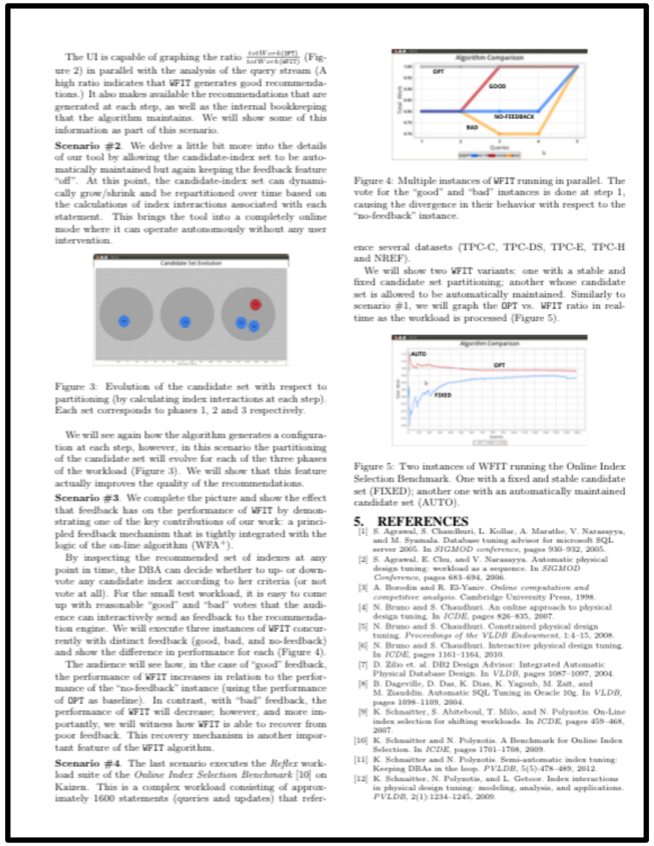
Popper
falsifiable.us

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 $\frac{totWork(OPT)}{totWork(WFIT)}$ (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 keeping the feedback feature "off". At this point, the candidate-index set can dynamically 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.

We will see again how the algorithm generates a configuration at each step, however, in this scenario the partitioning of the candidate set will evolve for each of the three phases of the workload (Figure 3). We will show that this feature actually improves the quality of the recommendations.

Scenario #3. We complete the picture and show the effect that feedback has on the performance of WFIT by demonstrating one of the key contributions of our work: a principled feedback mechanism that is tightly integrated with the logic of the on-line algorithm (WFA$^+$).

By inspecting the recommended set of indexes at any point in time, the DBA can decide whether to up- or down-vote any candidate index according to her criteria (or not vote at all). For the small test workload, it is easy to come up with reasonable "good" and "bad" votes that the audience can interactively send as feedback to the recommendation engine. We will execute three instances of WFIT concurrently with distinct feedback (good, bad, and no-feedback) and show the difference in performance for each (Figure 4).

The audience will see how, in the case of "good" feedback, the performance of WFIT increases in relation to the performance of the "no-feedback" instance (using the performance of OPT as baseline). In contrast, with "bad" feedback, the performance of WFIT will decrease; however, and more importantly, we will witness how WFIT is able to recover from poor feedback. This recovery mechanism is another important feature of the WFIT algorithm.

Scenario #4. The last scenario executes the *Reflex* workload suite of the *Online Index Selection Benchmark* [10] on Kaizen. This is a complex workload consisting of approximately 1600 statements (queries and updates) that reference several datasets (TPC-C, TPC-DS, TPC-E, TPC-H and NREF).

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

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).

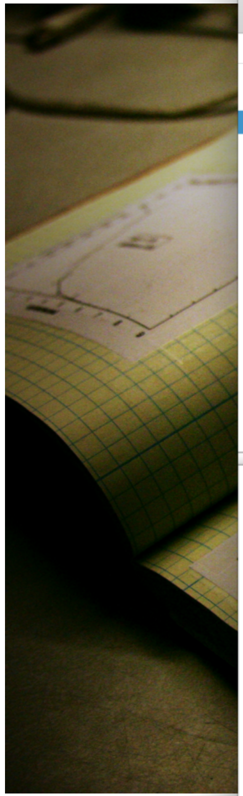
Figure 5: Two instances of WFIT running the Online Index Selection Benchmark. One with a fixed and stable candidate set (FIXED); another one with an automatically maintained candidate set (AUTO).

