Scaling Notebooks as Re-configurable Cloud Workflows Postprint

Authors: Yuandou, Wang, Spiros, Koulouzis, Riccardo, Bianchi, Na, Li, Yifang, Shi, Joris, Timmermans, W. Daniel, Kissling, Zhiming Zhao, Yuandou Wang, Zhiming, Zhao

Date: 2022-11-28T00:00:00+00:00

Abstract

Literate computing environments, such as the Jupyter (i.e., Jupyter Notebooks, JupyterLab, and JupyterHub),

have been widely used in scientific studies; they allow users to interactively develop scientific code, test

algorithms, and describe the scientific narratives of the experiments in an integrated document. To scale up

scientific analyses, many implemented Jupyter environment architectures encapsulate the whole Jupyter

notebooks as reproducible units and autoscale them on dedicated remote infrastructures (e.g., high#2;

performance computing and cloud computing environments). The existing solutions are still limited in many

ways, e.g., 1) the workflow (or pipeline) is implicit in a notebook, and some steps can be generically used

by different code and executed in parallel, but because of the tight cell structure, all steps in the Jupyter

notebook have to be executed sequentially and lack of the flexibility of reusing the core code fragments, and

2) there are performance bottlenecks that need to improve the parallelism and scalability when handling

extensive input data and complex computation.

In this work, we focus on how to manage the workflow in a notebook seamlessly. We 1) encapsulate the

reusable cells as RESTful services and containerize them as portal components, 2) provide a composition

tool for describing workflow logic of those reusable components, and 3) automate the execution on remote

cloud infrastructure. Empirically, we validate the solution' s usability via a use case from the Ecology and

Earth Science domain, illustrating the processing of massive Light Detection and Ranging (LiDAR) data. The

demonstration and analysis show that our method is feasible, but that it needs further improvement, especially

on integrating distributed workflow scheduling, automatic deployment, and execution to develop as a mature

approach.

Full Text

Preamble

RESEARCH PAPER ChinaXiv 合作期刊 Scaling Notebooks as Reconfigurable Cloud Workflows

Yuandou Wang[1]†, Spiros Koulouzis[1],[2], Riccardo Bianchi[1],[2], Na Li[1], Yifang Shi[2],[3], Joris Timmermans[2],[3], W. Daniel Kissling[2],[3] & Zhiming Zhao[1],[2]†

[1]Multiscale Networked Systems, Informatics Institute, University of Amsterdam, 1098XH Amsterdam, The Netherlands
[2]LifeWatch ERIC, Virtual Lab & Innovation Center (VLIC), Science Park 904, 1098XH Amsterdam, The Netherlands
[3]Institute for Biodiversity and Ecosystem Dynamics (IBED), 1098XH Amsterdam, The Netherlands

Keywords: Scientific experiments; Jupyter Notebooks; Workflow management; Ecosystem structure data products; Cloud; Scalability

Citation: Wang, Y.D., et al.: Scaling notebooks as re-configurable cloud workflows. Data Intelligence 4(2), 409-425 (2022). doi: 10.1162/dint_a_{00140}

Received: September 19, 2021; **Revised:** January 1, 2022; **Accepted:** March 1, 2022

Abstract

Literate computing environments such as Jupyter (i.e., Jupyter Notebooks, JupyterLab, and JupyterHub) have become widely adopted in scientific research, enabling users to interactively develop code, test algorithms, and document experimental narratives within integrated executable documents. To scale scientific analyses, many existing Jupyter architectures encapsulate entire notebooks as reproducible units and autoscale them on dedicated remote infrastructures (e.g., high-performance computing and cloud environments). However, these solutions suffer from significant limitations. First, workflows (or pipelines) remain implicit within notebooks; while some steps could be generically reused across different codes and executed in parallel, the rigid cell structure forces sequential execution and lacks flexibility for reusing core code fragments. Second, performance bottlenecks persist, particularly regarding parallelism and scalability when handling extensive input data and complex computations.

