Tackling the Reproducibility Problem in Systems Research with Declarative Experiment Specifications

Ivo Jimenez, Carlos Maltzahn
UC Santa Cruz
{ivo,carlosm}@cs.ucsc.edu

Jay Lofstead
Sandia National Lab
gflofst@sandia.gov

Adam Moody, Kathryn Mohror
Lawrence Livermore National Lab
{moody20,kathryn}@llnl.gov

Remzi Arpaci-Dusseau, Andrea Arpaci-Dusseau
UW Madison
{remzi,dusseau}@cs.wisc.edu

Abstract

Validating experimental results in the field of computer systems is a challenging task, mainly due to the many changes in software and hardware that computational environments go through. Determining if an experiment is reproducible entails two separate tasks: re-executing the experiment and validating the results. Existing reproducibility efforts have focused on the former, envisioning techniques and infrastructures that make it easier to re-execute an experiment. In this work we focus on the latter by analyzing the validation workflow that an experiment re-executioner goes through. We notice that validating results is done on the basis of experiment design and high-level goals, rather than exact quantitative metrics. Based on this insight, we introduce a declarative format for specifying the high-level components of an experiment as well as describing generic, testable conditions that serve as the basis for validation. We present a use case in the area of storage systems to illustrate the usefulness of this approach. We also discuss limitations and potential benefits of using this approach in other areas of experimental systems research.

Categories and Subject Descriptors D.3.2 [*Language Classifications*]: Very high-level languages; D.4.m [*Operating Systems*]: Miscellaneous; K.7.3 [*The Computing Profession*]: General

Keywords Reproducibility

, .

1. Introduction

A key component of the scientific method is the ability to revisit and reproduce previous experiments. Registering detailed information about an experiment allows scientists to understand and validate results. Reproducibility also plays a major role in education since a student can learn by looking at provenance information, re-evaluate the questions that the original experiment answered and thus "stand on the shoulder of giants".

Given the continuously increasing role that computers play in the discovery of new scientific findings, the issue of reproducibility in applied computer science has recently been the focus of considerable attention by the scientific community [1–3]. Coupled with this, the advent of cloud computing offers new opportunities for experiment validation. Cloud computing makes it easier to share code and data simplifying collaboration for implementing experiments. While it is becoming easier to collaborate, the same cannot be said about experiment validation. There is a serious lack of tools that help researchers identify when an experiment is reproducible. The goal of our work is to close this gap in the area of systems research and storage systems in particular. In systems research, performance *is* the subject of study and we need to look at it as a primary issue. As a consequence, repeating an experiment with the expectation of obtaining exact same results is unrealistic; an issue that further complicates the reproducibility problem.

When discussing reproducibility, the terms *reproducibility*, *repeatability*, *replicability* and *recomputability* (among many others) are often used, sometimes interchangeably. In our work we only employ *repeatability* and *reproducibility*. We borrow the definitions introduced by Vitek et al. [2]:

Repeatability. *The ability to re-run the exact same experiment with the same method on the same or* *similar system and obtain the same or very similar result.*

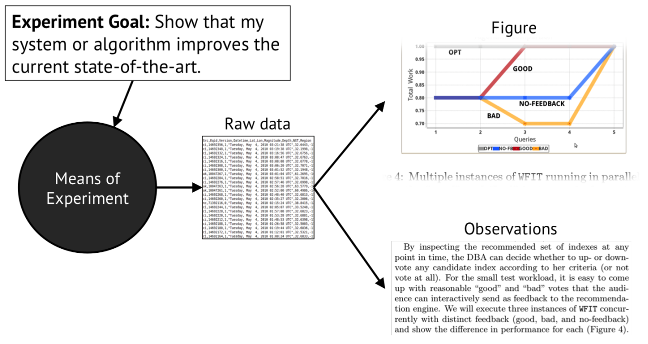
Reproducibility. *The independent confirmation of a scientific hypothesis through reproduction by an independent researcher/lab. The reproductions are carried out after a publication, based on the information in the paper and possibly some other information, such as data sets, published via scientific data repositories or provided by the authors on inquiry.*