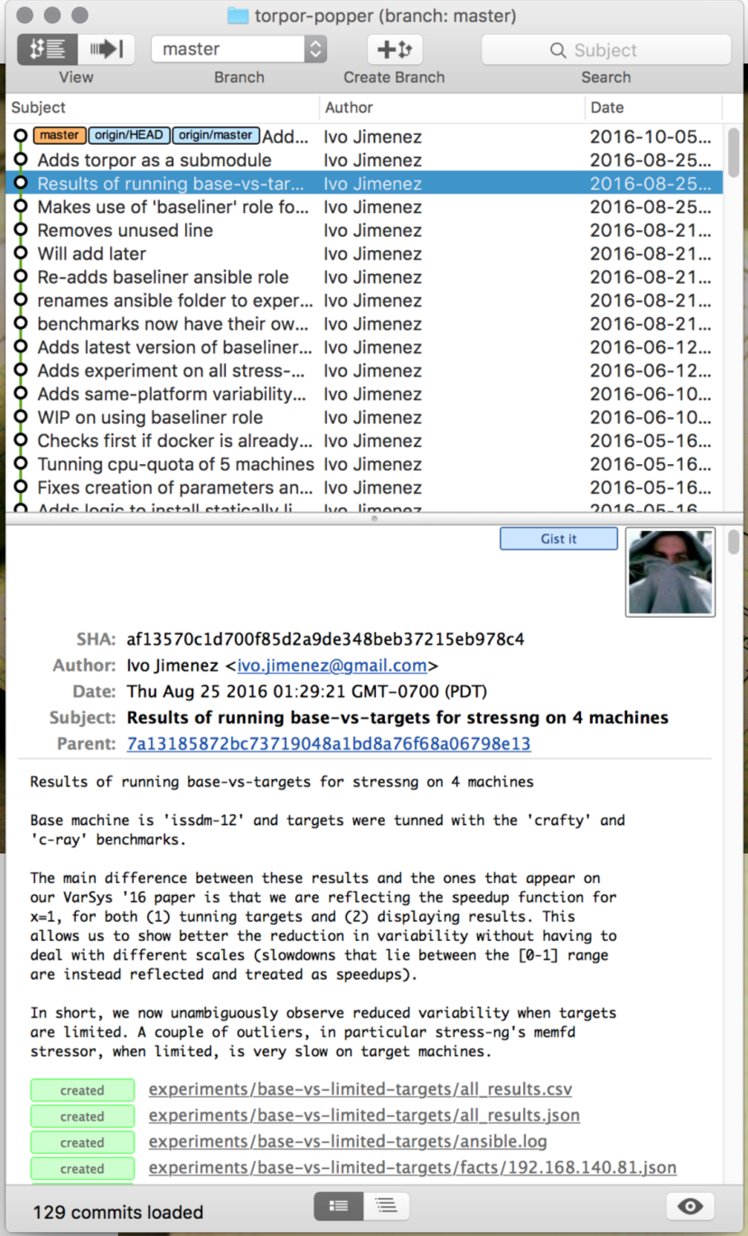
5. REFERENCES

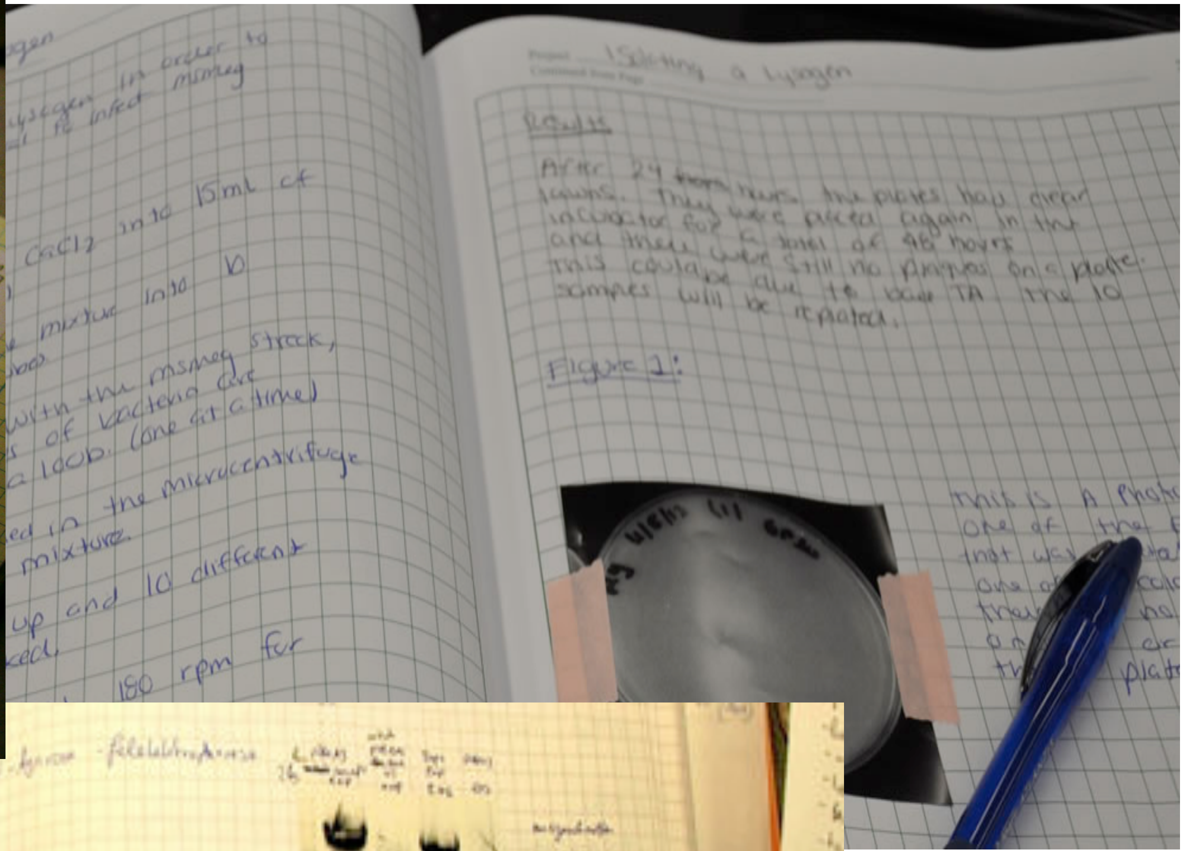
[1] S. Agrawal, S. Chaudhuri, L. Kollar, A. Marathe, V. Narasayya, and M. Syamala. Database tuning advisor for microsoft SQL server 2005. In *SIGMOD conference*, pages 930–932, 2005.
[2] S. Agrawal, E. Chu, and V. Narasayya. Automatic physical design tuning: workload as a sequence. In *SIGMOD Conference*, pages 683–694, 2006.
[3] A. Borodin and R. El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, 1998.
[4] N. Bruno and S. Chaudhuri. An online approach to physical design tuning. In *ICDE*, pages 826–835, 2007.
[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, 2010.
[7] D. Zilio et. al. DB2 Design Advisor: Integrated Automatic Physical Database Design. In *VLDB*, pages 1087–1097, 2004.
[8] B. Dageville, D. Das, K. Dias, K. Yagoub, M. Zait, and M. Ziauddin. Automatic SQL Tuning in Oracle 10g. In *VLDB*, pages 1098–1109, 2004.
[9] K. Schnaitter, S. Abiteboul, T. Milo, and N. Polyzotis. On-Line index selection for shifting workloads. In *ICDE*, pages 459–468, 2007.
[10] K. Schnaitter and N. Polyzotis. A Benchmark for Online Index Selection. In *ICDE*, pages 1701–1708, 2009.
[11] K. Schnaitter and N. Polyzotis. Semi-automatic index tuning: Keeping DBAs in the loop. *PVLDB*, 5(5):478–489, 2012.
[12] K. Schnaitter, N. Polyzotis, and L. Getoor. Index interactions in physical design tuning: modeling, analysis, and applications. *PVLDB*, 2(1):1234–1245, 2009.

