ISSN 1547-4771, Physics of Particles and Nuclei Letters, 2026, Vol. 23, No. 5, pp. 674–677. © Pleiades Publishing, Ltd., 2026.

COMPUTER TECHNOLOGIES IN PHYSICS

Unified Middleware for Data, Workflow, and Workload Management in SPD Online Filter

N. Greben[a, *], D. Oleynik[a], L. Romanychev[a, b], and P. Korshunova[a]

[a] *Joint Institute for Nuclear Research, Dubna, Moscow oblast, 141980 Russia*

[b] *Department of Computer Modeling and Multiprocessor Systems, St. Petersburg State University, St. Petersburg, 198504 Russia*

**e-mail: ngreben@jinr.ru*

Received December 5, 2025; revised March 20, 2026; accepted April 20, 2026

Abstract—The SPD online filter is a crucial intermediate system that bridges the continuous data stream from the DAQ and the offline computing system. SPD Online Filter is designed for the initial high-throughput, multi-step primary processing of data from the SPD detector. It's composed of several components: a dedicated compute cluster; applied software, and the middleware which is an intermediate software layer that connects both hardware resources and applied software. The middleware integrates three core components: data and storage management (DSM), workflow management system (WfMS), and workload management system (WMS). We describe the design principles, component interactions, that enable flexible workflow definition, and concurrent execution of multiple workflows.

DOI: 10.1134/S1547477126700718

INTRODUCTION

The SPD online filter is the high-throughput computing system responsible for the primary multi-step data processing for the Spin Physics Detector (SPD) experiment [1]. The system's design is based on a data-driven paradigm, where independently processable datasets trigger complex computational workflows. The data-driven aspect is revealed in the sense that the workflow management system (WfMS) dynamically generates chains (DAGs in the general case) of tasks based on the metadata of incoming datasets from the data management system (DSM). Each of these workflows is then executed at scale, where each step of execution is then decomposed into a set of smaller independent workload units called jobs [2]. This creates an opportunity for massive parallelism but requires sophisticated middleware to manage the interplay between data, workflow logic, and compute resources. The conceptual foundation for this middleware, based on a unified approach to data, workflow, and workload management, has been outlined in [3–5]. We have described how the data management system manages the full data lifecycle, how the workload management system translates tasks acquired from the workflow management system into a set of jobs, dispatches them to compute nodes, and manages their execution. The primary contribution of this work is the demonstration of key components that allowed both workflow and workload management systems to automatically instantiate and execute several concurrent workflow instances from a single template. We show that the current setup provides a solid proof of concept, however, it still requires more comprehensive support for complex workflows and testing on a larger scale.

WORKFLOW EXECUTION SCHEME

Workflow management system (WfMS) serves as a high level interpreter, executing the DAGs (directedacyclic graphs) that represent a multi-step data processing in large-scale. To generate a workflow, a sequence of tasks, WfMS matches the first primary dataset that comes from the data management system (DMS) with a CWL (common workflow Language) template [6], which is similar in format to YAML. CWL template contains a description of each step, including data processing program, dependencies, datasets (input and output), their metadata, etc. It establishes dependency between one step of execution and the next in sequence, where where the system allows authorized users to specify what output datasets should serve as input for the following processing step via an impersonal/generalized approach. Users can create, validate, and register new templates, specifying input-output dataset mappings, handlers and priorities. After the whole workflow, triggered from this template, has been generated, the first task is ready to be sent for execution to the workload management system.

