Same, Same, but Dissimilar: Exploring Measurements for Workload Time-series Similarity

Mark Leznik
Ulm University
Ulm, Germany
mark.leznik@uni-ulm.de

Johannes Grohmann
University of Würzburg
Würzburg, Germany
johannes.grohmann@wuerzburg.de

Nina Kliche
Ulm University
Ulm, Germany
nina.kliche@uni-ulm.de

André Bauer
University of Würzburg
Würzburg, Germany
andre.bauer@wuerzburg.de

Daniel Seybold
Ulm University
Ulm, Germany
daniel.seybold@uni-ulm.de

Simon Eismann
University of Würzburg
Würzburg, Germany
simon.eismann@uni-wuerzburg.de

Samuel Kounev
University of Würzburg
Würzburg, Germany
samuel.kounev@uni-wuerzburg.de

Jörg Domaschka
Ulm University
Ulm, Germany
joerg.domaschka@uni-ulm.de

ABSTRACT

Benchmarking is a core element in the toolbox of most systems researchers and is used for analyzing, comparing, and validating complex systems. In the quest for reliable benchmark results, a consensus has formed that a significant experiment must be based on multiple runs. To interpret these runs, mean and standard deviation are often used.

In case of experiments where each run produces a time series, applying and comparing the mean is not easily applicable and not necessarily statistically sound. Such an approach ignores the possibility of significant differences between runs with a similar average. In order to verify this hypothesis, we conducted a survey of 1,112 publications of selected performance engineering and systems conferences canvassing open data sets from performance experiments. The identified 3 data sets purely rely on average and standard deviation. Therefore, we propose a novel analysis approach based on similarity analysis to enhance the reliability of performance evaluations. Our approach evaluates 12 (dis-)similarity measures with respect to their applicability in analysing performance measurements and identifies four suitable similarity measures. We validate our approach by demonstrating the increase in reliability for the data sets found in the survey.

CCS CONCEPTS

• **Computing methodologies** → *Modeling methodologies*; • **Information systems** → **Similarity measures**; • **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

ICPE '22, April 9–13, 2022, Bejing, China

ACM ISBN 978-1-4503-9143-6/22/04.
https://doi.org/10.1145/3489525.3511699

KEYWORDS

data sets, workload analysis, time series, time series similarity, distance metrics

ACM Reference Format:
Mark Leznik, Johannes Grohmann, Nina Kliche, André Bauer, Daniel Seybold, Simon Eismann, Samuel Kounev, and Jörg Domaschka. 2022. Same, Same, but Dissimilar: Exploring Measurements for Workload Time-series Similarity. In *Proceedings of the 2022 ACM/SPEC International Conference on Performance Engineering (ICPE '22), April 9–13, 2022, Bejing, China.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3489525.3511699

1 INTRODUCTION

Benchmarking is a core element in the toolbox of most systems researchers and is used for analyzing, comparing, and validating complex systems. In the quest for reliable benchmark results, a consensus has formed that a significant experiment must be based on multiple runs [20, 27]. To interpret the different runs various approaches exist, but often mean and standard deviation are used to present the outcome of an experiment. In case of experiments where each run produces a time series applying and comparing the mean is not easily applicable, besides being not necessarily statistically sound.

One challenge of such experiments is to ensure that individual runs do not differ too much. Yet, it may happen that the system under test presents with multiple types of behavior. This is to be expected in volatile environments such as cloud resources [1, 21], but also applies to other complex systems such as distributed database management systems [9, 32, 33]. Another possibility could also be an unnoticed system component crash during a run.

Here, it is mandatory to identify the different behavior groups in the gathered data, because: (i) If all runs are compared without this filter, the analysis of the entire experiment is potentially misleading; (ii) Identified groups can indicate problems, non-determinism, disturbances, or uncontrollable events in the experiment set-up; (iii) Outlier measurements can be used to investigate anomalies;

(iv) Reporting the results of each group increases the experiment validity as well as data quality.

However, a method to automatically analyze benchmarking results regarding their time series similarity or dissimilarity is missing. Moreover, any applied solution should avoid predefined thresholds since these are application-specific and are usually calibrated by domain knowledge experts. To address this situation, we seek to answer the following three research questions:

- **RQ.1:** Are open data sets available to validate our approach to detect measurement similarity or dissimilarity?
- **RQ.2:** Given the large number of available similarity and dissimilarity measures, how comparable are their results and is there a meaningful subset of these metrics to be used by practitioners?
- **RQ.3:** Can the use of a subset of similarity and dissimilarity measures help identify outliers and problems in the data sets and if so, does their use provide an improvement over the mean and standard deviation?

Answering these research questions, we analyse publicly available open data sets from earlier top-level cloud and systems conferences in order to investigate if the sketched problems are actually observed in practice. Using visual analysis as well as statistical methods we (i) evaluate various measures to quantify the similarity or dissimilarity of time series data from the same experiment; (ii) evaluate the correlation between the similarity and dissimilarity measures and from that derive a list of four such measures; (iii) introduce a novel approach for these measures to identify outliers in the data sets; and (iv) evaluate the capability of the measures to identify outliers in comparison to mean and standard deviation.

The remainder of this paper is structured as follows: Section 2 gives an overview of our approach. Section 3 discusses the collection of open data sets, Section 4 discusses the selection of (dis-)similarity metrics, and Section 5 outlines our approach to determine (dis-)similarity in repeated measurements. Further, Section 6 showcases the results of the proposed approach when applied to the open data sets. Finally, Section 7 discusses related work, Section 8 covers threats to validity, and Section 9 concludes the paper with a call to action to the community.

