How to Build a Benchmark

Jóakim v. Kistowski
University of Würzburg
joakim.kistowski@
uni-wuerzburg.de

Jeremy A. Arnold
IBM Corporation
arnoldje@us.ibm.com

Karl Huppler
karl.huppler@gmail.com

Klaus-Dieter Lange
Hewlett-Packard Company
klaus.lange@hp.com

John L. Henning
Oracle
john.henning@oracle.com

Paul Cao
Hewlett-Packard Company
paul.cao@hp.com

ABSTRACT

Standardized benchmarks have become widely accepted tools for the comparison of products and evaluation of methodologies. These benchmarks are created by consortia like SPEC and TPC under confidentiality agreements which provide little opportunity for outside observers to get a look at the processes and concerns that are prevalent in benchmark development. This paper introduces the primary concerns of benchmark development from the perspectives of SPEC and TPC committees. We provide a benchmark definition, outline the types of benchmarks, and explain the characteristics of a good benchmark. We focus on the characteristics important for a standardized benchmark, as created by the SPEC and TPC consortia. To this end, we specify the primary criteria to be employed for benchmark design and workload selection. We use multiple standardized benchmarks as examples to demonstrate how these criteria are ensured.

Categories and Subject Descriptors

C.4 [**Computer Systems Organization**]: Performance of Systems—*Performance attributes*

General Terms

Measurement, Performance, Standardization

Keywords

SPEC; TPC; SPECpower_ssj2008; SERT; SPEC CPU

1. INTRODUCTION

Standardized benchmarks have become widely accepted tools for the comparison of software and hardware products. They are also regularly used for the evaluation of methodological approaches to problems in multiple fields of computer science and beyond.

In order to be accepted for standardization, benchmarks must meet a host of quality criteria. Benchmarks candidates must undergo a process of several steps, including the definition of measurement methodologies, workload selection, and a number of rigorous benchmark acceptance tests. Benchmark inception, development, and acceptance, however, are conducted under consortia confidentiality agreements, with little opportunity for outside observers to profit from the processes that these consortia have developed over time. Being unable to access these processes, calls for benchmark development processes have become louder [6]. We address this situation by offering this paper as a description of the processes for benchmark inception, development, testing, release, and support for the SPEC and TPC consortia.

As part of this work, we provide a definition of the term "benchmark" in the context of performance evaluation. Note that we differentiate between benchmarks with the purpose of product comparison and rating tools, which are intended for standardized measurements as part of a product development or evaluation process. We explain the differences between the three types of benchmarks: specification-based, kit-based, and hybrid. We also present the properties and criteria that any quality benchmark or rating tool must fulfill. For this, we focus on the properties that workloads of quality benchmarks must meet. We present multiple examples of standardized benchmarks and demonstrate how these benchmarks ensure the specific quality criteria.

The remainder of this paper is structured as follows: Section 2 presents our benchmark and rating tool definitions, it also explains the three benchmark types. Section 3 presents the properties of good benchmarks. The paper concludes in Section 4.

2. WHAT IS A BENCHMARK

Before discussing the development process of a benchmark, we define what a benchmark is. We also explain the difference between major types of benchmarks.

2.1 Definition of Benchmark

We 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".

This definition is a variation of a definition provided in [12] with a focus on the competitive aspects of benchmarks, as that is the primary purpose of standardized benchmarks as developed by SPEC and TPC.

ICPE'15, Jan. 31–Feb. 4, 2015, Austin, Texas, USA.

ACM 978-1-4503-3248-4/15/01 ...$15.00.
http://dx.doi.org/10.1145/2668930.2688819.

We define tools for the non-competitive system evaluation and comparison as *rating tools*. Rating tools are primarily intended for a standardized method of evaluation for research purposes, regulatory programs, or as part of a system improvement and development approach. Rating tools can also be standardized and should generally follow the same design and quality criteria as benchmarks. SPEC's Server Efficiency Rating Tool (SERT), e.g., has been designed and developed using a similar process as the SPECpower_ssj2008 benchmark.

2.2 Types of Benchmarks

Computer benchmarks typically fall into three general categories: specification-based, kit-based, and a hybrid between the two. Specification-based benchmarks describe functions that must be achieved, required input parameters and expected outcomes. The implementation to achieve the specification is left to the individual running the benchmark. Kit-based benchmarks provide the implementation as a required part of official benchmark execution. Any functional differences between products that are allowed to be used for the benchmark must be resolved ahead of time and the individual running the benchmark is typically not allowed to alter the execution path of the benchmark.

Specification-based benchmarks begin with a definition of a business problem to be simulated by the benchmark. The key criteria for this definition are the relevance topics discussed in section 3 and novelty. Specification-based benchmarks have the advantage of allowing innovative software to address the business problem of the benchmark by proving the specified requirements of the new implementation [5]. On the other hand, specification-based benchmarks require substantial development prior to running the benchmark, and may have challenges proving that all the requirements of the benchmark are met.

