Tobias Wies, Sami Kharma, Tobias Meuser, Björn Scheuermann *Accurate Synthetic Tasks for Scientific Workflow Benchmarks*. Proceedings of the 26th IEEE International Symposium on Cluster, Cloud, and Internet Computing (CCGrid26), 2026.

Accurate Synthetic Tasks for Scientific Workflow Benchmarks

Tobias Wies*, Sami Kharma†, Tobias Meuser*, Björn Scheuermann*
*Communication Networks Lab, Technical University of Darmstadt, Germany
†Distributed Algorithms, Zuse Institute Berlin, Germany
Email: {*tobias.wies, tobias.meuser, bjoern.scheuermann*}@kom.tu-darmstadt.de, kharma@zib.de

***Abstract*—Scientific workflows are fundamental for automating complex data processing tasks. Research on workflow scheduling and execution strategies often relies on real-world workflows for testing and validation. The setup of such workflows is time-consuming and error-prone due to their dependence on large datasets, specialized software, or specific hardware requirements. Synthetic workflows have been proposed as a solution, offering realistic benchmarking scenarios. Existing workload generators, however, primarily model tasks based on aggregated resource consumption metrics, such as total CPU time or peak memory usage, which do not capture the dynamic resource usage patterns exhibited by real-world tasks. This limits the expressiveness of experiments performed on synthetic workflows. In this work, we introduce a novel approach for generating synthetic workflow tasks that capture dynamic resource usage patterns. In addition, we derive realistic task models by monitoring real workflow task executions and applying segmented linear regression to the measurement data. These models serve as the basis for generating synthetic tasks. Our method allows researchers to create synthetic workflows that more accurately reflect the behavior of real-world scientific workflows, while still being efficient and without requiring external datasets.**

***Index Terms*—scientific workflows, synthetic workflow tasks, workflow benchmarking, resource usage modeling**

I. Introduction

Scientific workflows play a central role in large-scale scientific computing, enabling researchers to automate complex data processing steps and supporting the reproducibility of experiments. They are widely used for data analysis across domains such as climate modeling, and bioinformatics.

Workflow engines are essential for executing workflows and provide the foundation for optimizing for metrics such as reduction in makespan or monetary cost. Researchers already proposed such optimizations for memory consumption of tasks [1] or data transfer between tasks [2]–[4], for example.

To test and validate new scheduling or execution strategies, researchers often rely on existing real-world workflows and datasets or non-standard benchmarks [5]. This approach comes with several limitations. Real workflows often require access to large domain-specific datasets, specialized software, or substantial computational resources. This can make it difficult to use them for testing and benchmarking purposes. Additionally, using only some real workflows may not be representative of the wide variety of scenarios that researchers wish to explore, limiting the generalizability of findings.

In this work, we address this gap by proposing a novel approach for generating synthetic tasks that accurately replicate the resource usage patterns observed in real-world tasks. These patterns are derived from monitoring real workflow executions and fitting segmented linear models to the collected measurement data. Our method models temporal resource consumption – such as CPU time, memory usage, and I/O operations – using piecewise linear function and allows downscaling them to run on less performant hardware. The resulting resource consumption models can be easily shared with other researchers, allowing them to validate experiment results. Our high-resolution resource models enable validating advanced scheduling strategies that exploit dynamic resource usage, such as co-scheduling tasks with complementary memory demands.

II. Related Work

In [6], the authors propose an automated generator for executable scientific workflow benchmarks. Through comprehensive datasets such as [7], researchers can create a wide variety of representative workflow benchmarks using execution traces of real-world workflow executions. This approach is suitable for identifying performance bottlenecks or comparing HPC platforms. However, the underlying task models are rather simple and therefore not suited for verifying or validating advanced execution strategies, such as file streaming [4] or memory prediction methods [8]. These strategies exploit varying resource consumption patterns over the execution of the task.

The workload generator fio [9] focuses on generating synthetic workloads for I/O benchmarking. It allows users to specify realistic I/O patterns that are then executed. While it also supports defining static CPU utilization, it is not designed to model workloads with coupled resource usage.

