational goals to support sustainable system evolution.

These recommendations aim to ensure that the PanDA system remains robust, scalable, and adaptable to future challenges such as the scale of the High-Luminosity LHC [6].

We introduced a new project management structure to streamline PanDA system development in a form of four key streams. Each stream is led by a dedicated manager who oversees task assignments, timelines, and progress. Streams cut across all components to enhance the mobility of developers and ensure a more flexible and integrated development process. These aspects will be discussed in detail in the following sections of this paper.

2 Stream 1: Code-base optimization and modernization

PanDA has operated continuously for almost 20 years without a single declared downtime for code updates by delivering backward compatible changes and ensuring transparent transitions. Downtime occurs only during unavoidable database interventions or broader outages within CERN's infrastructure, where the PanDA system is deployed. The continuous delivery of features and the age of the project, however, have introduced complexities into the codebase and it would be useful to transition to more state-of-the-art libraries. Recognizing the need for modernization, we are undertaking a comprehensive effort to enhance our system, improve workflows, and adopt contemporary practices.

We aim to improve our testing infrastructure significantly. Currently, developers rely on independent development machines, but we plan to establish a staging environment for validating new releases. This environment will be extended with CI/CD capabilities and deployed on Kubernetes [7], providing us with hands-on experience in operating Kubernetes-based services. Such preparation will provide experience if we decide to migrate our production infrastructure during a long shutdown.

A key priority is simplifying and modernizing our codebase. Focused on the oldest modules, we are replacing obsolete technologies such as XML and `pickle` with JSON and adopting modern libraries like `requests` in place of `curl` or `pycurl`. Additionally, we are rewriting our API to adhere to standards and publishing its documentation using OpenAPI [8]. Obsolete code and unused database tables are being removed, reducing the complexity of the environment and the total number of lines of code (fig.1).

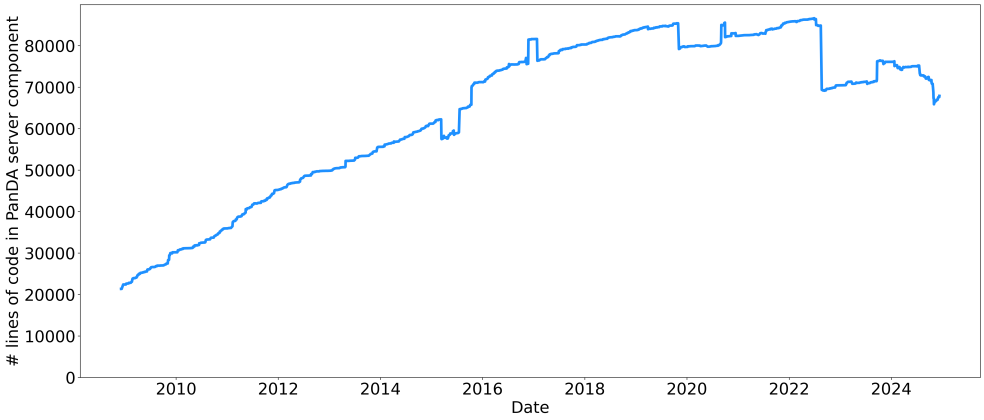

Figure 1. Evolution of lines of code number in PanDA server component over time

Most components have fully transitioned to Python 3, allowing us to leverage its advanced features, such a f-strings and type hints. Python 2 compatibility has been dropped, and we have integrated tools like pre-commit hooks to automatically format code using utilities like `Black` [9], `Flynt` [10], and `isort` [11]. This ensures a consistent and modern codebase.

We have reduced reliance on external agents and packages by incorporating them into our main codebase. This change not only simplifies operations but also reduces the number of machines required, optimizing resources. To monitor service health, we introduced email and Mattermost [12] notifications for immediate awareness of availability issues. These alerts have enabled to debug issues on the spot, while they are still happening and in the same conditions, instead of imagining some time later that there could have been a memory or a network issue. Since the introduction of the alerts, we have significantly improved our service quality.

Improved notifications also extend to team updates: whenever a team member updates PanDA packages on a production machine, a Mattermost alert informs the rest of the team. Combined with availability notifications, this ensures full infrastructure oversight. To align with best practices, we are moving from informal communication methods like emails and chats to a ticket-based system. With JIRA [13], we have automated processes such as closing inactive tickets after six months and closing resolved tickets after two weeks.

