PanDA Questionnaire

Questions from CMS, answers from the PanDA team
Version 1 - Jan 14 2025

1) Can you describe the architecture of your WM solution? What are its components and how do they interact?

The architecture is described in detailed documentation on the PanDA system that has been developed in recent years and is continuously maintained. See the architecture description here.

The PanDA system is comprehensively described in a 2024 paper.

No document on the capability of PanDA should be without mention of Rucio. The Rucio data management system and its Don Quixote precursors grew up together with PanDA, the two systems integrating tightly to deliver workload/data management orchestration that is essential to our data-intensive distributed computing, and even more to the sophisticated orchestrations needed to achieve the efficiencies demanded by the HL-LHC, such as the data carousel.

2) What workflows are supported in the workflow management solutions?

- Production processing
  - MC production
  - Offline data processing
  - Data carousel
- Processing optimization
  - Job cloning
    Running multiple jobs simultaneously to produce the same output. Once the first job successfully completes, other jobs are terminated.
  - Job clustering
    Consolidating multiple payloads into a single PanDA job and monitoring their execution at the payload-level granularity.
  - Multi-node jobs
    Utilizing multiple compute nodes concurrently in a single PanDA job through various frameworks such as MPI, Ray, and Horovod.
- Analysis
  - Production-style managed analysis
  - End-user analysis
  - NTuple creation
  - Workflows specified via workflow description languages (e.g. Snakemake)
- Near-realtime workflows
  - Pseudo-interactive analysis (e.g. jupyter based submissions to cloud resources)
  - Prompt processing (latencies from 10's of seconds to ˜1min and above)
- HPCs, opportunistic resources
  - Event service, streaming workflows
  - Job-packing for whole node scheduling at HPC
- ML workflows

- Hyperparameter optimization
- Distributed training
- Distributed optimization algorithms: active learning in analysis, Bayesian optimization for detector design
- Adaptable to other complex iterative workflows

3) How many developer FTEs are supporting your solution?

The table below shows the headcount and FTEs supported by each funding source.

	ATLAS	Rubin	NP/EIC	AID2E	REDWOOD	Total (Core)
Headcount	11	3	1	2	4	13 (6)
FTEs	6.95	1.4	0.3	0.55	1.8	11 (3.1)

The 'Total (Core)' column presents both the overall figures and those specifically allocated for experiment-agnostic core development. Since some developers receive support from multiple sources, the total headcount is lower than the sum of the left-hand columns.

4) How many people are involved in operating the system and documenting it?

All developers included above participate in system operation and documentation efforts, on the principle that developers should also be involved in operations (and of course documentation). There is not a dedicated documentation effort beyond that of the developers. The ATLAS distributed computing operations team (1.6 FTEs) manages the entire grid operation without any dedicated PanDA operations component. A principal tenet of PanDA has been maximizing automation to minimize operational load and this is reflected in low ATLAS operational demands despite the complexity and operational scale.

Other experiments using PanDA also allocate operations effort in combination with development effort. In Rubin, for example, the PanDA development/operations time shares have been roughly 30/70 to date, as existing PanDA functions have effectively met most of Rubin's needs with minimal customization, and the hands-on operations effort required during implementation and (presently) commissioning is high. The planned 1 FTE asymptote for Rubin effort once they reach steady state is an indication of the dev/ops required in the steady state for a substantial non-ATLAS experiment using PanDA.

5) Where does funding / person power for the solution come from?

Funding and person power come from the experiments and projects using PanDA, and from experiment-agnostic DOE-funded projects that are leveraging PanDA. These sources include US DOE HEP (ATLAS, Rubin), US NSF HEP (ATLAS), CERN ATLAS, BNL NP (EIC), US DOE NP (AID2E), and US DOE ASCR (REDWOOD). See the table in 3) above.

6) What are the evolutionary plans for this component?

