The Common Workflow Scheduler Interface: Status Quo and Future Plans

Fabian Lehmann
fabian.lehmann@informatik.hu-berlin.de
Humboldt-Universität zu Berlin
Berlin, Germany

Jonathan Bader
jonathan.bader@tu-berlin.de
Technische Universität Berlin
Berlin, Germany

Lauritz Thamsen
lauritz.thamsen@glasgow.ac.uk
University of Glasgow
Glasgow, United Kingdom

Ulf Leser
leser@informatik.hu-berlin.de
Humboldt-Universität zu Berlin
Berlin, Germany

ABSTRACT

Nowadays, many scientific workflows from different domains, such as Remote Sensing, Astronomy, and Bioinformatics, are executed on large computing infrastructures managed by resource managers. Scientific workflow management systems (SWMS) support the workflow execution and communicate with the infrastructures' resource managers. However, the communication between SWMS and resource managers is complicated by a) inconsistent interfaces between SMWS and resource managers and b) the lack of support for workflow dependencies and workflow-specific properties.

To tackle these issues, we developed the Common Workflow Scheduler Interface (CWSI), a simple yet powerful interface to exchange workflow-related information between a SWMS and a resource manager, making the resource manager workflow-aware. The first prototype implementations show that the CWSI can reduce the makespan already with simple but workflow-aware strategies up to 25%. In this paper, we show how existing workflow resource management research can be integrated into the CWSI.

KEYWORDS

Scientific Workflow, Scheduling, Workflow Management System, Cluster Resource Management, Common Workflow Scheduler

ACM Reference Format:
Fabian Lehmann, Jonathan Bader, Lauritz Thamsen, and Ulf Leser. 2023. The Common Workflow Scheduler Interface: Status Quo and Future Plans. In *Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis (SC-W '23), November 12–17, 2023, Denver, CO, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3624062.3626283

This work was presented at the 18th Workshop on Workflows in Support of Large-Scale Science (WORKS 2023) and was published as part of the workshop paper "Novel Approaches Toward Scalable Composable Workflows in Hyper-Heterogeneous Computing Environments" in the Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis (SC-W '23) https://doi.org/10.1145/3624062.3626283

1 INTRODUCTION

Analyzing large datasets is the daily business of many scientists [13, 17, 21, 22, 29]. The data analysis often involves multiple dependent steps, which can be organized as a workflow [16]. As these workflows are becoming increasingly complex and datasets easily exceed hundreds of gigabytes or even terabytes [1, 17], scientists use scientific workflow management systems (SWMS), such as Nextflow, Airflow, or Argo, and computer clusters. One essential feature of SMWSs is communication with a resource manager, such as SLURM, Kubernetes, or OpenPBS. Therefore, the SWMS submits ready-to-run tasks to the resource manager, and the resource manager takes over the responsibility for assigning these tasks to a node that executes them. This simplifies the workflow execution on large-scale computing infrastructures and hides the complexity from the scientist. However, as with SWMS, there is also a variety of resource managers available, and different clusters may use a different one. In a worst-case scenario, the SWMS preferred by the scientist does not support the cluster's resource manager at all. Even if the SWMS supports a given resource manager, features beyond submitting tasks and awaiting their completion are frequently not supported.

In this paper, we first give an overview of the Common Workflow Scheduler (CWS) and the Common Workflow Scheduler Interface (CWSI) which we both first presented in [15]. The CWSI is used to exchange workflow-related information between SWMSs and resource managers. Second, we present prior results, showing promising outcomes when using the CWSI with workflow-aware resource management methods. Moving on, we outline SWMS, where we started to implement CWSI support and demonstrate how the CWS can serve as a central point for provenance. Last, we illustrate how the CWS can be extended with new scheduling, resource allocation, and runtime prediction methods.

2 COMMON WORKFLOW SCHEDULER

In the present landscape, each resource manager has its own unique way of handling task submissions. For example, a task's definition significantly differs between SLURM and Kubernetes. While SLURM supports task dependencies, Kubernetes lacks this feature. To address the challenge that resource managers schedule workflow tasks without workflow awareness, we developed the Common Workflow Scheduler (CWS) [15]. The CWS allows for the transfer of essential information, such as input files, CPU, and memory requests, along with task-specific parameters using the Common Workflow Scheduler Interface (CWSI). Task-specific parameters vary for each task invocation and are passed on to the utilized tools. For further details on the interface, we refer to our previous paper [15].

