Towards a Service-based Adaptable Data Layer for Cloud Workflows

Wang, Y.; Janse, N.; Bianchi, R.; Koulouzis, S.; Zhao, Z.

DOI
10.1109/COMPSAC57700.2023.00121

Publication date
2023

Document Version
Author accepted manuscript

Published in
2023 IEEE 47th Annual Computers, Software, and Applications Conference

Towards a service-based adaptable data layer for Cloud workflows

Yuandou Wang[1], Nikita Janse[1], Riccardo Bianchi[2], Spirios Koulouzis[1], Zhiming Zhao[1]
[1]Multiscale Networked Systems, University of Amsterdam, the Netherlands
[2]LifeWatch ERIC, Virtual Lab & Innovation Center (VLIC), the Netherlands
Email: {y.wang8, z.zhao}@uva.nl

***Abstract*—Many scientific workflows are data-driven and need to be continuously executed for the large volume of datasets transferred from distributed data sources. The overhead arising from data transfers must be considered when optimizing workflow performance. Many workflow systems support various data transfer protocols (DTPs) and file systems, which lack flexibility for adopting new options. The major problem is to determine which protocol or file system should be selected. In this paper, we prototype a container-native workflow data layer that supports multiple DTPs, e.g., FTP, WebDAV, and IPFS, and assess their data transfer performance. We base this tool has demonstrated the feasibility of using combinations of Docker, CWL, and Argo to provide performance analysis for workflow scenarios. Our results show that: 1) IPFS outperforms WebDAV in uploading large files; 2) the makespan via IPFS executed in Argo is comparable with WebDAV, and IPFS's total data transferred is always less than WebDAV.**

***Index Terms*—Container-native workflow data layer, IPFS, WebDAV, data transfers, performance analysis, Cloud computing**

I. Introduction

In today's data-intensive or data-driven science era, many large-scale scientific experiments are data-driven processing. They frequently involve the movement of extensive datasets across infrastructure set up on file systems in support of many domains such as astronomy, seismology, experimental biology, and environmental science [1]. The changing landscape of scientific research has created a pressing need for computational pipelines capable of efficiently orchestrating complex analysis stages while handling large volumes of data across heterogeneous computational environments [2].

During past decades, scientific workflow management systems (WfMSs) have demonstrated their great value in enabling domain-specific researchers or scientists to automate and parallelize computational analyses from local High-performance Computing (HPC) to remote Cloud or hybrid environments [3], for example, Pegasus [4], Galaxy [5], Arvados [6], and Argo[7]. The underlying data management plays a key role in these WfMSs. Many different approaches have been developed to address the problem of delivering of extensive data sets across geographically distributed and heterogeneous storage systems [8]. Most of such approaches aim to deal mostly with data transfer, data sharing and synchronization, authorisation, security and privacy regarding data compliance (e.g., GDPR[1] and HIPAA[2]), provenance, and reproducibility. The available data services mentioned in their respective documentations adopt various file systems and data transfer protocols for data transfers, such as FTP [9] and WebDAV [10]. However, these WfMSs lack flexibility for adopting new options.

In applications like clinical diagnostic, there is a special focus on the integrity and regulated access to data throughout each pipeline stage as per applicable laws and regulations [2]. In this context, decentralised data management is a typical paradigm for enabling trustworthiness among distributed parties, e.g., handling, storing, and managing medical datasets [11]. Correspondingly, decentralised storage systems and protocols such as IPFS [12], Storj [13], SWARM [14], and Sia [15] are widely working together with blockchain-based solutions or frameworks to address the inefficient and costly problem of storing large volumes of data [16]–[18]. Some researchers have pointed out that IPFS, if built right, could complement or replace HTTP and some other protocols in the discussion about data management tools and technologies [8]. However, the study of decentralised data management (e.g., IPFS) in WfMSs has not been taken seriously and published research in this field is countable. Besides, various approaches exist to study data transfer protocols adopted in WfMSs, such as WebDAV [19] and peer-to-peer (P2P) data sharing [20], while the performance benchmark among FTP, WebDAV, and IPFS for remote data transfers are still non-existent.

To explore the difference between conventional and decentralised data solutions within a workflow context, this paper proposes a container-native workflow data layer, adaptable to multiple data transfer protocols, i.e., FTP, WebDAV, and IPFS, that combines with Docker, Common Workflow Language (CWL), and Argo workflow engine for workflow deployment, execution, and

[1]GDPR. https://gdpr-info.eu/
[2]HIPAA. https://www.hhs.gov/hipaa/index.html

performance analysis. The primary contributions of this paper can be summarized as follows:

- We extensively assess the upload and download performance between remote sites via FTP, WebDAV, and IPFS in transferring different file sizes. Our results present that the P2P protocol (i.e., IPFS) data transfers can output files much more quickly than that FTP and WebDAV in bigger sizes.
- We prototype a container-native workflow data layer for supporting WebDAV and IPFS data services for workflow execution using the Argo workflow engine for Kubernetes in a Cloud environment. The Docker container images and software tool named `dm-interface` are open-source and published in the Docker Hub[3] and PyPI[4], respectively.
- We adopt the proposed data layer on the Argo workflow engine using four data processing workflows written in CWL and compare the workflow execution time between IPFS and WebDAV. Our results show that in some cases, using IPFS data service for Argo workflows outperforms WebDAV.