Our plans include modernizing and refactoring system components, avoiding disrupting ongoing experiments due to big-bang migration and ensure compatibility with existing workflows.

Modernization plans are described in a forthcoming CHEP 2024 proceedings paper "Modernizing ATLAS PanDA for a sustainable multi-experiment future" which we summarize here.

In early 2024 ATLAS conducted an architectural review to evaluate the functionality and performance of its workflow and workload management system. 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 HL-LHC.

As part of following up on the review and addressing the recommendations, we introduced a new project management structure to streamline PanDA system development as a multi-experiment project, in the 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. The streams, discussed in more detail in the CHEP proceedings paper, are

- 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. However the continuous delivery of features and the age of the project have introduced complexities into the code base and it would be useful in many cases 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.

- 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, automation, and scalability are essential for addressing these limitations and supporting the system's growing complexity. This includes applying AI/ML within the PanDA system itself, which has had recent successes in improving efficient use of resources.

- Stream 3: Outreach
  Outreach and community engagement have become pivotal components of the PanDA WMS strategy. Since its foundation, PanDA has been an open-source software project (PanDAWMS on GitHub, see also pandawms.org) 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.

- Stream 4: Interactive and dynamic workflow-oriented platform
  The current PanDA monitoring system is a Django-based 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 capable of integrating external monitoring components, such as ATLAS Athena Nightlies CI monitoring. 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. We will approach this by moving from server-side to client-side rendering, with a redesign of the backend architecture to implement a fully separated REST API. The importance of this transition is supported by the fact that more than half of requests to the monitor retrieve data in JSON format (ie programatically).

7) Is your solution used by other organizations and does it constrain evolution?

PanDA has been adopted by experiments beyond ATLAS including most prominently the Vera C. Rubin Observatory, where it is in daily production use as they commission the LSST telescope. PanDA also plays a role in funded DOE projects that leverage its capabilities including AID2E [ref] supported by DOE NP and REDWOOD [ref] supported by DOE ASCR. A BNL instance of PanDA

has just been established for use by AID2E, EIC and other NP/HEP applications and evaluations as they emerge. The EIC experiment ePIC plans to evaluate PanDA in 2025 and make a workload management system selection. PanDA enhancements to support multi-experiment environments have included versioned releases, k8s-based deployment, and greatly improved and systematized documentation.

All of this work has also benefited PanDA's primary stakeholder, ATLAS, while bringing no constraints. The effect has been more to accelerate evolution towards an experiment agnostic system, a trajectory supported and encouraged by ATLAS and by the DOE. Technical extensions driven by non-ATLAS requirements such as realtime job log monitoring and fast (in seconds) workload initiation, both motivated by Rubin applications, have also fed back general benefit.

PanDA's adoption by Rubin, with the attendant importance of making PanDA a manifestly multi-experiment system, motivated turning longstanding intentions towards building an inclusive PanDA community into actual action. A community mailing list and meeting series, US timezone friendly core meetings, an explicitly welcoming and open development and ticketing policy are all aspects of this that have been established. We plan to keep moving in this direction; further adoptions of PanDA will accelerate it.

When Rubin adopted PanDA and included PanDA dev/ops support in their budget, they chose to have the effort added to the BNL core PanDA team, with the proviso that Rubin support be spread across multiple core team members, avoiding a 'Rubin person' point of failure and better coupling Rubin needs and interests into the PanDA core. This has worked well, but it means we have yet to address the integration of core developers from outside of ATLAS and the existing PanDA institutes. We expect and hope that this will come up eventually, and we look forward to adapting to it with an inclusive approach to integration of new members from new communities in the core team.

An important architectural precept we follow in the multi-experiment PanDA era is that there is one PanDA. The system's modularity supports experiment-specific component implementations where required, but as much as possible the experiment-driven changes become part of the core code base, typically bringing benefit to all as mentioned. There is not an ATLAS PanDA and a Rubin PanDA and we aim to keep it that way. This demands the inclusive approach to core team membership and contributions mentioned above.