All operational states of tasks and datasets are stored in a database instance. WfMS periodically queries the WMS about the current status of tasks, and after a task has been executed, the corresponding dataset changes state to "closed". WfMS collects a pool of tasks for which the input datasets are filled with new

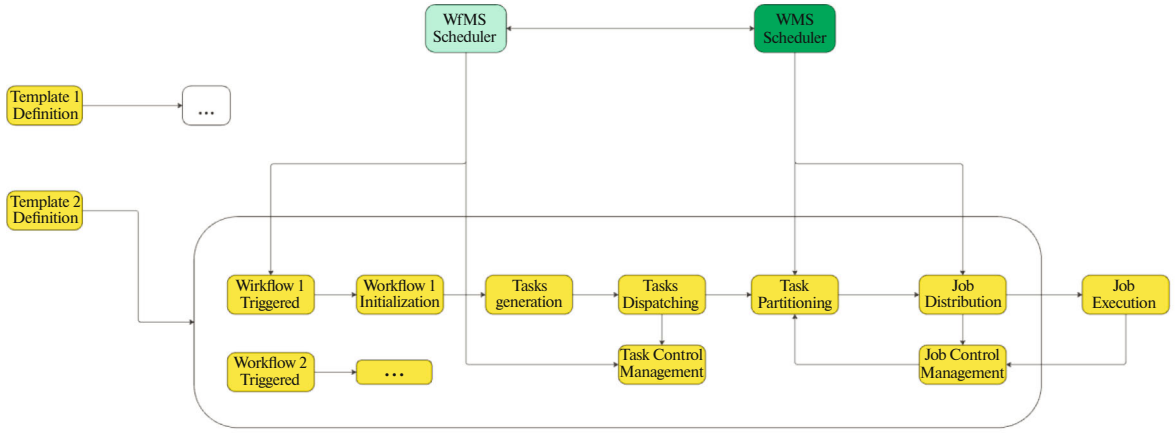


Fig. 1. General workflow execution scheme in SPD online filter.

