Understanding Storage Trade-offs for Containerized Dataflows Across the Computing Continuum

Wesley Ferreira[1], Luiz Viana[1], Lucas Natan[1], Liliane Kunstmann[2] , Daniel de Oliveira[1]

[1]Instituto de Computação – Universidade Federal do Fluminense (UFF)

[2]Universidade Federal Rural do Rio de Janeiro (UFRRJ)

{wesleyferreira,luizmvb,lucasnatan}@id.uff.br

liliane.kunstmann@ufrrj.br,danielcmo@ic.uff.br

Abstract. *Dataflows model and execute workloads in scientific and data-intensive domains by representing applications as tasks linked by data dependencies. This structure supports scheduling, execution, and reproducibility. Many dataflows process large datasets in distributed environments such as HPC clusters and cloud platforms. However, tasks often have heterogeneous computational and data requirements, motivating the use of the computing continuum, where workloads migrate across environments. Containers enable portable execution across heterogeneous infrastructures, but storage strategies for data sharing remain insufficiently explored. This paper evaluates multiple storage approaches, analyzing how data size, granularity, and role influence performance and efficiency.*

1. Introduction

Over the past decade, the adoption of dataflows as a paradigm for modeling and executing computational workloads has increased at a fast pace [Suter et al. 2026]. Dataflows became a *de facto* standard across domains such as Bioinformatics [Wang et al. 2018] and Astronomy [Ponte Ahón et al. 2025]. By structuring computation as tasks with explicit data dependencies, dataflows provide an intuitive representation of computational processes. This abstraction eases human interpretation, simplifies scheduling and execution, and promotes reproducibility [Plale et al. 2021]. Dataflows can be executed using either simple scripts or specialized systems [de Oliveira et al. 2019], which offer features such as fault tolerance and resource optimization.

Many dataflows process large datasets and require parallel execution in distributed environments such as High Performance Computing (HPC) clusters or cloud platforms. However, tasks often vary in computational requirements, thus requiring the Computing Continuum [Al-Dulaimy et al. 2024], which spans local machines, cloud platforms, edge, and IoT devices. In this context, dataflows can dynamically adapt execution across environments depending on task requirements. Implementing such flexibility is challenging because orchestrating across heterogeneous infrastructures is complex. Existing systems often depend on specific environments. For example, Pegasus relies on HTCondor [Shah et al. 2014], which may not be available. Additionally, several software dependencies create complex ecosystems that hinder portability, particularly in HPC environments.

Containers offer a solution by packaging applications with their full software environment [Struhár et al. 2020]. They enable execution across heterogeneous platforms, allowing containerized dataflows to move across the computing continuum with minimal reconfiguration. Several systems, including Nextflow [Di Tommaso et al. 2017] and Snakemake [Köster and Rahmann 2012], support containerized dataflows. While these systems

provide flexible scheduling and execution capabilities, they largely overlook the challenge of selecting appropriate storage strategies. Containers require mechanisms to persist and share data during and after execution. Common strategies include shared persistent volumes, network-based storage (*e.g.*, object storage systems), and external repositories such as Dataverse [Crosas 2011]. Each strategy suits different data characteristics, such as size, format, and role in the dataflow. This paper evaluates these storage strategies across synthetic and real-world dataflows to identify suitable approaches for different scenarios. We analyze how factors such as data size and granularity affect performance in containerized environments. The results aim to inform scheduling decisions in dataflow systems, improving efficiency and resource usage across the computing continuum.

2. Related Work

Several studies have investigated data sharing and storage management strategies for containerized environments, independently of their deployment across the computing continuum. [Bachiega et al. 2020] analyze the performance of containers using either the Network File System (NFS) or Docker's native volume-sharing mechanisms. Their evaluation considers three types of applications: (i) database access workloads, (ii) a file-intensive script performing read and write operations, and (iii) an MPI-based parallel application. [Chazapis et al. 2021] propose the Unified Storage Layer (USL), a middleware that provides cloud-native applications with transparent access to heterogeneous storage backends, including distributed file systems, object stores, and key-value databases. Similarly, [Tarasov et al. 2019] evaluate Docker storage drivers and analyze their impact on both system-level and application-level performance.

