

ePIC workflow management system requirements

V1 draft - Apr 1 2025

1) Core architecture and design priorities

1. Robust scalability and availability supported by careful technology choices.
2. Low system overheads enabling fast (near) real time operation.
3. Insulating the system and its users/operators as much as possible from the complexities of a heterogeneous distributed computing fabric.
4. Tight integration with distributed data management (DDM) for seamless streaming and data driven workflow orchestration.
5. Fast, flexible, powerful, centralized brokerage and other system decision-making, informed by comprehensive self-gathered information on WFMS and DDM system state, operation and history.
6. Minimal operational overhead through maximal automation.
7. Comprehensive monitoring and diagnostics for system understanding and rapid problem identification/resolution.
8. System updates with minimal or no downtime.
9. Avoid requiring custom tailoring of resources. Use what the system provides -- batch queues, K8s clusters etc. -- for all use cases.
10. Support workflows as complex as the experiment's use cases require, not constrained by system capability.
11. Support horizontal scale-up of the system to flexibly and economically handle evolving data rates and processing scales.
12. Support complete and permanent recording of comprehensive system and payload bookkeeping, metadata and provenance information, supporting reproducibility, fine-grained retries/recovery, intelligent analytics and system automation

2) Processing use cases

1. Support all of ePIC's processing use cases:
 - a. Streaming processing for near real time workflows (reconstruction, monitoring, validation, calibration) at Echelon 1s, with capability to extend to Echelon 2s
 - b. First pass reconstruction at E1-E2
 - c. Calibration and alignment workflows at E1-E2, both prompt and offline
 - d. Simulation production at E1-E2
 - e. Reprocessing at E1-E2
 - f. Analysis workflows at E1-E3
2. The workflow description should be structured around these use cases and should allow it to be described from the perspective of the detector, DAQ, or analysis expert, rather than from that of a computing expert.

3) Streaming processing

1. Support continuous stream-based Echelon 1 processing for prompt validation and reconstruction/analysis, with the capability to process data arriving from DAQ within $O(10\text{sec})$
2. Support ePIC's 'always on' data streaming, actively streaming e.g. accelerator and detector information when ePIC is not in running state. Automatically suspend/resume streaming processing depending on state and stream activity.
3. Integrate with a detector/data state machine describing the current state of the ePIC detector/accelerator complex and the data currently being delivered from DAQ
4. Support prompt validation of the integrity of complete copies of the data arriving at the two E1 centers
5. Support prompt validation of the integrity of the data stream, e.g. by comparing metadata appearing in the stream with independently provided metadata from DAQ, conditions data etc.
6. Process timeframes and super timeframes (STFs) as the atomic data and processing unit of ePIC raw data processing workflows.
7. Support automated merge/gather processing that aggregates the results of STF processing.
8. Support near real-time monitoring of streaming data quality with latencies from beam collision to E1 monitoring ranging from seconds (via a fine grained TF-based stream with a subset of the data) for fast monitoring and data quality feedback to the collider, to minutes for STF-based monitoring.
9. Support streaming processing within the context of batch-based processing resources. Streaming agents can run as batch jobs, with automated replacement/replenishment.
10. Enable rapid feedback loops for prompt calibration.
11. Support express operations for time-critical high-priority workflows, e.g., the monitoring of the detector status and feedback to the detector via alarms etc.
12. It must be possible to access, process and react to events at a finer granularity than the $\sim 1\text{sec}$ STF. We need results down to event by event level to achieve low latency on finding anomalies in event processing.

4) Data management integration and processing orchestration

1. Support tight integration between the WFMS and DDM systems, enabling them to jointly orchestrate complex data-driven workflows such as streaming processing
2. Enable processing workflows to be driven in real time by DDM-originating events like data availability
3. Use real time communication with the DDM system to ensure each system has a current and complete knowledge of data in the system at any given time.
4. Support different input data locality modes such as remote access (input data source is remote and data is streamed over the net) and local replication (WFMS requires and/or drives the collocation of input data and processing)
5. Support output data management that uses resources efficiently, e.g. move outputs quickly to local storage and asynchronously drive data movement to a final location.
6. Support data carousel workflows for efficient tape storage utilization.
7. As data is ingested from DAQ, data/workflow management must simultaneously and independently trigger the archiving of the data, and the processing of the data, with no possibility of the latter blocking the former.
8. Record data provenance information throughout the processing chain.

