Accepted Version

Self-Aware Scientific Workflow Orchestration based on a Blackboard Architecture

Samuel Kounev, David Linke, David Dingel, Nicola Ferger, Lorenz Gruber,
Nikolas Herbst, Maximilian Schwinger, Kyle Chard, and Ian Foster

This is the accepted version of the following paper:

Samuel Kounev, David Linke, David Dingel, Nicola Ferger, Lorenz Gruber, Nikolas Herbst, Maximilian Schwinger, Kyle Chard, and Ian Foster, "Self-Aware Scientific Workflow Orchestration based on a Blackboard Architecture," in *2026 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C)*, ACSOS 2026, Sept. 7–11, Cesena, Italy, 2026.

© 2026 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

The final published version will be available via IEEE Xplore.

BibTeX citation:

```
@inproceedings{KoLiDiFeGrHeScChFo26-ACSOS-Blackboard,
  author = {Kounev, Samuel and Linke, David and Dingel, David and Ferger,
            Nicola and Gruber, Lorenz and Herbst, Nikolas and Schwinger,
            Maximilian and Chard, Kyle and Foster, Ian},
  booktitle = {2026 IEEE International Conference on Autonomic Computing
               and Self-Organizing Systems Companion (ACSOS-C)},
  month = {09},
  publisher = {IEEE},
  title = {Self-Aware Scientific Workflow Orchestration based on a
           Blackboard Architecture},
  year = {2026}
}
```

Self-Aware Scientific Workflow Orchestration based on a Blackboard Architecture

Samuel Kounev
University of Würzburg
Würzburg, Germany
samuel.kounev@uni-wuerzburg.de

David Linke
University of Würzburg
Würzburg, Germany
david.linke@uni-wuerzburg.de

David Dingel
University of Würzburg
Würzburg, Germany
david.dingel@uni-wuerzburg.de

Nicola Ferger
University of Würzburg
Würzburg, Germany
nicola.ferger@stud-mail.uni-wuerzburg.de

Lorenz Gruber
University of Würzburg
Würzburg, Germany
lorenz.gruber@uni-wuerzburg.de

Nikolas Herbst
University of Würzburg
Würzburg, Germany
nikolas.herbst@uni-wuerzburg.de

Maximilian Schwinger
German Aerospace Center (DLR)
Oberpfaffenhofen, Germany
maximilian.schwinger@dlr.de

Kyle Chard
University of Chicago
Chicago, IL, USA
chard@uchicago.edu

Ian Foster
Argonne National Laboratory
Lemont, IL, USA
foster@anl.gov

***Abstract*—Workflow management systems are by now established as a major tool in scientific computing, enabling the composition and execution of complex analysis tasks on distributed computing resources. In this vision paper, we first review the state-of-the-art in workflow management systems, discussing the open challenges and shortcomings of existing work in the area. We then present a vision and propose a research agenda towards self-aware workflow orchestration that aims to integrate online learning and reasoning processes to achieve a new level of self-awareness and operational efficiency. A central part of the envisioned approach is a new blackboard architecture serving as a shared workspace for both the workflow system and the underlying resource management systems. Our aim is to inspire novel research in the area, addressing the gaps in existing work by leveraging on recent advances in machine learning and artificial intelligence.**

***Index Terms*—Workflow systems, scientific computing, workflow orchestration, self-aware computing, performance and resource management**

I. State-of-the-Art in Scientific Workflow Management Systems

Amstutz et al. [1] list more than 360 different workflow systems designed to support diverse scientific applications by automating computational processes, making efficient use of cluster, cloud, and high-performance computing (HPC) infrastructures, and representing complex processing pipelines [2]. Various surveys attempt to review and classify such workflow systems [3]–[10]. Some examples of popular workflow systems used in the context of scientific computing are Pegasus [11], [12], NextFlow [13], [14], Snakemake [15], Hyperflow [16], Argo Workflows [17], Parsl [18], and Apache Airflow [19]. In addition, many more general frameworks and tools have often been used in the context of scientific workflows, such as Globus Compute (funcX) [20], Globus Flows [2], DAGMan [21], KubeFlow [22], Camunda [23], as well as Amazon's Simple Workflow Service [24] and Step Functions [25]. There have been several community-driven initiatives aiming to standardize workflow representations in order to support their portability and reusability in a workflow-engine-independent and vendor-neutral manner [26]–[28].

Modern workflow management systems separate workflow orchestration from resource management tasks. Workflow engines normally send ready-to-run tasks to underlying workload scheduling and resource management systems (also referred to as resource managers), which are responsible for the placement and execution of tasks in a managed infrastructure such as a cluster, cloud, or HPC environment. Resource managers typically execute tasks as batch jobs or inside managed containers. An attempt to abstract the scheduling process used in common resource managers in modern data centers was presented in [29], providing a reference architecture for datacenter scheduling. Some examples of popular resource managers are Kubernetes [30], [31], HTCondor [32], [33], Slurm [34], [35], OpenPBS [36], Nomad [37], Apptainer [38], and Shifter [39]. A survey of scheduling strategies for workflows in cloud environments was presented in [7]. In recent years, workload schedulers specifically targeted at utilizing serverless Function-as-a-Service (FaaS) platforms, such as PyWren [40], NumPyWren [41], [42], Lithops [43], and Fission [44], have started to emerge.

