Technical Report: SPEC-RG-2019-01
Version: 1.0

Methodological Principles for Reproducible Performance Evaluation in Cloud Computing

SPEC RG Cloud Working Group

Alessandro V. Papadopoulos
Mälardalen University
Västerås
Sweden
alessandro.papadopoulos@mdh.se

Laurens Versluis
Vrije Universiteit
Amsterdam
The Netherlands

André Bauer
Julius-Maximilian University
Würzburg
Germany

Nikolas Herbst
Julius-Maximilian University
Würzburg
Germany

Jóakim von Kistowski
Julius-Maximilian University
Würzburg
Germany

Ahmed Ali-Eldin
Umeå University, Sweden
and UMass Amherst
USA

Cristina L. Abad
Escuela Superior Politecnica del Litoral
Ecuador

José Nelson Amaral
University of Alberta
Canada

Petr Tůma
Charles University
Czech Republic

Alexandru Iosup
Vrije Universiteit
Amsterdam
The Netherlands

A revised version of this technical report has been submitted for peer-review to IEEE Transactions on Software Engineering (ToSE).

Contents

Executive Summary

The rapid adoption and the diversification of cloud computing technology exacerbate the importance of a sound experimental methodology for this domain.
This work investigates how to measure and report performance in the cloud, and how well the cloud research community is already doing it.
We propose a set of eight important methodological principles that combine best-practices from nearby fields with concepts applicable only to clouds, and with new ideas about the time-accuracy trade-off.
We show how these principles are applicable using a practical use-case experiment. To this end, we analyze the ability of the newly released SPEC Cloud IaaS benchmark to follow the principles, and showcase real-world experimental studies in common cloud environments that meet the principles.
Last, we report on a systematic literature review including top conferences and journals in the field, from 2012 to 2017, analyzing if the practice of reporting cloud performance measurements follows the proposed eight principles. Worryingly, this systematic survey and the subsequent two-round human reviews, reveal that few of the published studies follow the eight experimental principles.
We conclude that, although these important principles are simple and basic, the cloud community is yet to adopt them broadly to deliver sound measurement of cloud environments.

Keywords

Experimental evaluation, observation study, experimentation

1 Introduction

Experimental methodology problems are common in many domains (SNTH13). In computer science research they seem to appear also due to a lack of agreement on standard techniques for measuring, reporting, and interpreting performance (HB15; MDHS09). A domain that raises new and important performance evaluation challenges is cloud computing. The first commercial cloud has opened for the general public in 2007 (Amazon AWS), and today cloud computing is an established field that is growing rapidly. Cloud computing requires advances in performance engineering, computer systems, and software engineering, which require meaningful experimentation to test, evaluate, and compare systems. In this relatively new field, experiments focusing on cloud computing performance raise new methodological challenges (IPE14) related to technological aspects such as dynamic environments, on-demand resources and services, diverse cost models, and new non-functional requirements such as elasticity (HKO$^+$16) or elasticity-correlated metrics (LPM$^+$17). In this work, we focus on the principles and feasibility of experimental performance studies in cloud computing settings.

Sound experimental methodology, and in particular reliable, consistent, and meaningful performance evaluation, is challenging but necessary. Poor experimental design and/or execution are often the cause for many pitfalls encountered by well-meaning researchers and practitioners (HB15; SNTH13). As we show here, examples of problems that occur repeatedly, even for research published in top venues, include: flawed or no definition of meaningful metrics, inadequate number of experiment repetitions, unreproducible cloud experiments, simplistic summarization of data from multiple measurements across single or multiple clouds, and inconsistencies between the experimental results and published claims. Consequently, it is difficult to reproduce experiments, to make fair comparisons between different techniques of the same class, or to benchmark fairly across competing products. This situation is particularly undesirable for a service-based industry such as cloud computing, which operates on the promise (or guarantee) of performance delivery.

Although many top-level conferences accept cloud computing articles that include performance results obtained experimentally, this work shows that the domain's adherence to sound methodological principles is lacking. Motivated also by increasing awareness about such principles in other domains, e.g., software engineering over a decade ago (SHH$^+$05) and high-performance computing in the last few years (HB15), we propose to revisit the basic principles that underpin the performance evaluation of cloud computing artifacts. Toward this end, this work makes three contributions, with each contribution structured around a main question of experimental methodology:

RQ1 What methodological principles are needed for sound experimental evaluation of cloud performance? (addressed in Section 2)

Reproducibility of experiments is a key part of the scientific method (Fei05). We focus on technical reproducibility (defined in Section 2.2), and propose eight methodological principles that target diverse aspects of experimental evaluation methodology and execution. These include designing experiments, obtaining and reporting results, and sharing software artifacts and datasets. Cloud-specific aspects include reporting cost, based on two classes of pricing models.

RQ2 Can the methodological principles be applied in common practice? (addressed in Section 3)

Pragmatism is important for the adoption of methodological principles for experimentation. If the principles are meaningful, but cannot be easily applied in practice or cost too much to perform, the community will delay or even refuse to use them. This work provides evidence that the proposed principles can be used in two common situations: (i) the commercial benchmark SPEC Cloud IaaS, and (ii) a set of experiments with common cloud and self-owned infrastructure conducted by a research team.

RQ3 How are cloud performance results currently obtained and reported?[1] (addressed in Section 4)

To understand the state of practice, we conduct a systematic review (BB06; PVK15) of a representative sample of papers on cloud computing that have been published, between 2012 and 2017, **in 16 leading venues**. A highlight of this review is the careful, multi-reviewer examination of these papers for the most important factors in experimental design and execution that may limit technical reproducibility. The review also focuses on comparative performance evaluation with competing approaches to the one presented in the paper.

Last, Section 5 compares this work with the body of work on principles and practice of experimental evaluation, across four related research communities: cloud computing itself, performance engineering, computer systems in general, and software engineering.

[1] In various related communities: cloud systems, control systems, performance engineering, general computer systems.

2 Experiment Methodology

This section addresses the question of how to design and report cloud performance experiments (**RQ1**).

In other areas of experimental computing, the scope of the system under evaluation seems relatively narrow. For example, both in the evaluation of infrastructure for High-Performance Computing (HPC) (SBZ+08; FMMS14), or in the evaluation of Java Virtual Machines (JVMs) and Just-in-Time compilers (GBE07; HLST15), the experimental methodology may consist of a relatively compact, prescriptive list of artifacts and factors. In contrast, the open and interconnected nature of cloud computing introduces too many relevant factors to be covered exhaustively, of which the HPC systems and JVMs may be merely some of the evaluated aspects.

The experimental methodology we propose focuses on emphasizing the selection of suitable metrics and the reproducibility of experimental results. This work presents eight principles that are particularly relevant for reproducibility of performance experiments on cloud computing platforms.

2.1 Metric Selection

A *measurement* is the assignment of values to objects or events, according to a defined procedure. Based on collected raw measurement data, measures can be computed, each with the purpose of capturing certain aspects of the experiment outcome. In mathematics, the term *metric* is explicitly distinguished from the term measure (the former referring to a *distance function*). However, in computer science and engineering the terms metric and measure overlap in meaning, and are often used interchangeably. One way to distinguish between metric and measure is to regard a metric as a value that can be derived from some measures obtained from experimental evaluation. Metrics and measures may be defined for different *scales*. There are absolute scales with a natural unit, ratio scales with a given zero point, and interval scales with a given distance function.

When several metrics have similar value range and distribution in practical settings, the definition of aggregated metrics may improve the ability to establish valid claims about the experimental results. Examples of established aggregation approaches include (i) the arithmetic mean of raw measurements (but never of rates or percentages (FW86)) (ii) a (weighted) geometric mean for speedup ratios compared to a reference implementation, (iii) a pairwise comparison for closed sets of alternatives, and (iv) an L_p-norm as distance to the theoretical optimum. More details are provided in previous works covering metric aggregation and metric quality attributes (HBK+18).

2.2 Reproducibility

Although reproducibility of experiments is one of the pillars of the scientific method (Fei05), it is rare in practice. In a 2016, *Nature*-published survey, Baker finds that 70% of the 1,500 researchers surveyed have tried and failed to reproduce prior work done by others, and over 50% failed to reproduce *their own* experimental results (Bak16). Even in computer science, where the use of open-source code, versioning, and virtualization are among the obvious techniques that enhance reproducibility, Collberg and Proebsting showed in 2016 that more than 50% of the work published in top venues, including ASPLOS, SOSP, and VLDB, cannot be reproduced due to missing or un-compilable code. They also showed that authors are sometimes unable to reproduce *their own* published results (CP16).

Technical solutions to improve reproducibility have been proposed. One of the earliest, PlanetLab, provided functional reproducibility of experiments, but could not ensure reproducible

non-functional properties, such as measured latency and bandwidth, due to uncontrolled resource sharing (SPBP06). Handigol et al. (HHJ+12) promote the use of containers for repeatable datacenter networking experiments; systems such as APT (RWS+15), EmuLab (ESL07), and FlexLab (RDS+07) aim to provide environments for repeatable distributed networked systems research; and frameworks such as DataMill (POZ+16) offer some control over experimental variability. Among the more extreme examples is the work by Governor et al. (GSG+15) that provides the means for the reader to reproduce everything in a paper, including the graphs, by executing very simple steps. Another example is the work by Cavazos et al. (CFA+07), that yielded an open toolkit[2] for portable experiment definition, implementation, execution and evaluation. Industry solutions, such as the Jupyter Notebook,[3] are also gaining popularity because they ease prototyping, sharing, and full reproduction of data processing projects.

Alongside the technical solutions to specific reproducibility issues, research communities now try to improve the review and publication process. SIGMOD, a leading conference on data management and databases, has been an early leader in championing experiment repeatability—accepted papers since 2008 are invited to submit code and data required for third-party experiment reproduction[4]. Other leading computer-science venues follow suit (Boi16). In performance evaluation, the ICPE conference is encouraging authors to share research artifacts (both code and data) in a public repository maintained by the SPEC Research Group[5]. SIGCOMM is running a workshop dedicated to reproducibility of networking research[6]. SIGPLAN is currently requesting feedback about a best-practices Empirical Evaluation Checklist to "help [programming languages] authors produce stronger scholarship, and to help reviewers evaluate such scholarship more consistently"[7]. As a publisher, ACM has an artifact evaluation (AE) process[8], used now by conferences such as SC and PPoPP, which aims to reduce the expenses of experiment reproduction.

There are many facets of reproducibility that need to be considered. In cloud benchmarking, the experimental environment often includes opaque elements of the cloud infrastructure, external workload over shared resources, and other complicating factors. An exact reproduction of measurement results is rarely possible, and typically unnecessary (Fei05). Instead, the focus should be on *technical reproducibility*, which requires only a description of the technical artifacts needed for the experiment and a clear description of what was measured and how it was measured—i.e., the information needed to repeat the same experiment.

