Methodological Principles for Reproducible Performance Evaluation in Cloud Computing

Alessandro V. Papadopoulos, *Senior Member, IEEE,* Laurens Versluis, *Member, IEEE,* André Bauer, Nikolas Herbst, *Member, IEEE,* Jóakim von Kistowski, *Member, IEEE,* Ahmed Ali-Eldin, Cristina L. Abad, *Member, IEEE,* José Nelson Amaral, *Senior Member, IEEE,* Petr Tůma, *Member, IEEE,* and Alexandru Iosup, *Member, IEEE*

Abstract—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.

Index Terms—Experimental evaluation, observation study, experimentation.

1 Introduction

Experimental methodology problems are common in many domains [1]. In computer science research they seem to appear also due to a lack of agreement on standard techniques for measuring, reporting, and interpreting performance [2], [3]. 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 [4] related to technological aspects such as dynamic environments, on-demand resources and services, diverse cost models, and new non-functional requirements such as elasticity [5] or elasticity-correlated metrics [6]. In this work, we focus on the principles and feasibility of experimental performance studies in cloud computing settings.

- *A.V. Papadopoulos is with the Mälardalen University, Västerås, Sweden. E-mail: alessandro.papadopoulos@mdh.se*
- *L. Versluis and A. Iosup are with Vrije Universiteit Amsterdam, The Netherlands.*
- *A. Bauer, N. Herbst and J. von Kistowski are with University of Würzburg, Germany.*
- *A. Ali-Eldin is with Umeå Univ., Sweden and UMass, Amherst, USA.*
- *C. L. Abad is with Escuela Superior Politecnica del Litoral, Ecuador.*
- *J. N. Amaral is with University of Alberta, Canada.*
- *P. Tůma is with Charles University, Czech Republic.*

Manuscript received Sept, 2018.

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 [1], [2]. 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 [7] and high-performance computing in the last few years [2], 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 [8]. 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 [9], [10] 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.

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) [11], [12], or in the evaluation of Java Virtual Machines (JVMs) and Just-in-Time compilers [13], [14], 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.

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

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 [15]. 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 [16]. 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 [17].

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 [18]) (ii) a (weighted) geometric mean for speedup ratios compared to a reference implementation [19], (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 [20].

2.2 Reproducibility

Although reproducibility of experiments is one of the pillars of the scientific method [8], 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 [21]. 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 the ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), the ACM Symposium on Operating Systems Principles (SOSP), and the International Conference on Very Large Data Bases (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 [22].

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 [23]. Handigol et al. [24] promote the use of containers for repeatable datacenter networking experiments; systems such as APT [25], EmuLab [26], and FlexLab [27] aim to provide environments for repeatable distributed networked systems research; and frameworks such as DataMill [28] offer some control over experimental variability. Among the more extreme examples is the work by Governor et

al. [29] 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. [30], 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. The ACM Symposium on Principles of Database Systems (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 [31]. In performance evaluation, the ACM/SPEC International Conference on Performance Engineering (ICPE) is encouraging authors to share research artifacts (both code and data) in a public repository maintained by the SPEC Research Group[5]. The conference of the ACM Special Interest Group on Data Communication (SIGCOMM) is running a workshop dedicated to reproducibility of networking research[6]. The conference of the ACM Special Interest Group on Programming Languages (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 the International Conference for High Performance Computing, Networking, Storage, and Analysis (SC) and the ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (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 [8]. 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

In this section, we propose a set of eight principles that represent a *minimal* set of guidelines for performance evaluation in the cloud computing domain. The main purpose of such principles is to provide guidelines for designing experiments and reporting results in a sound way, in order to promote reproducibility, and generality of the conclusions that can be drawn from the experiments. Such a set of principles must be considered as a core that could be extended and specified better for given application domains in cloud computing. The set of principles could also be seen as a reviewer's checklist for assessing the quality and reproducibility of an evaluation in a experimental cloud research paper.

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 [32]. Each experiment should be designed to answer a specific set of questions (see P3). The experiment design will be guided by factors such as the performance model used by the experimenter, but should also test and quantify how that model can differ from the real system due to, e.g., interference from uncontrolled variables (see P1 and P2).