In Figure 1, we provide an architectural overview for a single resource manager, in this case, for Kubernetes. The CWS runs as a component in the resource manager and exposes the CWSI. A resource manager has to implement the CWS with its interface once. Conversely, a workflow engine needs to implement support for CWSI to work with all resource managers already offering CWSI. SWMSs such as Airflow, Nextflow, or Argo send their requests, which are then kept in memory of CWS. From this storage, the CWS can fetch the workflow graph and task dependencies and use this information for scheduling. This storage can further be used for provenance to trace the workflow execution; we elaborate on this in Section 4. The CWS can be extended with task runtime and resource predictors that read task information from the storage and learn characteristics. Such learned characteristics can then be used to predict the demands for upcoming tasks, which is helpful for better scheduling. We provide examples for such prediction strategies in Section 5. Notably, workflow engines with CWSI support do not need their own scheduler component. Instead, all ready-to-run tasks are submitted to the resource manager and the scheduling happens there.

We have implemented a plugin[1] for the SWMS Nextflow to communicate with the CWSI and the CWS for the resource manager Kubernetes[2]. Figure 2 shows the results from running nf-core workflows with the original Nextflow-Kubernetes interaction (Original strategy) and the Rank (Min) Round Robin scheduling algorithm. nf-core is a collection of best-practice Nextflow workflows which all come with small test sets. The Rank (Min) Round Robin, on average, outperformed other strategies tested with a median runtime improvement of up to 24.8% and an average reduction of 10.8% compared to the original strategy [15].

3 SWMS SUPPORT

We started by implementing the CWSI for a resource manager, Kubernetes, and a SWMS, Nextflow. We are now actively working to extend our project to support other popular SWMSs, namely Airflow and Argo, to further explore and demonstrate the benefits of the CWSI. Below, we describe these three SWMSs and discuss the integration of the CWSI.

Nextflow is a workflow engine initially designed for bioinformatics but getting uptake also in different domains and is used by more than 1,000 organizations [12, 16, 24]. One of the main advantages of Nextflow is its support for, at the time of writing, 20 different resource managers. The large support makes it easy to port Nextflow workflows between environments. The support is achieved by abstracting the resource manager from the scientist but also from the internal Nextflow logic. Accordingly, Nextflow only supports the

[1]https://github.com/CommonWorkflowScheduler/nf-cws
[2]https://github.com/CommonWorkflowScheduler/KubernetesScheduler

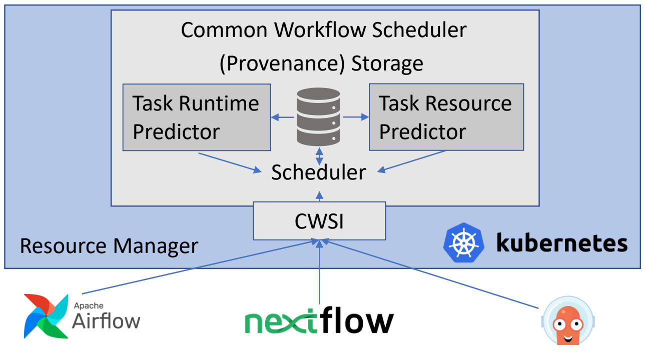

Figure 1: Architecture overview: The Common Workflow Scheduler with its interface and task runtime and task resource predictor component for Kubernetes as an exemplary resource manager.

basic features of resource managers. For example, on SLURM, the task dependency feature is not used. Thus, Nextflow can profit from providing additional workflow context to the resource manager.

Airflow is an Apache Incubator project designed for workflow management. Similar to Nextflow, Airflow supports Kubernetes as a resource manager and is also not exclusively tied to it. Airflow supports workflow-aware scheduling for Kubernetes through a tailor-made strategy exclusively implemented for the Airflow-Kubernetes interplay. Therefore, Airflow starts a big worker on every node for the whole workflow execution and assigns tasks into these worker pods bypassing Kubernetes' task assignment logic. However, this strategy has a significant drawback: the big containers will request resources for the entire workflow execution time regardless of the actual load. As many workflows have a merge point somewhere, where the entire execution is waiting for one particular task, this strategy leads to substantial resource wastage. By integrating the CWSI into Airflow, we aim to retain its workflow-aware scheduling capabilities while preventing unnecessary resource requests throughout the runtime. This optimization ensures more efficient utilization of resources and minimizes wastage on a large scale. One big difference to our already existing Nextflow interaction is the knowledge of the physical DAG in Airflow. While this was foreseen in the development of the CWSI, we have to make use of it in our CWS implementation.