Another important concept is the *reproducibility of claims*, which asserts that a reproduction of an experiment should support the same claims that were derived from the original study, despite possible variations in the measurement results. Claim reproducibility requires not only accounting for measurement variability, but also awareness and control for external factors. While tools often address technical reproducibility factors, the reproducibility of claims depends on the conditions under which the experiment is executed, on the computation, and on the interpretation of the experiment results. Claim reproducibility focuses on the generality of the findings, rather than on the specific technical aspects.

[2] `http://www.cknowledge.org`
[3] `http://jupyter-notebook-beginner-guide.readthedocs.io`
[4] `http://db-reproducibility.seas.harvard.edu/`
[5] `https://icpe.spec.org/artifact-repository.html`
[6] `http://conferences.sigcomm.org/sigcomm/2017/workshop-reproducibility.html`
[7] `http://sigplan.org/Resources/EmpiricalEvaluation/`
[8] `http://www.acm.org/publications/policies/artifact-review-badging`

2.3 Methodological Principles

How to design experiments? One of the core parts of a performance evaluation process is the *experiment design*, i.e., the organization of the experiment to ensure that the right type of data, and enough of it, is available to answer the questions of interest as clearly as possible (EM97).

P1: **Repeated experiments**. After identifying the sources of variability, decide how many repetitions **with the same configuration** of the experiment should be run, and then quantify the confidence in the final result.

It is essential to identify sources of variability in measured performance, because one of the main aims of any designed experiment is to reduce their effect on the answers to questions of interest. Cloud computing platforms exhibit significant performance variability (SDQR10; IYE11; LC16; AB17). This is due to many factors, from multi-tenant workloads sharing the cloud resources to minute differences in hardware of individual machine instances. Often, these factors cannot be explicitly controlled. Instead, performing a sufficient number of equally configured randomized experiments is needed to make sure that the results are not due to chance, and to provide a statistically sound assertion about the confidence with which the data supports the claims.

A decision on the number of repeated experiments (or the duration of an experiment when observations are collected continuously) may need to account for factors such as random or seasonal performance variations, the required accuracy, and the acceptable probability of an erroneous conclusion. The number of repetitions can typically be reduced when a model that explains or predicts the experimental results is available, e.g., a change in performance can be judged significant depending on whether it reasonably correlates with changes in other system metrics.

Violation Examples. In our survey (see Section 4), Principle 1 is often partially fulfilled by performing certain number of repetitions or by selecting a long duration for the continuous experimental run. However, often this choice is not justified and sometimes not even reported, and is not explicitly connected to the required accuracy and confidence—evaluating peers appear to have been content with merely seeing that some repetitions were done, even when the number of repetitions is specified *ad hoc*. This is not sufficient, because it may not be possible to determine whether sufficient repetitions were done, with only aggregate results.

P2: **Independent experiments**. Experiments should be conducted in **different (possibly randomized) configurations** of relevant parameters, especially parameters that are not completely under control or those that may interact with the platform in unexpected ways, e.g., the workload. Parameter values should be randomized according to realistic probabilistic distributions or using historical data. The confidence in the final result should be quantified.

In cloud computing, experimental setup often includes configurable parameters. However, some parameters that have significant effect on performance may not be under the (complete) control of the experimenter. For example, although the experimenter may select the number of allocated machine instances, the aggregate computing power across these instances may fluctuate, and with it the observed performance. Similarly, some parameters may interact with the platform unexpectedly. For example, when the allocation of resources for the experiment happens to perfectly utilize the underlying host or rack, the observed performance might be better than in a slightly larger setup that would require communication between multiple hosts or racks. Carrying out independent experiments where the relevant parameters are randomized can provide more robust results.

To avoid affecting the experiment, the choice of parameter values should be based on realistic probabilistic distributions. Usually, the performance evaluation should examine common system operation and should not be affected by rare events. If, on the other hand, the evaluation

targets robustness with respect to rare events, the number of independent experiments and their parameters should be adjusted accordingly. In both cases, decisions should be reflected in the quantification of the result confidence.

Violation Examples. The surveyed papers often partially address Principle 2 by using multiple workloads, for example multiple benchmarks or multiple replay traces, which stress the system in different ways. Unfortunately, the choice of workloads appears to be motivated partially by ease of use. Despite existence of relevant techniques (HE07; JPEJ06), workload coverage is not considered. Randomization is rare, in part perhaps because systematic treatment of workload randomization is relatively recent (Fei16).

How to report experimental data? Reports should include all information needed to evaluate the quality of the used data, to assess the soundness and correctness of the evaluation approach, and to reproduce the results.

P3: **Experimental setup description**. Description of the hardware and software setup used to carry out the experiments, and of other relevant environmental parameters, must be provided. This description should include the operating system and software versions, and all the information related to the configuration of each experiment.

This principle requires that the conditions under which the experiments were carried out are described in sufficient detail to enable technical reproducibility (see Section 2.2). Some details that are traditionally omitted may have a significant effect on the experimental results and thus in the claims derived from the results. Descriptions should always include: (i) the system under test (SUT), (ii) the environment under which this system is tested, including its non-default configuration parameters, (iii) the workload, possibly as a reference to a detailed characterization or standardization, (iv) the monitoring system and how its data is converted into metrics, either formulas or plain text, but with enough information if the metrics are not obtained from the monitoring data straightforwardly.

In a cloud environment, the exact parameters of the system under test, e.g., both the host platform of the virtual machine (VM) and the guest-VM instances, may change. This type of information must therefore be documented in the experimental setup. The configuration information may also be important to determine the number of experiments that is needed to make sure that the results obtained are not dependent on the VM instance type.

Violation Examples. Our survey identified some papers that omit the experimental setup information. However, a more salient point is that even for papers that provide such a description, there is much variation in how the setup is described. Some papers merely list the (marketing) names of the VM instances, whereas others provide memory sizes and core counts, and others further provide details such as kernel versions of the guest-VM. There is a marked lack of certainty about what information to record and report, and there is little discussion of whether and how unknown experimental setup parameters may affect the results.

As a practical constraint, the page limits applied to research papers may prevent inclusion of the experimental setup description in appropriate detail, going beyond the ACM AE guidelines. We believe a full description can only be presented in a separate document, e.g., a web page, an appendix or a technical report in an archival form, and suggest that publishers provide supplementary archival resources for the storage of such descriptions.

P4: **Open access artifact**. At least a representative subset of the developed software and data (e.g., workload traces, configuration files, experimental protocol, evaluation scripts) used for the experiment should be made available to the scientific community. The metadata of the released artifact should uniquely identify the artifact, including timestamping and version in a versioning system.

This principle is related to Principle 3 and, more generally, to the technical reproducibility of results. A typical cloud experiment setup is quite complex, possibly with large third-party

components (guest operating system images, middleware, benchmark applications and workload generators). It may not be practical to record the relevant details of such a setup other than by preserving the entire experiment artifact. The experimenter may also have little control over what happens in external data centers, e.g., for experiments run on Amazon S3 the actual machine configurations may change from one run to the next. Preserving the experiment artifact may be the only way to make the experiment pertinent in quickly developing environments.

Artifacts also have educational value. When made available, students can build on top of prior results, acquire training by setting up experiments, reproducing them, and comparing the results with those in published papers. These exercises accelerate the acquisition of skills and expertise.

Finally, we should accept that experiment results may be distorted due to bugs that can only be found through an external scrutiny of the artifact.[9]

Violation Examples. Our survey points to a dearth of published artifacts. Keeping the artifacts private appears to be the default choice, with no justification offered, when the opposite should be true—unless public artifacts are the norm, authors will find it difficult to justify the extra effort needed to prepare artifacts for publication. On the technical side, we have encountered examples of artifact web pages becoming inaccessible after publication, suggesting that more robust archival options should be used.

Publishing reduced artifacts may be needed to meet intellectual property and privacy requirements. When publishing partial data, sampling methods should preserve Principles 1 and 2, and the general result trends.

P5: **Probabilistic result description of measured performance**. Report a full characterization of the empirical distribution of the measured performance, including aggregated values and variations around the aggregation, with the confidence that the results lend to these values.

Reporting aggregated values, their variations, and the confidence in these values is useful to understand the statistical features of the measured performance. However, aggregation does not magically guarantee a faithful characterization of the measured performance. Averaged values only make practical sense if the measurements have a distribution clustered around their central tendency, such as the Gaussian distribution. Variation as a measure of dispersion can mask the difference between a few big outliers and constantly fluctuating measurements. Experiments with complex distributions of results should employ tools such as box plots, violin plots, and empirical distribution function plots. Fundamentally, the experimenter must carefully examine the raw data before computing aggregated values.

Violation Examples. One striking observation in our study was the focus on analyzing average performance, without looking at tail performance metrics such as latency quantiles, which are of obvious practical importance. Sometimes, average values were accompanied in the plots by error bars whose meaning was not defined, creating an impression of reporting on measurement variation when in fact little can be derived from such data. Sometimes, inappropriate mean computations are also used (FW86; Mas04).

P6: **Statistical evaluation**. When comparing different approaches, provide a statistical evaluation of the significance of the obtained results.

The results of an experimental evaluation are often used to compare an artifact against competing approaches to solve the same problem. In such cases, a claim that one method is better than another is made based on an evaluation that considered a (necessarily) limited number of scenarios. The likelihood and representativeness of these scenarios becomes a factor in

[9]Several other sciences have taken steps toward this, including medical sciences (Soe15) and economics (HAP14).

the (statistical) significance of the experimental results, and is therefore essential in establishing the validity of the claim.

Sometimes, the competing approach may have been evaluated by someone else, and there may not be enough data available for a complete statistical analysis of the comparative measurements. The claims should then be carefully worded to warn about the threats to validity. The statistical evaluation is to be considered in combination with Principles 1 and 2, to provide sufficient backing for the results.

Cloud computing experiments may include numerous hidden services. This can produce high volumes of data with particularly artificial statistical properties, such as unusual distributions or complex dependencies. Experiments may also depend on factors that are outside experimenter control. The methods used for statistical analysis should therefore be considered with care. For example, although Student's or Welch's t-test are often viable options for data with a normal distribution, more advanced methods may be needed in other cases (FBZL16; Jam91; EM97; PAEÅ+16). With the availability of ample computing power, bootstrap and similar sampling methods make for much more robust alternatives (Hes15).

Violation Examples. Although the experimenters cannot be unaware of the stochastic nature of their measurements, statistical evaluation of result significance is rare for cloud experimentation. In fact, Principle 6 is the one most often violated in our survey. It is possibly also the most difficult point to meet because, without robust statistical grounding of the experiment design around Principles 1 and 2, simple application of statistical formulas cannot lend support to claims reported in the surveyed papers.

P7: **Measurement units**. For all the reported quantities, report the corresponding unit of measurement.

Although this principle may seem trivial, reporting the units of measurement of the different reported quantities is essential to better understand the relevance of the presented results, to compare with other approaches, and to analyze the correctness of the mathematical operations involved.

