Three Pillars of Practical Reproducibility

Kate Keahey*, Jason Anderson†, Mark Powers†, Adam Cooper†
* *Argonne National Laboratory*
Lemont, United States of America
Email: keahey@mcs.anl.gov
† *University of Chicago*
Chicago, United States of America

Abstract—**Practical reproducibility is the ability to reproduce results is a manner that is cost-effective enough to become a vehicle of mainstream scientific exploration. Since computational research artifacts usually require some form of computing to interpret, open and programmable infrastructure, such as a range of NSF-supported testbeds spanning infrastructure from datacenter through networks to wireless systems, is a necessary – but not sufficient – requirement for reproducibility. The question arises what other services and tools should build on the availability of such programmable infrastructure to foster the development and sharing of findable, accessible, integrated, and reusable (FAIR) experiments that underpin practical reproducibility. In this paper, we propose three such services addressing the problems of packaging for reuse, findability, and accessibility, respectively. We describe how we developed these services in Chameleon, an NSF-funded testbed for computer science research which has supported the research of a community of 8,000+ users, and discuss their strengths and limitations.**

Index Terms—**reproducibility, infrastructure, scientific platforms, resource management**

I. Introduction

There is a broad agreement that in the digital age science should be shared digitally, through artifacts such as code and data, and that adopting practices enabling or facilitating reproducibility of computational results can lead to more robust science and increased scientific productivity [1]–[4]. To support this consensus, the computer science community supports activities such as artifact evaluation at conferences, data preservation initiatives, and reproducibility hackathons [5], [6], and sponsors incentives such as reproducibility badges awarded by major computing organizations [7], [8]. But despite all these efforts it is not clear how much science gets reproduced beyond those targeted initiatives, as part of individual quest for knowledge. We argue that unless reproducing research becomes as vital and mainstream part of scientific exploration as reading papers is today, reproducibility will be hard to sustain in the long term because the incentives to make research results reproducible won't outweigh the still considerable costs of making them so. Thus, in addition to seeking ways to ensure that every experiment can be repeated regardless of the effort, we should also explore mechanisms for *practical reproducibility*, i.e., a practice where many – or even most – experiments or results are packaged in such a way that they can be repeated cost-effectively.

In a research ecosystem that supports practical reproducibility, scientists could "click through" results presented in a paper, reproduce the experiment that yielded them or redo the data analysis to try it on different hardware, introduce variation into the algorithm and provide a side-by-side comparison, or answer new questions with the available data. This mode of exploration means that rather than just reading about new research, a scientist is exploring science interactively and can immediately verify or challenge new results, extend them on-the-fly by inserting new ideas, or more easily integrate new research into teaching, thus accelerating evolution of curricula to keep up with advancing science. Furthermore, going beyond sharing results through papers, scientists could access experiments – representing electronic packaging of results – directly through a digital hub, similarly to how they might look for relevant papers in the ACM Digital Library [9] today. Such direct access to reproducible experiments could be an effective method not just for sharing results themselves, but also for exploring the means via which they were obtained, i.e., the experimental methodology, or ways in which the experimental data were analyzed – placing more emphasis and stimulating more discussion of different aspects of scientific exploration. This kind of dissemination of research has the potential to invert the role of papers and results and make the papers indeed "an advertisement for scholarship" [10].

Such research ecosystem may be more readily within reach than it currently appears, as its underpinnings are already present. In [11] we note the enabling power of two factors. The first one is the availability of resources through open platforms or clouds with unique hardware and configurations on which research can be readily re-played. An open platform means that all investigators have equal access to the same experimental hardware, no matter how rare or expensive; thus, it is no longer the case that "I can do my research because I have a GPU cluster – but you can't reproduce it, because you don't". The second factor is the platform's programmability, i.e., the ability to establish an arbitrarily complex experimental environment in programmatic ways, that can be saved and then replayed many times. To support computer science experimentation in particular, the National Science Foundation (NSF) has created a collection of open, programmable experimental platforms – Chameleon, CloudLab, FABRIC, COSMOS, POWDER, AERPAW, and ARA [12]–[18]. Supplemented by commercial cloud resources openly available via NSF-funded CloudBank [19], these platforms can collectively support most computer science experiments ranging from topics in performance variability, operating systems, or networking, to machine learning, artificial intelligence, and robotics research.