Argo is a SWMS designed exclusively for Kubernetes. However, since Kubernetes lacks support for task dependencies, Argo also submits each task individually, and Kubernetes then schedules them in a FIFO manner. This is comparable to the strategy of Nextflow and, thus, makes Argo an ideal candidate to support our CWSI. Just like Nextflow, Argo is expected to benefit in a similar way. We are currently working on developing an Argo extension to achieve this.

4 PROVENANCE WITH THE CWSI

Workflow provenance is one aspect that needs to be addressed in all SWMS [1, 4, 11]. Since the CWSI takes a central role in workflow execution, possessing comprehensive knowledge of the resource

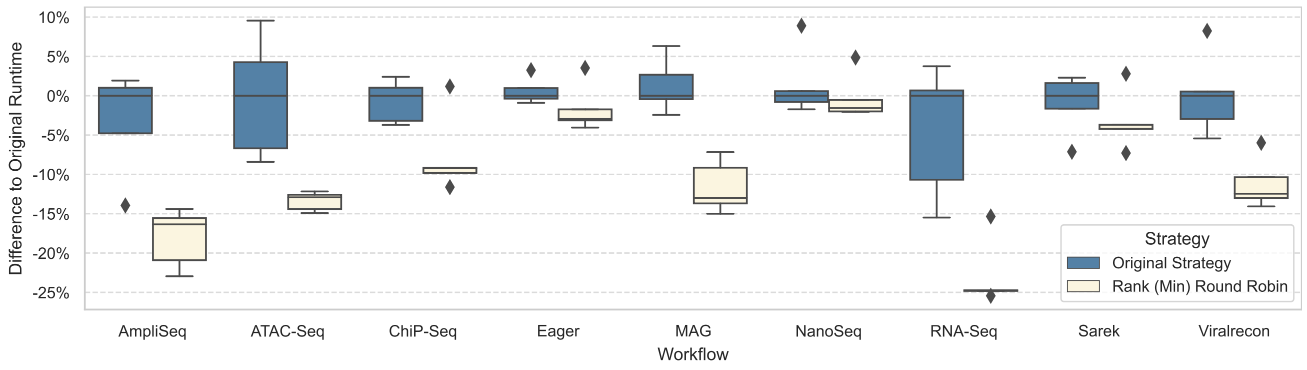

Figure 2: The runtime difference between the original runs' median and the Rank (Min) Round Robin scheduling for the nine most popular nf-core workflows.

manager and the SWMS, it emerges as the most suitable entity for the management of provenance data.

All SWMS represent provenance differently, so it is very heterogeneous [8]. Further, resource managers and SWMSs are only designed to gather a portion of the available data, each focusing on collecting data in its own scope [4]. Accordingly, the resource manager traces the node states while the SWMS collects task-related metrics. The CWSI is particularly implemented for each resource manager and can support a resource manager's specific APIs to collect traces while it has knowledge about the workflow. By gathering and storing all metrics and task dependencies in a centralized manner, provenance becomes more streamlined and manageable.

Another significant advantage of using the CWSI for provenance is that the data will be available across different SWMS, even if a particular SWMS does not yet provide built-in provenance data. This interoperability ensures that provenance information can be maintained consistently and comprehensively, enhancing workflow traceability and reproducibility. In turn, researchers and scientists can have greater confidence in the reliability and trustworthiness of their results.

5 ADVANCED RESOURCE MANAGEMENT WITH THE CWSI

As we saw in the previous section, the CWS provides information about task executions and performance metrics. Using this information allows possible interface extensions to derive task characteristics from it. Task characteristics can be predicted runtime, CPU or memory usage, which can be used for scheduling and fed back to the SWMS. Many scheduling strategies, such as HEFT [25], require knowledge of this.

In the following, we will show how the CWSI can be used to implement approaches for task resource prediction, task runtime prediction, and scheduling with real workflow systems.