Violation Examples. Although the few cases where a quantity without units appears are possibly simple omissions, there are cases where wrong or ill-defined units are used. For example, memory usage can be measured in many different ways, such as page-level VM-instance metrics vs. heap occupation by an application with byte granularity—these two measurements inform about very different quantities despite having the same unit.

P8: **Cost**. Every cloud experiment should include (i) the cost model used or assumed for the experiment; (ii) accounted resource usage (per second), independently of the model; and (iii) charged cost according to the model.

A distinguishing feature of cloud settings is the offer of services subject to cost, with explicit and implicit guarantees (Service Level Agreements, SLAs). The charged cost (reported as item 3 of this principle) is derived from a cost model (i), provided by the cloud operator and accepted by the cloud user, and accounted resource usage (ii), provided by the cloud monitoring system. For point 1, the cost model and the SLA used in the experiments must be documented explicitly, especially when they may differ per customer. For example, TPC[10] and LDBC[11] require explicit descriptions of the cost model that a generic client would have to follow to use the system-under-test over a period of three years, including licenses and maintenance pricing, and all other elements relevant for real-world use. In contrast, Amazon's EC2 has an explicit cost model, i.e., piece-wise linear addition of hourly usage intervals (and a per-second billing model since the end

[10] As defined for all benchmarks, see `http://www.tpc.org/pricing/`. For example, TPC-C, Clause 7: `http://www.tpc.org/tpc_documents_current_versions/pdf/tpc-c_v5.11.0.pdf`

[11] In its by-laws, v.1.0, published in 2017.

of 2017[12]), and several implicit clauses explaining the SLA applied to all clients, but also uses other models, such as advance reservation and spot markets, with finer but different granularities. All cost elements that still differ across public cloud operators should be documented.

2.4 The Principles behind the Principles

Although we have presented and analyzed the principles in turn, many *principles are connected*. For example, a cloud experiment that seeks to present general results applicable across multiple cloud providers must balance between the need for many experiments across many platforms and the cost of conducting all those experiments. By following Principle 6, the experimenter may express the robustness of the results in specific and quantifiable terms. In turn, this permits keeping the repetitions required by Principles 1 and 2 to the minimum required to achieve selected result significance. Additionally, Principle 8 informs the reader about reproducibility costs, and Principle 4 facilitates doing so without the need to re-implement the experiment setup.

Our list includes only principles that our experience shows pose interesting challenges in cloud environments. However, *our list of principles related to cloud computing is non-exhaustive*. For example, from the more general issues in experiment design and evaluation that we survey in Section 5, we draw attention to *measurement perturbation*.

The act of measuring performance in a computing system may affect the system behavior, creating perturbations that affect the measurements (KABM12). These perturbations increase if the measurements are frequent or if the instrumentation is particularly intrusive (MR91). One way to address this problem is to obtain a model that quantifies the perturbation effects, and use this model in the analysis of the results, to remove or reduce the impact of these effects during analysis (MR91; MRW92). In some cases this approach may be impractical as the configuration of such model is not trivial. For metrics that are captured by the cloud provider, any perturbations introduced by the instrumentation are already part of the variability of the experimental results, because such perturbations are not under the control of the experimenter. For metrics that are captured optionally or for those where the benchmark introduces additional instrumentation, careful implementations can minimize perturbations by limiting the frequency at which the system is observed and by leveraging efficient sampling and processing techniques.

[12] `https://aws.amazon.com/de/blogs/aws/new-per-second-billing-for-ec2-instances-and-ebs-volumes/`

3 Can the methodological principles be applied in common practice?

This section focuses on **RQ2**, and analyses how the principles relate to common practice. Through an example of each, we analyze: (i) how the principles are embodied by commercial benchmarks, and (ii) how the principles can be supported in a common use-case.

3.1 Principle use in Benchmarks

In this section, we describe how the experiment principles defined in Section 2.3 are embodied in standard, commercial, independent industry benchmarks. We discuss Standard Performance Evaluation Corporation (SPEC) and Transaction Processing Performance Council (TPC) benchmarks, with a focus on SPEC Cloud IaaS (Cor16).

Kistowski et al. define a benchmark as a "Standard tool for the competitive evaluation and comparison of competing systems or components according to specific characteristics, such as performance, dependability, or security" (vKAH$^+$15). This definition has several consequences that set benchmarks apart from other performance experiments. For example, it implies that benchmarks are designed to be run by third parties on their systems, without the intervention of the original developers of the benchmark. In particular, commercial benchmarks must have clear run-rules and system definitions, and define beyond controversy their system scope, context, reporting rules, and acceptance criteria and processes for benchmark results. In other words, commercial benchmarks must ensure a form of technical reproducibility.

P1: Many industry-standard benchmarks define the number of runs that are required to achieve a valid result. This number is usually programmed into the benchmark harness and automatically run by every benchmark user. For example, SPEC Cloud IaaS (Cor16) uses five separate re-runs for its baseline experiment. Some benchmarks also allow a varying number of runs, for example, SPEC CPU 2017 (BLvK18) allows either two or three runs.

P2: Many standard industry benchmarks use multiple workloads that are run independently. SPEC Cloud IaaS runs a transaction workload using Apache Cassandra, and a K-Means MapReduce workload. SPEC CPU is based on nearly twenty integer and floating point programs. For research purposes, additional workloads have been created for the SPEC CPU 2017 suite (ABA$^+$18). In addition to their workload collections, the standard benchmarks usually feature rules on the order of workload execution, workload durations, sequences, and even potential pauses, such as the 10-second idle pause between each workload execution phase in SPECpower_ssj2008 (Lan09).

P3: Standard industry benchmarks feature a set of reporting rules that specify how to report the characteristics of the SUT for benchmark acceptance. Reports are reviewed by a committee or auditor. The reviewers evaluate whether the report ensures technical reproducibility by third-parties.

P4: Typically, industry-standard benchmarks document their methodology and execution in great detail, enabling deep understanding of their internal workings. SPEC Cloud 2016, in particular, released its benchmarking harness *cbtool* as an Apache-licensed open-source artifact.

P5: Standard benchmarks report average performance values, with some exceptions. SPEC CPU 2017 reports the median value of three runs, or the minimum if only two runs where executed. SPEC Cloud IaaS reports the average of the 99th percentile measured for latency. The SPEC SERT$^{\text{TM}}$2 reports the coefficient of variation (CV) for performance.

P6: is usually out of scope for released benchmarks. The variance of result scores is usually checked thoroughly by prior to a release by a commitee. Evaluation of multiple results consists simply of comparing the final metric score. Notably, SPEC SERT$^{\text{TM}}$2 sets CV-thresholds above which a test result is considered invalid.

P7: All benchmarks report their unit of measurement, which is usually throughput, response time and some additional metrics, e.g., the number of application instances in SPEC Cloud IaaS 2016.

P8: Benchmarks may use a cost component as part of their metric, depending on domain. The TPC requests a monetary pricing element and details the rules for specifying it. SPEC Cloud IaaS 2016 counts application instances as its cost component, and benchmarks focused on power consumption report Watts. Some benchmarks (e.g., TPC-C) include a price-per-performance metric.

3.2 Applying the Principles

We now show, by example, how all principles proposed in Section 2.3 can be applied and reported in common practice. The case study in this section investigates the hypothesis *H: The scaling behavior of a standard, reactive, CPU utilization-rule-based auto-scaler depends on its environment.* Measurements are presented in accordance to **Principles 1–8**. The case study uses an auto-scaler for a CPU-intensive application—an implementation of the LU (lower-upper decomposition of a $n \times n$ matrix) worklet from the SPEC SERT^{TM}2 benchmark—that is used as a benchmark in three different environments: (i) a CloudStack-based private cloud (CSPC), (ii) AWS EC2, and (iii) the DAS-4 IaaS cloud of a medium-scale multi-cluster experimental environment (MMEE) used for computer science (BEdLm16).

To reject or accept the hypothesis, we use an analysis of variance (ANOVA) test to determine if the performance of the auto-scaler depends on the environment. For these experiments we use the open-source framework BUNGEE adopting its measurement methodology and metrics for auto-scaler evaluations (HKWG15).[13] The performance of the auto-scaler can be described with a set of system- and user-oriented metrics. The system-oriented metrics are elasticity metrics endorsed by the Research Group of the SPEC (HKO$^+$16). The *under/(over)-provisioning accuracy* θ_U (θ_O) is the amount of missing (superfluous) resources required to meet the SLO in relation to the current demand normalized by the experiment time. The *under/(over)-provisioning time share* τ_U (τ_O) is the time relative to the measurement duration, in which the system has insufficient resources (resources in excess). A precise definition of each system-oriented metric can be found in earlier related works (HBK$^+$18; IAEH$^+$18) Knowing the load intensity over time from the replayed trace, the ideal resource supply is derived from repeated and systematic load tests for each scaling level of each environment as part of the BUNGEE measurement methodology (HKWG15).

The implemented auto-scaler and experiment data are available at `https://doi.org/10.5281/zenodo.1169900` (ano18). We use the authentic, time-varying trace of the FIFA championship 1998.[14](→ **Principle 4 is fulfilled because all the experiment software and data are open-access, online.**)

We choose a sub-trace containing three similar days for internal repetitions, and run each trace in each environment. To cover setups with background noise, the application is deployed in both the public AWS EC2 IaaS cloud and in an OpenNebula[15]-based IaaS cloud of a medium-scale multi-cluster experimental environment (MMEE) used exclusively for these experiments. (→ **The use of different environments fulfills Principle 2.**) To have long, representative experiments, each experiment lasts 9.5 hours—a duration that includes the main concerns, e.g., the daily peaks. Due to the scope of this work and space constraint, we skip the analysis of different worklets or applications, including other load traces. (→ **The combination of long-time experiments with internal repetitions fulfills Principle 1.**)

In the CSPC scenario, the application is deployed in a private Apache CloudStack[16] cloud that manages 8 identical virtualized Xen-Server (v6.5) hosts (HP DL160 Gen9 with 8 physical cores @2.4Ghz (Intel E5-2630v3)). We deactivate hyper-threading to limit VM overbooking and rely on a constantly stable performance per VM. Dynamic frequency scaling is enabled as default and also further CPU-oriented features are not changed. The hosts have installed 2x16 RAM. The specification of each VM in all setups is listed in Table 3.1. For all scenarios, Tomcat 7 is the application server. As the LU worklet is CPU-bound, we do not have relevant disk I/O during the experiments and only low utilisation on the Gigabit Ethernet of the hosts. In all three deployments, the auto-scaler is configured identically to up-scale VMs when an average CPU

[13]BUNGEE Elasticity Measurement Framework: `https://descartes.tools/bungee`
[14]FIFA Source: `http://ita.ee.lbl.gov/html/contrib/WorldCup.html`
[15]OpenNebula: `https://opennebula.org/`
[16]Apache CloudStack: `https://cloudstack.apache.org/`

utilization threshold of 90% is exceeded for 1 minute and to scale VMs down when the average CPU utilization falls below 60% for 1 minute. CPU utilization is measured inside the VMs using the TOP command and averaged across all VMs currently running. (→ **This experimental description fulfills Principle 3.**)

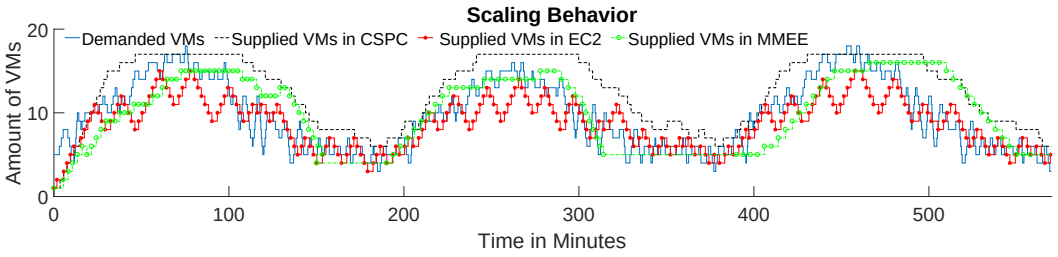

Figure 3.1: The number of VMs allocated by the Reactive auto-scaler, in the Private, EC2, and MMEE IaaS Cloud experiments.

