PDSW-DISCS Artifact Evaluation Criteria

Infrastructure

We have an instance of Jenkins running at http://ci.falsifiable.us, maintained by members of the Systems Research Lab (SRL) at UC Santa Cruz. Detailed instructions on how to create an account on this service and how to use it is available here (also includes instructions on how to self-host it). This service allows researchers and students to easily automate the execution and validation of experimentation pipelines. Users of this service follow a convention for structuring their experiment repositories, which allows the service to be domain-agnostic. The service reports the status of an experimentation pipeline (fail, success or validated).

NOTE: Using our server is not obligatory. However, we will require authors to provide a URL to the service they use (e.g., TravisCI, GitLabCI, CircleCI, Jenkins, etc) so reviewers can validate the repeatability of the submission using that service.

Evaluation Criteria

In order to be considered for an ACM badge, the pipelines associated to a submission must be in a healthy (runnable) state and automate the following:

- Code and data dependencies. Code must reside on a version control system (e.g. github, gitlab, etc.). If datasets are used, then they should reside in a dataset management system (datapackage, gitlfs, dataverse, etc.). The experimentation pipelines must obtain the code/data from these services on every execution.
- Setup. The pipeline should build and deploy the code under test. For example, if a pipeline is using containers or VMs to package their code, the pipeline should build the container/VM images prior to executing them. The goal of this is to verify that all the code and 3rd party dependencies are available at the time a pipeline runs, as well as the instructions on how to build the software.
- Resource allocation. If a pipeline requires a cluster or custom hardware to reproduce results, resource allocation must be done as part of the execution of the pipeline. This allocation can be static or dynamic. For example, if an experiment runs on custom hardware, the pipeline can statically allocate (i.e. hardcode IP/hostnames) the machines where the code under study runs (e.g. GPU/FPGA nodes). Alternatively, a pipeline can dynamically allocate nodes (using infrastructure automation tools) on CloudLab, Chameleon, Grid5k, SLURM, Terraform (EC2, GCE, etc.), etc.
- Validation. Scripts must verify that the output corroborates the claims made on the article. For example, the pipeline might check that the throughput of a system is within

an expected confidence interval (e.g. defined with respect to a baseline obtained at runtime).

A list of example pipelines meeting the above criteria:

- BLIS paper. We took an appendix and turned it into executable pipeline.
- Proxy app. Runs LULESH linked against MPIp to capture runtime MPI perf metrics.
- Linux kernel development. Uses a VM to compile, test and deploy Linux
- Relational database performance. Runs pgbench to compare two versions of postgres.

Many more available at this page.

Questions & Answers

Since the reproducibility submission process is new, we expect quite a few questions. To make the process of answering questions more efficient, please use our gitter room to browse for possible answers or post your question.