Kit-based benchmarks may appear to restrict some innovative approaches to a business problem, but have the significant advantages of providing near "load and go" implementations that greatly reduce the cost and time required to run the benchmarks. For kit-based benchmarks, the "specification" is used as a design guide for the creation of the kit. For specification-based benchmarks, the specification is presented as a set of rules to be followed by a third party who will implement and run the benchmark. This allows for substantial flexibility in how the benchmark's business problem will be resolved - a principal advantage for specification-based benchmarks.

A hybrid of these may be necessary if the majority of the benchmark can be provided in a kit, but there is a desire to allow some functions to be implemented at the discretion of the individual running the benchmark.

While both specification-based and kit-based methods have both been successful in the past, current trends have favored kit-based development.

3. WORKLOAD PROPERTIES

Workload designers must balance several, often conflicting, criteria in order to be successful. Several factors must be taken into consideration, and trade-offs between various design choices will influence the strengths and weaknesses of the workload. Since no single workload can be strong in all of these areas, there will always be a need for multiple workloads and benchmarks [8].

It is important to understand the characteristics of a workload and determine whether or not it is applicable for a particular situation. When developing a new workload, the goals should be defined so that choices between competing design criteria can be made in accordance with those goals to achieve the desired balance. Several researchers and industry participants have listed various desirable characteristics of benchmarks [3, 4, 8, 1, 2, 11, 7]. The contents of the lists vary based on the perspective of the author and their choice of terminology and grouping of characteristics, but most of the concepts are similar. The key characteristics can be organized in the following groups, which will be discussed in more detail in the next sections:

- **Relevance** How closely the benchmark behavior correlates to behaviors that are of interest to consumers of the results
- **Reproducibility** The ability to consistently produce similar results when the benchmark is run with the same test configuration
- **Fairness** Allowing different test configurations to compete on their merits with-out artificial limitations
- **Verifiability** Providing confidence that a benchmark result is accurate
- **Usability** Avoiding roadblocks for users to run the benchmark in their test environments

All benchmarks are subject to these same criteria, but each category includes additional issues that are specific to the individual benchmark, depending on the benchmark's goals.

3.1 Relevance

"Relevance" is perhaps the most important characteristic of a benchmark. Even if the workload was perfect in every other regard, it will be of minimal use if it doesn't provide relevant information to its consumers. Yet relevance is also a characteristic of how the benchmark results are applied; benchmarks may be highly relevant for some scenarios and of minimal relevance for others. For the consumer of benchmark results, an assessment of a benchmark's relevance must be made in context of the planned use of those results. For the benchmark designer, relevance means determining the intended use of the benchmark and then designing the benchmark to be relevant for those areas [10]. A general assessment of the relevance of a benchmark or workload involves two dimensions: the breadth of its applicability, and the degree to which the workload is relevant in that area. For example, an XML parsing benchmark may be highly relevant as a measure of XML parsing performance, somewhat relevant as a measure of enterprise server application performance, and not at all relevant for graphics performance of 3D games. Conversely, a suite of CPU benchmarks such as SPEC CPU2006 may be moderately relevant for a wide range of computing environments. The behavior illustrated in these examples is generally true: benchmarks that are designed to be highly relevant in a specific area tend to have narrow applicability, while benchmarks that attempt to be applicable to a broader spectrum of uses tend to be less meaningful for any particular scenario [4].

Scalability is an important aspect of relevance, particularly for server benchmarks. Most relevant benchmarks are multi-process and/or multi-threaded in order to be able to take advantage of the full resources of the server [8]. Achieving scalability in any application is difficult; for a benchmark, the challenges are often even greater because the benchmark is expected to run on a wide variety of systems with significant differences in available resources. Benchmark designers must also strike a careful balance between avoiding artificial limits to scaling and behaving like real applications (which often have scalability issues of their own).

3.2 Reproducibility

Reproducibility is the capability of the benchmark to produce the same results consistently for a particular test environment. It includes both run-to-run consistency and the ability for another tester to independently reproduce the results on another system.

Ideally, a benchmark result is a function of the hardware and software configuration, so that the benchmark is a measure of the performance of that environment; if this were the case, the benchmark would have perfect consistency. In reality, the complexity inherent in a modern computer system introduces significant variability in the performance of an application. This variability is introduced by several factors, including things such as the timing of thread scheduling, dynamic compilation, physical disk layout, network contention, and user interaction with the system during the run [4]. Energy efficiency benchmarks often have additional sources of variability due to power management technologies dynamically making changes to system performance and temperature changes affecting power consumption.