2 OVERVIEW

In order to answer our research questions, we proceed as follows: In an initial step, we analyse 1,000+ conference papers to identify openly available data sets from the cloud computing and performance engineering domain. This step is detailed in Section 3. In parallel, we analyse existing measures for similarity and dissimilarity which are applicable to time series data (cf. Section 4). This is followed by applying the similarity measures to the performance metrics found in the data sets. Due to the large number of similarity measures, we then identify measures with group-wise similar behaviour and exclude them from the set of used similarity measures. In the following step, we apply the remaining similarity measures to the performance metrics in the data set and identify whether forming subgroups of runs increases the similarity within them. Section 5 elaborates this procedure and details how this allows us to detect conspicuous time series measurements. In a final step, we evaluate whether the found time series are indeed different from other series from the same experiment.

Table 1: List of analyzed conferences

Conferences	papers	Conferences	papers
UCC 2018	27	CLOUD 2018	34
UCC 2019	28	CLOUD 2019	84
UCC 2020	28	CLOUD 2020	83
ICPE 2019	33	SOCC 2018	70
ICPE 2020	29	SOCC 2019	54
ICPE 2021	27	SOCC 2020	35
MASCOTS 2018	34	SIGMOD 2018	107
MASCOTS 2019	45	SIGMOD 2019	104
MASCOTS 2020	34	SIGMOD 2020	145
		Total	1,112

3 OPENDATA AND DATA SETS

For the sake of clarity, we introduce the terms *Time Series*, *Experiment*, and *Run*. They are used in the remainder of the text to describe the data sets.

Definition 1 (Time Series). *A time series $X = \{x_1, \ldots, x_n\}$ is a vector of n real-valued observations ordered in time. We do not impose equidistance in time for the observations.*

Definition 2 (Experiment). *An experiment defines a repeatable procedure in order to create data as well as a framework for executing the procedure and analysing the resulting data set. As such, the experiment defines the used software and hardware components, specifies timing, parameters, and repetitions.*

A data set may contain the outcome of multiple experiments. Usually, these experiments are performed in the same domain, but vary with respect to their execution parameters.

Definition 3 (Run). *A run is a unique execution of the experiment's procedure. In our scope, its outcome is one time series for each performance metric under observation.*

For addressing our research questions, we ideally require access to a large collection of data sets from different performance-oriented domains. Here, our approach is to conduct a study of published research papers from recent years. Due to the open data policy gaining momentum, we expect to find more than enough data sets for our purposes.

3.1 Methodology

For discovering the data sets for our study, we systematically review all publications from the last three years of selected top-level systems and cloud conferences. We decided to include the conferences shown in Table 1.

Table 1 shows a list of conference iterations from 2018 to 2021 together with the respective number of papers that we analysed. Overall, these conferences were chosen as they give a broad overview over the cloud and performance community, with over 1,000 published papers. The table gives us our first and only inclusion criteria:

- **IC1**: The paper was published in any of the three last iterations of our defined conference list.

Next, our study concentrates mainly on exclusion criteria for filtering the relevant data sets. We define the following exclusion criteria:

- **EC1**: The paper does not publish a data set.
- **EC2**: The published data set is not concerned with performance measurements.
- **EC3**: The measurements do not contain real performance measurements, but are simulated etc.
- **EC4**: The measurements contain only aggregated values, instead of entire measurement time series.
- **EC5**: The performance measurements contains less than 10 measurement repetitions.

We reviewed all published papers according to the given criteria and excluded papers matching any of the given exclusion criteria.

3.2 Data sets

In contrast to our expectation, we only identified **3** data sets, denoted as DS.1–DS.3, qualified for further analysis.

During our literature review, we noticed that many researchers would not execute multiple runs when dealing with time series, but instead argue that the time series itself is already a repeated measurement, and apply statistical analysis to their single time series. It should be noted, that our elimination criteria might be considered tough, however, we do consider repeated measurements a cornerstone of performance engineering.

DS.1: In [8], the authors conduct measurements on the variability of performance measurement of microservice applications in the cloud. The experiments consist of ten runs of 21 different configurations of the TeaStore [38] application on the Google Kubernetes Engine. Each experiment was 15 minutes long, with an average of 40 service instances deployed per experiment.

DS.2: DS.2 [30] is conducted by the Mowgli framework [31] and applied in [14] to model and predict the performance of cloud-hosted distributed Database Management Systems (DBMSs). The experiments consist of ten runs of 102 different cloud-hosted DBMS configurations. The measurements are conducted in a private OpenStack cloud and comprise three VM flavours, two NoSQL DBMSs and eleven DBMS runtime configurations such as cluster size, replication factor and consistency settings. All measurements are based on the Yahoo Cloud Serving Benchmark (YCSB) [5] with a write-heavy workload configuration.

DS.3 This data set was originally published together with [28]. The authors evaluate the stability of cloud application performance using micro-benchmarks. For that purpose, they repeatedly start different types of virtual machines on various regions of Amazon Web Services. On each virtual machine, they run multiple iterations of a series of micro-benchmarks. The results of each benchmark iteration on each virtual machine are published in a csv file on GitHub[1]. For the sake of this paper, we consider all iterations of a micro-benchmark on one virtual machine as a single experiment. Hence, each virtual machine maps to a set of very short time series, one for each performance metric investigated. In addition, we consider the time series (of a performance metric) from different virtual machines of the same type and regions, e.g., `m3.medium` in `eu-west-1`, as runs of one experiment.