III. Approach

In this section, we describe our approach for generating synthetic tasks from real-world workflow executions. We first discuss how we extract resource usage patterns from real tasks, then how we generate models for synthetic tasks based on this data. To increase the flexibility of synthetic tasks, we introduce a method for downscaling resource consumption and runtime. Finally, we describe the implementation of our approach.

This work received funding from the German Research Foundation (DFG), CRC 1404: *FONDA: Foundations of Workflows for Large-Scale Scientific Data Analysis*. The authors thank Florian Jochum for his help with the implementation of the synthetic task.

The documents distributed by this server have been provided by the contributing authors as a means to ensure timely dissemination of scholarly and technical work on a non-commercial basis. Copyright and all rights therein are maintained by the authors or by other copyright holders, not withstanding that they have offered their works here electronically. It is understood that all persons copying this information will adhere to the terms and constraints invoked by each author's copyright. These works may not be reposted without the explicit permission of the copyright holder.

Tobias Wies, Sami Kharma, Tobias Meuser, Björn Scheuermann *Accurate Synthetic Tasks for Scientific Workflow Benchmarks*. Proceedings of the 26th IEEE International Symposium on Cluster, Cloud, and Internet Computing (CCGrid26), 2026.

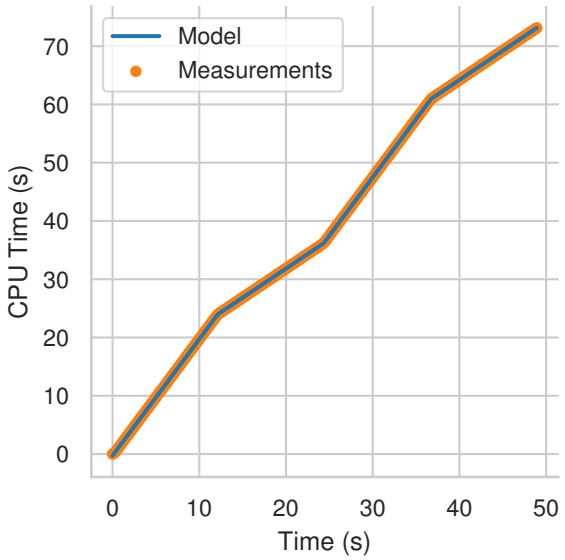

Fig. 1: Measured CPU time pattern of the task `STAR_ALIGN` from the RNA-seq workflow [12] using the provided test dataset and the corresponding piecewise linear model fit using segmented regression [13] with $k = 4$.

A. Extracting Resource Usage Patterns

Previous work on synthetic workflows primarily focused on modeling aggregated resource consumption metrics, such as CPU time, peak memory usage, and total I/O operations [6]. These metrics can often be directly obtained from workflow management systems through their logging and monitoring capabilities. Nextflow [10] is an example of such a workflow management system that provides task-level resource usage statistics, however, it only provides aggregate metrics for the entire task execution.

Without modifying either the workflow definition or the workflow engine, it is not possible to observe temporal task behavior directly during workflow execution. Nevertheless, Nextflow executes each task within its own, task-specific working directory that contains both the input data and the execution script used in the original workflow run. This enables the re-execution and monitoring of selected tasks with high temporal resolution.

To this end, we developed a lightweight monitoring tool that captures CPU time and memory usage over the duration of task execution and stores the recorded data for further processing.

This method monitors CPU time, memory usage and total I/O over time using the Linux `procfs` interface. It can be extended in the future to include additional resources, such as file-specific disk I/O as investigated in past research [11]. This is beneficial to support per-file resource consumption models.

B. Generating Models for Synthetic Tasks

Replaying resource usage measurements directly from measurement data is, depending on the sampling rate, computationally expensive and can produce a significant overhead during synthetic task execution. Thus, the simulation overhead needs to be considered when modeling resource consumption. Generating a simplified model per resource that approximates the resource usage over time solves this issue. During our experiments, we observed that real-world tasks, especially those with longer runtimes, often exhibit resource usage patterns that can be approximated using piecewise linear functions.