Task resource prediction: Predicting the resources a task instance will utilize enables workflow performance optimization by reducing resource wastage and increasing performance [26]. Several research approaches tackle this challenge by analytic methods, regression models, or reinforcement learning and achieve a significant reduction in resource wastage [5, 20, 26–28]. A key challenge is to avoid underprovisioning of resources, as this leads to task failures while overprovisioning leads to high resource wastage [26]. These approaches frequently assume a relationship between input data size and a task's resource usage to predict peak memory consumption, i.e., a task's memory usage increases with bigger inputs. Further, many of these approaches conduct a form of online learning, incorporating monitoring data from task executions as feedback.

The CWSI provides information to train such models, e.g., the number of file inputs, input sizes, or peak memory, which are retrieved and stored from monitoring. As these metrics are constantly gathered and updated, also online learning approaches are applicable. Therefore, we plan to integrate existing task resource prediction methods in our CWSI prototype to a) increase workflow performance and b) evaluate them under real-world conditions.

Task Runtime Prediction: Predicting task runtimes is essential as many resource management techniques, such as scheduling, rely on accurate runtime estimates beforehand. To this end, many existing research approaches rely on historical data to build prediction models [14, 18, 19]. Many of them build on machine-learning models like neural networks, clustering methods, or regression methods [9, 10, 14, 18, 19]. While, especially complex models, showed to achieve low prediction errors, they also require a lot of training data. As an alternative, we recently presented Lotaru [2], an online approach that can cope with cold-start problems and is able to predict task runtimes without historical traces. To do this, Lotaru executes microbenchmarks and quickly runs the workflow with reduced input data locally. Next, it predicts a task's execution time using a Bayesian linear regression based on the data points collected from the local workflow profiling and the microbenchmarks.

Since Lotaru and other research approaches that support heterogeneous infrastructures require machine characteristics, we are extending our CWSI to store such information and extend the prototype to gather these metrics with Kubestone[3]. We are currently incorporating Lotaru into the CWSI prototype to handle unknown workflows or workflows with a lack of historical data. Further, we plan to implement other research methods that perform better with more training data provided by the provenance store of CWSI.

[3]kubestone.io

Workflow Task Scheduling: Applying sophisticated scheduling algorithms helps to achieve optimization objectives such as a makespan reduction, cost reduction, or energy reduction. Although extensive research in this field exists, many approaches are missing uptake in real-world scenarios. For instance, Yarn schedules tasks in a fair manner [23], while Kubernetes applies a Round-robin-like strategy [7]. Due to the dynamic nature of workflows and infrastructures, in practice, only dynamic scheduling approaches should be considered, i.e., approaches that can adjust their execution plans or react to failures in the infrastructure. Some of these dynamic approaches [6, 30] are based on the static heuristic HEFT [25] and require knowledge about task runtimes and communication times between the nodes. Our own prior work Tarema [3], does not require such metrics but dynamically classifies incoming tasks according to their resource usage to select the best-fitting node.

The CWSI, together with task runtime and resource prediction, provides additional information to apply more sophisticated scheduling techniques. We are currently implementing the Tarema strategy into our CWSI prototype and plan other more sophisticated approaches enabled through the additional data provided by the CWSI and their plugins.

6 CONCLUSION

In this paper, we presented the status quo of the Common Workflow Scheduler Interface and described the available plugin for Nextflow and the integration into Kubernetes. Further, we have demonstrated that by implementing the CWSI alongside basic scheduling approaches like rank and file size, we achieve an average runtime reduction of 10.8%. Next, we outlined upcoming support for the workflow engines Airflow and Argo and how to extend the storage to become the central place for workflow provenance. Additionally, we presented our next steps to implement resource allocation, runtime prediction, and new scheduling methods. We assume that the planned workflow algorithms that consider cluster heterogeneity and task runtime, as we outlined in this paper, will further improve resource efficiency.

ACKNOWLEDGMENTS

This work was funded by the German Research Foundation (DFG), CRC 1404: "FONDA: Foundations of Workflows for Large-Scale Scientific Data Analysis."

REFERENCES

[1] Khairul Alam and Banani Roy. 2022. Challenges of Provenance in Scientific Workflow Management Systems. In *2022 IEEE/ACM Workshop on Workflows in Support of Large-Scale Science (WORKS)*. IEEE.