[1] https://github.com/joe4dev/cwb-analysis/blob/9da7a83d4f4cec7f39c0188c9138ff7b41adf5db/data_raw/cwb-data-raw.csv

4 MEASURES OF (DIS-)SIMILARITY

Dissimilarities and similarities are widely used in many fields of science. Although the concept of a distance is mathematically well defined, there exists no appropriate concept for similarity [4] let alone similarity for time series. Consequently, we have to elaborate the term "similiarity" in our case to assess how dissimilar or similar two time series are. First, we define the dissimilarity as:

Definition 4 (Dissimilarity). *Let X be a set. A function $d : X \times X \to \mathbb{R}$ is called a dissimilarity, if d is non-negative, symmetric and reflexive, i.e. if for all $x, y \in X$ holds $d(x, y) \geq 0$, $d(x, y) = d(y, x)$ and $d(x, x) = 0$.*

Analogously, we define the similarity as:

Definition 5 (Similarity). *Let X be a set. A function $s : X \times X \to \mathbb{R}$ is called a similarity, if s is non-negative, symmetric, i.e. if for all $x, y \in X$ holds $s(x, y) \geq 0$, $s(x, y) = s(y, x)$; and $s(x, x) \geq s(x, y)$ for all $x, y \in X$ with equality if and only if $x = y$.*

Moreover, we define the normalised (dis-)similarity:

Definition 6 (Normalised (Dis-)Similarity). *Let X be a set consisting of measurements. A dissimilarity d is a normalised dissimilarity if and only if $0 \leq d(x, y) \leq 1$ for all $x, y \in X$. A similarity s is a normalised similarity if and only if $0 \leq s(x, y) \leq 1$ for all $x, y \in X$.*

Based on the definitions of normalised (dis-)similarity, the relationship between a normalised dissimilarity d and normalised similarity s is $d = 1 - s$.

4.1 Metric Selection

After formalising the concept of (dis-)similarity, we introduce 12 (dis-)similarities for numerical data comprising traditional distance metrics and metrics found in literature [3, 7, 35]. The considered (dis-)similarity are labeled as s_i and d_i in Table 2.

Addressing RQ.2, we compare the considered (dis-)similarity metrics when applying them to DS.1–DS.3. Our goal is to determine a set of metrics that yield the most diverse results assuming that this set of metrics detect dissimilarities in the data most efficiently. Initially, we transform each similarity to a dissimilarity: $m_{dis} = 1 - m_{sim}$. From here on we only consider dissimilarities.

The approach for filtering the best set of metrics is described in Algorithm 1: The goal of the algorithm is to group dissimilarities that perform similarly. Then, each of the groups is represented by one of the dissimilarities in it. For that, we first compute the dissimilarities over the selected time series of all data sets. For each dissimilarity, this yields a vector of pairwise dissimilarities, for each combination of two repetitions from the same data set. As these correlation vectors are aligned, we can then compute Pearson's r correlation between the resulting vectors.

Figure 1 shows the density chart of the correlations between the dissimilarities. From Figure 1, we observe that there appears to be a grouping between some dissimilarities (indicated by light squares),

Table 2: Considered (dis-)similarity metrics.

s_1: Kulczynski 2	$\frac{1}{2}\sum \min(x_i, y_i)\left(\frac{1}{\sum x_i} + \frac{1}{\sum y_i}\right) \in [0, 1]$
s_2: Bray-Curtis	$\frac{2\sum \min(x_i, y_i)}{\sum (x_i + y_i)} \in [0, 1]$
s_3: Roberts	$\frac{\sum (x_i + y_i)\frac{\min(x_i, y_i)}{\max(x_i, y_i)})}{\sum (x_i + y_i)} \in [0, 1]$
s_4: Ruzicka	$\frac{\sum \min(x_i, y_i)}{\sum \max(x_i, y_i)} \in [0, 1]$
s_5: Cosine	$\frac{\langle x, y\rangle}{\|x\|_2^2 \|y\|_2^2} \in [0, 1]$
s_6: Kumar-Hassebrook	$\frac{\langle x, y\rangle}{\|x-y\|_2^2 + \langle x, y\rangle} \in [0, 1]$
d_1: Scaled L_2	$\frac{\|x-y\|_2}{\|x\|_2 + \|y\|_2} \in [0, 1]$
d_2: Wave-Hedgets	$\sum \frac{\|x_i - y_i\|}{\max(x_i, y_i)} \in [0, 1]$
d_3: Soergel	$\frac{\sum \|x_i - y_i\|}{\sum \max(x_i, y_i)} \in [0, 1]$
d_4: normal. Canberra	$1 - \frac{2}{n}\sum \frac{\min(x_i, y_i)}{y_i + y_i} \in [0, 1]$
d_5: Clark	$\sqrt{\frac{1}{n}\sum \left(\frac{x_i - y_i}{x_i + y_i}\right)^2} \in [0, 1]$
d_6: Scaled RMSE	$L_2 = \frac{\|x-y\|_2}{2\max(\|x\|_2, \|y\|_2)} \in [0, 1]$

with $x = x_1, \ldots, x_n \in \mathbb{R}^n$ and $y = y_1, \ldots, y_n \in \mathbb{R}^n$ be time series, where for all i: $x_i, y_i \geq 0$.

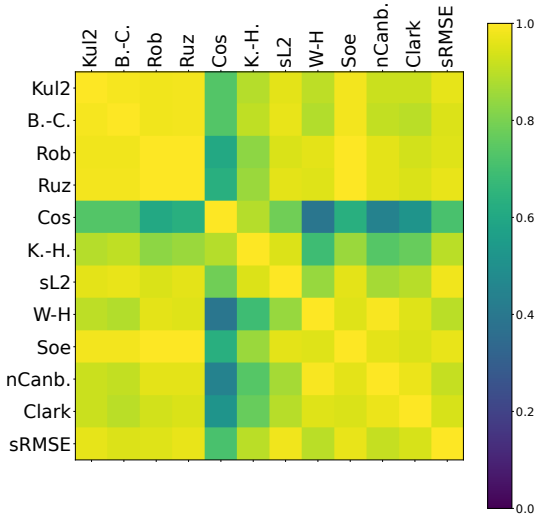

Figure 1: Correlation matrix for all dissimilarities

but there are also some dissimilarities that produce different results than any others (indicated by dark squares). In summary, the correlations from Figure 1 motivate our approach for grouping and selecting representative metrics for each group.