Let $r_i : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ denote the resource consumption of resource i over time. We define a resource consumption model as either cumulative, which aggregates over time (e.g., CPU time), or instantaneous, which describes the resource state at a given point in time (e.g., memory usage).

We model $r_i(t)$ as a piecewise linear function with k segments, i.e.,

$$r_i(t) = \begin{cases} a_{i,1}t + b_{i,1}, & t \in [t_0, t_1), \\ a_{i,2}t + b_{i,2}, & t \in [t_1, t_2), \\ \vdots & \\ a_{i,k}t + b_{i,k}, & t \in [t_{k-1}, t_k], \end{cases}$$

where $0 = t_0 < t_1 < \cdots < t_k$ define the segment boundaries.

To generate these models from our resource usage measurements, we employ a segmented linear regression approach [13]. Figure 1 shows an example of a measured CPU usage pattern and the corresponding linear segmented model fit.

C. Downsizing Resource Consumption

An advantage of using synthetic tasks with resource consumption models is the ability to downscale resource consumption or the runtime of tasks. This enables researchers to test and validate their approaches faster and on less performant hardware. By adjusting the parameters of the resource consumption models, synthetic tasks can be created that require fewer resources and have shorter runtimes, while still preserving the overall behavior and key characteristics of the original task.

To downscale the resource consumption of a task, we introduce a resource-specific scaling factor $s_i \in (0, 1]$ for each resource i. This scaling factor s_i is applied multiplicatively to the original resource consumption model $r_i(t)$, resulting in a downscaled resource consumption model $\hat{r}_i(t)$ defined as:

$$\hat{r}_i(t) = s_i \, r_i(t)$$

By assigning individual scaling factors to each resource, the task can be adapted to the capabilities of the evaluation hardware while preserving the relative resource usage characteristics of the original execution. For example, memory consumption can be reduced to fit within available limits while maintaining the original CPU utilization. Executing the synthetic task with the downscaled resource consumption model $\hat{r}_i(t)$ thus preserves the temporal behavior of the original task while proportionally reducing its resource consumption.

In addition to scaling individual resource demands, it is often desirable to reduce the overall runtime of a task. Downscaling the resource consumption model does not alter the runtime of the task. However, reducing the runtime is particularly relevant for testing or validation phases, for example

The documents distributed by this server have been provided by the contributing authors as a means to ensure timely dissemination of scholarly and technical work on a non-commercial basis. Copyright and all rights therein are maintained by the authors or by other copyright holders, not withstanding that they have offered their works here electronically. It is understood that all persons copying this information will adhere to the terms and constraints invoked by each author's copyright. These works may not be reposted without the explicit permission of the copyright holder.

Tobias Wies, Sami Kharma, Tobias Meuser, Björn Scheuermann *Accurate Synthetic Tasks for Scientific Workflow Benchmarks*. Proceedings of the 26th IEEE International Symposium on Cluster, Cloud, and Internet Computing (CCGrid26), 2026.

during the iterative development of scheduling algorithms, where long-running tasks can significantly slow down experimentation. To address this, we introduce a global temporal scaling factor $\tau \in (0, 1]$ to reduce the runtime of the task. The downscaled resource consumption model $\tilde{r}_i(t)$, incorporating both resource-specific and temporal scaling, is then defined as:

$$\tilde{r}_i(t) = s_i \, c_\tau \, r_i \left(\frac{t}{\tau} \right), \quad c_\tau = \begin{cases} 1, & \text{if } r_i \text{ is instantaneous,} \\ \tau, & \text{if } r_i \text{ is cumulative.} \end{cases}$$

The correction factor c_τ is introduced to mitigate unintended side effects of the global temporal scaling factor while keeping the resource-specific scaling factor s_i fully effective. For cumulative resource consumption models, such as aggregated I/O operations, τ without a correction factor c_τ would naturally increase the slope of the resource consumption curve, leading to e.g. higher effective data rates. Depending on the evaluation hardware, this could introduce execution bottlenecks. For instantaneous resource consumption models, scaling the time dimension has no unintended side effects and therefore requires no correction.

