Experimental Testbeds as a Foundation for Reproducibility of Experiments

Kate Keahey

Computer Science experimental testbeds allow investigators to explore a broad range of different state-of-the-art hardware options, assess scalability of their systems, and provide conditions that allow deep reconfigurability and isolation so that one user does not impact the experiments of another. Although the primary purpose of those testbeds is to provide resources to users who would not be able to satisfy their experimental needs otherwise, an important side-effect is that multiple users and user groups have access to the same resources, that are compatible with the same experimental artifacts, such as appliances/images or orchestration templates. This creates conditions which allow users to share experiments and replicate each other's work more easily and creates an opportunity to foster good experimental practices as well as create a sharing ecosystem.

To facilitate the creation of such ecosystem we have developed two mechanisms that we call respectively reproducibility by side-effect and reproducibility by default. Reproducibility by side-effect is intended to aid users in much the same the Linux "history" command allows you to see what commands you typed while working on a problem. Similarly in a testbed, we often need to refresh our memory on the exact configuration, conditions, or steps taken in the conduct of an experiment. Chameleons Experiment Pre´cis captures all the distributed events generated in a testbed by a user and presents them with a summary (a pre´cis) of an experiment; the user can then filter or preview the events to include only the relevant ones thus working with an accurate and impartial representation of their work. The pre´cis can be further used to generate a description of the experiment in English or potentially a set of experimental artifacts - images, orchestration templates, or commands - that will reproduce the experiment.

Searching for the best expression of an experiment led us to experiment with notebooks which combine *ideas* in the form of text, *experimental process* expressed as code, and *results* expressed as data processing. Using notebooks, users can develop their experiments step by step; as each step is then modifiable and the notebooks can be shared, each represents a convenient vehicle for repeating and replicating an experiment. To facilitate the use of notebooks we integrated Jupyter with Chameleon by providing a JupyterHub server integrated with Chameleon authentication methods for our users, as well as developed python and bash libraries representing the testbed API. While notebook code is generally limited to executing in containers (such as Docker containers), with this integration, Chameleon users can define arbitrarily complex containers powered by the testbed -- and then use them from their Jupyter notebooks to run complex experiments. The integration thus combines the flexibility of notebooks with the power of experimental testbeds.

Our presentation will describe new capabilities targeted at improving experiment management, monitoring, and analysis as well as tying together testbed features to improve experiment repeatability and replicability. We seek to engage a broader community in the discussion of general approach to reproducibility, best experimental practices, as well as interaction between testbeds leading to the support of a broader range of experiments and developing common practices and platforms for reproducibility.