However, while open platforms and programmability create an opportunity, they don't offer a complete solution. To illustrate: over the years of operating Chameleon [12] – a bare metal reconfigurable platform which to date has supported a community of over 8,000 users working on over 1,000 research and education projects – we noted that experimenters using the testbed created thousands of digital artifacts. Those artifacts – images, orchestration templates, and digital notebooks – were created entirely as a side-effect of using the platform, i.e., with no special effort made towards reproducibility. At the same time, they represent experimental environments that supported research on Chameleon, and could be used to re-establish such environments – often the most complex step in reproducing a computer science experiment. This is a treasure trove of digital content – most of it publicly available – that should in principle significantly improve potential for interactive science – but it does not. There are several reasons for this: those artifacts represents only part of an experiment and are not much use without the other parts; despite being publicly available they can be very hard to find; there are no incentives to keep them current; potential "consumers" of this content simply lack access to hardware, content, or both. To remedy this situation, we sought to create services, tools, and processes that would help our users make their research more shareable; the proposals shared below are a result of these efforts.

In this paper, we make the case that in order to support practical reproducibility we need to support findable, accessible, integrated, and reusable (FAIR) experiments, represented as a combination of hardware, experimental environment, experiment body, and data analysis components. Further, we also argue, that no one tool, but an ecosystem of integrated tools is needed to create progress in providing a practical reproducibility platform, and propose three infrastructure-related capabilities fundamental to the support of practical reproducibility: (1) an effective method of packaging for reuse that associates access to hardware with the digital representation of an experiment (a "compute capsule"); in as much as possible such method should be a side-effect of developing an experiment, (2) providing ways of sharing and finding packaged experiments integrated with platforms on which they can be executed, and (3) supporting access to platforms specifically for reproducibility purposes, potentially outside of regular policies. Our viewpoint is naturally informed by the experience of operating a testbed; we therefore present specific proposals for these integrated tools, explain how interacting with our community shaped our development decisions, and explain how they were implemented in Chameleon.

II. Experiments and Experimenters

As operators of Chameleon [12], an NSF-funded experimental infrastructure for computer science research , our perspective is informed primarily by the need of computer science experiments; therefore the needs of those experiments specifically define the scope of this paper. This means that we explore many particularly challenging computational scenarios – but at the same time may de-emphasize common experimental patterns and needs present in other sciences.

In general, these types of experiments can usually be seen as composed of three parts: (1) the creation of *experimental environment* or topology: the allocation, configuration, and orchestration of resources in which the experiment will execute (e.g., "create Linux cluster with a distributed storage system"), (2) *experiment body*, i.e., the actual execution of experimental actions (e.g., benchmarking the created environment), and (3) *data analysis* and presentation. Experiments may emphasize or de-emphasize some of those stages; for example, they may skip the creation of an experimental environment if e.g., the objective is to discover new properties of existing environments (e.g., a specific datacenter configuration). Further, reproducing the experiment may involve reproducing all of those stages, or only one of them: for example, a reviewer may be interested in repeating the data analysis without replaying the whole experiment. Similarly, variation can also be introduced at different stages, by e.g., re-running the same experiment in different experimental environments.

Reproducing experiments is primarily a dialogue between *experiment authors*, who create experiments and may package them for reproducibility, and *experiment reviewers*, who attempt to reproduce those experiments. We note that scientists taking on those different roles have different perspectives and motivations, and argue that to provide effective support for reproducibility we need to develop infrastructure, services, and tools that take those motivations into account and align them to the extent possible. We discuss these factors in explaining our design decisions and explain how they influenced our approach.

