Sharing and Replicability of Notebook-Based Research on Open Testbeds

MAXINE KING and JASON ANDERSON, The University of Chicago
KATE KEAHEY, Argonne National Laboratory

CCS Concepts: • **Information Systems → Collaborative and social computing systems and tools**; *Information Storage Technologies.*

Additional Key Words and Phrases: replicability, open testbeds, notebooks, sharing

ACM Reference Format:
Maxine King, Jason Anderson, and Kate Keahey. 2019. Sharing and Replicability of Notebook-Based Research on Open Testbeds. 1, 1 (October 2019), 2 pages.

1 BACKGROUND

Replicability of experiments is a cornerstone of all scientific research. Without replicating, repeating, and reproducing experiments, the scientific communities cannot easily leverage existing work. However, as research becomes increasingly complex, reproducing it becomes increasingly expensive and therefore scientists need increasingly sophisticated tools to make their experiments replicable.

Today, there are many large-scale computer science research experiments that use more than a laptop's worth of computing power. These kinds of experiments can be run on open testbeds, which provide a consistent, powerful, and customizable place to run code. Additionally, as the standard rises for documentation in computer science research, scientists have started to use notebooks (such as Jupyter Notebooks) as an easy way to integrate ideas, results, and process. While open testbeds provide a common denominator in terms of an environment, and features such as Chameleon's integration of Jupyter make configuring repeatable experiments easier, some challenges remain. For example, a researcher using Chameleon's Jupyter environment has limited means of sharing their experimentation with others.

From the perspective of a researcher looking to replicate an experiment, there's another set of hurdles. First, there's no easy way to get published code into the open testbed environment, even if that's where it was originally written. Second, experiments designed to run on open testbeds are not easily findable using existing search engines. While artifacts can be found using a known unique identifier, the absence of a means of finding relevant work is a missed opportunity for researchers to exchange knowledge about how to design reproducible experiments on an open testbed and replicate each other's experiments.

Authors' addresses: Maxine King, maxineking@uchicago.edu; Jason Anderson, jasonanderson@uchicago.edu, The University of Chicago, Chicago, Illinois, 60637; Kate Keahey, keahey@anl.gov, Argonne National Laboratory, Lemont, Illinois, 60439.

2 PROBLEM

We want to create an easy way to share and replicate research experiments while making use of the computing power and consistency of an open testbed; in our case, Chameleon. In essence, we hope to combine the existing marriage between writing code in notebooks and running on open testbeds with the ability to share notebooks easily, in order to allow researchers to write well-documented code, run it in a powerful, reproducible environment, and share it. We seek to answer the question: is there an effective, user-friendly way to connect open testbeds, research storage, and interactive notebooks?

3 APPROACH

The existing partial solutions are open testbeds, notebooks, and sharing services, so we picked one of each to come up with an integrated solution. For our open testbed we are using Chameleon, which is a large-scale, bare-metal reconfigurable testbed for Computer Science research with significant hardware diversity that makes it capable of supporting a broad set of experiments. Our notebooks are Jupyter notebooks, which are already integrated with the Chameleon testbed. For a permanent artifact storage solution, we settled on Zenodo, which guarantees long-term storage (backed by CERN), assigns a DOI to each deposition, has no file size limit, and is already widely used.

We wanted to create two distinct user paths. First, a researcher who just finished a project on Chameleon should be able to easily upload their files to both Zenodo and our sharing portal. Second, a researcher looking for relevant research should be able to find experiments that work well with Chameleon's Jupyter environment and easily import and re-run them. We deployed this solution to the Chameleon testbed in three parts.

Part 1: A Sharing Platform: A sharing platform on Chameleon lets its users browse through existing research. This portal has search features including filtering by labels, searching by keyword, author, or description, and grouping associated artifacts (such as notebooks, appliances, or data). The artifacts are stored on Zenodo, and the portal includes links to view the item there.

Part 2: An Import Mechanism: A simple button press from the sharing portal pulls all files and setup info from Zenodo and lands the user in Chameleon's JupyterHub environment. This server exists separately from the user's main project server so that all environment configuration and files are kept separate. All files are pre-loaded, and python requirements are pre-installed.

Part 3: An Export Mechanism: An extension for JupyterHub allows users to publish their work directly to both Zenodo and our portal from the Jupyter environment, without needing to learn any additional tools or download anything. Upon doing so, they are shown related research. By sharing to Zenodo, they are guaranteed storage of their artifacts and permanent access to it online. By sharing to our portal, they make it easy for other Chameleon/Jupyter users to find and replicate their work.

4 CONCLUSION AND NEXT STEPS

With this solution, all relevant work (artifacts using Chameleon and Jupyter) is in one searchable place and all code is stored via Zenodo, which makes it accessible outside of Chameleon and guarantees that it will be stored permanently. Additionally, sharing code in and out of Chameleon's JupyterHub environment is now much easier than before. While this sharing platform is currently somewhat limited–for example, it works only with zip files, not other formats, only installs python requirements, and only does so if they're in exactly the right place–these gaps are only due to the relative immaturity of this project, and are not fundamental challenges.