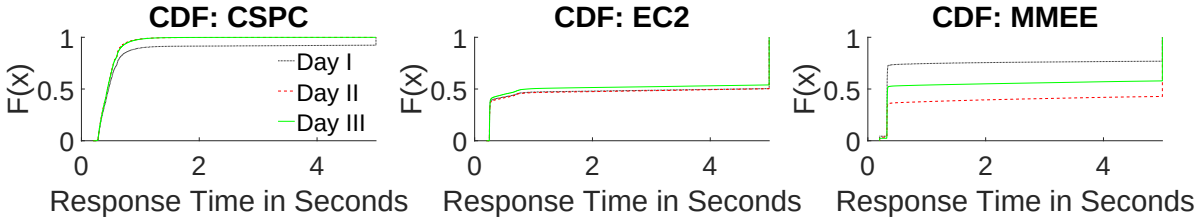

Figure 3.2: Distribution of response times, per experiment.

Figure 3.1 shows the scaling behavior of the auto-scaler, for each environment. The horizontal axis shows the time of the measurement, in minutes, since the beginning of the experiment; the vertical axis shows the number of concurrently running VMs; the blue curve shows the ideal number of supplied VMs. The green, dashed line represents the supplied VMs in MMEE; the red line shows the supplied VMs in EC2; and the black, dashed curve the shows the supplied VMs in CSPC. Figure 3.2 depicts the distributions of the response times per day and Figure 3.3 shows the distribution of allocated VMs per day. In both figures, the dotted black curve represents the first day, the dashed red curve the second day, and the solid green curve the last day. Whereas the distributions of each day in CSPC and EC2 are similar, they differ from MMEE distributions. This can be explained by the scaling behavior depicted in Figure 3.1: during the first day, the auto-scaler allocates too few instances, during the second day the auto-scaler almost satisfies the demand, and during the third day the auto-scaler over-provisions the system. Table 3.2 shows the average metrics and their standard deviation. Furthermore, Table 3.3 shows the used instance hours and the charged instance hours. EC2 uses an hourly-based pricing scheme, whereas for CSPC and MMEE we have applied a minute-based pricing scheme. (→ **The presentation of the results fulfills Principles 5, 7, and 8.**)

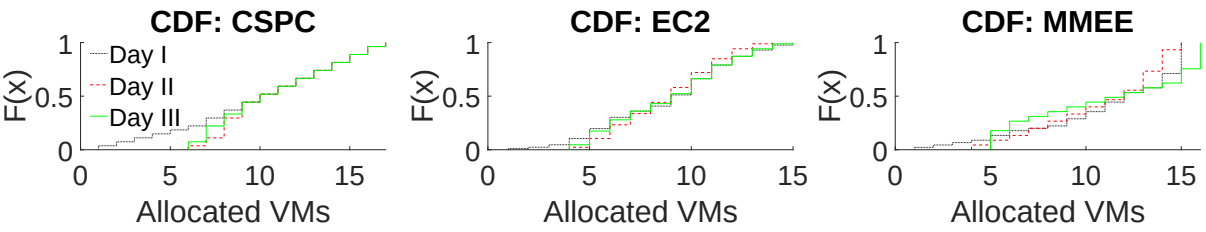

Figure 3.3: Distribution of allocated VMs, per experiment.

Table 3.1: Specification of the VMs.

Component	**CSPC**	**EC2 (m4.large)**	**MMEE**
Operating System	CentOS 6.5	CentOS 6.5	Debian 8
vCPU	2 cores	2 cores	2 cores
Memory	4GB	8GB	2GB

Table 3.2: Average metric (and standard deviation) for a day in each scenario.

Metric	**CSPC**	**EC2**	**MMEE**
θ_U (accuracy$_U$)[%]	2.39 (1.54)	14.05 (1.82)	19.42 (5.04)
θ_O (accuracy$_O$)[%]	43.22 (4.38)	10.09 (1.75)	54.98 (11.87)
τ_U (time share$_U$)[%]	9.76 (4.77)	57.20 (2.60)	42.16 (1.76)
τ_O (time share$_O$)[%]	82.95 (5.46)	27.53 (4.42)	53.06 (3.08)
ψ (SLO violations)[%]	2.70 (3.68)	49.30 (1.71)	53.02 (7.11)
Avg. response time [s]	0.60 (0.17)	2.68 (0.08)	2.32 (0.68)
#Adaptations	25.67 (1.88)	80.66 (3.40)	39.67 (7.54)
Avg. #VMs [VMs]	10.53 (0.44)	8.84 (0.07)	11.01 (0.12)

To investigate the hypothesis "*The scaling behavior of a standard, reactive, CPU utilization-rule-based auto-scaler depends on the environment*", we formulate the null hypothesis H_0: "*The elasticity metrics do not depend on the environment*". We conducted an ANOVA test with the confidence set to the strict value of 1%. Table 3.4 shows the proportion of variance and p-value

Table 3.3: Cost overview of the experiments.

Instance hours	**CSPC**	**EC2**	**MMEE**
Used [h]	121.0	83.4	93.8
Charged [h]	121.0	131.0	93.8

Table 3.4: ANOVA results per metric.

Statistic	θ_U	θ_O	τ_U	τ_O
$Pr(> F)$	0.006	0.001	0.003	0.003
Prop. of Var. due to Env. [%]	82	84	98	97

($Pr(> F)$) for each elasticity metric subject to the environment. The proportion of variance is the variance of each elasticity metric caused by the environment. As each $Pr(> F)$ is less than 1% together, and a high proportion of variance is due to the environment, we can reject the null hypothesis. This confirms our claim is statistically correct. ($\rightarrow$ **The hypothesis analysis fulfills Principle 6.**)

4 How are cloud performance currently obtained and reported?

This section addresses **RQ3**, by analysing the current status of published academic research and industrial practice in cloud computing. Concretely, we analyse the adherence to our methodological principles of papers focusing on cloud computing. We adopt a systematic literature-review approach (BB06), covering papers published in 16 top-level conferences and journals, between 2012 and 2017.

4.1 Systematic Literature Review

The systematic literature review (BB06) is a structured method to provide an overview of a research area. In this work, we follow the guidelines discussed by Peterson et al. (PFMM08; PVK15). Figure 4.1 summarizes our workflow:

Specify Research Questions (RQs). The RQs that we considered are **RQ1**–**RQ3** defined in Section 1. The systematic literature review process focuses mostly in answering **RQ2**. The answers to these questions provide an overview of the existing studies including the number of publications, and the distribution of publications over publication venues and years in the cloud-computing research area.

Specify Search String. After defining the RQs, the next step is to specify the search string that is used to search for relevant publications. The search string is based on the keywords and their alternative words that are in line with the main research goal of the paper. We use the Boolean operators OR and AND to join the keywords and their synonyms in the search string. The following string is used to search relevant publications in the known databases:

`"cloud"` AND `"management"`
AND NOT(`"security"`) AND (YEAR>=2012)

Identify Publication Sources/Databases. We selected 16 venues where cloud computing papers are published; they are in our view the top-level conferences in the field. We query the DBLP computer science bibliography (Ley02) to obtain bibliographic information on all the papers published in the selected venues and years. For each paper, we obtain the link to Semantic Scholar[17], a comprehensive database that allows to easily automate the extraction of publication metadata. Using the Scrapy web crawling framework and the Splash JavaScript rendering service, we obtain and parse the metadata of each paper, including title and abstract. Finally, we check if the title or abstract contain the specified search string, and remove duplicates. Table 4.1 summarizes the results for the query through Semantic Scholar.

Study Selection Criteria The search results from the previous step provide a pool of 358 research publications which constitute the current body of knowledge in experimental evaluation in cloud computing. This analysis includes papers that either had an *impact* in the scientific community *OR* that have been *published recently*. We quantified the *impact* with the number of citations on Google Scholar, requiring it to be greater or equal to 15, and the *published recently* by selecting all the papers published since 2016 (i.e., in the last two years). After this step, 191 papers remain for analysis.

Data Analysis. The relevance of each paper is manually classified, as:

- **Relevant** (`R`) – Papers that include experimental results obtained in a real (or realistic) environment, and the paper is not based exclusively on simulation results.
- **Not Relevant** (`NR`) – Papers that do not include any experimental result, or that include only simulation results.

The relevant (`R`) papers are then manually analyzed, to identify, for every principle in Section 2.3, if they are:

[17] `https://www.semanticscholar.org/`

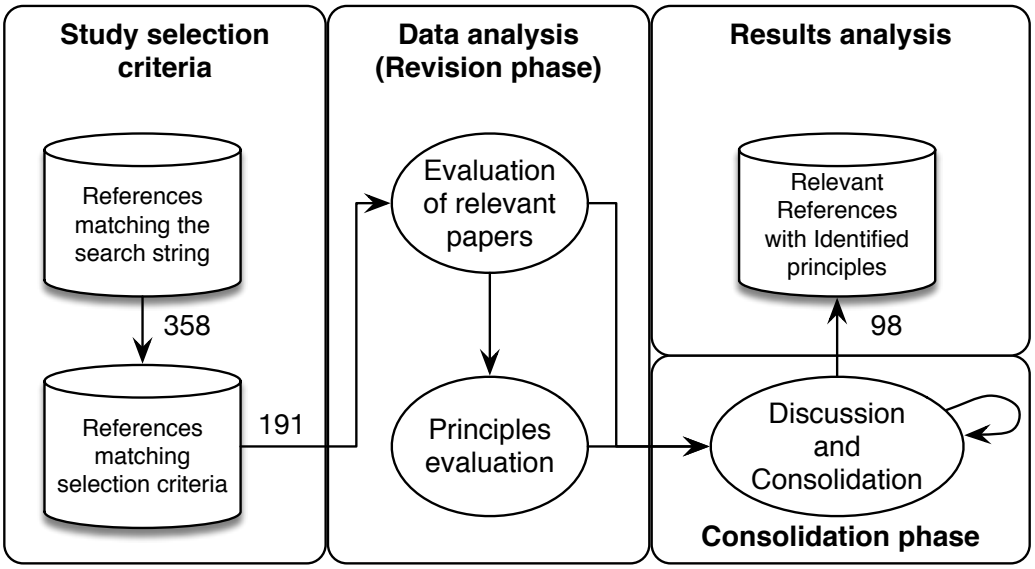

Figure 4.1: Workflow for the systematic literature review.