III. Packaging for Integration and Reuse

From the perspective of an experiment reviewer, the completeness of packaging (i.e., availability of all the preconditions for reproduction) is of course the necessary condition for reproducibility of an experiment – but it is not sufficient from the perspective of practical reproducibility: the time to reproduce could still be prohibitive or uncertain. This factor led to the creation of tools streamlining the establishment of experimental environments – often the most complex stage of an experiment setup – so that they can be re-run as much as possible "with one click" [20]–[22]. Many such efforts focus on declarative packaging, i.e., describing a desired state/outcome rather than ways to achieve it, expressed via orchestration templates or experiment profiles [13], [23]–[25]. In practice this is often impractical as the same state can be achieved via different means – that are associated with different interpretations of the desired state, or different side-effects – and thus not always yielding consistent results. In addition, this approach is by nature transactional (i.e., the state is not reached until the orchestration transaction is finished); this makes it difficult to inject variation to an experiment or handle natural variation in the platform (e.g., temporary unavailability of some resources). We find that for this reason experiment reviewers prefer experiments expressed in imperative and non-transactional style, where the reviewer can build up to the desired experimental environment state gradually, addressing issues or trying new approaches as they go. Last but not least, a critical requirement for the experiment reviewer is that all the artifacts pertaining to experimentation – images, deployment configuration, experiment code, analysis, as well as the argument that it supports – are connected and integrated in a way that allows the reviewer to not only grasp the relationships between them, but manage those relationships easily (e.g., deploy the same image on a different hardware configuration), and follow the way the experiment supports the argument with which it is associated.

These considerations are of less direct importance to the experiment author, who may care about them as a way of ultimately achieving impact, but whose direct attention is more likely to be focused on the cost of packaging an experiment – in particular, as the time spent on it diverts resources from new research. In other words, from the author's perspective it is critical that packaging an experiment is as cheap as possible, preferably a side-effect of experimentation itself – and with cost of reproduction being only a secondary consideration.

We explored two approaches of packaging experiments as a side-effect of experimentation: (1) monitoring the events an experiment generates on the testbed and then using programmatic methods to reconstruct those events and the resulting experimental environment, and (2) aligning experimental methodology and reproducibility so that the most convenient tools to package experiments happen to be the same ones as to develop them in the first place. We explored the former in [26] where we instrumented Chameleon to monitor user's actions; presented a summary of those actions to the user (experiment precis); and then attempted to generate an orchestration template to reproduce the same state. The results were mixed: while it was possible to get good results in relatively simple cases, it was hard to ensure correctness in the general case. Given however that many of our users already used a programmatic approach to creating complex experimental environments, either via imperative or declarative methods (CLI/python-chi [27] or Heat templates [23], respectively), we turned our attention to adapting them to packaging experiments. Specifically, we looked for an imperative, non-transactional approach that would also provide an integration of the various experimental artifacts, experiment components, and tie the experiment to analysis and results presented in the paper. We noticed that our users increasingly use Jupyter to program different stages of their experiments, and leverage the combination of programmability of the bash and python kernels with analysis expressed in text and images. However, from the perspective of complex experimentation Jupyter had one shortcoming: the code executed in a Docker container – not sufficient to users who typically need complex and distributed experimental environments. To remedy this, we decided to provide a Jupyter interface to the testbed so that users could use Jupyter to create experimental environments as well as use it for the experiment body and data analysis.

The resulting Jupyter interface to Chameleon allows users to establish a secure session to the testbed by logging in through Chameleon's JupyterHub. From the user's perspective, this has a few advantages: the programmatic interface to the testbed (now through the bash and python Jupyter kernels) becomes much richer by leveraging the integration of text, programs, and visualization elements supported by Jupyter; the user's credentials are implicit in Jupyter cells allowing the user direct access to the testbed for programmatic creation of complex experimental environments, more powerful than the Docker containers that typically serve as a back-end for Jupyter computation; and since there is no explicit authentication step, there is no need for the notebook to include secrets so that it can be shared more easily. Further, in order to support programming the body of the experiment (as well as the experimental container) from Jupyter we have configured a Chameleon image with a bare bones JupyterLab server; users can deploy it, and use it to program the body of their experiment as well as the resulting analysis.

