Benchmarking Crimes (Seminar Paper)

Mark Waschkeit

Disclaimer: All benchmarking crime names and numerations have been quoted from the original paper and get represented in cursive characters. Furthermore, Chapter 2 summarizes paper [2] which is also referred to indirectly in many other places in this document.

1 INTRODUCTION

Benchmarking, a fundamental practice in assessing system performance, plays a critical role in research and technological advancements. However, the field is fraught with challenges known as benchmarking crimes, which, if unaddressed, can compromise the integrity of benchmarking studies. In this examination, we systematically categorize and analyze benchmarking crimes, elucidating their potential impact on benchmark quality.

Our focus is on understanding how these crimes, ranging from misrepresentations to critical omissions, can influence the completeness, relevance, soundness, and reproducibility of benchmarks. By presenting a comprehensive catalog of benchmarking crimes, we aim to provide researchers with valuable insights and practical suggestions to fortify their methodologies against these pitfalls.

This exploration seeks to raise awareness within the research community, promoting meticulous and ethical benchmarking practices to enhance the reliability of benchmarking outcomes.

2 SUMMARY OF "BENCHMARKING CRIMES: AN EMERGING THREAT IN SYSTEM SECURITY"

2.1 Types of Benchmarking Crimes

Some benchmarking crimes are interrelated and often share common characteristics. For this reason, 22 crimes have been identified which have been grouped into six categories.

The first category, *A: selective benchmarking*, addresses the fact that a system as a whole cannot be adequately measured by a single number due to the complexity and multifacetedness of such systems. Benchmarking crime *A1: not evaluating potential performance degradation* involves the omission of benchmarks that not only show the system's impovements compared to the current state of the art. Thus they fail to highlight potential degradation in other aspects. Committing benchmarking crime *A2: benchmark subsetting without proper justification* occurs when subbenchmarks measuring specific aspects of a system are executed to represent the system's overall performance without adequate justification. A single aspect or subset of aspects of a system is often not representative of the system as a whole, given that the system's function frequently involves multiple roles. Therefore, the paper must ensure that the specified number(s) appropriately represent the system's performance. Benchmarking crime *A3: selective data set hiding deficiencies* is committed when a paper does not complete a range of benchmarks based on all possible settings, thus showing only a fraction of the system's performance and resulting in inaccurate readings. For example, if these readings suggest linear scalability, it is almost certain that this growth is finite, and limitations to that increase must be identified for the sake of completeness. [1] [2]

B: Improper handling of benchmark results as the second category concerns misinterpreting and misrepresenting benchmarks that actually have been thoroughly performed. Firstly, *B1: microbenchmarks representing overall performance* refers to microbenchmarks being depicted as if they would reflect the system's general performance, even though they only measure a certain aspect or set of operations. Secondly, the CPU's overhead may be misinterpreted. For example, if system A has a lower runtime for a specific set of operations compared to system B, the runtime suggests system A is better than system B in general. This may not take the CPU's load into account. If system A only has a CPU usage of 20%, while system B's CPU usage is at 50%, the first system may be more available for other tasks being handled in parallel. This benchmark crime is referred to as *B2: throughput degraded by x% ⇒ overhead is x%*. Thirdly, *B3: creative overhead accounting* alludes to any fault in terms of miscalculating, misinterpreting, or misrepresenting overhead. For example, if the difference between 20% overhead and 40% overhead is depicted as an increase of 20% (40% - 20%) rather than an increase of 100% (40% / 20% - 100%), the depiction is not only misleading but also incorrect. Fourthly, the variability of measurements has to be taken into account. For example, a 10% decrease in runtime sounds positive, but is negligible if the random variation due to the uncertainty of measurement is also around 10%. We make mention of this as *B4: no indication of significance of data*. Lastly, the crime *B5: incorrect averaging across benchmark scores* is committed if average overhead is not calculated by using the geometric mean. Using the arithmetic mean or any other method of averaging does not create a suitable mean, as choosing a different baseline for the calculation results in a different outcome each time. [1] [2]

The third category, *C: using the wrong benchmarks*, comments on three types of benchmarking crimes in which alternative ways of measurement not appropriate for achieving correct results have been used. The first type is *C1: benchmarking of a simplified simulated system*. As the naming implies, simplified systems such as emulated versions have been used to conduct the benchmarks. The properties of an emulated system vary compared to a real system, implying a difference in the results. The second type refers to benchmarks not suitable for measuring what they intend to do. For example, quantifying overhead in a test using user-space computations while overhead usually mainly occurs on system calls in the kernel. We allude to this crime with *C2: inappropriate and misleading benchmarks*. Finally, *C3: same dataset for calibration and validation* refers to the set of training data intersecting the set of test data. For example, a system optimized for a specific workload being tested on this exact workload or even parts of it is expected to perform well. [1] [2]