P1: **Repeated experiments (statistical)**. 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 [33], [34], [35], [36]. 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: **Workload and configuration coverage**. Experiments should be conducted in **different (possibly randomized) configurations** of relevant parameters to cover a representative sample of the space of the controlled variables, such as, the workload, especially parameters that exhibit stochastic behaviour in real scenarios, and are thus not completely under control or those that may interact with the platform in unexpected ways, e.g., workloads can change from diurnal to bursty. 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 and configurations, e.g., the type of hardware configuration used, and the amount of resources allocated . However, some parameters that have significant effect on performance may not be under the (complete) control of the experimenter due to, e.g., their stochastic behaviour in real life. While both these types of parameters can be considered controller variables in the experiment, the stochastic nature of the system and workloads can yield significantly different results. One such example would be when the system is subject to momentarily overload due to, e.g., workload spikes, queuing effects occur which can highly impact the measured performance. . Similarly, although the experimenter may select the number of allocated machine instances, the aggregate computing power across these instances may fluctuate as typically seen in public cloud experiments, and with it the observed performance. In addition, 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 , i.e., perform experiments with many realistic configurations. The experimenter should thus start by carefully noting the controlled variables in the system that can impact the results, and then choose a random, but realistic, set of values for these variables. For example, an admission control algorithm that was designed assuming diurnal patterns of customer arrival rates should be tested with multiple different arrival patterns, including more challenging bursty arrival patterns. In addition, an admission control algorithm should be tested against multiple different assumptions on the amount of resources available and their granularity..

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 [37], [38], workload coverage is not considered. Randomization is rare, in part perhaps because systematic treatment of workload randomization is relatively recent [39].

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. In addition, the description should clearly state the objective 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*, i.e., if a customized configuration has been used, (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. (v) The objective of each experiment before the results are discussed.

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 objective of the experiments, and the configuration information are important to determine the number of experiments that is needed to ensure 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.

According to the definition provided by ACM[9], "by *artifact* we mean a digital object that was either created by the authors to be used as part of the study or generated by the experiment itself. For example, artifacts can be software systems, scripts used to run experiments, input datasets, raw data collected in the experiment, or scripts used to analyze results." 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.[10]

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.

9. https://www.acm.org/publications/policies/artifact-review-badging

10. Several other sciences have taken steps toward this, including medical sciences [40] and economics [41].

P5: **Probabilistic result description of measured performance**. Report a 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, aggregations must be suitable to the distribution of the data points that are aggregated. 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. It is incumbent on researchers to carefully examine the distribution of data points and select appropriate summarization methods. For instance, for complex distributions use tools such as box plots, violin plots, and empirical distribution function plots.

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 not in compliance with discussed guidelines [18], [19].

P6: **Statistical evaluation**. When making conclusions from experimental data, provide a statistical evaluation of the significance of the obtained results.

The results of an experimental evaluation are often not just reported, but also used to derive conclusions, such as comparing an artifact against competing approaches. In these cases, conclusions are made based on an evaluation that considered a (necessarily) limited number of scenarios and observations. The likelihood and representativeness of these scenarios becomes a factor in the (statistical) significance of the experimental conclusions, and is therefore essential in establishing the validity of the claims.

When the conclusions involve competing approaches, the information available on the competing approaches may not suffice for a robust 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 must therefore be chosen to reflect the experimental circumstances, the properties of the data that is collected, and the conclusions that are made. The choice of methods is not necessarily portable between studies – for example, although Student's or Welch's t-test are often viable options for data with

a normal distribution, non parametric methods [32] may be needed in other cases. Rather than recommending a single evaluation procedure, we therefore provide example references that may help the reader build an appreciation for the pitfalls of the cloud computing experiments and select appropriate statistical methods depending on particular circumstances. For general advice, the classic work of Huff [42] is well complemented by a more recent work of Reinhart [43]. Exploratory data analysis [44] can be combined with bootstrap and similar sampling methods [45] for increased robustness in non parametric evaluation. Examples of practical experience illustrate the need to carefully construct experiments [3], [46], evaluate the accuracy of employed tools [47], [48], avoid reporting statistics that mask variability [2], [49], or reflect the variability of the experimental environment in the reported conclusions [50]. We also find it useful to adopt a self-reflective view of the experimental evaluation as described in the article by Blackburn et al. [51].

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[11] and LDBC[12] 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 of 2017[13]), 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 [52]. These perturbations increase if the measurements are frequent or if the instrumentation is particularly intrusive [53]. 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 [53], [54]. 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.

11. 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
12. In its by-laws, v.1.0, published in 2017.
13. 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 The Use of Principles 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 the SPEC Cloud IaaS 2018 benchmark [55].

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" [56]. 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, the SPEC Cloud IaaS benchmark [55] uses five separate re-runs for its baseline experiment. Some benchmarks also allow a varying number of runs, for example, the SPEC CPU 2017 benchmark [57] allows either two or three runs.

