The Popper Convention: Practical Reproducible Evaluation of Systems

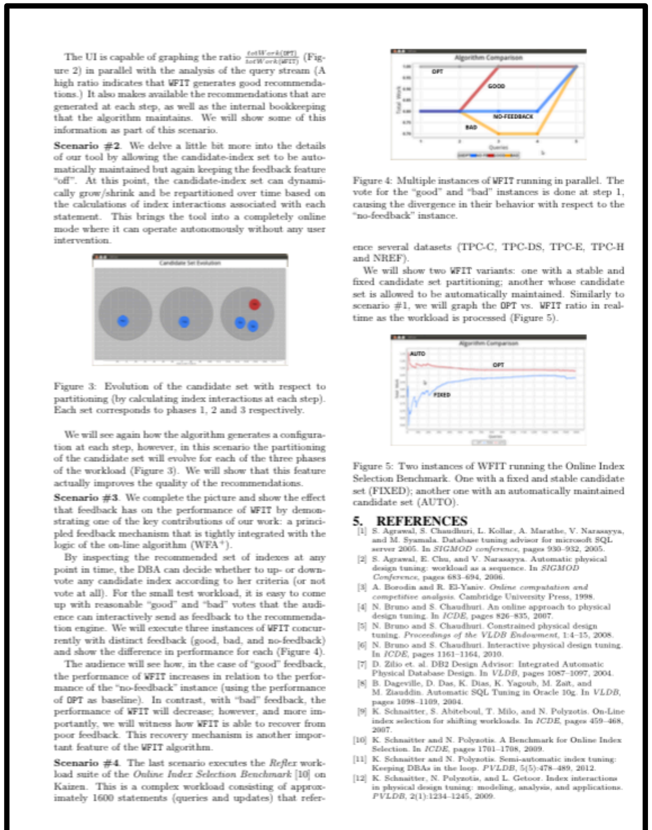
Ivo Jimenez, Michael Sevilla, Noah Watkins, Carlos Maltzahn (UC Santa Cruz)

Jay Lofstead (SNL)

Kathryn Mohror, Adam Moody (LLNL)

Andrea Arpaci-Dusseau, Remzi Arpaci-Dusseau (UW-M)

Problem of Reproducibility in Computation and Data Exploration



The UI is capable of graphing the ratio $\frac{totWork(\texttt{OPT})}{totWork(\texttt{WFIT})}$ (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.