This work addresses the challenge of seamlessly managing workflows within notebooks. We propose a solution that (1) encapsulates reusable cells as RESTful services and containerizes them as portal components, (2) provides a composition tool for describing the workflow logic of these reusable components, and (3) automates execution on remote cloud infrastructure. We empirically validate the solution' s usability through a use case from the ecology and earth science domain, demonstrating the processing of massive Light Detection and Ranging (LiDAR) data. Our demonstration and analysis confirm the feasibility of our method while identifying areas for further improvement, particularly in integrating distributed workflow scheduling, automatic deployment, and execution to mature into a robust approach.

Corresponding authors: Yuandou Wang (Email: y.wang8@uva.nl; ORCID: 0000-0003-4694-9572) and Zhiming Zhao (Email: z.zhao@uva.nl; ORCID: 0000-0002-6717-9418).

1. Introduction

Many scientific problems—such as significant environmental challenges or cancer diagnosis—require large data volumes, advanced modeling techniques, and distributed computing facilities [1, 2]. Literate computing environments like Jupyter Notebook, JupyterLab, and JupyterHub have exploded in popularity and emerged as a de facto standard across engineering and science domains [4, 5, 6], including ecology [10, 11], biology [12], and medical research [13, 14]. These environments elegantly combine explanatory text, software code, computational output, and multimedia resources into executable interactive documents where users input programming code or text in rectangular cells.

The narratives of scientific experiments in notebooks can be described as pipeline steps or workflows containing executable code fragments. However, workflows typically remain implicit within notebooks without explicit, structured workflow-oriented descriptions. For instance, some steps could

be represented as atomic tasks with defined input/output dependencies, but the tight cell structure prevents effective extraction of workflow patterns, limiting the reusability of generic code fragments [24]. Furthermore, from an input/output dependency perspective, performance bottlenecks persist in workflow execution; some steps require parallelization or scaling out, particularly when processing large data volumes.

Current approaches typically enable computational notebooks to run as whole jobs scaled out on dedicated remote infrastructures such as high-performance computing (HPC) settings and cloud environments [7, 15, 16, 17, 18, 19, 20, 21]. Generally, to bridge the gap between exploratory scientific analysis and computing environments, many Jupyter architectures couple with pre-configurable infrastructure (e.g., via IaaS Cloud) to dynamically deploy and manage notebook-based application instances. However, widely-used solutions neglect workflow-oriented representation and management within single notebooks—specifically, the workflow structure regarding internal dependencies and efficient parallelism for task input sizes. Consequently, managing such workflows within notebooks becomes a major bottleneck for domain researchers seeking to scale their scientific experiments.

Motivated by this challenge, we address the research question: how can we seamlessly manage workflows within notebooks? This paper provides an initial answer by proposing the Notebook-as-a-Workflow (NaaW) method. Our contributions include: (1) a prototyped component containerizer that encapsulates reusable code fragments (cells) as RESTful services and containerizes them as science portal components; (2) a workflow composition tool for visually presenting the workflow logic of these reusable components; and (3) an integrated infrastructure automation tool that automates workflow execution on remote cloud infrastructure.

The remainder of this paper is organized as follows. Section 2 presents a problem statement using a use case from the ecology and earth science domain, outlining challenges and requirements, reviewing existing work on scaling scientific applications on distributed computing infrastructures, and summarizing limitations of current solutions. Section 3 details our method' s prototype design and implementation. Section 4 demonstrates the prototype through a massive LiDAR data processing example, discusses current limitations, and Section 5 concludes with future research directions.

2.1 Problem Statement

Consider an example from ecology and earth science where researchers monitor ecosystem structure changes over time or derive vegetation structure metrics to model animal distributions and habitat suitability. This requires processing raw data (3D point clouds) from country-wide airborne Light Detection and Ranging (LiDAR) datasets (~16TB) using the Laserfarm workflow [11] to generate LiDAR-derived metrics for ecosystem height, cover, and structural complexity.

Figure 1 [Figure 1: see original paper] illustrates two statistical metrics derived from LiDAR point clouds: the 95th percentile of normalized height (measuring ecosystem height) and pulse penetration ratio (measuring ecosystem cover). The 95th percentile quantifies the height of the tallest vegetation in a grid cell (e.g., trees in a forest patch), while the pulse penetration ratio describes vegetation openness as the ratio of ground points to total points within a grid cell. By measuring these across two time periods from country-wide LiDAR datasets, researchers can derive ecosystem structural change measures, extracting statistical properties from Airborne Laser Scanning (ALS) surveys typically stored in LAS/LAZ format.