[2] Jonathan Bader, Fabian Lehmann, Lauritz Thamsen, Ulf Leser, and Odej Kao. 2024. Lotaru: Locally predicting workflow task runtimes for resource management on heterogeneous infrastructures. *Future Generation Computer Systems* 150 (2024).

[3] Jonathan Bader, Lauritz Thamsen, Svetlana Kulagina, Jonathan Will, Henning Meyerhenke, and Odej Kao. 2021. Tarema: Adaptive Resource Allocation for Scalable Scientific Workflows in Heterogeneous Clusters. In *2021 IEEE International Conference on Big Data (Big Data)*.

[4] Jonathan Bader, Joel Witzke, Soeren Becker, Ansgar Lößer, Fabian Lehmann, Leon Doehler, Anh Duc Vu, and Odej Kao. 2022. Towards Advanced Monitoring for Scientific Workflows. In *2022 IEEE International Conference on Big Data (Big Data)*.

[5] Jonathan Bader, Nicolas Zunker, Soeren Becker, and Odej Kao. 2022. Leveraging Reinforcement Learning for Task Resource Allocation in Scientific Workflows. In *2022 IEEE International Conference on Big Data (Big Data)*.

[6] Jorge G Barbosa and Belmiro Moreira. 2011. Dynamic scheduling of a batch of parallel task jobs on heterogeneous clusters. *Parallel computing* 37, 8 (2011).

[7] Carmen Carrión. 2022. Kubernetes Scheduling: Taxonomy, ongoing issues and challenges. *ACM Computing Surveys (CSUR)* (2022).

[8] Sérgio Manuel Serra da Cruz, Maria Luiza M. Campos, and Marta Mattoso. 2009. Towards a Taxonomy of Provenance in Scientific Workflow Management Systems. In *2009 Congress on Services - I*.

[9] Rafael Ferreira da Silva, Gideon Juve, Ewa Deelman, Tristan Glatard, Frédéric Desprez, Douglas Thain, Benjamin Tovar, and Miron Livny. 2013. Toward Fine-Grained Online Task Characteristics Estimation in Scientific Workflows. In *Proceedings of the 8th Workshop on Workflows in Support of Large-Scale Science (WORKS '13)*. Association for Computing Machinery, New York, NY, USA.

[10] Rafael Ferreira Da Silva, Gideon Juve, Mats Rynge, Ewa Deelman, and Miron Livny. 2015. Online task resource consumption prediction for scientific workflows. *Parallel Processing Letters* 25, 03 (2015).

[11] Susan B. Davidson and Juliana Freire. 2008. Provenance and Scientific Workflows: Challenges and Opportunities. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data (SIGMOD '08)*. Association for Computing Machinery, New York, NY, USA.

[12] Paolo Di Tommaso, Maria Chatzou, Evan W Floden, Pablo Prieto Barja, Emilio Palumbo, and Cedric Notredame. 2017. Nextflow enables reproducible computational workflows. *Nature Biotechnology* 35, 4 (2017).

[13] Maxime Garcia, Szilveszter Juhos, Malin Larsson, Pall I. Olason, Marcel Martin, Jesper Eisfeldt, Sebastian DiLorenzo, Johanna Sandgren, Teresita Díaz De Ståhl, Philip Ewels, Valtteri Wirta, Monica Nistér, Max Käller, and Björn Nystedt. 2020. Sarek: A Portable Workflow for Whole-Genome Sequencing Analysis of Germline and Somatic Variants. *F1000Research* 9 (2020).

[14] Muhammad Hafizhuddin Hilman, Maria Alejandra Rodriguez, and Rajkumar Buyya. 2018. Task runtime prediction in scientific workflows using an online incremental learning approach. In *2018 IEEE/ACM 11th International Conference on Utility and Cloud Computing (UCC)*. IEEE.

[15] Fabian Lehmann, Jonathan Bader, Friedrich Tschirpke, Lauritz Thamsen, and Ulf Leser. 2023. How Workflow Engines Should Talk to Resource Managers: A Proposal for a Common Workflow Scheduling Interface. In *2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*. Bangalore, India.

[16] Fabian Lehmann, David Frantz, Sören Becker, Ulf Leser, and Patrick Hostert. 2021. FORCE on Nextflow: Scalable Analysis of Earth Observation Data on Commodity Clusters. In *Proceedings of the CIKM 2021 Workshops (CEUR Workshop Proceedings, Vol. 3052)*, Gao Cong and Maya Ramanath (Eds.).