II. Background

Many implemented data management approaches in WfMSs support several file systems and data transfer protocols for local and remote file transfers.

Pegasus, in practice, refers to a Python script called pegasus-transfer as the executable in the transfer jobs to transfer the data. It provides smooth data transfers for data staging via many protocols and remote file systems, including Amazon S3 and compatible services, Docker, File / Symlink, Globus Online, GridFTP, GridFTP over SSH, Google Storage, HTTP, HPSS, iRODS, OSG Stash, SCP, Singularity, and WebDAV [21]. Likewise, uploading data into Galaxy can be done in many ways, including local file upload, FTP, URL, and SRA uploads [22]. In later releases, Galaxy also supports WebDAV transfers and resumable file uploads with tus. Besides, within the bioinformatics community, Arvados combines content-addressing with a distributed storage architecture inspired by Google File System (GFS) for data management, primarily to address the unique needs of biomedical data, such as provenance, reproducibility, and data validation. It utilises a CAS system named Keep, in which the data is referenced by a cryptographic hash of its contents (e.g., the MD5 hash), and can flexibly store data in S3, S3-compatible storage systems (e.g. Ceph) and Azure blob storage, as well as store data on POSIX file systems. Data transfer protocols like HTTP, WebDAV, and Amazon S3 correspond to this data management approach [23]. However, these mainstream WfMSs lack the flexibility for new options, such as decentralised storage systems and related P2P protocols.

Similarly, IPFS [12] adopts CAS technique to uniquely identify each file in a global namespace connecting IPFS hosts. A few work has been done in distributed workflow management with IPFS. For instance, Hukkinen *et al.* [24] reported a blockchain application for the real estate sector, which enables distributed workflow management in a complicated transaction process more efficiently and transparently. IPFS is used to store the files from several public and private information pools; Ethereum-based smart contracts written in Solidity are compiled and deployed to the blockchain. Besides, decentralised storage systems and P2P protocols such as Storj [13], SWARM [14], and Sia [15] are widely used in blockchain-based solutions or frameworks to address the inefficient and costly problem of storing large volumes of data. However, such technologies are seldom connected to WfMSs.

III. Prototype Design and Implementation

To bridge the gap between WfMSs and new data solutions (either centralised or decentralised ones), we design and prototype an adaptable data layer for WfMSs, which can be used for adopting new data transfer protocols and file systems flexibly. Two typical examples are considered at this stage: IPFS and WebDAV.

A. Design

Fig. 1 presents core components of the proposed data layer for atomic task execution in a machine (e.g., Cloud virtual machine or HPC node), in which the data service directly interoperates with the workflow task deployed in a machine through HTTP request and response APIs. The service also interoperates with local data, such as read and write data operations of the file system. Besides, when handling distributed workflow tasks, the task should communicate with other nodes in the cluster network, and all the data transfers satisfy the tasks' predecessor and successor constraints.

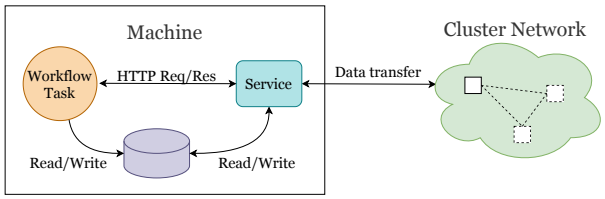


Fig. 1: An overview of core components of our data layer in an atomic task execution deployed in a machine.

B. Implementation

For efficient implementation of the above design, we adopt containerisation technology combined with the FastAPI Web framework because of their popularity in the communities and easy-to-use features, as shown in

[3]Docker images. https://hub.docker.com/u/nicoja
[4]dm-interface. https://pypi.org/project/dm-interface/

Fig. 2. In this work, we consider the Docker container as the critical technology for packaging IPFS and WebDAV services. Hence, such data services run quickly and reliably from one computing environment to another. Meanwhile, FastAPI[5] is easy to develop RESTful APIs in Python with a well-established HTTP protocol, which can flexibly combine new options such as IPFS with a P2P setting. Specifically, we prototype an IPFS-based data service to enable the data transfers with a P2P manner, working in the data layer of distributed workflows through the use of the `ipfshttpclient` module. The corresponding service images are `ipfs-node`[6] and `ipfs-service2`[7].

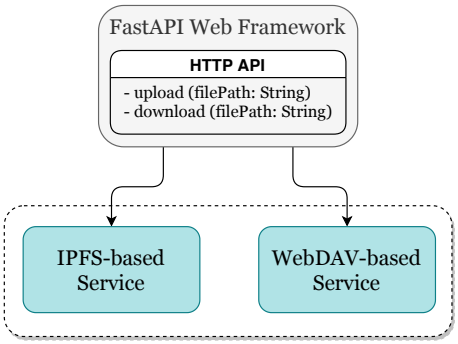


Fig. 2: An overview of the implementation of our prototype with IPFS and WebDAV.