Suter et al. [10] argue that progress in workflows research is hindered by inconsistent terminology, leading to proliferation of workflow systems with overlapping capabilities and

This research is funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 522760169 (FOR 5696) https://dfg-sos.de

ultimately confusion for users. They propose a community-driven terminology that defines core concepts and shared vocabulary for clear comparison and communication. Ejarque et al. [45] argue that the development of modern workflow applications is hampered by the lack of proper programming models and environments that support the integration of HPC, data analytics (DA), and artificial intelligence (AI) tasks, as well as by the lack of tools to easily deploy and execute the workflows in HPC systems. The authors present use cases with complex workflows and discuss the issues that need to be addressed for the HPC/DA/AI convergence. They propose a development approach for a workflow platform comprised of a software stack and HPC Workflow-as-a-Service (HPCWaaS) concept. This effort was part of the EuroHPC eFlows4HPC project [46]. In [47], the challenges in integrating HPC/DA/AI workflows and respective developments were compared to the evolution of the FaaS model in cloud computing, drawing parallels between the eFlows4HPC approach and the serverless computing paradigm.

The Globus research data management platform developed at University of Chicago is an example of a general-purpose platform that facilitates the integration and automation of research process tasks. In the context of workflow management systems, the most relevant parts are Globus Compute (also known as funcX) [20] and Globus Flows [2]. Globus Compute is a distributed FaaS platform that enables scalable remote function execution on federated computing resources with a focus on applications with fine grained tasks [48]. It decouples the cloud-hosted management functionality from the edge-hosted execution functionality using containers (Docker, Singularity, and Shifter) to provide common execution environments across endpoints. Globus Transfer [49], which is based on the GridFTP [50] protocol, enables secure and high-performance data movement between different functions within an endpoint (intra-endpoint). Globus Flows [2] enables managed automation via a hosted Platform-as-a-Service (PaaS) that supports the specification of reusable workflows using a simple declarative notation based on the Amazon States Language [51] used to define Amazon Step Functions [25] state machines in the AWS cloud. Individual steps in a Flow can call *action providers*—external services implementing a common API. The Globs Compute action provider enables tasks to be executed on remote computers. Generally, Globus Flows is not designed for computational workflows involving many tasks, for which specialized workflow tools exist [2].

In summary, the evolution of workflow management systems has been driven by the following major developments and long-term trends:

- Separation of workflow orchestration from resource management tasks, where workflow engines interact with resource managers responsible for the placement and execution of tasks in a managed infrastructure.
- Increasing support for integrated use of federated and heterogeneous computing resources, including workstation, cluster, cloud, and HPC environments, spanning both CPU and GPU resources.
- Increasing use of containers to provide common execution environments across endpoints.
- Increasing integration of machine learning, data science and AI algorithms in scientific workflows executed on HPC infrastructure.

II. Discussion

Although significant progress has been made over the past few decades in workflow scheduling and orchestration, a number of challenges remain. Modern scientific workflows are increasingly dynamic with their behavior often influenced by the initial input data and intermediate results available only during execution [52], [53]. The dynamic nature of workflows coupled with the growing diversity of resource environments makes the prediction of workflow resource demands and execution time challenging [54]–[58]. This is further complicated by the fact that workflows typically exhibit varying degrees of parallelism and resource consumption over their execution path [53]. Generally, the type of components, number of components in each phase, and number of phases heavily depend on the operation and input used to invoke the workflow. Executing such dynamic workflows on traditional HPC clusters with a fixed amount of computing resources leads to over- and/or under-provisioning of resources.

The serverless computing model is increasingly explored as an execution environment for dynamic workflows as it can provide elastic resource provisioning [47], [48], [59], [60]. However, current work on scheduling scientific workflows on serverless computing platforms is limited to the FaaS model [48], [53], [61]–[63], and although showing promising initial results, existing approaches have only been applied to selected scenarios and domains, imposing a number of assumptions and limitations. For example, serverless FaaS functions, which are short-lived and lack direct communication, face significant challenges in managing large amounts of data effectively [64]. The traditional approach of using cloud storage to carry the data in a download–compute–upload sequence is inefficient and expensive. Although some approaches to alleviate this issue have recently been proposed [64], [65], currently no approach exists that can cope with complex data-intensive workflows, such as Earth Observation workflows, where data volume, parallelization, and locality constraints play a central role for the performance, energy efficiency, and scalability of the execution. Furthermore, as argued in [53], the lack of predictive patterns in workflow task invocations, task phases, and their concurrency, makes it challenging to leverage existing serverless techniques for warming up and elastically scaling the required task runtime environment. Effective task invocation and resource demand prediction mechanisms are required to minimize keep alive costs and avoid cold starts for serverless computing to make it beneficial for dynamic workflows.

Another important issue is that while (in an increasingly heterogeneous resource landscape) the decoupling of the resource management layer from the workflow orchestration layer in existing workflow management systems has many advantages,

there is increasing realization that this strict separation of concerns comes with some costs and drawbacks [45], [66], [67]. The fact that common HPC resource allocation policies and schedulers are not designed for workflow-aware abstractions was identified as a challenge in the recent workflow community summits [68], [69]. Indeed, given that resource managers are normally unaware of the workflow structure and task dependencies, their task placement and scheduling decisions may be suboptimal from a performance and resource efficiency standpoint. Further, while the workflow engine will normally have knowledge about the abstract workflow structure and task dependencies, as already discussed, the exact execution order may depend on the initial input data and intermediate results only available during execution.

These observations motivate increasing efforts to enable workflow-aware scheduling by providing underlying resource managers with dynamic information on workflow structure, resource requirements, and current execution plan to enable better scheduling decisions [52], [66], [67]. Yet individual resource managers still make decisions largely independently of one another, without the ability to coordinate or to provide feedback that may influence decisions at the workflow orchestration layer. Indeed, in reality, a placement and scheduling decision made by a resource manager may render the current workflow execution plan suboptimal with respect to the overall workflow priorities and execution deadlines.

Thus, to further improve efficiency, we see the need for *bidirectional synchronization and exchange of information* from the workflow orchestration layer to the underlying resource managers and the other way round. On the one hand, efficient scheduling and prioritization of tasks at the workflow orchestration level should take into account the placement and resource allocation decisions of the individual resource managers. On the other hard, optimizing task placement and scheduling by individual resource managers should take into account the scheduling decisions at the workflow orchestration level (e.g., current execution plan, expected processing phases, and resource demands).

