Requirements for an ePIC Distributed Workflow Management System

Markus Diefenthaler, Torre Wenaus
and the ePIC Streaming Computing Model Working Group

Version 1.0 – October 3 2025

Contents

Introduction

The ePIC workflow and workload management system (WFMS) software will manage the computing use cases defined in the ePIC computing model [1] across heterogeneous geographically distributed computing resources. Within the ePIC Streaming Computing Model Working Group, the WFMS for the streaming data of the ePIC experiment is also referred to as streaming orchestration. A WFMS for ePIC is required from the outset, as it is essential for any type of system integration in the distributed computing model of ePIC.

This document, developed by the ePIC Streaming Computing Model Working Group, describes the requirements for the ePIC Distributed Workflow Management System. It is structured by areas including architecture, processing use cases, streaming, distributed processing, workflow management, resource utilization, interfaces, and monitoring. It will guide the further development of ePIC's streaming computing model [1] (from which it is derived), the prototyping of WFMS systems, and the selection of WFMS tools. AI may be incorporated into the WFMS itself and will be involved in workflows, as described in several examples throughout the text.

1 Core architecture and design priorities

1.1 Robust scalability and availability supported by careful technology choices.

1.2 Low system overheads enabling fast (near) real time operation.

1.3 Insulating the system and its users/operators from the complexities of a heterogeneous distributed computing fabric.

1.4 Tight integration with distributed data management (DDM) for seamless streaming and data driven workflow orchestration.

1.5 Fast, flexible, centralized brokerage informed by system state and history.

1.6 Minimal operational overhead through maximal automation.

1.7 Comprehensive monitoring and diagnostics.

1.8 System updates with minimal or no downtime.

1.9 Avoid requiring custom tailoring of resources (use batch queues, K8s clusters, etc.).

1.10 Support workflows as complex as the experiment requires.

1.11 Support horizontal scale-up to handle evolving data rates.

1.12 Support permanent recording of bookkeeping, metadata, provenance.

2 Processing use cases

Support all of ePIC's processing use cases:

2.1 Streaming processing for near real time workflows (monitoring and validation, with the requisite reconstruction and calibration for subsets of the data) at Echelon 1, with capability to extend to Echelon 2.

2.2 First pass reconstruction at E1–E2.

2.3 Calibration and alignment workflows at E1–E2, both prompt and offline.

2.4 Simulation production at E1–E2.

2.5 Reprocessing at E1–E2.

2.6 Analysis workflows at E1–E3.

2.7 The workflow description should be structured around these use cases and allow user-centered descriptions from the perspective of detector, DAQ, or analysis experts (not only computing experts).

3 Streaming processing

3.1 Support continuous stream-based Echelon 1 processing for prompt validation and reconstruction/analysis, with capability to process data arriving from DAQ within $O(10\text{s})$.

3.2 Support ePIC's "always on" data streaming, including accelerator and detector information when ePIC is not in running state; automatically suspend/resume processing depending on state and stream activity.

3.3 Integrate with a detector/data state machine describing current ePIC detector/accelerator state and data delivered from DAQ.

3.4 Support prompt validation of the integrity of complete copies of data arriving and archived at both E1 centers, maintaining a continuously communicated "running checksum" between them.

3.5 Support continuous sampling of the data stream to validate via checksum and other metadata against independent sources (DAQ metadata, conditions data, etc.) as an early alarm against data corruption/stream faults.

3.6 Process timeframes (TFs) and super timeframes (STFs) as the atomic units of raw data processing workflows. Data transfers and batch processing are organized by TF ranges via the STF mechanism (an STF file contains a contiguous series of TFs). Streaming agents may run as batch jobs with automated replacement/replenishment so they behave as persistent agents.

3.7 Support automated merge/gather processing aggregating results of STF processing.

3.8 Support near real-time monitoring of streaming data quality with latencies from seconds (fine-grained TF-based stream subsets) to minutes (STF-based monitoring).

3.9 Provide fast monitoring: access, process, and react to data at finer granularity than $\sim 1\,\text{s}$ STFs. TF-level results should support rapid anomaly detection. Skims of STF files (by TF range, TF metadata selections, sampling frequency, etc.) can be tuned for latency/content/statistics and delivered also to E2 with relaxed latency.

3.10 Support streaming processing for both STF and TF streams within batch-based resources (agents as batch jobs with automated replacement).

3.11 Enable rapid feedback loops for prompt calibration.

3.12 Support express operations for time-critical, high-priority workflows (e.g., detector status monitoring and alarm feedback).

4 Data management integration and processing orchestration

4.1 Tight integration between WFMS and DDM to jointly orchestrate complex data-driven workflows (e.g., streaming processing).

4.2 Enable workflows to be driven in real time by DDM-originating events such as data availability.

4.3 Use real-time communication so both systems have current and complete knowledge of data at any given time.

4.4 Support input data locality modes: remote access (stream over network) and local replication (WFMS requires/drives data collocation).

