ReproZip: Computational Reproducibility With Ease

Fernando Chirigati, Rémi Rampin, Dennis Shasha, and Juliana Freire

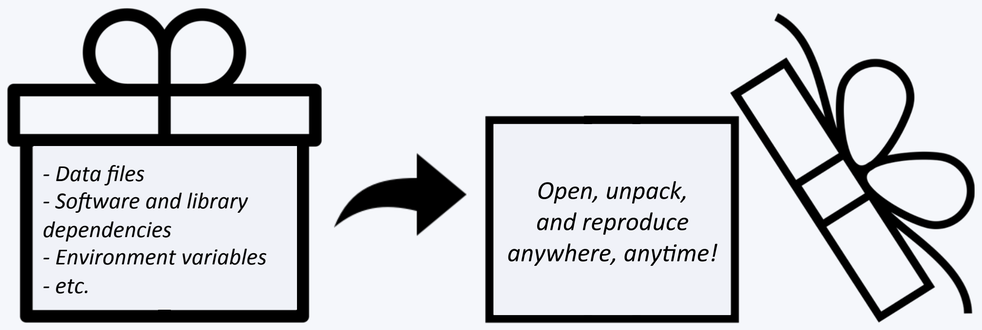
NYU

Reproducibility is Necessary, but Hard ...

... Until now!

ReproZip!

Pack your experiment on your system ***S*** ...
... and **unpack** on another system ***S'*** ...
... using as few as 2 commands for each step !

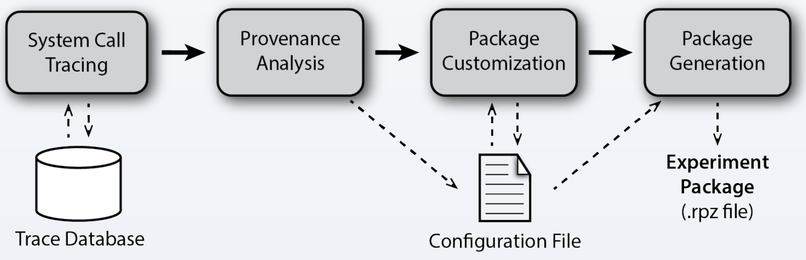


reprozip trace
reprozip pack

reprounzip setup
reprounzip run

Come Try ReproZip!

Packing Experiments on Operating System *S*



System Call Tracing (reprozip trace)

- ReproZip transparently captures the provenance of the execution of the experiment, i.e., all the required information to correctly reproduce the experiment, including data files, databases, programs, library dependencies, and OS information.
- The execution trace is stored in SQLite.

Provenance Analysis (reprozip trace)

- Given the files that were read and using the package manager of the OS, ReproZip identifies the software packages on which the experiment depends. ReproZip also uses some heuristics to identify input and output files.
- All the required information is written to a human-readable configuration file.

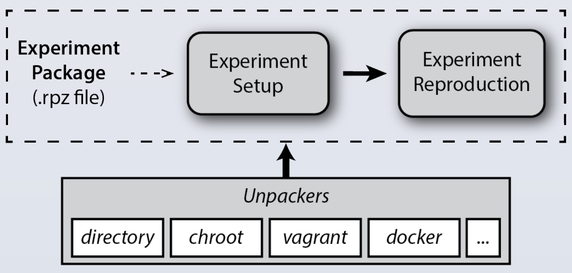
Package Customization

- The configuration file can be edited by researchers, e.g., to remove large files that can be obtained elsewhere, or to remove sensitive or proprietary information.

Package Generation (reprozip pack)

- All the required files are packed on the author's system *S* in a **.rpz** file.

Unpacking Experiments on Operating System *S'*



Unpackers

- *S* and *S'* are incompatible: *vagrant*, *docker*
- *S* and *S'* are compatible: *directory*, *chroot*, *vagrant*, *docker*

Experiment Setup (reprounzip setup)

- The experiment is automatically extracted and set up depending on the chosen unpacker.