While desirable, it is impractical to assume that the exact same experiment can be run on the same or a similar system, thus our main focus is reproducibility. Today's computational environments undergo a continual stream of changes that make it difficult for an experiment to observe the same state across multiple executions. Version-control systems (VCS) are sometimes used to ease the recreation of an experimental environment. Having a specific version ID for the source code and data associated with an article on a public repository significantly reduces the burden of recreating the means of an experiment [4]. However, availability of the source code does not guarantee reproducibility [3] since the code might not compile and, even if compilable, the resulting program might not generate the same results. Recreating an environment that resembles the one where an experiment was originally executed is a challenging task [5]. Virtualization technologies can play a big role in accomplishing this [6,7]. In the end, the re-implementation of an experiment has to be audited by experts to confirm that they resemble the original.

The reproduction of an experiment can be seen as being composed of (1) its execution and (2) the validation of the results. Generally, these two tasks are conflated when designating an experiment as reproducible. In our case, we treat them separately and focus on the latter. At this validation stage, the reviewer has to answer the question: *"are the re-generated results corroborating the original ones?"* An alternative but problematic validation criterion can rely on the exact quantitative observations, that is, results validate the original work if the exact same numerical values of the original output are obtained. This leaves little leeway for validation since more often than not an experiment will get executed on environments that differ from the original. Thus, ideally, we would like to have a way of specifying validation criteria that are as independent as possible from the particular implementation details, i.e., a way of testing the validity of the original work that is agnostic to the implementation of the experiment. A potential solution is to have an experiment specification that describes the expected outcome in abstract rather than absolute terms. We propose to take experiment goals as the basis for validation and treat quantitative observations in the context of these goals.

2. Goals, Means, and Observations

The high-level structure of an experiment can be described as having three components: goals, means, and observations. Two additional transient components, output data and result visualizations, are created as part of running the experiment and are used as a basis for observations (Figure 1).


Experiment Goal: Show that my system or algorithm improves the current state-of-the-art.
Means of Experiment
Raw data
Figure
OPT
GOOD
NO-FEEDBACK
BAD
Total Work
Queries
OPT NO-FB GOOD BAD
4: Multiple instances of WFIT running in parallel
Observations

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).

Figure 1. High-level structure of an experiment.

Goals: An experiment is designed with a particular goal in mind, for example, to show that under certain circumstances, a new system or algorithm improves the state-of-the-art by an order of magnitude.

Means: An experiment is composed of a relatively complex computational environment that includes one or more of the following: hardware, virtualization, OS, configuration, code, data and workload. We refer to these as the means of the experiment and use this term to denote the particularities of how the experimental environment and procedures are carried out.

Observations: As part of the experiment execution, metrics are collected into an output dataset. This raw data can optionally be summarized (e.g., with statistical descriptors) before being displayed in a figure and described in the form of observations made in the prose of the article. The observations made about the output data properties are the basis on which an author proves and corroborates the hypothesis of her work.

A declarative format provides a way to express, at a high-level, the relationship among the goals and means of an experiment, the raw data it produces, and the figures and observations discussed in the article. In other words, it enables the author to provide an experiment design description, its means of execution, and the expected observations that validate the author's claims. Such a description serves two purposes. First, a reviewer or reader with access to this description can, on her own, validate the original work by testing whether the original observations hold on the re-generated output data. Secondly, the explicit specification of these high-level elements aid an author in enhancing the design and presentation of the experimental evaluation of a system by forcing her to think about all aspects of the experiment rather than just generating results for a paper.