To implement this integration we used the "Zero to JupyterHub on Kubernetes" deployment of JupyterHub [28]; it spawns per-user instances of JupyterLab with minimal allocation of resources, as its intended usage is for orchestration of testbed resources rather than direct computation. The JupyterLab environment comes with OpenStack [29] command line interface (CLI) (the CHameleon Infrastructure (CHI) builds on top of OpenStack) and python software development kits (SDK)s installed, along with Chameleon's python SDK (python-chi). To create a seamless interface, we authenticate JupyterHub users to our central OAuth Keycloak server [30] and then use the OAuth token granted by the JupyterHub authenticator to authenticate the user to all Chameleon sites. This token, along with other relevant user information, is loaded into environment variables when the user spawns a JupyterLab instance, and is automatically read by OpenStack's CLI and SDK as well as python-chi. We have implemented a periodic check for OAuth token expiration, which refreshes the user's session in the background.

To facilitate the transition between the creation of an experimental environment and the execution of the experiment body inside the JupyterLab environment, we also create an SSH key for the user; this key is placed on all OpenStack instances created with python-chi. This allows users to connect to any resources they deploy and run commands over SSH to upload data and execute experiments on the deployed instances (in a sense, any experiment body command becomes "wrapped" by SSH). Some users prefer to execute the experiment body and/or the analysis part of the experiment on remote resources directly from a Jupyter notebook. To support this, we provide methods to install and run a bare bones Jupyter server on the remote instance via SSH and create a tunnel from the user's local machine to this new Jupyter server; this allows the user to execute the body of the experiment by running the kernels directly on the created cloud instances. Neither method provides a completely seamless experience and devising better ways of relating experimental containers to experiment body and analysis is one of the open questions of an ecosystem for reproducibility.

IV. Findable via Trovi

The principal challenge of sharing digital artifacts rests in the nature of their readability: we are all well-equipped to read papers and require no additional infrastructure to interpret them, whereas data, code, and other digital artifacts generally require computation, visualization or other means of digital interpretation. Support requirements for a digital artifacts repository are therefore analogous to a library that has a microfilm collection: in addition the artifacts themselves, such library might also provide a microfilm reader as a means of interpretation. Similarly, having a digital artifact, even one that is complete, integrated, otherwise ready to reproduce, is but half the battle. The lack of hardware to reproduce it on – especially in computer science where so many experiments rely on novel, specialized, or even simply just sufficiently powerful hardware – often means that reproducing an experiment in practice is impossible. Platforms like Google Collab [31] and CodeOcean [32] recognize this and associate hardware with computation, but have significant limitations: they are restricted to a specific platform or a small set of platforms, have ceilings on use, and a relatively narrow range of capabilities, generally limited to execution on a single node rather than complex experimental environments.

We argue that to provide an effective hub for executable experiments, a service should provide open APIs that would enable integration with multiple open testbeds/clouds and thus support many different types of experiments on multiple platforms. Further, a sharing hub of this type should also be well positioned to propose measures of impact, a critical incentive that creates a bridge between the interests of the experiment authors and reviewers. This can be done by connecting to the existing publishing and incentive structures, e.g., making digital artifacts citable via assigning them digital object identifiers (DOIs), or by providing its own by supporting e.g., metrics of how much a specific artifact was executed. In addition, a service of this kind should of course also provide all the features that allow users to effectively index and discover experiments.