The collection of all $\tilde{r}_i$ models can be directly translated into a configuration for a synthetic task generator, enabling the creation of synthetic tasks with the desired resource consumption profiles and runtime characteristics.

D. Implementing Synthetic Tasks

A concrete synthetic task includes the resource consumption models for all resources and the respective scaling factors. During execution, the synthetic task continuously monitors its current resource usage and adjusts it to match the target resource usage defined in the model $\tilde{r}_i(t)$. This design compensates for side effects that other resource models introduce, e.g. I/O operations cause CPU load. Algorithm 1 shows the pseudocode of the closed-loop synthetic resource consumption algorithm. The algorithm runs until the defined end time t_{end} is reached. In each iteration, the current resource usage for all resources is updated and compared to the target resource usage defined in the model. If the current usage is lower than the target, the resource usage is increased accordingly. If the current usage is higher than the target, the resource usage is decreased. A decrease only happens if the resource model is instantaneous. After adjusting the resource usage for all resources, the algorithm sleeps for a defined interval Δt before proceeding to the next iteration.

While this is comparatively simple for memory usage and I/O operations, adjusting CPU time is more complex. To this end, we developed a dynamic load generator that adjust the number of iterations of our compute task to match the actual required CPU time delta. The CPU load generator queries the `procfs` more often than the other load generators. This problem could also be solved by running a short calibration run before the synthetic task execution.

This design allows for flexible and accurate synthetic tasks that can mimic the resource usage patterns of real-world tasks while self-regulating their resource consumption based on the defined models. The closed-loop control mechanism ensures that each resource stays within the desired usage bounds, even with potential side effects from other resources e.g., increased memory usage due to high I/O operations.

Algorithm 1 Closed-loop synthetic resource consumption

```
Initialize time t ← 0
while t < t_end do
    Update usage vector U for all resources
    for resource i ∈ R do
        current_i ← U_i
        target_i ← r̃_i(t)
        δ_i ← target_i − current_i
        if δ_i > 0 then
            Increase resource usage of r_i by δ_i
        else if r_i is instantaneous then
            Decrease resource usage of r_i by |δ_i|
    Sleep for Δt
    t ← t + Δt
```

IV. Evaluation

To evaluate our approach, we conducted a proof-of-concept implementation of the synthetic task with an artificial resource consumption model that models CPU time, memory usage, and I/O operations. We decided to use a simple, artificial model for this evaluation to demonstrate the feasibility of our approach. Our artificial task is designed to model a standard behavior of a simple processing task and consists of two distinct phases. In the first phase, it reads a file and performs light computation. In the second phase, which is designed as the processing phase, it uses more computational resources and memory while writing the output file. The evaluation was conducted on a single machine with an Intel Core i7-13700H CPU, 32 GB of RAM, and NVMe storage. The experiment was repeated 10 times and the different results were compared. While the errors slightly differ between runs, the overall behavior is consistent. For this reason, we present the results of one representative run in the following.

Figure 2 shows the results of one evaluation run. The left column shows the target resource usage defined in the model and the actual resource usage during the execution of the synthetic task. The right column shows the relative error between the target and actual resource usage. The results demonstrate that our approach can effectively follow the resource usage patterns of real-world tasks, with a low deviation between the target and actual resource usage across all resources. However, our approach can get inaccurate when resource consumption changes rapidly in the model, which is especially visible in the memory usage plot. The peak memory error varied between 2% and 3.5% while the other errors stayed almost constant throughout multiple runs. Because we decided to allocate memory in small chunks to allow for minor adjustments during execution, allocating a large amount of memory at once can take more time, leading to a higher error. This can be mitigated by using a more dynamic memory allocation strategy that employs variable-sized chunks based on the memory model.