Algorithm 1: Filtering Dissimilarity Metrics

Input: measurement repetitions m, dissimilarity metrics s
Output: filtered dissimilarity metrics s_f

```
1  cm = calculateAverageCorrelation(m, s)
2  s_s = sortByAverageCorrelation(s, cm)
3  s_f = []
4  while len(s_s) > 0 do
5      m_s = s_s[0]
6      s_f.append(m_s)
7      for m_d in s_s do
8          if correlation(m_d, m_s) ≥ 0.9 then
9              s_s.remove(m_d)
10 return s_f
```

Table 3: Selected metrics for different data sets

Dataset	Selected metrics
DS.1	`Cosine`, `Wave-Hedgets`, `Scaled RMSE`
DS.2	`Cosine`, `Wave-Hedgets`, `Bray-Curtis`, `Scaled RMSE`
DS.3	`Cosine`, `Kumar-Hassebrook`, `Wave-Hedgets`
All	`Cosine`, `Wave-Hedgets`, `Kumar-Hassebrook`, `Scaled RMSE`

Therefore, Algorithm 1 calculates the presented correlation matrix as cm in the first step. We use this correlation matrix to group the metrics as follows: First, we compute the average dissimilarity for each metric in comparison to all other metrics. Following, the list s_s is sorted in ascending order. This way, the metric that has the most disagreement with all other metrics is listed first. Then, we pick the first metric from s_s in line 5 as our selected metric m_s and add it to the list of filtered metrics m_f. We then discard all other metrics m_d with $correlation(m_d, m_s) \geq 0.9$ from the list of remaining metrics s_s, as we assume that its properties are already covered by the newly included metric. This also removes m_s from s_s, as $correlation(m_s, m_s) = 1$. This procedure is repeated until all metrics are either added to the list of resulting filtered metrics m_s or discarded as they do not add enough information.

Note that the defined threshold of 0.9 was chosen as the goal of this experiment was to retain a sizeable set of metrics. By increasing the threshold, the final number of metrics can be higher, by lowering it, the resulting number will decrease. Future experiments might elaborate on different thresholds for these or other data sets.

4.2 Selected Metrics

Table 3 lists the resulting metrics for each of our data sets. In addition, we list the metrics chosen, if the correlation matrix of Figure 1 is used, i.e., if we execute Algorithm 1 on all data sets. Note that the chosen metrics are not sorted, as the ordering reflects the order in which the metrics were selected. Hence, the given ordering also describes the initial correlation ranking of Algorithm 1. Metrics on the left (i.e., `Cosine`) achieve the lowest average correlation and are chosen before the following metrics on the right.

While there are noticeable differences between the different data sets, we observe that there is a large consensus for the used metrics. `Cosine` and `Wave-Hedgets` are included in all data sets

as important metrics, with `Cosine` being the first metric chosen by all data sets. Additionally, the `Scaled RMSE` metric was chosen in two of the three data sets and is also included in the final selection. `Bray-Curtis` and `Kumar-Hassebrook` are only included in DS.2 and DS.3, respectively. However, `Kumar-Hassebrook` was also chosen as an important metric, when all data sets are considered.

We conclude that the results are to a certain degree transferable between data sets. Therefore, for the remainder of this work, we continue with the four dissimilarity metrics `Cosine`, `Wave-Hedgets`, `Kumar-Hassebrook` and `Scaled RMSE`.

5 DETERMINE DISSIMILARITY

We have identified four dissimilarity metrics, that together are capable of describing if the time series obtained from two runs of an experiment are similar. However, for a single experiment with ten runs (measurement repetitions) this results in a total of $4 * \sum_{i=1}^{9} i = 180$ scores, which is hard to interpret manually. Therefore, in a first step, we calculate the average dissimilarity across all dissimilarity metrics and combinations of measurement repetitions. This results in a single value that gives an overview on the dissimilarity between the measurement repetitions. If this aggregated value is low, we can already conclude that there are no divergent measurements. Yet, in case the value is at e.g., 0.1, it is unclear if there is some noise in the measurements that makes every run slightly different, or if there is a run that is different from the remaining runs, or if there is an effect that is observed in only a subset of runs.

Algorithm 2: Identification of Measurement Differences

Input: measurement repetitions m
Output: overall similarity s_t, similarity improvement s_i, cluster labels *labels*

```
labels = AgglomerativeClustering(m)
sumAll, countAll, sumCl, countCl = 0
for i = 0; i < len(m); i = i + 1 do
    for j = i + 1; j < len(m); j = j + 1 do
        sumAll = sumAll+dissimilarity(m[i], m[j])
        countAll = countAll + 1
        if labels[i] == labels[j] then
            countCl = countCl + 1
            sumCl = sumCl+dissimilarity(m[i], m[j])
s_t = sumAll/countAll
s_i = sumCl/countCl
return labels, s_t, s_i
```

To answer this question, we apply Algorithm 2: First, we use agglomerative clustering to group the measurements based on their dissimilarity [25]. We use this clustering method over others for two reasons: a) it dynamically selects the number of clusters, b) it can be used with pre-computed distance values, which means we can use the mean over the four dissimilarity metrics as distance measure. If the clustering creates more than one cluster, we conclude that there are divergent runs. If clustering returns more than one cluster, we calculate the difference between the average dissimilarity across all pairs of measurements and the average dissimilarity between pairs of measurements in the same cluster. By comparing the difference between these two values, we can quantify how different the identified clusters of runs are. A cluster may contain a single run that is considered to be different from the other runs. In any scenario where no cluster contains more than half of the measurements, we do not calculate the decrease in dissimilarity from clustering and instead return -1, as it seems that there are not enough similar measurements.