P2: Many standard industry benchmarks use multiple workloads that are run independently. The SPEC Cloud IaaS 2018 benchmark runs a transaction workload using Apache Cassandra, and a K-Means MapReduce workload. The SPEC CPU benchmark is based on nearly twenty integer and floating point programs. For research purposes, additional workloads have been created for the SPEC CPU 2017 suite [58]. 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 the SPECpower_ssj2008 benchmark [59].

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. The SPEC Cloud IaaS 2018 benchmark, in particular, integrates a benchmarking harness *cbtool* as an Apache-licensed open-source artifact.

P5: Standard benchmarks report average performance values, with some exceptions. The SPEC CPU 2017 benchmark reports the median value of three runs, or the minimum (worse) value if only two runs were executed. The SPEC Cloud IaaS 2018 benchmark reports the average of the 99th percentile measured for latency. The SPEC SERT 2 suite reports the coefficient of variation (CV) for performance.

P6: is usually out of scope for released standard benchmarks. The variance of result scores is usually checked thoroughly prior to a release by a committee. Evaluation of multiple results consists simply of comparing the final metric score. Notably, the SPEC SERT 2 suite 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 the SPEC Cloud IaaS 2018 benchmark.

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. The SPEC Cloud IaaS 2018 benchmark counts application instances as its cost component, and benchmarks focused on power consumption report Watts. Some benchmark reports (e.g., TPC-C) include a price-per-performance metric. The mentioned benchmarks rate individual cloud services. In the case, a cloud benchmark assesses a composition of cloud services, a cost breakdown per component would be part of a transparent result report.

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 2 suite—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 [60].

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 [61] adopting its measurement methodology and metrics for auto-scaler evaluations[14]. Other measurement and load testing frameworks could replace our choice here. 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 SPEC Research Group [5]. The *under/(over)-provisioning accuracy*

14. BUNGEE Elasticity Measurement Framework: https://descartes.tools/bungee

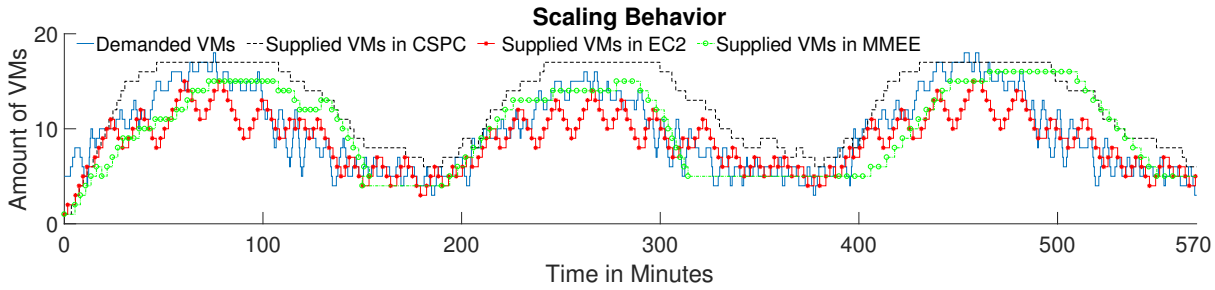


Fig. 1. The number of VMs allocated by the Reactive auto-scaler, in the Private, EC2, and MMEE IaaS Cloud experiments.

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 [20], [62]. 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 [61].

The implemented auto-scaler [63] and experiment data are online available[15]. We use the authentic, time-varying trace of the FIFA championship 1998.[16](→ **Principle 4 (open access artifact) 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[17]-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 (workload and configuration coverage).**) 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 (repeated experiments).**)

In the CSPC scenario, the application is deployed in a private Apache CloudStack[18] 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 each 2 × 16GB RAM (DIMM DDR4 RAM operated @ 1866 MHz) deployed. The specification of each VM in all setups is listed in Table 1. For all scenarios, Tomcat 7 is the application server. As the

15. Auto-scaler and experiment data: https://doi.org/10.5281/zenodo.1169900

16. FIFA Source: http://ita.ee.lbl.gov/html/contrib/WorldCup.html

17. OpenNebula: https://opennebula.org/

18. Apache CloudStack: https://cloudstack.apache.org/

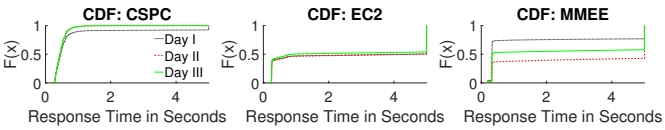


Fig. 2. Distribution of response times, per experiment.