Further, learning accurate models for predicting the workflow and task resource demands and execution times requires information and context from both the workflow orchestration and resource management layers [54], [56], [57]. For example, monitoring data at the workflow level combined with resource consumption measurements could be used by the workflow orchestration system to learn models capturing task execution phases, resource demands, and parametric dependencies. Finally, as argued in [70], novel techniques to automatically identify identical computations across workflows are needed to reduce energy consumption by avoiding redundant computations across multiple workflows or repeated executions of the same workflow.

III. Vision and Research Agenda

We now propose a long-term vision and research agenda towards self-aware workflow orchestration, inspired by the self-aware computing paradigm [71], [72]; our goal is to inspire novel research in the area, addressing the described challenges by leveraging on recent advances in machine learning and artificial intelligence. The presented vision is pursued in the context of a recently started large-scale research project SOS.[1]

> Self-aware computing systems are computing systems that:
> 1) *learn models* capturing *knowledge* about themselves and their environment (such as their structure, design, state, possible actions, and run-time behavior) on an ongoing basis, and
> 2) *reason* using the models (for example predict, analyze, consider, plan) enabling them to *act* based on their knowledge and reasoning (for example explore, explain, report, suggest, self-adapt, or impact their environment),
>
> in accordance with *higher-level goals*, which may also be subject to change [72].

The continuous learning and reasoning enable model-based analysis at run-time that goes beyond applying simple rules, heuristics, or feedback loops, explicitly programmed at system design-time. Figure 1 illustrates the concept of a model-based learning and reasoning loop (LRA-M loop) capturing the main activities in a self-aware computing system. The figure shows the *self* and its interfaces to the environment in which it is operating. The activities within the self are driven by its goals and its observations collected as empirical data about relevant aspects of the system and its environment, its users, and so on. The empirical data is used as a basis for the ongoing *learning* process, as part of which observations are abstracted into models capturing potentially relevant aspects of the system and its environment (such as their structure, design, state, possible actions, and run-time behavior). The models are used as a basis for the *reasoning* process, which may result in *actions* affecting both the behavior of the system itself (self-adaptation) as well as possibly impacting the environment.

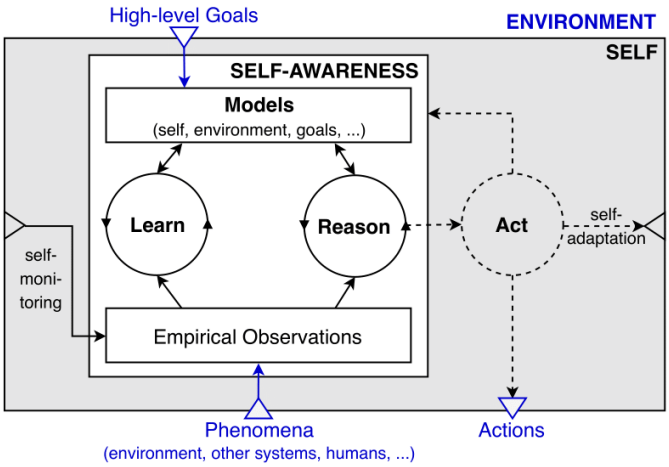

Fig. 1. Self-aware learning and reasoning loop [72].

[1]DFG Research Unit 5696 (SOS): https://dfg-sos.de

In the context of scientific workflows, a scientist executing a workflow should only be required to specify execution goals and constraints at a high level of abstraction. These may include optimization objectives (such as time-critical, cost-efficient, or carbon-neutral execution), execution deadlines (time-to-result), budget constraints, infrastructure constraints (e.g., dedicated resources versus public clouds, potentially restricted to certain geographical regions), and reproducibility requirements. Based on these parameters, the workflow platform should automatically optimize the workflow orchestration through self-aware learning and reasoning loops [71] integrating system monitoring and diverse predictive modeling techniques to achieve a new level of self-awareness and operational efficiency.

A. Approach

To realize this vision and address the challenges discussed in Section II, a major goal of the proposed research agenda is to design a flexible and dynamic architecture, through which the workflow orchestration system (WOS) and the underlying resource management systems (RMS), spanning cluster, cloud, and HPC environments, can share relevant information (data and models) and interact to collectively optimize workflow executions and resource efficiency. The architecture includes online performance models to capture relevant workflow aspects for predicting task execution phases and their resource demands as input to the task placement and scheduling algorithms. Further, the models should enable predicting the total resource consumption and time-to-result of a workflow execution.

The approach by Lehmann et al. [52] can be used as a starting point for our design of an interaction protocol between the WOS and the underlying RMS, enabling knowledge exchange for coordinated and workflow-aware resource management. As discussed in Section II, we see the need for bidirectional synchronization and exchange of information between the WOS and RMS. On the one hand, efficient scheduling and prioritization of tasks at the WOS level should take into account the placement and resource allocation decisions of the individual RMS. On the other hard, optimizing task placement and scheduling at the individual RMS should take into account the decisions at the WOS level (i.e., current execution plan, expected processing phases, and resource demands). While the WOS will normally have knowledge about the abstract workflow structure and task dependencies, the exact execution order may depend on the initial input data and intermediate results only available during execution.

We envision a novel coordinated and collaborative approach to workflow orchestration and resource management, where the WOS layer is designed to closely interact with the underlying RMS. The goal is to enable coordinated mutual scheduling, placement, and resource management decisions, where the different layers can influence each other to achieve better overall performance and energy efficiency. We envision an architecture illustrated in Figure 2, inspired by the blackboard paradigm and software architecture pattern [73]. This architectural style originally emerged in the AI domain as an approach to integrate cooperating software modules in order to achieve a flexible, brainstorming style of problem solving; the blackboard is a shared repository containing the data from various problem solving states [74], [75].