Challenges. Researchers design algorithms and prototypes in Jupyter environments (e.g., Jupyter notebooks) and conduct experiments on local platforms or small clusters. These studies depend on appropriate algorithms but especially on large-scale data handling (e.g., multi-terabyte datasets). However, local compute and storage capacities often prove inefficient for large-scale scientific analysis, burdened by massive data inputs and the need for scalable computing. Additionally, a notebook' s pipeline steps or component modules can be represented as a workflow (Figure 2 [Figure 2: see original paper]). Some modules contain generic algorithms that could be reused across different codes and parallelized on remote infrastructures like cloud environments. Nevertheless, managing such workflows within Jupyter notebooks remains inadequate. While issues of (un)expected overheads or performance bottlenecks caused by data transfer between containerized components in NaaW fall outside this paper' s scope, we have considered this problem. When processing approximately 16TB of data, we currently employ a splitter module and merger module. The splitter partitions large data volumes into smaller chunks defined by users, eliminating large data transfers between containerized components. The merger module consolidates distributed processing results. This approach avoids frequent transmission of enormous data volumes between components while improving data processing parallelism.

2.2 Related Work

Many Jupyter environment architectures rely primarily on HPC settings and cloud environments [15, 16, 18, 19, 20, 21]. For instance, Milligan et al. [15, 16] implemented classic architectures integrating Jupyter with supercomputing resources to bridge exploratory data analysis and HPC, particularly leveraging Jupyter for interactive data-intensive supercomputing services. Their solutions provide HPC notebook services, a science portal for the GEMS argoinformatics data sharing and analysis platform, and BinderHub services. The HPC notebooks service at MSI (Minnesota Supercomputing Institute) enables users to seamlessly run interactive Jupyter notebook web applications using normal batch-scheduled cluster resources. BinderHub automatically launches Jupyter-ready Docker containers built by JupyterHub with repo2docker within Kubernetes clusters using public cloud resources. Similarly, Zonca et al. [19] pro-

posed three deployment strategies for scaling scientific applications on XSEDE resources at different scalability levels: JupyterHub on HPC via batch scheduler, JupyterHub on XSEDE Jetstream with Docker Swarm mode, and with Kubernetes. The primary difference between Docker Swarm and Kubernetes deployments is that the former provides notebooks with persistent storage and quotas, while the latter offers fault-tolerant JupyterHub deployment with elasticity.

Henderson et al. [18] proposed the NERSC (National Energy Research Scientific Computing Center) Jupyter infrastructure and JupyterHub deployment model, using notebooks as reusable curated recipes or applications (i.e., capable of running on different data or with varying inputs without copying or editing the notebook each time) to simplify execution processes. Although Henderson et al.' s work resembles NaaW in its potential for scaling different data inputs and running steps in parallel across multiple HPC nodes, it differs substantially. Their solution relies on NERSC HPC resources rather than cloud environments with on-demand resource provisioning and lacks flexibility for reusing critical components from notebooks as part of distributed workflows.

Yin et al. [17] proposed the CyberGIS-Jupyter framework for scalable geospatial analytics, adopting Jupyter notebooks instead of web GIS as the front-end interface to provide a consistent and agile playground for developers and users. It uses JupyterHub and Docker Swarm as the CyberGIS-Jupyter server, encapsulating CyberGIS capabilities within pre-configured containerized environments to achieve on-demand resource provisioning through OpenStack for elastic deployment and management of multiple virtual machine (VM) application instances. The hybrid ROGER computing environment integrating HPC and cloud resources provides underlying infrastructure support for reproducible deployment. Nevertheless, their approach has several limitations regarding scientific workflow representation. For instance, it does not emphasize pipelines or workflows, and the notebook functions as a whole job rather than a workflow-oriented application that can be packaged and submitted to the CyberGIS-Jupyter cloud environment.