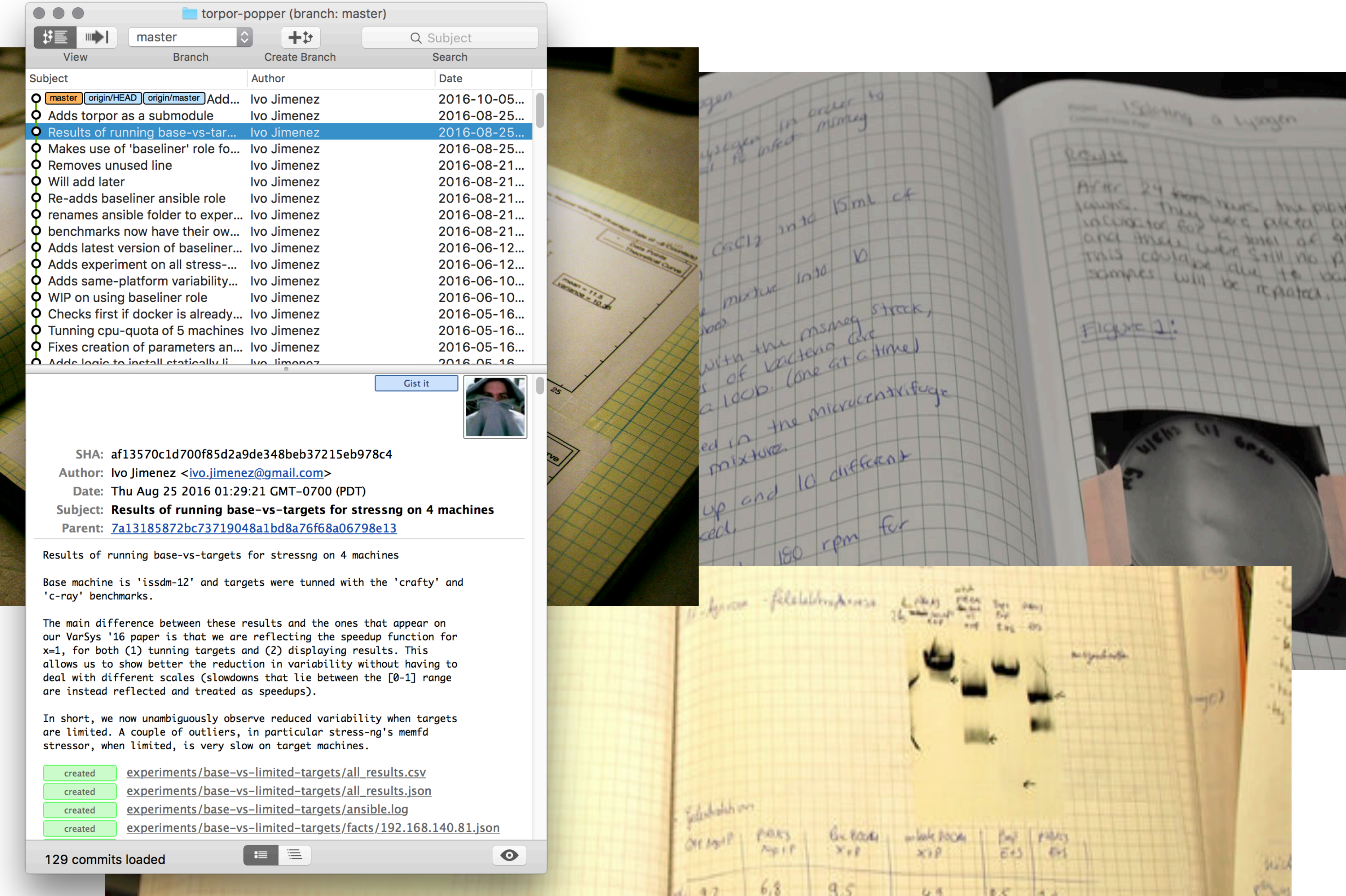
[10] K. Schnaitter and N. Polyzotis. A Benchmark for Online Index Selection. In *ICDE*, pages 1701–1708, 2009.

[11] K. Schnaitter and N. Polyzotis. Semi-automatic index tuning: Keeping DBAs in the loop. *PVLDB*, 5(5):478–489, 2012.

