Tackling the Reproducibility Problem in Storage Systems Research with Declarative Experiment Specifications

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

Jay Lofstead
Sandia National Laboratories
gflofst@sandia.gov

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

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

ABSTRACT

Validating experimental results in the field of storage 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 position paper, 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 distributed storage systems to illustrate the usefulness of this approach.

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 scientific discoveries, the issue of reproducibility in applied computer science has recently been the focus of considerable attention by the scientific community [1–3]. The advent of 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. The goal of our work is to close this gap in the area of systems research and storage systems in particular. To complicate the things further, in systems research, performance is the subject of study and we need to look at it as a primary issue.

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 [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 it resembles 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 designing 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. In this position paper, 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).

![Figure 1: High-level structure of an experiment.](image)

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 rationale behind the experiment design, its means of execution, and the expected observations that validate the author’s claims.

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.

```json
{
    "goal_location": { "sec": "6.1", "par": 5 },
    "goal_text": "demonstrate that Ceph scales linearly with the size of the cluster",
    "goal_category": ["proof_of_concept"],
    "experiments": [{
        "reference": "figure-8",
        "name": "scalability experiment",
        "tags": ["throughput"],
        "hardware_dependencies": {
            "type": "hdd",
            "bv": "50MB/s"
        },
        "software_dependencies": {
            "type": "storage",
            "name": "ceph",
            "version": "0.1.67"
        },
        "workload": {
            "type": "rados-benchmark",
            "configuration": {
                "object-size": "4mb",
                "time": "120s",
                "threads": "16",
                "mode": "write"
            },
            "independent_variables": {
                "type": "method",
                "values": ["raw", "ceph"],
                "desc": "raw corresponds to hdd sequential write performance, expressed in MB/s"
            },
            "dependent_variable": {
                "type": "throughput",
                "scale": "mb/s"
            },
            "statistical_functions": {
                "functions": ["avg", "stdev"],
                "repetitions": 10
            }
        }
    }]
}
```

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)*
 ;

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:

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

A statement for this experiment might be:

```python
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.

### Table 1: Components of original and reproduced environments of the scalability experiment.

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

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

²The complete platform specs, as well as the means (software and workloads) and results of the reproduced experiments are available at https://github.com/ivotron/pds15.
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), corre-

csponding to the 24 * 58 = 1400 MB/s combined throughput

shown in the original paper (the breaking point in Figure 2).

The (simplified) specification shown earlier (Section 3)
corresponds to this experiment. Without considering bottle-

necks, 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 be-

havior 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 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 not network_saturated

The boolean value for network_saturated should come from

network metrics that are captured at runtime. For example, some switches implement the SNMP protocol

that allows to identify if the network is getting saturated.

To evaluate the feasibility of this particular validation, we

recreated the original environment using the means specified

in the Reproduced 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. 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.

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.

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 specifi-
cation 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

The ESF can in principle be produced or consumed by

existing tools in order to automate the validation process.

3We throttle I/O with the purpose of slowing down the exper-

iment. The hard drives used for the reproduced experiment

can perform at 120 MB/s which would saturate the network

link rapidly.
For example, 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, where not only the correctness of a system is checked (via unit/integration tests) but also the performance is tested against changes made to the codebase.

5.3 Codified Observations Express 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. 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. 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).

![Figure 4: Validation workflow.](image)

5.5 Early Feedback

The following are quotes from authors who 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 [13,14]. This and other related issues have gained substantial attention lately in systems research [2,3,15–20], computational science [1,17,18,21] and science in general [22–24]. Similarly, efforts such as The Recomputation Manifesto [25] and the Software Sustainability Institute [26] 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. The use of declarative specifications has been explored in the context of cloud recovery testing [27], bug reproduction [28] and cloud resource orchestration [29].

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. 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:** Work performed under auspices of U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 LNL CONF-669866. Sandia National Laboratories is managed and operated by Sandia Corporation, for the U.S. Department of Energy’s National Nuclear Security Administration under contract DE-AC04-94AL85000.
8. REFERENCES