Benchmarks can address this run-to-run variability by running for long enough periods of time to include representative samples of these variable behaviors. Some benchmarks require submission of multiple runs with scores that are near each other as evidence of consistency. Benchmarks also tend to run at steady state, unlike more typical applications which have variations in load due to factors such as the usage patterns of users.

The ability to reproduce results in another test environment is largely tied to the ability to build an equivalent environment. Industry standard benchmarks require results submissions to include a description of the test environment, typically including both hardware and software components as well as configuration options. Similarly, published research that includes benchmark results generally includes a description of the test environment that produced those results. However, in both of these cases, the description may not provide enough detail for an independent tester to be able to assemble an equivalent environment.

Hardware must be described in sufficient detail for another person to obtain identical hardware. Software versions must be stated so that it is possible to use the same versions when reproducing the result. Tuning and configuration options must be documented for firmware, operating system, and application software so that the same options can be used when re-running the test. TPC benchmarks require a certified auditor to audit results and ensure compliance with reporting requirements. SPEC uses a combination of automatic validation and committee review to establish compliance.

3.3 Fairness

Fairness ensures that systems can compete on their merits without artificial constraints. Because benchmarks always have some degree of artificiality, it is often necessary to place some constraints on test environments in order to avoid unrealistic configurations that take advantage of the simplistic nature of the benchmark.

Benchmark development requires compromises among multiple design goals; benchmarks developed by a consensus of experts is generally perceived as being more fair than a benchmark designed by a single company [1]. While "design by committee" may not be the most efficient way to develop an application, it does require that compromises are made in such a way that multiple interested parties are able to agree that the final benchmark is fair. As a result, benchmarks produced by organizations such as SPEC and the TPC (both of which are comprised by members from companies in the industry as well as academic institutions and other interested parties) are generally regarded as fair measures of performance.

Benchmarks require a variety of hardware and software components to provide an environment suitable for running the benchmark. It is often necessary to place restrictions on what components may be used. Careful attention must be placed on these restrictions to ensure that the benchmark remains fair. Some restrictions must be made for technical reasons. For example, a benchmark implemented in Java requires a Java Virtual Machine (JVM) and an operating system and hardware that supports it. A benchmark that performs heavy disk IO may effectively require a certain number of disks to achieve acceptable IO rates, which would therefore limit the benchmark to hardware capable of supporting that number of disks.

Benchmark run rules often require hardware and software to meet some level of support or availability. While this restricts what components may be used, it is actually intended to promote fairness. Because benchmarks are by nature simplified applications, it is often possible to use simplified software to run them; this software may be quite fast because it lacks features that may be required by real applications. For example, enterprise servers typically require certain security features in their software which may not be directly exercised by benchmark applications; software that omitted these features may run faster than software that includes them, but this simplified software may not be usable for the customer base that the benchmark is targeted to. Rules regarding software support can be a particular challenge when using open source software, which is often supported primarily by the developer community rather than commercial support mechanisms.

Both of these situations require a careful balance. Placing too many or inappropriate limits on the configuration may disallow results that are relevant to some legitimate situations. Placing too few restrictions can pollute the pool of published results and, in some cases, reduce the number of relevant results because vendors can't compete with the "inappropriate" submissions. Portability is an important aspect of fairness. Achieving portability with benchmarks written in Java is relatively simple; for C and C++, it can be more difficult [3].

Benchmark run rules often include stipulations on how results may be used. These requirements are intended to promote fairness when results are published and compared, and often include provisions that require certain basic information to be included any time that results are given. For example SPECpower_ssj2008 requires that if a comparison is made for the power consumption of two systems at the 50%

target load level, the performance of each system at the 50% load level as well as the overall ssj_ops/watt value must also be stated. SPEC has perhaps the most comprehensive fair use policy which further illustrates the types of fair use issues that benchmarks should consider when creating their run rules [9].

3.4 Verifiability

Within the industry, benchmarks are typically run by vendors who have a vested interest in the results. In academia, results are subjected to peer review and interesting results will be repeated and built upon by other researchers. In both cases, it is important that benchmark results are verifiable so that the results can be deemed trustworthy.

Good benchmarks perform some amount of self-validation to ensure that the workload is running as expected, and that run rules are being followed. For example, a workload might include configuration options intended to allow researchers to change the behavior of the workload, but standard benchmarks typically limit these options to some set of compliant values which can be verified at runtime. Benchmarks may also perform some functional verification that the output of the test is correct; these tests could detect some cases where optimizations (e.g. experimental compiler options) are producing incorrect results.

Verifiability is simplified when configuration options are controlled by the benchmark, or when these details can be read by the benchmark. In this case, the benchmark can include the details with the results. Configuration details that must be documented by the user are less trustworthy since they could have been entered incorrectly.