3. Experiment Specification Format

An experiment specification format (ESF) allows a scientist to explicitly and declaratively capture an experiment's high-level structure. An example JSON file is shown below. It corresponds to a simplified version of the specification of a published experiment (see Section 4). We describe each section of the ESF next.

```
1  {
2    "goal_location": { "sec": "6.1", "par": 5 },
3    "goal_text": "demonstrate that Ceph scales linearly
4       with the size of the cluster",
5    "goal_category": ["proof_of_concept"],
6    "experiments":[ {
7       "reference":"figure-8",
8       "name":"scalability experiment",
9       "tags":["throughput"],
10      "hardware_dependencies": [{
11         "type": "hdd",
12         "bw": "58MB/s"
13      },{
14         "type": "network",
15         "bw": "1GbE"
16      }],
17      "software_dependencies": [{
18        "type": "os",
19        "kernel": "linux 2.6.32",
20        "distro": "debian 6.0"
21      },{
22        "type": "storage",
23        "name": "ceph",
24        "version": "0.1.67"
25      }],
26      "workload": {
27        "type": "rados-benchmark",
28        "configuration": [
29          "object-size": "4mb", "time": "120s",
30          "threads": "16", "mode": "write"
31      ]},
32      "independent_variables": [{
33        "type": "method",
34        "values": ["raw", "ceph"],
35        "desc": "raw corresponds to hdd sequential write
36           performance, expressed in MB/s"
37      },{
38        "type": "size",
39        "values": ["2-24", 2]
40      }],
41      "dependent_variable": {
42        "type":  "throughput",
43        "scale": "mb/s"
44      },
45      "statistical_functions": {
46        "functions": ["avg", "stddev"],
47        "repetitions": 10
48      },
49      "validations": [
50        "for    size=*
51         expect ceph >= (raw * 0.9)"
52 ]}]}
```

3.1 Experiment Goals

The first elements in the ESF specify the experimental goal (lines 2-8) and link it with one or more experiments that appear in the article that serve to accomplish the goal.

3.2 Means of an Experiment

While computational systems are complex, advances in version-control and cloud computing technologies reduce the burden of recreating the environment on which an experiment runs. Immutability makes it easier to fix a large majority of the components of an experiment as well as infer and package its dependencies [8]. For those components that cannot be fixed to a particular state, tools can automatically obtain and format detailed information about the state of the execution platform, making it easier to compare between original and re-execution environments. The challenge lies in finding, when present, the root cause(s) of the differences in original and reproduced results [9].

The ESF contains a section to specify the means of the experiment. In the example, this corresponds to lines 13-36. This is a simplified list of dependencies for this experiment, used only to illustrate the type of information that is captured in this section. A real example would be more comprehensive, potentially relying on tools that obtain this information automatically.[1]

3.3 Schema of Raw Data

While it is important to capture the output data, making it part of the ESF would be cumbersome and, as has been mentioned, exact numerical repeatability is a very limited validation criterion. Instead, it is preferable to have a description of the metrics being captured, i.e., the metadata of the experiment's output. For example, if the measurements are stored in a CSV file, the experiment specification should include the metadata of each column such as name, aliases, types, scales and ranges.

The ESF has two entries for independent and dependent variables that are used to specify the schema of the output data (lines 37-49). The latter refers to the metric being captured while the former corresponds to the values over which the measurements are taken. Additionally, if statistical functions are applied to the raw data, these should also be specified (lines 50-53), along with the number of experiment repetitions and summarization techniques used, if any.

3.4 Observations and Validation Clauses

We propose using a declarative language for codifying observations. Such a language provides an author with a mechanism to succinctly write descriptive statements that can be used to test for reproducibility. The simplified syntax for the language is the following:

```
validation
 : 'for' condition ('and' condition)*
   'expect' result ('and' result)*
 ;
condition
 : vars ('in' range) | vars ('='|'<'|'>'|'!=') value
 ;
result : condition ;
range
 : 'between' value 'and' value | '['value(','value)*']'
 ;
value
 : '*' | NUMBER | STRING '*' NUMBER
 ;
vars   : STRING (',' STRING)* ;
```