We propose to build upon the concept of *coordinated reflection* [76]. The goal is to design a flexible and dynamic architecture based on a blackboard that will serve as a shared knowledge base and workspace, through which the WOS and RMS will share relevant information (data and models) and interact to collectively optimize workflow executions and resource efficiency. The WOS should be designed to manage a set of online models for each executed workflow and task, capturing relevant aspects for optimizing scheduling and resource management efficiency. The workflow models should be based on static information obtained from the workflow design models and specifications (e.g., abstract execution graphs, input and output data, execution schemes) combined with dynamic information obtained at run time (e.g., task processing phases and respective resource demands, as well as parametric dependencies capturing the influences of input data and configuration parameters on the workflow and task behavior). During the execution of a workflow, the workflow models and/or relevant data extracted from them should be placed on the blackboard such that they can be leveraged by the RMS to optimize resource management decisions. Similarly, the various RMS should maintain local information (data and models) for each executed task and will share relevant information that may potentially be of use to the WOS by storing such information on the blackboard. For example, resource consumption measurements and statistics could be used by the WOS to learn models capturing the task resource demands and parametric dependencies. RMS may further provide information how to access stored execution traces, which could in turn be used by the WOS to extract models of the task execution phases.

The blackboard should be designed to maintain an annotated physical execution graph for each executed workflow. The graph will capture the current state of the workflow execution, including relevant information on: (1) workflow execution priorities and constraints, (2) scheduled, ready-to-run, completed, and upcoming tasks, (3) current task placement and scheduling decisions, (4) estimated actual resource consumption and task completion times, and (5) requested system-level monitoring data. The blackboard can thus serve as a shared workspace providing the WOS and the RMS up-to-date information on the state of their current decisions and execution plans, while allowing them to also use this information to interact and coordinate their scheduling, placement, and resource management decisions.

To give an example, when a workflow execution is started, the WOS will first load the relevant workflow models on the blackboard and then instantiate the physical execution graph (based on the abstract execution graph), while adding (annotated) information on the task priorities, constraints, and requested system-level monitoring. The WOS will then mark

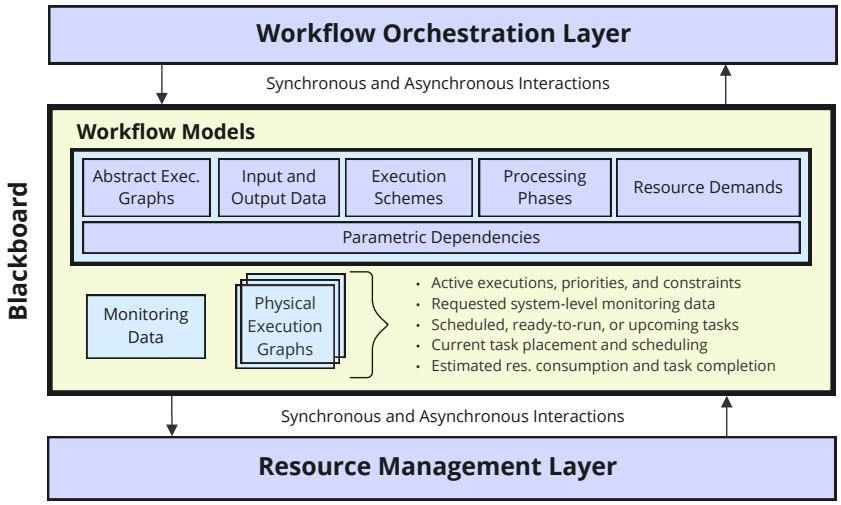

Fig. 2. Envisioned blackboard architecture for self-aware workflow orchestration and resource management.

all ready-to-run tasks on the graph and notify the respective RMS. The latter can then examine the blackboard and start making placement and scheduling decisions (changing the state of respective tasks from 'ready-to-run' to 'scheduled'), while leveraging all relevant information to optimize resource efficiency and taking into account the workflow priorities and constraints such as execution deadlines. Based on the current execution plan, the RMS can add information to the physical execution graph about the expected task completion times. This in turn may influence future decisions made by the WOS and in some cases may even lead to the WOS revising a previously made decision, for example, if a given set of tasks is expected to take too long to complete, blocking resources needed for executing other higher priority workflows that may have arrived in the mean time.

B. Research Questions and Challenges

We now formulate a set of research questions that need to be answered in order to realize the proposed approach. These questions are intended as a research agenda, aimed at spawning new research directions and interdisciplinary collaborations. We expect that some of the questions may be approached in a generic manner, while others may be considered for different subdomains within the domain of scientific workflows (e.g., geosciences vs. genomics). Understanding under what assumptions individual questions can be addressed in a satisfactory manner is a research challenge in itself.

1) Blackboard Architecture for Self-Aware Workflow Orchestration and Resource Management:

- Which specific tasks and responsibilities should be assigned to the WOS and RMS layers to ensure a balance between low coupling and high cohesion (separation of concerns), on the one hand, and support for coordinated and optimized decision making, on the other hand?
- Which knowledge (data and models) should be maintained and managed locally in the WOS and RMS layers and which knowledge should be shared between the layers using the blackboard?
- Which parts of the blackboard should be managed by the WOS and which by the RMS? What access rights (read, write) should be assigned to different participants for accessing information stored on the blackboard?
- How should the blackboard be designed (structure and topology) and physically implemented in order to support fast, efficient, and reliable concurrent access by multiple independent parties without the danger of it becoming a bottleneck or reaching an inconsistent state?
- What decentralized architectures and access protocols can be used to avoid the blackboard turning into a single point of failure?
- How should information exchange be realized: Which data should be passed through asynchronous (event-based) vs. synchronous communication?
- What protocols should be used to enable fast and efficient access to external data (e.g., raw monitoring data such as execution traces) that is not part of the blackboard but is stored locally at the RMS or the WOS?