The documents distributed by this server have been provided by the contributing authors as a means to ensure timely dissemination of scholarly and technical work on a non-commercial basis. Copyright and all rights therein are maintained by the authors or by other copyright holders, not withstanding that they have offered their works here electronically. It is understood that all persons copying this information will adhere to the terms and constraints invoked by each author's copyright. These works may not be reposted without the explicit permission of the copyright holder.

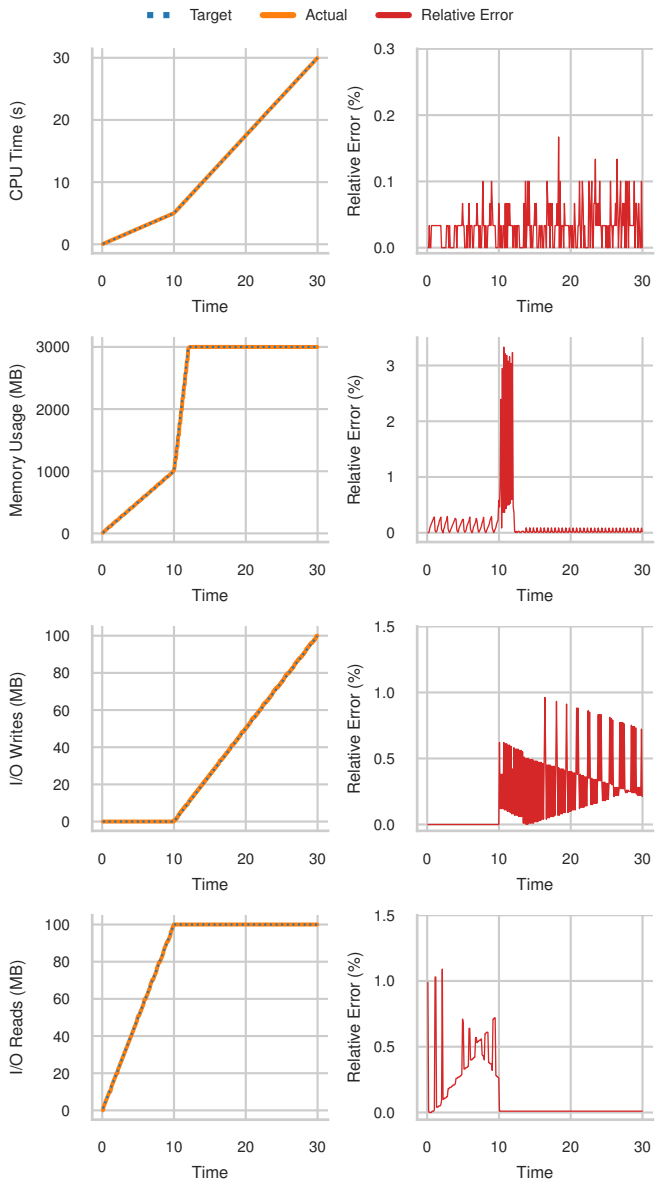

Fig. 2: Results of the evaluation run for a synthetic task with defined models for CPU time, memory usage and I/O operations. The left column shows the target resource usage and the actual resource usage during the execution of the task. The right column shows the relative error between the target and actual resource usage.

V. Outlook and Conclusion

Our proof-of-concept implementation demonstrates the feasibility of generating synthetic tasks that capture dynamic resource usage patterns. The results show that our approach can effectively mimic the behavior of real-world tasks, providing a valuable tool for testing and benchmarking scheduling and execution strategies in scientific workflows. It is able to replicate real-world task behavior accurately while enabling the downscaling of resource usages.

However, there are some limitations to our current approach. Current workflow engines, such as Nextflow [10], do not provide the necessary monitoring capabilities to capture temporal resource usage patterns directly during workflow execution. This requires researchers to manually re-execute tasks to obtain the necessary data for model generation, which can be time-consuming and may not always be feasible. Additionally, the model generation process is not fully automated and requires the manual selection of the number of segments for the piecewise linear regression.