[1] https://github.com/sosreport/sos

The statements constructed via this language refer to elements on the schema of the output data. In other words, the schema specification that precedes the `validations` section of the ESF introduces syntactic elements into the language that provide an easy way to write validation statements. For example, suppose there is an experiment that evaluates concurrency control methods and the experiment measures their performance while varying the number of worker threads. The schema for such an experiment might be the following:

```
{
  "independent_variables": [ {
    "type": "method",
    "values": ["baseline", "mine"]
    }, {
    "type": "threads",
    "values": ["2", "4", "8", "16"]
  }],
  "dependent_variable": {
    "type":  "throughput",
    "scale": "ops/s"
  }
}
```

A statement for this experiment might be:

```
for    threads > 4
expect mine = (10 * baseline)
```

In prose form, the above describes that when the number of worker threads goes beyond 4, `mine` outperforms `baseline` by an order of magnitude. When re-executing this experiment, the data should reflect this behavior in order to validate the results.

4. Case Study

We illustrate our approach by taking a published paper and describing the goals, means, and observations, including the validation clauses, that define the reproducibility criteria for one of the experiments contained in it. We take the Ceph OSDI ’06 paper [10] and reproduce the scalability experiment from the data performance section (6.1 on the original paper). Results of the scalability experiment are presented in Section 6.1.3 of the Ceph paper (reprinted in Figure 2). The goal of this experiment is to *show that Ceph scales linearly with the number of storage nodes, assuming the network switch is never saturated*. This linear scalability is the validation criteria for this experiment and thus what we would like to capture in the specification.

We present the original environment in Table 1 (`Original` column).[2] The original scalability experiment ran with 20 clients per node on 20 nodes (400 clients total) and varied the number of object storage devices (OSDs) from 2-26 in increments of 2. Every node was connected via a 1 GbE link yielding a theoretical upper bound of 2GB/s when there was enough capacity of the OSD cluster to have 20 1Gb connections or alternatively when the connection limit of the switch was reached. The paper experiments were executed on a Netgear switch. This device has a capacity of approximately 14 GbE in *real* total traffic (from a 20 advertised), corresponding to the 24 * 58 = 1400 MB/s combined throughput shown in the original paper (the breaking point in Figure 2).

Table 1: Components of original and recreated environments.

Component	Original	Reproduced
CPU	AMD 2212 @2.0GHz	Intel E5-2630 @2.3GHz
Disk drive	Seagate ST3250620NS	HP 6G 658071-B21
Disk BW	58 MB/s	120 MB/s (15 MB/s throttled)
Linux	2.6.9	3.13.0
Ceph	commit from 2005	0.87.1
Storage	26 nodes	12 nodes
Clients	20 nodes	1 node
Network	Netgear GS748T	Same as original
Network BW	1400 MB/s	110 MB/s

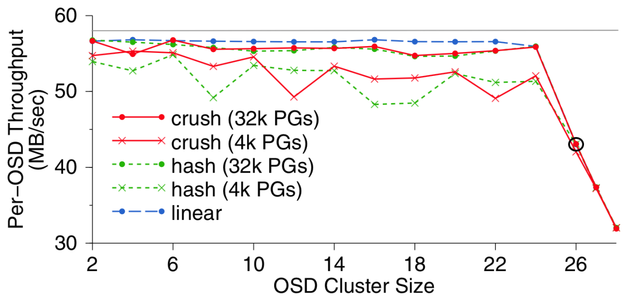


Figure 2. Reprinting Figure 8 from the original paper [10]. The original caption reads: “*object storage device (OSD) write performance scales linearly with the size of the OSD cluster until the switch is saturated at 24 OSDs. CRUSH and hash performance improves when more PGs lower variance in OSD utilization*.” The experiment sequentially writes 4 MB objects to minimize random I/O. Our main focus is on the red solid line with circle markers. The point where linear scalability breaks is encircled in black.