To compare the data transfer performance between P2P and client/server settings in benchmark experiments, we also prototype the WebDAV-based data service and adopt an FTP-based data service, respectively. The WebDAV server is reproduced from the docker image `webdav`[8], while `webdav-service2`[9] is developed through the use of the `webdavclient3` module. For the FTP-based service, we adopt a pure FTP data service, i.e., `pure-ftpd`[10], detailed in the Docker Hub.

IV. Performance Benchmark

This section describes the benchmark experiments we conduct for assessing data transfer performance among FTP, IPFS, and WebDAV. Analysis and insights are given based on the observations.

A. Experimental setup

a) Cloud virtual machines (VMs): We employed four VMs from cloud infrastructure to conduct benchmark experiments, as presented in Table. I. The average network speed (i.e., throughput) between any two nodes is 245Mbits/s in a public network environment. We assume the network among these nodes is stable enough to enable the correct scope and practical implementation of benchmark experiments.

TABLE I: Specifications of Cloud virtual machines

VM	CPU	RAM (gb)	SDD (gb)	Public IP
v1	2	7	50	51.210.39.xx
v2	2	7	50	146.59.192.xx
v3	4	15	100	146.59.206.xx
v4	4	15	100	152.228.166.xx

b) Data sizes: Due to the storage limits, the corresponding file sizes are set from 2^0KB to 2^{24}KB. To automate the benchmark, we adopt binary files created by code for data transfers. We also consider that data transfers are separate from uploading and downloading specific sizes of files between two machines.

c) Measurement points: To assess the data transfer performance in support of FTP, IPFS, and WebDAV, we consider the following measurement points set as timestamps to log files for statistical analysis. We also set benchmark runner parameters, e.g., run count=4 and interval seconds=30 for the executable. Tables IIa and IIb describe when uploading and downloading a file the timestamps we set in different protocols, respectively. Collecting the timestamps in the log files can calculate the data transfer times in practice.

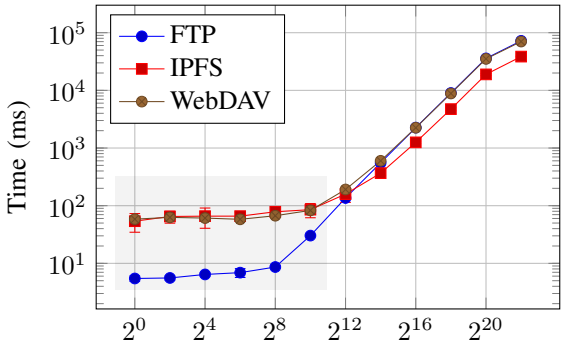
B. Statistical analysis results

We collect about 306 log files ($17 \times 6 \times 3$), where 3 indicates the FTP, IPFS, and WebDAV network protocols, 6 refers to the number of VM pairs, and 17 is the number of file sizes. We work out the average data transfer times (either upload and download times) by calculating available timestamps in these log files by statistical analysis.

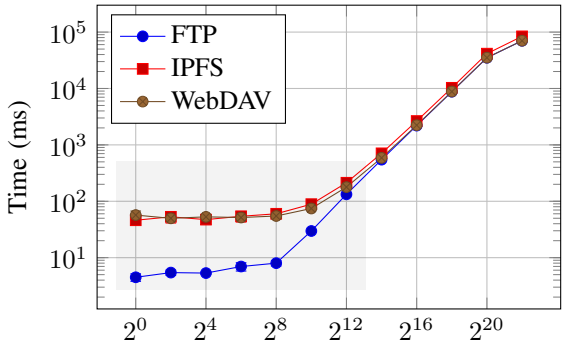
Fig. 3 compares average upload and download times of FTP, IPFS, and WebDAV between V1 and V2. Fig. 3a shows that for small files, the average upload times of FTP consistently outperform that of IPFS and WebDAV, whose splitting point is at 2^{12}KB. Afterwards, the average upload times of IPFS always outperform that of FTP and WebDAV; simultaneously, it indicates no differentiation between the average upload times of FTP and WebDAV. Likewise, the average download times of FTP outperform that of IPFS and WebDAV for small files, as depicted in Fig. 3b, and for larger files, the average download times of FTP, IPFS, and WebDAV have almost no difference in our statistical results.

Fig. 4 illustrates more detailed comparisons of the average upload and download times among six VM pairs for FTP, IPFS, and WebDAV. The trend of average download times of FTP, IPFS, and WebDAV is pretty stable, in which all curves in FTP (Fig. 4b), IPFS (Fig. 4d), and WebDAV (Fig. 4f) are almost coincident, respectively.