Most existing solutions focus on notebook execution on pre-configured infrastructure (e.g., HPC clusters or cloud-provisioned VMs). These approaches lack explicit workflow management within notebooks; users control experiment steps via notebook cells, with workflow logic and data flows implicitly described by cell order, hampering reconfiguration at the cell level for different purposes. We aim to overcome these limitations by providing additional Jupyter extensions to extract notebook cells as reusable components, compose new logic, and automate execution on remote infrastructure, covering key workflow management steps.

3.1 Requirements

To address these challenges and gaps, we identify the following requirements for managing workflows in notebooks, forming the basis for our NaaW solution design.

The workflow management process—including decomposition of single notebooks containing code fragments and workflow composition—must remain flexible within the native Jupyter environment for scientific research. Users should not face restrictions in selecting code fragments or designing workflows for their experiments.

The workflow building blocks (reusable components from notebooks) must be interoperable with self-defined workflow logic. The tool should enable users to select these components to construct scientific pipelines while guaranteeing dependency constraints between components.

The approach must provide scalable solutions for large-scale scientific experimental analysis, particularly for large datasets or complex computational demands, and automatically execute workflows on remote infrastructures such as the cloud. The scientific experimental process must be scalable and parallel to resolve performance bottlenecks.

The solutions should embed within the current notebook ecosystem so scientists need not sacrifice the advantages of native Jupyter.

3.2 Proposed Jupyter Extensions

Based on these requirements, we propose four Jupyter extensions.

Component Containerizer creates reusable workflow building blocks from notebook cells. Technical details appear in our earlier paper [22]. This extension allows users to effectively select code cell(s) from Jupyter Notebooks and generate reusable workflow building blocks (REST services) in a WYSIWYG manner. It can also containerize these services as self-contained deployable containers (i.e., Docker) and store them in a local catalog or remote Docker Hub.

Experiment Manager enables users to compose workflows using containerized notebook code (created by the component containerizer). Users can define dependencies among different blocks and construct distributed workflows by visually connecting input and output parameters within metadata descriptions. This extension can save generated workflows as specification documents in YAML or CWL syntax. Figure 3 [Figure 3: see original paper] illustrates a conceptual diagram of the experiment manager usage.

Distributed Workflow Bus plans and schedules workflow deployment and execution on remote infrastructures. Automation of infrastructure service provisioning and deployment is provided by the remote infrastructure automator extension discussed next. This extension enables users to submit workflow specifications (created by the experiment manager) along with input data sources

and resource budgets (for cloud services). The workflow bus first invokes the remote infrastructure automator to initialize runtime infrastructure, then uses built-in scheduling algorithms to schedule workflow execution, as shown in Figure 4 [Figure 4: see original paper]. Currently, this function builds upon the Argo workflow engine, enabling developers to select built-in scheduler policies or add self-defined scheduling strategies.

Remote Infrastructure Automator plans cloud infrastructure capacity and automates resource provisioning and service deployment. This extension extends our earlier Dynamic Real-time Infrastructure Planner work [24].

3.3 How They Work Together

The four key components install as Jupyter extensions, as shown in Figure 5 [Figure 5: see original paper]. The grey boxes represent the four components —component containerizer, experiment manager, distributed workflow bus, and remote infrastructure automator—along with the catalog containing workflow building blocks and the dedicated remote infrastructure for scalable experiments.

Users prototype scientific pipeline steps in Jupyter environments (e.g., Notebooks) to conduct small-scale experiments on local computers or small clusters. Using the native Jupyter notebook front-end, users can employ the component containerizer to encapsulate a cell as a REST service and containerize it (step 1), store containerized components in a local catalog (step 2), and make further modifications (step 3). Using the experiment manager extension, users design new workflow experiments (step 4) by selecting containerized components from the local catalog (step 5) and creating workflow representations (step 6). The distributed workflow bus executes the workflow description (step 7) by first initializing virtual infrastructures (e.g., VMs or Kubernetes clusters) from providers (step 9) via the remote infrastructure automator extension. Runtime status and results can be monitored through a dashboard (step 10).