5) Distributed processing

1. Support the "butterfly model" in which two Echelon 1 computing facilities at BNL and JLab, symmetric in their capabilities, are operated as parts of a coherent distributed Echelon 1 system.
2. Support geographically distributed processing across host labs (BNL and JLab) and global collaborating institutions.
3. Enable efficient data transfer between processing locations with minimal latency.
4. Provide dynamic load balancing across available resources.
5. Support conventional batch processing across globally distributed Echelon 1 and 2 resources for simulation and reprocessing.
6. Support concurrent processing on $O(100k)$ cores distributed across $O(10-20)$ distributed facilities.

6) Workflow management

1. Support community standard workflow description tools (e.g. DAG, Snakemake, CWL) to describe complex processing chains, automatically translating them into executable workflows on the WFMS system
2. Support complex workflow dependencies and conditionals such as input data readiness, upstream processing completion and validation, calibration readiness. *It should be examined whether workflow description tools are able to describe all the dependency-driven calibration workflows of ePIC. (We will learn about this from the workflow testbed.)*
3. Provide user level tools for convenient workflow definition such as parameterized workflow templates, supporting workflow description from the perspective of the detector, DAQ, or analysis expert, rather than from that of a computing expert.
4. Provide flexible and scalable support for AI/ML and related workflows such as distributed training, hyperparameter optimization and other optimization and iterative processing tasks
5. Support composite workflows encompassing different workloads/operating environments (e.g. involving simulation production on grid resources and GPU-accelerated ML training/optimization on specialized resources)

7) Production campaign management

1. Support aggregation of associated processing into high level processing tasks
2. Support both pre-defined and dynamic prioritization among tasks
3. Support automated retry for tasks, either the complete task or the incomplete components
4. Provide flexible, granular prioritization mechanisms to match resource utilization with the needs and priorities of physics-driven production campaigns.
5. Support expediting techniques for high priority tasks like preferential submission to high-throughput facilities and job cloning (concurrently submitting duplicate jobs and using the first to finish)

8) Resource utilization

1. Interface with diverse computing resources across all Echelons
2. Support heterogeneous computing platforms: x86, ARM, GPUs, TPUs, specialized capabilities like high-memory
3. Optimize processing location based on data locality and current processing availability

4. Enable efficient utilization of opportunistic resources (OSG, HPCs, clouds) by supporting them all as prospective destinations in brokerage decisions. (Some workflows are well suited to any opportunistic resource, e.g. simulation, but not all.)
 - a. *ToDo: add criticality characteristics to our list of processing workflows/use cases and implications for what resource types can be used*
5. Support resource allocation and fair share mechanisms.
6. Provide transparency into resource utilization across computing facilities.

9) Payloads and integration

1. Support standardized, Collaboration-provided payloads and templates handling the standard production workflows.
2. Flexibly accommodate non-standard, user-defined payloads for individual and group level analysis, with traceability mechanisms to monitor and manage usage.
3. Support containerization as the standard approach to encapsulating payloads. Support community standard services (Singularity, Docker). Flexibly support container registries and specifically all those used by the Collaboration.
4. Seamlessly support integrations used by the payloads: the Jana2 framework, conditions database, and the ePIC software stack in general.

10) User interfaces

1. A high level of usability by users and operators is a fundamental requirement. Follow the user-centered design principle of the Collaboration.
2. Provide command-line, web-based and programmatic REST interfaces to the system, supporting both passive and active functionality by authenticated and authorized users.
3. Individual users should have their own dashboard page with well designed views of their usage of the system, including intelligent diagnostics of usage and failures.
4. System operators should have comprehensive views of the system for sites, workflows, tasks, campaigns, together with aggregating summaries.
5. Facility operators should have overviews and detailed drill-downs on the operation, performance and failures of processing sites.
6. The REST interface should be well designed following best practices, with a robust and backward-compatible API.
7. Users and operators should be able to use the REST interface to create their own tailored informational, monitoring and notification systems.
8. Usage of the REST interface should be tracked and monitored for appropriate use and to spot uses that should be integrated with official monitoring.
9. Resource usage should be tracked by workflow, user, facility, etc.
10. Enable integration with Jupyter and other analysis environments.
11. Provide clear error messages, error interpretation and troubleshooting guidance.
12. Active, configurable notification mechanisms should be provided for error conditions, state and workflow status changes etc.