- **Present** (`Y`) – The evaluated principle is fully met.
- **Partially present** (`P`) – The evaluated principle is present, but does not completely match the described criteria.
- **Not present** (`N`) – The evaluated principle is not present.

To account for the bias of manual classification and analysis, we have conducted a two-step review process. Similarly to a conference-review process, we have assembled a team of 9 expert reviewers in the field of cloud computing, performance engineering, and related topics. Because each expert can review any paper for use of our principles, we have randomly assigned them to the papers. Every paper was reviewed by exactly two different reviewers. In the first step, two reviewers were assigned to each publication, and they had to evaluate *independently* both if the paper is `R` or `NR`, and, if the paper was considered to be `R`, to what extent the principles described in Section 2.3 are fulfilled in the paper. In the second step, the two reviewers have access to the information of the other reviewer, and can discuss to reach an agreement (i.e., on the relevance of the paper, and on the principle evaluation). As in typical conferences, full agreement was not required, and reviewers could change their first-round decisions (i.e., on the fulfillment of each principle, and even on the relevance of the paper).

At the end of the two-round reviewing process, of the 191 papers, 98 were considered relevant by both reviewers, there was a disagreement on 2 papers, and the remaining 91 papers were considered not relevant by both reviewers. We next discuss these results in detail.

Table 4.1: Matching keywords per venue, per year.

Venue	**Total**	**2017**	**2016**	**2015**	**2014**	**2013**	**2012**
IEEECloud	96	0	24	24	14	15	19
UCC	68	0	6	14	30	13	5
CCGrid	31	10	4	0	11	0	6
TPDS	31	8	6	6	4	5	2
IC2E	22	0	5	0	12	5	0
CloudCom	20	2	7	5	6	0	0
ICAC	18	3	3	6	4	1	1
ICPE	18	4	1	3	4	5	1
TCC	15	4	6	3	1	1	0
SoCC	11	0	0	3	4	3	1
HPDC	9	0	3	1	0	2	3
SIGMETRICS	6	1	0	0	2	1	2
FGCS	5	1	1	0	3	0	0
EuroSys	3	0	1	1	0	0	1
SC	3	0	0	0	1	2	0
NSDI	2	0	0	0	1	1	0
Total	358	33	67	66	97	54	41

4.2 Analysis of Reviewer Agreement

To assess the level of agreement between reviewers, we performed a statistical evaluation for the relevance analysis and for the principle analysis. We use the Fleiss' Kappa analysis (Coh60; Fle71) to measure the degree of agreement between reviewers who rate a sample of a cohort; in our work, the cohort is the complete set of papers in the reviewing process. The Fleiss' Kappa analysis computes a value κ, which quantifies the level of agreement and factors out agreement due to chance. According to (LK77), $\kappa < 0$ corresponds to poor agreement, $\kappa \in [0.01, 0.2]$ corresponds to slight agreement, $\kappa \in (0.2, 0.4]$ corresponds to fair agreement, $\kappa \in (0.4, 0.6]$ corresponds to moderate agreement, $\kappa \in (0.6, 0.8]$ corresponds to substantial agreement, and, last, $\kappa \in (0.8, 1]$ corresponds to almost perfect agreement. We also evaluate the percentage of agreement, computed as the number of agreed values over the total number of decisions.

Table 4.2 quantifies the reviewer agreement in terms of percentage of agreement (`%A`), and in terms of the κ statistic. Reviewers reached an almost perfect agreement ($\kappa > 0.8$) for P1–5 and for P7-8. For the remaining P6, "Statistical evaluation", the κ value of 0.59 indicates moderate agreement. However, the percentage of agreement for P6 is high, at 94.9%. For P6, the low value of κ is due to some disagreement in the evaluation: 5 papers out of 98 were evaluated differently by the two reviewers. This result is mainly due to two factors: (i) as discussed in the next section, in most ($> 92\%$) of the papers, the principle has been assessed by the reviewers as not present. This affects the probability distribution of the evaluation, and therefore, even small disagreement between the reviewers significantly decreases the κ value, and (ii) this result may be due to the wide scope of the definition, which can leave room for different interpretations. Since the percentage of agreement is very high, we argue that factor (i) is more likely to have led to a low value of κ, and keep the definition of P6 as stated.

Although the level of agreement of the reviewers is very high, it is not perfect. This highlights that even experts can find it difficult to assess consistently the proposed principles. Cases where such difficulty was apparent include:

P1: It is not clear how the experiments were repeated, and if they had the same configuration across repetitions.

P2: It is not clear how to define the "configuration", especially if it needs to change contextually.

P3: For the experimental setup, too few details are given, especially for (opaque) commercial cloud infrastructures.

P4: It is unclear if the principle is met when the paper links to the code/data, but it is not currently accessible.

P5: The measure of variance appears only in the graph, and in particular is not quantified or discussed in text.

P6: Though statistical evaluation methods are used, they are not the most appropriate, and/or important hypotheses are not tested (e.g., normality of data, equal variance).

P7: Only one '`Y` vs. `P`' (strong) disagreement appeared, related to incomplete measurement units in graphs.

P8: Some experiments may not directly relate to cost model, because they focus on the more technical aspects.

For this study, the two-step reviewing process used formulated principles to evaluate cloud experiments—when in doubt, the reviewers can read again the principles. This led to reduced presence of subjective judgment, as results in this section indicate. For current peer-reviewing process in conferences and journals, common knowledge about the principles presented here should be helpful for peers to agree about their application. We envision that, in the future, automated tools could facilitate following the principles.

Table 4.2: Fleiss' Kappa agreement analysis.

	Rel.	P1	P2	P3	P4	P5	P6	P7	P8
%A	99.0%	94.9%	88.8%	91.8%	92.9%	92.9%	94.9%	99.0%	90.8%
κ	0.98	0.90	0.82	0.86	0.83	0.86	0.59	0.96	0.82

4.3 Evaluation of Principle Application

This section analyses how the principles are currently applied, using the results of the two-step review process. Figure 4.2 shows the numerical results of the analysis over the eight identified principles. Overall, we find that most of the principles are not followed in the analyzed papers, which is an an important result because some of the principles seem easy to verify. This finding is further worrying due to the quality of the publications we study, implied by the selection process (see Section 4.1) for which: (i) we have selected some of the top venues in the area of cloud computing, and, (ii) among the found papers, we have selected either papers with a high number of citations or very recent.

P1 highlights that more than two-thirds of the analyzed papers do not execute any repeated experiments or long runs (label `N` in Figure 4.2, sub-plot P1), and only 21% do both. This can significantly impact the generalization of the obtained results, because there are a number of factors that can affect the results even without any changes to the controlled configuration of the tested cloud-systems.

P2 shows that less than 47% of the analyzed papers include a complete performance evaluation with multiple configurations. Varying configurations can be challenging in some scenarios, due to timing, cost of cloud service, and other factors. However, from a scientific perspective, different configurations may significantly impact the overall performance and more extensive evaluations are needed (PAEÅ^{+}16).

P3 discusses the experimental setup description. Even though more than 52% of the analyzed papers fully cover this principle, a substantial number of papers do not or only partially describe the experimental setup in which the performance evaluation is conducted. This significantly impacts the technical reproducibility of the results.

P4 partly complements P3, as it considers the accessibility of the datasets used in the analysis and whether the authors have released the source code. In more than 70% of the cases, the code is not released and the datasets are not publicly available. Reproducibility seems impossible in this situation. The joint effect of not having a complete experimental setup description (P3), and not having the code and datasets used for the evaluation (P4) makes it also difficult for future experimental evaluations of novel techniques to compare with the published approach.

P5 focuses on how results are reported. Although many of the authors of the papers we study argue themselves that uncertainty and stochastic processes are common in cloud computing, and use this as motivation for their work, more than 63% of the papers do not report their measurement variances and limit their reporting to averages.

P6 analyzes if a statistical evaluation has been performed, to include some (statistical) confidence in the results. This principle is the most disregarded by the papers we study, with `N` $>$ 90%. Statistical evaluations are practically the standard practice in other fields (e.g., medicine, physics, biology, and in computing science database systems and software engineering), but according to this result, the field of cloud computing has paid less attention to this aspect. This may be due to the low rates of fulfillment of P3 and P4, which complicates the comparison of different approaches, and also it can be related to the fast evolution of cloud technologies. In the long run P6 should be much more carefully considered, and assessed for by reviewers.