[17] Paul Muir, Shantao Li, Shaoke Lou, Daifeng Wang, Daniel J. Spakowicz, Leonidas Salichos, Jing Zhang, George M. Weinstock, Farren Isaacs, Joel Rozowsky, and Mark Gerstein. 2016. The Real Cost of Sequencing: Scaling Computation to Keep Pace with Data Generation. *Genome Biology* 17, 1 (2016).

[18] Farrukh Nadeem, Daniyal Alghazzawi, Abdulfattah Mashat, Khalid Fakeeh, Abdullah Almalaise, and Hani Hagras. 2017. Modeling and predicting execution time of scientific workflows in the grid using radial basis function neural network. *Cluster Computing* 20, 3 (2017).

[19] Thanh-Phuong Pham, Juan J Durillo, and Thomas Fahringer. 2017. Predicting workflow task execution time in the cloud using a two-stage machine learning approach. *IEEE Transactions on Cloud Computing* 8, 1 (2017).

[20] Thanh Son Phung, Logan Ward, Kyle Chard, and Douglas Thain. 2021. Not All Tasks Are Created Equal: Adaptive Resource Allocation for Heterogeneous Tasks in Dynamic Workflows. In *2021 IEEE Workshop on Workflows in Support of Large-Scale Science (WORKS)*. IEEE.

[21] Matthias Schramm, Edzer Pebesma, Milutin Milenković, Luca Foresta, Jeroen Dries, Alexander Jacob, Wolfgang Wagner, Matthias Mohr, Markus Neteler, Miha Kadunc, Tomasz Miksa, Pieter Kempeneers, Jan Verbesselt, Bernhard Gößwein, Claudio Navacchi, Stefaan Lippens, and Johannes Reiche. 2021. The openEO API–Harmonising the Use of Earth Observation Cloud Services Using Virtual Data Cube Functionalities. *Remote Sensing* 13, 6 (2021).

[22] Martin Sudmanns, Dirk Tiede, Hannah Augustin, and Stefan Lang. 2019. Assessing global Sentinel-2 coverage dynamics and data availability for operational Earth observation (EO) applications using the EO-Compass. *International journal of digital earth* 13, 7 (2019).

[23] Shanjiang Tang, Bu-Sung Lee, and Bingsheng He. 2016. Fair resource allocation for data-intensive computing in the cloud. *IEEE Transactions on Services Computing* 11, 1 (2016).

[24] Paolo Di Tommaso. 2022. *A quick overview of Nextflow workflow system.* https://workflows.community/stories/2022/09/28/nextflow/ accessed 18-October-2022.

[25] Haluk Topcuoglu, Salim Hariri, and Min-you Wu. 2002. Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE transactions on parallel and distributed systems* 13, 3 (2002).

[26] Benjamin Tovar, Rafael Ferreira da Silva, Gideon Juve, Ewa Deelman, William Allcock, Douglas Thain, and Miron Livny. 2017. A job sizing strategy for high-throughput scientific workflows. *IEEE Transactions on Parallel and Distributed Systems* 29, 2 (2017).

[27] Ben Tovar, Ben Lyons, Kelci Mohrman, Barry Sly-Delgado, Kevin Lannon, and Douglas Thain. 2022. Dynamic Task Shaping for High Throughput Data Analysis Applications in High Energy Physics. In *2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE.

[28] Carl Witt, Dennis Wagner, and Ulf Leser. 2019. Feedback-Based Resource Allocation for Batch Scheduling of Scientific Workflows. In *2019 HPCS*. IEEE.

[29] James A Fellows Yates, Thiseas C Lamnidis, Maxime Borry, Aida Andrades Valtue na, Zandra Fagernäs, Stephen Clayton, Maxime U Garcia, Judith Neukamm, and Alexander Peltzer. 2021. Reproducible, portable, and efficient ancient genome reconstruction with nf-core/eager. *PeerJ* 9 (2021).

[30] Zhifeng Yu and Weisong Shi. 2007. An Adaptive Rescheduling Strategy for Grid Workflow Applications. In *2007 IEEE International Parallel and Distributed Processing Symposium*. IEEE, Long Beach, CA, USA.