

Popper
falsifiable.us

Practical Reproducible Evaluation of Computer Systems

Ivo Jimenez, Michael Sevilla, Noah Watkins,
Sina Hamedian, Pete Wilcox, Carlos Maltzahn,
Jay Lofstead, Kathryn Mohror, Adam Moody,
Andrea Arpaci-Dusseau, Remzi Arpaci-Dusseau



Problem of Reproducibility in Computation and Data Exploration

The UI is capable of graphing the ratio $\frac{\text{totWork}(OPT)}{\text{totWork}(WFIT)}$ (Figure 2) in parallel with the analysis of the query stream (A high ratio indicates that WFIT generates good recommendations). It also makes available the recommendations that are generated at each step, as well as the internal bookkeeping that the algorithm maintains. We will show some of this information as part of this scenario.

Scenario #2. We delve a little bit more into the details of our tool by allowing the candidate-index set to be automatically maintained but again skipping the feedback feature (so that the candidate set can grow and shrink automatically grow/shrink and be repartitioned over time based on the calculations of index interactions associated with each statement). This brings the tool into a completely online mode where it can operate autonomously without any user intervention.

Figure 3: Evolution of the candidate set with respect to partitioning (by calculating index interactions at each step). Each set corresponds to phases 1, 2 and 3 respectively.

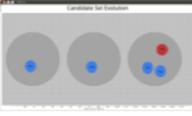


Figure 4: Multiple instances of WFIT running in parallel. The ratio for the "good" and "bad" instances is done at step 1, causing the divergence in their behavior with respect to the "no-feedback" instance.



ence several datasets (TPC-C, TPC-DS, TPC-E, TPC-H and NREF).

We will show two WFIT variants: one with a stable and fixed candidate set partitioning; another whose candidate set is allowed to be automatically maintained. Similarly to scenario #1, we will graph the OPT vs. WFIT ratio in real-time as the workload is processed (Figure 5).

Figure 5: Two instances of WFIT running the Online Index Selection Benchmark. One with a fixed and stable candidate set (FIXED); another one with an automatically maintained candidate set (AUTO).

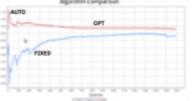


Figure 5: Two instances of WFIT running the Online Index Selection Benchmark. One with a fixed and stable candidate set (FIXED); another one with an automatically maintained candidate set (AUTO).

REFERENCES

- [1] K. Schanatter, S. Abiteboul, L. Kollar, A. Marathe, V. Narasayya, and M. Syriac. Database tuning advisor for microsoft SQL server 2005. In *SIGMOD conference*, pages 930–932, 2005.
- [2] S. Abiteboul, D. Suciu, and J. Ullman. *Databases: the complete design tuning workload as a sequence*. In *SIGMOD Conference*, pages 483–494, 2006.
- [3] A. Agapiou and C. Faloutsos. *Online computation and competitive analysis*. Cambridge University Press, 1998.
- [4] N. Bruno and S. Chaudhuri. An online approach to physical design tuning. *VLDB Journal*, 14(2):289–308, 2005.
- [5] N. Bruno and S. Chaudhuri. Constrained physical design tuning. *Proceedings of the VLDB Endowment*, 1:4–15, 2008.
- [6] N. Bruno and S. Chaudhuri. Interactive physical design tuning. In *ICDE*, pages 1161–1164, 2008.
- [7] D. Zilio et al. DB2 Design Advisor: Integrated Automatic Physical Design Tuning. In *VLDB*, pages 109–120, 2004.
- [8] B. Dageville, D. Das, K. Dinkar, K. Vagnino, M. Zait, and M. Zoundi. Automatic SQL Tuning in Oracle 10g. In *VLDB*, pages 109–120, 2004.
- [9] K. Schanatter, S. Abiteboul, T. Milo, and N. Polyotis. On-line index selection for shifting workloads. In *ICDE*, pages 459–468, 2009.
- [10] K. Schanatter and N. Polyotis. A Benchmark for Online Index Selection. In *ICDE*, pages 1761–1768, 2009.
- [11] K. Schanatter and N. Polyotis. On-line automatic index tuning: the case of Oracle 11g. *VLDB*, 35(5):478–489, 2012.
- [12] K. Schanatter, N. Polyotis, and L. Gotto. Index interactions in physical design tuning: modeling, analysis, and applications. *VLDB*, 2(1):1234–1245, 2009.

- What compiler was used?
- Which compilation flags?
- How was subsystem X configured?
- How does the workload look like?
- What parameters can be modified?
- What if I use input dataset Y?
- And if I run on platform Z?
- ...