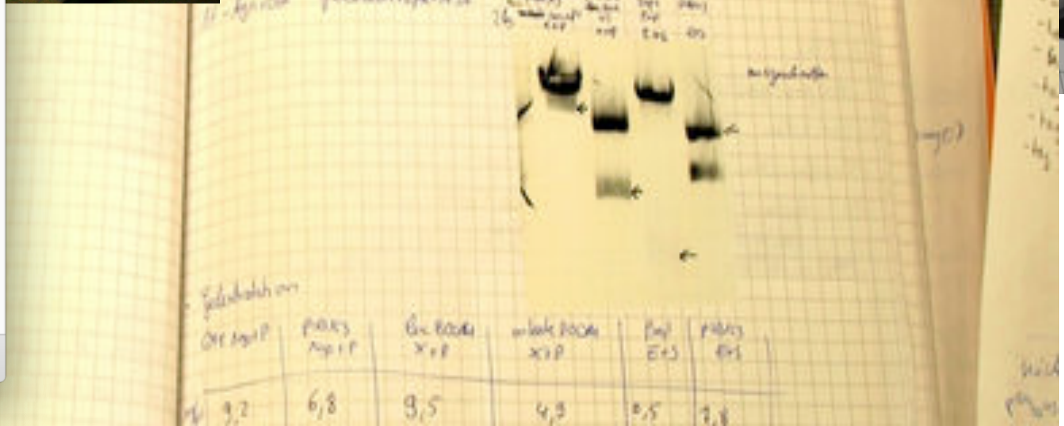
- What compiler was used?
- Which compilation flags?
- How was subsystem X configured?
- How does the workload look like?
- What parameters can be modified?
- What if I use input dataset Y?
- And if I run on platform Z?
- …