In the context of cloud-native orchestration platforms, [Mercl and Pavlik 2019] evaluates the performance of Kubernetes clusters deployed over public cloud storage infrastructures, including Amazon Elastic Kubernetes Service, Google Kubernetes Engine, and Azure Kubernetes Service. Furthermore, [Zhao et al. 2021] analyzes several terabytes of Docker Hub images to characterize their storage and performance properties, with particular emphasis on the impact of image size and file-level deduplication on storage efficiency and execution performance. CNSBench [Merenstein et al. 2021] introduces a Kubernetes-based benchmark framework for evaluating cloud-native storage systems under I/O-intensive workloads. Likewise, KubeStorage [Liu et al. 2019] addresses container storage challenges in cloud environments by integrating Kubernetes with Haystack to efficiently manage large collections of small files, particularly for read-dominated workloads.

To support the storage and publication of data artifacts enriched with metadata and semantic annotations, Dataverse [Crosas 2011] plays an important role in scientific dataflow ecosystems [Woyames et al. 2025]. Dataverse is an open-source research data management platform designed to support the publication, sharing, and preservation of datasets and associated metadata. Although these approaches represent a step forward in container storage and cloud-native data management, they focus on generic cloud workloads and infrastructure-level performance evaluation. On the other hand, our paper focuses on containerized dataflows across the computing continuum, investigating how different storage strategies affect dataflow execution efficiency.

3. Experimental Design

This section describes the experimental design adopted to evaluate storage strategies for executing containerized dataflows across the computing continuum and sharing produced files.

Our goal is to understand how different storage approaches behave under varying storage strategies and execution environments, particularly with respect to factors such as data size, granularity, and the role of data within dataflows. The overall process, presented in Figure 1a, comprises four activities: (i) Data Preparation, (ii) Environment Setup, (iii) Dataflow Execution, and (iv) Quantitative Evaluation. In the following, we describe each step.

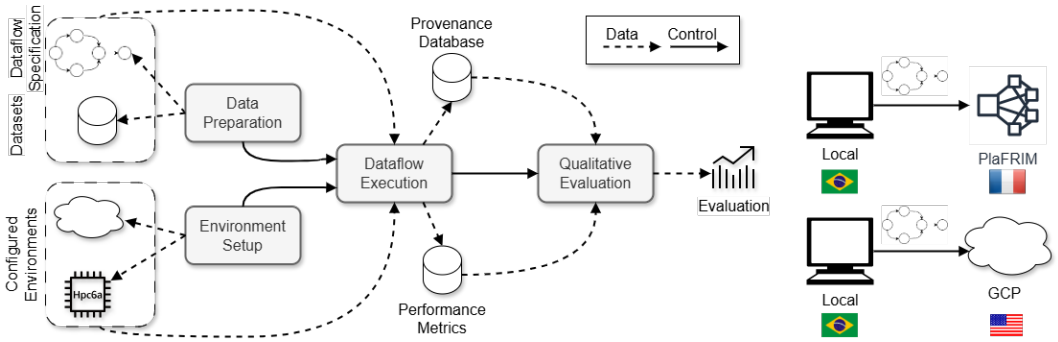

Figure 1. (a) The experimental design (b) The environment settings.

In the *Data Preparation* activity, we define the dataflows and datasets used in the evaluation. We consider two classes of dataflows designed to evaluate storage strategies under different computing scenarios: (i) two synthetic dataflows designed to stress storage systems through massive file manipulation, and (ii) a real dataflow represented by Montage [Sakellariou et al. 2009]. The synthetic dataflows (Figure 2a) generate a total of 100,000 files per execution, differing only in how the data is organized and transferred. The first synthetic dataflow manipulates 100,000 individual files ($\approx$ 40 bytes each) directly across the activities, while the second generates the same data volume but groups the files into five shards, each containing 20,000 compressed files. In addition to the synthetic dataflows, we evaluate the Montage dataflow ($\approx$ 6.55 GB), an astronomy application for generating image mosaics from multiple telescope observations. Montage is characterized by intensive data manipulation and the generation of large intermediate datasets, making it suitable for evaluating storage strategies. The dataflow is composed of seven activities: (i) `mProject`, which projects images into a common coordinate system; (ii) `mDiffFit`, which computes differences between overlapping images and adjusts background values; (iii) `mConcatFit`, which aggregates fitting results; (iv) `mBgModel`, responsible for background modeling; (v) `mBackground`, which applies background corrections; (vi) `mImgtbl`, used to extract metadata; and (vii) `mAdd`, which generates the final astronomical mosaic. Figure 2b presents the Montage.