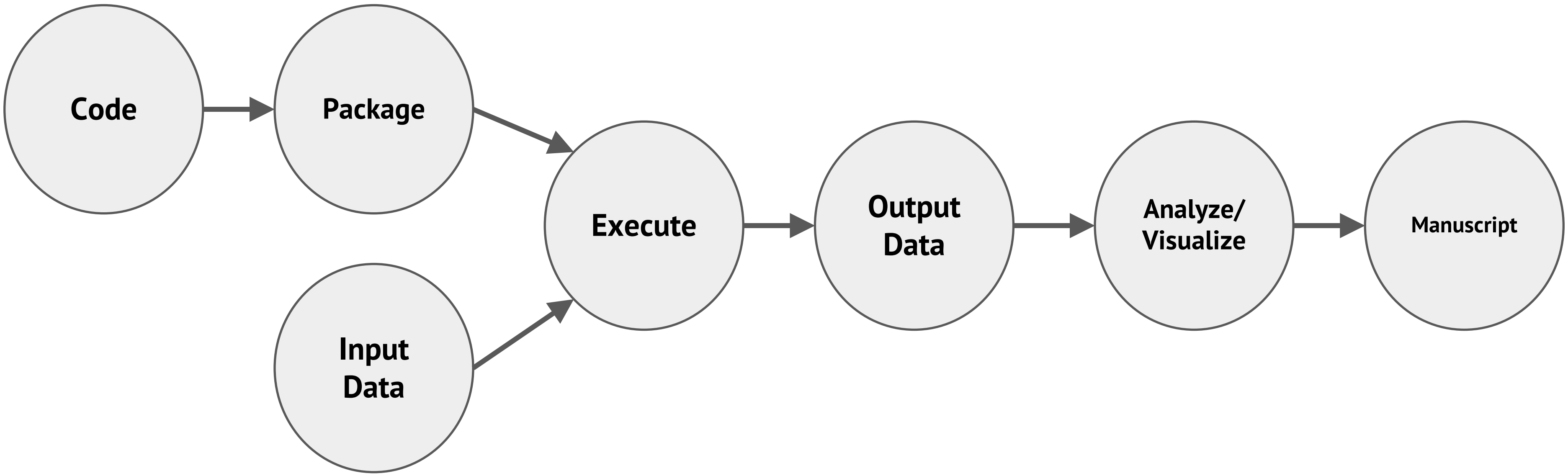
[12] K. Schnaitter, N. Polyzotis, and L. Getoor. Index interactions in physical design tuning: modeling, analysis, and applications. *PVLDB*, 2(1):1234–1245, 2009.