D: Improper comparison of benchmarking results as the fourth category represents three benchmarking crimes revolving around multiple issues when comparing the outcomes of benchmarks. Firstly, *D1: no proper baseline* addresses items of the benchmark

having changed in regard to the test conditions, such as adding or removing demands to the system. Following is *D2: only evaluate against yourself*, which alludes to the paper only referring to improvements of their own systems rather than to the state of the art. Lastly, there is unfair benchmarking, such as a comparison with suboptimal parameters for the competitor's system. This will be referred as *D3: unfair benchmarking of competitors*. [1] [2]

The fifth category *E: benchmarking omissions* is about ignoring or neglecting certain required aspects in benchmarking. Firstly, *E1: not all contributions evaluated* alludes to not adjudicating whether self-made goals and claims, which have been made previously, have been met or not. Secondly, *E2: only measure runtime overhead* means disregarding any other forms of overhead besides runtime, like memory consumption. Thirdly, the reliability of statements, which have been generated automatically, has to be checked to determine their significance. Disrespecting that results in committing benchmark crime *E3: false positives/negatives not tested*. Fourthly, if a system consists of multiple independent components, these components can be optimized to increase performance at the cost of also increasing complexity. To consider whether these optimizations are worth it, their impact has to be measured individually. If those measurements aren't included in a benchmark, it commits benchmarking crime *E4: elements of solution not tested incrementally*.

Finally, category *F: missing information* as the sixth category relates to cases in which a paper does not specify enough information about made benchmarks. *F1: missing platform specification* means that the hardware is described insufficiently to replicate said benchmarks. Not only the hardware itself has to be specified, but also its architecture and the whole setup. The software counterpart to this benchmarking crime is *F2: missing software versions*, which exclusively concerns all types of software, such as the operating system and programs used, as well as their versions. Next, *F3: sub-benchmarks not listed* refers to separate independent sub-benchmarks not being listed, but only a universal value. Necessary information for the reader is missing, since they contain valuable details about the system's strengths and weaknesses. Lastly, *F4: relative numbers only* refers to papers not presenting absolute numbers. For example, only stating that system A has twice the overhead of system B contains less information than specifying system A having an overhead of 20%, while system B's overhead is 10%. [1] [2]

2.2 Impact of Benchmarking Crimes

The 22 benchmarking crimes elaborated above all affect the quality of a benchmark individually and in different aspects. Therefore, the influence on the value of the paper using such benchmarks is divided into the influence on *Completeness*, *Relevancy*, *Soundness*, and *Reproducibility*. A paper is *complete* when it confirms every assertion it makes and points out all possible weaknesses in the system. The benchmark has to be *relevant* in informing the reader about significant details of the said system. All measured numbers have to be *sound* within sensible preciseness and actually mean what is intended for them to state. Everything mentioned in the paper has to be sufficient to *reproduce* the tests and their results. [2]

The category *A: selective benchmarking* mostly threatens the paper's completeness as it is mainly about hiding negative aspects of the system. This makes it seem better than it actually is. The same problem applies when subbenchmarks are used to represent the whole system (A2) which also influences its relevance negatively since the computed number(s) may not be as meaningful. If not every setting has been tested (A3), the lack of information about it may hide deficiencies and at least make the test incomplete. [2]

Regarding the second category *B: improper handling of benchmark results*, a paper's soundness is heavily influenced by these individual benchmarking crimes. Assuming the overhead by interpreting the lesser throughput without taking the CPU's load into account (B2) results in a threat to soundness like any other wrong creative overhead accounting (B3) does. Unsound results are also induced by incorrect averaging (B5) over measured numbers. Since microbenchmarks are not useful for representing the general performance of a system (B1), even if the microbenchmarks have been executed without committing any benchmarking crimes, relevancy suffers when ignored. If the expected variation of measurement results isn't specified, the significance of data cannot be evaluated (B4), which lacks completeness. [2]

In the third category *C: using the wrong benchmarks*, a specific threat to relevancy appears. Testing a system on the data it has been trained on (C3) normally results in very positive outcomes, but does not take other - probably more realistic - scenarios into account. The same problem applies to benchmarks not measuring what they are built to quantify (C2), which questions the relevancy of the outcome. For instance, if intricate graphical operations are the focus of the system, it's essential to evaluate the system's performance in handling them rather than solely relying on assessments based on simple arithmetic computations. Benchmarking on an emulated system (C1), on the other hand, is relevant, but due to the differences between a real system and an emulated one, the soundness is rather questionable [2]