In the *Environment Setup* activity, we define the execution environments used in the evaluation. The experiments were executed in two different infrastructures spanning the computing continuum. The first environment is the PlaFRIM cluster[1] (zonda01-21 compute node equipped with 2 $\times$ 32-core AMD Zen2 processors and 256 GB RAM), an HPC infrastructure operated by INRIA Bordeaux. The second environment is the Google Cloud Platform (GCP) (Kubernetes cluster composed of three e2-medium virtual machines). In the *Dataflow Execution* activity, we execute the dataflows across two representative computing continuum settings: (i) Local Workstation + PlaFRIM and (ii) Local Workstation + Cloud. For each,

[1] `https://www.plafrim.fr/`

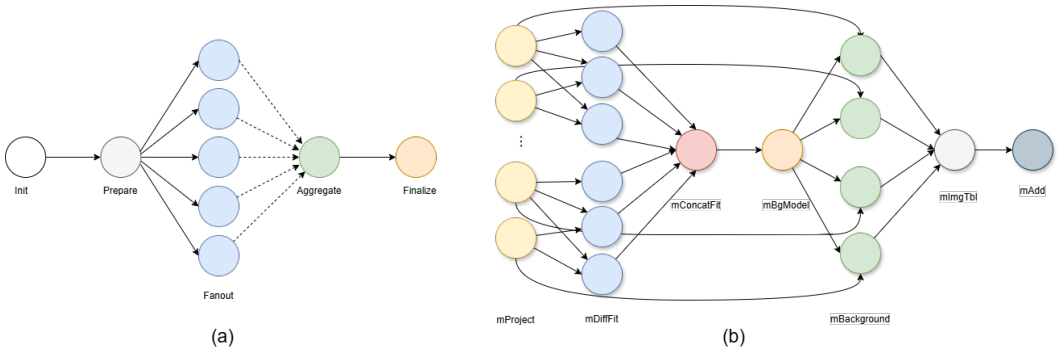

Figure 2. (a) Synthetic dataflow and (b) Montage scientific dataflow.

we evaluate three storage strategies and add an artificial activity to the workflow to share data files (according to the chosen storage strategy). The first storage strategy uses persistent volumes backed by local or shared file systems (*e.g.*, SSD storage or Lustre). In this configuration, the datasets remain co-located with the execution environment, avoiding remote data transfers during dataflow execution. The second strategy relies on object storage services (*e.g.*, Google Cloud Storage – GCS). The third considers external repositories such as Dataverse [Crosas 2011], which support data management and reproducibility but may introduce access overhead. All dataflows are executed as containerized tasks using `AkôFlow` dataflow engine, ensuring portability across environments. Tasks read and write data through storage drivers for each storage strategy implemented in `AkôFlow`. During execution, we collect metrics including total runtime (*i.e.*, makespan), data transfer time, and throughput. These support the *Quantitative and Qualitative Analysis*, enabling evaluation of storage overheads and the impact of data size and granularity on performance. Table 1 presents the experimental design, enabling analysis of the results to identify the conditions under which each approach performs best and to provide guidelines for efficient data management in containerized dataflows across the computing continuum.

Table 1. Experimental factors and evaluated configurations.

Factor	Configurations
Workflow	(i) Synthetic workflow with 100,000 individual files
	(ii) Synthetic workflow with 5 files containing 20,000 compressed files
	(iii) Montage scientific workflow
Environment Combination	(i) Local Workstation + PlaFRIM Cluster
	(ii) Local Workstation + Google Cloud Platform (GCP)
Storage Strategy	(i) Persistent Volume (shared POSIX file system)
	(ii) Object Storage (Google Cloud Storage – GCS)
	(iii) External Repository (Dataverse)
Total Experiments	$3 \times 2 \times 3 = 18$ configurations

4. Results and Discussion

This section presents the results of our experimental evaluation, focusing on the impact of different storage strategies on the execution of containerized dataflows across the computing continuum. Table 2 summarizes the execution time, data transfer overhead, and throughput obtained for the Montage and the synthetic dataflows under the PlaFRIM and Google Cloud Platform (GCP) environments. Overall, the results demonstrate that storage strategies strongly influence dataflow performance, particularly in scenarios involving intensive data exchange or large numbers of files. Among the evaluated configurations, GCS achieved the

best balance between portability and performance across environments. For the Montage workflow, GCS introduced relatively small transfer overheads in the cloud environment, requiring only 37 seconds of additional transfer time and reaching a throughput of 177.02 MB/s on GCP, while in PlaFRIM the transfer overhead increased to 402 seconds with a throughput of 16.29 MB/s. These results indicate that object storage can support distributed executions across the computing continuum, especially when storage services are co-located with cloud resources.