Lab Notebook

torpor-popper (branch: master)

View Branch Create Branch Search

Subject	Author	Date
master	Ivo Jimenez	2016-10-05...
Adds torpor as a submodule	Ivo Jimenez	2016-08-25...
Results of running base-vs-targets for stressng on 4 machines	Ivo Jimenez	2016-08-25...
Makes use of 'baseliner' role fo...	Ivo Jimenez	2016-08-25...
Removes unused line	Ivo Jimenez	2016-08-21...
Will add later	Ivo Jimenez	2016-08-21...
Re-adds baseliner ansible role	Ivo Jimenez	2016-08-21...
renames ansible folder to exper...	Ivo Jimenez	2016-08-21...
benchmarks now have their ow...	Ivo Jimenez	2016-08-21...
Adds latest version of baseliner...	Ivo Jimenez	2016-06-12...
Adds experiment on all stress-...	Ivo Jimenez	2016-06-12...
Adds same-platform variability...	Ivo Jimenez	2016-06-10...
WIP on using baseliner role	Ivo Jimenez	2016-06-10...
Checks first if docker is already...	Ivo Jimenez	2016-05-16...
Tunning cpu-quota of 5 machines	Ivo Jimenez	2016-05-16...
Fixes creation of parameters an...	Ivo Jimenez	2016-05-16...
Adds logic to install statically li...	Ivo Jimenez	2016-05-16...

Gist it

SHA: af13570c1d700f85d2a9de348beb37215eb978c4
Author: Ivo Jimenez <ivo.jimenez@gmail.com>
Date: Thu Aug 25 2016 01:29:21 GMT-0700 (PDT)
Subject: Results of running base-vs-targets for stressng on 4 machines
Parent: [7a13185872bc73719048a1bd8a76f68a06798e13](#)

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

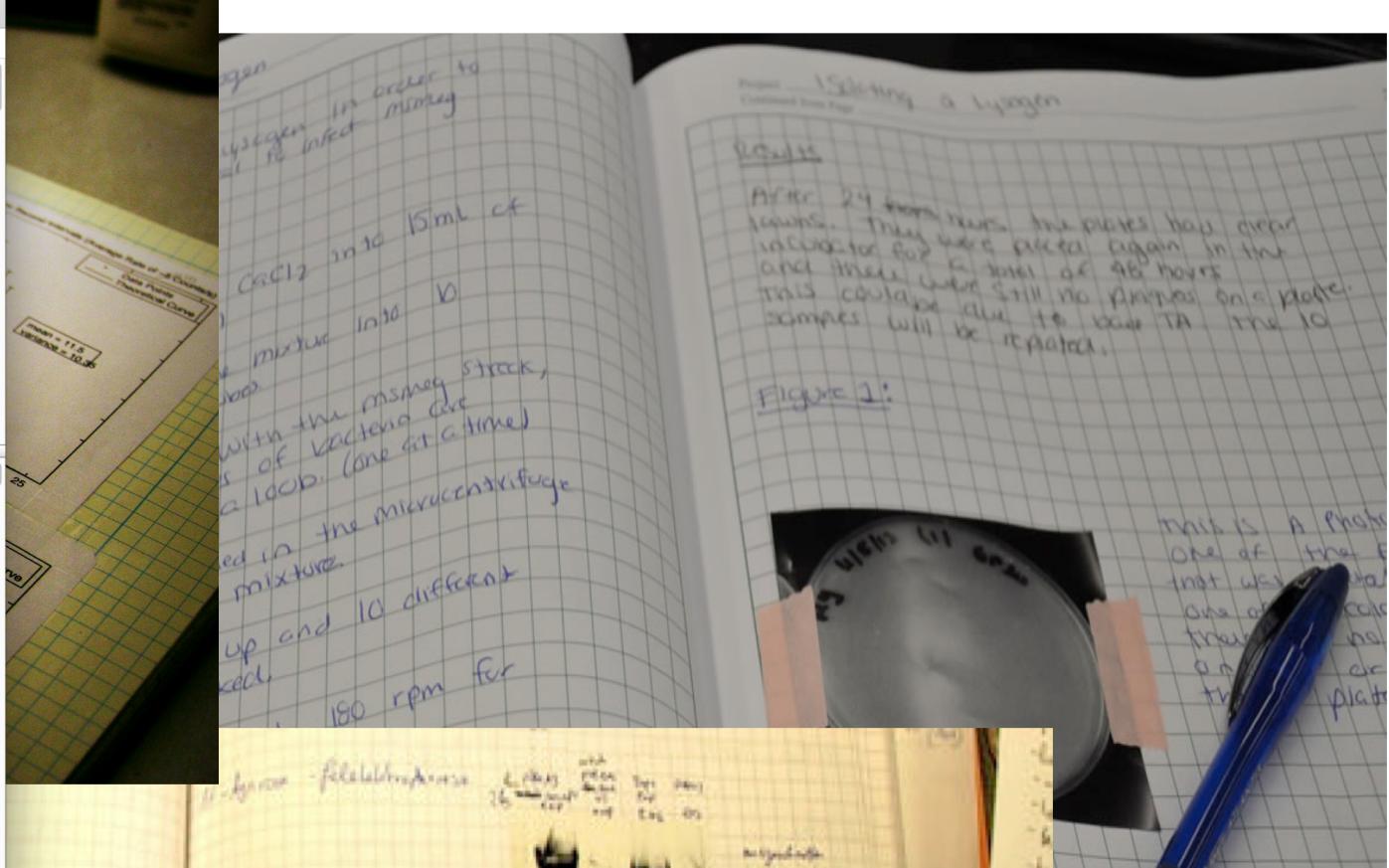
Base machine is 'issdm-12' and targets were tunned with the 'crafty' and 'c-ray' benchmarks.