2) Workflow Scheduling, Monitoring, and Provenance for Fault-Tolerant, Explainable, and Reproducible Execution:

- For each executed workflow, what specific information on the workflow state and execution plan should be stored as part of the annotated physical execution graph?
- Which parts of the execution graph should be managed by the WOS and which by the RMS?
- How to design scheduling algorithms taking into account factors such as constraints, workflow prioritization, user quotas, budget limits, as well as targeted tradeoff between processing costs and time-to-result?

- How to dynamically map end-to-end workflow deadlines to individual task deadlines to allow for maximum flexibility such that contention- and energy-aware task placement and scheduling can be leveraged?
- What operational metrics need to be monitored to track progress and resource consumption of different users for workflow and task execution?
- What interfaces should be used to communicate system-level monitoring requirements to the RMS and integrate provided monitoring data?
- What workflow and task-level provenance data should be stored and managed by the WOS and by the individual RMS to support the explainability and reproducibility of workflow executions?
- How to track progress in workflow and task execution such that workflow execution can be automatically restarted and resumed with minimum loss of work after a failure?

3) Online Performance Models for Run-Time Optimization, Resource Demand Estimation, and Time-to-Result Prediction:

- What data should be shared between the WOS and RMS as a basis for learning online performance models?
- What models are suitable to capture task execution phases, their parallelism, and their resource consumption?
- How can parametric dependencies be modeled?
- What interfaces should be used by RMS to access relevant information from online performance models as input for contention- and energy-aware task placement and scheduling?
- How can information on the expected workload be passed to the RSM in a proactive manner, so that it can be used for warming up and elastically scaling the required task runtime environment?
- How can the developed models be used to predict the overall time-to-result of a given workflow execution?
- How to manage identity of individual task executions applied on the same data in order to automatically identify identical computations across workflows?
- How to support caching and reuse of processing results from multiple task executions on the same data?

While the envisioned self-aware learning and reasoning processes will introduce some overhead, the potential efficiency gains are expected to far outweigh the incurred costs. As argued in [70], scientific workflows are particularly energy intensive and the potential to reduce their energy costs and carbon footprint is significant; novel approaches for energy-aware task placement, scheduling, and research data management, as well as techniques to automatically identify and avoid identical computations across workflows, are urgently needed as energy costs skyrocket.

C. Towards Proof-of-Concept

As a first step towards the described vision, we have implemented an initial prototype of the envisioned blackboard architecture.

To overcome the aforementioned challenges, it is essential to establish a robust and extensible foundation. In our framework, we distinguish between the static aspect of workflows and their dynamic execution. A static workflow model is independent of user input and can be reused. Our workflow models are based on the Common Workflow Language (CWL) [26], which facilitates CWL workflow execution, supports the community standard, and promotes the FAIR principles. Workflow execution involves instantiating the static model and combining it with user input. The key difference is that a single task from the static workflow model can be instantiated as multiple jobs in the execution graph if parallelization is required. The number of jobs normally depends on the user input provided at the beginning of execution.

The *Blackboard* is the primary component of the framework. It serves as a central orchestration engine that receives, models, and stores workflows; maintains resource-related metadata; and coordinates all framework components. It combines static workflow models with dynamic run-time information. Internally, the *Blackboard* consists of a workflow manager and an execution manager that use a publish/subscribe mechanism to respond to job-state changes. Thus, external components communicating with the *Blackboard* can subscribe to specific objects, enabling precise, event-driven notifications. Workflow execution is handled by the *PlatformAdapter*, which is placed on each heterogeneous execution platform and supports different execution backends. We can adjust these adapters to specific platforms that connect to the *Blackboard*, enabling a global view of the resource landscape and bidirectional communication for exchanging control-plane information. The *scheduler* is responsible for scheduling and placing jobs based on information from the *Blackboard*. The scheduling algorithms consider constraints, user quotas, budget limits, workflow priorities, and efficiency. To execute heterogeneous workflows across multiple execution platforms, we must track and transfer the files required as job inputs. The *FileManagement* component handles these responsibilities. It maintains a central file registry and coordinates file transfers between registered execution platforms.

We have already tested the prototype with some basic synthetic workflows to demonstrate its parallelization capabilities. Currently, we are executing real-world heterogeneous scientific workflows that require the ability to run different steps on different platforms, all of which are orchestrated through a blackboard.

IV. Conclusion

We reviewed the state-of-the-art in workflow management systems, discussing the shortcomings of existing work, and then presented a long-term vision and research agenda towards self-aware workflow orchestration. A central part of our envisioned approach is a new blackboard architecture serving as a shared workspace of the workflow system and the underlying resource management systems. The vision is pursued as part of a large-scale research project SOS (https://dfg-sos.de).

References

[1] P. Amstutz, M. Mikheev, M. R. Crusoe, N. Tijanić, S. Lampa *et al.*, "Existing workflow systems," Common Workflow Language wiki, GitHub, 2024, updated August 18, 2024, accessed August 18, 2024. [Online]. Available: https://s.apache.org/existing-workflow-systems

[2] R. Chard, J. Pruyne, K. McKee, J. Bryan, B. Raumann, R. Ananthakrishnan, K. Chard, and I. T. Foster, "Globus automation services: Research process automation across the space–time continuum," *FGCS*, vol. 142, pp. 393–409, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X23000183

[3] C. S. Liew, M. P. Atkinson, M. Galea, T. F. Ang, P. Martin, and J. I. V. Hemert, "Scientific workflows: Moving across paradigms," *ACM Computing Surveys*, vol. 49, no. 4, dec 2016. [Online]. Available: https://doi.org/10.1145/3012429

[4] E. Deelman, D. Gannon, M. Shields, and I. Taylor, "Workflows and e-science: An overview of workflow system features and capabilities," *Future Generation Computer Systems*, vol. 25, no. 5, pp. 528–540, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X08000861

[5] J. Liu, E. Pacitti, P. Valduriez, and M. Mattoso, "A survey of data-intensive scientific workflow management," *Journal of Grid Computing*, vol. 13, no. 4, pp. 457–493, Dec. 2015.