4. System Prototype and Demonstration

As previously discussed, Figure 2 presents a sample Laserchicken software workflow from the ecology domain, comprising six functional modules: load, normalize, filter, compute neighbors, extract features, and export. Each module contains different code fragments with various input and output parameters. In practice, the entire process must run sequentially within the notebook' s rigid cell structure, severely impacting experimental performance and scalability by reducing parallelism and code fragment reusability. To validate our method, we demonstrate key workflow-oriented management components in a Jupyter notebook environment using this use case.

Figure 6 [Figure 6: see original paper] shows the component containerizer interface alongside a Jupyter notebook. The notebook contains scientific pipeline

steps with different component functions. Users can select code fragments to encapsulate as components. The left bar automatically inspects corresponding metadata (e.g., inputs, outputs, parameters, package dependencies) for each component encapsulation. As workflow building blocks, these can be added to the catalog for future use.

Figure 7 [Figure 7: see original paper] illustrates the experiment manager function. The local catalog contains encapsulated component metadata. Users can select different components as reusable workflow building blocks to compose self-defined workflow logic (e.g., tasks and dependencies) and export the workflow. The experiment manager's desirable output is a generated workflow specification document, which serves as input for the distributed workflow bus module.

As shown in Figure 8 [Figure 8: see original paper], the distributed workflow bus currently implements the Argo workflow engine for Kubernetes. It uses Argo' s default scheduler for workflow planning and scheduling, while underlying infrastructure such as virtual machines is still manually allocated for launching the Kubernetes platform.

Figure 9 [Figure 9: see original paper] displays the remote infrastructure automator interface (also called Cloud-cells). This Jupyter notebook extension allows users to deploy Docker containers generated by Cloud-Cells to the cloud, though it has not yet been fully integrated into the NaaW method.

4.1 Discussion

This paper discusses the key components of the Notebook-as-a-Workflow (NaaW) solution. The four components are prototyped as extensions embedded within scientists' Jupyter working environments.

This solution extends our previous work [22, 24] into a workflow-oriented management approach that bridges the gap between Jupyter environments and large-scale workflow management. Demonstrated through a LiDAR processing example from ecology and earth science, the proposed solution achieves the workflow management process outlined in our requirements. Using these Jupyter extensions, scientists can: (1) interactively encapsulate code fragments (one or more notebook cells) as reusable workflow components; (2) compose workflows using these components and customize execution logic based on data volume and location; (3) automate cloud infrastructure provisioning based on workflow requirements (data volume and resource budgets); and (4) interactively execute composed workflows on cloud infrastructure to achieve required scale.

This paper focuses on key workflow management steps: component containerization from notebooks, workflow composition, infrastructure automation, and workflow execution. Several features remain unimplemented or require improvement. For instance, the experiment manager cannot currently verify dependency correctness between workflow building blocks and requires inspection before users save generated workflow specifications and submit them to the distributed

workflow bus. Additionally, the distributed workflow bus design and implementation remains immature, as it still depends on third-party solutions like the Argo workflow engine. Many workflow scheduling algorithms require further development; we currently use Argo' s default scheduler for deploying and executing submitted workflows. The integration between distributed workflow bus and remote infrastructure automator does not yet operate seamlessly. Although both modules function as Jupyter extensions within the Jupyter environment, some steps must still be completed manually.

5. Conclusion and Future Work

This work focuses on managing workflows within Jupyter notebook architectures. We propose four core components to achieve this goal: component containerizer, experiment manager, distributed workflow bus, and remote infrastructure automator. Our solution (1) encapsulates reusable cells as RESTful services and containerizes them as portal components; (2) provides a composition tool for describing workflow logic of these reusable components; and (3) automates execution on remote cloud infrastructure through the distributed workflow bus and remote infrastructure automator. We validate solution usability through a LiDAR use case from ecology and earth science. This work remains under active development and continuous improvement. The missing features discussed will be prioritized in our next development steps.

Acknowledgements

This work has been partially funded by the European Union' s Horizon 2020 research and innovation programme through the CLARIFY project under Marie Sklodowska-Curie grant agreement No 860627, the ARTICONF project grant agreement No 825134, the ENVRI-FAIR project grant agreement No 824068, the BLUECLOUD project grant agreement No 862409, and by LifeWatch ERIC.