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 [0-1] range are instead reflected and treated as speedups).

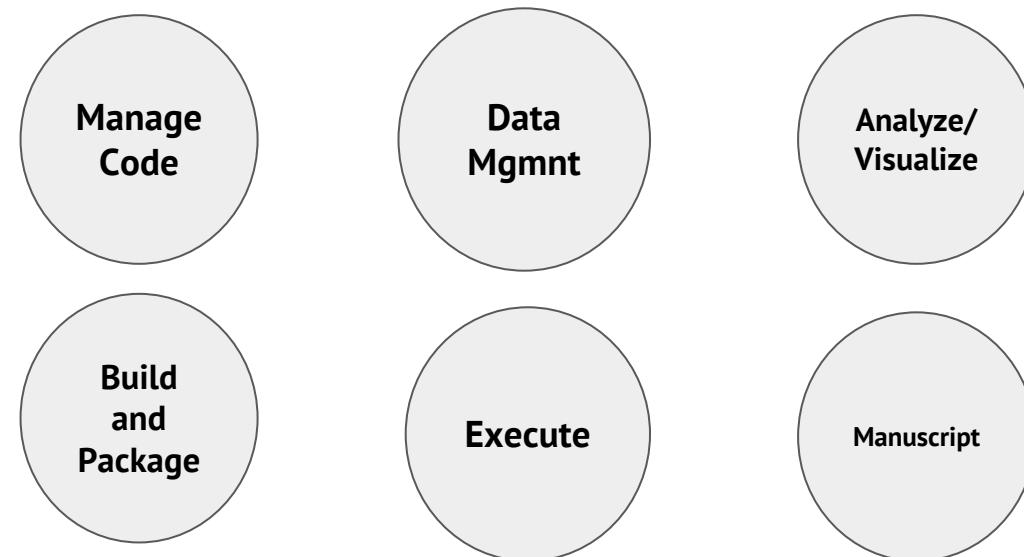
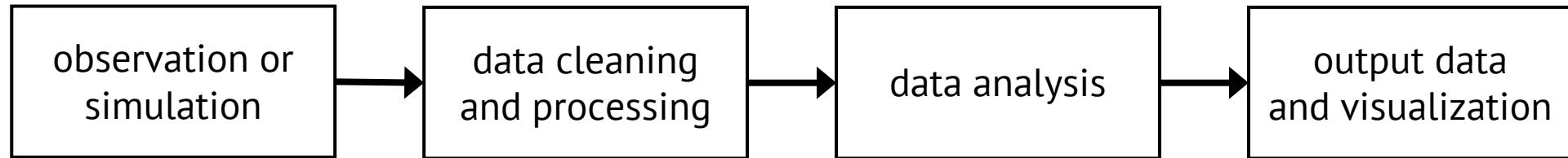
In short, we now unambiguously observe reduced variability when targets are limited. A couple of outliers, in particular stressng's memfd stressor, when limited, is very slow on target machines.

created [experiments/base-vs-limited-targets/all_results.csv](#)
created [experiments/base-vs-limited-targets/all_results.json](#)
created [experiments/base-vs-limited-targets/ansible.log](#)
created [experiments/base-vs-limited-targets/facts/192.168.140.81.json](#)