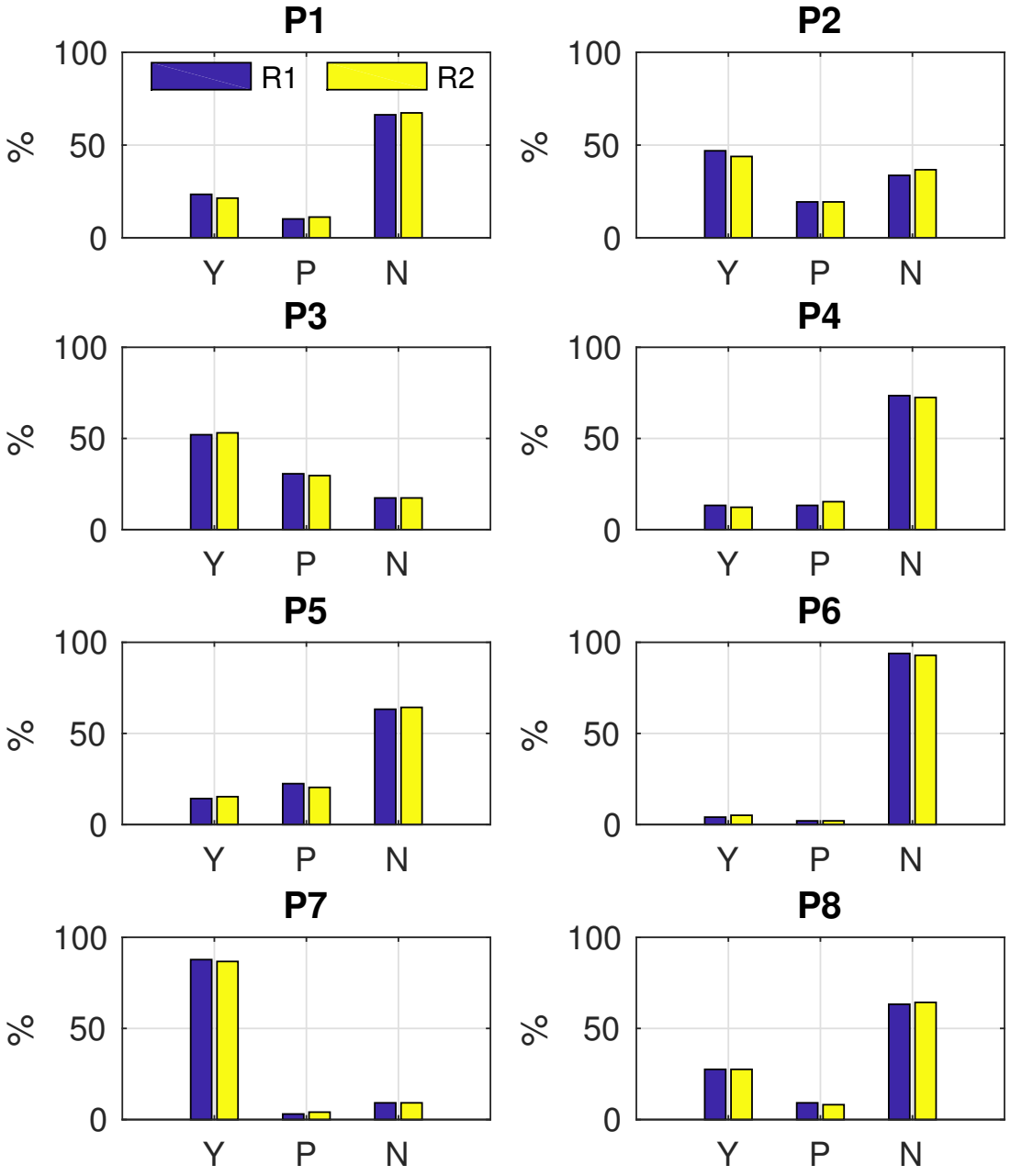

Figure 4.2: Evaluation of P1–8, one sub-plot per principles. Columns R1 and R2 summarize results by 1st/2nd reviewer.

P7 refers to the presence of the units of measurement throughout the paper. We find P7 is the principle with the highest fulfillment, with `Y` $> 85\%$. Only about 12% of the papers do not or only partially include all the units of measurement. Even though the percentage of "`P` or `N`" is low, its non-zero value justifies why P7 remains relevant.

P8 is not present in more than 63% of the considered papers. A cost model may be sometimes difficult to include in the performance evaluation of the system at hand, but it quantifies the economic advantages of adopting one technique rather than another. The lack of standardized benchmarks in cloud computing is partly justifying the scarce discussion of cost models in scientific papers.

The analysis shows that the proposed principles have not yet been extensively adopted in the cloud computing community. Their usage would improve the reproducibility of results and enable more comparative studies in the future.

5 Related work

An increasing body of work focuses on the methodological principles of experimental performance evaluation, in communities focused on high performance computing, performance engineering, security, and general computer science. We survey the body of knowledge closest to our work, grouped by community. In contrast to this body of work, our work focuses on cloud computing environments, for which we raise distinctive challenges due to closed and opaque environments (affects P1–P3, and occasionally P4 for closed-source stacks), dynamic (elastic) resources and services (P5–P6), and use of explicit cost models for operation (P8).

Cloud community: Closest to our work, Folkerts et al. (FAS$^+$13) and Iosup et al. (IPE14) identify challenges in cloud benchmarking; some of the principles in this work formulate an approach to address these challenges. Also close to this work, Schwartzkopf et al. (SMH12) propose a set of seven principles, of which the second and the sixth apply to cloud computing and roughly overlap with our principles P5 and P6. They do not survey published work in the community. The examples they propose (their own) do not or only partially meet 5 of our principles and 5 of their own.

Performance engineering community: Much attention is paid to the ability to reproduce experiments with reasonable effort. Frameworks like the Collective Knowledge Framework (FLP16) aim at systematic recording of individual experiment steps that permits independent reproduction and contribution of additional results. As unexpected effects can appear during performance evaluation due to relatively unexpected properties of the experimental platform (MDHS09; KT06), environments such as DataMill (dOPRF13) can randomize selected environmental conditions and thus improve the ability to generalize from particular measurements. Also, works such as (GBE07; HLST15) provide guidelines how to avoid most common experimental evaluation pitfalls on specific platforms.

Computer Systems community: the high-performance computing community has proposed and updated its guidelines for conducting experiments and for benchmarking. Closest to our work, and most recently, Hoefler and Belli (HB15) summarize and propose 12 rules that enable interpretability, which they define as "a weaker form of reproducibility". Their rules are consistent with our principles but apply to supercomputing environments. As main differences from the cloud community: the supercomputing community focuses on speedup as primary metric (this is slowly changing, witness several keynotes and award-lectures at SC17), uses primarily kernel-based benchmarks and not entire workloads to experiment with (our P4), much of the tested software is difficult to compile and run in other environments (invalidates the meaning of shared software in our P4), does not report operational costs (in contrast to our P8), etc.

Frachtenberg and Feitleson (FF05) focus on scheduling in large-scale computing systems, in particular supercomputing facilities, for which they provide a framework with 32 pitfalls, each combining practical principles and research challenges. Their seminal work combines general and domain-specific issues, and even proposes a rate of severity for each pitfall, but is not validated against real-world publications and does not provide examples of experiments that avoid the pitfalls. Iosup et al. (IEF$^+$06) focus on domain-specific issues, and propose principles of metrics and workload characteristics to use in grid-computing experiments. Mytkowitz et al. (MDHS09) and Zhang et al. (ZIP$^+$10) identify technology counterparts to our principles: sources of measurement bias in experimental work corresponding to large-scale systems.

For networked systems, Krishnamurthy et al. (KWGA11) provide 12 activities that researchers can use to check their experiments; 5 of these overlap roughly with 5 of our principles. They draw observations about the application of the proposed activities in several of their articles, and in one hundred articles experimenting with a particular dataset. They do not conduct a systematic survey such as ours, or a single experiment that follows all proposed activities.

Closely related to this work, Vitek and Kalibera discuss common mistakes made by re-

searchers that decrease the value of their work (VK12). Examples are proprietary data (our P4), weak statistics (our P5 and P6), and meaningless measurements. A case study is performed to illustrate such mistakes. Several recommendations are provided to improve quality with respect to the use of statistical methods (our P5 and P6), documentation, repetition (our P1 and P2), reproducibility, and benchmarks.

Others have discussed the status of the performance evaluation of computer systems, including the characterization of "sins" of reasoning when performing experimental evaluations or reporting its results (BDH+16).

Software engineering community: One of the fields of computing science to first use systematic literature reviews is software engineering (KPBB+09). The decade-long longitudinal study of Sjøberg et al. (SHH+05) surveys the use of controlled experiments in software engineering conferences; our work complements theirs with focus on explicit, fine-grained principles. The seminal study of Zannier et al. (ZMM06) addresses in particular hypothesis-driven experimentation in software engineering; this approach is not common today in computer systems research, in part because the field may not lend itself to the correct yet succinct formulation of meaningful hypotheses. Pieterse and Flater discuss several aspects of software performance including CPU utilization, memory usage, and power consumption (PF14). Collberg et al. (CP16) focus on sharing of software, but conduct a study which methodologically overlaps with ours: they conduct a large study of over 600 articles, published in 2012 in over 10 top-quality publication venues covering several areas in software engineering, but also in computer systems, database systems, security systems, and computer architecture. Similarly to our study, they conduct a systematic analysis of all articles published in the target-venues, but their study is only for 2012 and is thus not also longitudinal.

6 Conclusion

This report presents a first attempt to define fundamental methodological principles to allow for reproducible performance evaluation in cloud environments. The main goal of this work is to establish and analyze a minimal set of principles that could be adopted by the cloud computing community to improve the way performance evaluation is conducted. We identified eight principles, combining best-practices from nearby fields, concepts applicable only to the cloud computing domain, but we do not claim that such principles are complete, but this report represents a first attempt to formulate methodological aspects for the performance evaluation in the cloud community.

We showed how such principles can be used in a practical scenario, and we surveyed some of the main venues of the cloud community analyzing to what extent such principles are considered in the papers published in the last 6 years. One of the main results of this study is showing that most of the principles are not or only partially considered in the analyzed papers. The principles are rather simple and basic, yet we are still far from seeing a broad adoption of sound measurement principles for cloud environments.

We strongly believe that, as a community, adopting and possibly complementing these important principles will both improve the quality of our research, and the reproducibility of the results in the cloud community, setting a sound basis for future work.

References

[AB17] A. Abedi and T. Brecht, "Conducting repeatable experiments in highly variable cloud computing environments," in *ICPE*, 2017, pp. 287–292.

[ABA+18] J. N. Amaral, E. Borin, D. Ashley, C. Benedicto, E. Colp, J. H. S. Hoffman, M. Karpoff, E. Ochoa, M. Redshaw, and R. E. Rodrigues, "The Alberta workloads for the SPEC CPU 2017 benchmark suite," in *ISPASS*, 2018.

[ano18] anonymous, "Comparison of state-of-the-art reactive auto-scaler in 3 different environments," feb 2018.

[Bak16] M. Baker, "Is there a reproducibility crisis?" *Nature*, vol. 533, no. 7604, pp. 452–454, 2016.

[BB06] D. Budgen and P. Brereton, "Performing systematic literature reviews in software engineering," in *ICSE*, 2006, pp. 1051–1052.

[BDH+16] S. M. Blackburn, A. Diwan, M. Hauswirth, P. F. Sweeney, J. N. Amaral, T. Brecht, L. Bulej, C. Click, L. Eeckhout, S. Fischmeister, D. Frampton, L. J. Hendren, M. Hind, A. L. Hosking, R. E. Jones, T. Kalibera, N. Keynes, N. Nystrom, and A. Zeller, "The truth, the whole truth, and nothing but the truth: A pragmatic guide to assessing empirical evaluations," *ACM Trans. Program. Lang. Syst.*, vol. 38, no. 4, pp. 15:1–15:20, 2016.

[BEdLm16] H. Bal, D. Epema, C. de Laat, and more, "A medium-scale distributed system for computer science research: Infrastructure for the long term," *IEEE Computer*, vol. 49, no. 5, pp. 54–63, 2016.

[BGHK18] A. Bauer, J. Grohmann, N. Herbst, and S. Kounev, "On the value of service demand estimation for auto-scaling," in *MMB*, 2018.

[BLvK18] J. Bucek, K.-D. Lange, and J. von Kistowski, "SPEC CPU2017 – next-generation compute benchmark," in *ICPE*, 2018.

[Boi16] R. F. Boisvert, "Incentivizing reproducibility," *Commun. ACM*, vol. 59, no. 10, pp. 5–5, 2016.