Based on this approach, we can quantify the dissimilarity between a large number of measurement repetitions in two values: the overall dissimilarity between measurements and the dissimilarity within the identified clusters. If a high overall dissimilarity between measurements, or a significantly lower dissimilarity within clusters is identified, we can conclude that the measurement repetitions are not homogeneous. For a more in-depth understanding of the differences, a manual/visual analysis is still required, but can use the identified clusters as a starting point.

Table 4: Similarity distribution across data sets

Sim.	DS.1	DS.2	DS.3
all	21	180	100
≤ 0.99	21	179	51
≤ 0.98	4	165	37
≤ 0.95	3	76	6
≤ 0.90	3	49	2
≤ 0.85	1	38	1
≤ 0.80	0	26	0

Table 5: Distribution of cluster sizes across data sets

	1	2	3	4	5	6	7	8	9
DS.1	21	–	–	–	–	–	–	–	–
DS.2	136	12	7	5	6	3	4	5	2
DS.3	91	8	–	1	–	–	–	–	–

6 RESULTS

This section discusses the results of applying our methodology to the identified data sets.

6.1 General Observations

In DS.1, we analyze one performance metric, latency; in DS.2, we look at latency and throughput; and finally DS.3 contains 20 different performance metrics from various system resources. In any case, we consider all time-series as univariate and analyze each performance metric independent from other performance metrics. An analysis of multivariate time-series is subject to future work.

This leads to the analysis of 301 experiments from the three data sets with more than 4,200 runs. Table 4 presents the similarity distribution for each data set and Table 5 shows the distribution of cluster sizes per data set. Finally, Table 6 lists the experiments that are flagged by our approach to have dissimilar repetitions.

Observation 1: Overall, we can state a very high similarity between all runs across all data points and across all data sets as illustrated by Table 4. This high similarity was expected as the time series come from performance benchmarks in laboratory environments. Only 26 set-ups show a similarity below 0.80. For all of these, more than one cluster is found as discussed later.

In accordance with the high similarity and as visible in Table 5, the number of experiments for which only a single cluster is formed is 248; 82% of all experiments. In particular, DS.1 does not unveil any clusters and will not be considered in the following discussion.

Observation 2: There is a correlation between high similarity and few clusters: Out of the 247 experiments with similarity $>$ 0.9, 237 (96%) have a single cluster only. In contrast, there are 11 experiments with a single cluster and similarity $\leq$ 0.9. The most dissimilar, yet single cluster data set, has a similarity of 0.813.

Further, all experiments with similarities $<$ 0.813 render multiple clusters. The highest similarity for a non-single cluster is about 0.985, while the overwhelming majority of non-single cluster results yield similarities $\leq$ 0.9 (43/53). The highest similarities with more than two clusters are 0.920 and 0.916 respectively and come from DS.3. All other 3+ clusters unveil a similarity $\leq$ 0.85.

Observation 3: Latency seems to be more sensitive to disturbances than throughput, which has also been observed elsewhere [23]. Table 6 contains all experiments that result in more than one cluster and hence are suspected to have dissimilar repetitions. Overall, 20 experiments address a throughput-oriented performance metric (throughput and iops), while the majority of 33 suspicions are due to latency. Moreover, in DS.2 all experiments that are flagged due to throughput are also flagged due to latency, but not vice versa.

Observation 4: Table 6 shows that there are a 15 cases for which the number of clusters exceeds half of the number of runs. That is with 10 runs, the number of clusters is $>$ 5. In nine of these scenarios, all but one cluster contain only a single element. In six cases, two clusters contain more than one element. The fact that so many clusters are generated for these cases allows us to conclude that overall the measurements are extremely heterogeneous and no conclusions can be drawn from them. In these cases a manual (visual) inspection of the data is highly recommended (cf. Section 5).

Observation 5: Investigating Table 6, we can state that clustering is not always successful in reducing dissimilarity. Particularly, for AF, AO, AR, AT, AX there is barely any improvement visible. Despite the fact that clustering isolates a single time series in all these experiments. On the other hand, there are also very impressive improvements such as A, B, C, AB.

6.2 Illustration by Example

For the sake of space, we are not able to illustrate the benefits and shortcomings of our approach using examples based on DS.2 and DS.3. Our supplementary material[2] contains five such examples. They have been chosen such that each example represents one archetype of time series similarity. As a baseline, we consider the common two sigma rule, where every measurement that deviates by more than two sigma from the mean is considered anomalous.

[2]https://doi.org/10.5281/zenodo.5786869

7 RELATED WORK

Experimental evaluation in many scientific disciplines usually involves uncertainty. Therefore, NIST and ISO have standards and guidelines for evaluating, expressing, and reporting uncertainty in measurements [19, 34]. In addition, there are also several scientific papers focusing on uncertainties in measurements: (i) articles focusing on providing different principles for a good and clean measurement approach; (ii) articles investigating performance variability during measurements; (iii) articles proposing frameworks for reproducible measurements with reasonable effort.