[6] R. Ferreira da Silva, R. Filgueira, I. Pietri, M. Jiang, R. Sakellariou, and E. Deelman, "A characterization of workflow management systems for extreme-scale applications," *Future Generation Computer Systems*, vol. 75, pp. 228–238, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X17302510

[7] M. Adhikari, T. Amgoth, and S. N. Srirama, "A survey on scheduling strategies for workflows in cloud environment and emerging trends," *ACM Computing Surveys*, vol. 52, no. 4, aug 2019. [Online]. Available: https://doi.org/10.1145/3325097

[8] M. Atkinson, S. Gesing, J. Montagnat, and I. Taylor, "Scientific workflows: Past, present and future," *Future Generation Computer Systems*, vol. 75, pp. 216–227, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X17311202

[9] J. Yu and R. Buyya, "A taxonomy of scientific workflow systems for grid computing," *ACM SIGMOD Record*, vol. 34, no. 3, p. 44–49, sep 2005. [Online]. Available: https://doi.org/10.1145/1084805.1084814

[10] F. Suter, T. Coleman, İ. Altintaş, R. M. Badia, B. Balis, K. Chard, I. Colonnelli, E. Deelman, P. Di Tommaso, T. Fahringer, C. Goble, S. Jha, D. S. Katz, J. Köster, U. Leser, K. Mehta, H. Oliver, J.-L. Peterson, G. Pizzi, L. Pottier, R. Sirvent, E. Suchyta, D. Thain, S. R. Wilkinson, J. M. Wozniak, and R. Ferreira da Silva, "A terminology for scientific workflow systems," *Future Generation Computer Systems*, vol. 174, p. 107974, 2026. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X25002699

[11] E. Deelman, K. Vahi, M. Rynge, R. Mayani, R. F. da Silva, G. Papadimitriou, and M. Livny, "The evolution of the pegasus workflow management software," *Computing in Science & Engineering*, vol. 21, no. 4, pp. 22–36, 2019.

[12] "Pegasus project," https://pegasus.isi.edu/, accessed: 2024-04-05.

[13] P. D. Tommaso, M. Chatzou, E. W. Floden, P. P. Barja, E. Palumbo, and C. Notredame, "Nextflow enables reproducible computational workflows," *Nature Biotechnology*, vol. 35, no. 4, pp. 316–319, Apr. 2017.

[14] "Nextflow project," https://www.nextflow.io/, accessed: 2024-04-05.

[15] "Snakemake project," https://snakemake.github.io/, accessed: 2024-04-05.

[16] "Hyperflow project," https://github.com/hyperflow-wms/hyperflow, accessed: 2024-04-05.

[17] "Argo workflows," https://argoproj.github.io/argo-workflows/, accessed: 2024-04-05.

[18] "Parsl project," https://parsl-project.org/, accessed: 2024-04-05.

[19] "Apache airflow," https://airflow.apache.org/, accessed: 2024-04-05.

[20] Z. Li, R. Chard, Y. Babuji, B. Galewsky, T. J. Skluzacek, K. Nagaitsev, A. Woodard, B. Blaiszik, J. Bryan, D. S. Katz, I. Foster, and K. Chard, "funcx: Federated function as a service for science," *IEEE TPDS*, vol. 33, no. 12, pp. 4948–4963, 2022.

[21] "Dagman," https://htcondor.org/dagman/dagman.html, accessed: 2024-04-05.

[22] "Kubeflow," https://www.kubeflow.org/, accessed: 2024-04-05.

[23] "Camunda platform," https://camunda.com/, accessed: 2024-04-05.

[24] Amazon Web Services (AWS), "Amazon simple workflow service," https://aws.amazon.com/swf/, accessed: 2024-04-05.

[25] ——, "Aws step functions," https://aws.amazon.com/step-functions/, accessed: 2024-04-05.

[26] "Common workflow language (cwl)," https://www.commonwl.org, accessed: 2024-04-05.

[27] M. R. Crusoe, S. Abeln, A. Iosup, P. Amstutz, J. Chilton, N. Tijanić, H. Mé nager, S. Soiland-Reyes, B. Gavrilović, and C. Goble, "Methods included: Standardizing computational reuse and portability with the cwl," *CACM*, vol. 65, no. 6, p. 54–63, may 2022. [Online]. Available: https://doi.org/10.1145/3486897

[28] "Workflow description language (wdl)," https://github.com/openwdl/wdl, accessed: 2024-04-05.

[29] G. Andreadis, L. Versluis, F. Mastenbroek, and A. Iosup, "A reference architecture for datacenter scheduling: Design, validation, and experiments," in *SC: Intl. Conf. for High-Perf. Computing, Networking, Storage and Analysis*, 2018, pp. 478–492.

[30] "Kubernetes project," https://kubernetes.io/, accessed: 2024-04-05.

[31] B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes, "Borg, omega, and kubernetes: Lessons learned from three container-management systems over a decade," *Queue*, vol. 14, no. 1, p. 70–93, jan 2016. [Online]. Available: https://doi.org/10.1145/2898442.2898444

[32] "Htcondor software suite," https://htcondor.org/, accessed: 2024-04-05.

[33] D. Thain, T. Tannenbaum, and M. Livny, "Distributed computing in practice: The condor experience: Research articles," *Concurrency and Computation: Practice and Experience*, vol. 17, no. 2–4, p. 323–356, feb 2005.

[34] "Slurm workload manager," https://slurm.schedmd.com/, accessed: 2024-04-05.