In future work, we plan to extend our approach to support additional resources, such as file-specific I/O operations, GPU usage or network usage. Additionally, we will test our synthetic tasks in a wider range of scenarios and with more complex resource usage patterns. Finally, we will integrate our synthetic task generator and the synthetic tasks itself into existing workflow benchmarking systems, such as WfBench [6] to facilitate its adoption by the research community.

References

[1] J. Bader, A. Lößer, L. Thamsen, B. Scheuermann, and O. Kao, "KS+: Predicting Workflow Task Memory Usage Over Time," in *2024 IEEE 20th International Conference on E-Science (e-Science)*, Sep. 2024, pp. 1–7.

[2] F. Lehmann, J. Bader, F. Tschirpke, N. De Mecquenem, A. Lößer, S. Becker, K. E. Lewińska, L. Thamsen, and U. Leser, "WOW: Workflow-Aware Data Movement and Task Scheduling for Dynamic Scientific Workflows," in *2025 IEEE 25th International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, 2025, pp. 525–538.

[3] M. E. Santimaria, I. Colonnelli, B. Cantalupo, M. Torquati, D. Medić, N. Tuccari, E. Sciacca, and M. Aldinucci, "Dynamic transparent streaming in file-based workflows with CAPIO," *Future Generation Computer Systems*, vol. 176, p. 108159, Mar. 2026.

[4] S. Lampa, M. Dahlö, J. Alvarsson, and O. Spjuth, "SciPipe: A workflow library for agile development of complex and dynamic bioinformatics pipelines," *GigaScience*, vol. 8, no. 5, May 2019.

[5] A. Vivas, A. Tchernykh, and H. Castro, "Trends, Approaches, and Gaps in Scientific Workflow Scheduling: A Systematic Review," *IEEE Access*, vol. 12, pp. 182 203–182 231, 2024.

[6] T. Coleman, H. Casanova, K. Maheshwari, L. Pottier, S. R. Wilkinson, J. Wozniak, F. Suter, M. Shankar, and R. F. Da Silva, "WfBench: Automated Generation of Scientific Workflow Benchmarks," in *2022 IEEE/ACM International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)*, Nov. 2022, pp. 100–111.

[7] R. F. da Silva, "wfcommons/wfinstances: v1.5," Jun. 2024. [Online]. Available: https://doi.org/10.5281/zenodo.12510982

[8] J. Bader, F. Skalski, F. Lehmann, D. Scheinert, J. Will, L. Thamsen, and O. Kao, "Sizey: Memory-Efficient Execution of Scientific Workflow Tasks," in *2024 IEEE International Conference on Cluster Computing (CLUSTER)*, 2024, pp. 370–381.

[9] J. Axboe, "Flexible i/o tester," 2022. [Online]. Available: https://github.com/axboe/fio

[10] P. Di 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, 2017.

[11] J. Witzke, A. Lößer, V. Bountris, F. Schintke, and B. Scheuermann. (2024) Low-level I/O Monitoring for Scientific Workflows. [Online]. Available: http://arxiv.org/abs/2408.00411

[12] H. Patel, "nf-core/rnaseq: nf-core/rnaseq v3.22.1 - polished palladium penguin," Dec. 2025. [Online]. Available: https://doi.org/10.5281/zenodo.17833254

[13] A. Lößer, M. Schlecht, F. Schintke, J. Witzke, M. Weidlich, and B. Scheuermann, "Fast Min-ε Segmented Regression Using Constant-Time Segment Merging," in *2025 Proceedings of the 42nd International Conference on Machine Learning*, 2025, pp. 40 312–40 327.

The documents distributed by this server have been provided by the contributing authors as a means to ensure timely dissemination of scholarly and technical work on a non-commercial basis. Copyright and all rights therein are maintained by the authors or by other copyright holders, not withstanding that they have offered their works here electronically. It is understood that all persons copying this information will adhere to the terms and constraints invoked by each author's copyright. These works may not be reposted without the explicit permission of the copyright holder.