129 commits loaded



End-to-end Scientific Experimentation Pipelines



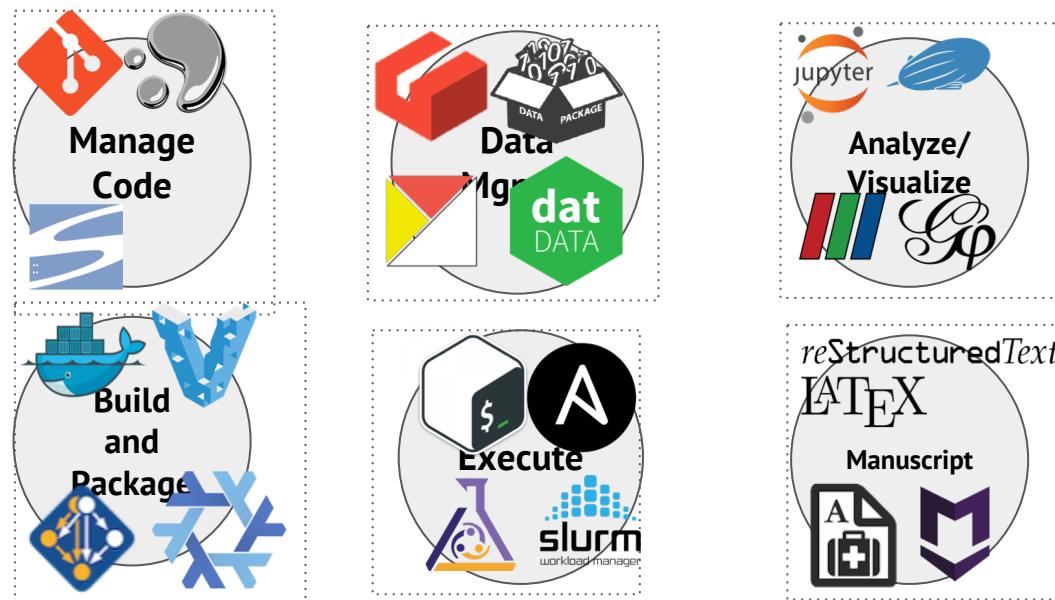
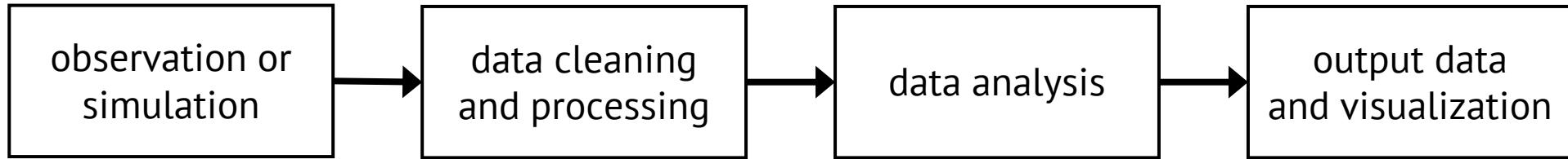
Analogies With Modern SE Practices (aka DevOps)

Scientific exploration	Software project
Experiment code	Source code
Data management	Test examples
Analysis / visualization	Test analysis
Validation	CI / Regression testing
Manuscript / notebook	Documentation / reports

Key Idea: manage a scientific exploration like software projects

SciOps

~~DevOps~~ View of The Experimentation Pipeline



What is DevOps?

Typical



```
ivo@mbp:~          ivo@mbp:~ [44/527]
$ hostname
mbp
ivo@mbp:~          $ date
Tue Jan 31 09:31:14 PST 2017
ivo@mbp:~          $ cat src/popper/popper/README.md
# Popper-CLI

A CLI tool to help bootstrap projects that follow the
[Popper](https://github.com/systemslab/popper) convention.

## Install

Download from
[popper/releases](https://github.com/systemslab/popper/releases). Note
that we have only tested on OSX and Linux (Windows coming soon). Once
downloaded, uncompress and place the binary in a folder that is
included in your '$PATH' (e.g. '/usr/bin').

## Usage

To get an overview and list of commands check out the command line
help:

```bash
popper help
```

ivo@mbp:~          $ find $HOME/tmp -path "*login*"
ivo@mbp:~          $ find $HOME/tmp -path "*git-meme*"
ivo@mbp:~          $ find $HOME/tmp -path "*git-meme.jpg*"
ivo@mbp:~          $ find $HOME/tmp -path "*meme.jpg*"

ivo@mbp:~          $ find $HOME/tmp/ -path "*meme.jpg*"
/Users/ivo/tmp//git-meme.jpg
ivo@mbp:~          $ clear
ivo@mbp:~          1:[tmux]* 2:bash- 3:bash                         mbp
```

DevOps



myscript.sh

\$ bash myscript.sh

The Popper Convention

1. Pick one or more tools from the DevOps toolkit
2. Write scripts for an experiment pipeline
3. Put all scripts in a version control repository



Popper
falsifiable.us

[1]: Jimenez et al. *Standing on the Shoulders of Giants by Managing Scientific Experiments Like Software*, ;login: Winter 2016, Vol. 41, No. 4.