D: Improper comparison of benchmarking results almost exclusively affects relevancy due to the nature of wrong comparisons. Changing the requirements of the system from one benchmark to another and then comparing the outcomes (D1) results in an irrelevant comparison. Only comparing one's results with one's past results (D2) also results in an irrelevant answer. This is due to the reason that even though the new solution is better, it might still be worse than the state of the art and may trick the reader into thinking that the system is better than it actually is. Unfair benchmarking of competitors (D3) also threatens relevancy because comparing system A with optimal settings to system B with suboptimal settings results in a meaningless outcome. [2]

E: Benchmarking omissions essentially affects completeness since it is about ignoring or neglecting certain required aspects in benchmarking. If a paper does not clarify whether and how claims could be proven or disproven (E1), an important aspect of the benchmark is missing. This also applies to disregarding any overhead besides runtime overhead (E2) or not identifying the accuracy of statements, which have been made automatically (E3). For systems containing multiple independent components or optimizations, they have to be tested individually (E4). Otherwise, the importance of individual components or optimizations is not established,

and it is impossible to decide whether it is worth it to apply that optimization at the cost of increasing complexity or not. [2]

The benchmarking crimes in the last category *F: missing information* threaten completeness and reproducibility. If neither hardware specification (F1) nor software specification (F2) is included, it is impossible to replicate the benchmark. Depending on both these missing specifications, trying to reproduce the benchmark may result in an error, which can be small enough to be hardly noticeable or big enough to produce completely different results. Also, not having subbenchmarks listed (F3) makes it impractical to find out the system's strengths and weaknesses. In addition to that, not having this information makes the benchmark lack completeness. Not listing absolute numbers but relative ones makes it hard to find out whether certain aspects make a big difference. For example, an overhead of 1% relatively compared to an overhead of 2% is the same increase of 100% (e.g. x2) as it would be for 25% overhead compared to 50% overhead. Depending on the system, this may or may not have a big impact and makes the benchmark lack completeness, preventing proper interpretation of benchmark numbers. [2]

2.3 Suggestions on how to prevent Benchmarking Crimes

This chapter is a three-step guidance on how to prevent the highest impact benchmarking crimes, easily avoidable ones, and one of the most common crimes.

First of all, we focus on those crimes that can have the biggest influence on the benchmark's quality and, therefore, the paper's quality as a whole. *A1: not evaluating potential performance degradation* can be mitigated by considering everything that can influence the result. Common interferences include CPU, memory, input/output, system calls, and typical problems in the context of working in parallel. Investigating each of these interferences, analyzing them in benchmarks, and presenting them for the reader can help eliminate this crime. Another high-impact crime is *B2: throughput degraded by x% ⇒ overhead is x%*. This crime can be resolved by ensuring that the system is used at its full capacity (e.g. 100% CPU load) when benchmarking because otherwise, input/output latencies can be the bottleneck. This can often be guaranteed by parallelizing the system adequately. If it is not possible to guarantee a 100% load on the system, then its load by time has to be represented in the benchmark to be obvious for the reader for interpretation. *D1: no proper baseline* also heavily impacts the benchmark's quality. Most importantly, a proper baseline has to be defined, which is a reference point for other benchmarking results. To find a proper baseline it may help to think about the problems the system encounters without having the solution applied, which you want to benchmark. [2]

From now on the focus is on easily avoidable benchmarking crimes. The first one being *B4: no indication of significance of data*. This can be prevented by measuring the accuracy of the benchmark's results and including them within the benchmark. The second easily avoidable benchmarking crime is *B5: incorrect averaging across benchmark scores* and can be eliminated by using the geometric mean, as meantioned earlier. Next, both the benchmarking crimes *F1: missing platform specification* and *F2: missing software versions* are prevented by adding the related information needed. This being hardware information for F1 and programs and their respective versions for F2. Also *F3: subbenchmarks not listed* is prevented as easily as the other crimes in category *F: missing information* as it can be addressed by performing various subbenchmarks testing a system's strengths and weaknesses and adding said subbenchmarks to the overall benchmark. Lastly, benchmarking crime *F4: relative numbers only* can be avoided by adding the measured absolute number instead of, or in addition to their relative counterparts. [2]

Finally, to cover one of the most prominent benchmarking crimes, we have to address *A2: benchmark subsetting without proper justification*. To give the reader a precise overview of the system, as many subbenchmarks as possible have to be performed. If a specific subbenchmark has not been undertaken, the reason for this has to be clarified. In addition to running those benchmarks, it has to be made clear that the overall result is not representative for the system. The system is represented best by the subbenchmarks. [2]

3 ASSESSMENT

3.1 Addressing Challenges in Benchmarking

Preventing benchmarking crimes isn't always as straightforward as augmenting the system's specifications, a step crucial for avoiding crimes in the category *F: missing information*. It requires a critical examination of benchmarking techniques used and a clear definition of the goals regarding what to measure.