[5]FastAPI. https://fastapi.tiangolo.com/
[6]ipfs-node. https://hub.docker.com/r/nicoja/ipfs-node
[7]ipfs-service2. https://hub.docker.com/r/nicoja/ipfs-service2
[8]webdav. https://hub.docker.com/r/bytemark/webdav/
[9]webdav-service2. https://hub.docker.com/r/nicoja/webdav-service2
[10]pure-ftpd. https://hub.docker.com/r/stilliard/pure-ftpd/



(a) Average upload time



(b) Average download time

Fig. 3: Average transfer times: uploading and downloading times between VMs: v1-v2

TABLE II: The description of measurement points for FTP, IPFS, and WebDAV

Protocol	No. (#)	Description
FTP	1-2	**start (#1)** to **end (#2)**, steps `open_file`, `storbinary`, and `close_file` are included.
IPFS	1-2	**start (#1)** to `add_file`
	2-3	`add_file`
	3-4	`write_file_hash`
	4-5	`get_root_hash`
	5-6	`publish_root`
	6-7	`add_path_hash`
	7-8	`add_path_hash` to **end (#8)**
WebDAV	1-2	**start (#1)** to `webdav.mkdir`
	2-3	`webdav.mkdir`
	3-4	`webdav.upload`
	4-5	`webdav.upload` to **end (#5)**

(a) Measurement points set for uploading a file

Protocol	No. (#)	Description
FTP	1-2	**start (#1)** to **end (#2)**, steps `open_file`, `retrbinary`, and `close_file` are included.
IPFS	1-2	**start (#1)** to `get_path_hash`
	2-3	`get_path_hash`
	3-4	`get_peer_ids`
	4-5	`get_file_hash`
	5-6	`download_file`
	6-7	`download_file` to **end (#7)**
WebDAV	1-2	**start (#1)** to `webdav.download`
	2-3	`webdav.download`
	3-4	`webdav.download` to **end (#4)**

(b) Measurement points set for downloading a file

Likewise, the curves of average upload times in FTP (Fig. 4a) and WebDAV (Fig. 4e) are nearly overlapped. Yet the curves of average upload times in IPFS (Fig. 4c) slightly deviate from each other, as well as it costs less time than WebDAV and FTP.

Based on observations in Fig. 3, we further analyze the upload and download times of WebDAV and IPFS for small file transfers (i.e., $2^0 \sim 2^{12}$KB), as depicted in Fig. 5, with measurement points described in Table II. It is seen that for uploading files, the section `add_file` accounts for the largest proportion of the average upload time of IPFS data transfers (Fig. 5a). Regarding WebDAV (Fig. 5c), the time cost by `webdav.mkdir` section accounts for a large proportion but is steady and as the size increased to 2^{12}KB, `webdav.upload` increases sharply. Similarly, for downloading files, the `download_file` section in IPFS (Fig. 5b) accounts for the largest proportion compared with that of the section `webdav.download` in WebDAV (Fig. 5d).

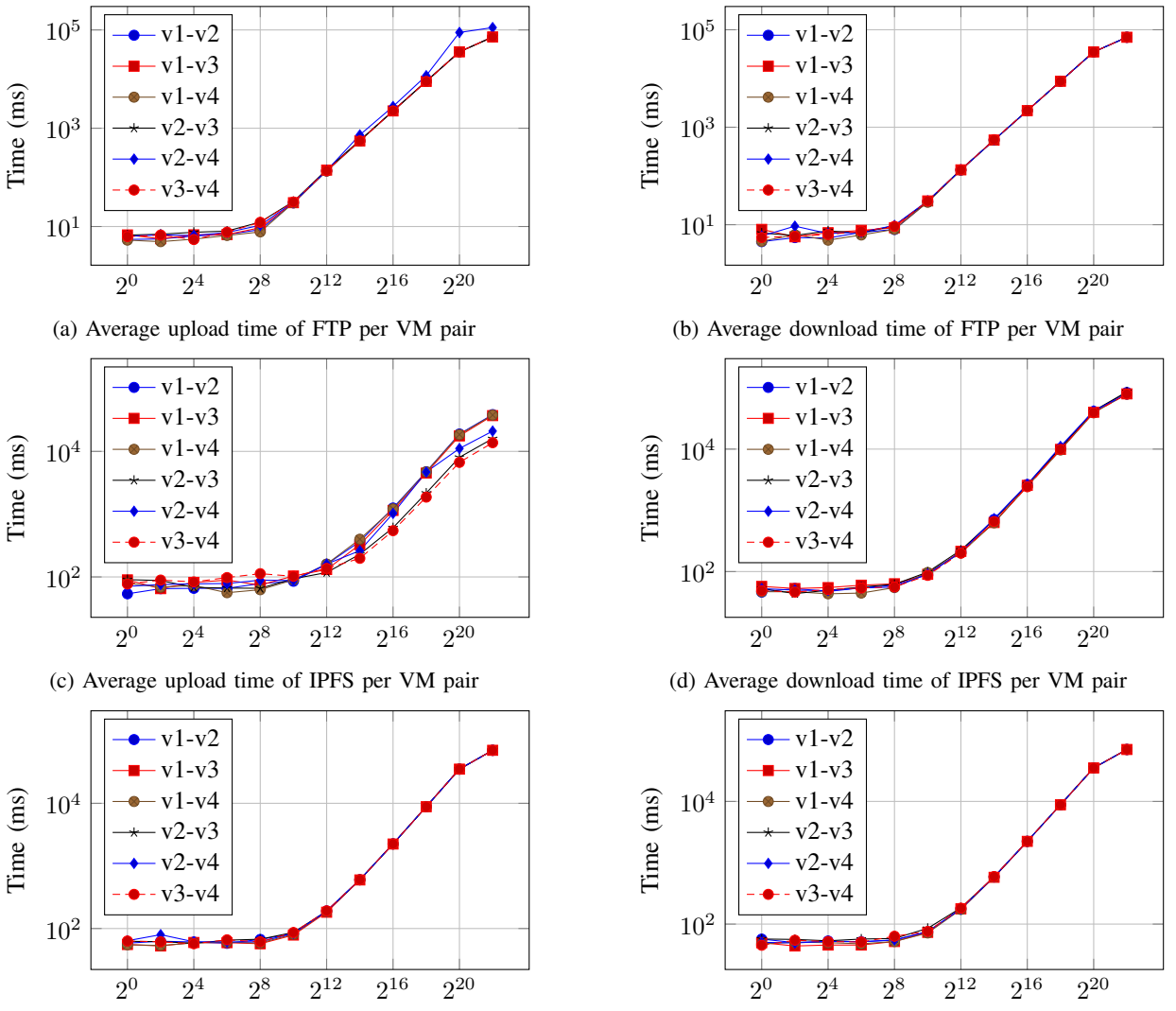
Based on these observations, we can see that IPFS has advantages in uploading large files, while for small files, it performs almost the same as WebDAV. Meanwhile, our results show that IPFS and WebDAV almost have no difference in the average download times.