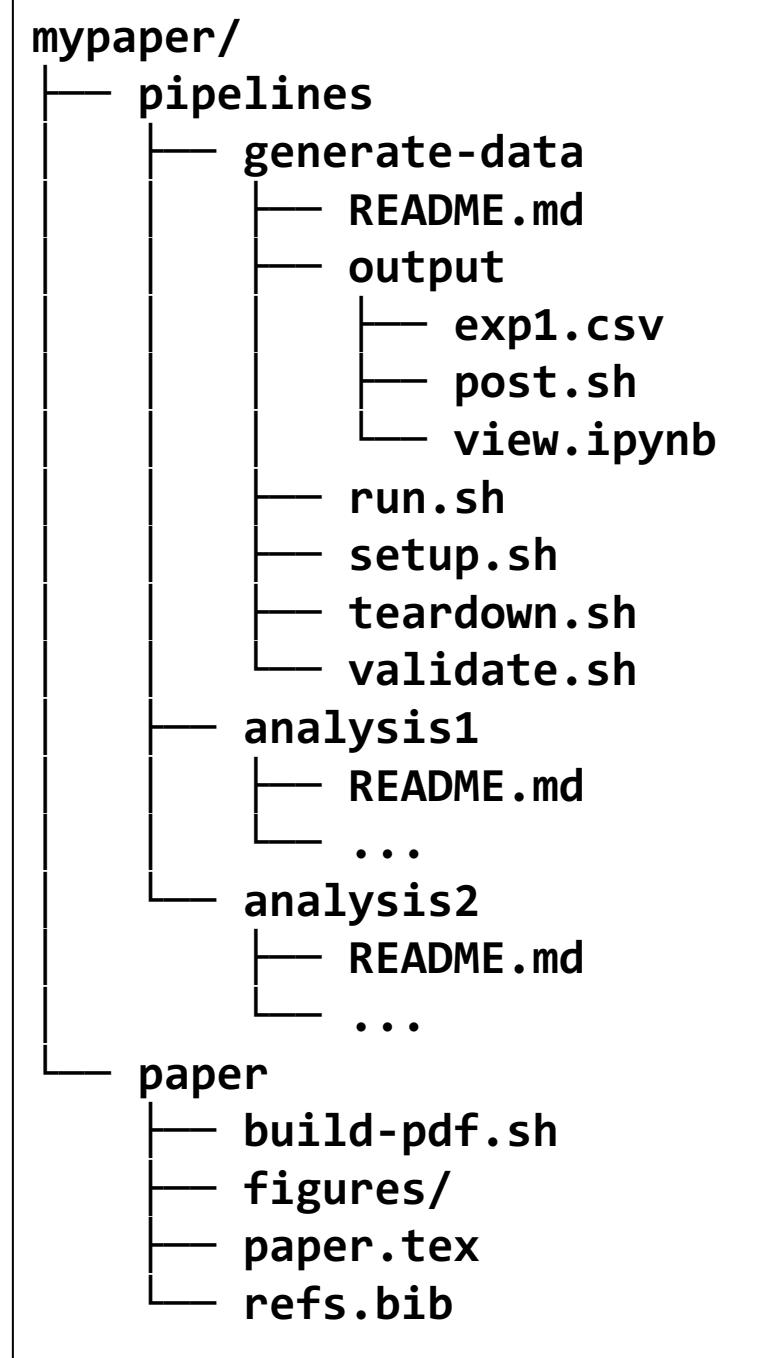
[2]: Jimenez et al. *The Popper Convention: Making Reproducible Systems Evaluation Practical*, REPPAR 2017.

Popper CLI tool



- Make it **super easy** to automate execution and validation of experimentation pipelines
 - easy → low-overhead → more likely it'll be used
- Common convention to organize the contents of a repo
- CLI tool that helps users to implement pipeline stages
- Provide domain-specific examples
 - Today: Genomics, MPI, Ceph, Atmospheric Science
 - Working with domain-experts to contribute more examples