One way to improve verifiability is to include more details in the results than are strictly necessary to produce the benchmark's metrics. Inconsistencies in this data could raise questions about the validity of the data. For example, a benchmark with a throughput metric might include response time information in addition to the transaction counts and elapsed time.

3.5 Usability

Most users of benchmarks are technically sophisticated, making ease of use less of a concern than it is for more consumer-focused applications. There are, however, several reasons why ease of use is important. One of the most important ease of use features for a benchmark is self-validation. This was already discussed in terms of making the benchmark verifiable. Self-validating workloads give the tester confidence that the workload is running properly.

Another aspect of ease of use is being able to build practical configurations for running the benchmark. For example, the current top TPC-C result has a system under test with over 100 distinct servers, over 700 disk drives and 11,000 SSD ash modules (with a total capacity of 1.76 petabytes), and a system cost of over $30 million USD. Of the 18 non-historical accepted TPC-C results published between January 1, 2010 and August 24, 2013, the median total system cost was $776,627 USD. These configurations aren't economical for most potential users [4].

Accurate descriptions of the system hardware and software configuration are critical for reproducibility, but can be a challenge due to the complexity of these descriptions. Benchmarks can improve ease of use by providing tools to assist with this process.

4. CONCLUSIONS

This paper provides an insight into the benchmark development criteria as employed be the SPEC and TPC consortia. We provide a definition for benchmarks and rating tools, differentiating between benchmarks for competitive purposes and rating tools for research purposes, regulatory programs, or as part of a system improvement and development approach. We explain the differences between the three major types of benchmarks: specification-based, kit-based, and hybrid. Finally, we describe the major quality criteria of industrial benchmarks: relevancy, repeatability, fairness, verifiability, and usability, including examples on how the criteria are ensured in standardized benchmarks.

5. REFERENCES

[1] R. García-Castro and A. Gómez-Pérez. Benchmark Suites for Improving the RDF(S) Importers and Exporters of Ontology Development Tools. In Y. Sure and J. Domingue, editors, *The Semantic Web: Research and Applications*, volume 4011 of *Lecture Notes in Computer Science*, pages 155–169. Springer Berlin Heidelberg, 2006.

[2] J. Gustafson and Q. Snell. HINT: A new way to measure computer performance. In *System Sciences, 1995. Proceedings of the Twenty-Eighth Hawaii International Conference on*, volume 2, pages 392–401 vol.2, Jan 1995.

[3] J. L. Henning. SPEC CPU2000: measuring CPU performance in the New Millennium. *Computer*, 33(7):28–35, Jul 2000.

[4] K. Huppler. The Art of Building a Good Benchmark. In R. Nambiar and M. Poess, editors, *Performance Evaluation and Benchmarking*, volume 5895 of *Lecture Notes in Computer Science*, pages 18–30. Springer Berlin Heidelberg, 2009.

[5] K. Huppler and D. Johnson. TPC Express - A New Path for TPC Benchmarks. In R. Nambiar and M. Poess, editors, *Performance Characterization and Benchmarking*, volume 8391 of *Lecture Notes in Computer Science*, pages 48–60. Springer International Publishing, 2014.

[6] K. Sachs. *Performance Modeling and Benchmarking of Event-Based Systems*. PhD thesis, TU Darmstadt, 2010. SPEC Distinguished Dissertation Award 2011.

[7] S. E. Sim, S. Easterbrook, and R. C. Holt. Using Benchmarking to Advance Research: A Challenge to Software Engineering. In *Proceedings of the 25th International Conference on Software Engineering*, ICSE '03, pages 74–83, Washington, DC, USA, 2003. IEEE Computer Society.

[8] K. Skadron, M. Martonosi, D. I. August, M. D. Hill, D. J. Lilja, and V. S. Pai. Challenges in Computer Architecture Evaluation. *Computer*, 36(8):30–36, Aug. 2003.

[9] Standard Performance Evaluation Corporation. SPEC fair use rule. http://www.spec.org/fairuse.html.

[10] Standard Performance Evaluation Corporation. SPEC Power and Performance Benchmark Methodology. http://spec.org/power/docs/SPEC-Power_and_Performance_Methodology.pdf.

[11] F. Stefani, A. Moschitta, D. Macii, and D. Petri. FFT benchmarking for digital signal processing technologies. In *17th IMEKO World Congress*, 2003.

[12] M. Vieira, H. Madeira, K. Sachs, and S. Kounev. Resilience Benchmarking. In K. Wolter, A. Avritzer, M. Vieira, and A. van Moorsel, editors, *Resilience Assessment and Evaluation of Computing Systems*, XVIII. Springer-Verlag, Berlin, Heidelberg, 2012. ISBN: 978-3-642-29031-2.