The Popper Experimentation Protocol

Ivo Jimenez, Sina Hamedian, Michael Sevilla, Noah Watkins, Carlos Maltzahn (UC Santa Cruz)
Kathryn Mohror (LLNL)
Jay Lofstead (SNL)
Andrea Arpcaci-Dusseau, Remzi Arpac-Dusseau (UW Madison)
Problem of Reproducibility in Computation and Data Exploration

- What compiler was used?
- Which compilation flags?
- How was subsystem X configured?
- How does the workload look like?
- What if I use input dataset Y?
- And if I run on platform Z?
- …
Lab Notebook

SHA: a13570c1d700f85d2a9de348beb37215eb9784
Author: Ivo Jimenez <ivo.jimenez@gmail.com>
Date: Thu Aug 25 2016 01:29:21 CMT-0700 (PDT)
Subject: Results of running base-vs-targets for stressing on 4 machines

Results of running base-vs-targets for stressing on 4 machines

Base machine is 'issdm-12' and targets were tuned with the 'crafty' and 'c-ray' benchmarks.

The main difference between these results and the ones that appear on our VarSys '16 paper is that we are reflecting the speedup function for x=1, for both (1) tuning targets and (2) displaying results. This allows us to show better the reduction in variability without having to deal with different scales (slowdowns that lie between the 10-12 range are instead reflected and treated as speedups).

In short, we now unambiguously observe reduced variability when targets are limited. A couple of outliers, in particular stressing-ng's memfd stressor, when limited, is very slow on target machines.
Common Experimentation Workflow

- Code
- Package
- Execute
- Output Data
- Analyze/Visualize
- Manuscript

Input Data
Analogies with DevOps Practice

<table>
<thead>
<tr>
<th>Scientific exploration</th>
<th>Software project</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experiment code</td>
<td>Source code</td>
</tr>
<tr>
<td>Input data</td>
<td>Test examples</td>
</tr>
<tr>
<td>Analysis / visualization</td>
<td>Test analysis</td>
</tr>
<tr>
<td>Validation</td>
<td>CI / Regression testing</td>
</tr>
<tr>
<td>Manuscript / note book</td>
<td>Documentation / reports</td>
</tr>
</tbody>
</table>

Key Idea behind The Popper Protocol: manage a scientific exploration like software projects
Common Experimentation Workflow
DevOps in Practice

Typical

```bash
$ bash myscript.sh
```

DevOps

```bash
myscript.sh
$ bash myscript.sh
```
1. Pick one or more DevOps tools.
   – At each stage of experimentation workflow.
2. Put all associated scripts in version control.
   – Make experiment self-contained.
   – For external dependencies (code and data), reference specific versions.
3. Document changes as experiment evolves.
   – In the form of commits.

Popper-compliant Experiments

• An experiment is *Popper-compliant* if all of the following is available (self-contained):
  – Experiment code.
  – Data dependencies.
  – Parameterization.
  – Results.
  – Validation.
$ cd mypaper-repo
$ popper init
   -- Initialized Popper repo mypaper-repo

$ popper experiment list
   -- available templates ---------------
  ceph-rados proteustm mpi-comm    adam
  cloverleaf  gasyysfs  zlog    bww
  spark-stand torpor malacology genevo
  hadoop-yarn kubsched alg-encycl macrob

$ popper add gasyysfs
   -- Added gasyysfs experiment to mypaper-repo
Popper-compliant Experiments
Popper and CI Systems
Continuous Validation of Experiment Results
Continuous Integration (CI)

“In software engineering, **continuous integration** (CI) is the practice of merging all developer working copies to a shared mainline several times a day.”

source: https://insights.sei.cmu.edu/devops/2015/01/continuous-integration-in-devops-1.html
Continuous Validation of Scientific Experiments

• Project structure follows a convention.
  – One experiment per subfolder
  – Optional paper folder
• Bash-oriented interface to execution:
  – `setup.sh`: hardware allocation/configuration, software deployment.
  – `run.sh`: run experiment, obtain results.
  – `teardown.sh`: cleanup, release resources.
• Goal: automate end-to-end execution
$ popper experiment init myexp
-- Initialized exp1 experiment.

$ ls -l experiments/myexp/
total 20K
-rw-r---- 1 ivo ivo 8 Apr 29 23:58 README.md
-rwxr-x-- 1 ivo ivo 210 Apr 29 23:58 run.sh
-rwxr-x-- 1 ivo ivo 206 Apr 29 23:58 setup.sh
-rwxr-x-- 1 ivo ivo 61 Apr 29 23:58 teardown.sh

#!/bin/bash
# request remote resources
docker run google/cloud-init...

#!/bin/bash
# trigger execution of experiment
docker run google/kubectl run ...