Lab Notebook

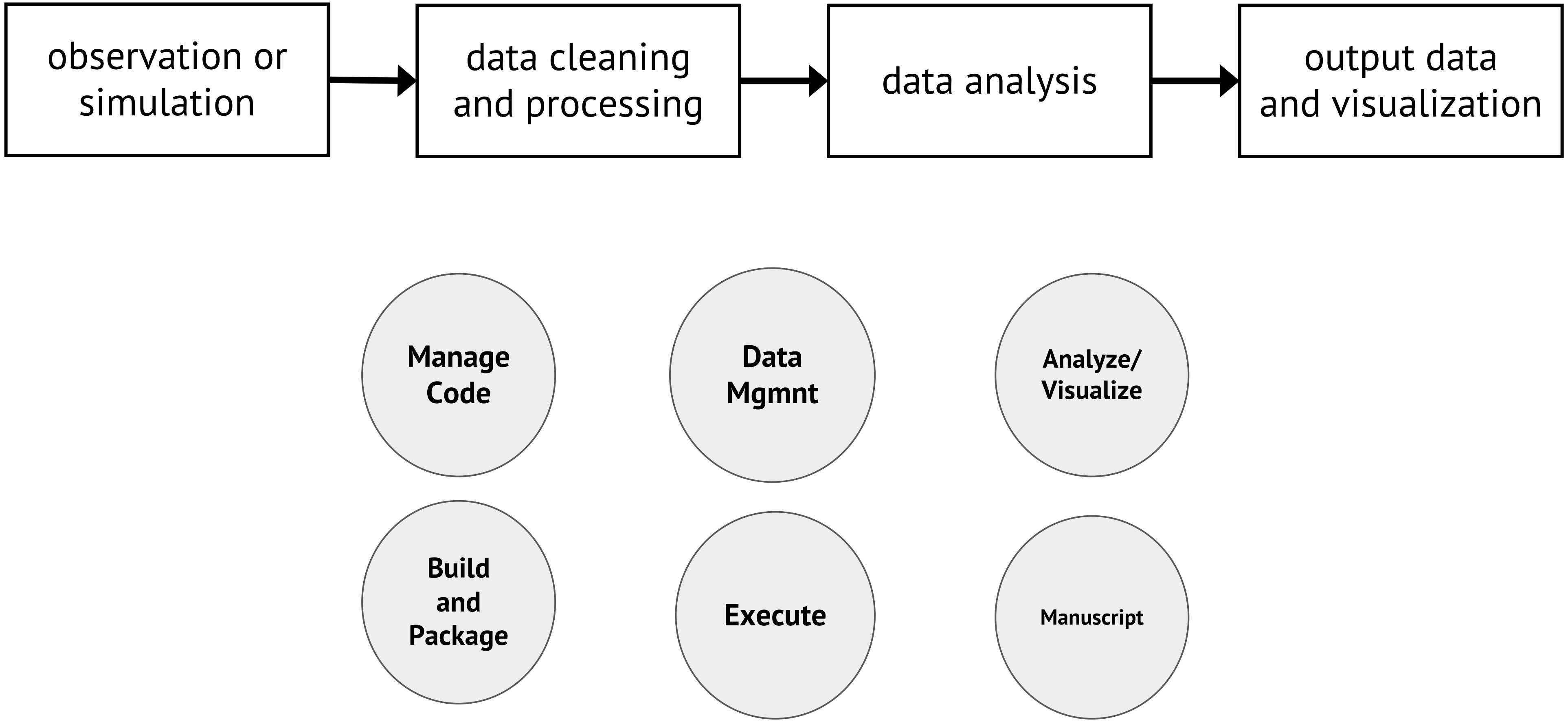








End-to-end Scientific Experimentation Pipelines

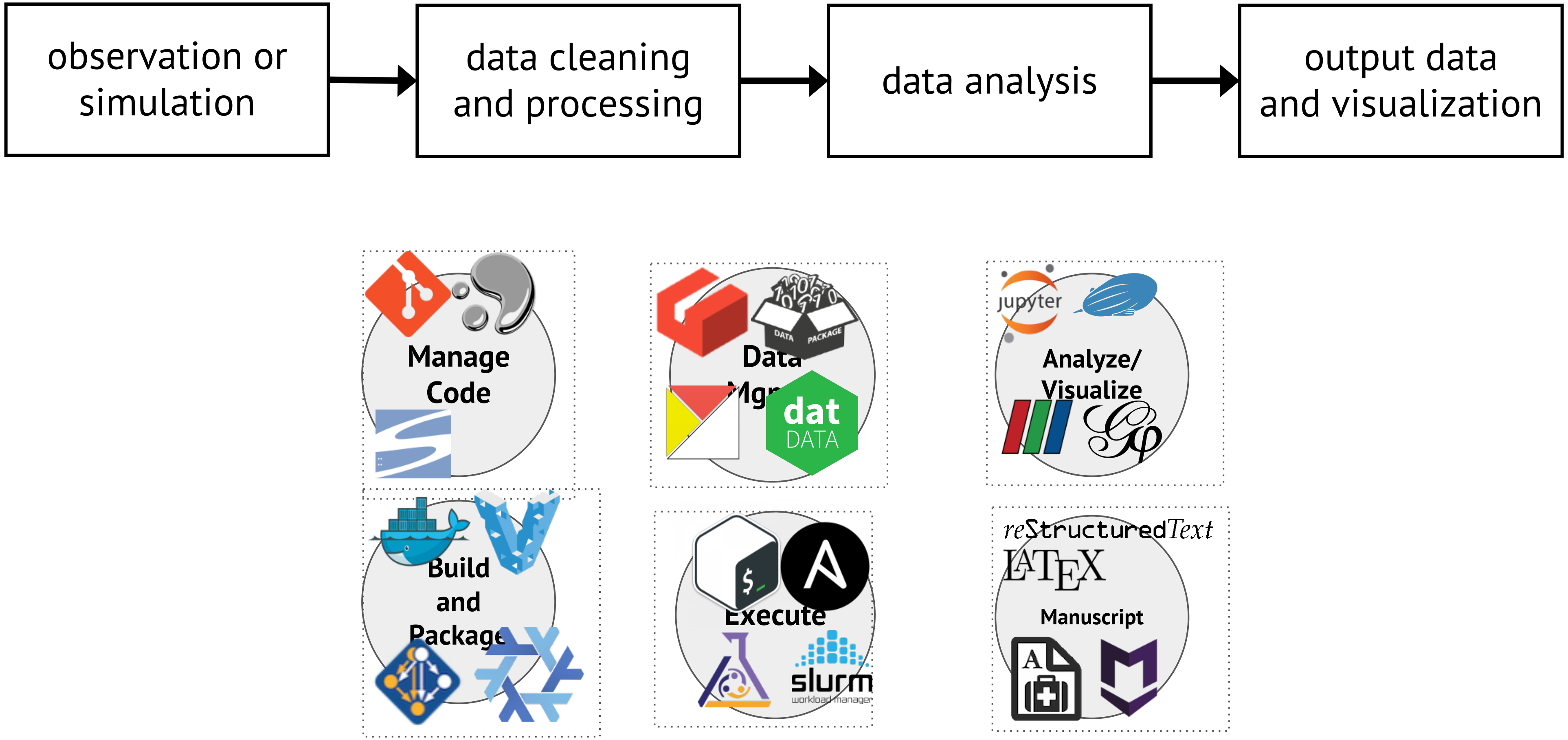


Analogies With Modern SE Practices (aka DevOps)

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

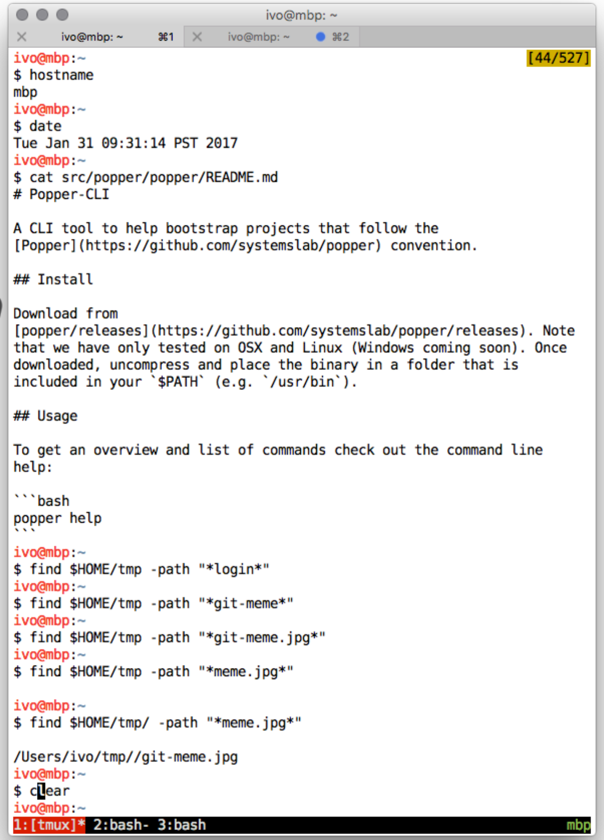
Key Idea: manage a scientific exploration like software projects

SciOps
~~DevOps~~ View of The Experimentation Pipeline



What is DevOps?

Typical



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, Athmospheric Science
  - Working with domain-experts to contribute more examples

Common convention to organize the contents of a repo

```
mypaper/
├── pipelines
│   ├── generate-data
│   │   ├── README.md
│   │   ├── output
│   │   │   ├── exp1.csv
│   │   │   ├── post.sh
│   │   │   └── view.ipynb
│   │   ├── run.sh
│   │   ├── setup.sh
│   │   ├── teardown.sh
│   │   └── validate.sh
│   ├── analysis1
│   │   ├── README.md
│   │   └── ...
│   └── analysis2
│       ├── README.md
│       └── ...
└── paper
    ├── build-pdf.sh
    ├── figures/
    ├── paper.tex
    └── refs.bib
```