We implemented a service, called Trovi (from Esperanto "find"), that allows users to share artifacts packaged as Jupyter notebooks integrated with open platforms. Trovi is not a part of Chameleon, but rather a general-purpose service that is designed to ulitmately connect to many platforms via open API (see below). An experiment author can add an artifact to Trovi for public availability or limited sharing and index it by using appropriate keywords. Once the artifact is in Trovi, the author can create new versions or otherwise edit the artifact metadata. Users can create versions by uploading file archives or by importing from a supported storage backend, such as a public Git repository. Once an experiment is ready for publication, it can be published to Zenodo [33] which assigns it a DOI allowing the user to reference the experiment as a first-class entity. Experiment reviewers search for relevant Trovi artifacts, and once they find experiments of interest, they can inspect them and interact with them by clicking on a launch button that supports seamless execution of the associated notebook on an integrated platform.

To support this last feature effectively, Trovi exposes an open API [34] allowing the system to support different back-end implementations (experiment storage), connect to different platforms, and support different front-end implementations (experiment presentation and search). The back-end API, implemented in Chameleon Swift [35], Zenodo [33], and GitHub [36], fetches artifact retrieval metadata (rather than content itself), and then uses implementation-specific ways of fetching content. For example, an artifact version stored in Git provides information about the Git remote URL and protocol, while one stored in Chameleon's object store will provide a temporary HTTP archive URL.

This content retrieval data is passed into a JupyterLab "startup hook" (i.e., an import handler that we implemented with JupyterLab and JupyterHub extensions) which, when run, fetches the artifact's contents and loads them into the user's working directory. This allows a user to click on the "Launch" button and be taken to a Jupyter environment with the artifact's contents loaded into the working directory. To connect to different platforms, Trovi's API effectively exchanges a valid OAuth token from an approved platform, such as Chameleon, for a Trovi bearer token, which must be included with testbed API calls which require authentication; this allows a user to use multiple platforms from the same notebook (via platform-specific interfaces).

Lastly, the front-end API represents an implementation that allows users to view and manage artifacts represented as Jupyter notebooks. This front-end implementation is currently surfaced as a web page in the Chameleon portal for convenience, though we anticipate it to be eventually replaced by an independent implementation. Each Jupyter notebook has an integrated "Launch" button supported by the JupyterHub extension described above. This extension also reports when a user executes a cell from an artifact – a functionality that allows us to keep track of the number of times an artifact was executed at least partially (we also keep track of artifact views). While this is not a perfect measure of impact, it is a useful approximation, as it helps align the incentives of experiment authors with those of experiment reviewers.

V. Accessible via Chameleon Daypass

Even open platforms typically place some restrictions on their usage; for example, an open research platform might disallow commercial use. Further, platform use is often associated with some kind of allocation, or explicit payment in the case of commercial clouds – resources that an experiment reviewer may be unwilling to commit. These factors create a potential barrier to practical reproducibility on the experiment reviewer side: even experiments that are packaged for reuse, integrated, and findable may not be accessible.

One proposal to overcome this challenge is to give experiment authors limited allocation to explicitly support reproducibility of their experiments: we piloted this capability in Chameleon by implementing Chameleon Daypass. Experiment authors can request an allocation for 10 experiment executions of 1000 allocation units (rough default numbers determined by the requirements of the current pilot) by providing a brief justification. Once the allocation is granted, the author's experiment in Trovi exposes an additional interface allowing users to request a reproducibility allocation for this specific experiment – and the author can now advertise their experiment as available via links associated with experiment description in the paper, QR code on a poster, or otherwise linking traditional and digital artifacts.

When an experiment reviewer requests a reproducibility allocation, the request sends an email notification to the experiment author who reviews and potentially grants the request. Once the request is granted, the system generates another email, notifying the experiment reviewer that their Daypass was approved, and inviting the user to join the project by clicking on a link. When the user is added to the daypass allocation, the system notes the time the invitation was accepted. A periodic task checks all active reproducibility allocations: if any accepted daypass invitations exceed the reproducibility allocation time specified by the experiment author, we automatically remove the user from the allocation. The current implementation keeps track of invitations and Daypass allocations in Chameleon's portal and thus its enforcement is reliant entirely on allocation limits.