LU worklet of the SERT 2 suite 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 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`[19] command and averaged across all VMs currently running. (→ **This experimental description fulfills Principle 3 (experimental setup description).**)

Figure 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 2 depicts the distributions of the response times per day and Figure 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 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 2 shows the average metrics and their standard deviation. Furthermore, Table 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 (probabilistic result description of measured performance), 7 (measurement units), and 8 (Cost).**)

19. `top` command manual: http://man7.org/linux/man-pages/man1/top.1.html

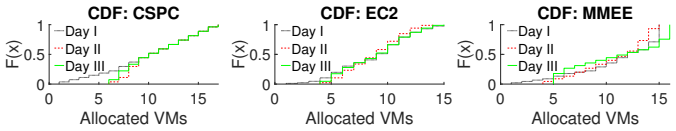


Fig. 3. Distribution of allocated VMs, per experiment.

TABLE 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

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 4 shows the proportion of variance and p-value ($Pr(> F)$) 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 (statistical evaluation).**)

This example has the main purpose of illustrating how the presented principles can be used in a practical case, and that the adoption of the proposed principles can improve the presentation of the results without significantly affecting the length of the paper.

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 [9], covering papers published in 16 top-level international conferences and journals, between 2012 and 2017. In particular, the selected conferences and journals are: IEEE International Conference on Cloud Computing (IEEECloud), IEEE/ACM International Conference on Utility and Cloud Computing (UCC), IEEE/ACM International Symposium in Cluster, Cloud, and Grid Computing (CCGrid), IEEE Transactions on Parallel and Distributed Systems (TPDS), IEEE International Conference on Cloud Engineering (IC2E), IEEE International Conference on Cloud Computing Technology and Science (CloudCom), IEEE International Conference on Autonomic Computing (ICAC), ACM/SPEC International Conference on Performance Engineering (ICPE), IEEE Transactions on Cloud Computing (TCC), ACM Symposium on Cloud Computing (SoCC), ACM Symposium on High-Performance Parallel and Distributed Computing (HPDC), ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS), Future Generation Computer Systems (FGCS), European Conference on Computer Systems (EuroSys), International Conference for High Performance Computing, Networking, Storage, and Analysis (SC), USENIX Symposium on Networked Systems Design and Implementation (NSDI).

TABLE 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)

TABLE 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

4.1 Systematic Literature Review

The systematic literature review [9] is a structured method to provide an overview of a research area. In this work, we follow the guidelines discussed by Peterson et al. [10], [64]. Figure 4 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. We did not explicitly include 'performance' as a keyword, since (1) if it is introduced with the AND operator, it narrows down the scope of the query to a subset of the analyzed papers, and (2) if it is introduced with the OR operator, the results of the query would include papers that deal with performance engineering, but not necessarily related to the cloud domain. The following string is

TABLE 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

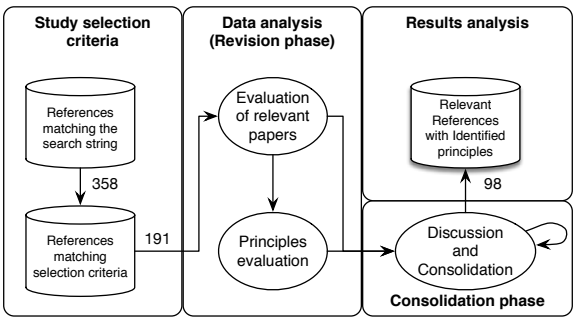


Fig. 4. Workflow for the systematic literature review.

TABLE 5
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

therefore 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 [65] 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[20], 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 5 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:

- **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.

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 [66], [67] and the weighted Cohen's Kappa analysis [68] 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 [69], $\kappa < 0$ 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.

The weighted Cohen's Kappa computes a value κ_w, that has an analogous interpretation to κ, with the difference that its computation accounts for differences in the type of disagreement with more than one category (in our case Y,

20. https://www.semanticscholar.org/

TABLE 6
Results of the 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
κ_w	0.98	0.91	0.83	0.88	0.87	0.86	0.61	0.97	0.85

P, and N). For example, if the two reviewers disagreed with a Y/N evaluation, its weight is higher than a disagreement with Y/P or P/N.

Finally, we also evaluate the percentage of agreement (**%A**), computed as the number of agreed values over the total number of decisions.