[35] A. B. Yoo, M. A. Jette, and M. Grondona, "Slurm: Simple linux utility for resource management," in *Job Scheduling Strategies for Parallel Processing*, D. Feitelson, L. Rudolph, and U. Schwiegelshohn, Eds. Berlin, Heidelberg: Springer, 2003, pp. 44–60.

[36] "Openpbs project," https://www.openpbs.org/, accessed: 2024-04-05.

[37] "Nomad project," https://www.nomadproject.io/, accessed: 2024-04-05.

[38] "Apptainer project," https://apptainer.org/, accessed: 2024-04-05.

[39] "Shifter nersc project," https://docs.nersc.gov/development/containers/shifter/, accessed: 2024-04-07.

[40] "Pywren," http://pywren.io/, accessed: 2024-04-05.

[41] V. Shankar, K. Krauth, K. Vodrahalli, Q. Pu, B. Recht, I. Stoica, J. Ragan-Kelley, E. Jonas, and S. Venkataraman, "Serverless linear algebra," in *Proceedings of the 11th ACM Symposium on Cloud Computing*, ser. SoCC '20. New York, NY, USA: ACM, 2020, p. 281–295. [Online]. Available: https://doi.org/10.1145/3419111.3421287

[42] V. Shankar, K. Krauth, Q. Pu, E. Jonas, S. Venkataraman, I. Stoica, B. Recht, and J. Ragan-Kelley, "numpywren: serverless linear algebra," *arXiv:2305.08409*, 2018.

[43] "Lithops project," https://github.com/lithops-cloud/lithops, accessed: 2024-04-05.

[44] "Fission: Open source functions-as-a-service for kubernetes," https://platform9.com/fission/, accessed: 2024-04-05.

[45] J. Ejarque, R. M. Badia, L. Albertin, G. Aloisio, E. Baglione, Y. Becerra, S. Boschert, J. R. Berlin, A. D'Anca, D. Elia, F. Exertier, S. Fiore, J. Flich, A. Folch, S. J. Gibbons, N. Koldunov, F. Lordan, S. Lorito, F. Løvholt, J. Macías, F. Marozzo, A. Michelini, M. Monterrubio-Velasco, M. Pienkowska, J. de la Puente, A. Queralt, E. S. Quintana-Ortí, J. E. R. guez, F. Romano, R. Rossi, J. Rybicki, M. Kupczyk, J. Selva, D. Talia, R. Tonini, P. Trunfio, and M. Volpe, "Enabling dynamic and intelligent workflows for HPC, data analytics, and AI convergence," *Future Generation Computer Systems*, vol. 134, pp. 414–429, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X22001364

[46] eFlows4HPC Consortium, "eflows4hpc project website," https://eflows4hpc.eu/, accessed: 2024-04-05.

[47] J. Ejarque, "Keynote: Introducing the faas model in complex hpc workflows: The eflows4hpc approach," 31st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP 2023), Mar. 2023, https://eflows4hpc.eu/wp-content/uploads/2023/03/eFlow4HPC_PDP_2023.pdf; Accessed: 2024-04-05.

[48] M. Malawski and B. Balis, "Serverless comp. for scient. appl." *IEEE Inter. Comp.*, vol. 26, no. 4, pp. 53–58, 2022.

[49] K. Chard, S. Tuecke, and I. Foster, "Efficient and secure transfer, synchronization, and sharing of big data," *IEEE Cloud Computing*, vol. 1, no. 3, pp. 46–55, 2014.

[50] W. Allcock, J. Bresnahan, R. Kettimuthu, and M. Link, "The globus striped gridftp framework and server," in *SC'05: Proceedings of the 2005 ACM/IEEE Conference on Supercomputing*, 2005, pp. 54–54.

[51] "Amazon states language," https://states-language.net/, accessed: 2024-04-05.

[52] F. Lehmann, J. Bader, F. Tschirpke, L. Thamsen, and U. Leser, "How workflow engines should talk to resource managers: A proposal for a common workflow scheduling interface," in *IEEE/ACM CCGrid 2023*. Los Alamitos, CA, USA: IEEE, may 2023, pp. 166–179. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CCGrid57682.2023.00025

[53] R. B. Roy, T. Patel, and D. Tiwari, "Daydream: Executing dynamic scientific workflows on serverless platforms with hot starts," in *SC'22: Intl. Conf. on High Performance Computing, Networking, Storage and Analysis*. IEEE, 2022.

[54] J. Bader, F. Lehmann, L. Thamsen, U. Leser, and O. Kao, "Lotaru: Locally predicting workflow task runtimes for resource management on heterogeneous infrastructures," *Future Generation Computer Systems*, vol. 150, pp. 171–185, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X23003229

[55] J. Bader, N. Diedrich, L. Thamsen, and O. Kao, "Predicting dynamic memory requirements for scientific workflow tasks," in *2023 IEEE International Conference on Big Data (Big Data)*, Dec. 2023.

[56] A. M. Chirkin, A. S. Belloum, S. V. Kovalchuk, M. X. Makkes, M. A. Melnik, A. A. Visheratin, and D. A. Nasonov, "Execution time estimation for workflow scheduling," *Future Generation Computer Systems*, vol. 75, pp. 376–387, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X17300304

[57] C. Witt, M. Bux, W. Gusew, and U. Leser, "Predictive performance modeling for distributed batch processing using black box monitoring and machine learning," *Information Systems*, vol. 82, pp. 33–52, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306437918301765

[58] K. Ousterhout, C. Canel, S. Ratnasamy, and S. Shenker, "Monotasks: Architecting for performance clarity in data analytics frameworks," in *Proceedings of the 26th Symposium on Operating Systems Principles*, ser. SOSP'17. New York, NY, USA: ACM, 2017, p. 184–200. [Online]. Available: https://doi.org/10.1145/3132747.3132766

[59] E. Jonas, Q. Pu, S. Venkataraman, I. Stoica, and B. Recht, "Occupy the cloud: Distributed computing for the 99%," in *Proceedings of the 2017 Symposium on Cloud Computing*, ser. SoCC'17. New York, NY, USA: Association for Computing Machinery, 2017, p. 445–451. [Online]. Available: https://doi.org/10.1145/3127479.3128601

[60] S. Werner and S. Tai, "A reference architecture for serverless big data processing," *Future Generation Computer Systems*, vol. 155, pp. 179–192, Jun. 2024.