This is particularly crucial in the realm of category *A: selective benchmarking*, where the objective is to encompass as many aspects of the system as possible, providing the reader with the most comprehensive understanding. To achieve this, one should contemplate the system's strengths and weaknesses, focusing on specific components for testing. In a pinch, testing randomly with diverse benchmarks can be considered to compile a spectrum of both positive and negative characteristics of the system, especially if these aspects haven't been identified yet.

B: Improper handling of benchmark results takes a different angle, as the benchmark itself might be flawless, but its representation could be inaccurate. The representation is inaccurate, if it is incomplete or vaguely depicted in confusing or misleading diagrams. Rectifying the misrepresentation of a benchmarking result involves clearly defining what is being benchmarked (e.g. what tests are being used to determine a system's performance). This specification is crucial not only for the benefit of the reader but also for the benchmarkers themselves. Without such clarity, there is a risk of utilizing results to portray the system's aspects in an unjustifiable and misleading manner.

Another challenge arises when dealing with chapter *C: using the wrong benchmarks*. Testing with simplified simulated systems might be unavoidable, especially if the actual system cannot be tested directly. This prompts the question of how to simulate a system in the most realistic manner, including factors that the simulated system may not necessarily need to function correctly but should emulate the real system closely nevertheless. If simulating a system is unavoidable, it is recommended to validate the benchmarking results with data measured within the real system and therefore

using realistic scenarios in the tests. While there's no one-size-fits-all solution to this question, it is generally advisable to avoid simulated systems whenever possible to mitigate this challenge.

Finally, the most significant challenge lies in avoiding the omission of crucial benchmarks, relevant information, and evidence supporting claims. Crucial benchmarks encompass all types of sub-benchmarks that provide the most comprehensive overview of the system, with the selection depending heavily on the specific system under examination. Various tests demonstrating the importance of the benchmark, including assessments of false negatives/positives and the significance of results, such as their standard deviations, pertain to relevance information. Lastly, ensuring evidence supporting claims involves testing all assertions to determine whether they have been met or not.

Benchmarking is a complex task, and there can't be a one-size-fits-all instruction for conducting the perfect benchmark. However, adhering to the recommendations outlined in *2.3 Suggestions on how to prevent Benchmarking Crimes* and staying mindful of all the benchmarking crimes detailed in *2.1 Types of Benchmarking Crimes* provides a solid starting point to avoid many benchmarking pitfalls.

3.2 Relevance

A comparison of system security defenses in 2010 and 2015 indicates that there have been no significant changes in the frequency and types of benchmarking crimes committed by the papers during those years. Specifically, 25% (86 out of 346) of crime/paper pairs involved high-impact crimes, while 33% (167 out of 505) of crime/paper pairs involved non-high-impact crimes. A crime/paper pair is defined as an element of the cross-product of all benchmarking crimes and all considered papers. The varying number of crime/paper pairs considered in both scenarios (high impact vs. non-high impact crimes) is due to the fact that not every crime is applicable to each paper, often caused by papers being underspecified. [2]

As already stated before the differences in the amount and types of benchmarking crimes committed in 2010 compared to 2015 are generally minor. The only noteworthy crimes being *E1: not all contributions evaluated* as the amount of papers which committed that crime decreased significantly over the period of those 5 years. Since there is no fundamental improvement in other benchmarking crimes it suggests a lack of awareness regarding benchmarking crimes in general rather than malicious intentions. It emphasizes the need to raise awareness of benchmarking crimes and to sensitize the research community to more conscious and ethical benchmarking practices. [2]

4 CONCLUSION

In the pursuit of achieving meaningful benchmarking results, it's essential to recognize the diverse range of benchmarking crimes that can compromise the integrity of the assessment. Some of these crimes, falling under category *F: missing information*, can be rectified by providing additional details in the system's specification. These might include comprehensive hardware and software specifications, along with other essential components of the benchmarking setup.

However, addressing other benchmarking crimes is a more intricate task, demanding careful consideration of the benchmarking techniques employed and the goals set for measurement. This complexity underscores the necessity for a comprehensive revision of the benchmarking process, wherein practitioners should approach this paper as a systematic checklist. By meticulously reviewing each aspect outlined in the paper, benchmarkers can ascertain that their methodologies are free from any committed benchmarking crimes.

This diligent approach not only safeguards the quality of the benchmark but also enhances the reliability and utility of the results. The effort invested in revisiting and refining the benchmarking process aligns with the overarching goal of fostering credible and ethical benchmarking practices within the research community.

REFERENCES

[1] Gernot Heiser. Systems benchmarking crimes.
[2] Erik van der Kouwe, Dennis Andriesse, Herbert Bos, Cristiano Giuffrida, and Gernot Heiser. Benchmarking crimes: An emerging threat in systems security. 2018.