Table 6 quantifies the reviewer agreement in terms of percentage of agreement (**%A**), and in terms of the κ and κ_w statistics. Reviewers reached an almost perfect agreement ($\kappa_w \geq \kappa > 0.8$) for P1–5 and for P7-8. For the remaining P6, "Statistical evaluation", the κ value of 0.59, and the κ_w value of 0.61 indicate moderate agreement. However, the percentage of agreement for P6 is high, at 94.9%. For P6, the low value of κ and κ_w 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 and κ_w values, 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 of κ_w, 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.

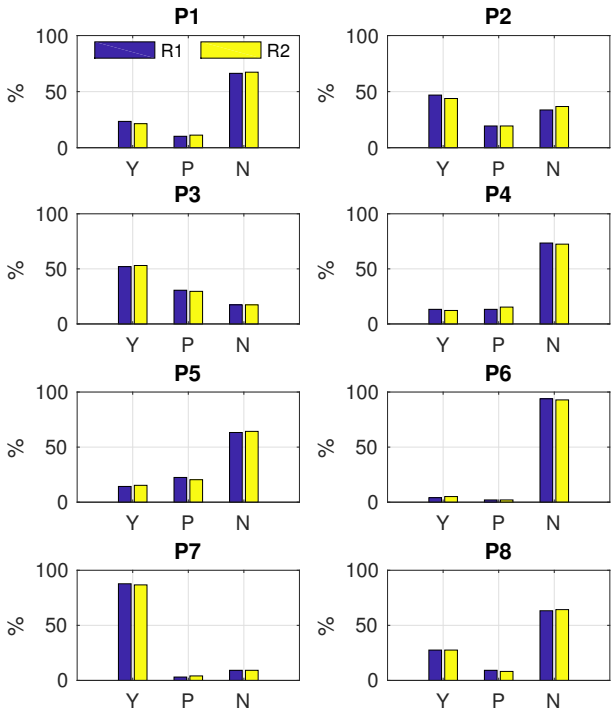


Fig. 5. Evaluation of P1–8, one sub-plot per principles. Columns R1 and R2 summarize results by 1st/2nd reviewer.

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 5 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 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 5, sub-plot P1), and only 21% do both. Such information is inferred by the experimental setup description provided in the papers, and by the provided experimental results. 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 [50].

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 of the assessed technique is not publicly released and the datasets used for the evaluation 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 $\texttt{N} > 90\%$. 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.

P7 refers to the presence of the units of measurement throughout the paper. We find P7 is the principle with the highest fulfillment, with $\texttt{Y} > 85\%$. 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

Dealing with uncertainty in experimental evaluation is a common issue in many scientific disciplines. NIST and ISO have standards and guidelines to evaluate, express, and report the uncertainty in measurements [70], [71]. 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. [72] and Iosup et al. [4] 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. [73] 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 [74] 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 [3], [75], environments such as DataMill [46] can randomize selected environmental conditions and thus improve the ability to generalize from particular measurements. Also, works such as [13], [14] 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 [2] 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 [76] 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. [77] focus on domain-specific issues, and propose principles of metrics and workload characteristics to use in grid-computing experiments. Mytkowitz et al. [3] and Zhang et al. [78] identify technology counterparts to our principles: sources of measurement bias in experimental work corresponding to large-scale systems.

For networked systems, Krishnamurthy et al. [79] pro-

vide 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 researchers that decrease the value of their work [80]. 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 [51].

Software engineering community: One of the fields of computing science to first use systematic literature reviews is software engineering [81]. The decade-long longitudinal study of Sjøberg et al. [7] 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. [82] 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 [83]. Collberg et al. [22] 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 paper 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 paper 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.

Acknowledgments

This work was also supported by Swedish Foundation for Strategic Research under the project "Future factories in the cloud (FiC)" with grant number GMT14-0032, by the Knowledge Foundation (KKS), by the German Research Foundation (DFG) under grant No. KO 3445/11-1, by the US NSF grant No. 1836752, by the Wallnberg Foundation, by the Swedish strategic research programme eSSENCE, by the Google Faculty Research Award, by the National Science and Engineering Research Council of Canada (NSERC), by the ECSEL Joint Undertaking (JU) grant No 783162, by the Dutch NWO Vidi grant MagnaData, and by generous donations by Oracle and Intel Labs, both USA, and by Solvinity, the Netherlands.

References

[1] G. K. Sandve, A. Nekrutenko, J. Taylor, and E. Hovig, "Ten simple rules for reproducible computational research," *PLOS Computational Biology*, vol. 9, pp. 1–4, 2013.

[2] 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.

[3] T. Mytkowicz, A. Diwan, M. Hauswirth, and P. F. Sweeney, "Producing wrong data without doing anything obviously wrong!" *SIGPLAN Not.*, vol. 44, pp. 265–276, 2009.