Papadopoulos et al. [27] suggested eight principles (e.g., open access artifacts of measurements) for reproducible performance evaluation in cloud computing. Another seven principles were introduced by Schwarzkopf et al. [29]. For supercomputing environments, Hoefler and Belli [15] introduced 12 principles similar to [27]. Frachtenberg and Feitleson [11] provide a framework of 32 pitfalls in supercomputing environments, each combining practical principles and research challenges. For networked systems, Krishnamurthy et al. [22] provide 12 checkpoints that researchers can use to audit their experiments. Similarly, Vitek and Kalibera discuss common mistakes made by researchers [37].

Uta et al. [36] examined the question whether big data performance is reproducible based on performance variability. Similar work for measurements in cloud computing was done by Laaber et al. [24], Iosup et al. [18], and Folkerts et al. [10]. Others investigated sources of measurement bias in experimental work [26, 39]

The Collective Knowledge Framework [12] enables systematic recording of individual experimental steps, which permits independent reproduction and contribution of additional results. Another framework is DataMill [6], which randomizes selected environmental conditions to improve the generalizability of specific measurements. In addition, works such as [13, 16] provide principles on how to avoid the most common pitfalls of experimental evaluation.

8 THREATS TO VALIDITY

The goal of this paper is to investigate approaches to detect dissimilarities in time series gained from benchmarking. Yet, for several reasons, this explorative study cannot claim generalizability to arbitrary data sets nor to all performance benchmarking experiments.

First, we only worked with three data sets. Further, the experiments in data set DS.3 have many runs ($>$ 25), but in most cases the time series of a run only consist of three data points. Some of these time series are even shorter than that. For all of these cases the dissimilarity is very low. This may be a consequence of the short time series or of the large number of repetitions.

There are also several threats to the validity of the metric selection in Algorithm 1 (Section 4): (i) all of our results are based on the selected metrics presented in Table 2. Therefore, the presented results might be substantially different, if other metrics are added to the list. (ii) We rely on Pearson's r coefficient to measure the correlation of two metrics and choose a threshold of 0.9 in the filtering step of Algorithm 1. Yet, Pearson's coefficient only captures linear correlations and fails for other correlations. Applying a different measure of correlation, as well as using a different threshold is very likely to impact the results.

Table 6: Experiements with similarity suspects found using clustering. The last column lists the improvement after applying the clustering. A after dissimilarity of — indicates that more than half of all elements were put in different clusters.

id	origin	metric	dissim.	runs/ clust.	after
A	DS.2	latency	0.110	10/2	0.018
B	DS.2	latency	0.186	10/3	0.026
C	DS.2	latency	0.126	10/2	0.027
D	DS.2	latency	0.077	10/1	0.031
E	DS.2	latency	0.379	10/8	—
F	DS.2	latency	0.549	10/8	—
G	DS.2	latency	0.573	10/9	—
H	DS.2	latency	0.262	10/3	0.118
I	DS.2	latency	0.255	10/4	0.135
J	DS.2	latency	0.357	10/4	0.162
K	DS.2	latency	0.325	10/8	—
L	DS.2	latency	0.340	10/7	—
M	DS.2	latency	0.259	10/5	0.143
N	DS.2	latency	0.232	10/5	-1
O	DS.2	latency	0.212	10/3	0.106
P	DS.2	latency	0.300	10/9	—
Q	DS.2	latency	0.321	10/5	0.128
R	DS.2	latency	0.206	10/5	0.130
S	DS.2	latency	0.261	10/6	0.174
T	DS.2	latency	0.268	10/4	0.144
U	DS.2	latency	0.376	10/8	—
V	DS.2	latency	0.280	10/8	—
W	DS.2	latency	0.281	10/7	—
X	DS.2	latency	0.167	10/3	0.127
Y	DS.2	latency	0.136	10/2	0.100
Z	DS.2	latency	0.373	10/6	—
AA	DS.2	latency	0.422	10/7	—
AB	DS.2	throughput	0.107	10/2	0.020
AC	DS.2	throughput	0.085	10/2	0.045
AD	DS.2	throughput	0.124	10/2	0.031
AE	DS.2	throughput	0.074	10/2	0.030
AF	DS.2	throughput	0.206	10/5	0.191
AG	DS.2	throughput	0.193	10/3	0.112
AH	DS.2	throughput	0.173	10/3	0.131
AI	DS.2	throughput	0.240	10/6	—
AJ	DS.2	throughput	0.182	10/2	0.177
AK	DS.2	throughput	0.235	10/7	—
AL	DS.2	throughput	0.208	10/5	0.111
AM	DS.2	throughput	0.124	10/2	0.104
AN	DS.2	throughput	0.189	10/3	0.171
AO	DS.2	throughput	0.175	10/2	0.167
AP	DS.2	throughput	0.202	10/4	0.183
AQ	DS.2	throughput	0.178	10/4	0.139
AR	DS.2	throughput	0.165	10/2	0.156
AS	DS.3	latency	0.084	35/2	0.057
AT	DS.3	latency	0.780	61/2	0.070
AU	DS.3	iops	0.038	35/2	0.025
AV	DS.3	latency	0.064	33/2	0.054
AW	DS.3	latency	0.039	35/2	0.026
AX	DS.3	latency	0.028	35/2	0.017
AY	DS.3	throughput	0.101	33/2	0.071
AZ	DS.3	throughput	0.196	29/4	0.056
BA	DS.3	latency	0.029	61/2	0.021

Finally, the metric selection in Section 5 relies on a number of somewhat arbitrarily chosen thresholds, namely, the clustering algorithm and the interpretation of what is considered a large overall dissimilarity or a significant reduction in dissimilarity from clustering.

In general, the proposed approach is limited to time series of measurement data obtained from software performance experiments. Other time series might have different properties, which could inhibit the applicability of our approach.