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

Lab Notebook

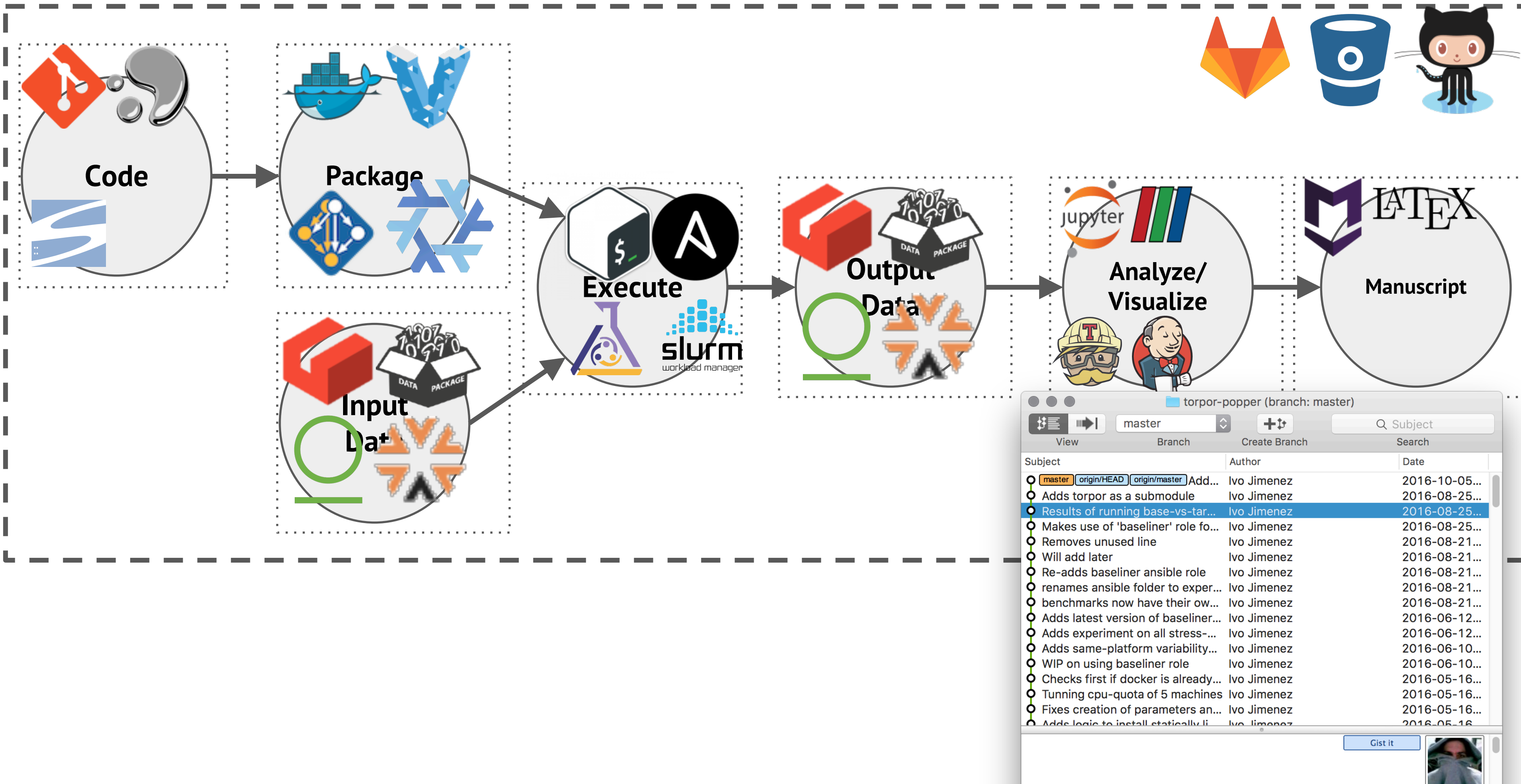


Common Experimentation Workflow



Popper

falsifiable.us



Popper
falsifiable.us

1. Pick a DevOps tool for each stage.
   – Each component of experimentation workflow.
2. Put all associated scripts in version control.
   – Make experiment self-contained.
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) **and** running correctly:
  - Experiment code
  - Orchestration
  - Data dependencies
  - Parameterization
  - Results
  - Validation

Popper-CLI

```
$ cd mypaper-repo
$ popper init
-- Initialized Popper project mypaper-repo

$ popper experiment list
-- available templates ---------------
ceph-rados   proteustm  mpip        adam    sirius    comd-openmp
cloverleaf   gassyfs    zlog        bww     unum-py   cuddn-deeplrn
spark-bench  torpor     malacology  genevo  mantle    rita-idx
hadoop-yarn  kubsched   alg-encycl  macrob  dadvisor  obfuscdata

$ popper experiment add gassyfs
-- Added gassyfs experiment to mypaper-repo

$ popper experiment init mynewexp
-- Initialized mynewexp experiment in mypaper-repo
```

Popper

falsifiable.us

Automated Validation

Branch: master ▾ **torpor-popper / experiments / cgroups-cpuquota /**

Create new file | Upload files | Find file | History

ivotron committed on **GitHub** Update README.md — Latest commit `96b1c0a` 10 minutes ago

..

.popper.yml	empty experiment	12 hours ago
README.md	Update README.md	10 minutes ago
merge_results	renames ansible folder to experiments	3 months ago
playbook.yml	renames ansible folder to experiments	3 months ago
run.sh	empty experiment	12 hours ago
uncompress	renames ansible folder to experiments	3 months ago
validate.sh	empty experiment	12 hours ago