[4] 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.

[5] N. Herbst *et al.*, "Ready for rain? A view from SPEC research on the future of cloud metrics," *CoRR*, vol. abs/1604.03470, 2016.

[6] 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.

[7] D. I. K. Sjoberg *et al.*, "A survey of controlled experiments in software engineering," *IEEE Trans. Softw. Eng.*, vol. 31, pp. 733–753, 2005.

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

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

[10] 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.

[11] M. Sayeed *et al.*, "Measuring high-performance computing with real applications," *Computing in Science Engineering*, vol. 10, pp. 60–70, 2008.

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

[13] A. Georges, D. Buytaert, and L. Eeckhout, "Statistically rigorous java performance evaluation," *SIGPLAN Not.*, vol. 42, pp. 57–76, 2007.

[14] 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.

[15] S. S. Stevens, "Measurement, psychophysics, and utility," in *Measurement: Definitions and Theories*, C. Churchman and P. Ratoosh, Eds. Wiley, 1959, pp. 18–63.

[16] D. Lilja, *Measuring Computer Performance: A Practitioner's Guide*. Cambridge University Press, 2008.

[17] S. S. Stevens, "On the Theory of Scales of Measurement," *Science*, vol. 103, pp. 677–680, Jun 1946.

[18] P. J. Fleming and J. J. Wallace, "How not to lie with statistics: The correct way to summarize benchmark results," *Commun. ACM*, vol. 29, pp. 218–221, 1986.

[19] J. R. Mashey, "War of the benchmark means: Time for a truce," *SIGARCH Comput. Archit. News*, vol. 32, pp. 1–14, 2004.

[20] N. Herbst *et al.*, "Quantifying Cloud Performance and Dependability: Taxonomy, Metric Design, and Emerging Challenges," *ACM ToMPECS*, vol. 3, pp. 19:1–19:36, August 2018. [Online]: http://doi.acm.org/10.1145/3236332

[21] M. Baker, "Is there a reproducibility crisis?" *Nature*, vol. 533, pp. 452–454, 2016.

[22] C. Collberg and T. A. Proebsting, "Repeatability in computer systems research," *Commun. ACM*, vol. 59, pp. 62–69, 2016.

[23] 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, pp. 17–24, 2006.

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

[25] R. Ricci *et al.*, "Apt: A platform for repeatable research in computer science," *SIGOPS Oper. Syst. Rev.*, vol. 49, pp. 100–107, 2015.

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

[27] R. Ricci *et al.*, "The Flexlab approach to realistic evaluation of networked systems," in *NSDI*, 2007, pp. 1–14.

[28] 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, pp. 1411–1440, 2016.

[29] M. P. Grosvenor *et al.*, "Queues don't matter when you can jump them!" in *NSDI*, 2015, pp. 1–14.

[30] J. Cavazos *et al.*, "Rapidly selecting good compiler optimizations using performance counters," in *CGO*, 2007, pp. 185–197.

[31] R. F. Boisvert, "Incentivizing reproducibility," *Commun. ACM*, vol. 59, pp. 5–5, 2016.

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

[33] J. Schad, J. Dittrich, and J.-A. Quiané-Ruiz, "Runtime measurements in the cloud: Observing, analyzing, and reducing variance," *Proc. VLDB Endow.*, vol. 3, pp. 460–471, 2010.

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

[35] 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, pp. 15:1–15:23, 2016.

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

[37] K. Hoste and L. Eeckhout, "Microarchitecture-independent workload characterization," *IEEE Micro*, vol. 27, pp. 63–72, 2007.

[38] A. Joshi, A. Phansalkar, L. Eeckhout, and L. K. John, "Measuring benchmark similarity using inherent program characteristics," *IEEE Transactions on Computers*, vol. 55, pp. 769–782, 2006.

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

[40] D. A. W. Soergela, "Rampant software errors may undermine scientific results," *F1000Research*, vol. 3, pp. 1–13, 2015.

[41] 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, pp. 257–279, 2014.

[42] D. Huff, *How to Lie with Statistics*, reissue edition ed. New York: W. W. Norton & Company, Oct. 1993.

[43] A. Reinhart, *Statistics Done Wrong: The Woefully Complete Guide*, 1st ed. San Francisco: No Starch Press, Mar. 2015.

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

[45] T. C. Hesterberg, "What teachers should know about the bootstrap: Resampling in the undergraduate statistics curriculum," *The American Statistician*, vol. 69, pp. 371–386, 2015.