These updates reflect our commitment to modernizing PanDA's infrastructure, operations and codebase, while continuing to deliver new features and services regularly.

3 Stream 2: System optimization and enhancement

PanDA faces several key challenges that require optimization in site definitions, job generation, workflow management, and resource allocation. Enhanced transparency, automa-

tion, and scalability are essential for addressing these limitations and supporting the system's growing complexity.

The relationship between PanDA queues and ATLAS sites reflects a foundational aspect of the ATLAS workload management system. PanDA queues, introduced as an early abstraction layer, were later used as the foundation for defining ATLAS sites. To maintain consistency, PanDA queue objects are retrieved from the Computing Resource Information Catalogue (CRIC) [14] and subsequently used to construct ATLAS site objects. However, this indirect linkage introduces several constraints, limiting flexibility in data management and resource allocation. Addressing these constraints requires refactoring the internal object structures and enabling more granular configurations of PanDA queues, which would better support diverse storage requirements.

Job Execution and Definition Interface (JEDI) [1] experiences notable delays during job generation, with approximately 1,000 jobs requiring several dozen seconds to create. The main sources of this slowness include file replica lookups during the brokerage process, job object creation, and database record insertions. To mitigate these delays, introducing caching for frequently accessed data, optimizing database transactions, and implementing more efficient threading strategies could improve performance.

Current workflow management within Database Engine For Tasks (DEFT) [15] has limited flexibility in workflow descriptions, lacking support for branching, looping or conditional execution. This restricts the ability to optimize task orchestration, such as delaying merge tasks until they are necessary. Incorporating a more expressive workflow engine capable of handling these advanced patterns would significantly improve task orchestration.

The brokerage algorithm employs a simplified approach, reducing multi-dimensional weight calculations to a single scalar value. While effective in many cases, this approach could be further optimized to better balance factors such as input data location, site performance, and resource availability. For instance, giving greater weight to input data location might reduce unnecessary data transfers, particularly for low-priority jobs where waiting for an optimal site is acceptable. Refining the brokerage algorithm by adopting a multi-dimensional scoring approach and dynamically adjusting weights based on operational priorities could optimize resource allocation and reduce unnecessary overhead.

Inaccurate resource definitions during task submission often lead to inefficiencies. Tasks requesting more memory or resources than required disrupt the brokerage process, necessitating corrective actions through scout jobs. Grouping tasks by request and tag during submission could allow shared resource evaluations, reducing the overhead associated with scout jobs. Moreover, automating the detection and definition of retry rules for failed jobs using machine learning or pattern recognition could minimize human intervention and improve recovery efficiency.

The lack of comprehensive system-wide metrics hampers effective workload management and resource planning. While individual components such as Harvester [16] provide detailed metrics, aggregated metrics at the system level, such as task time to completion (TTC), network utilization, and fairness across stakeholders remain underdeveloped. For network usage, optimizing data transfers based on priority and utilizing predictive analytics for data staging could reduce load without negatively impacting TTC or other critical metrics.

PanDA and JEDI were initially designed for ATLAS workloads requiring [1-core, 8-core] CPUs with [2GB/core, 4GB/core] memory configurations. However, growing resource heterogeneity, including the integration of GPUs and workloads with high memory or scratch disk requirements, challenges the original system design. Establishing resource-specific matchmaking strategies and updating scheduling algorithms to accommodate multi-GPU configurations and varying memory requirements would ensure efficient utilization of hetero-

geneous resources. Encouraging community discussions on workload priorities and resource allocation policies would further align system design with emerging needs.

4 Stream 3: Outreach

Outreach and community engagement have been pivotal components of the PanDA WMS strategy throughout 2024. Since its foundation, PanDA has been an open-source software project [17], welcoming contributions from outside the core team. Recognizing the importance of fostering a vibrant and collaborative user base, the PanDA team has initiated several efforts aimed at strengthening ties with current users and encouraging new collaborations.

A major highlight has been the establishment of the PanDA Community Forum, which serves as a platform for open discussions, information exchange, and collaboration among various experiments utilizing or evaluating PanDA. The forum has brought together representatives from ATLAS [2], COMPASS [18], the Electron-Ion Collider (EIC) [19], DarkSide-20k [20], and the Vera Rubin Observatory [21]. To date, nine Community Fora have been held, consistently attracting participation ranging from 11 to 24 attendees per session. These meetings have facilitated topical discussions, experiment-specific updates, and valuable feedback, enabling the PanDA team to tailor developments to the community's needs. To enhance accessibility and transparency, the PanDA team has circulated meeting minutes and created a Google Group for announcements, ensuring that stakeholders remain informed. Furthermore, plans are underway to organize a half-day workshop, potentially sometime in 2025, to deepen engagement and collaboration.

