# Enhancing Usability and Performance in Experimental Environments Management

Zahra Temori
zahratm@udel.edu
University of Delaware
Newark, DE, USA

Paul Marshall [Advisor]
marshalp@uchicago.edu
UChicago Department of Computer Science
Chicago, IL, USA

Kate Keahey [Advisor]
keahey@uchicago.edu
Argonne National Laboratory
Lemont, IL, USA

## ABSTRACT

Maintaining an identical setup and reproducing environments is a challenge in high-performance computing (HPC) and research [1]. HPC experiments are resource-intensive and depend on complex software environments. Existing methods, such as orchestration with containers, create controlled environments, but require careful setup and maintenance. However, snapshotting captures the complete state of a system in a single step, allowing researchers to automatically rebuild and restore identical environments. Although, concerns remain about snapshot efficiency and usability.

For snapshotting to be useful in HPC research,the tools need to be simple and straightforward to use. They also need to perform quickly on large bare metal environments. Therefore, we improved usability and evaluated the performance of `cc-snapshot`, a snapshotting tool on the Chameleon Cloud [2] testbed. Usability enhancements included new command line options, modular code, and automated tests. To optimize performance, we benchmarked alternative image formats and compression algorithms. The results show that `zstd` delivered up to 80% faster compression time during snapshot creation compared to `zlib`, while maintaining similar compression efficiency. These findings demonstrate that snapshotting can be a practical and effective tool to support reproducibility in HPC experiment.

**ACM Reference Format:**
Zahra Temori, Paul Marshall [Advisor], and Kate Keahey [Advisor]. 2025. Enhancing Usability and Performance in Experimental Environments Management. In *SC '25: The International Conference for High Performance Computing, November 16–21, 2025, St. Louis, MO.* ACM, New York, NY, USA, 2 pages. https://doi.org/XXXXXXX.XXXXXXX

## 1 INTRODUCTION

Reproducibility ensures that experiments can be repeated, validated, and extended with confidence. Achieving a reproducible environment requires identical software stacks, with the exact same dependencies, and configuration. The Chameleon Cloud testbed provides the cc-snapshot tool to support reproducibility by capturing the complete state of a running system. This allows researchers to rerun experiments exactly as before, share setups among each other, and avoid potential environmental issues such as missing dependencies or version mismatches. In this work, we explore general techniques to enhance snapshotting as a reproducible method, applying them to cc-snapshot on Chameleon as a case study, but with strategies that can extend to other HPC platforms.


*Supercomputing '25, November 15–11, 2025, St. Louis, MO*

ACM ISBN 978-1-4503-XXXX-X/25/11
https://doi.org/XXXXXXX.XXXXXXX

## 2 APPROACH

The project was divided into two phases. The first phase focused on usability, reorganizing the tool, and expanding its capabilities. The second phase was benchmarking to evaluate alternative image formats and compression methods to improve snapshotting performance.

### 2.1 Usability Enhancements

The original snapshotting tool had challenges including a limited command line, tightly coupled logic, and minimal testing support, which made it difficult for users to interact with and developers to maintain. To enhance the command line interface, we added a flag to disable automatic updates, giving users more control over when to pull the latest version. We also introduced a dry-run flag to simulate actions before running a snapshot, allowing users to preview changes and correct errors before investing time in a full snapshot. This enables faster iteration and prevents wasted time on failed runs. Moreover, we implemented support for a custom source path, enabling snapshots of specific directories, which helps developers test smaller subsets of the system rather than complete environments. To improve maintainability, we refactored the codebase into five modular functions, allowing developers to make future changes more easily. In addition, we added automated tests with GitHub Actions to validate new and existing features and ensure that changes work as expected.

### 2.2 Performance Optimization

The default format and compression on snapshotting was Qcow2 with zlib, which often resulted in long snapshot creation time. To address this performance issue, we benchmarked other alternatives such as QCOW2 with zstd compression, and RAW with no compression. We also chose three images of varying sizes: small 4.47 GiB, medium 7.62 GiB, and large 12.7 GiB. The medium size image was user created to demonstrate the snapshotting and compression works for both Chameleon-supported images and user-created images.

## 3 RESULTS

We ran each image with different compression methods and recorded four key metrics: creation time, upload time, boot time, and final image size. We calculated the overall time of each compression method from experiments on three different image sizes to evaluate which performed better. The results revealed that zstd compression reduced the creation time around 80.6% across the three image sizes. The upload time for zstd was nearly equal to the zlib method, while RAW images, due to no compression and larger size, uploaded much slower compared to images compressed with zlib and zstd. The boot time was nearly the same across all images, confirming that zlib and zstd take about the same time to uncompress, while RAW images take longer to boot due to large size.

Our work suggested that QCOW2 with zstd compression should be used instead of QCOW2 with zlib compression when creating a snapshot. Importantly, we were able to achieve these significant improvements in creation time without sacrificing boot performance, since faster compression with zstd did not introduce additional decompression overhead. For HPC users, this enables to generate and share reproducible environments faster and improve turnaround time.

## 4 CONCLUSION

Snapshotting is a practical way to support reproducibility in HPC, but to be effective, it should be easy to use and fast enough for real research workflows. Our results show that using zstd compression can drop the snapshot creation time by over 80% compared to the common default zlib compression, without affecting upload or boot performance. While our implementation targeted cc-snapshot on Chameleon, the approaches we applied are broadly applicable to snapshotting systems across platforms. Looking ahead, we plan to integrate zstd , try it on more workloads and image types, and explore ways to improve snapshotting for even greater speedups and reliable results.

## 5 ACKNOWLEDGMENTS

The results of this poster were obtained on the Chameleon Testbed funded by the National Science Foundation (Award No. 2431425).

## REFERENCES

[1] Kate Keahey, Marc Richardson, Rafael Tolosana Calasanz, Sascha Hunold, Jay Lofstead, Tanu Malik, and Christian Perez. Report on challenges of practical reproducibility for systems and hpc computer science, April 2025.

[2] Kate Keahey, Jason Anderson, Zhuo Zhen, Pierre Riteau, Paul Ruth, Dan Stanzione, Mert Cevik, Jacob Colleran, Haryadi S. Gunawi, Cody Hammock, Joe Mambretti, Alexander Barnes, François Halbach, Alex Rocha, and Joe Stubbs. Lessons learned from the chameleon testbed. In *Proceedings of the 2020 USENIX Annual Technical Conference (USENIX ATC '20)*. USENIX Association, July 2020.