V. Case Study and Application

In this section, we adopt an open source container-native workflow engine - Argo, workflow application scenarios, and our tool to conduct the case study for demonstrating the feasibility and evaluating the workflow performance purposes.

A. Usage

In theory, a workflow G is formalized as a Directed-Acyclic-Graph (DAG) with several $n+2$ tasks $T = \{t_0, t_1, t_2, ..., t_n, t_{n+1}\}$ satisfying the predecessor and successor constraints. Every DAG starts and concludes with an entry task t_0 that does not have predecessors and an exit task t_{n+1} without successors. The set $E = T \times T$ represents the set of directed edges of the DAG, where $e_{ij} = (t_i, t_j) \in E$ indicates the communication overhead between a task t_i and its successor t_j, both in T. In practice, a workflow execution usually involves three dimensions: control-flow, data, and resource, which always go together with WfMS. To demonstrate the feasibility of our approach and simplify operations, we assume each task is executed in a resource node at one time, and all tasks are data processing operations thus, the control flow is essentially in agreement with the data flow.



(a) Average upload time of FTP per VM pair

(b) Average download time of FTP per VM pair

(c) Average upload time of IPFS per VM pair

(d) Average download time of IPFS per VM pair

(e) Average upload time of WebDAV per VM pair

(f) Average download time of WebDAV per VM pair

Fig. 4: Comparisons of the average upload and download times among six vm pairs for FTP, IPFS and WebDAV

B. *Workflow scenarios*

To evaluate the performance of workflow execution based on IPFS and WebDAV data transfers, we consider both synthetic workflow and real-world application scenarios.

a) ***Synthetic Workflows****:* As shown in Fig. 6, we adopt one pure sequential and two sequential/parallel-combined workflows. Supported task types include `create`, `read`, `write`, `partition`, and `merge` files. To task executions, the corresponding timestamps consist of `time_start`, `time_create`, `time_read`, `time_write`, `time_upload`, and `time_download`, respectively.

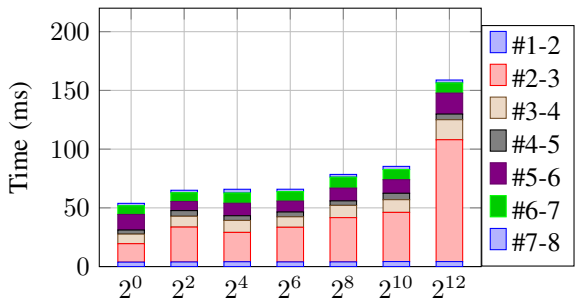
b) ***EcoLidar Application****:* A real-world application named ecoLidar - a data processing pipeline related to Light Detection and Ranging (LiDAR) point clouds, is also adopted. It is mainly comprised of three different task types, namely `retile`, `process`, and `rasterize`.