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

[47] T. Mytkowicz, A. Diwan, M. Hauswirth, and P. F. Sweeney, "Evaluating the Accuracy of Java Profilers," in *Proceedings of the 31st ACM SIGPLAN Conference on Programming Language Design and Implementation*, ser. PLDI '10, 2010, pp. 187–197.

[48] A. Shipilev, "Nanotrusting the Nanotime," 2014. [Online]: https://shipilev.net/blog/2014/nanotrusting-nanotime/

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

[50] 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, pp. 15:1–15:31, 2016.

[51] S. M. Blackburn *et al.*, "The truth, the whole truth, and nothing but the truth: A pragmatic guide to assessing empirical evaluations," *ACM Trans. Program. Lang. Syst.*, vol. 38, pp. 15:1–15:20, 2016.

[52] 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.

[53] A. D. Malony and D. A. Reed, "Models for performance perturbation analysis," *SIGPLAN Not.*, vol. 26, pp. 15–25, 1991.

[54] A. D. Malony, D. A. Reed, and H. A. G. Wijshoff, "Performance measurement intrusion and perturbation analysis," *TPDS*, vol. 3, pp. 433–450, 1992.

[55] S. P. E. Corporation, "SPEC cloud IaaS 2018 benchmark design overview," Tech. Rep., 2018.

[56] J. von Kistowski *et al.*, "How to build a benchmark," in *ICPE*, 2015, pp. 333–336.

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

[58] J. N. Amaral *et al.*, "The Alberta workloads for the SPEC CPU 2017 benchmark suite," in *ISPASS*, 2018.

[59] K.-D. Lange, "Identifying shades of green: The SPECpower benchmarks," *Computer*, vol. 42, pp. 95–97, 2009.

[60] 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, pp. 54–63, 2016.

[61] 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.

[62] A. Ilyushkin *et al.*, "An experimental performance evaluation of autoscalers for complex workflows," *ToMPECS*, vol. 3, pp. 8:1–8:32, 2018.

[63] A. Bauer, N. Herbst, S. Spinner, A. Ali-Eldin, and S. Kounev, "Chameleon: A Hybrid, Proactive Auto-Scaling Mechanism on a Level-Playing Field," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, pp. 800 – 813, April 2019.

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

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

[66] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and Psychological Measurement*, vol. 20, 1960.

[67] J. L. Fleiss, "Measuring nominal scale agreement among many raters," *Psychological Bulletin*, vol. 76, 1971.

[68] J. Cohen, "Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit," *Psychological bulletin*, vol. 70, p. 213, 1968.

[69] J. R. Landis and G. G. Koch, "The measurement of observer agreement for categorical data," *Biometrics*, vol. 33, pp. 159–174, 1977.

[70] B. N. Taylor and C. E. Kuyatt, "Guidelines for evaluating and expressing the uncertainty of NIST measurement results," National Institute of Standards and Technology (NIST), Tech. Rep. 1297, 1994.

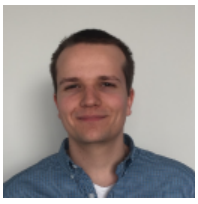
[71] ISO, "Evaluation of measurement data – guide to the expression of uncertainty in measurement," Joint Committee for Guides in Metrology (JCGM/WG 1), Tech. Rep. JCGM 100:2008, 2008.

[72] E. Folkerts *et al.*, "Benchmarking in the cloud: What it should, can, and cannot be," in *Selected Topics in Performance Evaluation and Benchmarking*, 2013, pp. 173–188.

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

[74] G. Fursin, A. Lokhmotov, and E. Plowman, "Collective knowledge: Towards R&D sustainability," in *DATE*, 2016, pp. 864–869.
[75] T. Kalibera and P. Tůma, "Precise regression benchmarking with random effects: Improving mono benchmark results," in *EPEW*, 2006, pp. 63–77.
[76] E. Frachtenberg and D. G. Feitelson, "Pitfalls in parallel job scheduling evaluation," in *JSSPP*, 2005, pp. 257–282.
[77] A. Iosup *et al.*, "On grid performance evaluation using synthetic workloads," in *JSSPP*, 2006, pp. 232–255.
[78] 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.
[79] B. Krishnamurthy, W. Willinger, P. Gill, and M. Arlitt, "A socratic method for validation of measurement-based networking research," *Comput. Commun.*, vol. 34, pp. 43–53, 2011.
[80] J. Vitek and T. Kalibera, "R3: Repeatability, reproducibility and rigor," *SIGPLAN Not.*, vol. 47, pp. 30–36, 2012.
[81] B. Kitchenham *et al.*, "Systematic literature reviews in software engineering - a systematic literature review," *Inf. Softw. Technol.*, vol. 51, pp. 7–15, 2009.
[82] 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.
[83] V. Pieterse and D. Flater, "The ghost in the machine: don't let it haunt your software performance measurements," Tech. Rep. 1830, 2014.