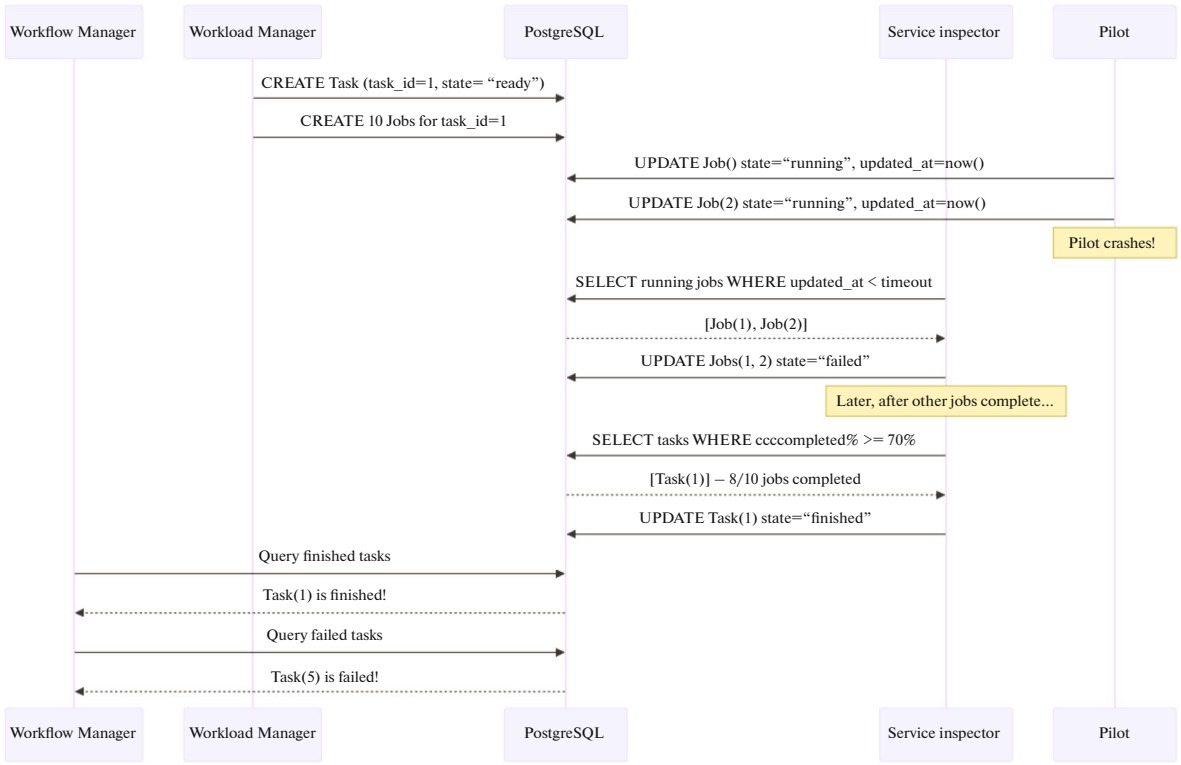


Fig. 2. Sequence diagram of the workload management system monitoring daemon.

data, and creates Pythonic coroutines to asynchronously fill tasks with necessary metadata, after that it dispatches them to WMS. It should be mentioned that the WfMS scheduler (Fig. 1) only performs state-transitions between one successfully executed task and the next one for the moment. Workload management system partitions each task into individual processing jobs, dispatches these jobs to Pilot agents on compute nodes, and monitors their execution. It ensures efficient resource utilization by generating the appropriate number of jobs, tracking status, handling retries or failures, and aggregating output files into new datasets.

WORKLOAD MANAGEMENT INSPECTION SERVICE

The Service Inspector is a continuous monitoring daemon that maintains system reliability and state consistency in a workload management system by periodically querying the operational states (“running”, “failed”, “finished”, etc.) of the tasks and jobs in the PostgreSQL database and performing automated recovery actions (Fig. 2). It’s the part of the WMS that is responsible for performing state transitions of tasks and jobs, and the one that indirectly alerts WfMS about the current status of tasks (Fig. 1).

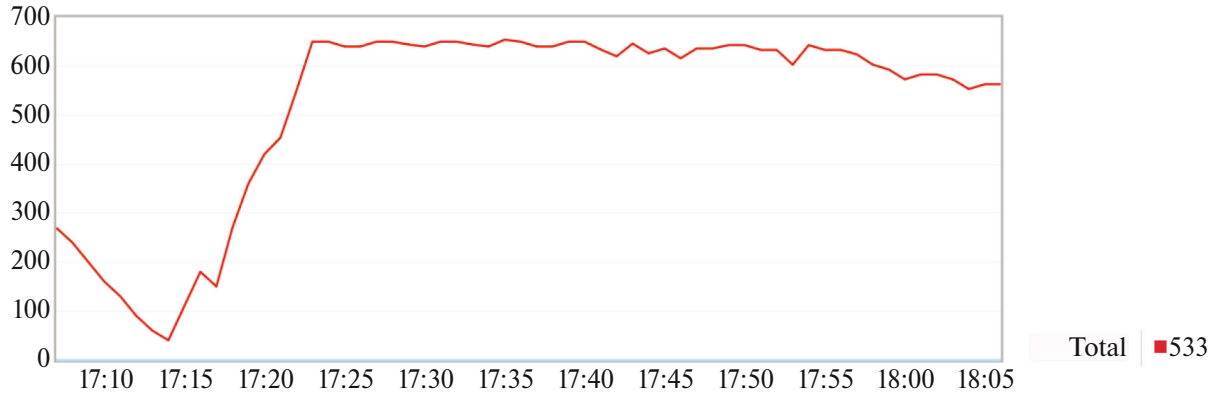


Fig. 3. Pilot queue saturation—70 workflows (1 dataset/min, 10 × 10 MB files).

Operating on a configurable interval (default 60 s), it executes three independent checks: first, it identifies jobs that remain in a "running" state beyond a timeout threshold (300 s) and marks them as "failed" to recover from pilot, job crashes or network failures. Second, it performs analogous timeout detection for tasks using a longer threshold (3600 s).

And third, it calculates completion percentages for running tasks by aggregating job states and marks tasks as 'finished' when the proportion of completed jobs exceeds a configurable success threshold (70% by default), which enables a partial success. The 70% criterion for successfully completed jobs is optional, and the exact value cannot yet be determined.