C. *Computational environment*

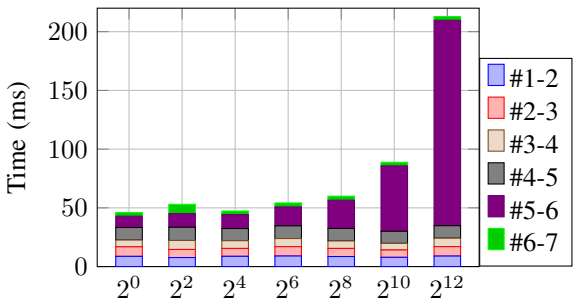
To simplify the operations and demonstrate the feasibility of our approach, we establish an Argo workflow engine for K8S atop four Cloud VMs, then submit the above workflow scripts written in CWL to the system for workflow execution.

D. *Results and Analysis*

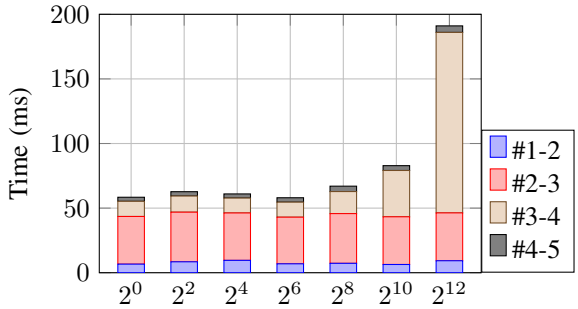
Fig. 7 presents the comparison of the average makespan between IPFS and WebDAV data transfers in synthetic workflow execution. Both scenarios #1 (Fig. 7a) and #2 (Fig. 7b) in most cases the average makespan of IPFS is less than that of WebDAV for workflows fed with larger input files. However, in scenario #3 (Fig. 7c), the average makespan of WebDAV outperforms that of IPFS.



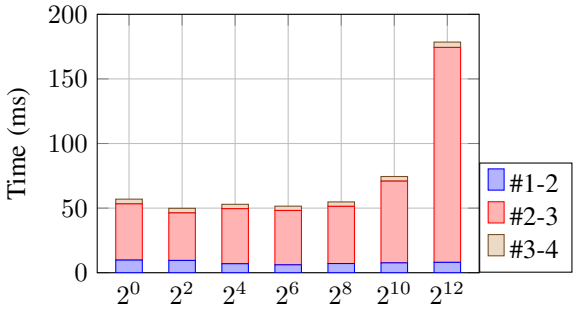
(a) Average upload time of IPFS per section



(b) Average download time of IPFS per section

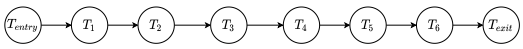


(c) Average upload time of WebDAV per section

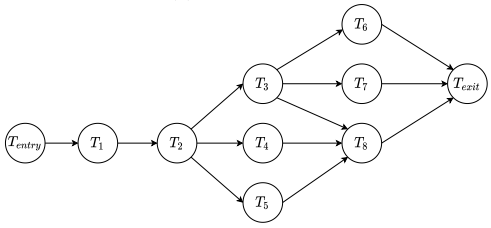


(d) Average download time of WebDAV per section

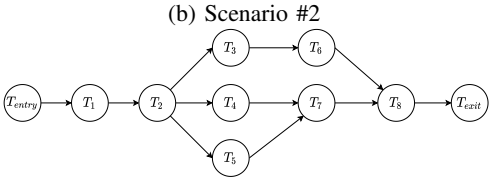
Fig. 5: Comparison of the average upload and download time cost per section between v1 and v2 for IPFS and WebDAV



(a) Scenario #1



(b) Scenario #2



(c) Scenario #3

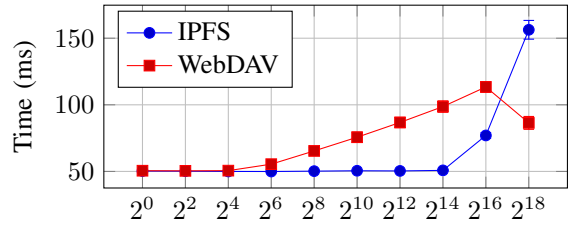
Fig. 6: An overview of the three synthetic workflows

Regarding the ecoLidar application, we ran it fed with an 8.9MB input file for 10 times in the computational environment. We did not conduct experiments in different input sizes because the datasets of the application are not regular ascending series. Table. III lists the comparisons between IPFS and WebDAV in terms of ecoLidar's makespan and total data transferred. Six-tenth run times of IPFS are no more than that of WebDAV. At the same time, on average, IPFS is considerable with WebDAV regarding the workflow makespan. Meanwhile, all data (MB) transferred during the workflow executions, the data sizes transferred with IPFS are always less than that with WebDAV in practice.