Persistent volumes based on shared POSIX file systems consistently achieved the lowest execution times because no data transfer was required. For example, the Montage workflow completed in 1,846 seconds on PlaFRIM and 1,898 seconds on GCP using persistent volumes. Similarly, the synthetic dataflows also showed the best raw execution performance with this strategy. Although persistent volumes are efficient in tightly coupled environments, they depend on shared storage infrastructure and offer limited portability across geographically distributed systems. This limitation becomes more evident in continuum scenarios where compute and storage resources are decentralized. The experiments with Dataverse revealed higher overheads compared with the other approaches. In the Montage dataflow, Dataverse increased transfer times to 1,512 seconds on PlaFRIM and 3,507 seconds on GCP, reducing throughput to 4.33 MB/s and 1.87 MB/s, respectively. The impact was even clearer in the synthetic dataflows composed of 100K small files, where execution became impractical. These results demonstrate that external repositories designed for data publication and preservation are not suitable for intensive, intermediate data exchanges during workflow execution, although they can be used for the final outcomes of dataflows.

Data granularity also affected performance. The synthetic workload with 100K individual files exposed scalability limitations for both GCS and Dataverse. Although GCS remained operational, transfer overheads became dominant, reaching 9,616 seconds on PlaFRIM and 2,883 seconds on GCP, resulting in low throughput. Dataverse, on the other hand, was unable to handle this dataflow efficiently. However, when the same dataset was grouped into `tar.gz` files, performance improved for both GCS and Dataverse. In this configuration, transfer times were reduced to under 105 seconds across environments, and throughput became comparable between storage strategies. This result highlights that aggregating small files into larger data objects is important for reducing metadata overhead and improving storage efficiency in distributed environments. Comparing the two execution environments, GCP generally provided better transfer performance for GCS due to native integration with the cloud object storage infrastructure. PlaFRIM, on the other hand, presented more stable execution times for persistent volumes due to its high-performance local storage system. These observations reinforce that storage efficiency depends not only on the storage strategy itself but also on its proximity and integration with the execution infrastructure. In summary, the experiments reveal that no single storage strategy is optimal for all scenarios. Persistent volumes remain the most efficient solution for localized environments with shared infrastructure, while GCS offers the best balance of scalability, portability, and performance for executions spanning the computing continuum. Dataverse, although unsuitable for intermediate dataflow communication, remains valuable for final data publication, sharing, and reproducibility. These findings provide important insights for designing efficient containerized dataflows and selecting appropriate storage mechanisms in heterogeneous and distributed computing environments.

Table 2. Experimental results.

Dataflow	Environment	Storage Strategy	Exec. (s)	Transfer (s)	Throughput (MB/s)
Montage	Local + PlaFRIM	Persistent Volume	1,846.00	0.00	—
		GCS	2,248.00	402.00	16.29
		Dataverse	3,358.00	1,512.00	4.33
	Local + Cloud (GCP)	Persistent Volume	1,898.00	0.00	—
		GCS	1,935.00	37.00	177.02
		Dataverse	5,405.00	3,507.00	1.87
Synthetic(100K Files)	Local + PlaFRIM	Persistent Volume	970.00	0.00	—
		GCS	10,586.00	9,616.00	0.0004
		Dataverse	∞	∞	0.00
	Local + Cloud (GCP)	Persistent Volume	505.00	0.00	—
		GCS	3,388.00	2,883.00	0.0013
		Dataverse	∞	∞	0.00
Synthetic(Compressed Files)	Local + PlaFRIM	Persistent Volume	516.00	0.00	—
		GCS	533.00	17.00	1.23
		Dataverse	534.00	18.00	1.16
	Local + Cloud (GCP)	Persistent Volume	639.00	0.00	—
		GCS	735.00	96.00	0.22
		Dataverse	744.00	105.00	0.20

5. Conclusion

This paper evaluated storage strategies for executing containerized dataflows across the computing continuum, considering heterogeneous environments composed of HPC and cloud infrastructures. Through experiments using synthetic dataflows and the Montage scientific dataflow, we analyzed the impact of persistent volumes, object storage services, and external repositories on execution time, transfer overhead, and throughput. The results demonstrate that storage selection affects the performance and scalability of containerized dataflows. Persistent volumes achieved the best execution performance in local environments due to the absence of explicit data transfer overhead. However, this approach depends on tightly coupled infrastructures and offers limited portability across distributed environments.