4.5 Support efficient output data management: move outputs quickly to local storage and asynchronously drive movement to final locations.

4.6 Support data carousel workflows for efficient tape utilization.

4.7 As DAQ data is ingested, data/workflow management must independently and simultaneously trigger archiving and processing, with no possibility of processing blocking archiving.

4.8 Record data provenance across the processing chain.

5 Distributed processing

5.1 Support the "butterfly model": two symmetric Echelon 1 facilities at BNL and JLab operating as an integrated distributed E1 system with failover. If for example the instance at BNL goes down, the one at Jefferson Lab should take over.

5.2 Support geographically distributed processing across host labs and collaborating institutions.

5.3 Enable efficient inter-site data transfer with minimal latency.

5.4 Provide dynamic load balancing across resources.

5.5 Support conventional batch processing across globally distributed E1 and E2 resources for simulation and reprocessing.

5.6 Support concurrent processing on $O(100\text{k})$ cores across $O(10\text{–}20)$ facilities.

5.7 Support GPU-as-a-Service for dispatching GPU workloads, concurrently with local CPU processing and with integration into sub-event level parallelism in the software framework. Example use cases include ML inference, Cherenkov optical photon simulations, and GPU-accelerated algorithms used in event reconstruction.

6 Streaming metadata and APIs

6.1 Data access APIs should support reading stream data at TF and beam crossing (BX) levels. Reading at BX level returns the TF containing that BX. Each TF has a clearly specified BX range; this metadata must be cataloged for efficient BX-level access.

6.2 Metadata for physics data products (e.g., tracks, particles), their event aggregations, and channel/stream event aggregations must provide traceability to TF and BX.

6.3 BX and TF have monotonically increasing IDs uniquely identifying them for the lifetime of the EIC; cataloging/access must support retrieval by these IDs.

6.4 A metadata database (or similar) holds records of named run periods (configuration, TF range, etc.). Run periods can overlap; a TF need not be uniquely associated with a single run period.

6.5 Support prompt validation of integrity of complete copies of data archived at both E1 centers with a continuously maintained cross-facility running checksum.

6.6 Support continuous sampling/validation against DAQ and conditions metadata as an early alarm (complements facility-integrity checks above).

6.7 E1 interfaces/APIs for streaming workflow and data handling should be implementation-agnostic to enable distinct/evolving implementations (files, tape, object stores, hierarchical caching, direct block writes, etc.).

6.8 Data integrity: written TFs of the main DAQ data stream cannot be overwritten without administrative intervention; data management/storage must enforce protections.

7 Workflow management

7.1 Support community-standard workflow description tools (e.g., DAG, Snakemake, CWL) to describe complex chains and translate them into executable workflows.

7.2 Support complex dependencies/conditionals (input readiness, upstream completion/validation, calibration readiness).

7.3 Provide user-level tools (parameterized templates, domain-centric descriptions) enabling detector/DAQ/analysis experts to define workflows without deep computing details.

7.4 Provide flexible, scalable support for AI/ML workflows (distributed training, hyperparameter optimization, iterative/optimization tasks).

7.5 Support composite workflows spanning heterogeneous environments (e.g., grid-based simulation plus GPU-accelerated ML training).

8 Production campaign management

8.1 Support aggregation of related processing into high-level tasks.

8.2 Support both pre-defined and dynamic prioritization among tasks.

8.3 Support automated retry at task or component granularity.

8.4 Provide flexible, granular prioritization to align resource utilization with physics-driven campaign priorities.

8.5 Support expediting for high-priority tasks (preferential submission to high-throughput facilities, job cloning with first-finish selection).

8.6 Provide meaningful logs analyzable individually and in aggregate, including current workflow state and progress.

8.7 Provide sufficient data accountancy while data is in transit.

9 Resource utilization

9.1 Interface with diverse computing resources across all Echelons.

9.2 Support heterogeneous platforms, e.g. x86, ARM, or GPU.

9.3 Optimize processing location based on data locality and current availability.

9.4 Enable efficient use of opportunistic resources (OSG, HPCs, clouds) by treating them as brokerage destinations; match workflows to suitable resource types.

9.5 Support resource allocation and fair-share mechanisms.

9.6 Support criticality/suitability levels on workflows and use them in brokerage decisions given resource reliability and characteristics.

9.7 Provide transparency into resource utilization across facilities.

10 Payloads and integration

10.1 Support standardized, Collaboration-provided payloads/templates for standard production workflows.

10.2 Flexibly accommodate non-standard, user-defined payloads for analysis, with traceability/monitoring.

10.3 Support containerization as standard payload encapsulation (Singularity, Docker); flexibly support Collaboration registries.

10.4 Seamlessly support integrations used by payloads: JANA2, conditions and other databases, and the ePIC software stack.

11 User interfaces

11.1 High usability for users/operators is fundamental; follow user-centered design.

11.2 Provide CLI, web, and programmatic REST interfaces for authenticated/authorized users, supporting both passive and active use.