The (simplified) specification shown earlier (Section 3) corresponds to this experiment. Without considering bottlenecks, a reasonable validation statement should specify that the performance of Ceph is within 90% of the raw hard-disk bandwidth, which is what the validation clause in lines 54-57 of the example specifies. In practice, the linear scalability behavior is ultimately limited by the capacity of the underlying network. We would like to express this bottleneck as part of the specification. We can accomplish this by introducing a new clause, for example `for size > 24 expect ceph < (raw * 0.5)`, which specifies that when the size of the cluster exceeds 24, the performance degrades to less than 50% of the raw hard disk bandwidth. However, the network switch capacity is a function of the environment and may ultimately affect the experiment results. An alternative

[2] The complete platform specs, as well as the means (software and workloads) and results of the reproduced experiments are available at https://github.com/ivotron/socc15.

is to extend the grammar to incorporate subclauses that qualify simple validation statements. Using these, the complete clause for this experiment would be:

```
for    size=*
expect ceph >= (raw * 0.9)
when   network not saturated
```

To evaluate the feasibility of this validation, we recreated the original environment using the means specified in the `Recreated` column of Table 1. Due to constraints in hardware resources, we had to scale down the experiment by reducing the number of client nodes to 1 running 16 client threads and 12 storage nodes. This means that our network upper bound is approximately 110 MB/s (the new network bottleneck), corresponding to the capacity of the 1GbE link from the client to the switch. We throttled I/O to 15 MB/s for each storage node.[3] We used this per-OSD increment as our scaling unit. Figure 3 shows results of this scaled-down, throttled re-execution of the scalability experiment.

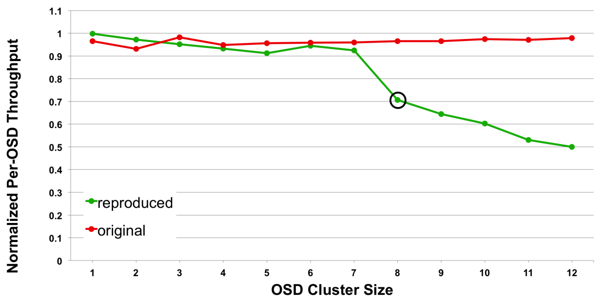


Figure 3. Reproducing a scaled-down version of the original OSDI '06 scalability experiment. The x-axis corresponds to the size of the cluster (in number of OSDs). The y-axis represents normalized throughput (to meaningfully compare against original results) with respect to the raw performance of the hard disk drives in the cluster. The red line corresponds to the original results and the green line to the one generated by the re-execution of the experiment. The point where linear scalability breaks is encircled in black.

Our experiment corroborates that Ceph scales linearly with the number of OSDs until it saturates the available network capacity (1GbE link of the client at 8 OSDs). As can be noted, this is where the declarative specification stands out since the validation is independent of the particularities of the means of each experiment. Even though the recreated environment is significantly different from the original, we are able to reproduce the results by validating on the basis of the experiment goal, schema of the output and validation clauses expressed as relative rather than absolute throughput measurements.

[3] We throttle I/O with the purpose of slowing down the experiment. The hard drives used for the reproduced experiment can perform at 120 MB/s which would saturate the network link rapidly.

5. Discussion

5.1 Usability

Given that the high-level components (Section 2) abstract a large number of experiments that people usually implement in the storage systems literature and since this is what a researcher usually goes through anyway, creating the specification for an experiment represents little extra effort. The exception being documenting the experiment means which, as we mentioned before (Section 3.2), is a task that can be automated using currently available tools.

5.2 Integration into Existing Infrastructure

Experimental platforms such as CloudLab [11] can incorporate the notion of *execution* so that for every experiment a record of executions is maintained. For each execution, the means section of the ESF can be automatically populated. Validation statements can also provide another testability layer for continuous integration (CI) systems such as Jenkins.

5.3 Codified Observations As Falsifiable Statements

Validation clauses serve to succinctly codify observations. Given the descriptive language design, validation ranges have to be provided for each observation so that it can be tested. This has the implication of turning observations into falsifiable statements [12]. These validation clauses are conditions that should hold in order to corroborate the conclusions of the paper. In other words, if the means of the experiment are properly recreated, the specified behavior should be observed.