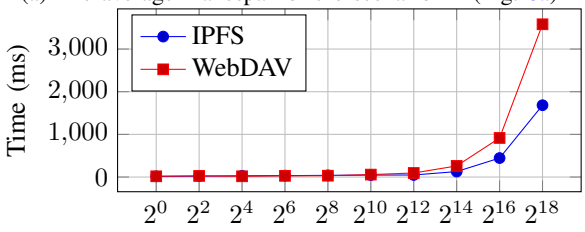
TABLE III: The makespan and total data (MB) transferred for each run of Lidar workflow

ecoLidar	Makespan		Data Transferred	
Round (#)	IPFS	WebDAV	IPFS	WebDAV
1	176	173	**139**	405
2	**157**	213	**143**	406
3	**158**	**180**	**121**	406
4	**183**	**186**	**136**	405
5	**156**	**174**	**138**	405
6	183	167	**136**	406
7	215	155	**63**	405
8	**180**	**180**	**165**	405
9	**188**	**189**	**106**	405
10	204	178	**165**	406
Avg.	**180**	**180**	**131**	405

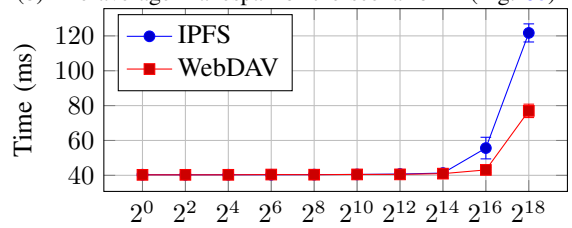
As a result, both synthetic workflows and ecoLidar application scenarios show that the makespan of IPFS-based workflow executed in the Cloud platform is comparable with that of WebDAV. The total MB files transferred for each run of the ecoLidar workflow through IPFS are always less than that through WebDAV.



(a) The average makespan of the scenario #1 (Fig. 6a)



(b) The average makespan of the scenario #2 (Fig. 6b)



(c) The average makespan of the scenario #3 (Fig. 6c)

Fig. 7: Comparisons of the average makespan between IPFS and WebDAV data transfers adopted in the three synthetic workflows

VI. Discussion

A. Observations

We observe that when uploading relatively large files (i.e., no less than 2^{12}KB) in a public network environment, IPFS will cost less time than FTP and WebDAV, and when downloading relatively large files (i.e., no less than 2^{14}KB), there is no difference in the average download times among the three protocols. On the contrary, while uploading and downloading small files (e.g., in the range of $2^0 \sim 2^{10}$KB), FTP always outperforms IPFS and WebDAV, where WebDAV and IPFS perform almost the same in the average upload and download times. Moreover, the case study also shows that the makespan via IPFS executed in the Argo workflow engine is comparable with that via WebDAV and the total MB data transferred through IPFS is always less than that through WebDAV.

B. Weaknesses

Regarding the performance benchmark, we did not focus on the underlying level but the application level of networking protocols. The data sizes are also limited because we did not have a larger volume of storage resources. These factors may influence the performance assessment results of remote data transfers. In the case study, our proposed tool was utilized in the Argo workflow engine for K8S, a centralized cloud environment configured with distributed VMs. Then, the IPFS-based data service was used in a centralized manner. However, it is natural decentralized, and so much potential deserves to be tapped.

VII. Conclusion and Future Work

This paper presents our experiments for assessing the performance of remote data transfers with IPFS, WebDAV, and FTP between remote VMs. The statistical results present that IPFS has the advantage of uploading large files than FTP and WebDAV, while for small file transfers, it has almost no difference from WebDAV in our experimental settings. Inspired by these findings, we conducted a case study to compare WebDAV and IPFS-based data transfers for three synthetic workflows and a real-world application scenario, using combinations of Docker container images, CWL-based software pipelines, and Argo workflow engine, and our container-native workflow tool. The empirical results show that in terms of the average makespan (i.e., the total workflow execution time), using IPFS for data transfers is comparable with that of WebDAV.

In future work, we plan to test our techniques in federated learning (FL) pipelines and compare them with classical data transfer protocols for FL message communication. There is a growing need for reliable and trustworthy artificial intelligence (AI) and for making scientific data FAIR - findable, accessible, interoperable, and reusable. We also plan to explore more IPFS possibilities for responsible artificial intelligence in a combination of FAIR computational workflows [25]. We believe that the incorporation of AI into routine FAIR workflow management may provide a solution.

CRediT authorship contribution statement

Yuandou Wang: Conceptualization, Methodology, Investigation, Formal analysis, Writing - Original Draft, Writing - Review & Editing. **Nikita Janse:** Conceptualization, Methodology, Software, Writing - Review & Editing. **Zhiming Zhao:** Conceptualization, Resources, Writing - Review & Editing, Supervision, Project administration, Funding acquisition.

Acknowledgment

This work has been partially funded by the European Union's Horizon 2020 research and innovation programme with the project CLARIFY under the Marie Sklodowska-Curie (860627), ENVRI-FAIR (824068), BlueCloud (862409) and ARTICONF(825134).

References

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

[2] A. E. Ahmed, J. M. Allen, T. Bhat, P. Burra, C. E. Fliege, S. N. Hart, J. R. Heldenbrand, M. E. Hudson, D. D. Istanto, M. T. Kalmbach *et al.*, "Design considerations for workflow management systems use in production genomics research and the clinic," *Scientific reports*, vol. 11, no. 1, pp. 1–18, 2021.