VI. Discussion and Open Challenges

In this paper, we argue that a *compute capsule* combining experiment enactment with hardware on which it can be executed is necessary – but not sufficient – to achieving practical reproducibility for computer science experiments. A key component removing friction from reproducing experiments is *how* these elements are combined, i.e., how integrated, findable, and accessible experiemnts are in practice. Experiences gained in operating Chameleon indicate that not one service but an ecosystem of services each supporting a specific aspect of FAIR experimentation is needed to achieve a "critical mass" of reproducibility.

Our approach to packaging fulfills the critical requirement in that it ties hardware available via an open platform to all three elements of an experiment: environment creation, experiment body, and data analysis. This allows experiment authors to ensure that experiment reviewers will find the necessary resources to reproduce experiments. We encourage the use of Jupyter notebooks, since many of our users use them to structure experiments anyway; thus, a representation of an experiment viable for practical reproducibility is effectively created by side-effect. By integrating them with Chameleon we allow users to extend their use over all three elements of experiment described in Section II, from experimental environment creation, to experiment body, and analysis. We note however, that currently the transition between the creation of an experimental environment and experiment body and analysis is a challenge; we propose two methods of overcoming it in our approach, but more seamless transitions can be imagined. Structuring the relationship of an experimental environment to experiment body and analysis stages in general is an open problem and depends on how those environments can be created.

Further, our approach to sharing experiments fulfills its mission in that it further ties findable experiments to open platforms on which they can be executed. While Jupyter is a promising approach, different types of packaging may be appropriate in different contexts or for different problems; although our current implementation focuses on Jupyter, an abstraction of experiment representation that could be executed on different platforms would therefore make the system more general. In addition, a closer integration of the system with existing ecosystems of tools that already provide versioning and serve as a repository of useful digital artifacts related to experimentation (e.g., GitHub or Zenodo) would also improve user experience.

The biggest set of open challenges in developing an ecosystem for practical reproducibility however deals with community interaction with the process. A significant issue in this space is that of incentives as they create an alignment between the interests of experiment authors and reviewers: while experiment authors themselves may not be interested in ease of replication directly, they will become interested if we can tie it to the impact of their research. From this perspective, one-time acknowledgement such as reproducibility badges are limited as they do not provide the kind of differentiated recognition that is available via e.g., citation counts (themselves an imperfect measure). Our approach to addressing this problem is to provide metrics of Jupyter notebook views and execution counts (of at least one code cell) on viable platforms for a given experiment; while not a perfect indicator, it does give an idea of an artifact's popularity. Another potential approach is the integration of ongoing open reviews and comments into the system – translating them into feedback is less instantaneous than a numerical metric but could provide a good alternative to use in conjunction. A related issue is that of sustaining the artifacts: software and thus experiments needs upgrades and updates (e.g., to keep up with security vulnerabilities), and thus the ability to maintain a FAIR experiment in some form. Experiment authors are not always willing or able to support the experiments; one possible way to resolve this problem is via community collaboration where interesting experiments could be "forked" as reviewers find ways to upgrade or otherwise fix them, potentially borrowing policies and processes from the open source community. This in itself could provide a measure of impact, as maintained (and therefore clearly used) experiments would likely see enough attention to keep them current.

Lastly, our pilot solution for accessibility provides limited enforcement and relies strongly on the networking of researchers sharing their experiments with others. This suggests that some form of social network mechanisms could play a useful role in both popularizing reproducibility and providing a better incentive structure.

VII. Conclusions

Effectively supporting reproducibility in the digital age requires rethinking and refining the current research sharing ecosystem. Since computational research artifacts require some form of computing – sometimes unique and rare – to interpret, we argue that open infrastructure will play key role in any such ecosystem. Further, we posit that an effective sharing ecosystem will support not only reproducibility, but practical reproducibility, where many artifacts will be not only reproducible, but reproducible in a cost-effective manner so that computational reproducibility can become a mainstream vehicle of research sharing.