9 CONCLUSIONS

This paper investigates the problem of how to analyze benchmarking results that produce time series. Because running the same experiment multiple times may yield very different results due to non-deterministic elements, an analysis may benefit from understanding that results fall in different groups so that each group may be investigated independently. Our approach centers around using measures to determine the (dis-)similarities between time series using openly available data sets for analysis and validation.

9.1 Summary

For gathering data sets, we evaluated 1,000+ scientific papers from the cloud and performance engineering domain. Surprisingly, we could only identify three of data sets with 301 experiments in total.

Afterwards, we applied 12 different (dis-)similarities to the data sets and compared their behaviour. Filtering out (dis-)similarities with alike behaviour, a total of four (dis-)similarities stood out. We used these in a clustering algorithm in order to group runs of experiments according to their (dis-)similarity. Understanding what clustering improves similarity gives some guidance to which experiments may require a manual review. Overall, our approach identified 53 experiments where more than one cluster was found.

Finally, we performed a detailed analysis of five experiments where our approach detected dissimilar runs. This showed that outliers exist in the data sets that would not have been detected by only using means and standard deviations. Yet, it also taught us to not blindly trust our approach and that further research is needed.

9.2 Future Work

Our explorative study raises multiple questions and can only be considered as starting point for further research. While we identified

four (dis-)similarities with heterogeneous behaviour, we have not covered all (dis-)similarities. For instance, Iglesias and Kastner [17] partially use different similarities for their pattern analysis for time series clusters. Also, approaches like Dynamic Time Warping have not been considered yet [2], nor have strategies of pre-processing the data, and multivariate data in general. Unfortunately, we have not managed to avoid thresholds completely.

Finally, the capabilities and features of our approach are not fully understood. This includes the impact of the time-series length and the number of runs. Also, an exact analysis of the decisions of the clustering algorithm is missing. In order to address these open questions access to more and larger data sets is necessary.

9.3 Call to Action

A disappointing outcome of the research conducted in this paper is that in our community OpenData is not as wide-spread as in, e.g., medicine and physics. Analyzing 1,000+ papers resulted only in 3 data sets useful for our work despite very generic acceptence criteria. Admittedly, we found several published data sets that did not fit our scope. Ironically, many authors used publicly available data as input for their evaluations.

Hence, we call on the systems community to publish alongside their research papers not only code, but also their evaluation data. This helps to confirm outcomes, but equally as important, it also enables more and higher quality meta-studies.

REFERENCES

[1] David Bermbach, Erik Wittern, and Stefan Tai. 2017. *Cloud service benchmarking*. Springer.

[2] Donald J Berndt and James Clifford. 1994. Using dynamic time warping to find patterns in time series.. In *KDD workshop*, Vol. 10. Seattle, WA, USA:.

[3] Sung-Hyuk Cha. 2007. Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions. *International Journal of Mathematical Models and Methods in Applied Sciences* 1, 4 (2007).

[4] Shihyen Chen, Bin Ma, and Kaizhong Zhang. 2007. The Normalized Similarity Metric and Its Applications. In *2007 IEEE/BIBM*.

[5] Brian F. Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, and Russell Sears. 2010. Benchmarking Cloud Serving Systems with YCSB. In *Proceedings of the 1st ACM Symposium on Cloud Computing* (Indianapolis, Indiana, USA) *(SoCC '10)*. Association for Computing Machinery, 12 pages.

[6] Augusto Born de Oliveira, Jean-Christophe Petkovich, Thomas Reidemeister, and Sebastian Fischmeister. 2013. DataMill: Rigorous Performance Evaluation Made Easy. In *ACM/SPEC International Conference on Performance Engineering*.

[7] Michel Marie Deza and Elena Deza. 2009. *Encyclopedia of Distances* (1 ed.). Springer, Berlin, Heidelberg.

[8] Simon Eismann, Cor-Paul Bezemer, Weiyi Shang, Dušan Okanović, and André van Hoorn. 2020. Microservices: A Performance Tester's Dream or Nightmare?. In *ACM/SPEC International Conference on Performance Engineering (ICPE '20)*. 12 pages.

[9] Victor A.E. Farias, Flávio R.C. Sousa, José Gilvan R. Maia, João Paulo P. Gomes, and Javam C. Machado. 2018. Regression based performance modeling and provisioning for NoSQL cloud databases. *Future Generation Computer Systems* 79 (2018).

[10] Enno Folkerts, Alexander Alexandrov, Kai Sachs, Alexandru Iosup, Volker Markl, and Cafer Tosun. 2013. Benchmarking in the Cloud: What It Should, Can, and Cannot Be. In *Selected Topics in Performance Evaluation and Benchmarking*.

[11] Eitan Frachtenberg and Dror G. Feitelson. 2005. Pitfalls in Parallel Job Scheduling Evaluation. In *JSSPP*.

[12] Grigori Fursin, Anton Lokhmotov, and Ed Plowman. 2016. Collective Knowledge: Towards R&D sustainability. In *DATE*.

[13] Andy Georges, Dries Buytaert, and Lieven Eeckhout. 2007. Statistically Rigorous Java Performance Evaluation. *SIGPLAN Not.* 42, 10 (2007).

[14] Johannes Grohmann, Daniel Seybold, Simon Eismann, Mark Leznik, Samuel Kounev, and Jörg Domaschka. 2020. Baloo: Measuring and Modeling the Performance Configurations of Distributed DBMS. In *MASCOTS*.

[15] Torsten Hoefler and Roberto Belli. 2015. Scientific Benchmarking of Parallel Computing Systems: Twelve Ways to Tell the Masses when Reporting Performance Results. In *SC*.

[16] Vojtěch Horký, Peter Libič, Antonin Steinhauser, and Petr Tůma. 2015. DOs and DON'Ts of Conducting Performance Measurements in Java. In *ACM/SPEC International Conference on Performance Engineering*.