Common convention to organize the contents of a repo



CLI tool

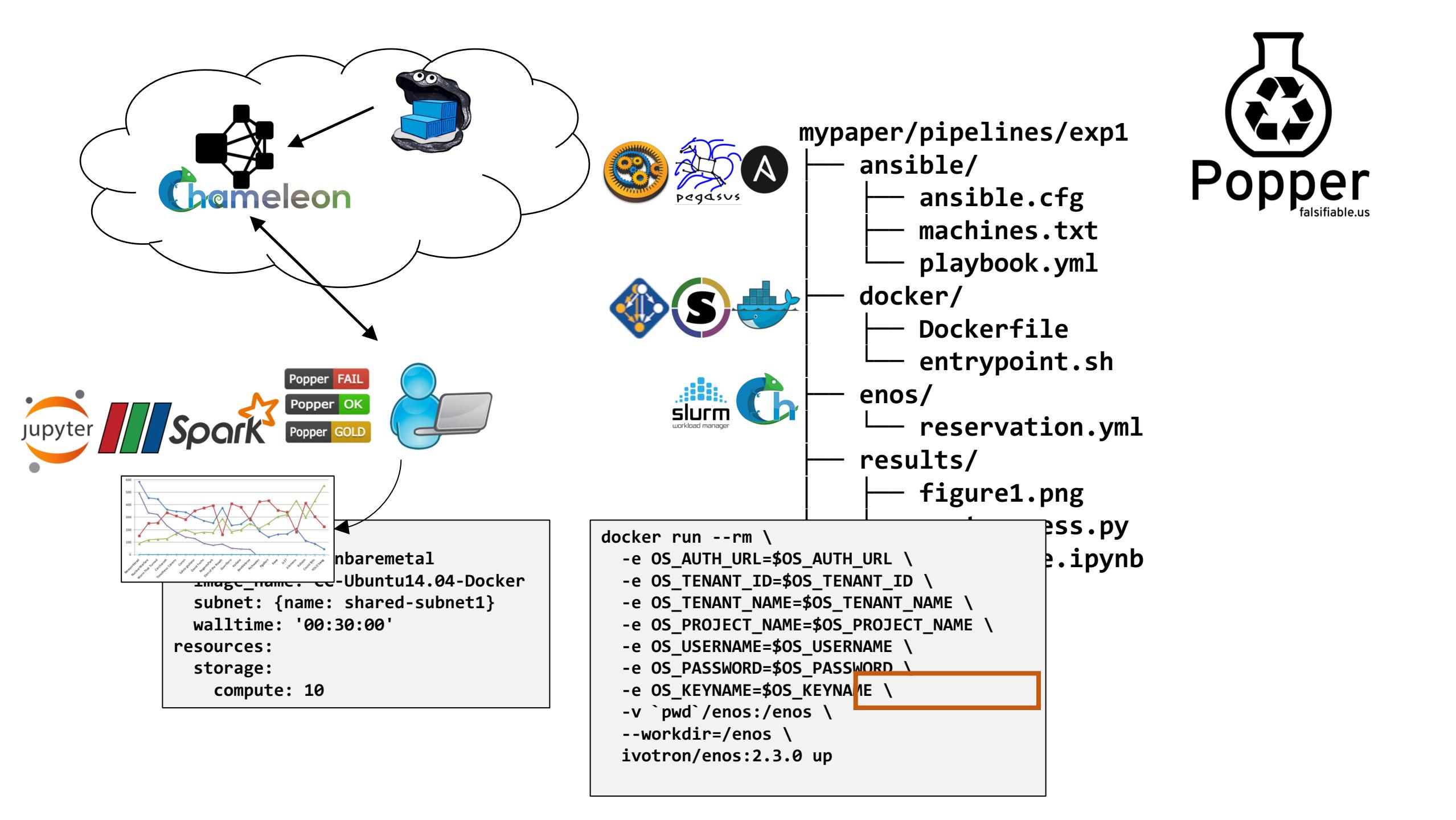
```
$ cd my-paper-repo
$ git init
Initialized empty Git repository in my-paper-repo/.git
$ popper init
Initialized popper repository.
$ popper pipeline init mypipeline --stages=prepare,execute,cleanup
-- Initialized exp1 pipeline.
```

```
$ ls -l pipelines/mypipeline
total 20K
-rw-r----- 1 ivo ivo 8 Apr 29 23:58 README.md
-rwxr-x--- 1 ivo ivo 210 Apr 29 23:58 execute.sh
-rwxr-x--- 1 ivo ivo 206 Apr 29 23:58 prepare.sh
-rwxr-x--- 1 ivo ivo 61 Apr 29 23:58 cleanup.sh
```

```
#!/bin/bash

# trigger execution of experiment
docker run google/kubectl run ...
.

...
```



```
$ popper run exp1
```

Popper run started

Stage: setup.sh

Stage: run.sh
.....

Stage: validate.sh .
.

Stage: teardown.sh ...