In addition to fostering community dialog, the team has focused on improving user interaction and documentation. A revamped website at https://pandawms.org, updated documentation [22] powered by Read the Docs [23], and new journal publications [1] have ensured that users have access to clear and comprehensive resources. Initiatives such as the "Smiley Task Feedback" feature have been implemented to gather user insights, further enhancing the usability of the system.

Looking forward, the PanDA team aims to continue these community-building efforts, expand outreach activities, and enhance training materials. By actively engaging our user base and responding to feedback, PanDA is positioning itself as a robust and adaptable solution for diverse workflow management needs.

5 Stream 4: Interactive and dynamic workflow-oriented platform

The current PanDA monitoring system, BigPanDAmon, is a Django-based [24] web application that provides comprehensive insights into the PanDA WMS, ranging from high-level summaries to detailed computational job logs. Over time, it has evolved into a modular platform [25] capable of integrating external monitoring components, such as ATLAS Athena Nightlies CI monitoring [26].

The review identified the need for greater dynamism and interactivity in the monitoring interface, along with the ability to submit user tasks and workflows directly through the GUI. The current reliance on Server-Side Rendering (SSR) using Django's built-in template engine, `jQuery` [27], and `AngularJS` [28] limits the application's interactivity and scalability. To address these limitations, we explored modern frontend frameworks that support Client-Side Rendering (CSR). This shift highlighted the necessity of decoupling the frontend from the backend, which requires a redesign of the backend architecture as well to implement a fully separated REST API. The importance of this transition is supported by the fact that more than half of requests to BigPanDAmon retrieves data in JSON format.

We tested the two most popular frontend frameworks, React [29] and Angular [30], by creating a simple application that implements the full cycle of the create-read-update-delete (CRUD) functionality for an artificial task object. This application was built with a Django-based backend and two separate frontends, one using React and the other Angular, to evaluate their capabilities for our needs. Angular seemed a more sophisticated option for the new platform as it provides a comprehensive, out-of-the-box solution with a strict enforced structure, ensuring separation of components, services, HTML, and TypeScript [31] code. Its built-in testing tools further streamline development. While Angular has a steep learning curve due to its core concepts, TypeScript, routing, and state management, it becomes more manageable over time. In contrast, React offers greater flexibility but relies heavily on external libraries to create a complete project.

We are at the initial stages of developing the new platform application, laying the foundation for a more modern and robust system. From the beginning, we are going to incorporate code quality tools like `pylint` [32] and `black` [9] to ensure consistency and maintainability throughout the development process. Our primary focus is on implementing the authentication and authorization service. Following this, we will develop an interactive GUI for user analysis task and workflow submission and monitoring. Once these core features are in place, we plan to migrate existing modules from the BigPanDA monitoring to the new platform incrementally.

6 Summary

The PanDA WMS, vital to the ATLAS experiment for almost 20 years, is evolving to meet growing distributed computing demands. A 2024 architectural review highlighted the need for code modernization, enhanced interactivity, improved project management, and sustainable R&D. Key efforts include transitioning to modern libraries, adopting CI/CD pipelines, optimizing job generation and resource allocation, and fostering collaboration through community forums and improved documentation. A new interactive monitoring platform is being developed using Angular with a decoupled Django-based backend, starting with a GUI for user analysis task submission and monitoring. This systematic modernization ensures PanDA will remain robust and scalable for the High-Luminosity LHC and beyond.

References

[1] T. Maeno, A. Alekseev, F.H. Barreiro Megino, K. De, W. Guan, E. Karavakis, A. Klimentov, T. Korchuganova, F. Lin, P. Nilsson et al., PanDA: Production and Distributed Analysis System, Computing and Software for Big Science **8**, 4 (2024). 10.1007/s41781-024-00114-3

[2] ATLAS Collaboration, The ATLAS Experiment at the CERN Large Hadron Collider, J. Inst. **3**, S08003 (2008). 10.1088/1748-0221/3/08/S08003

[3] Worldwide LHC Computing Grid (WLCG), https://wlcg.web.cern.ch/, accessed 13 Dec 2024.