Author Contributions

Y.D. Wang (y.wang8@uva.nl): Conceptualization, Investigation, Writing—original draft, Writing—reviewing & editing.
S. Koulouzis (S.Koulouzis@uva.nl): Conceptualization, Software development and maintenance.
Riccardo Bianchi (r.bianchi@uva.nl): Conceptualization, Software development and maintenance.
N. Li (n.li@uva.nl): Discussion.
Y.F. Shi (y.shi@uva.nl): Writing—review & editing, Visualization.
J. Timmermans (j.timmermans@uva.nl): Conceptualization, Writing—original draft, Writing—review & editing.
W.D. Kissling (wdkissling@gmail.com): Conceptualization, Writing—original draft, Writing—review & editing, Visualization, Funding acquisition.
Z.M. Zhao (Z.Zhao@uva.nl): Conceptualization, Writing—original draft, Writ-

ing—review & editing, Supervision, Project administration, Funding acquisition, Coordinated the technical development of the system.

References

[1] Vermeulen, A., et al.: Supporting cross-domain system-level environmental and earth science. In: Zhao, Z., Hellström, M. (eds.) Towards Interoperable Research Infrastructures for Environmental and Earth Sciences, pp. 3–16. Springer, Cham (2020)
[2] Tuli, S.: Next generation technologies for smart healthcare: Challenges, vision, model, trends and future directions. Internet Technololy Letters 3(2), e145 (2020)
[3] Tudoran, R., et al.: Adaptive file management for scientific workflows on the Azure cloud. In: 2013 IEEE International Conference on Big Data, pp. 273–281 (2013)
Jupyter. Available at: https://jupyter.org/. Accessed 11 March 2020
Jupyter hub. Available at: https://jupyter.org/hub. Accessed 11 March 2020
[6] Perkel, J.M.: Why Jupyter is data scientists' computational notebook of choice. Nature 563(7729), 145–146 (2018)
[7] Henderson, M. L., et al.: Accelerating experimental science using Jupyter and NERSC HPC. In: Juckeland, G., Chandrasekaran, S. (eds.) Tools and Techniques for High Performance Computing, pp. 145–163. Springer International Publishing, Cham (2020)
LifeWatch virtual lab and innovation center. Available at: https://www.lifewatch.eu/vlabs-innovations-centre. Accessed 11 March 2020
[9] The EU ENVRI-FAIR project. Available at: http://www.envriplus.eu/. Accessed 11 March 2020
[10] Meijer, C., et al.: Laserchicken—A tool for distributed feature calculation from massive LiDAR point cloud datasets. SoftwareX 12, 100626 (2020)
[11] Laserfarm. Available at: https://github.com/eEcoLiDAR/Laserfarm/. Accessed 11 March 2020
[12] BioExcel. Available at: https://mmb.irbbarcelona.org/biobb/. Accessed 13 September 2021
[13] Deep histopath. Available at: https://github.com/CODAIT/deep-histopath. Accessed 13 September 2021
[14] Building medical 3D image segmentation using Jupyter notebooks from the NGC catalog. Available at: https://developer.nvidia.com/blog/building-medical-3d-image-segmentation-using-jupyter-notebooks-from-the-ngc-catalog/. Accessed 13 September 2021
[15] Milligan, M.: Interactive HPC gateways with Jupyter and Jupyterhub. In: PEARC17: Proceedings of the Practice and Experience in Advanced Research Computing 2017 on Sustainability, Success and Impact, Article No. 63 (2017)
[16] Milligan, M.B.: Jupyter as common technology platform for interactive HPC services. In: PEARC ' 18: Proceedings of the Practice and Experience on Advanced Research Computing, Article No. 17 (2018)
[17] Yin, D., et al.: CyberGIS-Jupyter for reproducible and scalable geospatial