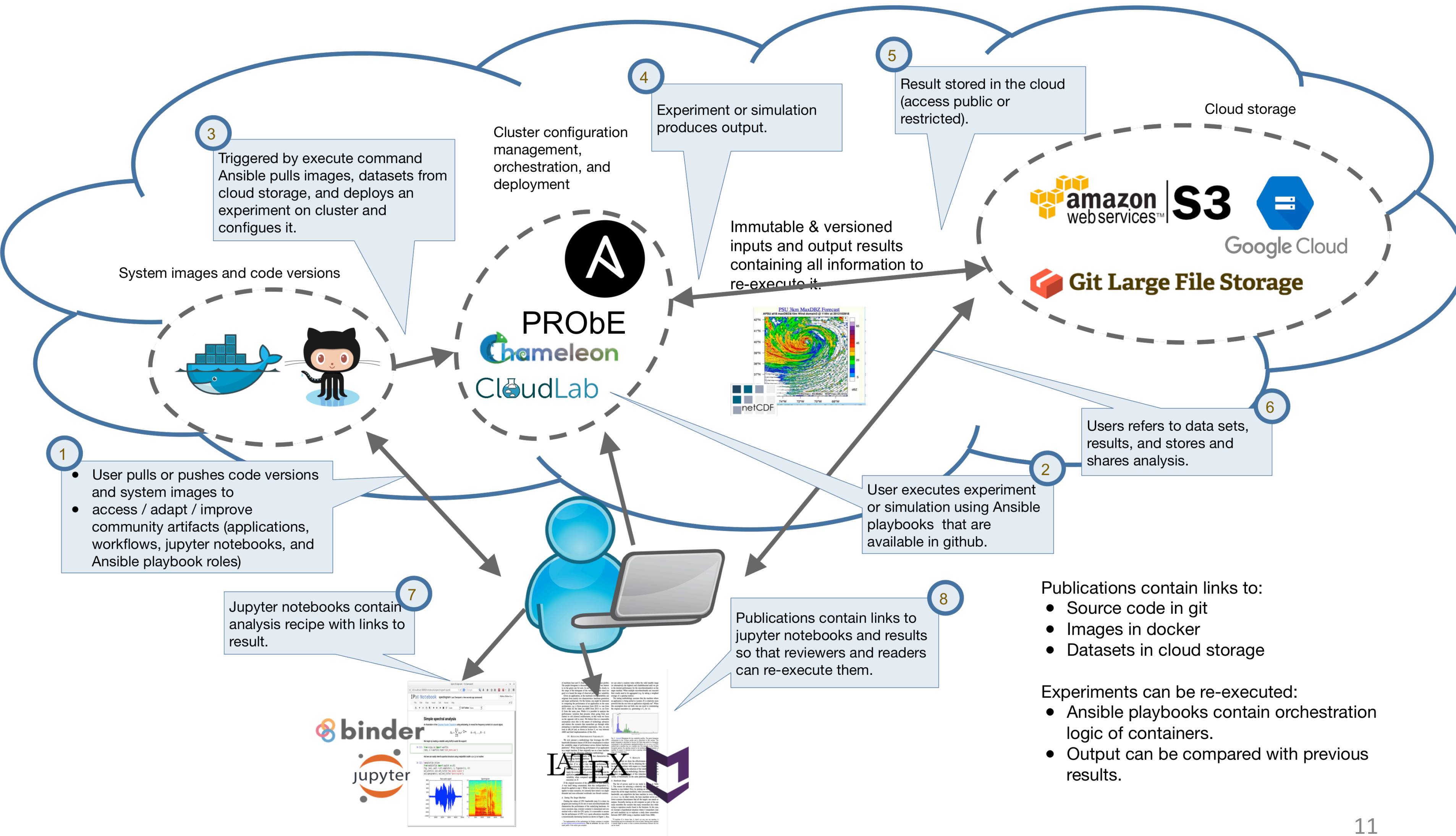
README.md

cgroups CPU subsystem experiment

Popper GOLD

This experiment evaluates the effectiveness of modifications to the cgroups cpu subsystem.

Reviewer/Reader Workflow



Publications contain links to:

- Source code in git
- Images in docker
- Datasets in cloud storage

Experiments can be re-executed:

- Ansible playbooks contain orchestration logic of containers.
- Output can be compared with previous results.

Other Use Cases

- Parallel Algorithms Encyclopedia
- ctuning extended artifact description
- HPC Proxy applications (mini-apps)
- Elsevier's 2011 executable paper challenge

Communities

- Numerical weather prediction as part of the Big Weather Web (bigweatherweb.org)
- Distributed Systems (UCSC / UW Madison)
- Game design as part of the generative methods effort at the (UCSC Augmented Design Lab)
- HPC at LLNL and Sandia
- Genomics at UCSC

Analogies with DevOps Practice

Scientific exploration	Software project
Experiment code	Source code
Input data	Test examples
Analysis / visualization	Test analysis
Validation	CI / Regression testing
Manuscript / note book	Documentation / reports

Key Idea: manage a scientific exploration like software projects