In this paper, we summarize our experiences of encouraging reproducibility on the Chameleon testbed for computer science research and propose three services that in our experience are fundamental to the support of such ecosystem: packaging experiments with the hardware on which they can be executing (i.e., "compute capsules"), ways of sharing experiments that tie them to the hardware, and explicitly supporting infrastructure access for reproducibility purposes. We describe how we developed these services in Chameleon and discuss their strengths and limitations.

Acknowledgement

Results presented in this paper were obtained using the Chameleon testbed supported by the National Science Foundation. This material is based upon work supported by the U.S. Department of Energy, Office of Science, under contract number DE-AC02-06CH11357.

References

[1] O. E. Gundersen, Y. Gil, and D. W. Aha, "On reproducible ai: Towards reproducible research, open science, and digital scholarship in ai publications," *AI magazine*, vol. 39, no. 3, pp. 56–68, 2018.

[2] M. S. Krafczyk, A. Shi, A. Bhaskar, D. Marinov, and V. Stodden, "Learning from reproducing computational results: introducing three principles and the reproduction package," *Philosophical Transactions of the Royal Society A*, vol. 379, no. 2197, p. 20200069, 2021.

[3] D. Rosendo, P. Silva, M. Simonin, A. Costan, and G. Antoniu, "E2clab: Exploring the computing continuum through repeatable, replicable and reproducible edge-to-cloud experiments," in *2020 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2020, pp. 176–186.

[4] V. Stodden, M. McNutt, D. H. Bailey, E. Deelman, Y. Gil, B. Hanson, M. A. Heroux, J. P. Ioannidis, and M. Taufer, "Enhancing reproducibility for computational methods," *Science*, vol. 354, no. 6317, pp. 1240–1241, 2016.

[5] J. Pineau, P. Vincent-Lamarre, K. Sinha, V. Larivière, A. Beygelzimer, F. d'Alché Buc, E. Fox, and H. Larochelle, "Improving reproducibility in machine learning research (a report from the neurips 2019 reproducibility program)," *The Journal of Machine Learning Research*, vol. 22, no. 1, pp. 7459–7478, 2021.

[6] T. Malik, A. Vahldiek-Oberwagner, I. Jimenez, and C. Maltzahn, "Expanding the scope of artifact evaluation at hpc conferences: Experience of sc21," in *Proceedings of the 5th International Workshop on Practical Reproducible Evaluation of Computer Systems*, 2022, pp. 3–9.

[7] R. F. Boisvert, "Incentivizing reproducibility," *Communications of the ACM*, vol. 59, no. 10, pp. 5–5, 2016.

[8] A. C. Frery, L. Gomez, and A. C. Medeiros, "A badging system for reproducibility and replicability in remote sensing research," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 13, pp. 4988–4995, 2020.

[9] "ACM Digital Library," https://dl.acm.org/.

[10] D. L. Donoho, A. Maleki, I. U. Rahman, M. Shahram, and V. Stodden, "Reproducible research in computational harmonic analysis," *Computing in Science & Engineering*, vol. 11, no. 1, pp. 8–18, 2008.

[11] K. Keahey, "The Silver Lining," *IEEE Internet Computing*, vol. 24, no. 4, pp. 55–59, 2020.

[12] K. Keahey, J. Anderson, Z. Zhen, P. Riteau, P. Ruth, D. Stanzione, H. Gunawi, C. Hammock, and M. Joe, "Lessons Learned from the Chameleon Testbed," in *2020 USENIX Annual Technical Conference (USENIX ATC 20)*, 2020, pp. 219–233.

[13] D. Duplyakin, R. Ricci, A. Maricq, G. Wong, J. Duerig, E. Eide, L. Stoller, M. Hibler, D. Johnson, K. Webb *et al.*, "The design and operation of {CloudLab}," in *2019 USENIX annual technical conference (USENIX ATC 19)*, 2019, pp. 1–14.