Alessandro Vittorio Papadopoulos is an associate professor in computer science at the Mälardalen University, Västerås, Sweden. He received his B.Sc. and M.Sc. (summa cum laude) degrees in computer engineering from the Politecnico di Milano, Milan, Italy, and his Ph.D. (Hons.) degree in information technology from the Politecnico di Milano, in 2013. His research interests include cloud computing, real-time systems, autonomic computing and control theory.



Laurens Versluis received his B.Sc. and M.Sc degrees in computer science (Distributed Systems, Software Technology) from the Technical University of Delft, The Netherlands. Currently, he is a Ph.D. student with the Massiving Computer Systems Group of the Department of Computer Science, Faculty of Sciences, VU Amsterdam, The Netherlands. His research interests include cloud computing, distributed systems, scheduling, complex workflows, image processing, and privacy enhancing technologies.

André Bauer is PhD student at the chair of software engineering at the University of Würzburg. He serves as elected newsletter editor of the SPEC Research Group. His research topics include elasticity in cloud computing, auto-scaling and resource management, autonomic and self-aware computing, and forecasting.

Nikolas Herbst is a research group leader at the chair of software engineering at the University of Würzburg. He received a PhD from the University of Würzburg in 2018 and serves as elected vice-chair of the SPEC Research Cloud Group. His research topics include elasticity in cloud computing, auto-scaling and resource management, performance evaluation of virtualized environments, autonomic and self-aware computing, as well as time series analysis and machine learning.

Jóakim von Kistowski is a researcher at the chair of software engineering at the University of Würzburg, Germany. He serves as the Chair of the SPEC Research Power Group and as a contributor to the SPEC OSG Power Committee. His research is focused on measurement, analysis, and modelling of energy efficiency of computing systems, as well as computing system performance modelling and management.

Ahmed Ali-Eldin has a joint position as a Researcher at Umeå university, Sweden, and as a Postdoc at UMass, Amherst. In addition, he works part time at Elastisys AB, Sweden as a senior data scientist. He obtained his PhD from Umeå University in 2015. His research interests lie in the intersection of Computer systems, and performance modeling.

Cristina L. Abad is a professor at Escuela Superior Politecnica del Litoral, ESPOL, in Guayaquil–Ecuador. She obtained her MS and PhD degrees from the Computer Science department of the University of Illinois at Urbana-Champaign, where she was a recipient of the Computer Science Excellence Fellowship and a Fulbright Scholarship. From 2011 through 2014, she was a member of the Hadoop Core team at Yahoo, Inc. Cristina has been the recipient of two Google Faculty Research Awards (2015 and 2017). Her main research interests lie in the area of distributed systems.

José Nelson Amaral received the PhD degree in electrical and computer engineering from the University of Texas at Austin, in 1994. He is a professor of computing science at the University of Alberta, Canada. His current research interests include compiler design, static analysis, feedback directed compilation, computer architecture, high performance computer systems, transactional memory and the application of learning methods to the design of compilers.

Petr Tůma is a professor with the Department of Distributed and Dependable Systems of Charles University, Czech Republic. He has received his PhD from Charles University in 1998 and MSc from the Czech Technical Institute in 1994. His long term research interests include performance evaluation of computer systems and integration of performance awareness into the software development process, together with the development of system tools and measurement methods for this purpose.

Alexandru Iosup is tenured full Professor and University Research Chair with the Vrije Universiteit Amsterdam, the Netherlands. He is also Chair of the SPEC Research Cloud Group. He received a Ph.D. from TU Delft, the Netherlands (2009) and an M.Sc. from Politehnica University of Bucharest, Romania (2004), both in computer science. He was awarded the yearly Netherlands Prize for Research in Computer Science (2016), the yearly Netherlands Teacher of the Year (2015), several SPECtacular awards (2012-2017), and numerous research awards and nominations. His research interests are in massivizing computer systems, that is, making computer systems combine desirable properties such as elasticity, performance, and availability, yet maintain their ability to operate efficiently in controlled ecosystems. Topics include cloud computing and big data, with applications in big science, big business, online gaming, and (upcoming) massivized education. His work is funded by a combination of prestigious personal grants, generous industry gifts and collaborations, and EU and national projects.