MULTIPLE RUNNING WORKFLOWS

To test concurrent execution, we fed the WfMS with a list of primary datasets coming from the data management system (DMS). The DMS registers each initial dataset and sends a trigger to the WfMS via the message broker [7]. To emulate a continuous DAQ (data acquisition system) stream [8], we wrote a script that continuously generated files and registered them with the DMS via RabbitMQ [9]; the DMS then grouped the files into primary datasets (one dataset per minute, each containing 10 × 10 MB files) and forwarded those datasets to the WfMS to trigger workflow execution. Each workflow was based on a CWL template that defined three consecutive tasks: meaning the next one started on output datasets from the previous, and only after that was successfully finished. Each primary dataset started a workflow that spawned tasks, each respectively generated 10 jobs for the pilots; the jobs filled the pilot queue very quickly (Fig. 3). After the dataset generation stopped, pilots continued to consume jobs and the queue gradually drained and stabilised. Each bump represents a new batch of jobs that are ready to be executed and have been pushed by the WMS scheduler.

CONCLUSIONS

We have performed tests on a simulated continuous data stream that enabled us to trigger and execute multiple workflows concurrently, to achieve this the Service Inspector—a monitoring daemon that bridges the, was introduced. We showed that the workflow management system can process several workflows, at the same time the workload management system can execute multiple tasks, generate a sufficient number of jobs, and successfully run them at scale. Our next goal is to collect a complete list of metrics necessary for monitoring the entire system, perform tests on more complex CWL templates with a greater computational load, and develop a more comprehensive scheduling mechanism for managing workflows.

FUNDING

This work was supported by ongoing institutional funding. No additional grants to carry out or direct this particular research were obtained.

CONFLICT OF INTEREST

The authors of this work declare that they have no conflicts of interest.

REFERENCES

1. V. Abazov et al. (SPD Collab.), "Technical design report of the spin physics detector at NICA," Natural Science Review **1**, 1 (2024). https://doi.org/10.54546/naturalscirev.100101
2. N. Greben, L. Romanychev, D. Oleynik, and A. Degtyarev, "SPD on-line filter: Workload management system and pilot agent," Phys. Part. Nucl. **55**, 612–614 (2024). https://doi.org/10.1134/S1063779624030419
3. D. Tereschenko, E. Ponomarev, D. Oleynik, and V. Korkhov, "SPD on-line filter: Workflow and data management systems," Phys. Part. Nucl. **55**, 603–605

(2024).
https://doi.org/10.1134/S1063779624030869

4. Workflow Management System for SPD Online Filter. https://indico.jinr.ru/event/5170/contributions/31753/. Accessed November 18, 2025.

5. N. Greben, D. Oleynik, and L. Romanychev, "Workload management system for SPD online filter," Phys. Part. Nucl. Lett. **22**, 1044–1047 (2025).
https://doi.org/10.1134/s1547477125701067

6. Common Workflow Language. https://www.commonwl.org/. Accessed November 18, 2025.

7. Data Management System for SPD Online Filter. https://indico.jinr.ru/event/5517/contributions/33331/. Accessed November 18, 2025.

8. L. Afanasyev, A. Berngardt, A. Boikov, V. Borshch, A. Boykov, D. Cherepkov, D. Erofeev, V. Frolov, K. Gritsay, A. Isupov, K. Kotov, D. Kozyrev, A. Kulikov, O. Mamoutova, N. Reshetikhin, D. Ryabikov, S. Tereshchenko, and V. Tereshchenko, "The design of the data acquisition system for SPD experiment," J. Instrum. **20**, P11004 (2025).
https://doi.org/10.1088/1748-0221/20/11/p11004

9. RabbitMQ Message Broker. https://www.rabbitmq.com/. Accessed November 19, 2025.

Publisher's Note. Pleiades Publishing remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.
AI tools may have been used in the translation or editing of this article.