Popper
falsifiable.us

CLI tool

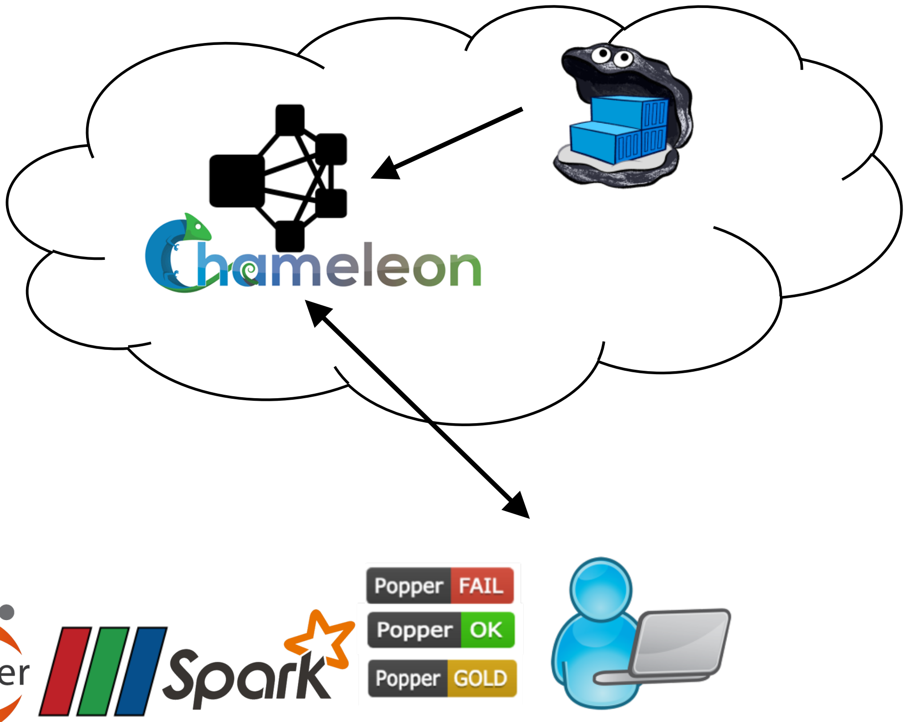
```
$ cd my-paper-repo
$ git init
Initialized empty Git repository in my-paper-repo/.git
$ popper init
Initialized popper repository.
$ popper pipeline init mypipeline --stages=prepare,execute,cleanup
-- Initialized exp1 pipeline.

$ 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 ...

...
```



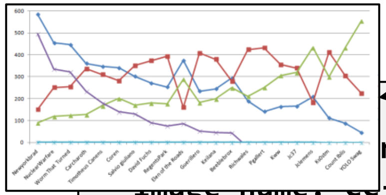

```
mypaper/pipelines/exp1
├── ansible/
│   ├── ansible.cfg
│   ├── machines.txt
│   └── playbook.yml
├── docker/
│   ├── Dockerfile
│   └── entrypoint.sh
├── enos/
│   └── reservation.yml
├── results/
│   ├── figure1.png
│   ├── ess.py
    e.ipynb
```

Popper falsifiable.us

Popper FAIL

Popper OK

Popper GOLD



```
nbaremetal
-Ubuntu14.04-Docker
  subnet: {name: shared-subnet1}
  walltime: '00:30:00'
resources:
  storage:
    compute: 10
```

```
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 run exp1

Popper run started

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

Popper run finished

Status: OK
```

Codified Validations

Aver[1,2]

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

```
Src,Eqid,Version,Datetime,Lat,Lon,Magnitude,Depth,NST,Region
ci,14692356,1,"Tuesday, May  4, 2010 03:21:38 UTC",32.6443,-1
ci,14692348,1,"Tuesday, May  4, 2010 03:19:38 UTC",32.1998,-1
ci,14692332,1,"Tuesday, May  4, 2010 03:16:56 UTC",32.6756,-1
ci,14692324,1,"Tuesday, May  4, 2010 03:08:47 UTC",32.6763,-1
ci,14692316,1,"Tuesday, May  4, 2010 03:08:08 UTC",32.6778,-1
ci,14692308,1,"Tuesday, May  4, 2010 03:06:20 UTC",32.7071,-1
ci,14692300,1,"Tuesday, May  4, 2010 03:01:52 UTC",32.1948,-1
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,-1
ci,14692276,1,"Tuesday, May  4, 2010 02:57:46 UTC",32.6998,-1
ak,10047263,1,"Tuesday, May  4, 2010 02:56:28 UTC",63.5779,-15
ak,10047261,1,"Tuesday, May  4, 2010 02:52:00 UTC",60.4986,-14
ci,14692268,1,"Tuesday, May  4, 2010 02:48:40 UTC",32.6813,-1
ci,14692260,1,"Tuesday, May  4, 2010 02:35:27 UTC",32.2006,-1
nc,71392116,0,"Tuesday, May  4, 2010 02:15:24 UTC",38.8415,-12
ci,14692244,1,"Tuesday, May  4, 2010 02:05:07 UTC",33.5248,-1
ci,14692228,1,"Tuesday, May  4, 2010 01:57:08 UTC",32.6823,-1
ci,14692220,1,"Tuesday, May  4, 2010 01:53:28 UTC",32.6881,-1
ci,14692212,1,"Tuesday, May  4, 2010 01:48:53 UTC",32.6398,-1
ci,14692188,1,"Tuesday, May  4, 2010 01:26:58 UTC",32.5003,-1
ci,14692180,1,"Tuesday, May  4, 2010 01:19:44 UTC",32.6836,-1
ci,14692172,1,"Tuesday, May  4, 2010 01:12:01 UTC",32.5321,-1
ci,14692164,1,"Tuesday, May  4, 2010 01:08:24 UTC",32.6833,-1
```