Experiment goals (Section 3.1) set the tone in which these falsifiable statements are treated. For an experiment that proves a concept or design, a validation clause has more weight than, say, an experiment that quantifies an expected overhead. For example, for a system that claims to achieve linear scalability, the corresponding validation clauses are more significant than those for an experiment that shows the overhead of a new file system implemented as a FUSE module. In the former, a failed validation invalidates the whole work while in the latter the failed test invalidates a minor aspect. In other words, some experiments evaluate a high-level claim while others refer to low-level aspects, hence the importance of looking at experiment goals while looking at validations; goals set the mindset of the reader or reviewer that validates the work whenever she encounters failed validations. This is the main motivation for having goals as an explicit entry on the ESF.

5.4 The Validation Workflow

The ESF has the structure of a conditional statement: given the goals and means of an experiment, the observations on the output data should hold. Thus, if the validation statements are false with respect to the output data of the re-execution of an experiment, it is either because the differences between the means of the original and reproduced experiment are significantly different, or the original claims cannot be corroborated. Thus, before one can determine the latter, one has to audit the differences between the means of experimentation and account for all of them (Figure 4).

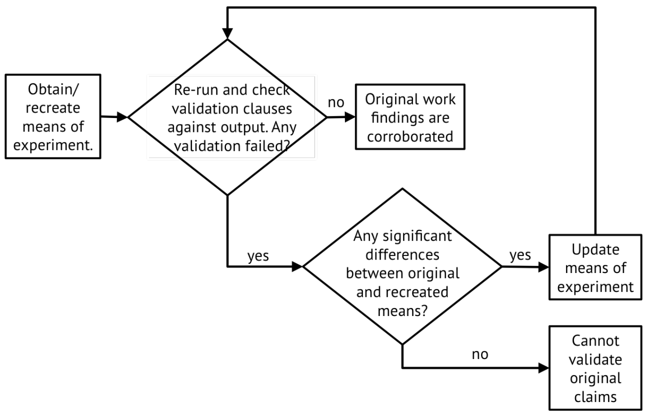

Obtain/ recreate means of experiment.
Re-run and check validation clauses against output. Any validation failed?
no
Original work findings are corroborated
yes
Any significant differences between original and recreated means?
yes
Update means of experiment
no
Cannot validate original claims

Figure 4. Validation workflow.

5.5 Early Feedback

The following are quotes from authors that have kindly worked with us by creating specifications for one or more of their published experiments:

> Author 1: *Writing an experiment specification makes you think clearly about the overall experiment design*.
>
> Author 2: *The ESF provides a nice template for carrying out experiments*.
>
> Author 3: *This approach helps to find meaningful baselines. Reporting raw numbers in figures and observations makes it harder to validate results. Specifying validation clauses respective to baselines and normalized values makes it easier to report reproducible results*.

In general, we have noticed that the exercise of explicitly specifying the validation criteria creates a feedback loop in an author's mind that results in insightful ideas for experiment design, baseline selection, and validation criteria. Additionally, the author's point of view is explicitly expressed. Usually, figures contain more information than necessary to back a claim. This might lead readers to draw other conclusions. Lastly, every article has an implicit temporal context associated to it that the reader has to keep in mind, for example, the bottleneck at the time that an article was published might be in storage (e.g., hard disk drives) while at other times they might have moved to the network instead (e.g., because of the availability of faster storage such as SSDs). A possibility would be to create a community-maintained knowledge base that an author can link the paper to so that a semantic context is available to the reader.

6. Related Work

The challenging task of evaluating experimental results in applied computer science has been long recognized [13–15]. This and other related issues have gained substantial attention lately in systems research [2,3,16–21], computational science [1,18,19,22] and science in general [23–25]. Similarly, efforts such as *The Recomputation Manifesto* [26] and the *Software Sustainability Institute* [27] have reproducibility as a central part of their endeavour but leave performance as a secondary problem. In systems research, performance *is* the subject of study, thus we need to look at it as a primary issue.