8) What is going well and what functionality is missing?

The PanDA system excels in scalability and flexibility, adapting well to new computing resources, services, and use cases. Principal areas targeted for evolutionary improvement are modernization and refactoring of the code base and infrastructure for long-term sustainability, improving the end-user experience, enhancing documentation, and improving PanDA's accessibility and utility for other experiments through both technical and community-building work.

9) What are operators/users most unhappy about?

Users often express dissatisfaction with the predominantly command-line-based and overly simplistic user interface, as well as challenges in understanding system operations. While command-line tools are efficient for advanced users, contemporary preferences lean towards more sophisticated graphical user interfaces that offer abundant visualized information, facilitating intuitive insights into system functionality. Enhancing usability and user experience through the development of a more refined interface and comprehensive system dynamics visualization is a key focus for system evolution.

Eliciting feedback, even from unhappy users, is always a challenge. Our latest attempt is a set of smiley-to-frowny buttons on PanDA monitor pages to guide the improvement priorities of the developers.

Overall system complexity is in general a hurdle for users. While the complexity itself is unavoidable given what PanDA does, an objective for improving usability and the user experience is to better insulate users from complexity that they are not interested in (while still supporting comprehensive transparency and drill-down for experts who are interested). For example in a proposed extension of the AID2E project to be submitted this month, our colleagues in the project plan to apply LLM technologies to create a layer over the PanDA monitor and other information sources in the system to create an easier to use interface.

Thoughout PanDA's existence we've frequently heard 'PanDA is slow'. Almost always, the slowness is actually the grid; PanDA's principal activity remains the management of processing on the grid. Close to 20 years of development have gone into avoiding it being 'even slower'! An

exciting PanDA development in recent years has been applying PanDA to low-latency near-realtime tasks, exploiting that PanDA inherently can be fast. In applications like ATLAS pseudo-interactive analysis, Rubin prompt processing of transients, and coming use cases in EIC streaming readout, PanDA as a rapid processing tool is being developed and applied.

10) What progress have you made in testing the scalability of your solution to Run4 requirements?

The early design decisions to build PanDA on the foundation of highly scalable web service technologies have served us well, with no scaling limitations encountered thus far and none foreseen to achieve HL-LHC scale. That said, specific evaluations of HL-LHC scalability have not been performed to date (in part because it's not seen as a substantial risk).

The system is mostly horizontally scalable (i.e. you can add more JEDIs, harvesters, PanDA servers as scale and load increase). The one central component without trivial horizontal scalability is the database. We have reduced our database usage in the last years by optimizing the heaviest queries and we continue to monitor the usage closely. The expertise to do so lies mostly within the PanDA team (until recent years it resided mainly with a CERN DBA). With the last major Oracle upgrade, ATLAS also got new hardware and we are at a very low usage (10% CPU usage), giving a reasonable expectation that there is room to grow. The code modernization and refactoring underway is also bringing ancillary benefit in system speedup 'for free' (tens of percent).

ATLAS constantly exploits computing resources exceeding the pledges through projects linking to HPC and cloud resources, as well as Tier facilities with over-pledge resources. HPCs and clouds often require rapid spin-up/shut-down scenarios, which are challenging for a central orchestrator; they resemble brief periods of extremely high load. Yet we experience extremely rare service disruptions in PanDA due to overloaded servers/DB. These rare cases we solve by optimizing the specific bottleneck, we never had to throw more hardware at it.

We have also learned from the Rubin experience. Although nominally at a much smaller computing scale than ATLAS, their workflows are complex (very large DAGs) with processing units (jobs) that are extremely short, often about a minute. Consequently job count and pressure on the workload management system is much higher than the processing scale would suggest. This applied particular pressure on PanDA's complex workflow management component, iDDS, and drove scaling and hardening efforts well beyond the demands of ATLAS applications, giving improved overall system robustness and scalability.