Popper run finished

Status: OK

Popper FAIL

Popper OK

Popper GOLD

Codified Validations

num_nodes, throughput, raw_bw, net_saturated

```
Src,Egid,Version,Datetime,Lat,Lon,Magnitude,Depth,NST,Region  
ci,14692356,1,"Tuesday, May 4, 2010 03:21:38 UTC",32.6443,-11  
ci,14692348,1,"Tuesday, May 4, 2010 03:19:38 UTC",32.1998,-11  
ci,14692332,1,"Tuesday, May 4, 2010 03:16:56 UTC",32.6756,-11  
ci,14692324,1,"Tuesday, May 4, 2010 03:08:47 UTC",32.6763,-11  
ci,14692316,1,"Tuesday, May 4, 2010 03:08:08 UTC",32.6778,-11  
ci,14692308,1,"Tuesday, May 4, 2010 03:06:20 UTC",32.7071,-11  
ci,14692300,1,"Tuesday, May 4, 2010 03:01:52 UTC",32.1948,-11  
ak,10047267,1,"Tuesday, May 4, 2010 03:01:04 UTC",61.2695,-14  
ci,14692284,1,"Tuesday, May 4, 2010 02:58:51 UTC",32.7016,-11  
ci,14692276,1,"Tuesday, May 4, 2010 02:57:46 UTC",32.6998,-11  
ak,10047263,1,"Tuesday, May 4, 2010 02:56:28 UTC",63.5779,-11  
ak,10047261,1,"Tuesday, May 4, 2010 02:52:00 UTC",66.4986,-14  
ci,14692268,1,"Tuesday, May 4, 2010 02:48:40 UTC",32.6813,-11  
ci,14692260,1,"Tuesday, May 4, 2010 02:35:27 UTC",32.2006,-11  
nc,71392116,0,"Tuesday, May 4, 2010 02:15:24 UTC",38.8415,-11  
ci,14692244,1,"Tuesday, May 4, 2010 02:05:07 UTC",33.5248,-11  
ci,14692228,1,"Tuesday, May 4, 2010 01:57:08 UTC",32.6823,-11  
ci,14692220,1,"Tuesday, May 4, 2010 01:53:28 UTC",32.6881,-11  
ci,14692212,1,"Tuesday, May 4, 2010 01:48:53 UTC",32.6398,-11  
ci,14692188,1,"Tuesday, May 4, 2010 01:26:58 UTC",32.5003,-11  
ci,14692180,1,"Tuesday, May 4, 2010 01:19:44 UTC",32.6836,-11  
ci,14692172,1,"Tuesday, May 4, 2010 01:12:01 UTC",32.5321,-11  
ci,14692164,1,"Tuesday, May 4, 2010 01:08:24 UTC",32.6833,-11
```

```
Stage: run.sh .....  
Stage: validate.sh ....
```

```
[true] check linear scalability  
[true] check system throughput
```

```
Popper run finished
```

```
Status: GOLD
```



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

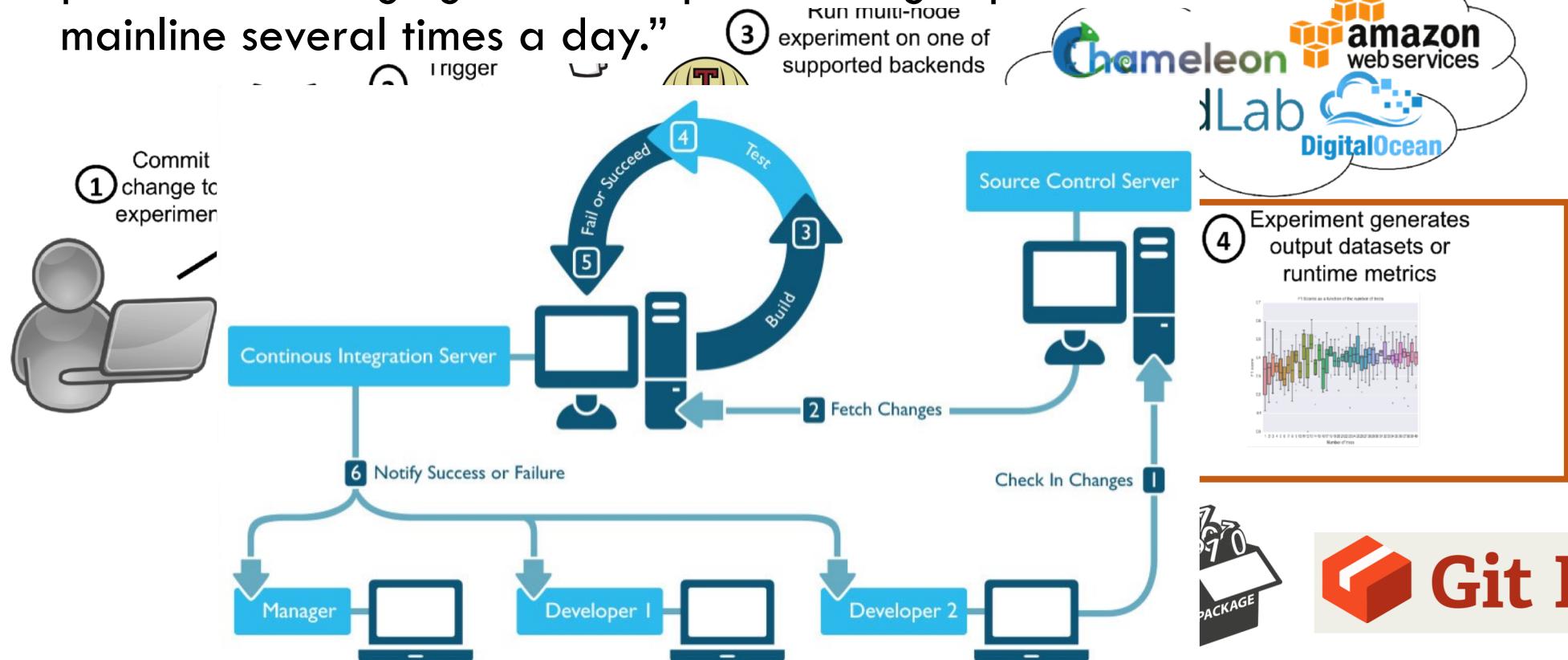
```
when  
not net_saturated  
expect  
throughput >= (raw_bw * 0.9)
```