In [3] the authors took 601 articles published in 13 top-tier systems research conferences and found that 32.3% of associated experiments are repeatable (under their definition of *weak repeatability*). The authors did not validate the original results. In our case, we are interested not only in being able to rebuild and execute binaries (repeat/reproduce the execution) but also in validating the original claims by testing falsifiable statements on the output of the experiment.

Krishnamurthi and Vitek [5] emphasize the importance of repeatability and describe recent efforts by the systems research community to encourage the submission of experiment artifacts as assets associated to an article. We see our work as complementary to these since an experiment specification could also be part of this list of assets, making it easier to validate a re-generated result.

The use of declarative specifications has been explored in the context of cloud recovery testing [28], bug reproduction [29] and cloud resource orchestration [30].

7. Conclusion and Future Work

In the words of Karl Popper: "*the criterion of the scientific status of a theory is its falsifiability, or refutability, or testability*". By providing a way to specify the high-level components of an experiment and validation clauses for observed metrics we effectively incorporate falsifiability to the field of experimental storage systems. We are in the process of studying the viability of the ESF on experiments from other areas of systems research. As part of our work, we are working with colleagues in our field to create descriptions for already-published experiments and analyze them to check if they capture the appropriate validation criteria. While we envision our findings to be applicable in the area of systems research, we plan to evaluate its suitability on other areas of computer science.

Acknowledgements: This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. LLNL-TR-670295 . Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

8. References

[1] R.D. Peng, "Reproducible research in computational science," *Science*, vol. 334, Dec. 2011, pp. 1226–1227.

[2] J. Vitek and T. Kalibera, "Repeatability, reproducibility, and rigor in systems research," *Proceedings of the ninth ACM international conference on embedded software*, New York, NY, USA: ACM, 2011, pp. 33–38.

[3] C. Collberg, T. Proebsting, and A.M. Warren, "Repeatability and benefaction in computer systems research," 2015.

[4] C.T. Brown, "How we make our papers replicable," 2014. Available at: http://ivory.idyll.org/blog/2014-our-paper-process.html.

[5] S. Krishnamurthi and J. Vitek, "The real software crisis: Repeatability as a core value," *Commun. ACM*, vol. 58, Feb. 2015, pp. 34–36.

[6] B. Clark, T. Deshane, E. Dow, S. Evanchik, M. Finlayson, J. Herne, and J.N. Matthews, "Xen and the art of repeated research," *Proceedings of the annual conference on USENIX annual technical conference*, Berkeley, CA, USA: USENIX Association, 2004, pp. 47–47.

[7] I. Jimenez, C. Maltzahn, J. Lofstead, A. Moody, K. Mohror, R.H. Arpaci-Dusseau, and A. Arpaci-Dusseau, "The role of container technology in reproducible computer systems research," *2015 IEEE international conference on cloud engineering (IC2E)*, Tempe, AZ: 2015.

[8] F. Chirigati, D. Shasha, and J. Freire, "ReproZip: Using provenance to support computational reproducibility," *Proceedings of the 5th USENIX conference on theory and practice of provenance*, Berkeley, CA, USA: USENIX Association, 2013, pp. 1–1.

[9] I. Jimenez, C. Maltzahn, A. Moody, and K. Mohror, *Redo: Reproducibility at scale*, UC Santa Cruz, 2014.

[10] S.A. Weil, S.A. Brandt, E.L. Miller, D.D.E. Long, and C. Maltzahn, "Ceph: A scalable, high-performance distributed file system," *Proceedings of the 7th symposium on operating systems design and implementation*, Berkeley, CA, USA: USENIX Association, 2006, pp. 307–320.

[11] R. Ricci and E. Eide, "Introducing CloudLab: Scientific infrastructure for advancing cloud architectures and applications,"*;login:* vol. 39, Dec. 2014, pp. 36–38.

[12] K. Popper, *The logic of scientific discovery*, New Delhi: Routledge, 2002.