[61] M. Malawski, A. Gajek, A. Zima, B. Balis, and K. Figiela, "Serverless execution of scientific workflows: Experiments with hyperflow, aws lambda and google cloud functions," *Future Generation Computer Systems*, vol. 110, pp. 502–514, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X1730047X

[62] M. Majewski, M. Pawlik, and M. Malawski, "Algorithms for scheduling scientific workflows on serverless architecture," in *2021 IEEE/ACM 21st International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, 2021, pp. 782–789.

[63] S. Burckhardt, B. Chandramouli, C. Gillum, D. Justo, K. Kallas, C. McMahon, C. S. Meiklejohn, and X. Zhu, "Netherite: Efficient execution of serverless workflows," *Proc. VLDB Endow.*, vol. 15, no. 8, p. 1591–1604, apr 2022. [Online]. Available: https://doi.org/10.14778/3529337.3529344

[64] D. Barcelona-Pons, P. García-López, and B. Metzler, "Glider: Serverless Ephemeral Stateful Near-Data Computation," in *Proceedings of the 24th Intl. Middleware Conference*, ser. Middleware '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 247–260. [Online]. Available: https://doi.org/10.1145/3590140.3629119

[65] M. Abdi, S. Ginzburg, X. C. Lin, J. Faleiro, G. I. Chaudhry, I. Goiri, R. Bianchini, D. S. Berger, and R. Fonseca, "Palette load balancing: Locality hints for serverless functions," in *Proceedings of EuroSys 2023*. New York, NY, USA: ACM, 2023, p. 365–380. [Online]. Available: https://doi.org/10.1145/3552326.3567496

[66] J. Bader, J. Belak, M. Bement, M. Berry, R. Carson, D. Cassol, S. Chan, J. Coleman, K. Day, A. Duque, K. Fagnan, J. Froula, S. Jha, D. S. Katz, P. Kica, V. Kindratenko, E. Kirton, R. Kothadia, D. Laney, F. Lehmann, U. Leser, S. Lichołai, M. Malawski, M. Melara, E. Player, M. Rolchigo, S. Sarrafan, S.-J. Sul, A. Syed, L. Thamsen, M. Titov, M. Turilli, S. Caino-Lores, and A. Mandal, "Novel approaches toward scalable composable workflows in hyper-heterogeneous computing environments," in *Workshops of 2023 SC Conference (SC-W'23)*, New York, NY, USA, 2023, p. 2097–2108. [Online]. Available: https://doi.org/10.1145/3624062.3626283

[67] G. P. Rodrigo, E. Elmroth, P.-O. Östberg, and L. Ramakrishnan, "Enabling workflow-aware scheduling on hpc systems," in *26th Intl. Symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC'17. New York, NY, USA: ACM, 2017, p. 3–14. [Online]. Available: https://doi.org/10.1145/3078597.3078604

[68] R. F. da Silva *et al.*, "Workflows cmty. summit 2022: A roadmap revolution."

[69] R. F. da Silva, H. Casanova, K. Chard, I. Altintas, R. M. Badia, B. Balis, T. a. Coleman, F. Coppens, F. Di Natale, B. Enders, T. Fahringer, R. Filgueira, G. Fursin, D. Garijo, C. Goble, D. Howell, S. Jha, D. S. Katz, D. Laney, U. Leser, M. Malawski, K. Mehta, L. Pottier, J. Ozik, J. L. Peterson, L. Ramakrishnan, S. Soiland-Reyes, D. Thain, and M. Wolf, "A community roadmap for scientific workflows research and development," in *2021 IEEE Workshop on Workflows in Support of Large-Scale Science (WORKS)*, 2021, pp. 81–90.

[70] U. Leser and L. Thamsen, "The elephant in the data center – energy consumption of workflow executions," https://workflows.community/stories/2023/01/13/elephant-in-the-data-center/, Jan. 2023, accessed: 2024-04-05.

[71] S. Kounev, J. O. Kephart, A. Milenkoski, and X. Zhu, Eds., *Self-Aware Computing Systems*. Berlin Heidelberg, Germany: Springer Verlag, 2017. [Online]. Available: http://www.springer.com/de/book/9783319474724

[72] S. Kounev, P. Lewis, K. Bellman, N. Bencomo, J. Camara, A. Diaconescu, L. Esterle, K. Geihs, H. Giese, S. Götz, P. Inverardi, J. Kephart, and A. Zisman, "The notion of self-aware computing." Berlin Heidelberg, Germany: Springer Verlag, 2017.

[73] F. Buschmann, R. Meunier, H. Rohnert, and P. Sommerlad, *Pattern-Oriented Software Architecture, Vol. 1: A System of Patterns*. Chichester: Wiley, 1996.

[74] D. Corkill, "Collaborating software: Blackboard and multi-agent systems & the future," in *Intl. Conf. on Information Fusion*, 2005.

[75] M. Reddy and G. O'Hare, "The blackboard model: a survey of its application," *Artif. Int. Review*, vol. 5, pp. 169–186, 1991. [Online]. Available: https://doi.org/10.1007/BF00143760

[76] H. Giese, T. Vogel, A. Diaconescu, S. Götz, and K. L. Bellman, *Generic Architectures for Individual Self-aware Computing Systems*. Cham: Springer, 2017, pp. 149–189.