Experiment Reproduction (reprounzip run)

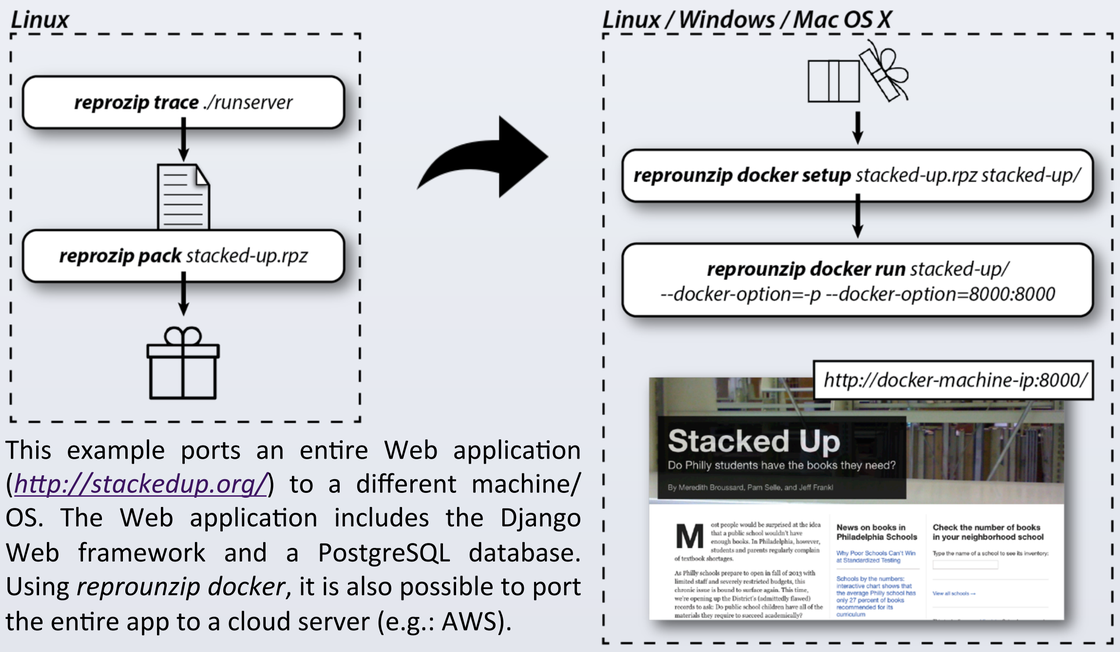
- The experiment is reproduced depending on the chosen unpacker. For instance, for *vagrant* and *docker*, this is done inside a virtual image and a Docker container, respectively, in an automatic and transparent way through *reprounzip* and its command-line interfaces.

File and Dataflow Management

- Input files can be replaced using ***reprounzip upload***.
- Output files can be retrieved using ***reprounzip download***.
- ReproZip can also derive a specification of the experiment for the VisTrails system, which represents the original workflow in a GUI and enables the dataflow to be modified to explore different techniques, perform analyses, and reuse some of the steps for your own research.

Demos

ReproZip supports a wide range of experiments, including client-server scenarios, databases, and graphical and interactive tools. Our repository includes a variety of examples from different domains (*http://bit.ly/reprozip-examples*). The following example is available at *http://bit.ly/stacked-up*:



This example ports an entire Web application (*http://stackedup.org/*) to a different machine/OS. The Web application includes the Django Web framework and a PostgreSQL database. Using *reprounzip docker*, it is also possible to port the entire app to a cloud server (e.g.: AWS).

Other examples include:

- An interactive visualization tool: *http://bit.ly/bus-vis*
- A simulation in statistical physics: *http://bit.ly/ising-model*
- Plots from a SIGMOD'16 paper: *http://bit.ly/data-polygamy*

Other Use Cases

- ReproZip has been used in the reproducibility section of Elsevier's Information Systems Journal.
- ReproZip was recommended by the ACM SIGMOD Reproducibility Review.
- ReproZip has been listed on the Artifact Evaluation Process guidelines.

References

ReproZip's Homepage: *https://vida-nyu.github.io/reprozip/*

ReproZip's Documentation: *https://reprozip.readthedocs.io*

Acknowledgments: This work was supported in part by NSF awards CNS-1229185 and CI-EN-1405927, and by the Moore-Sloan Data Science Environment at NYU.