[CFA+07] J. Cavazos, G. Fursin, F. Agakov, E. Bonilla, M. F. P. O'Boyle, and O. Temam, "Rapidly selecting good compiler optimizations using performance counters," in *CGO*, 2007, pp. 185–197.

[Coh60] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and Psychological Measurement*, vol. 20, no. 1, 1960.

[Cor16] S. P. E. Corporation, "SPEC cloud IaaS 2016 benchmark design overview," Tech. Rep., 2016.

[CP16] C. Collberg and T. A. Proebsting, "Repeatability in computer systems research," *Commun. ACM*, vol. 59, no. 3, pp. 62–69, 2016.

[dOPRF13] A. B. de Oliveira, J.-C. Petkovich, T. Reidemeister, and S. Fischmeister, "DataMill: Rigorous performance evaluation made easy," in *ICPE*, 2013, pp. 137–148.

[EM97] V. J. Easton and J. H. McColl, *Statistics glossary.* Steps, 1997, vol. 1.

[ESL07] E. Eide, L. Stoller, and J. Lepreau, "An experimentation workbench for replayable networking research," in *NSDI*, 2007, pp. 215–228.

[FAS+13] E. Folkerts, A. Alexandrov, K. Sachs, A. Iosup, V. Markl, and C. Tosun, "Benchmarking in the cloud: What it should, can, and cannot be," in *Selected Topics in Performance Evaluation and Benchmarking*, 2013, pp. 173–188.

[FBZL16] J. S. Firoz, M. Barnas, M. Zalewski, and A. Lumsdaine, "The value of variance," in *ICPE*, 2016, pp. 287–295.

[Fei05] D. G. Feitelson, "Experimental computer science: The need for a cultural change," Tech. Rep., 2005.

[Fei16] ——, "Resampling with feedback – a new paradigm of using workload data for performance evaluation," in *Euro-Par*, 2016, pp. 3–21.

[FF05] E. Frachtenberg and D. G. Feitelson, "Pitfalls in parallel job scheduling evaluation," in *JSSPP*, 2005, pp. 257–282.

[Fle71] J. L. Fleiss, "Measuring nominal scale agreement among many raters," *Psychological Bulletin*, vol. 76, no. 5, 1971.

[FLP16] G. Fursin, A. Lokhmotov, and E. Plowman, "Collective knowledge: Towards R&D sustainability," in *DATE*, 2016, pp. 864–869.

[FMMS14] M. Ferro, A. R. Mury, L. F. Manfroi, and B. Schulze, "High performance computing evaluation A methodology based on scientific application requirements," *CoRR*, 2014.

[FW86] P. J. Fleming and J. J. Wallace, "How not to lie with statistics: The correct way to summarize benchmark results," *Commun. ACM*, vol. 29, no. 3, pp. 218–221, 1986.

[GBE07] A. Georges, D. Buytaert, and L. Eeckhout, "Statistically rigorous java performance evaluation," *SIGPLAN Not.*, vol. 42, no. 10, pp. 57–76, 2007.

[GSG+15] M. P. Grosvenor, M. Schwarzkopf, I. Gog, R. N. Watson, A. W. Moore, S. Hand, and J. Crowcroft, "Queues don't matter when you can jump them!" in *NSDI*, 2015, pp. 1–14.

[HAP14] T. Herndon, M. Ash, and R. Pollin, "Does high public debt consistently stifle economic growth? A critique of Reinhart and Rogoff," *Cambridge journal of economics*, vol. 38, no. 2, pp. 257–279, 2014.

[HB15] T. Hoefler and R. Belli, "Scientific benchmarking of parallel computing systems: Twelve ways to tell the masses when reporting performance results," in *SC*, 2015, pp. 73:1–73:12.

[HBK+18] N. Herbst, A. Bauer, S. Kounev, G. Oikonomou, E. van Eyk, G. Kousiouris, A. Evangelinou, R. Krebs, T. Brecht, C. L. Abad, and A. Iosup, "Quantifying Cloud Performance and Dependability: Taxonomy, Metric Design, and Emerging Challenges," *ACM ToMPECS*, vol. 3, no. 4, pp. 19:1–19:36, August 2018. [Online]. Available: http://doi.acm.org/10.1145/3236332

[HE07] K. Hoste and L. Eeckhout, "Microarchitecture-independent workload characterization," *IEEE Micro*, vol. 27, no. 3, pp. 63–72, 2007.

[Hes15] T. C. Hesterberg, "What teachers should know about the bootstrap: Resampling in the undergraduate statistics curriculum," *The American Statistician*, vol. 69, no. 4, pp. 371–386, 2015.

[HHJ+12] N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, and N. McKeown, "Reproducible network experiments using container-based emulation," in *CoNEXT*, 2012, pp. 253–264.

[HKO+16] N. Herbst, R. Krebs, G. Oikonomou, G. Kousiouris, A. Evangelinou, A. Iosup, and S. Kounev, "Ready for rain? A view from SPEC research on the future of cloud metrics," *CoRR*, vol. abs/1604.03470, 2016.

[HKWG15] N. Herbst, S. Kounev, A. Weber, and H. Groenda, "BUNGEE: An elasticity benchmark for self-adaptive IaaS cloud environments," in *SEAMS*, 2015, pp. 46–56.

[HLST15] V. Horký, P. Libič, A. Steinhauser, and P. Tůma, "DOs and DON'Ts of conducting performance measurements in java," in *ICPE*, 2015, pp. 337–340.

[IAEH+17] A. Ilyushkin, A. Ali-Eldin, N. Herbst, A. V. Papadopoulos, B. Ghit, D. Epema, and A. Iosup, "An experimental performance evaluation of autoscaling policies for complex workflows," in *ICPE*, 2017, pp. 75–86.

[IAEH+18] A. Ilyushkin, A. Ali-Eldin, N. Herbst, A. Bauer, A. V. Papadopoulos, D. Epema, and A. Iosup, "An experimental performance evaluation of autoscalers for complex workflows," *ToMPECS*, vol. 3, no. 2, pp. 8:1–8:32, 2018.

[IEF+06] A. Iosup, D. H. J. Epema, C. Franke, A. Papaspyrou, L. Schley, B. Song, and R. Yahyapour, "On grid performance evaluation using synthetic workloads," in *JSSPP*, 2006, pp. 232–255.

[IPE14] A. Iosup, R. Prodan, and D. H. J. Epema, "IaaS cloud benchmarking: Approaches, challenges, and experience," in *Cloud Computing for Data-Intensive Applications*, 2014, pp. 83–104.

[IYE11] A. Iosup, N. Yigitbasi, and D. Epema, "On the performance variability of production cloud services," in *CCGrid*, 2011, pp. 104–113.

[Jam91] M. Jambu, *Exploratory and Multivariate Data Analysis*, 1991.

[JPEJ06] A. Joshi, A. Phansalkar, L. Eeckhout, and L. K. John, "Measuring benchmark similarity using inherent program characteristics," *IEEE Transactions on Computers*, vol. 55, no. 6, pp. 769–782, 2006.

[KABM12] S. Kell, D. Ansaloni, W. Binder, and L. Marek, "The JVM is not observable enough (and what to do about it)," in *VMIL*, 2012, pp. 33–38.

[KPBB+09] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering - a systematic literature review," *Inf. Softw. Technol.*, vol. 51, no. 1, pp. 7–15, 2009.

[KT06] T. Kalibera and P. Tůma, "Precise regression benchmarking with random effects: Improving mono benchmark results," in *EPEW*, 2006, pp. 63–77.

[KWGA11] B. Krishnamurthy, W. Willinger, P. Gill, and M. Arlitt, "A socratic method for validation of measurement-based networking research," *Comput. Commun.*, vol. 34, no. 1, pp. 43–53, 2011.

[Lan09] K.-D. Lange, "Identifying shades of green: The SPECpower benchmarks," *Computer*, vol. 42, no. 3, pp. 95–97, 2009.

[LC16] P. Leitner and J. Cito, "Patterns in the chaos – a study of performance variation and predictability in public IaaS clouds," *ACM Trans. Internet Technol.*, vol. 16, no. 3, pp. 15:1–15:23, 2016.

[Ley02] M. Ley, "The DBLP computer science bibliography: Evolution, research issues, perspectives," in *SPIRE*, 2002, pp. 1–10.

[LK77] J. R. Landis and G. G. Koch, "The measurement of observer agreement for categorical data," *Biometrics*, vol. 33, no. 1, pp. 159–174, 1977.

[LPM+17] E. B. Lakew, A. V. Papadopoulos, M. Maggio, C. Klein, and E. Elmroth, "KPI-agnostic control for fine-grained vertical elasticity," in *CCGrid*, 2017, pp. 589–598.

[Mas04] J. R. Mashey, "War of the benchmark means: Time for a truce," *SIGARCH Comput. Archit. News*, vol. 32, no. 4, pp. 1–14, 2004.

[MDHS09] T. Mytkowicz, A. Diwan, M. Hauswirth, and P. F. Sweeney, "Producing wrong data without doing anything obviously wrong!" *SIGPLAN Not.*, vol. 44, no. 3, pp. 265–276, 2009.

[MR91] A. D. Malony and D. A. Reed, "Models for performance perturbation analysis," *SIGPLAN Not.*, vol. 26, no. 12, pp. 15–25, 1991.

[MRW92] A. D. Malony, D. A. Reed, and H. A. G. Wijshoff, "Performance measurement intrusion and perturbation analysis," *TPDS*, vol. 3, no. 4, pp. 433–450, 1992.

[PAEÅ+16] A. V. Papadopoulos, A. Ali-Eldin, K.-E. Årzén, J. Tordsson, and E. Elmroth, "PEAS: A performance evaluation framework for auto-scaling strategies in cloud applications," *ToMPECS*, vol. 1, no. 4, pp. 15:1–15:31, 2016.

[PF14] V. Pieterse and D. Flater, "The ghost in the machine: don't let it haunt your software performance measurements," Tech. Rep. 1830, 2014.

[PFMM08] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," in *EASE*, 2008, pp. 68–77.

[POZ+16] J.-C. Petkovich, A. B. Oliveira, Y. Zhang, T. Reidemeister, and S. Fischmeister, "DataMill: A distributed heterogeneous infrastructure forrobust experimentation," *Softw. Pract. Exper.*, vol. 46, no. 10, pp. 1411–1440, 2016.

[PVK15] K. Petersen, S. Vakkalanka, and L. Kuzniarz, "Guidelines for conducting systematic mapping studies in software engineering: An update," *Inf. Softw. Technol.*, vol. 64, pp. 1–18, 2015.

[RDS+07] R. Ricci, J. Duerig, P. Sanaga, D. Gebhardt, M. Hibler, K. Atkinson, J. Zhang, S. Kasera, and J. Lepreau, "The Flexlab approach to realistic evaluation of networked systems," in *NSDI*, 2007, pp. 1–14.

[RWS+15] R. Ricci, G. Wong, L. Stoller, K. Webb, J. Duerig, K. Downie, and M. Hibler, "Apt: A platform for repeatable research in computer science," *SIGOPS Oper. Syst. Rev.*, vol. 49, no. 1, pp. 100–107, 2015.