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

[2]: Jimenez et al. *I Aver: Providing Declarative Experiment Specifications Facilitates the Evaluation of Computer Systems Research*, TinyTOCS, Vol. 3,

Popper and CI

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



Push-button Reproducible Evaluation

source: <https://insights.sfu.edu/devops/2015/01/continuous-integration-in-devops.html>

ACM/Popper Badges



| Result Status | Artifacts | Re-executed By | ACM | Popper |
|----------------------|------------------|-----------------------|---|--------------------|
| Repeatability | Original | Original Author(s) | | Popper GOLD |
| Replicability | Original | 3 rd Party |  | Popper GOLD |
| Reproducibility | Re-implemented | Anyone |  | |

Archiving/DOI service integration

```
$ popper archive --zenodo --user=ivotron --password=****  
Creating archive for repository on Zenodo.  
| #####| 100 %
```

Your DOI link is: <https://zenodo.org/record/1165550>



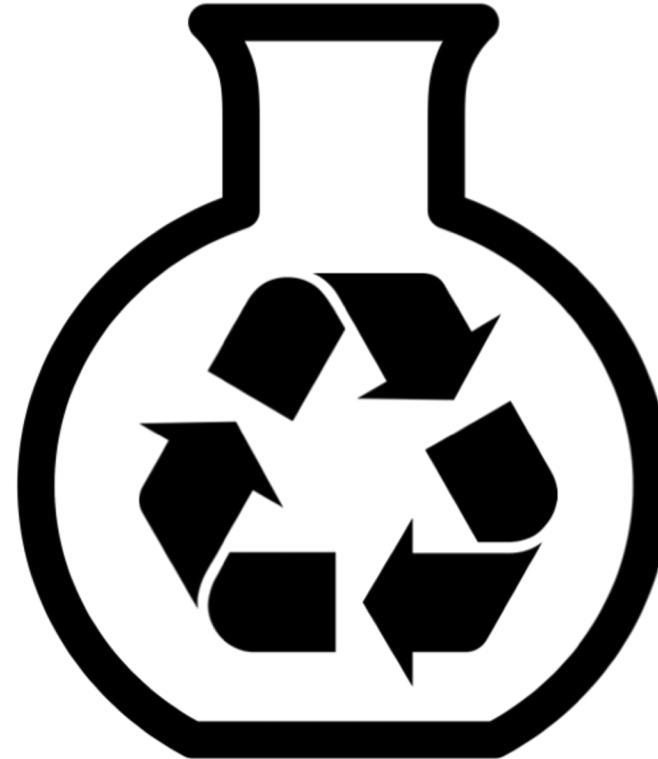
Push-button Reproducible Evaluation ⇒ New Possibilities and Challenges

New:

- SciOps approach. End-to-end automated execution & validation.
- Improve the study of computer systems. Portability allows to fix the SW stack; we can now easily report results w.r.t. distinct environments.

Challenges:

- Larger search space: entire software stack can be parameterized.
- Cannot execute on new platforms (hardware does not exist yet).
- Identifying root causes of irreproducibility.



Popper
falsifiable.us