[14] I. Baldin, A. Nikolich, J. Griffioen, I. I. S. Monga, K.-C. Wang, T. Lehman, and P. Ruth, "FABRIC: A National-scale Programmable Experimental Network Infrastructure," *IEEE Internet Computing*, vol. 23, no. 6, pp. 38–47, 2019.

[15] J. Yu, T. Chen, C. Gutterman, S. Zhu, G. Zussman, I. Seskar, and D. Kilper, "COSMOS: Optical Architecture and Prototyping," in *Optical Fiber Communication Conference*. Optical Society of America, 2019, pp. M3G–3.

[16] J. Breen, A. Buffmire, J. Duerig, K. Dutt, E. Eide, A. Ghosh, M. Hibler, D. Johnson, S. K. Kasera, E. Lewis *et al.*, "POWDER: Platform for Open Wireless Data-driven Experimental Research," *Computer Networks*, vol. 197, p. 108281, 2021.

[17] M. L. Sichitiu, I. Guvenc, R. Dutta, V. Marojevic, and B. Floyd, "AERPAW Emulation Overview," in *Proceedings of the 14th International Workshop on Wireless Network Testbeds, Experimental evaluation & Characterization*, 2020, pp. 1–8.

[18] H. Zhang, Y. Guan, A. Kamal, D. Qiao, M. Zheng, A. Arora, O. Boyraz, B. Cox, T. Daniels, M. Darr *et al.*, "Ara: A wireless living lab vision for smart and connected rural communities," in *Proceedings of the 15th ACM Workshop on Wireless Network Testbeds, Experimental evaluation & Characterization*, 2022, pp. 9–16.

[19] M. Norman, V. Kellen, S. Smallen, B. DeMeulle, S. Strande, E. Lazowska, N. Alterman, R. Fatland, S. Stone, A. Tan *et al.*, "Cloudbank: Managed services to simplify cloud access for computer science research and education," in *Practice and Experience in Advanced Research Computing*, 2021, pp. 1–4.

[20] L. Sarzyniec, S. Badia, E. Jeanvoine, and L. Nussbaum, "Scalability testing of the kadeploy cluster deployment system using virtual machines on grid'5000," in *SCALE Challenge 2012, held in conjunction with CCGrid'2012*, 2012.

[21] I. Baldine, Y. Xin, A. Mandal, C. H. Renci, U.-C. J. Chase, V. Marupadi, A. Yumerefendi, and D. Irwin, "Networked cloud orchestration: A geni perspective," in *2010 IEEE Globecom Workshops*. IEEE, 2010, pp. 573–578.

[22] K. Keahey and T. Freeman, "Contextualization: Providing One-click Virtual Clusters," in *2008 IEEE Fourth International Conference on eScience*. IEEE, 2008, pp. 301–308.

[23] "OpenStack Heat," https://docs.openstack.org/heat/latest/.

[24] "AWS CloudFormation," https://aws.amazon.com/cloudformation/.

[25] "Terraform by HashiCorp," https://www.terraform.io/.

[26] S. Wang, Z. Zhen, J. Anderson, and K. Keahey, "Reproducibility as Side Effect," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis (SC'18 Poster). IEEE Press*, 2018.

[27] "python-chi," https://python-chi.readthedocs.io/en/latest/.

[28] "Project Jupyter — JupyterHub," https://jupyter.org/hub.

[29] "OpenStack," https://www.openstack.org/.

[30] J. Anderson and K. Keahey, "Migrating towards single sign-on and federated identity," in *Practice and Experience in Advanced Research Computing*, 2022, pp. 1–8.

[31] "Welcome to Colaboratory," https://colab.research.google.com/.

[32] A. Clyburne-Sherin, X. Fei, and S. A. Green, "Computational reproducibility via containers in social psychology," *Meta-Psychology*, vol. 3, 2019.

[33] "Zenodo," https://zenodo.org/.

[34] "Trovi," https://chameleoncloud.gitbook.io/trovi/.

[35] "OpenStack Swift," https://docs.openstack.org/swift/latest/.

[36] "GitHub," https://github.com/.