[SBZ+08] M. Sayeed, H. Bae, Y. Zheng, B. Armstrong, R. Eigenmann, and F. Saied, "Measuring high-performance computing with real applications," *Computing in Science Engineering*, vol. 10, no. 4, pp. 60–70, 2008.

[SDQR10] J. Schad, J. Dittrich, and J.-A. Quiané-Ruiz, "Runtime measurements in the cloud: Observing, analyzing, and reducing variance," *Proc. VLDB Endow.*, vol. 3, no. 1-2, pp. 460–471, 2010.

[SHH+05] D. I. K. Sjoberg, J. E. Hannay, O. Hansen, V. By Kampenes, A. Karahasanovic, N.-K. Liborg, and A. C. Rekdal, "A survey of controlled experiments in software engineering," *IEEE Trans. Softw. Eng.*, vol. 31, no. 9, pp. 733–753, 2005.

[SMH12] M. Schwarzkopf, D. G. Murray, and S. Hand, "The seven deadly sins of cloud computing research," in *HotCloud*, 2012.

[SNTH13] G. K. Sandve, A. Nekrutenko, J. Taylor, and E. Hovig, "Ten simple rules for reproducible computational research," *PLOS Computational Biology*, vol. 9, no. 10, pp. 1–4, 2013.

[Soe15] D. A. W. Soergela, "Rampant software errors may undermine scientific results," *F1000Research*, vol. 3, no. 303, pp. 1–13, 2015.

[SPBP06] N. Spring, L. Peterson, A. Bavier, and V. Pai, "Using PlanetLab for network research: Myths, realities, and best practices," *SIGOPS Oper. Syst. Rev.*, vol. 40, no. 1, pp. 17–24, 2006.

[VK12] J. Vitek and T. Kalibera, "R3: Repeatability, reproducibility and rigor," *SIGPLAN Not.*, vol. 47, no. 4a, pp. 30–36, 2012.

[vKAH+15] J. von Kistowski, J. A. Arnold, K. Huppler, K.-D. Lange, J. L. Henning, and P. Cao, "How to build a benchmark," in *ICPE*, 2015, pp. 333–336.

[ZIP+10] B. Zhang, A. Iosup, J. A. Pouwelse, D. H. J. Epema, and H. J. Sips, "Sampling bias in BitTorrent measurements," in *Euro-Par*, 2010, pp. 484–496.

[ZMM06] C. Zannier, G. Melnik, and F. Maurer, "On the success of empirical studies in the international conference on software engineering," in *ICSE*, 2006, pp. 341–350.

Appendix A

1 System-oriented Metrics

Notably, the following metrics as part of the open-source BUNGEE auto-scaler evaluation framework have been endorsed by the Research Group of the Standard Performance Evaluation Corporation (SPEC) (HKO+16). The set of elasticity metrics has been adopted by a number of researchers and applied in several papers, e.g., (IAEH+17) or (BGHK18). For the following equations, let be:

- T the experiment duration with time $t \in [0, T]$
- s_t the resource supply at time t
- d_t the resource demand at time t

The demanded resource units d_t are the minimal amount of VMs required to meet the SLOs under the load intensity at time t. Δt denotes the time interval between the last and the current change either in demand d or supply s. The curve of demanded resource units d over time T is derived by as a preparation step of the BUNGEE measurement methodology (HKWG15). The resource supply s_t is the monitored number of running VMs at time t.

1.1 Provisioning accuracy θ_U and θ_O

These metrics describe the relative amount of resources that are under-provisioned, respectively, over-provisioned during the measurement interval, i.e., the *under-provisioning accuracy* θ_U is the amount of missing resources required to meet the SLO in relation to the current demand normalized by the experiment time. Similarly, the *over-provisioning accuracy* θ_O is the amount of resources that the auto-scaler supplies in excess of the current demand normalized by the experiment time. Values of this metric lie in the interval $[0, \infty)$, where 0 is the best value and indicates that there is no under-provisioning or over-provisioning during the entire measurement interval.

$$\theta_U[\%] := \frac{100}{T} \cdot \sum_{t=1}^{T} \frac{\max(d_t - s_t, 0)}{d_t} \Delta t \tag{A.1}$$

$$\theta_O[\%] := \frac{100}{T} \cdot \sum_{t=1}^{T} \frac{\max(s_t - d_t, 0)}{d_t} \Delta t \tag{A.2}$$

1.2 Wrong provisioning time share τ_U and τ_O

These metrics capture the time in percentage, in which the system is under-provisioned, respectively over-provisioned, during the experiment interval, i.e., the *under-provisioning time share* τ_U is the time relative to the measurement duration, in which the system has insufficient resources. Similarly, the *over-provisioning time share* τ_O is the time relative to the measurement duration, in which the system has more resources than required. Values of this metric lie in the interval $[0, 100]$. The best value 0 is achieved when no under-provisioning, respectively no over-provisioning, is detected within a measurement.

$$\tau_U[\%] := \frac{100}{T} \cdot \sum_{t=1}^{T} \max(\text{sgn}(d_t - s_t), 0) \Delta t \tag{A.3}$$

$$\tau_O[\%] := \frac{100}{T} \cdot \sum_{t=1}^{T} \max(\text{sgn}(s_t - d_t), 0) \Delta t \tag{A.4}$$

2 Discussion of Metric Characteristics

In the following, we include a more detailed discussion of metrics characteristics (cf. Section 2.1) and justify our selection of the elasticity metrics we adopted to showcase how the eight proposed principle can be applied in practice. Different ways to aggregate them have been applied and defined in a related experimental work [citation omitted for blind review]. In general, it is impossible to prove correctness of a metric; it is more a common agreement on how to quantify the given property. One can discuss to what degree metrics fulfill characteristics of a good, well-defined and intuitive metric and additionally demonstrate their usefulness. Let us go in the following step by step over a list of metric characteristics:

Definition. A metric should come along with a precise, clear mathematical (symbolic) expression and a defined unit of measure, to assure consistent application and interpretation. Our selected metrics are compliant with this requirement, as all of them come with a mathematical expression, a unit or are simply time-based ratios.

Interpretation. A metric should be intuitively understandable. The chosen metrics are simplistic metrics that can be described each in a compact sentence. Furthermore, it is important to specify (i) if a metric has a physical unit or is unite-free, (ii) if it is normalized and if yes how, (iii) if it is directly time-dependent or can only be computed ex-post after a completed measurement and (iv) clear information on the value range and the optimal point. Aggregate metrics should keep generality and fairness, combined with a way to customize by agreement on a weight vector.

Measureability. A transparently defined and consistently applied measurement procedure is important for reliable measurements of a metric. For example, it is important to state where the sensors need to be placed (to have an unambiguous view on the respective resource), the frequency of sampling idle/busy counters and the intervals for reporting averaged percentages. The easier a metric is to measure, the more likely it is that it will be used in practice and that its value will be correctly determined. We apply a defined measurement methodology (a.k.a benchmarking approach) for the selected elasticity metrics (HKWG15).

Repeatability. Repeatability implies that if the metric is measured multiple times using the same procedure, the same value is measured. In practice, small differences are usually acceptable, however, ideally, a metric should be deterministic when measured multiple times. For the chosen metrics, it has been shown that repeatability is possible in a controlled experiment environment and with some variability in the public cloud domain (HKWG15).

Reliability A metric is considered reliable if it ranks experiment results consistently with respect to the property that is subject of evaluation. In other words, if System A performs better than System B with respect to the property under evaluation, then the values of the metric for the two systems should consistently indicate this (e.g., higher value meaning better score). In the optimal case, a metric is linear, if its value is linearly proportional to the degree to which the system under test exhibits the property under evaluation. For example, if a performance metric is linear, then twice as high value of the metric would indicate twice as good performance. Linear metrics are intuitively appealing since humans typically tend to think in linear terms. In the long run, the elasticity metric's distributions in reality should be analyzed to improve the reliability of their aggregation.

Independence A metric is independent if its definition and behavior cannot be influenced by proprietary interests of different vendors or manufacturers aiming to gain competitive advantage by defining the metric in a way that favors their products or services. As our selected metrics come with a mathematical definition and a measurement methodology, we claim that it should be possible to verify that the experiments included in this paper have been conducted according to given run-rules.

3 ANOVA Results

In the following Tables 3.1 to 3.4, we include the complete results of the analysis of variance we conducted per elasticity metric comparing the environments and days/repetitions (cf. Section 3.2). The complete set of elasticity metric results per day for all three environments can be found in Table 3.5.

Table 3.1: ANOVA for θ_U considering environment and day.

	DoF	Sum of Squares	F value	Pr(>F)
env	2	455.1	41.865	0.00643
day	1	13.4	2.462	0.21465
env:day	2	63.7	5.860	0.09201
residuals	3	16.3		

Table 3.2: ANOVA for θ_O considering environment and day.

	DoF	Sum of Squares	F value	Pr(>F)
env	2	3801	151.176	0.000974
day	1	266	21.158	0.019315
env:day	2	185	7.373	0.069511
residuals	3	38		

Table 3.3: ANOVA for τ_U considering environment and day.

	DoF	Sum of Squares	F value	Pr(>F)
env	2	3527	67.561	0.0032
day	1	2	0.074	0.8035
env:day	2	17	0.328	0.7431
residuals	3	78		

Table 3.4: ANOVA for τ_O considering environment and day.

	DoF	Sum of Squares	F value	Pr(>F)
env	2	4617	77.243	0.00263
day	1	1	0.024	0.88589
env:day	2	86	1.443	0.36386
residuals	3	90		

Table 3.5: Metric overview for each day for the MMEE vs EC2 vs CSPC scenario.

Metric	**MMEE** Day1	Day2	Day3	**CSPC** Day1	Day2	Day3	**EC2** Day1	Day2	Day3
θ_U (accuracy$_U$) [%]	12.76	20.56	24.96	4.24	0.48	2.45	15.99	11.61	14.54
θ_O (accuracy$_O$) [%]	45.68	58.11	74.65	37.44	48.01	44.20	7.71	10.63	11.93
τ_U (timeshare$_U$) [%]	42.84	39.75	43.90	10.11	3.75	15.42	60.29	54.00	57.32
τ_O (timeshare$_O$) [%]	50.81	57.42	50.95	84.08	89.00	75.78	21.38	29.58	31.63
υ (instability) [%]	14.04	14.00	10.98	14.85	13.25	13.91	19.05	17.75	17.57
ψ (slo violations) [%]	58.05	58.05	42.96	7.91	0.13	0.06	50.38	50.63	46.89
#Adaptations	45	45	29	27	27	23	84	82	76
Avg. #VMs [VMs]	11.13	10.84	11.07	9.93	10.93	10.74	8.80	8.78	8.93
Avg. response time [s]	1.45	3.11	2.39	0.85	0.48	0.48	2.72	2.74	2.56