11.3 Individual users should have personal dashboards with intelligent diagnostics of usage and failures.

11.4 System operators should have comprehensive views for sites, workflows, tasks, campaigns, plus aggregate summaries.

11.5 Facility operators need overviews and drill-downs on site operation, performance, and failures.

11.6 The REST API should follow best practices and be robust and backward-compatible.

11.7 Users/operators should be able to use the REST API to create tailored monitoring/notification systems.

11.8 Track and monitor REST usage for appropriate use and to identify integrations that should graduate to official monitoring.

11.9 Track resource usage by workflow, user, facility, etc.

11.10 Enable integration with Jupyter and other analysis environments.

11.11 Provide clear error messages, interpretation, and troubleshooting guidance.

11.12 Provide active, configurable notifications for errors and state/workflow changes.

12 Monitoring and analytics

12.1 Comprehensive monitoring for users and operators that supports high-level summaries and detailed diagnostic drill-down is fundamental.

12.2 The full operational history should be available to monitoring and analytics.

12.3 Primary delivery via web interface.

12.4 Real-time state-of-system displays should be fast.

12.5 Historical monitoring (days to years) may be slower but should remain comfortable for interactive use.

12.6 Provide bulk export and ingest to community analytics tools (e.g., OpenSearch, Grafana).

12.7 Analytics displays should integrate and interoperate with WFMS-native monitoring.

12.8 Monitoring should include current state and time-evolution views, particularly for streaming workflows and the detector/data state machine; visual design should also serve as an impactful outreach display of distributed streaming processing.

12.9 Provide intuitive visualizations of workflow progress.

13 Resilience, fault tolerance, testing

13.1 Automatically recover from failures from the STF level up to tasks requiring resubmission; support intelligent, configurable retry logic.

13.2 Provide diagnostic tools and monitoring for efficient problem identification by operators and users.

13.3 Support catch-up modes to recover from processing/data-flow interruptions that create backlogs.

13.4 Gracefully handle network restrictions/degradations between facilities.

13.5 Provide automated tools for continuous testing/validation of centralized components, processing sites, and representative workloads.

13.6 Provide performance metrics and benchmarking tools to evaluate resource functionality/performance.

14 Security and access control

14.1 Support modern federated-identity, token-based authentication standards, and legacy standards as required (e.g., X.509).

14.2 Support role-based authorization for system functionality.

14.3 Support end-user authorization enabling appropriate actions via user interfaces (e.g., dynamic job control).

14.4 Support secure access and usability for all ePIC collaborators globally.

15 System development, versioning, releases

15.1 Employ modern software engineering practices in development and documentation.

15.2 Sources are open and managed in GitHub.

15.3 Maintain comprehensive automated testing.

15.4 Support continuous integration and deployment.

15.5 Maintain distinct development/test instances for centralized and distributed components separate from production.

15.6 Production deployments should be changelog-documented, version-stamped, and validated via CI/tests.

15.7 Major software versions should correspond to major releases with a defined release process.

15.8 Production deployment must allow very fast turnaround for urgent fixes/updates, beyond conventional releases (i.e., not constrained to major releases).

15.9 Fast-turnaround deployments need their own test/validation policies, processes, and tools, akin to DAQ-style changes in flight with best-possible validation; track actions, responsible persons, and system state, and be able to revert to a known working state.

16 Documentation and community

16.1 Provide comprehensive user, operator, and developer documentation, tailored to the experiment.

16.2 Support user training programs and materials.

16.3 Maintain up-to-date operational procedures.

16.4 Document system architecture and design decisions.

16.5 Foster an open development model engaging the community, encouraging contributions, and leveraging external developments.

16.6 Establish clear and transparent decision-making processes.

Glossary

Distributed data management, DDM

Infrastructure supporting organization/access to ePIC detector data across E0, E1, E2; includes bookkeeping, metadata (system/user), registration with notifications, replication (manual/automated), and monitoring/analytics. ePIC's chosen DDM is Rucio.

Echelon 0, E0
DAQ, encompassing IP6 and the DAQ enclave in the BNL computing center.

Echelon 1, E1
The two principal computing facilities at BNL and JLab.

Echelon 2, E2
Domestic/international facilities providing processing/storage to ePIC; integrated with WFMS and DDM.

Echelon 3, E3
Institutional facilities serving ePIC physicists (local clusters to desktops), served by E1/E2 services for analysis and development.

Rucio
ePIC-chosen community standard DDM system.

Super timeframe, STF
Atomic data unit of ePIC streaming processing at E1; contiguous aggregation of TFs, about a half second, ~2 GB.

Timeframe, TF
~0.5 ms slice containing all detector data for that period; basic unit of ePIC data taking.

WFMS
Workflow management system.

References

[1] Battaglieri, M., Deconinck, W., Diefenthaler, M., Huang, J., Joosten, S., Kalinkin, D., Landgraf, J., Lawrence, D., and Wenaus, T. (2024). The ePIC Streaming Computing Model. Zenodo. https://doi.org/10.5281/zenodo.14675920