[3] P. Amstutz, M. Mikheev, M. R. Crusoe, N. Tijanić, S. Lampa, and et al. (2022, Jun.) Existing workflow systems. [Online]. Available: https://s.apache.org/existing-workflow-systems

[4] E. Deelman, K. Vahi, G. Juve, M. Rynge, S. Callaghan, P. J. Maechling, R. Mayani, W. Chen, R. F. Da Silva, M. Livny *et al.*, "Pegasus, a workflow management system for science automation," *Future Generation Computer Systems*, vol. 46, pp. 17–35, 2015.

[5] D. Blankenberg, G. V. Kuster, N. Coraor, G. Ananda, R. Lazarus, M. Mangan, A. Nekrutenko, and J. Taylor, "Galaxy: a web-based genome analysis tool for experimentalists," *Current protocols in molecular biology*, vol. 89, no. 1, pp. 19–10, 2010.

[6] P. Amstutz, "Portable, reproducible analysis with arvados," *F1000Research*, vol. 4, p. 114, 2015.

[7] Argo. (2021, Sep.) Argo workflows. [Online]. Available: https://argoproj.github.io/argo-workflows/

[8] M. Bobák, A. S. Belloum, P. Nowakowski, J. Meizner, M. Bubak, M. Heikkurinen, O. Habala, and L. Hluchỳ, "Exascale computing and data architectures for brownfield applications," in *2018 14th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*. IEEE, 2018, pp. 450–457.

[9] J. Postel and J. Reynolds, "File transfer protocol," 1985.

[10] E. J. Whitehead and M. Wiggins, "Webdav: Ieft standard for collaborative authoring on the web," *IEEE Internet Computing*, vol. 2, no. 5, pp. 34–40, 1998.

[11] S. Warnat-Herresthal, H. Schultze, K. L. Shastry, S. Manamohan, S. Mukherjee, V. Garg, R. Sarveswara, K. Händler, P. Pickkers, N. A. Aziz *et al.*, "Swarm learning for decentralized and confidential clinical machine learning," *Nature*, vol. 594, no. 7862, pp. 265–270, 2021.

[12] J. Benet, "Ipfs-content addressed, versioned, p2p file system," *arXiv preprint arXiv:1407.3561*, 2014.

[13] S. Wilkinson, T. Boshevski, J. Brandoff, and V. Buterin, "Storj a peer-to-peer cloud storage network," 2014.

[14] J. H. Hartman, I. Murdock, and T. Spalink, "The swarm scalable storage system," in *Proceedings. 19th IEEE International Conference on Distributed Computing Systems (Cat. No. 99CB37003)*. IEEE, 1999, pp. 74–81.

[15] D. Vorick and L. Champine, "Sia: Simple decentralized storage," *Retrieved May*, vol. 8, p. 2018, 2014.

[16] M. Steichen, B. Fiz, R. Norvill, W. Shbair, and R. State, "Blockchain-based, decentralized access control for ipfs," in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 2018, pp. 1499–1506.

[17] H. Kordestani, K. Barkaoui, and W. Zahran, "Hapichain: A blockchain-based framework for patient-centric telemedicine," in *2020 IEEE 8th International Conference on Serious Games and Applications for Health (SeGAH)*, 2020, pp. 1–6.

[18] S. Khatal, J. Rane, D. Patel, P. Patel, and Y. Busnel, "FileShare: A Blockchain and IPFS Framework for Secure File Sharing and Data Provenance," in *Advances in Machine Learning and Computational Intelligence*, S. Patnaik, X.-S. Yang, and I. K. Sethi, Eds. Singapore: Springer Singapore, 2021, pp. 825–833.

[19] S. Pardi and G. Russo, "A performance study of WebDav access to storages within the Belle II collaboration," in *Journal of Physics: Conference Series*, vol. 898, no. 6, 2017, p. 62038.

[20] R. Agarwal, G. Juve, and E. Deelman, "Peer-to-peer data sharing for scientific workflows on amazon ec2," in *2012 SC Companion: High Performance Computing, Networking Storage and Analysis*, 2012, pp. 82–89.

[21] Pegasus. (2022, Aug.) Data management - pegasus wms 5.0.2 documentation. [Online]. Available: https://pegasus.isi.edu/documentation/reference-guide/data-management.html

[22] Galaxy. (2022, Aug.) Uploading data into galaxy. [Online]. Available: https://galaxyproject.org/tutorials/upload/

[23] Arvados. (2021, May) A look at duplication in keep. [Online]. Available: https://arvados.org/2021/05/11/a_look_at_deduplication_in_Keep/

[24] T. Hukkinen, J. Mattila, and T. Seppälä, "Distributed Workflow Management with Smart Con-

tracts," Tech. Rep. 78, 2017.
[25] C. Goble, S. Cohen-Boulakia, S. Soiland-Reyes, D. Garijo, Y. Gil, M. R. Crusoe, K. Peters, and D. Schober, "Fair computational workflows," *Data Intelligence*, vol. 2, no. 1-2, pp. 108–121, 2020.