- Log file
- CSV
- DB Table
- TSDB
- …

```
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,.

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
```

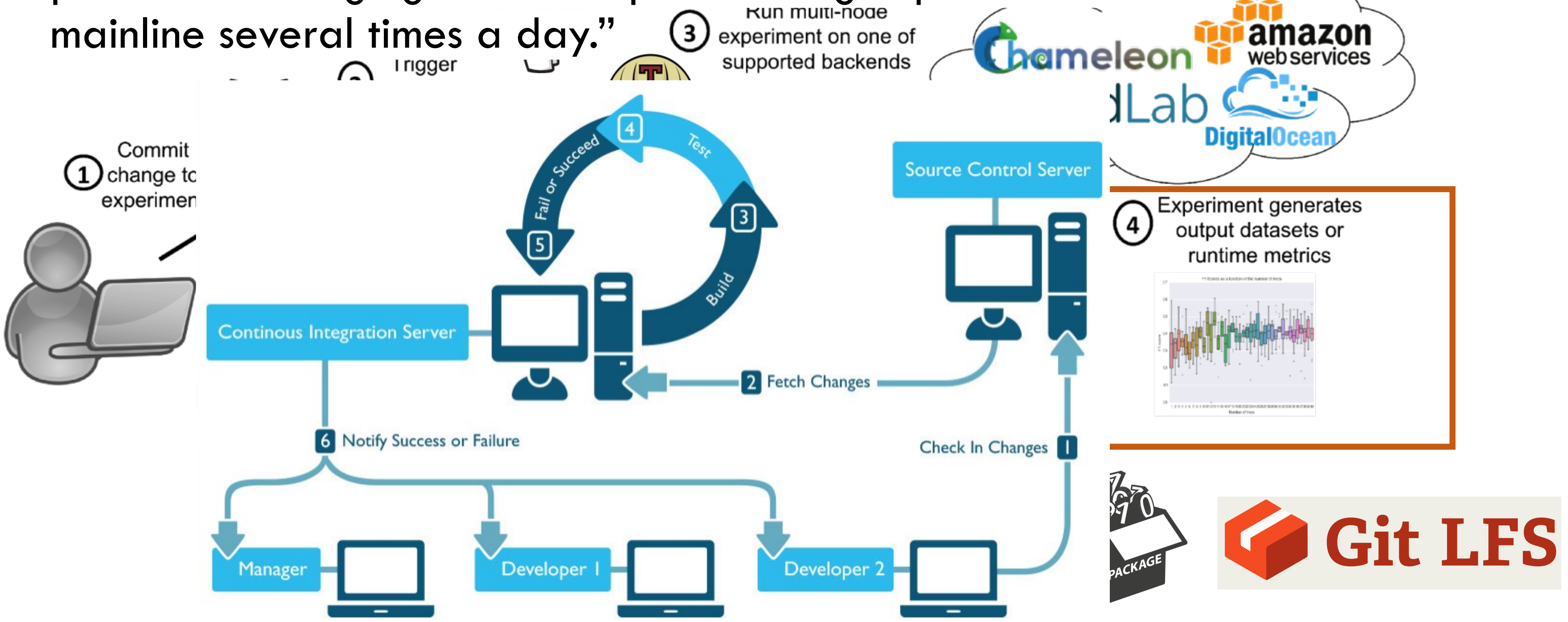
zenodo figshare Open Science Framework DRYAD

ACM/Popper Badges

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

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."



source: https://insights.sei.cmu.edu/devops/2015/01/continuous-integration-in-devops-1.html

Push-button Reproducible *Evaluation*

Popper

falsifiable.us