...
Automating Execution Stages

```
provider:
  type: chameleonbaremetal
image_name: CC-Ubuntu14.04-Docker
subnet: {name: shared-subnet1}
walltime: '00:30:00'
resources:
  storage:
    compute: 1
```

docker run --rm \
  -e OS_AUTH_URL=$OS_AUTH_URL \
  -e OS_TENANT_ID=$OS_TENANT_ID \
  -e OS_TENANT_NAME=$OS_TENANT_NAME \
  -e OS_PROJECT_NAME=$OS_PROJECT_NAME \
  -e OS_USERNAME=$OS_USERNAME \
  -e OS_PASSWORD=$OS_PASSWORD \
  -e OS_KEYNAME=$OS_KEYNAME \
  -v `pwd`/enos:/enos \
  --workdir=/enos \
  ivotron/enos:2.3.0 up
$ popper check exp1

Popper check started

Stage: setup.sh ..... 
Stage: run.sh .................
Stage: teardown.sh ..

Popper check finished

Status: SUCCESS
Codified Validations

```plaintext
num_nodes, throughput, raw_bw, net_saturated
```

- Log file
- CSV
- DB Table
- TSDB
- ...

```
expect
linear(num_nodes, throughput)
```

```
when not net_saturated
expect
throughput >= (raw_bw * 0.9)
```

[1]: Jimenez et al. Tackling the reproducibility problem in storage systems research with declarative experiment specifications, PDSW '15.
One More Experiment Stage

1. Setup
   – Resource allocation, software deployment.

2. Execution
   – Run experiment, obtain results.

3. Validation
   – Verify claims by checking validation statements against result datasets.

4. Teardown
   – Cleanup, release resources.
One More Type of Status

- **FAIL**
  - Any failure along execution pipeline.
  - Ignore teardown errors.

- **SUCCESS**
  - Experiment runs OK end-to-end.

- **GOLD**
  - Experiment runs OK *and* all validations pass.
CI Pipeline

1. Commit change to experiment
2. Trigger execution
3. Run multi-node experiment on one of supported backends
4. Experiment generates output datasets or runtime metrics
5. Validate experiment results by testing codified assertions on output
6. Keep track of execution and associated status to the corresponding commit

README.md

- build passing
- coverage 72%
- downloads 12
- stable No Release
- unstable dev-master
- license GNU


```
$ ls -l myrepo/
total 20K
-rw-r----- 1 ivo ivo  8 Apr 29 23:58 .git/
-rwxr-x--- 1 ivo ivo 210 Apr 29 23:58 experiments/
-rwxr-x--- 1 ivo ivo 206 Apr 29 23:58 paper/

$ popper ci travis
Created .travis.yml file.

$ ls -l myrepo/
total 20K
-rw-r----- 1 ivo ivo  8 Apr 29 23:58 .git/
-rw-r----- 1 ivo ivo  8 Apr 30 12:02 .travis.yml
-rwxr-x--- 1 ivo ivo 210 Apr 29 23:58 experiments/
-rwxr-x--- 1 ivo ivo 206 Apr 29 23:58 paper/
```
ACM’s Three Rs of Reproducibility[1]

<table>
<thead>
<tr>
<th>Result Status</th>
<th>Re-executed By</th>
<th>Artifacts</th>
<th>ACM Badge</th>
<th>PopperCI Badge</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repeatability</td>
<td>Author(s)</td>
<td>Original</td>
<td><img src="https://www.acm.org/publications/policies/artifact-review-badging" alt="Results Replicated" /></td>
<td><img src="https://www.acm.org/publications/policies/artifact-review-badging" alt="Popper GOLD" /></td>
</tr>
<tr>
<td>Replicability</td>
<td>Nonauthor(s)</td>
<td>Original</td>
<td><img src="https://www.acm.org/publications/policies/artifact-review-badging" alt="Results Replicated" /></td>
<td><img src="https://www.acm.org/publications/policies/artifact-review-badging" alt="Popper GOLD" /></td>
</tr>
<tr>
<td>Reproducibility</td>
<td>Anyone</td>
<td>Re-implemented</td>
<td><img src="https://www.acm.org/publications/policies/artifact-review-badging" alt="Results Reproduced" /></td>
<td></td>
</tr>
</tbody>
</table>

[1]: https://www.acm.org/publications/policies/artifact-review-badging
Conclusion

• Popper Experimentation Protocol
  – Three high-level steps for generating experiments that are easy to re-execute.

• PopperCI
  – Convention for structuring Popper repositories.

• Popper CLI check command
  – CLI tool to test (locally) for PopperCI-compliance.

• PopperCI Web Service
  – Track experiment status; share/re-run experiments.

• Repeatability/Replicability Badges
  – “Compatible” with ACM’s policy on reproducibility.