analytics. Concurrency Computation 31(11), e5040 (2019)
[18] Henderson, M.L., et al.: Accelerating experimental science using Jupyter and NERSC HPC. In: HUST 2019, SE-HER 2019, WIHPC 2019: Tools and Techniques for High Performance Computing, pp. 145–163 (2019)
[19] Zonca, A., Sinkovits, R.S.: Deploying Jupyter notebooks at scale onXSEDE resources for science gateways and workshops. arXiv preprint arXiv:1805.04781 (2018)
[20] Stubbs, J., et al.: Integrating Jupyter into research computing ecosystems. In: PEARC ' 20: Practice and Experience in Advanced Research Computing, pp. 91–98 (2020)
[21] Zhou, S., Liu, X., Lin, L.: Design and implementation of Python teaching platform based on container and Jupyter. In: ACM International Conference Proceeding Series, pp. 446–450 (2020)
[22] Kruijer, W., et al.: FAIR-Cells: An interactive tool for enabling the FAIRness of code fragments in Jupyter notebooks. In: The proceedings of International Conference of High Performance Computing and Simulation (HPCS) (2020). (To appear)
[23] Cloud-Cells. Available at: https://github.com/QCDIS/Cloud-Cells. Accessed 26 June 2021
[24] Koulouzis, S., et al.: Time-critical data management in clouds: Challenges and a dynamic real-time infrastructure planner (DRIP) solution. Concurrency and Computatation Practice and Experience 32(16), e5269 (2019)
[25] The default workflow of Laserchicken. Available at: https://laserchicken.readthedocs.io/en/latest/. Accessed 27 June 2021

Author Biography

Yuandou Wang is currently a Ph.D. student at the University of Amsterdam, the Netherlands. Her research interests include cloud computing, workflow scheduling, and decentralized workflow management. ORCID: 0000-0003-4694-9572

Spiros Koulouzis obtained his Ph.D. from the University of Amsterdam, the Netherlands. Currently, he is a research associate at the LifeWatch VLIC. His research interests include distributed and parallel systems, workflow systems, and scientific data management. ORCID: 0000-0001-8652-315X

Riccardo Bianchi obtained his joint M.Sc. in Big Data Engineering from the University of Amsterdam (UvA) in 2020. Currently, he is a research associate at the LifeWatch VLIC. His research interests include distributed and parallel systems, workflow systems, and scientific data management. ORCID: 0000-0002-8550-3824

Na Li is currently a Ph.D. student in the Informatics Institute at the University of Amsterdam, the Netherlands. Her research interests include research asset discovery, dataset search, and data quality assessment. ORCID: 0000-0001-7799-876X

Yifang Shi is a scientific developer for ecological applications of LiDAR Remote Sensing at LifeWatch-ERIC and the University of Amsterdam. Her research interest is developing scientific workflows for ecological applications in biodiversity and ecosystem science using airborne and spaceborne LiDAR. ORCID: 0000-0001-8770-8906

Joris Timmermans is Scientific Developer at the Lifewatch European infrastructure (ERIC), Virtual Laboratory Innovation Center (VLIC), and a guest researcher at the University of Amsterdam (UvA) and Leiden University (LU), the Netherlands. He currently works on developing essential biodiversity variables from remote sensing, combining multi-sensor monitoring, data-intensive fusion techniques, and ecology to track policy targets set for 2030. ORCID: 0000-0002-6359-8092

W. Daniel Kissling is Associate Professor at the University of Amsterdam (UvA), the Netherlands. His research interest is data-intensive biodiversity science, ecology, and global biodiversity change, using large datasets, ecoinformatic tools, digital sensors, and remote sensing. ORCID: 0000-0002-7274-6755

Zhiming Zhao received his Ph.D. degree in Computer Science in 2004 from the University of Amsterdam (UvA). He is an assistant professor in Multiscale Network Systems (MNS) at UvA and the technical manager in the Virtual Lab and Innovation Center (VLIC) of LifeWatch ERIC, a European research infrastructure in ecology and biodiversity science. His research focuses on data management, cloud computing, trustworthy service level agreement, and blockchain. He leads the UvA effort in several EU projects, including ARTICONF, CLARIFY, ENVRI-FAIR, and SWITCH projects. (https://staff.fnwi.uva.nl/z.zhao/) ORCID: 0000-0002-6717-9418

Note: Figure translations are in progress. See original paper for figures.