11) Monitoring and analytics

1. Comprehensive system monitoring for users and operators that both summarizes at a high overview level and supports detailed drill-down for diagnostics is a fundamental requirement.
2. The entire operational history of the system should be available to monitoring and analytics.
3. Monitoring and analytics should be provided primarily via the web interface.
4. Monitoring should show the state of the system in real time. It should be fast.
5. Monitoring of older information, from several days to weeks/months/years prior, may be slower, but should be comfortable for an interactive user.
6. System information should be available for bulk export and ingest to community standard analytics tools such as OpenSearch and Grafana.
7. Analytics displays should integrate and interoperate with the WFMS-native monitoring.
8. Monitoring should include current state-of-the-system and displays and time-evolution displays, particularly for streaming workflows and representation of the detector/data state machine. This monitoring should have visual design and utility supporting it both as a monitoring/diagnostic tool and as an impactful 'outreach' display of ePIC distributed streaming processing in action.
9. Provide intuitive visualizations of workflow progress.

12) Resilience, fault tolerance, testing

1. Automatically recover from failures at all processing scales from the atomic data unit (STF) on up to processing tasks that require resubmission. Support intelligent, configurable retry logic.
2. Provide diagnostic tools and monitoring to allow operators and users to drill down and identify problems efficiently.
3. Support 'catch-up' modes to recover from interruptions to processing or data flows that result in backlogs.
4. Gracefully handle network restrictions and degradations between facilities.
5. Provide automated tools to continuously test and validate the operation of the system, including centralized components, processing sites and representative workloads.
6. Provide performance metrics and benchmarking tools to evaluate resource functionality/performance.

13) Security and access control

1. Support modern federated-identity, token-based authentication standards, and legacy standards as required (e.g. X509)
2. Support role-based access/authorization control for system functionality.
3. Support end user authorization that enables users to act on the system in appropriate ways through its user interfaces, e.g. dynamically controlling their jobs.
4. Support system access, authorization and usability for all ePIC collaborators globally.

14) System development, versioning, releases

1. Employ modern software engineering practices in development and documentation.
2. Sources are entirely open and managed in GitHub.
3. Maintain comprehensive automated testing.

4. Support continuous integration and deployment.
5. Both the centralized and distributed components of the system should have development/test instances distinct from the production instances.
6. Software deployed to production should be changelog-documented, version-stamped, and validated through CI and test procedures.
7. Major software versions should be reflected in major software releases that follow a well-defined release management process.
8. Deployment to production has to be possible with very fast turnaround to address bug fixes and urgent updates, in addition to conventional release deployments. ie. Production deployments are not constrained to be major software releases.
9. Fast-turnaround deployments need their own test/validation policies, processes and tools, distinct from practices for releases.

15) Documentation and community

1. Provide comprehensive user, operator, and developer documentation, for the system in general and tailored to the experiment.
2. Support user training programs and materials.
3. Maintain up-to-date operational procedures.
4. Document system architecture and design decisions.
5. Foster an open development model that engages the community, encourages contributions, and draws on external developments.
6. Establish clear and transparent decision-making processes.

Glossary

- Distributed data management, DDM - the infrastructure supporting the organization of and access to ePIC detector data across E0, E1, E2 facilities. Includes bookkeeping, metadata (system and user defined), registration (with automated notification services), replication (manual and automated), and monitoring/analytics. ePIC's chosen DDM system is Rucio.
- Echelon 0, E0 - Echelon 0 (DAQ, encompassing IP6 and the DAQ enclave in the BNL computing center)
- Echelon 1, E1 - The two Echelon 1 principal computing facilities at the host labs BNL and JLab
- Echelon 2, E2 - Domestic and international computing facilities committed to providing processing and storage resources to ePIC. Integrated with the ePIC WFMS and DDM services.
- Echelon 3, E3 - Institutional computing facilities serving ePIC physicists, from local clusters to desktops. Served by E1 and E2 based services in support of ePIC physics doing analysis and development 'at home'.
- Rucio - ePIC chosen community-standard distributed data management system.
- Super timeframe, STF - The atomic data unit of ePIC streaming processing at E1. A contiguous aggregation of time frames, about a half second in duration, corresponding to ~2GB of data.

- Timeframe, TF - a ~half millisecond time slice containing all detector data for that period. The basic unit of ePIC datataking.
- WFMS - workflow management system