[4] F.B. Megino et al., Accelerating science: The usage of commercial clouds in ATLAS Distributed Computing, EPJ Web Conf. **295**, 07002 (2024). 10.1051/epjconf/202429507002

[5] P. Nilsson, A. Anisenkov, D. Benjamin, W. Guan, T. Javurek, D. Oleynik, Harnessing the power of supercomputers using the PanDA Pilot 2 in the ATLAS Experiment, EPJ Web Conf. **245**, 03025 (2020). 10.1051/epjconf/202024503025

[6] I. Zurbano Fernandez et al., High-Luminosity Large Hadron Collider (HL-LHC): Technical design report, **10/2020** (2020). 10.23731/CYRM-2020-0010
[7] Kubernetes - Production-Grade Container Orchestration, https://kubernetes.io/, accessed 13 Dec 2024.
[8] OpenAPI Specification, https://spec.openapis.org/oas/latest.html, accessed 13 Dec 2024.
[9] Black - The Uncompromising Code Formatter, https://github.com/psf/black, accessed 13 Dec 2024.
[10] flynt - string formatting converter, https://pypi.org/project/flynt/, accessed 13 Dec 2024.
[11] isort, https://pycqa.github.io/isort/, accessed 13 Dec 2024.
[12] Mattermost, https://mattermost.com, accessed 13 Dec 2024.
[13] Jira - Issue & Project Tracking Software, https://www.atlassian.com/software/jira, accessed 13 Dec 2024.
[14] A. Anisenkov et al., CRIC: Computing Resource Information Catalogue as a unified topology system for a large scale, heterogeneous and dynamic computing infrastructure, EPJ Web Conf. **245**, 03032 (2020).
[15] M. Borodin, et al., The ATLAS Production System Evolution: New data processing and analysis paradigm for the LHC Run2 and High-Luminosity, Journal of Physics: Conference Series **898**, 052016 (2017). 10.1088/1742-6596/898/5/052016
[16] T. Maeno, F.H. Barreiro Megino, D. Benjamin, D. Cameron, J.T. Childers, K. De, A. De Salvo, A. Filipcic, J. Hover, F. Lin et al., Harvester : an edge service harvesting heterogeneous resources for ATLAS, EPJ Web Conf. **214**, 03030 (2019). 10.1051/epjconf/201921403030
[17] PanDA WMS GitHub project, https://github.com/PanDAWMS, accessed 13 Dec 2024.
[18] P. Abbon et al., The COMPASS experiment at CERN, Nuclear Instruments and Methods in Physics Research, Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **577**, 455 (2007).
[19] F. Willeke, J. Beebe-Wang, Electron ion collider conceptual design report 2021 (2021). 10.2172/1765663
[20] C.E. Aalseth et al. (DarkSide-20k), DarkSide-20k: A 20 tonne two-phase LAr TPC for direct dark matter detection at LNGS, Eur. Phys. J. Plus **133**, 131 (2018), 1707.08145. 10.1140/epjp/i2018-11973-4
[21] E. Karavakis et al., Integrating the PanDA Workload Management System with the Vera C. Rubin Observatory, EPJ Web of Conf. **295**, 04026 (2024). 10.1051/epjconf/202429504026
[22] PanDA documentation, https://panda-wms.readthedocs.io/en/latest/index.html
[23] Read the Docs: Full featured documentation deployment platform, https://about.readthedocs.com, accessed 13 Dec 2024.
[24] Django Framework, https://www.djangoproject.com
[25] T. Korchuganova, A. Alekseev, A. Klimentov, T. Wenaus, Z. Yang, BigPanDA monitoring system evolution in the ATLAS experiment, EPJ Web of Conf. **295**, 04010 (2024). 10.1051/epjconf/202429504010
[26] J. Elmsheuser et al., A Roadmap to Continuous Integration for ATLAS Software Development, J. Phys.: Conf. Ser. **898**, 072009 (2017).
[27] jQuery, https://jquery.com, accessed 13 Dec 2024.
[28] AngularJS, https://angularjs.org, accessed 13 Dec 2024.
[29] React - the library for web and native user interfaces, https://react.dev, accessed 13 Dec 2024.

[30] Angular - the web development framework for building modern apps, https://angular.dev, accessed 13 Dec 2024.
[31] TypeScript: JavaScript With Syntax For Types, https://www.typescriptlang.org, accessed 13 Dec 2024.
[32] Pylint - code analysis for python, https://www.pylint.org, accessed 13 Dec 2024.