Object storage based on GCS provided the best trade-off between portability, scalability, and performance, particularly in cloud-integrated scenarios. On the other hand, Dataverse introduced high overhead during intermediate data exchanges, indicating that repository-oriented platforms are better suited to final data publication and reproducibility than to runtime communication. Overall, this paper contributes an experimental analysis of storage behavior for containerized dataflows spanning the computing continuum. The results provide practical guidelines for selecting storage strategies according to dataflow characteristics and execution environments. As future work, we plan to investigate adaptive scheduling mechanisms that dynamically select storage strategies during execution, and to evaluate additional continuum scenarios involving edge infrastructures.

Acknowledgments

The authors thank CAPES, CNPq, and FAPERJ for their partial support in carrying out this work. Additionally, the authors transparently declare that LLM-based tools were used exclusively for text revision.

References

Al-Dulaimy, A. et al. (2024). The computing continuum: From iot to the cloud. *Internet of Things*, 27:101272.

Bachiega, N. G. et al. (2020). Performance evaluation of container's shared volumes. In *2020 IEEE ICSTW*, pages 114–123.

Chazapis, A. et al. (2021). A unified storage layer for supporting distributed workflows in kubernetes. In *Proc. of the CHEOPS '21*, New York, NY, USA. ACM.

Crosas, M. (2011). The dataverse network®: An open-source application for sharing, discovering and preserving data. *D Lib Mag.*, 17(1/2).

de Oliveira, D. et al. (2019). *Data-Intensive Workflow Management: For Clouds and Data-Intensive and Scalable Computing Environments*. Morgan & Claypool Publishers.

Di Tommaso, P. et al. (2017). Nextflow enables reproducible computational workflows. *Nature Biotechnology*, 35(4):316–319.

Köster, J. and Rahmann, S. (2012). Snakemake—a scalable bioinformatics workflow engine. *Bioinformatics*, 28(19):2520–2522.

Liu, F. et al. (2019). Kubestorage: A cloud native storage engine for massive small files. In *2019 6th BESC*, pages 1–4.

Mercl, L. and Pavlik, J. (2019). Public cloud kubernetes storage performance analysis. In *ICCCI 2019*, page 649–660, Berlin, Heidelberg. Springer-Verlag.

Merenstein, A. et al. (2021). CNSBench: A cloud native storage benchmark. In *19th USENIX FAST*, pages 263–276. USENIX Association.

Plale, B. A. et al. (2021). Reproducibility practice in high-performance computing: Community survey results. *Comput. Sci. Eng.*, 23(5):55–60.

Ponte Ahón, S. A. et al. (2025). Retroh-unlp: Conservation of the historical observational work of the astronomical observatory of la plata with computer vision. *J. Comput. Cult. Herit.*, 18(4).

Sakellariou, R. et al. (2009). Mapping workflows on grid resources: Experiments with the montage workflow. In *ERCIM W. Group on Grids*, pages 119–132.

Shah, S. T., Lahaye, R. J. W. E., Kazmi, S. A. A., Chung, M. Y., and Hasan, S. F. (2014). Htcondor system for running extensive simulations related to D2D communication. In *ICTC*, pages 283–284. IEEE.

Struhár, V., Behnam, M., Ashjaei, M., and Papadopoulos, A. V. (2020). Real-time containers: A survey. In *Fog-IoT*, volume 80 of *OASIcs*, pages 7:1–7:9.

Suter, F. et al. (2026). A terminology for scientific workflow systems. *Future Gener. Comput. Syst.*, 174:107974.

Tarasov, V. et al. (2019). Evaluating docker storage performance: from workloads to graph drivers. *Cluster Computing*, 22(4):1159–1172.

Wang, L. et al. (2018). Sciapps: a bioinformatics workflow platform powered by xsede and cyverse. In *Proc. of the PEARC '18*, PEARC '18, New York, NY, USA. ACM.

Woyames, P. et al. (2025). Avaliação da capacidade de llms para especificar workflows. In *Anais do XIX Brazilian e-Science Workshop*, pages 81–88, Porto Alegre, RS, Brasil. SBC.

Zhao, N. et al. (2021). Large-scale analysis of docker images and performance implications for container storage systems. *IEEE Trans. Parallel Distrib. Syst.*, 32(4):918–930.