[17] Félix Iglesias and Wolfgang Kastner. 2013. Analysis of Similarity Measures in Times Series Clustering for the Discovery of Building Energy Patterns. *Energies* 6, 2 (2013).

[18] Alexandru Iosup, Radu Prodan, and Dick H. J. Epema. 2014. IaaS Cloud Benchmarking: Approaches, Challenges, and Experience. In *Cloud Computing for Data-Intensive Applications*.

[19] ISO. 2008. *Evaluation of measurement data – Guide to the expression of uncertainty in measurement*. Technical Report JCGM 100:2008. Joint Committee for Guides in Metrology (JCGM/WG 1).

[20] Samuel Kounev, Klaus-Dieter Lange, and Jóakim von Kistowski. 2020. *Systems Benchmarking*. Springer.

[21] Rouven Krebs, Christof Momm, and Samuel Kounev. 2014. Metrics and techniques for quantifying performance isolation in cloud environments. *Science of Computer Programming* 90 (2014). Special Issue on Component-Based Software Engineering and Software Architecture.

[22] Balachander Krishnamurthy, Walter Willinger, Phillipa Gill, and Martin Arlitt. 2011. A Socratic Method for Validation of Measurement-based Networking Research. *Comput. Commun.* 34, 1 (2011).

[23] Jörn Kuhlenkamp, Markus Klems, and Oliver Röss. 2014. Benchmarking Scalability and Elasticity of Distributed Database Systems. *Proc. VLDB Endow.* 7, 12 (Aug. 2014), 12 pages.

[24] Christoph Laaber, Joel Scheuner, and Philipp Leitner. 2019. Software microbenchmarking in the cloud. How bad is it really? *Empirical Software Engineering* 24, 4 (2019).

[25] Daniel Müllner. 2011. Modern hierarchical, agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378* (2011).

[26] Todd Mytkowicz, Amer Diwan, Matthias Hauswirth, and Peter F. Sweeney. 2009. Producing Wrong Data Without Doing Anything Obviously Wrong! *SIGPLAN Not.* 44, 3 (2009).

[27] Alessandro Vittorio Papadopoulos, Laurens Versluis, André Bauer, Nikolas Herbst, Jóakim Von Kistowski, Ahmed Ali-Eldin, Cristina Abad, José Nelson Amaral, Petr Tuma, and Alexandru Iosup. 2019. Methodological principles for reproducible performance evaluation in cloud computing. *IEEE Transactions on Software Engineering* (2019).

[28] Joel Scheuner and Philipp Leitner. 2018. Estimating Cloud Application Performance Based on Micro-Benchmark Profiling. In *International Conference on Cloud Computing (CLOUD)*.

[29] Malte Schwarzkopf, Derek Gordon Murray, and Steven Hand. 2012. The Seven Deadly Sins of Cloud Computing Research. In *HotCloud*.

[30] Daniel Seybold, Johannes Grohmann, Simon Eismann, Mark Leznik, Samuel Kounev, and Jörg Domaschka. 2020. Data Sets for Measuring and Modeling the Performance Configurations of Distributed DBMS.

[31] Daniel Seybold, Moritz Keppler, Daniel Gründler, and Jörg Domaschka. 2019. Mowgli: Finding Your Way in the DBMS Jungle. In *ACM/SPEC International Conference on Performance Engineering* (Mumbai, India) *(ICPE '19)*. Association for Computing Machinery, 12 pages.

[32] Daniel Seybold, Simon Volpert, Stefan Wesner, André Bauer, Nikolas Herbst, and Jörg Domaschka. 2019. Kaa: Evaluating Elasticity of Cloud-Hosted DBMS. In *International Conference on Cloud Computing Technology and Science (CloudCom)*.

[33] Daniel Seybold, Stefan Wesner, and Jörg Domaschka. 2020. King Louie: Reproducible Availability Benchmarking of Cloud-Hosted DBMS. In *Proceedings of the ACM Symposium on Applied Computing* (Brno, Czech Republic) *(SAC '20)*. Association for Computing Machinery, 10 pages.

[34] Barry N. Taylor and Chris E. Kuyatt. 1994. *Guidelines for Evaluating and Expressing the Uncertainty of NIST Measurement Results*. Technical Report 1297. National Institute of Standards and Technology (NIST).

[35] Mark A. Tschopp and Efrain Hernandez-Rivera. 2017. *Quantifying Similarity and Distance Measures for Vector-Based Datasets: Histograms, Signals, and Probability Distribution Functions*. Technical Report ARL-TN-0810. US Army Research Laboratory.

[36] Alexandru Uta, Alexandru Custura, Dmitry Duplyakin, Ivo Jimenez, Jan Rellermeyer, Carlos Maltzahn, Robert Ricci, and Alexandru Iosup. 2020. Is Big Data Performance Reproducible in Modern Cloud Networks?. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. USENIX Association.

[37] Jan Vitek and Tomas Kalibera. 2012. R3: Repeatability, Reproducibility and Rigor. *SIGPLAN Not.* 47, 4a (2012).

[38] Jóakim von Kistowski, Simon Eismann, Norbert Schmitt, André Bauer, Johannes Grohmann, and Samuel Kounev. 2018. TeaStore: A Micro-Service Reference Application for Benchmarking, Modeling and Resource Management Research. In *MASCOTS (MASCOTS '18)*. IEEE Computer Society.

[39] Boxun Zhang, Alexandru Iosup, Johan A. Pouwelse, Dick H. J. Epema, and Henk J. Sips. 2010. Sampling Bias in BitTorrent Measurements. In *Euro-Par*.