[13] J.P. Ignizio, "On the establishment of standards for comparing algorithm performance," *Interfaces*, vol. 2, Nov. 1971, pp. 8–11.

[14] J.P. Ignizio, "Validating claims for algorithms proposed for publication," *Operations Research*, vol. 21, May. 1973, pp. 852–854.

[15] H. Crowder, R.S. Dembo, and J.M. Mulvey, "On reporting computational experiments with mathematical software," *ACM Trans. Math. Softw.*, vol. 5, Jun. 1979, pp. 193–203.

[16] C. Dietrich and D. Lohmann, "The dataref versuchung: Saving time through better internal repeatability," *SIGOPS Oper. Syst. Rev.*, vol. 49, Jan. 2015, pp. 51–60.

[17] D.G. Feitelson, "From repeatability to reproducibility and corroboration," *SIGOPS Oper. Syst. Rev.*, vol. 49, Jan. 2015, pp. 3–11.

[18] J. Freire, P. Bonnet, and D. Shasha, "Computational reproducibility: State-of-the-art, challenges, and database research opportunities," *Proceedings of the 2012 ACM SIGMOD international conference on management of data*, New York, NY, USA: ACM, 2012, pp. 593–596.

[19] C. Neylon, J. Aerts, C.T. Brown, S.J. Coles, L. Hatton, D. Lemire, K.J. Millman, P. Murray-Rust, F. Perez, N. Saunders, N. Shah, A. Smith, G. Varoquaux, and E. Willighagen, "Changing computational research: The challenges ahead," *Source Code for Biology and Medicine*, vol. 7, Dec. 2012, pp. 1–2.

[20] R. LeVeqije, I. Mitchell, and V. Stodden, "Reproducible research for scientific computing: Tools and strategies for changing the culture," *Computing in Science Engineering*, vol. 14, Jul. 2012, pp. 13–17.

[21] V. Stodden, F. Leisch, and R.D. Peng, *Implementing reproducible research*, CRC Press, 2014.

[22] D.L. Donoho, A. Maleki, I.U. Rahman, M. Shahram, and V. Stodden, "Reproducible research in computational harmonic analysis," *Computing in Science & Engineering*, vol. 11, Jan. 2009, pp. 8–18.

[23] J. Achenbach, "The new scientific revolution: Reproducibility at last," *The Washington Post*, Jan. 2015.

[24] M.B. Yaffe, "Reproducibility in science," *Science Signaling*, vol. 8, Apr. 2015, pp. eg5–eg5.

[25] Editorial, "Journals unite for reproducibility," *Nature*, vol. 515, Nov. 2014, pp. 7–7.

[26] I.P. Gent, "The recomputation manifesto," *arXiv:1304.3674 [cs]*, Apr. 2013.

[27] S. Crouch, N. Hong, S. Hettrick, M. Jackson, A. Pawlik, S. Sufi, L. Carr, D. De Roure, C. Goble, and M. Parsons, "The software sustainability institute: Changing research software attitudes and practices," *Computing in Science Engineering*, vol. 15, Nov. 2013, pp. 74–80.

[28] H.S. Gunawi, T. Do, P. Joshi, P. Alvaro, J.M. Hellerstein, A.C. Arpaci-Dusseau, R.H. Arpaci-Dusseau, K. Sen, and D. Borthakur, "FATE and DESTINI: A framework for cloud recovery testing," *Proceedings of the 8th USENIX conference on networked systems design and implementation*, Berkeley, CA, USA: USENIX Association, 2011, pp. 238–252.

[29] K. Li, P. Joshi, A. Gupta, and M.K. Ganai, "ReproLite: A lightweight tool to quickly reproduce hard system bugs," *Proceedings of the ACM symposium on cloud computing*, New York, NY, USA: ACM, 2014, pp. 25:1–25:13.

[30] C. Liu, B.T. Loo, and Y. Mao, "Declarative automated cloud resource orchestration," *Proceedings of the 2Nd ACM symposium on cloud computing*, New York, NY, USA: ACM, 2011, pp. 26:1–26:8.